Tetration Documentation

Release 3.5.1.17

Tetration Team

Apr 14, 2021

CONTENTS

CHAPTER

ONE

OVERVIEW

1.1 License Agreement

Supplemental End User License Agreement

IMPORTANT: READ CAREFULLY

Dear Customer,

This Supplemental End User License Agreement ("SEULA") contains additional terms and conditions for the Software product(s) set forth herein and licensed under the End User License Agreement ("EULA") between you and Cisco Systems, Inc. or its Affiliates (collectively, the "Agreement"). Please note that there may be terms in this SEULA that do not apply to you. Only those terms related to the specific Software product(s) you purchased apply to you. Except as otherwise set forth in this SEULA, capitalized terms will have the meanings as in the EULA. To the extent that there is a conflict between the EULA and this SEULA, this SEULA will take precedence.

By downloading, installing, or using the Software you agree to comply with the terms of this SEULA.

SUPPLEMENTAL LICENSE TERMS FOR: Data Center Software – Tetration

Table1. SOFTWARE ENTITLEMENT:

Product License	Metric	License Duration
Tetration	Server/VM	Term, Subscription
Tetration Cloud	Server/VM	Term, Subscription
Tetration Policy Enforcement (add-on feature)	Server/VM	Term, Subscription

Tetration APIs and Tetration Apps.

Additional terms applicable for Tetration APIs and Tetration Apps, available (beginning with version 2.0) in the following products: (i) Tetration (on premises), (ii) Tetration Cloud (public cloud); and (iii) Tetration add-ons. Your license to Tetration APIs and Tetration Apps provide additional functionality that are subject to these additional terms, which you agree to if you make use of the Tetration APIs or Tetration Apps. Cisco hereby grants to you a worldwide, non-exclusive, non-transferable, non-sublicensable license to use and make calls to the Tetration APIs and Tetration Apps for the sole purpose of developing and implementing software applications that work, communicate, or interact with Your licensed Tetration products. You agree not to assert any of your intellectual property developed with use of and/or used with the Tetration APIs or Tetration Apps against Cisco or any of its affiliates, customers, resellers, distributors, or other licensees of the Tetration APIs and Tetration Apps for making, having made, using, selling, offering for sale, or importing: (i) any products or services implementing, interfacing with or operating in combination with the Tetration APIs or Tetration Apps; or (ii) any applications developed using the Tetration APIs or Tetration

Apps. If you do not agree with the foregoing terms for Tetration APIs and Tetration Apps, do not make use of such functionality as you are not licensed to use the Tetration APIs or Tetration Apps.

Tetration Policy Enforcement (add-on feature).

The Tetration Policy Enforcement feature is an add-on capability that is licensed (on a per server/VM basis) separately from, and in addition to, the Tetration base license. Your license to the Tetration Policy Enforcement feature covers up to the number of servers/VMs for which you have licensed for that feature. If You have not licensed the Tetration Policy Enforcement feature, or if no server/VM quantity is specified for a Tetration Policy Enforcement feature license, Your license does not extend to that feature.

Tetration, Tetration Cloud and Tetration Policy Enforcement Subscriptions.

1. Your license is valid only for the term of the subscription you purchased.
2. If you selected in the ordering tool to be billed periodically by Cisco, you unconditionally agree to make the required payments at each interval for the entire term of the subscription (in accordance with the payment terms between you and Cisco), regardless of any termination or force majeure provisions in your purchase agreement with Cisco.
3. If you fail to make such payments:
   (a) you will be in breach of your purchase agreement with Cisco and Cisco will have the right to exercise all rights and remedies available to it under that agreement and at law or in equity;
   (b) the outstanding balance of your subscription will become immediately due and payable, and Cisco may exercise its rights to recover any and all unpaid subscription amounts and other amounts due and owing under your subscription; and
   (c) to the extent you are entitled to any kind of refund from Cisco, such refund will be applied to any outstanding amounts due under your subscription.

Threat Intelligence Automatic Update

Cisco provides You with Cisco-collated datasets necessary to detect malware, known vulnerabilities, known command and control channels and geolocation of IP addresses, etc. Cisco strongly recommends that you set up Tetration platform connectivity to Tetration Cloud so that these datasets can be automatically updated.

Cisco Tetration Usage Telemetry

Cisco may collect Telemetry Data related to Your use of Tetration platform in order to maintain, improve, or analyze the effectiveness of Tetration and related solutions. You acknowledge that Cisco may freely use this non-personal Telemetry Data that does not identify You or any of Your users. Some Telemetry Data that Cisco collects or that You provide or make accessible to Cisco as part of Your use of Tetration is necessary for the essential use and functionality of the solution. Telemetry Data is also used by Cisco to provide associated services such as technical support and to continually improve the operation, security efficacy and functionality. For those reasons, You may not be able to opt out from some of the Telemetry Data collection other than by uninstalling or disabling Tetration.

End User License Agreement:

Please also refer to the Cisco Systems, Inc. End User License Agreement at Cisco End User License Agreement and Cisco Tetration End User License Agreement for additional terms applicable to Tetration.

1.2 Cisco Tetration Overview

Today's datacenters consist of applications running in a hybrid multicloud environments that use bare-metal, virtualized, and container-based workloads. Key challenges that once faces is how to better secure applications and data

without compromising agility. The Cisco Tetration platform is designed to address this security challenge by providing comprehensive workload protection capability by bringing security closer to applications and tailoring the security posture based on the application behavior. Cisco Tetration achieves this by using advanced machine learning and behavior analysis techniques. This platform provides a ready-to-use solution to support the following security use cases in the datacenter:

- Allow list based micro-segmentation, that allows implementation of a zero-trust model
- Behavioral baselining, analysis, and identifying anomalies on the workloads
- Detection of common vulnerabilities and exposures associated with the software packages installed on the servers
- Based on user intent, proactively quarantining server(s) when vulnerabilities are detected and blocking communication
- Understand the datacenter security posture and where to focus in order to improve the overall datacenter security

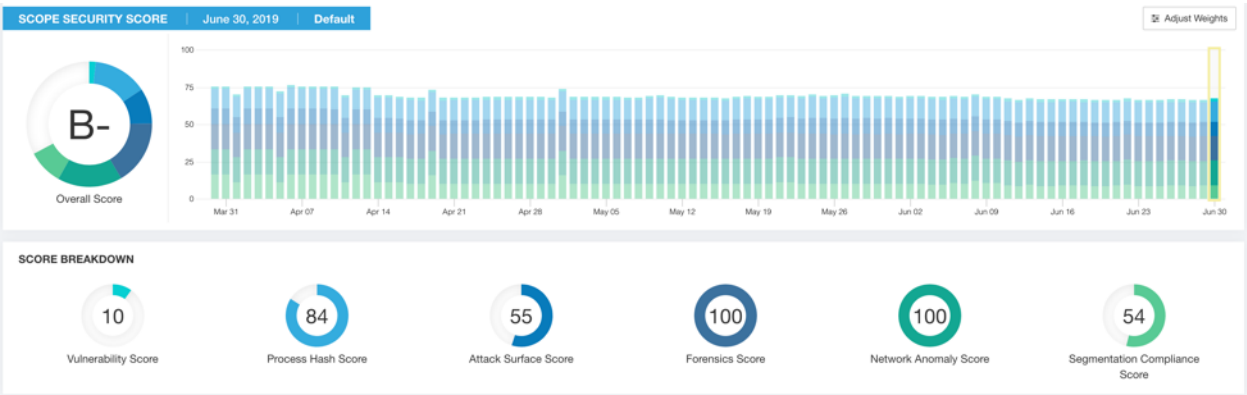


Fig. 1.2.1: Tetration Dashboard

Cisco Tetration user interface leverages an intuitive navigation menu to easily move between the main functions of the system including the security dashboard, workload profiles, application dependency maps, micro-segmentation policies, policy enforcement, process forensics, sensor management, etc. Each function page presents specific options for accessing and managing different components. See the table below for information on what each page provides.

1.2.1 Visibility

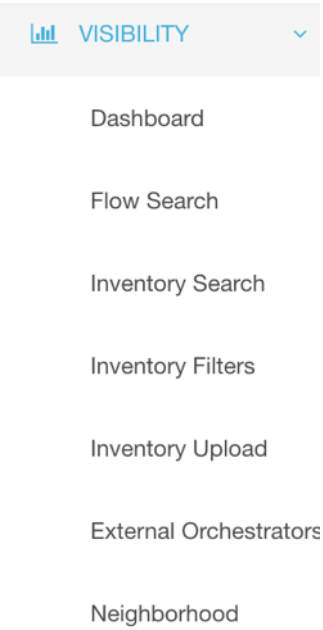


Fig. 1.2.1.1: Visibility Dropdown

Visibility menu option of the user interfaces provides the following capabilities:

Visibility Menu	Capabilities
Dashboard	Create custom dashboards and views
Flow Search	Capability to search and visualize network traffic flows
Inventory Search	Search specific workload inventory and view the profiles
Inventory Filters	Pre-define inventory filters for faster search and to use in segmentation policies
Inventory Upload	Upload additional context about the workloads through the user interface or REST API
External Orchestrators	Configure integrations with orchestrators such as VMware vCenter, Infobox, DNS servers, etc., to bring the
Neighborhood	Search for workload cluster or application neighbors, traffic profiles and logical hop information

1.2.2 Applications

Fig. 1.2.2.1: Applications Tab

Application menu enables a user to create application workspaces, define and generate segmentation policies based on application behavior, analyze the policies based historical or real time data and enforce the policies to implement a consistent segmentation.

1.2.3 Security

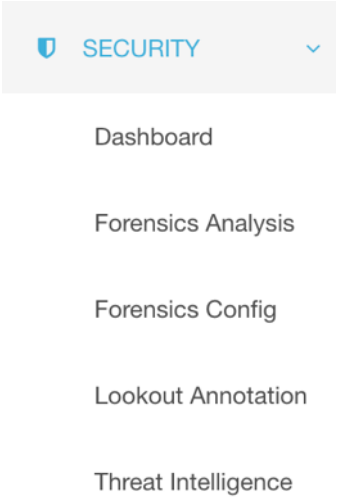


Fig. 1.2.3.1: Security Dropdown

Security menu option provides the following capabilities:

- Security dashboard to visualize and drill down on the datacenter and application security postures
- Configure and enable process behavior rules for an environment
- Track and visualize process behavior anomalies
- Configure external threat intelligence sources and updates

1.2.4 Data Platform

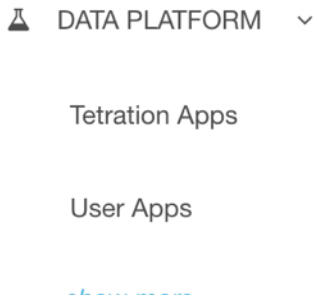


Fig. 1.2.4.1: Data Platform Dropdown

Data platform allows a user to enable any out-of-box Tetration apps or allows them to write their own apps. These apps can get access to the data in the HDFS datastore to analyze and define actions based on the results. In addition, this option also enables a user to view the status of the jobs, manage external API, etc.

1.2.5 Alerts

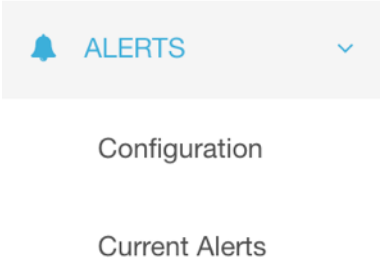
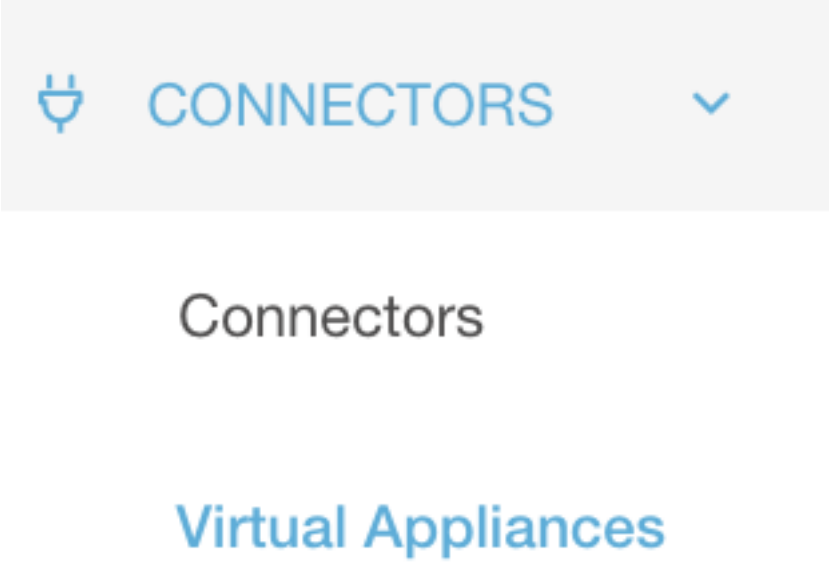


Fig. 1.2.5.1: Alerts Dropdown

Cisco Tetration supports sending alerts to other northbound systems using various mechanisms. Through the alerts menu, a user configures the specific alerts of interest and the delivery mechanism to receive alerts. One can also view the currents alerts from this menu option.

1.2.6 Connectors



Connectors are integrations that Tetration supports for a variety of use cases, including flow ingestion, inventory enrichment and alert notifications. Some connectors are agents that ingest flow observations to Tetration through standard protocols such as NetFlow v9 and IPFIX. Examples of such connectors are NetFlow, Citrix NetScaler, F5 BIG-IP, and AnyConnect. And, some connectors are alert notifiers. Examples of such connectors include Slack, Email, Syslog, PagerDuty and Kinesis. Each connector is enabled on one of three types of virtual appliances, namely: (1) *Tetration Ingest*, (2) *Tetration Edge*, and (3) *Tetration Export*.

1.2.7 Maintenance

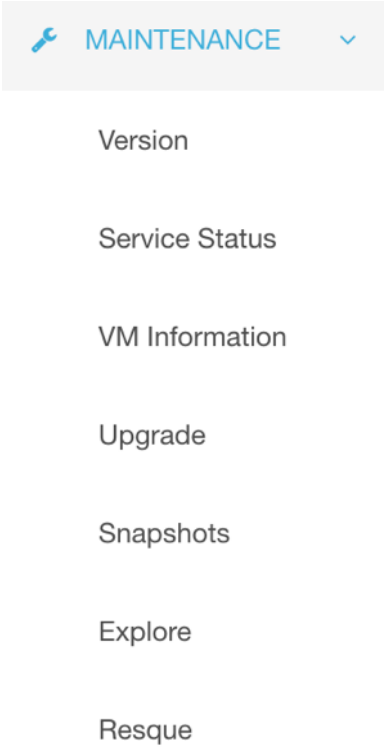


Fig. 1.2.7.1: Maintenance Dropdown

Maintenance menu option is specifically for administrative functions. Depending on the user privilege some of the options may not be visible. This menu option provides the following capabilities:

- Check the overall Tetration software service status and quickly identify if there are any specific service failures
- Status of about all the Tetration VMs in the cluster
- Initiate upgrades for applying patches or to upgrade to a major release
- Generate a full cluster snapshot for troubleshooting purposes
- Execute specific administrative comments based on instructions from Cisco support team

1.2.8 Monitoring

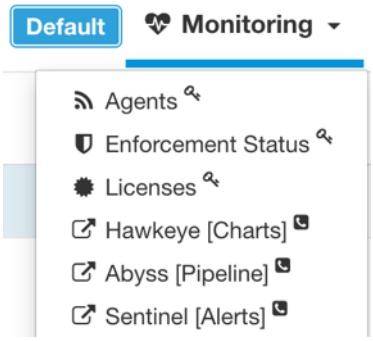


Fig. 1.2.8.1: Monitoring Dropdown

Monitoring menu option is also for administrative functions. Depending on the user privilege some of the functionalities may not be visible. Monitoring menu option provides the following capabilities:

- General agent status, such as deep visibility sensors, universal sensors, ERSPAN sensors, Anyconnect proxy and endpoints, etc.
- Number of enforcement agents and its status
- License registration and usage information
- Other platform functions and load

CHAPTER

TWO

DASHBOARD

The Dashboard feature enables users to visualize data generated from both Tetration sources as well as user-created application sources. We provide pre-populated dashboards to showcase some of the most interesting aspects of Tetration data. There are two types of pre-populated dashboards: The first type is **Flows** Dashboard that includes overall statistics. The second type of dashboards are user-customizable. Both types of dashboards are based on data filtered using the selected scope preference in the header menu.

The quickest way to begin constructing custom charts is to clone the provided **Sample Dashboard** and begin customizing.

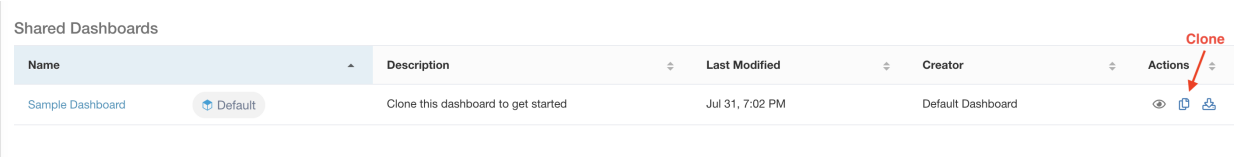


Fig. 2.1: Clone dashboard

A new dashboard will be created with edit capability. Edit Dashboard popup menu, which looks similar to the create popup menu, allows you to customize the name, description and share options.

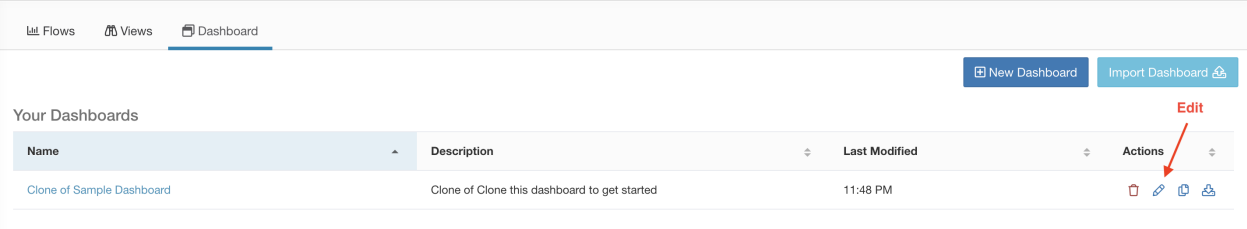


Fig. 2.2: Edit cloned dashboard

The creator of a new dashboard can share the dashboard to any scope to which the creator has access. This sharing allows anyone who has read access to the recipient scope and all of its child scopes to see the dashboard and all of the chart views inside the shared dashboard.

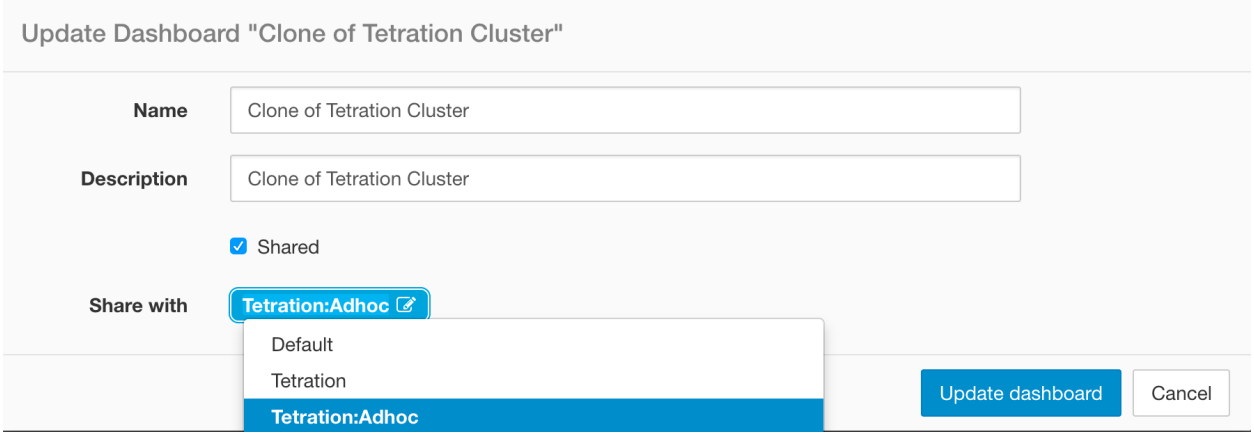


Fig. 2.3: Share dashboard

Once a dashboard is shared, it is indicated with the scope label next to the dashboard name.

Fig. 2.4: Dashboard list

Note: Some chart views under a shared dashboard may not show correctly because the viewer doesn't have access to the data source used in the chart view. When that happens, contact your scope administrator and request access to the scope to which the data source is shared.

To begin editing the dashboard, select the name of the dashboard. As shown below each chart view can be independently resized and repositioned anywhere on the dashboard. The dashboard data is fetched within the time range specified by the time picker as well as the scope preference chosen in the header menu.

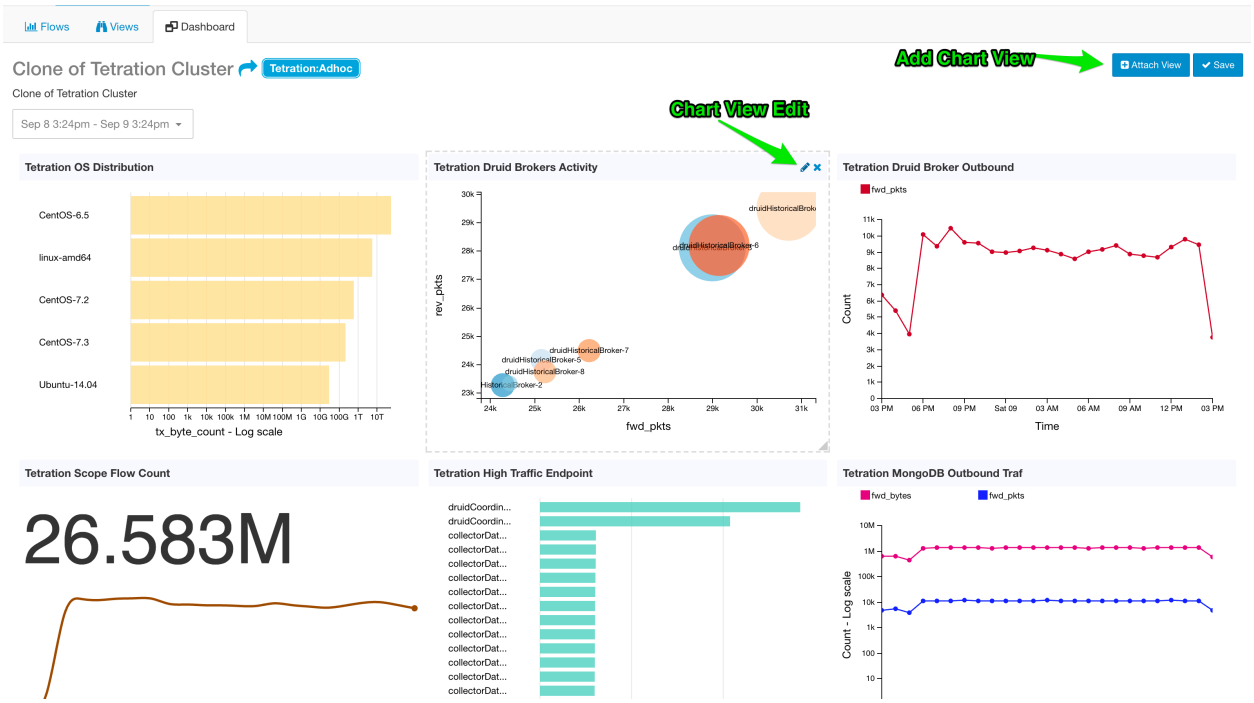


Fig. 2.5: Edit dashboard

Each view can be customized to use any VDS to render any chart type. The following shows an example of the Bubble Chart rendering based on the Aggregated Flows VDS.

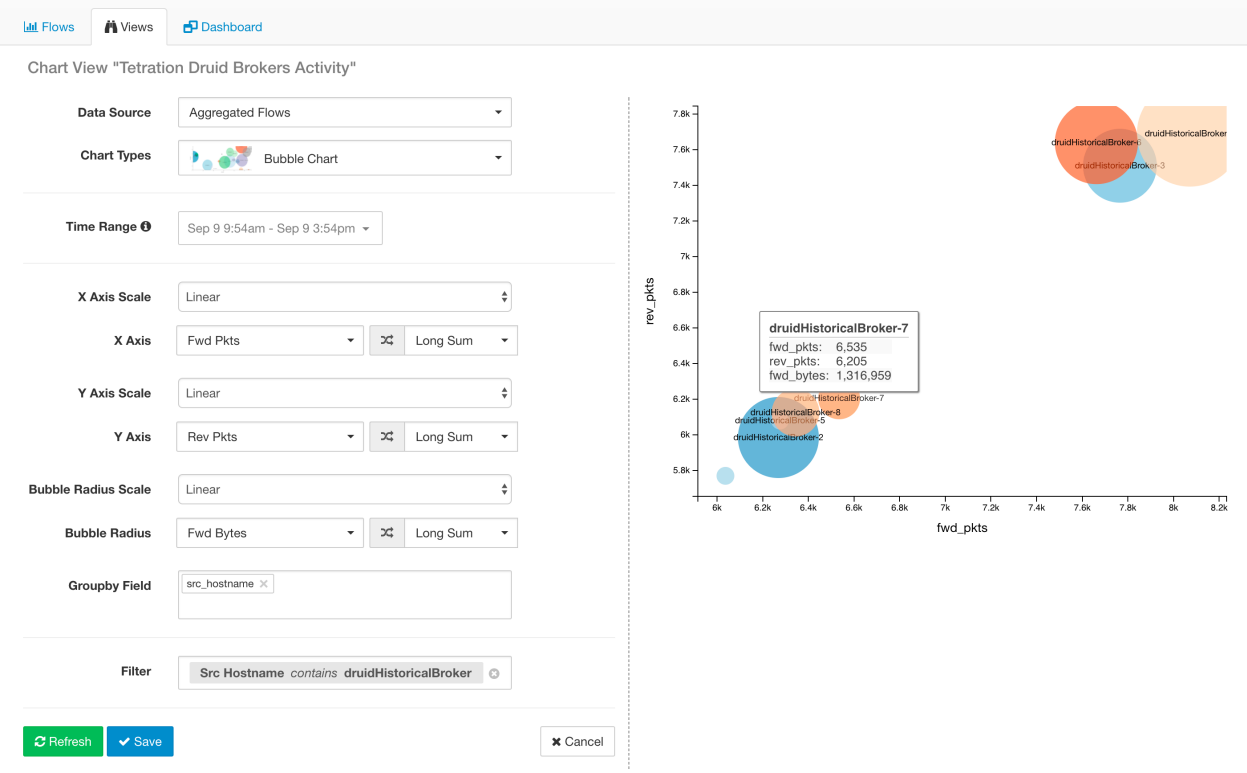


Fig. 2.6: Bubble Chart rendering based on the Aggregated Flows VDS

The number of Data Source available for a particular user depends on the user's role. Everyone will be able to use Tetration VDS's but **Shared** and **Private** VDS types will be selectively visible to the authorized users. Users with Lab Admin roles are authorized to change VDS visibility settings as described in *Visualization Data Sources*. This customized view can be attached in another dashboard.

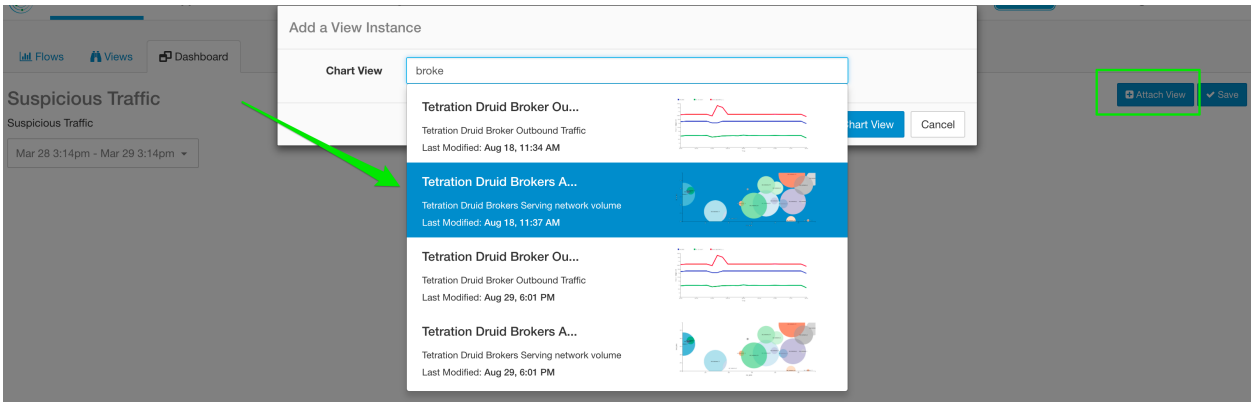


Fig. 2.7: Data sources

Charts can be filtered by the following dimensions:

Available Columns and Filters:

Columns	Description
Address Type	Filter flow observations by Address type (IPv4, IPv6, DHCPv4).
Flow Start Time	Filter flows by the Start time.
Protocol	Filter flow observations by Protocol type (TCP, UDP, ICMP).
Bandwidth Bytes Per Second	Filter flows by the bandwidth Bytes per second.
Address Type	Filter flow observations by Address type (IPv4, IPv6, DHCPv4).
Consumer Address (*src_address*)	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24). Matches flow observations whose consumer address overlaps with provided IP Address or subnet.
Consumer Name (*src_hostname*)	Matches flows whose consumer name overlaps with provided name.
Consumer Port (*src_port*)	Matches flows whose Consumer port overlaps with provided port.
Consumer Process ID	Matches flows by Consumers Process ID.
Consumer Scope	Matches flows whose consumer belongs to the specified Scope.
Consumer UUID	Matches flows whose consumer belongs to the specified UUID
Dst Epg Id	The Provider Enforcement ID is the ID of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the provider.
Dst Is Internal	Match only internal Providers.
Dst Scope Id	Matches flows whose provider belongs to the specified Scope.
Provider Address (*dst_address*)	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24) Matches flow observations whose provider address overlaps with provided ip address or subnet.
Provider Name (*dst_hostname*)	Matches flows whose provider hostname overlaps with provided hostname.
Provider Port (*dst_port*)	Matches flows whose Provider port overlaps with provided port.
Provider Scope	Matches flows whose provider belongs to the specified Scope.
Provider UUID	Matches flows whose provider belongs to the specified UUID
Src Epg Id	The Consumer Enforcement ID is the ID of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the provider.
Src Is Internal	Match only internal Consumers.
Src Scope Id	Matches flows whose Consumer belongs to the specified Scope.
SRTT Available	Matches flows which have SRTT measurements available using the values 'true' or 'false'. (This is equivalent to SRTT > 0).
Src Is Internal	Match only internal Consumers.
VRF ID	Match flows for the specific VRF.

Navigating to the preferences page via the cog icon in the top right of the screen, you can set your preference for which page you want to see when you first login. Out of the box, Security Dashboard is the set as the initial page.

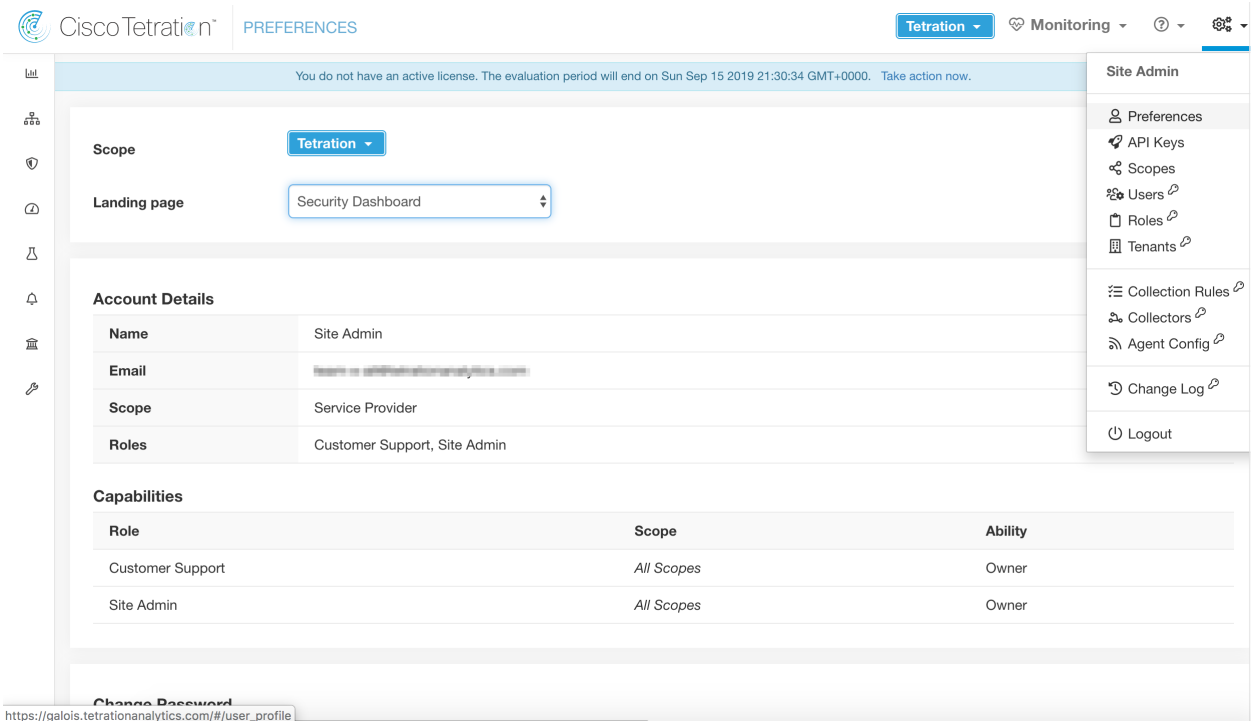


Fig. 2.8: Landing Page preferences

2.1 Note to Site Admin Users

Starting with release 3.4, Tetration data sources are tenant separated and hence for every Tetration data source before the upgrade, there will be as many Tetration data sources as there are tenants. They can be distinguished by their association with the tenant. If a site admin user has created any views with Tetration data sources before the upgrade, those views need to be updated with the appropriate tenant specific Tetration data source. When sharing a dashboard, Site admin users should ensure that the data sources behind charts in the dashboard are accessible to users of the tenant to which the dashboard is shared.

CHAPTER

THREE

SOFTWARE AGENTS

Tetration software agent is a piece of software running within a host operation system (such as Linux or Windows). Its core functionality is to monitor and collect network flow information. It also collects other host information such as network interfaces and active processes running in the system. The information collected by the agent is exported to a set of collectors running within Tetration cluster for further analytical processing. In addition, software agents also have capability to set firewall rules on the installed hosts.

In this user guide we will describe main functionalities of software agents, their communication interfaces, how to deploy, upgrade, configure and troubleshoot agent problems during/after deployment.

3.1 Deploying Software Agents

3.1.1 Supported platforms

Currently the following platforms are supported to run software agents:

1. Deep visibility agents
    1. Linux family: 64-bit variants
    - RHEL: 6.[0-10], 7.[0-9], 8.[0-3] (only x86_64 architecture for version 6.x)
    - CentOS: 6.[0-10], 7.[0-9], 8.[0-3] (only x86_64 architecture)
    - Oracle Linux: 6.[0-10], 7.[0-9], 8.[0-3] (only x86_64 architecture)
    - Ubuntu: 14.04, 16.04, 18.04, 20.04 (only x86_64 architecture)
    - SUSE Linux Enterprise Server: 11sp[2-4], 12sp[0-5], 15sp[0-2] (x86_64 architecture)
    - SUSE Linux Enterprise Server: 11sp4, 12sp[4-5], 15sp[0-2] (s390x architecture)
    - Amazon Linux 2 (only x86_64 architecture)
    2. Windows family: 64-bit variants
    - Windows Desktop: 8, 8.1, 10
    - Windows Server: 2008 R2, 2012, 2012 R2, 2016, 2019
    3. AIX family: ppc 64-bit variants
    - AIX: 7.1, 7.2 (no forensics support)

    Note: Process tree, Package (CVE), and Forensic Event reporting features are not yet available on AIX. Additionally, some aspects of those features may not be available on specific minor releases of otherwise-supported platforms due to OS limitations.

2. Enforcement agents
   1. Linux family: 64-bit variants
   - RHEL: 6.[2-10], 7.[0-9], 8.[0-3] (only x86_64 architecture for version 6.x)
   - CentOS: 6.[2-10], 7.[0-9], 8.[0-3] (only x86_64 architecture)
   - Oracle Linux: 6.[2-10], 7.[0-9], 8.[0-3] (only x86_64 architecture)
   - Ubuntu: 14.04, 16.04, 18.04, 20.04 (only x86_64 architecture)
   - SUSE Linux Enterprise Server: 11sp[2-4], 12sp[0-5], 15sp[0-2] (x86_64 architecture)
   - SUSE Linux Enterprise Server: 11sp4, 12sp[4-5], 15sp[0-2] (s390x architecture)
   - Amazon Linux 2 (only x86_64 architecture)
   2. Windows family: 64-bit variants
   - Windows Desktop: 8, 8.1, 10
   - Windows Server: 2008 R2, 2012, 2012 R2, 2016, 2019
   3. AIX family: ppc 64-bit variants
   - AIX: 7.1, 7.2
   4. Container orchestration systems
      - Kubernetes 1.[12-18]
      - OpenShift 3.[5-11]
      - OpenShift 4.[1-6]

      These plaforms are supported when using enforcement agents on RHEL 7.[0-9], CentOS 7.[0-8] or Ubuntu 16.04/18.04/20.04 nodes. For OpenShift, using CoreOS on worker nodes is not supported. IPVS based kube-proxy mode is not supported for OpenShift. These agents should be configured with the Preserve Rules option enabled. See *Creating an Agent Config Profile*.

      For enforcement to function properly, any installed CNI plugin must:

      1. Provide a flat IP network between all nodes and pods
      2. Not interfere with Linux iptables rules or marks used by the Tetration Enforcement Agent (mark bits 21 and 20 are used to allow and deny traffic for NodePort services)

      The following CNI plugins have been tested to meet the requirements above:

      - Calico (3.13) with the following Felix configurations: *(ChainInsertMode: Append, IptablesRefreshInterval: 0)* or *(ChainInsertMode: Insert, IptablesFilterAllowAction: Return, IptablesMangleAllowAction: Return, IptablesRefreshInterval: 0)*. All other options use their default values. Please refer to the Felix configuration reference for more information on setting these options.
   5. Kubernetes 1.[16-18]
   - RHEL: 7.[0-9] (only x86_64 architecture)
   - CentOS: 7.[0-8] (only x86_64 architecture)
   - Oracle Linux: 7.[0-8] (only x86_64 architecture)
   - Ubuntu: 16.04, 18.04, 20.04 (only x86_64 architecture)
   - SUSE Linux Enterprise Server: 12sp[0-5] (only x86_64 architecture)
   - Amazon Linux 2 (only x86_64 architecture)

3. AnyConnect agents

   Platforms supported by Cisco AnyConnect Secure Mobility agent with Network Visibility Module. No additional Tetration agent is required. AnyConnect connector registers these agents and exports flow observations, inventories, and labels to Tetration. For more information, please refer to *AnyConnect Connector*.

   For Windows, Mac, or Linux platforms, please refer to Cisco AnyConnect Secure Mobility Client Data Sheet

4. Universal agents

- Linux: x86 (32-bit) and amd64 (64-bit)
- Windows: Winidows Server 2003 (Standard/Enterprise), Winidows Server 2003 R2 (Standard/Enterprise), and Windows Vista (Business/Enterprise)
- Solaris: amd64 (64-bit) version 11.x (no SPARC support yet)
- AIX: PPC version 5.x and 6.x. Note that the Universal AIX agent is not supported for SaaS clusters.

5. SPAN agents

- CentOS-7.3 (64-bit)

6. Secure Connector Client

- RHEL 7 x86_64
- CentOS 7 x86_64

7. ISE agents.

   Endpoints registered with Cisco Identity Services Engine (ISE). No Tetration agent on the endpoint is required. ISE connector collects metadata about endpoints from ISE through pxGrid service on ISE appliance. It registers the endpoints as ISE agents on Tetration and pushes labels for the inventories on these endpoints. For more information, please refer to *ISE Connector*.

Note: Appliance agents such as NetFlow, NetScaler, F5, AWS and AnyConnect Proxy are now supported as connectors. For more information on connectors, please refer to *What are Connectors*.

3.1.2 Requirements

3.1.2.1 Linux (deep visibility agents)

Root privileges are required to install and execute the services.

The following dependencies must be met:

- For RHEL/CentOS/Ubuntu/Oracle minimum kernel version must be **2.6.32-71** (RHEL 6.0 base version). For SLES, minimum kernel version must be **3.0.13-0.27.1** (11Sp2 base version) For Amazon Linux, minimum validated kernel version is **4.14.72**
- curl: version 7.15 or later
- dmidecode: version 2.11 or later (except s390x)
- openssl: it is recommended to upgrade openssl to the latest version supported by the Linux distributor (such as RedHat or Oracle)
- sed

- awk
- flock
- lsof
- which
- rpm

In addition to the above dependencies, if agent is installed via installer script, the following are required:

- unzip

Note that a special user, **tet-sensor**, will be created in the host where the agent is installed. If PAM/SELinux is configured in the host, then **tet-sensor** user needs to be granted appropriate privileges such as executing tet-sensor process and making connections to collectors. For container orchestration systems (Kubernetes and OpenShift), a flat address space must be used for nodes and pods. Network plugins which masquerade the source pod IP for intra-cluster communication are not supported. If alternative install directory is provided and SELinux is configured, make sure the execution is allowed for those locations.

3.1.2.2 Linux/Solaris/AIX (universal agents)

Root privileges are required to install and run the cronjobs.

The following dependencies are required:

- lsof
- ps
- whoami
- which
- shell: default shell available in the platform, sh/ksh/bash are supported

Note that the available package version for universal agents to download is not necessarily the same the cluster is running. This is especially true after the cluster has been upgraded to a newer version.

3.1.2.3 Linux (enforcement agents)

Requirements:

- For RHEL/CentOS/Oracle/Ubuntu, minimum kernel version must be **2.6.32-220** (RHEL 6.2 base version).
- For SLES, minimum kernel version must be **3.0.101**
- For Amazon Linux, minimum validated kernel version is **4.14.72**
- All the dependencies required by the deep visibility agents
- ipset (minimum version 6.11-4) and the corresponding kernel modules
- iptables/ip6tables (minimum version 1.4.6-2.11.4 for SLES | 1.4.7-16 for RHEL/CentOS/Ubuntu/Oracle and must support ipset match syntax) and the corresponding kernel modules
- All other firewall applications (such as firewalld) be disabled

3.1.2.4 Linux (Secure Connector)

Requirements:

- RHEL/CentOS 7 (x86_64)
- 2 CPU cores and 4 GB RAM
- Enough network bandwidth for handling data from the on-prem orchestrators that will use the Secure Connector
- Outgoing connectivity to the Tetration cluster on port 443 (direct or through HTTP(S) proxy)
- Outgoing connectivity to internal Orchestrator API servers (direct)

3.1.2.5 Windows (deep visibility)

Requirements:

- Administrator privileges (both install and service execution)
- Npcap needs to be installed. For Windows OSs, other than Win2008 R2, the recommended NPCAP version is 0.9990. For Win2008 R2, the recommended version is 0.991. If the Npcap driver is not already installed, the recommended NPCAP version will be installed silently by Agent installer
- Powershell version 4.0 or later is required if agent is installed via installer script
- Update to the latest service packs available for the Windows platforms (provided by Microsoft)

3.1.2.6 Windows (enforcement agents)

Requirements:

- All the dependencies required by the deep visibility agents

Additional requirements when enforcement mode is WAF: - Windows Firewall with Advanced Security enabled - No active third-party firewall software - With Group Policy (GPO) enabled, turn ON Firewall state for **All active profiles** with Inbound/Outbound connections set to "Not configured". Also, the GPO must never configure any concrete firewall rules. - Without Group Policy (GPO) enabled, Advanced Firewall Settings must have **All active profiles** Firewall state turned ON.

No other requirements are needed when enforcement mode is WFP.

3.1.2.7 Windows (universal agents)

Requirements:

- tasklist
- netstat

3.1.2.8 AIX (deep visibility)

Root privileges are required to install and execute the services.

The following dependencies must be met:

- openssl: openssl to be version **1.0.1e** or newer for AIX 7.1 and version **1.0.2h** or newer for AIX 7.2
- ksh

- sed
- awk

Notes:

AIX only supports flow capture of 20 net devices (6 if version is AIX 7.1 TL3 SP4 or older). The deep visibility agent captures from at most 16 network devices, leaving the other 4 capture sessions available for exclusive generic system usage (e.g. tcpdump).

The deep visibilty agent does the following to ensure this behaviour

- The agent creates 16 bpf device nodes under the agents directory (/opt/cisco/tetration/chroot/dev/bpf0 - /opt/cisco/tetration/chroot/dev/bpf15)
- tcpdump and other system tools using bpf will scan thru the system device nodes (/dev/bpf0-/dev/bpf19) until they find an unused node (!EBUSY)
- The agent created bpf nodes and system bpf nodes will share the same major/minor, with each major/minor only be opened by one instance (either tcpdump or agent)
- The agent will not access the system device nodes, and not create them as tcpdump does (tcpdump -D will create /dev/bpf0. . . /dev/bpf19 if they do not exist)

Running iptrace on system will prevent in certain scenarios flow capture from tcpdump and deep visibilty agent. This is known design deficiency, please check with IBM.

- To check if this scenarios exist prior to installation of deep visibilty agent, run tcpdump. If error message is like **tcpdump: BIOCSETIF: en0: File exists** iptrace is blocking flow capture. Stopping iptrace will resolve the issue.

Not every deep visibility functionality is supported on AIX. Package and process accounting is among the ones not supported.

3.1.2.9 AIX (enforcement agents)

Requirements:

- All the dependencies required by the AIX deep visibility agents
- IPFilter installed (ipfl.rte version **5.3.0.7**)
- AIX native firewall or any third party firewall should not be running on the agents
- Do not use native AIX firewall commands (genfilt, chfilt, rmfilt, mkfilt, expfilt, impfilt). If IP Security Filter is enabled (i.e. smitty ipsec4), agent installation fails in pre-check. It is recommended to **disable IP Security Filter** before installing agent. When IP security is enabled, while Tetration enforcer agent is running, it will be reported as an error and enforcement agent will stop enforcing. To safely disable IP Security Filter while enforcement agent is running, please contact support.

3.1.3 Storage needed

- For AIX and IBM Z Platforms, Tetration agents require 500MB of storage.
- For other platforms, agents require 1GB of storage.
- The log files are typically stored inside the <install-location>/logs or <install-location>/log folder. These log files are monitored and rotated by the Tetration services.

3.1.3.1 Kubernetes Deployments

- In addition to the Linux agent requirements, Kubernetes deployments require the target Kubernetes cluster to be running Kubernetes 1.[16-18].
- The install script requires Kubernetes admin credentials to start privileged agent pods on the cluster nodes.
- The Tetration application entities will be created in a namespace named 'tetration'.
- The node/pod security policies should permit privileged mode pods.
- busybox:1.33 images should either be pre-installated or downloadable from Docker Hub.
- In order to run on Kubernetes control plane nodes, the –toleration flag can be used to pass in a toleration for the Tetration pods. This usually is the NoSchedule toleration that normally prevents pods from running on control plane nodes.

3.1.4 Deploying a Deep Visibility/Enforcement Linux Agent

3.1.4.1 Install the agent

There are two methods for installing deep visibility or enforcement agent as described below.

Using installer script

This is the recommended method to deploy deep visibility or enforcement agents on Linux platforms. The process is as follows:

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Installer** tab. The **Installer** tab opens.
4. Select **Auto-Install using Installers** workflow and then click **Next**.
5. In the step **Select Type**, choose either Deep Visibility or Enforcement and then click **Next**.
6. In the step **Download**, choose tenant that agents will be installed under. Note that in Tetration Saas cluster, no tenant selection is required.
7. In the step **Download**, choose Linux as platform.
8. In the step **Download**, enter http proxy url if needed.
9. Click **Download Installer** button and save the file to local disk.
10. Copy the installer shell script to all the Linux hosts for deployment.
11. Run command chmod u+x tetration_installer_default_sensor_linux.sh to grant execute permission for the script. (note: the script name may differ depending on agent type and scope)
12. Run command ./tetration_installer_default_sensor_linux.sh with root privilege to install agent.

Note that the script **will not proceed if agent has already been installed**.

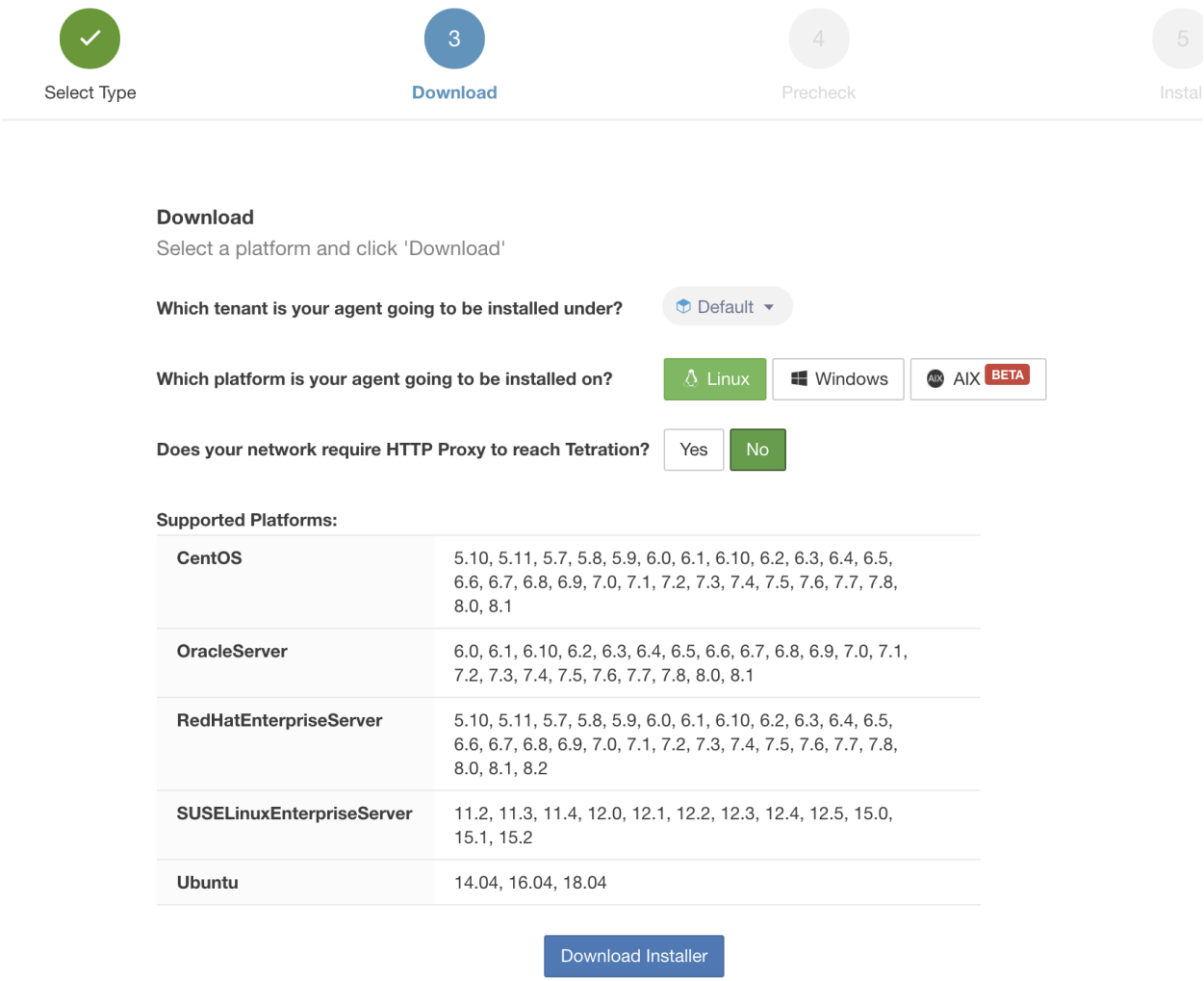


Fig. 3.1.4.1.1: Software Agent Installer Script Download Page (On-prem)

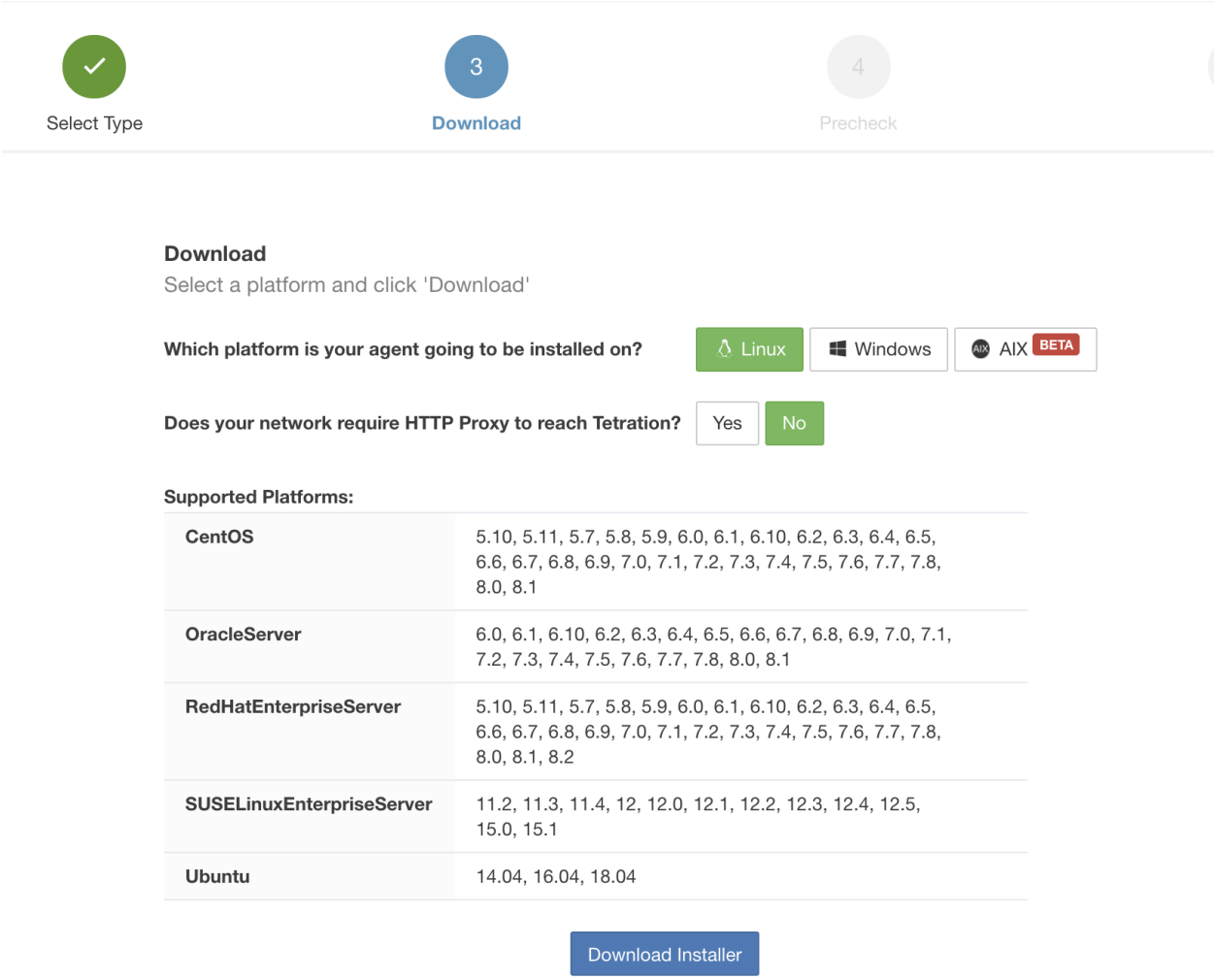


Fig. 3.1.4.1.2: Software Agent Installer Script Download Page (Saas)

The usage of this installer script is as follows:

$ bash tetration_linux_installer.sh [–pre-check] [–skip-pre-check=<option>] [–no-install] [–logfile=<filename>] [–proxy=<proxy_string>] [–no-proxy] [–help] [–version] [–sensor-version=<version_info>] [–ls] [–file=<filename>] [–save=<filename>] [–new] [–reinstall] [–unpriv-user] [–force-upgrade] [–upgrade-local] [–upgrade-by-uuid=<filename>] [–basedir=<basedir>] [–logbasedir=<logbdir>] [–visibility]

–pre-check: run pre-check only

–skip-pre-check=<option>: "skip pre-installation check by given option; Valid options include 'all', 'ipv6' and 'enforcement'; e.g.: '–skip-pre-check=all' will skip all pre-installation checks; All pre-checks will be performed by default

–no-install: will not download and install sensor package onto the system

–logfile <filename>: write the log to the file specified by <filename>

–proxy <proxy_string>: set the value of HTTPS_PROXY. Use this if proxy is needed to communicate with the cluster. The string should be formatted as http://<proxy>:<port>

–no-proxy: bypass system wide proxy; this flag will be ignored if –proxy flag was provided

–help: print this help

–version: print current script's version

–sensor-version <version_info>: select sensor's version; e.g.: '–sensor-version=3.4.1.0'; will download the latest version by default if this flag was not provided

–ls: list all available sensor versions for your system (will not list pre-3.1 packages); will not download any package

–file <filename>: provide local zip file to install sensor instead of downloading it from cluster

–save <filename>: download and save zip file as <filename>

–new: remove any previous installed sensor; previous sensor identity has to be removed from cluster in order for the new registration to succeed

–reinstall: reinstall sensor and retain the same identity with cluster; this flag has higher priority than –new

–unpriv-user=<username>: use <username> for unpriv processes instead of tetsensor

–force-upgrade: force sensor upgrade to version given by –sensor-version flag; e.g.: '–sensor-version=3.4.1.0 –force-upgrade'; apply the latest version by default if –sensor-version flag was not provided

–upgrade-local: trigger local sensor upgrade to version given by –sensor-version flag: e.g.: '–sensor-version=3.4.1.0 –upgrade-local'; apply the latest version by default if –sensor-version flag was not provided

–upgrade-by-uuid=<filename>: trigger sensor whose uuid is listed in <filename> upgrade to version given by –sensor-version flag; e.g.: '–sensor-version=3.4.1.0 –upgrade-by-uuid=/usr/local/tet/sensor_id'; apply the latest version by default if –sensor-version flag was not provided

–basedir=<base_dir>: instead of using /usr/local use <base_dir> to install agent. The full path will be <base_dir>/tetration

–logbasedir=<log_base_dir>: instead of logging to /usr/local/tet/log use <log_base_dir>. The full path will be <log_base_dir>/tetration

–visibility: install deep visibility agent only; –reinstall would overwrite this flag if previous installed agent type was enforcer

Notes:

- Ubuntu is now using the native .deb package, new installs and reinstalls will switch to this package type, upgrades from previous version will stay with rpm package.
- Ubuntu .deb package is installed under /opt/cisco/tetration.
- Due to lack of relocation support of .deb package the –basedir option is not supported for Ubuntu.

Using the downloaded bundle

This section explains how to download an agent image and install it onto the Linux hosts. This is the legacy method of deployment and is not recommended.

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Installer** tab. The **Installer** tab opens.
4. Select **Manual Install using classic packaged installers** workflow and then click **Next**.
5. Find the appropriate version/platform/architecture/agent type and click **Download** button.
6. Copy the rpm package to all the Linux hosts for deployment, and execute the rpm command with root privilege.

Note that if **the agent has already been installed, please do not reinstall**. If agent needs to be upgraded to a new version, please use follow the upgrade process described in *Upgrading Software Agents*.

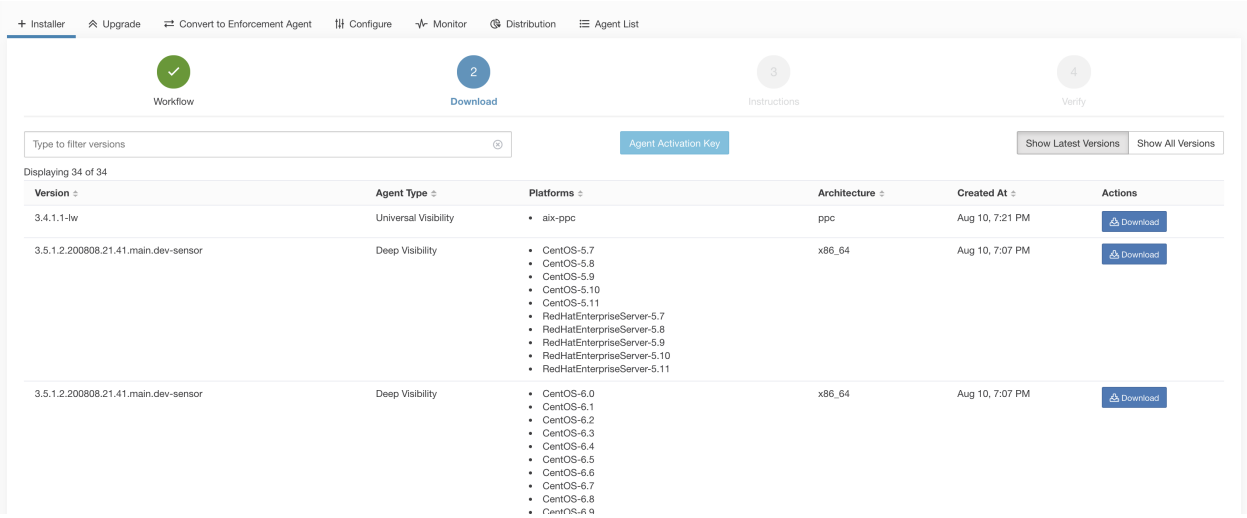


Fig. 3.1.4.1.3: Software Agent Bundle Download Page

For RHEL/CentOS/Oracle platforms:

1. Run command **rpm -ivh <rpm_filename>**

For Ubuntu platform:

1. First run command **rpm -qpR <rpm_filename>** to get the dependency list and make sure all dependencies are met.
2. Then install with "–nodeps" option: **rpm -ivh –nodeps <rpm filename>**

3.1.4.2 Verify that the agent is installed

1. Run command **sudo rpm -q tet-sensor**, confirm that there is one entry as follows (note: the specific output may differ depending on the platform and architecture):

$ sudo rpm -q tet-sensor

tet-sensor-3.1.1.50-1.el6.x86_64

3.1.5 Deploying a Deep Visibility/Enforcement Windows Agent

3.1.5.1 Install the agent

Similar to Linux platforms, there are two methods to install deep visibility or enforcement agent on Windows platforms.

Using installer script

This is the recommended method to deploy deep visibility or enforcement agents on Windows platforms (Make sure Powershell is 4.0 or newer). The process is as follows:

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Installer** tab. The **Installer** tab opens.
4. Select **Auto-Install using Installers** workflow and then click **Next**.
5. In the step **Select Type**, choose either Deep Visibility or Enforcement and then click **Next**.
6. In the step **Download**, choose tenant that agents will be installed under. Note that in Tetration Saas cluster, no tenant selection is required.
7. In the step **Download**, choose Windows as platform.
8. In the step **Download**, enter http proxy url if needed.
9. Click **Download Installer** button and save the file to local disk.
10. Copy the installer Powershell script to all the Windows hosts for deployment, and execute the script with Administrator privilege. Note that depending on the system settings, the command **Unblock-File** might needs to be executed first.

Note that the script **will not proceed if agent has already been installed**.

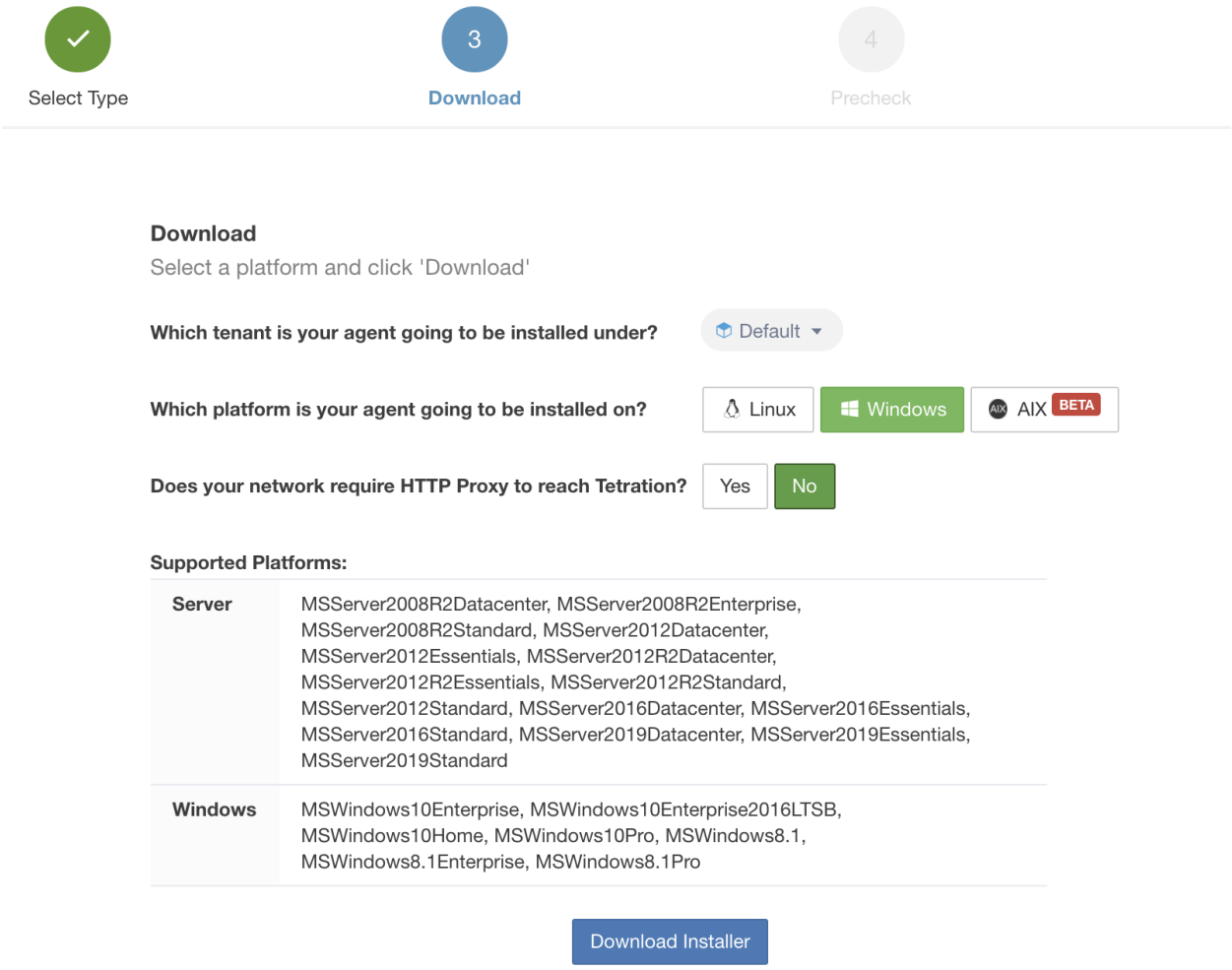


Fig. 3.1.5.1.1: Software Agent Installer Script Download Page (On-prem)

The usage of this installer script is as follows:

powershell -File tetration_windows_installer.ps1 [-preCheck] [-skipPreCheck <Option>] [-noInstall] [-logFile <FileName>] [-proxy <ProxyString>] [-noProxy] [-help] [-version] [-sensorVersion <VersionInfo>] [-ls] [-file <FileName>] [-save <FileName>] [-new] [-reinstall] [-npcap] [-forceUpgrade] [-upgradeLocal] [-upgradeByUUID <FileName>] [-visibility]

-preCheck: run pre-check only

-skipPreCheck <Option>: skip pre-installation check by given option; Valid options include 'all', 'ipv6' and 'enforcement'; e.g.: '-skipPreCheck all' will skip all pre-installation checks; All pre-checks will be performed by default

-noInstall: will not download and install sensor package onto the system

-logFile <FileName>: write the log to the file specified by <FileName>

-proxy <ProxyString>: set the value of HTTPS_PROXY. Use this if proxy is needed to communicate with the cluster. The string should be formatted as http://<proxy>:<port>

-noProxy: bypass system wide proxy; this flag will be ignored if -proxy flag was provided

-help: print this help

-version: print current script's version

-sensorVersion <VersionInfo>: select sensor's version; e.g.: '-sensorVersion 3.4.1.0.win64'; will download the latest version by default if this flag was not provided

-ls: list all available sensor versions for your system (will not list pre-3.1 packages); will not download any package

-file <filename>: provide local zip file to install sensor instead of downloading it from cluster

-save <filename>: download and save zip file as <filename>

-new: remove any previous installed sensor; previous sensor identity has to be removed from cluster in order for the new registration to succeed

-reinstall: reinstall sensor and retain the same identity with cluster; this flag has higher priority than -new

-npcap: overwrite existing npcap

-forceUpgrade: force sensor upgrade to version given by -sensorVersion flag; e.g.: '-sensorVersion 3.4.1.0.win64 -forceUpgrade'; apply the latest version by default if -sensorVersion flag was not provided

-upgradeLocal: trigger local sensor upgrade to version given by -sensorVersion flag; e.g.: '-sensorVersion 3.4.1.0.win64 -upgradeLocal'; apply the latest version by default if -sensorVersion flag was not provided

-upgradeByUUID <FileName>: trigger sensor whose uuid is listed in <FileName> upgrade to version given by -sensorVersion flag; e.g.: '-sensorVersion 3.4.1.0.win64 -upgradeByUUID "C:\Program Files\Cisco Tetration\sensor_id"'; apply the latest version by default if -sensorVersion flag was not provided

-visibility: install deep visibility agent only; -reinstall would overwrite this flag if previous installed agent type was enforcer

Using the downloaded bundle

This section explains how to download an agent image and install it onto the Windows hosts.

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Installer** tab. The **Installer** tab opens.
4. Select **Manual Install using classic packaged installers** workflow and then click **Next**.
5. Find the appropriate version/platform/architecture/agent type and click **Download** button.
6. Copy the zip package to all the Windows hosts for deployment, and follow the below steps with Administrator privilege.
7. Extract the **tet-win-sensor<version>.win64-<clustername>.zip** file, go to the uncompressed folder.
8. Run command **msiexec.exe /i TetrationAgentInstaller.msi** to install, some options are available.

Available options for msi installer:

- **agenttype=<AgentType>** - AgentType should be either "sensor" or "enforcer", depends on whether you need enforcement. By default the installer will check the content of **sensor_type** file in same folder (and overwrites the parameter you passed in). However if agent is installed in **/quiet** mode, this is required.
- **overwritenpcap=yes** - By default the installer do not attempt to upgrade Npcap if Npcap already exists. Pass this parameter and it will try to upgarde existing Npcap.
- **installfolder=<FullPathCustomFolder>** - Use the parameter at the end of the above command to install sensor in a custom folder.
- **serviceuser=<Service UserName>** - Use the parameter at the end of the above command to configure service user. Default service user is "LocalSystem".

  For local user, **serviceuser=.\<Service UserName>**

  For domain user, **serviceuser=<domain_name>\<samaccount name>**

  service user must have **Local Admin privileges**.
- **servicepassword=<Service UserPassword>** - Use the parameter at the end of the above command to configure password for the service user.Password must be in plain-text format.

Notes:

- For new deployment, never execute the binary file WindowsSensorInstaller.exe directly, since this file is only used for migration from 2.3.1.x version.
- If Npcap is not already installed, the installer will install Npcap automatically.
- If **the agent has already been installed, please do not reinstall**. If agent needs to be upgraded to a new version, please use follow the upgrade process described in *Upgrading Software Agents*.

If agent needs to be upgraded to a new version, please use follow the upgrade process described in *Upgrading Software Agents*.

3.1.5.2 Verify that the agent is installed

1. Verify that the folder **C:\Program Files\Cisco Tetration** (or the custom folder) exists.
2. Verify that the service TetSensor (for deep visibility) exists and running.

   Run command **cmd.exe** with **Admin** privileges

   Run command **sc query tetsensor**

   Check state **Running**

   Run command **sc qc tetsensor**

   Check DISPLAY-NAME **Cisco Tetration Deep Visibility**

   OR

   Run command **services.msc**

   Find name **Cisco Tetration Deep Visibility**

   Check status **Running**
3. Verify that the service TetEnforcer (for enforcement) exist and are running.

   Run command **cmd.exe** with **Admin** privileges

   Run command **sc query tetenforcer**

Check state **Running**

Run command **sc qc tetenforcer**

Check DISPLAY-NAME **Cisco Tetration Enforcement**

OR

Run command **services.msc**

Find name **Cisco Tetration Enforcement**

Check status **Running**

3.1.5.3 Verify that the agent is running in the configured service user context

1. Verify that the service TetSensor (for deep visibility) and TetEnforcer (for enforcement) running in the configured service user context. TetSensor and TetEnforcer run in the same service user context.

   Run command **cmd.exe** with **Admin** privileges

   Run command **sc qc tetsensor**

   Check SERVICE_START_NAME **<configured service user>**

   Run command **sc qc tetenforcer**

   Check SERVICE_START_NAME **<configured service user>**

   OR

   Run command **services.msc**

   Find name **Cisco Tetration Deep Visibility**

   Check **Log On As** for the <configured service user>

   Find name **Cisco Tetration Enforcement**

   Check **Log On As** for the <configured service user>

   OR

   Run command **tasklist /v | find /i "tet"**

   Check the user context for the running processes (5th column)

3.1.5.4 Windows Agent Installer and Npcap

1. For the Windows OSs other than Win2008 R2, current supported Npcap version is **0.9990**. For Win2008 R2, current supported Npcap version is **0.991**
2. Installation:

   If Npcap is not installed. Agent installer will install the supported version. If User have Npcap installed but is older than supported version, installation will be blocked. To unblock, upgrade/uninstall Npcap yourself, or run the Agent installer with option **overwritenpcap=yes**, or run installer script with **-npcap** If Npcap driver is in use by any application, Npcap will not be upgraded.
3. Upgrade:

   If Npcap is installed by Windows Agent and verison is older than the supported version, Npcap will be upgraded to the supported version. If Npcap is not installed by Windows Agent, Npcap will not be upgraded. If Npcap driver is in use by any application, Npcap will not be upgraded.

4. Uninstall:

   If Npcap is installed by Windows Agent, it will uninstall Npcap. If Npcap is installed by user, but upgrade by Agent Installer with **overwritenpcap=yes**, it will be uninstalled. If Npcap driver is in use by any application, Agent Installer will not uninstall Npcap.

3.1.6 Deploying a Deep Visibility/Enforcement AIX Agent

3.1.6.1 Install the agent

Deep Visibility/Enforcement AIX Agent can only be installed with the installation script.

Using installer script

The process is as follows:

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Installer** tab. The **Installer** tab opens.
4. Select **Auto-Install using Installers** workflow and then click **Next**.
5. In the step **Select Type**, choose either Deep Visibility or Enforcement and then click **Next**.
6. In the step **Download**, choose tenant that agents will be installed under. Note that in Tetration Saas cluster, no tenant selection is required.
7. In the step **Download**, choose AIX as platform.
8. In the step **Download**, enter http proxy url if needed.
9. Click **Download Installer** button and save the file to local disk.
10. Copy the installer shell script to all the AIX hosts for deployment.
11. Run command chmod u+x tetration_installer_default_sensor_aix.sh to grant execute permission for the script. (note: the script name may differ depending on agent type and scope)
12. Run command ./tetration_installer_default_sensor_aix.sh with root privilege to install agent.

Note that the script **will not proceed if agent has already been installed**.

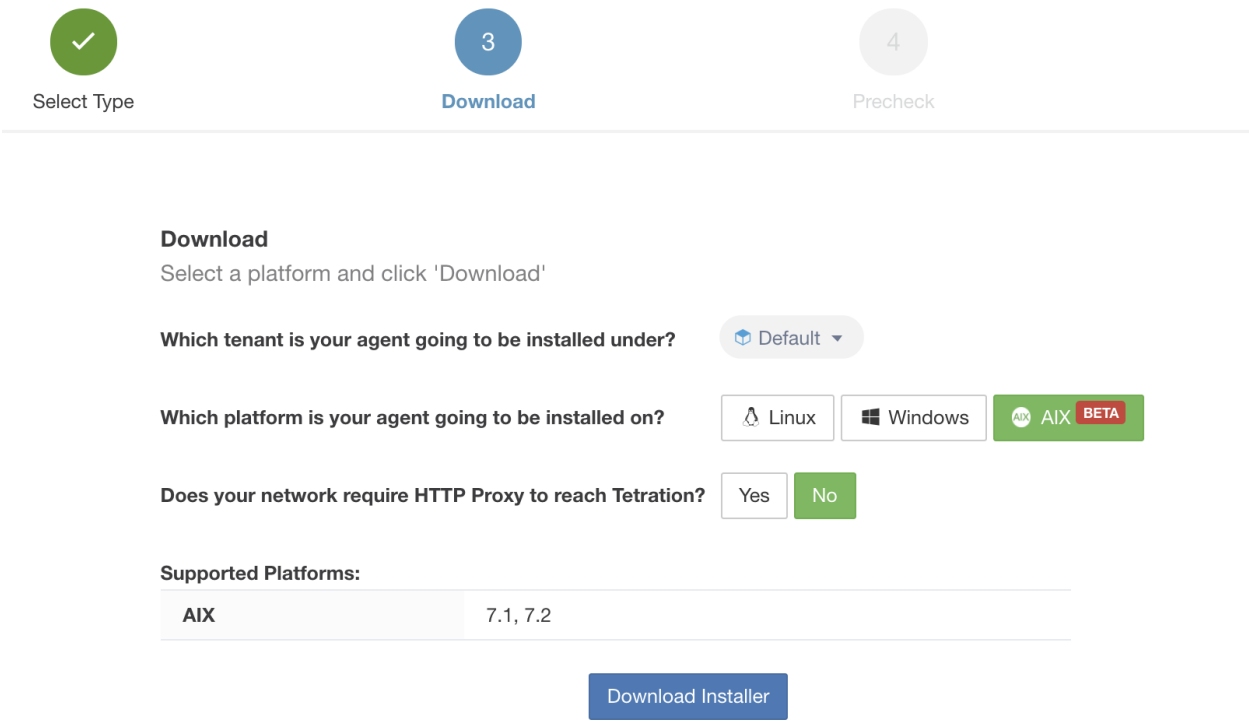


Fig. 3.1.6.1.1: Software Agent Installer Script Download Page (On-prem)

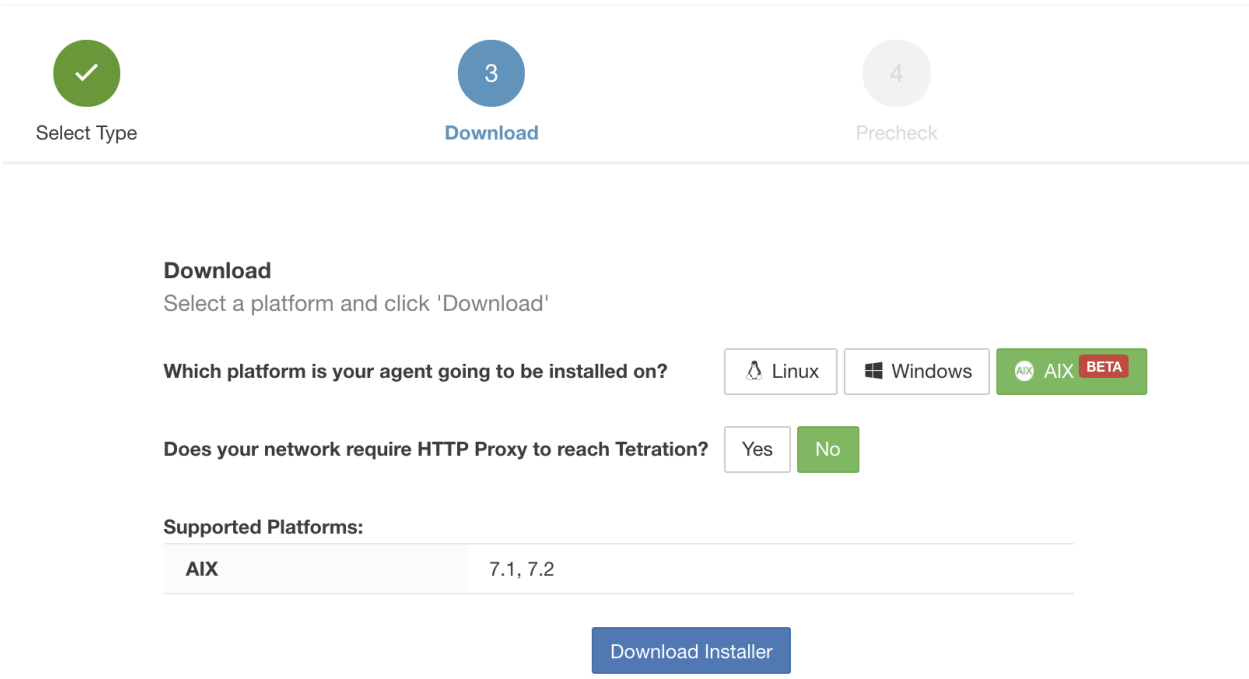


Fig. 3.1.6.1.2: Software Agent Installer Script Download Page (Saas)

The usage of this installer script is as follows:

$ ksh tetration_installer_aix.sh [–pre-check] [–pre-check-user] [–skip-pre-check=<option>] [–no-install] [–logfile=<filename>] [–proxy=<proxy_string>] [–no-proxy] [–help] [–version] [–sensor-version=<version_info>] [–ls] [–file=<filename>] [–osversion=<osversion>] [–save=<filename>] [–new] [–reinstall] [–unpriv-user] [–libs=<libs.zip|tar.Z>] [–force-upgrade] [–upgrade-local] [–upgrade-by-uuid=<filename>] [–logbasedir=<logbdir>] [–visibility]

–pre-check: run pre-check only

–pre-check-user: provide alternative to nobody user for pre-check su support

–skip-pre-check=<option>: skip pre-installation check by given option; Valid options include 'all', 'ipv6' and 'enforcement'; e.g.: '–skip-pre-check=all' will skip all pre-installation checks; All pre-checks will be performed by default

–no-install: will not download and install sensor package onto the system

–logfile <filename>: write the log to the file specified by <filename>

–proxy=<proxy_string>: set the value of HTTPS_PROXY, the string should be formatted as http://<proxy>:<port>

–no-proxy: bypass system wide proxy; this flag will be ignored if –proxy flag was provided

–help: print this help

–version: print current script's version

–sensor-version=<version_info>: select sensor's version; e.g.: '–sensor-version=3.4.1.0'; will download the latest version by default if this flag was not provided

–ls: list all available sensor versions for your system (will not list pre-3.3 packages); will not download any package

–file <filename>: provide local zip file to install sensor instead of downloading it from cluster

–osversion=<osversion>: specify osversion for –save flag

–save=<filename>: download and save zip file as <filename>; will download package for osversion given by –osversion flag; e.g.: '–save=myimage.aix72.zip –osversion=7.2'

–new: remove any previous installed sensor; previous sensor identity has to be removed from cluster in order for the new registration to succeed

–reinstall: reinstall sensor and retain the same identity with cluster; this flag has higher priority than –new

–unpriv-user=<username>: use <username> for unpriv processes instead of tet-snsr

–libs=<libs.zip>: Install provided libs to be used by agents

–force-upgrade: force sensor upgrade to version given by –sensor-version flag; e.g.: '–sensor-version=3.4.1.0 –force-upgrade'; apply the latest version by default if –sensor-version flag was not provided

–upgrade-local: trigger local sensor upgrade to version given by –sensor-version flag: e.g.: '–sensor-version=3.4.1.0 –upgrade-local'; apply the latest version by default if –sensor-version flag was not provided

–upgrade-by-uuid=<filename>: trigger sensor whose uuid is listed in <filename> upgrade to version given by –sensor-version flag; e.g.: '–sensor-version=3.4.1.0 –

upgrade-by-uuid=/usr/local/tet/sensor_id'; apply the latest version by default if –sensor-version flag was not provided

–logbasedir=<log_base_dir>: instead of logging to /opt/cisco/tetration/log use <log_base_dir>. The full path will be <log_base_dir>/tetration

–visibility: install deep visibility agent only; –reinstall would overwrite this flag if previous installed agent type was enforcer

3.1.6.2 Verify that the agent is installed

1. Run command **lslpp -c -l tet-sensor.rte**, confirm that there is one entry as follows (note: the specific output may differ depending on the version)

$ sudo lslpp -c -l tet-sensor.rte /usr/lib/objrepos:tet-sensor.rte:3.4.1.19::COMMITTED:I:TET tet sensor package:

$ sudo lssrc -s tet-sensor

Subsystem Group PID Status tet-sensor 1234567 active

$ sudo lssrc -s tet-enforcer

Subsystem Group PID Status tet-enforcer 7654321 active

3.1.7 Deploying a Deep Visiblity/Enforcement Kubernetes Agent

3.1.7.1 Installer Script

1. Download the agent Installer Script by choosing platform as Kubernetes, supplying the HTTP_PROXY, if needed.
2. Run the installer script on a Linux machine which has access to the Kubernetes API server and also has a kubectl configuration file with admin privileges as the default context/cluster/user.
3. The installer will attempt to read the file from its default location (~/.kube/config), but this can be specified explicitly with the –kubeconfig command line option.
4. The installation script, if successful, will print instructions on how to verify the Tetration Agent Daemonset and Pods that were installed.

Note: The HTTP_PROXY configured on the agent installer page prior to download only controls how Tetration agents and sensors connect to the Tetration cluster. This setting does not affect how Docker images are fetched by the customer's Kubernetes nodes, since the container runtime on those nodes uses its own proxy configuration. If the Docker images are unable to be pulled from the Tetration cluster, debugging the container runtime's image pulling process will be necessary and adding a suitable HTTP proxy might be necessary.

3.1.8 Deploying Universal Linux Agent

3.1.8.1 Install the agent

1. Download the agent bundle similar to the process to download deep visibility or enforcement agents, and choose the appropriate bundle for universal agents.
2. Extract the **tet-sensor-lw-<version>-lw-<arch>.zip** file.
3. Follow the README text file for detailed instructions. Alternatively, run the script **install.sh** with Root privilege to finish the installation.

3.1.8.2 Verify that the agent is installed

1. Verify that the base folder **/usr/local/tet-light** exists (using: ls).
2. Verify that the scheduled cron jobs "Tetration Lightweight Sensor Job: Send flow" and "Tetration Lightweight Sensor Job: Send machine info" exist and are active (using: crontab -l).

3.1.9 Deploying Universal Windows Agent

3.1.9.1 Install the agent

1. Download the agent bundle similar to the process to download deep visibility or enforcement agents, and choose the appropriate bundle for universal agents.
2. Extract the **tet-sensor-lw-<version>-lw-<arch>.zip** file.
3. Follow the README text file for detailed instructions. Alternatively, run the script **install.cmd** with Administrator privilege to finish the installation.

3.1.9.2 Verify that the agent is installed

1. Verify that the folder **C:\Program Files\Cisco Tetration** exists (using: dir)
2. Verify that the scheduled tasks "Tetration Lightweight Sensor - Flow" and "Tetration Lightweight Sensor - Machine" exist and are running. (using: schtasks | findstr Lightweight)

3.1.10 Deploying the Secure Connector Client

3.1.10.1 Deployment overview

Tetration Secure Connector creates a reverse tunnel from the Tetration cluster to your internal network in order to reach your orchestrator API servers. See *Tetration Secure Connector*.

Starting the secure connector client is done in three steps:

1. Download and install the Secure Connector client package on a supported platform.
2. Retrieve a single-use time-limited token through the Tetration API.
3. Copy the token to the client configuration.

Proxy support

The Secure Connector client supports connecting to the Tetration cluster through an HTTP(S) proxy. If needed, the proxy server must be configured by setting the HTTPS_PROXY environment variable for the client. To set the variable, add the following line in the *[Service]* section of the systemd service file located at */etc/systemd/system/tetration-secure-connector.service*. This setting will not persist across re-installations. For a sticky configuration, the line can be added in a new file at */etc/systemd/system/tetration-secure-connector.service.d/10-https-proxy.conf*. For either configurations to take effect, reload the systemd config by running *systemctl daemon-reload*.

```
[Service]
Environment="HTTPS_PROXY=<Proxy Server Address>"
```

3.1.10.2 Deployment Steps

Install the Secure Connector client

Use the following steps to download and install the Secure Connector client package on a supported Linux host:

1. Click the **settings** (gears icon) in the top-right corner.
2. Select **Agent Config** in the drop-down menu.
3. Click the **Software Agent Download** tab.
4. Click the **Show Classic Agent Packages** button on the top right-hand side.
5. The Secure Connector Client packages will have the agent type "Secure Connector".
6. Find the appropriate version (if multiple are available on the cluster) and click the **Download** button.
7. Copy the rpm package to the Linux host for deployment, and execute the following command with root privilege:

```
rpm -ivh <rpm_filename>
```

Retrieve a new token using the API

Secure Connector tokens can only be retrieved through OpenAPI (*Get Token endpoint*). The following python and bash snippets can be used to retrieve a new token. Note that the API key used must have the *external_integration* capability and must have write access to the specified root scope. See (*OpenAPI Authentication*) for information on installing the Tetration OpenAPI client for python and creating a new API key.

Python snippet for token retrieval

```
from tetpyclient import RestClient
from urllib import quote

API_ENDPOINT = "https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"
ROOT_SCOPE_NAME = r"""<ROOT_SCOPE_NAME>"""
API_CREDENTIALS_FILE = "<API_CREDENTIALS_JSON_FILE>"
OUTPUT_TOKEN_FILE = "registration.token"

if __name__ == "__main__":
  client = RestClient(API_ENDPOINT,
                      credentials_file=API_CREDENTIALS_FILE)  # Add (verify=False) to␣
↪skip certificate verification
  escaped_root_scope_name = quote(ROOT_SCOPE_NAME, safe='')
  resp = client.get('/secureconnector/name/{}/token'.format(escaped_root_scope_name))
  if resp.status_code != 200:
    print 'Error ({}): {}'.format(resp.status_code, resp.content)
    exit(1)
  else:
    with open(OUTPUT_TOKEN_FILE, 'w') as f:
      f.write(resp.content)
```

BASH snippet for token retrieval

```
#!/bin/bash
HOST="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"
API_KEY="<API_KEY>"
API_SECRET="<API_SECRET>"
```

(continues on next page)

(continued from previous page)

```
ROOTSCOPE_NAME="<ROOT_SCOPE_NAME>" # if the name contains spaces or special␣
↪characters, it should be url-encoded
TOKEN_FILE="registration.token"
INSECURE=1 # Set to 0 if you want curl to verify the identity of the cluster

METHOD="GET"
URI="/openapi/v1/secureconnector/name/$ROOTSCOPE_NAME/token"
CHK_SUM=""
CONTENT_TYPE=""
TS=$(date -u "+%Y-%m-%dT%H:%M:%S+0000")
CURL_ARGS="-v"
if [ $INSECURE -eq 1 ]; then
    CURL_ARGS=$CURL_ARGS" -k"
fi

MSG=$(echo -n -e "$METHOD\n$URI\n$CHK_SUM\n$CONTENT_TYPE\n$TS\n")
SIG=$(echo "$MSG"| openssl dgst -sha256 -hmac $API_SECRET -binary | openssl enc -
↪base64)
REQ=$(echo -n "curl $CURL_ARGS $HOST$URI -w '%{http_code}' -H 'Timestamp: $TS' -H
↪'Id: $API_KEY' -H 'Authorization: $SIG' -o $TOKEN_FILE")
status_code=$(sh -c "$REQ")
if [ $status_code -ne 200 ]; then
    echo "Failed to get token. Status: " $status_code
else
    echo "Token retrieved successfully"
fi
```

Copy the token and start the client

By the end of step 2 you should have a *registration.token* file that contains the single-use limited-time token for bootstrapping the client. On the host where you installed the Secure Connector client package, make sure the Secure Connector client is stopped before copying the token file. You can use the following command:

```
systemctl stop tetration-secure-connector
```

Place the token file at the following location:

```
/etc/tetration/cert/registration.token
```

Restart the Secure Connector Client.

```
systemctl start tetration-secure-connector
```

3.1.10.3 Verify the state of the Secure Connector client

You can check whether the Secure Connector client is installed by querying the rpmdb for the package *tet-secureconnector-client-site*

```
rpm -q tet-secureconnector-client-site
```

To check the current state of the installed client, you can check the status of the systemd service *tetration-secure-connector*

```
systemctl status tetration-secure-connector
```

3.1.10.4 Upgrading the Secure Connector Client

The Secure Connector client does not support automatic updates. To install a new version of the software, you can use the following command to uninstall the current version then proceed with following the (*installation steps*) for the new version.

```
rpm -e tet-secureconnector-client-site
```

3.1.11 User configuration for Tetration SaaS

Note that when Tetration agents are deployed for Tetration SaaS cluster (optionally called as TaaS), they need some extra information in order to properly register with the backend services. This information is in a file named **user.cfg** which is pre-populated in the Tetration Agent installation folder (Ex: "**/usr/local/tet**" on Linux or "**C:\Program Files\Cisco Tetration**" on Windows). The file contains a list of variables in the form of "key=value", one on each line.

In order for the agents to be able to register to the TaaS cluster, they need to be provided with the cluster activation key. The activation key can be retrieved from the "Software Agent Download" tab under "Agent Config" page, by clicking on "Agent Activation Key." The key then needs to be added to the **user.cfg** file via the **ACTIVATION_KEY** variable.

If the agents are deployed via the installer script, the backend will fill the value of ACTIVATION_KEY accordingly for each customer, therefore avoid the need to manually update this value in the user.cfg file.

In case a proxy is needed to reach the TaaS cluster, the information needs to be passed to the agents through the same user.cfg file. This is done by adding the **http** protocol proxy server and port information via the **HTTPS_PROXY** variable.

If agents are deployed using installer script, then user can give the proxy information as command line flag –proxy or -proxy in case of windows installer.

Following is an example of user.cfg file content with both ACTIVATION_KEY and HTTPS_PROXY populated.

ACTIVATION_KEY=7752163c635ef62e6568e9e852d07bd21bfd60d0 HTTPS_PROXY=http://proxy.my-company.com:80

3.1.12 Connectivity information

In general, once the agent is installed onto the workload, it will start making a number of network connections to the backend services hosted on Tetration cluster. Depending on agent type and its functionalities, the number of connections will look different.

The following table captures various permanent connections made by various agent types.

Table 3.1.12.1: Agent connectivity

Agent type	Config server	Collectors	Enforcement backen
visibility (on-prem)	CFG-SERVER-IP:443	COLLECTOR-IP:5640	N/A
visibility (taas)	CFG-SERVER-IP:443	COLLECTOR-IP:443	N/A
enforcement (on-prem)	CFG-SERVER-IP:443	COLLECTOR-IP:5640	ENFORCER-IP:5660
enforcement (taas)	CFG-SERVER-IP:443	COLLECTOR-IP:443	ENFORCER-IP:443
universal (on-prem)	CFG-SERVER-IP:443	COLLECTOR-IP:5640	N/A
universal (taas)	CFG-SERVER-IP:443	COLLECTOR-IP:443	N/A

Continued on next pag

Table 3.1.12.1 – continued from previous page

Agent type	Config server	Collectors	Enforcement backen
docker images	CFG-SERVER-IP:443	N/A	N/A

Legends:

- CFG-SERVER-IP represents the IP address of the config server
- COLLECTOR-IP represents the IP address of the collector. Deep visibility and enforcement agent will connect to all available collectors, while universal agent will randomly pick one.
- ENFORCER-IP represents the IP address of the enforcement endpoint. Enforcement agent will connect to only one of the available endpoints.
- For Kubernetes agent deployments, the installation script does not contain the agent software - Docker images containing the agent software will be pulled from the Tetration cluster by every Kubernetes node. These connections will be established by the container runtime image fetch component and directed at CFG-SERVER-IP:443.

Notes:

- Tetration agent always acts as a client to initiate the connections to the services hosted within the cluster, it will never open a connection as a server.
- In addition to the above permanent connections, for the given agent type that upgrade is supported, agent will periodically perform https requests (port 443) to the cluster sensor VIP to query the available packages.
- Agent is allowed to be located behind a NAT server.

It is important to note that if the workload is behind a firewall or the host firewall service is enabled, then the connections to the cluster might be denied. It is neccessary for the administrators to allow such connections by creating appropriate firewall policies.

3.2 Security Exclusions

Cisco Tetration Agents continuously interact with the host's operating system during their normal operations. This may sometimes cause other security applications (antivirus, security agents, . . .) installed on the host to raise alarms about the Tetration Agents, or even to block Tetration Agents' actions. To ensure a proper installation and an effective functioning of Cisco Tetration Agents, please configure the necessary security exclusions on the security applications that are monitoring the host.

Table 3.2.1: Security exclusions for Tetration Agents directories

Host OS	Directories
AIX	/opt/cisco/tetration
Linux	/usr/local/tet or /opt/cisco/tetration or <user chosen inst dir>
Windows	C:\Program Files\Cisco Tetration

Table 3.2.2: Security exclusions for Tetration Agents processes

Host OS	Processes
AIX	tet-engine, tet-sensor, tet-enforcer
Linux	tet-engine, tet-sensor, tet-enforcer, tet-main, enforcer
Windows	TetSenEngine.exe, TetSen.exe, TetEnfEgine.exe, TetEnfC.exe, TetEnf.exe, TetUpdate.exe, tet-main.exe

Table 3.2.3: Security exclusions for Tetration Agents actions

Host OS	Actions
AIX	Access /dev/bpf*, /dev/ipl, /dev/kmem, invokes: curl
Linux	Scan /proc, open netlink sockets, invokes: curl, rpm/dpkg, ip[6]tables-save, ip[6]tables-restore, ipset-restore
Windows	Access Registry, register to Firewall Events

Table 3.2.4: Security exclusions for Tetration Agents scripts/binaries executions

Host OS	Invoked scripts/binaries
AIX	ksh: fetch_sensor_id.sh, check_conf_update.sh
Linux	bash: fetch_sensor_id.sh, check_conf_update.sh
Windows	cmd: fetch_sensor_id.cmd, check_conf_update.cmd, dmidecode.exe, npcap-installer.exe, sensortools.exe, signtool.exe

3.3 Software Agents Service Management

With the exception of universal agents, the software agents are deployed as a service in all supported platforms. This section describes methods to manage the services for various functionalities and platforms.

Note that unless specified, all the below commands require root privileges (Linux/Unix) or Administrator privileges (Windows) to execute.

3.3.1 Service management for RHEL/CentOS/OracleLinux-6.x and Ubuntu-14

3.3.1.1 Starting a service

Execute the command **start <service-name>**

Examples: - **start tet-sensor** for deep visiblity service - **start tet-enforcer** for enforcement service

3.3.1.2 Stopping a service

Execute the command **stop <service-name>**

Examples: - **stop tet-sensor** for deep visiblity service - **stop tet-enforcer** for enforcement service

3.3.1.3 Restarting a service

Execute the command **restart <service-name>**

Examples: - **restart tet-sensor** for deep visiblity service - **restart tet-enforcer** for enforcement service

3.3.1.4 Checking service status

Execute the command **status <service-name>**

Examples: - **status tet-sensor** for deep visiblity service - **status tet-enforcer** for enforcement service

3.3.2 Service management for SLES-11

3.3.2.1 Starting a service

Execute the command **service <service-name> start**

Examples: - **service tet-sensor start** for deep visiblity service - **service tet-enforcer start** for enforcement service

3.3.2.2 Stopping a service

Execute the command **service <service-name> stop**

Examples: - **service tet-sensor stop** for deep visiblity service - **service tet-enforcer stop** for enforcement service

3.3.2.3 Restarting a service

Execute the command **service <service-name> stop || true** followed by **service <service-name> start**

3.3.2.4 Checking service status

Execute the command **status <service-name>**

Examples: - **status tet-sensor** for deep visiblity service - **status tet-enforcer** for enforcement service

3.3.3 Service management for RHEL/CentOS/OracleLinux-7.x and 8.x

The same commands can be also used for Ubuntu-16,18,20 and SLES-12.

3.3.3.1 Starting a service

Execute the command **systemctl start <service-name>**

Examples: - **systemctl start tet-sensor** for deep visiblity service - **systemctl start tet-enforcer** for enforcement service

3.3.3.2 Stopping a service

Execute the command **systemctl stop <service-name>**

Examples: - **systemctl stop tet-sensor** for deep visiblity service - **systemctl stop tet-enforcer** for enforcement service

3.3.3.3 Restarting a service

Execute the command **systemctl restart <service-name>**

Examples: - **systemctl restart tet-sensor** for deep visiblity service - **systemctl restart tet-enforcer** for enforcement service

3.3.3.4 Checking service status

Execute the command **systemctl status <service-name>**

Examples: - **systemctl status tet-sensor** for deep visiblity service - **systemctl status tet-enforcer** for enforcement service

3.3.4 Service management for Windows Server or Windows VDI

3.3.4.1 Starting a service

Execute the command **net start <service-name>**

Examples: - **net start tetsensor** for deep visiblity service - **net start tetenforcer** for enforcement service

3.3.4.2 Stopping a service

Execute the command **net stop <service-name>**

Examples: - **net stop tetsensor** for deep visiblity service - **net stop tetenforcer** for enforcement service

3.3.4.3 Restarting a service

Execute the command **net stop <service-name>** followed by a **net start <service-name>** command

3.3.4.4 Checking service status

Execute the command **sc query <service-name>**

Examples: - **sc query tetsensor** for deep visiblity service - **sc query tetenforcer** for enforcement service

3.3.5 Service management for AIX

3.3.5.1 Starting a service

Execute the command **startsrc -s <service-name>**

Examples: - **startsrc -s tet-sensor** for deep visiblity service - **startsrc -s tet-enforcer** for enforcement service

3.3.5.2 Stopping a service

Execute the command **stopsrc -s <service-name>**

Examples: - **stopsrc -s tet-sensor** for deep visiblity service - **stopsrc -s tet-enforcer** for enforcement service

3.3.5.3 Restarting a service

Execute the command **stopsrc -s <service-name>** followed by **startsrc -s <service-name>**

3.3.5.4 Checking service status

Execute the command **lssrc -s <service-name>**

Examples: - **lssrc -s tet-sensor** for deep visiblity service - **lssrc -s tet-enforcer** for enforcement service

3.3.6 Service management for Kubernetes Agent installations

3.3.6.1 Starting/Stopping a service

It is not possible to stop or start the agents on a specific node since they are not installed as individual services but rather as a cluster-wide daemonset.

3.3.6.2 Restarting an Agent on a node

Locate the Tetration agent Pod on the node and run the appropriate Kubernetes command to kill it. The pod will be restarted automatically.

3.3.6.3 Checking Status of Pods

kubectl get pod -n tetration will list the status of all Tetration agent pods in the Kubernetes cluster.

3.4 Upgrading Software Agents

3.4.1 Upgrade agents from UI

Agents can be upgraded using Agent Config Intent workflow detailed here - *Software Agent Config*. While configuring an Agent Config Profile, there is an 'Auto Upgrade' option which can be 'Enabled' or 'Disabled'. If the option is 'Enabled', then the agents matching inventory filter criteria are auto upgraded to the latest version of software available.

Following section describes how to use software agent config intent workflow to dictate software agent upgrade behavior:

1. Create an inventory filter on the Inventory Filters page. More details here - *Filters*.

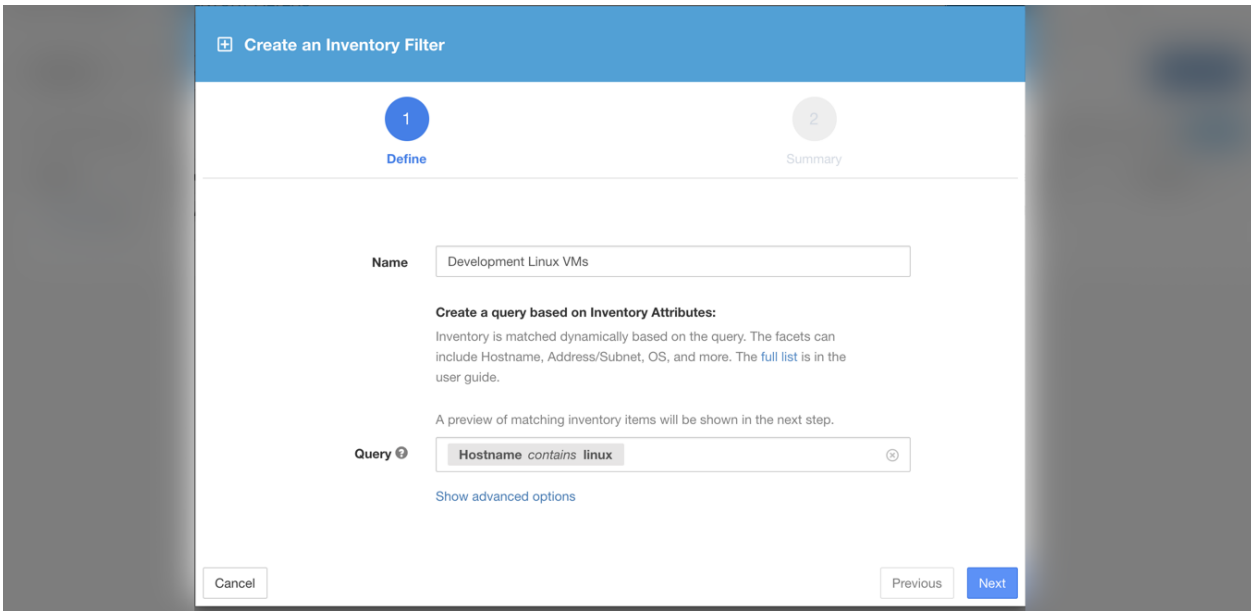


Fig. 3.4.1.1: Inventory Filter

2. Create an Agent Config profile user wants to apply to the agents chosen by the above inventory filter. Note, in agent config profile, there is an 'Auto Upgrade' option which governs whether chosen agents will get auto-upgraded or not.

Fig. 3.4.1.2: Agent Config

3. Finally, an agent config intent needs to be created which applies the chosen config profile to a chosen set of

agents (via inventory filter). If the auto upgrade option is enabled, all chosen agents will get auto upgraded. Usually, it can take up to 30 minutes for agents to upgrade once an agent profile is applied to them.

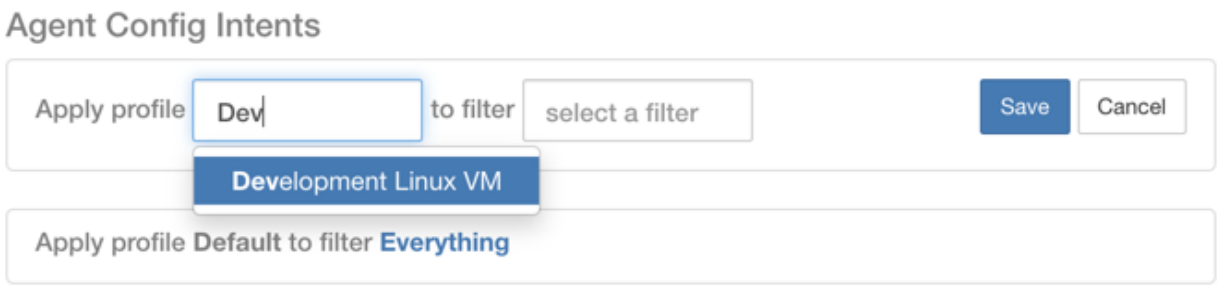


Fig. 3.4.1.3: Agent Config Intent

Note: Auto Upgrade setting in the default agent profile applies to ERSPAN or NETFLOW agents.

The following section explains how to manually upgrade agents without using the Sensor Config intent workflow.

1. Click the **Settings menu** in the top-right corner.
2. Select **Software Agent Configure**. The **Software Agent Configure** page displays.
3. Click the **Upgrade** tab. The **Upgrade** tab opens.
4. Only Deep Visibility and Enforcement agents will be shown and for each agent only upgradable newer versions will be shown in the list. As default the most recent version is selected.
5. Filter the agent list by entering your search queries in the search box.

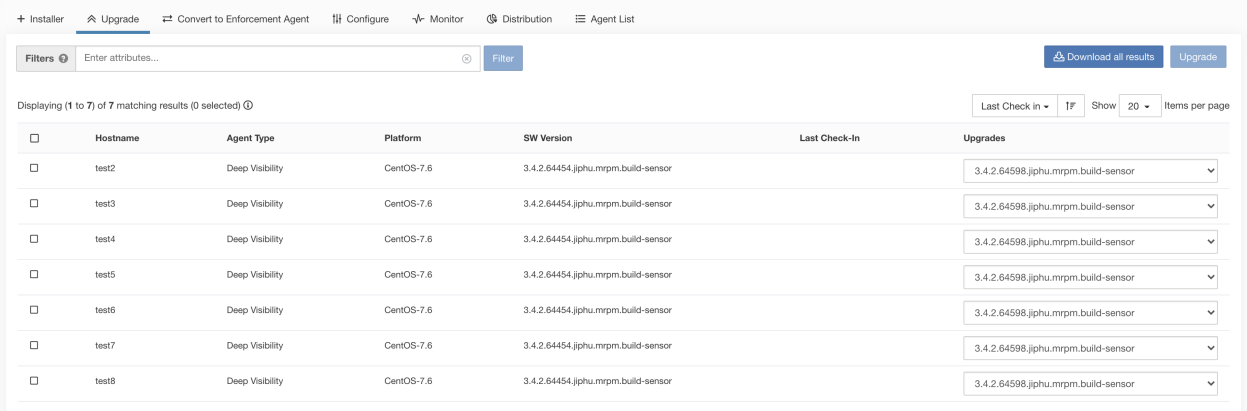


Fig. 3.4.1.4: Software Agent Upgrade

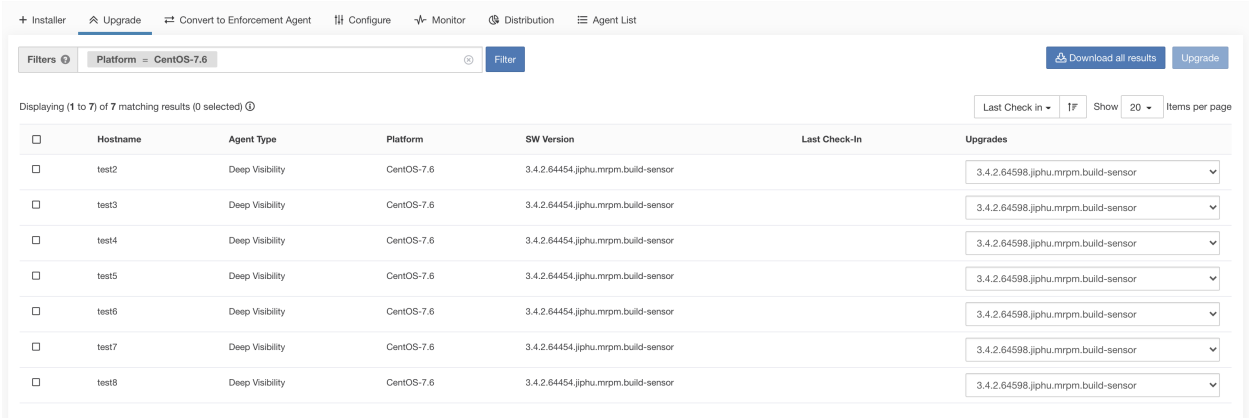


Fig. 3.4.1.5: Software Agent Upgrade - Search Agents

6. Select the agents to be upgraded to this version and click the **Upgrade** button.

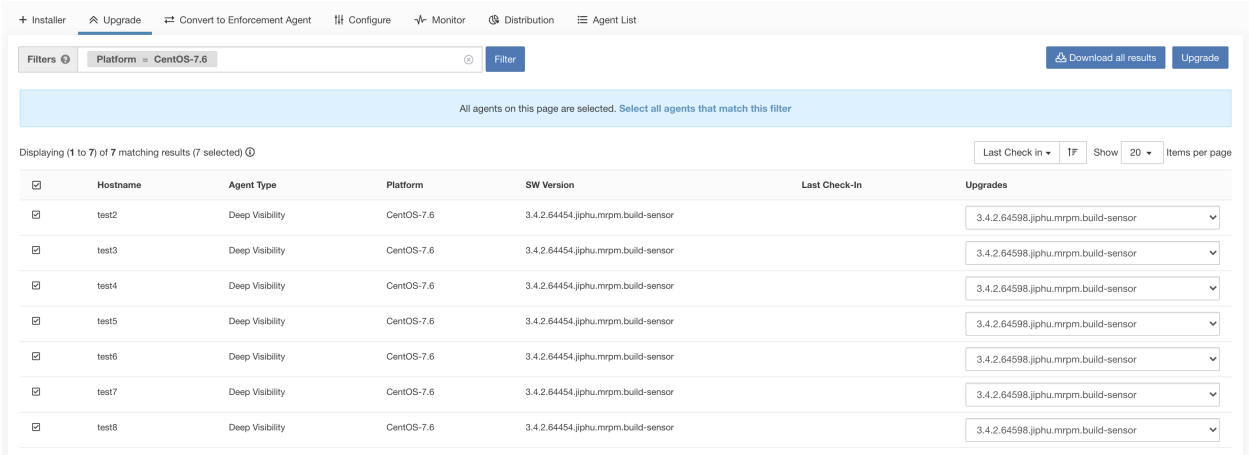


Fig. 3.4.1.6: Software Agent Upgrade - Select Agents

Notes:

- Under normal circumstances, letting the agent handle the upgrade is strongly recommended and is the only supported upgrade method. If users would like to control the upgrade by manually downloading the newever version and apply directly over running agents, be sure to follow the safety precautions when doing this.
- Universal agents currently do not support upgrade.

3.4.2 Migration behavior for Windows agents

In pre-3.1.1.x release, Windows agents has been deployed with an .exe file called WindowsSensorInstaller.exe. From 3.1.1.x release onwards, agent will be delivered as MSI package which is a native format supported by Microsoft. If agent is upgraded using the UI workflow, it will be able to perform a migration job from the old .exe format to the new MSI format. Therefore:

- Users should never download and execute the newer MSI package over the existing running agent. The result of this behavior will be undesirable.
- Users should never manually deploy an older version of agent MSI over an existing running agent. For example if agent is running 3.1.1.59 users should not deploy 3.1.1.55 package manually. The result of this behavior will be undesirable.

3.4.3 Upgrade Behaviour of Kubernetes Agent

Agents installed on Kubernetes nodes using the daemonset installer script are capable of self-upgrade. The upgrade process is controlled by either the auto-upgrade option or by manually triggering an upgrade for any node in the Kubernetes cluster. The mechanism of the upgrade in this environment is to upgrade the Docker image in the daemonset specification, which means that an upgrade of one agent affects all agents covered by the daemonset, as explained in the next paragraph.

When a Daemonset Pod specification changes, Kubernetes will trigger a graceful shutdown, fetch the new docker image(s) and start the Tetration agent pods on ALL nodes in the Kubernetes cluster. This will cause agents to be upgraded on other nodes, even if the policy to allow upgrades is applicable only to a subset of the nodes in the cluster.

If auto-upgrade is disabled for all nodes, manual upgrade is possible by downloading a new installer script and re-running the install. The installation script auto-detects the case of new installation vs upgrading an existing installation and will work to manually upgrade the daemonset pods when it detects an installation is already in place.

3.5 Removing Software Agents

3.5.1 Removing a Deep Visibility/Enforcement Linux Agent

RPM based installation:

1. Run command **'rpm -e tet-sensor'**
2. Delete the agent from UI on **Software Agent** page

Ubuntu .deb based installation:

Fresh installation of Ubuntu agents now uses the native .deb format.

1. Run command **'dpkg –purge tet-sensor'**
2. Delete the agent from UI on **Software Agent** page

Notes:

- By default not all the files are deleted after agent is uninstalled. Log files, for example, are preserved. Users can manually delete all these files.
- During the agent operations, it is possible that some kernel modules will be loaded automatically by the kernel. For example, if enforcement is enabled in Linux, Netfilter modules might be loaded. Agents do not have a list of modules loaded by kernel. Therefore, during agent uninstallation, it cannot possibly unloaded the kernel modules.

3.5.2 Removing a Deep Visibility/Enforcement Windows Agent

There are two options to uninstall Tetration agents:

1.1 Go to Control Panel / Programs / Programs And Features, and uninstall **Cisco Tetration Agent**. 1.2 Alternatively, run the shortcut **Uninstall.lnk** within **'C:\Program Files\Cisco Tetration'**. 2. Delete the agent from UI on **Software Agent** page

Notes:

- If Npcap has been installed during agent installation, it will also get uninstalled.
- By default log files, config files and certs will not get removed during uninstall. If you'd like to remove them, run the shortcut **UninstallAll.lnk** in same folder.

3.5.3 Removing a Deep Visibility/Enforcement AIX Agent

1. Run command **'installp -u tet-sensor'**
2. Delete the agent from UI on **Software Agent** page

Notes:

- By default not all the files are deleted after agent is uninstalled. Log files, for example, are preserved. Users can manually delete all these files.
- The Deep Visibility Agent is controlled by System Resource Controller as tet-sensor. As such it is possible to start, stop, restart and remove it. The service is made persistent with inittab as tet-sen-engine.
- The Enforcement Agent is controlled by System Resource Controller as tet-enforcer. As such it is possible to start, stop, restart and remove it. The service is made persistent with inittab as tet-enf-engine.
- During the agent operations, it is possible that some kernel modules will be loaded automatically by the kernel. For example, if enforcement is enabled in AIX, ipfilter modules are loaded. Agents do not have a list of modules loaded by kernel. Therefore, during agent uninstallation, it cannot possibly unloaded the kernel modules.

3.5.4 Removing Universal Linux Agent

1. Run the uninstall script **'/usr/local/tet-light/uninstall.sh'**
2. Delete the agent from UI on **Software Agent** page

3.5.5 Removing Universal Windows Agent

1. Run the uninstall script **'C:\Program Files\Cisco Tetration\Lightweight Sensor\uninstall.cmd'**
2. Delete the agent from UI on **Software Agent** page

3.5.6 Removing the Secure Connector Client

The Secure Connector Client can be uninstalled using command **'rpm -e tet-secureconnector-client-site'**

3.5.7 Removing a Enforcement Kubernetes Agent

1. Locate the original installer script or download a new script from the Tetration UI.
2. Run the uninstall option **install.sh –uninstall**. The same considerations apply as during the install.
    - Only supported on Linux x86_64 architectures.
    - Either ~/.kube/config contains an admin credentials user or use

the –kubeconfig option to point to the kubectl admin credentials file.

3. Delete the agents for all the Kubernetes nodes from UI on **Software Agent** page

3.6 Data collected and exported by deep visibility agents

This section describes the main components of a software agent, how it is registered with backend services, what data are collected and exported to the cluster for analytical purposes.

3.6.1 Registration

After the agent has been successfully installed onto the system, it needs to register with the backend services to obtain a valid unique identifier. The following information is sent in the registration request:

- Hostname
- BIOS-UUID
- Platform information (such as CentOS-6.5)
- Self-generated client certificate (generated with openssl command)
- Agent type (visibility or enforcement..)

If the agent fails to obtain a valid id from the server, it will keep retrying until it gets one. It is very important that the agent is registered, otherwise all the subsequent communication with other services (such as collectors) will be rejected.

3.6.2 Agent upgrade

Periodically (around 30 minutes), the agent sends a message to backend service to report its current version. The backend service uses the agent's id and its current version to decide whether a new software package is available for the agent. The following information is sent:

- Agent's id (obtained after successful registration)
- Current agent's version

3.6.3 Config server

Agents export the following information to the configured config server:

- Hostname
- Agent's id (obtained after successful registration)
- List of interfaces, each includes:

1. Interface's name
2. IP family (IPv4 or IPv6)
3. IP addresses
4. Netmask
5. Mac addresses

6. Interface's index

As soon as any interface property changes (such as an IP address of an existing interface changes, or a new interface comes up), this list is refreshed and reported to the config server.

3.6.4 Data export

The agent exports various information to collectors periodically. However, the rate at which the information is exported varies.

3.6.4.1 Network flow

Network Flow information is the summarization of all packets flowing through the system. There are two modes of capturing flow information: Detailed and Conversation. By default the Detailed mode of capture is used. The captured flows are exported to collector every one second (this can be changed via config). Exported information includes:

- Flow identifier: uniquely identify the network flow. It includes the general information such as: IP protocol, source and destination IP, and layer 4 ports
- IP Information: contains information seen in IP header, such as: TTL, IP flags, Packet ID, IP options and Fragmentation flags
- TCP Information: contains information seen in TCP header, such as: sequence number, Ack number, TCP options, Rcvd windows size
- Flow Information: flow's statistics (such as: total packets, total bytes, TCP flags statistics, packet length statistics and socket statistics), interface index from which flow was observed, flow's start time and end time

In Conversation mode, the agent will report active flows once every twenty five seconds to five minutes. The flow export time depends on the protocol, with newer and completed flows being reported within the next twenty five seconds after the flow was seen. No packet/byte count and TCP flag information are reported. Agents will only export TCP flows that are birectional in nature along with other connectionless flows. Conversation mode is only supported on Windows and Linux platforms. In case conversation mode is enabled, other platformss like AIX will still report flow information in Detailed mode.

Note that in either mode agent will not export the following flows:

- ARP/RARP conversations
- Agent's flows to collectors

3.6.4.2 Machine information

Machine info describes all the processes running on the host. In addition, it contains network information that is associated with the processes and the command used to launch the processes. Machine info is exported every minute and includes the following information:

- Process ID
- User ID: owner of the process
- Parent Process ID
- Command string used to launch the process
- Socket information: protocol (such as UDP or TCP), address type: IPv4 or IPv6, source and destination IP, source and destination port, TCP state, process's start and end time, path to process binary
- Forensic information: for more information please refer to section *Compatibility*

3.6.4.3 Agent statistics

Agent keeps track of various statistics, including system's statistics and its own, such as:

- Agent's start time and uptime
- Agent's run time in user mode and kernel mode
- Number of packets received and dropped
- Number of successful and failed SSL connections
- Total flow packets and bytes
- Total exported flows and packets to collectors
- Agent's memory and CPU usage

3.7 Tetration Enforcement Agent

This section describes Tetration Enforcement Agent components, messaging and interaction, UI configurations and troubleshooting.

3.7.1 Enforcement Agent

Enforcement Agent is a lightweight process deployed on the endpoints. It receives policies over a secured TCP/SSL channel from the controller via Enforcement Front End (EFE). The received policies are in a platform independent schema. Enforcement Agent converts these platform independent policies into platform specific policies and programs the firewall on the endpoint. Enforcement Agent actively monitors the firewall state. If the Enforcement Agent detects any deviation in the enforced policies, it enforces the cached policies into the firewall again. Enforcement Agent can control the complete firewall or work in conjunction with user configured rules. There is a configuration option to allow user-configured rules to co-exist with Tetration policies. Enforcement Agent runs in privileged domain. On linux machines, Enforcement Agent runs as root while on windows machines, Enforcement Agent runs as SYSTEM. Enforcement Agent also monitors its system resource consumption like CPU and memory. Enforcement Agent enforces policies on the endhost only when it is enabled on the UI. For more information on policies, refer *Policies*.

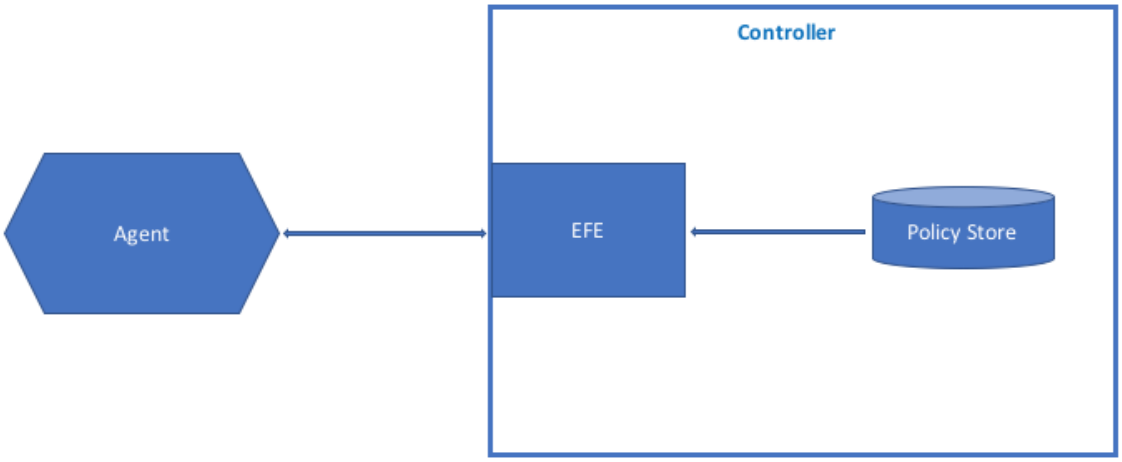


Fig. 3.7.1.1: Enforcement Agent

Enforcement Agent operations

- Communication with controller

Enforcement Agent communicates with EFE through a bidirectional and secure channel via TLS/SSL protocol. Messages from the controller are signed by the policy generator and verified by the Enforcement Agent.

- Tetration Network Policy Message

The Tetration Network Policy is the concrete set of rules corresponding to the effective intent applicable to the host. It consists of the following sections:

> **Firewall Rules**: This is an ordered set of rules that specify whether the firewall should ALLOW or DROP specific network traffic based on parameters such as the source, destination, port, protocol, direction, etc. Agents will program the rules according to the order received by the controller (for both ingress/egress and IPv4/IPv6).
>
> **Catch-all Rules**: These are default actions of ALLOW or DROP in each direction that cover the traffic that do not match any explicitly specified rules.

- Tetration Agent Config Message

The controller sends Agent configuration message which carries various flags to control the Enforcement Agent's behavior. These flags are explained as follows:

> **enable enforcement**: When this flag is set, Enforcement Agent is ready to enforce Tetration rules into the firewall. It programs golden rules which allow connections to the contoller and clears other firewall state depending on the preserve rules flag mentioned below. If any last known policy was received, Enforcement Agent enforces it soon after it is enabled. If enable enforcement flag is not set (default), Enforcement Agent is idle. If enforcement was enabled and then disabled, Enforcement Agent clears the firewall state and sets the catch-all default action to ALLOW.
>
> **preserve rules**: When preserve rules flag is set, Enforcement Agent controls only the Tetration rules and these rules will co-exist with user configured rules in the firewall. If this flag is not set, Enforcement Agent controls the complete firewall and only Tetration rules will be maintained in the firewall.

enable broadcast: When this flag is set (default), Enforcement Agent programs firewall to allow ingress and egress broadcast traffic.

enable multicast: When this flag is set (default), Enforcement Agent programs firewall to allow ingress and egress multicast traffic.

windows enforcement mode: Windows enforcement mode can be set to WAF (Default enforcement mode) or WFP. In WAF mode, network policies are enforced using Windows Advanced Firewall. In WFP mode, network policies are enforced by directly programming WFP filters in the Windows Filter Engine.

- Reports from agents to controller

Enforcement Agent sends periodic status and stats report to the controller via EFE. Status report includes the latest programmed policies status (success/failure/error if any). Stats report includes the policy stats (allowed/dropped packet and byte count) depending on the platform.

UI Configurations

Agent Config Profiles

To configure Agent Config Profile:

- Click on Settings at the left top corner.
- Click on Agent Config
- On the Software Agent Config Tab, click on Create Profile.
- In the Create Profile, enter the Name and select Enforcement Enable. If user wants to preserve their firewall rules, select Preserve Rules Enable. If user wants to allow broadcast or multicast traffic, select Allow Broadcast or Allow Multicast, respectively.
- Click on Save to create Agent Config Profile. The new profile will be listed under the Agent Config Profiles

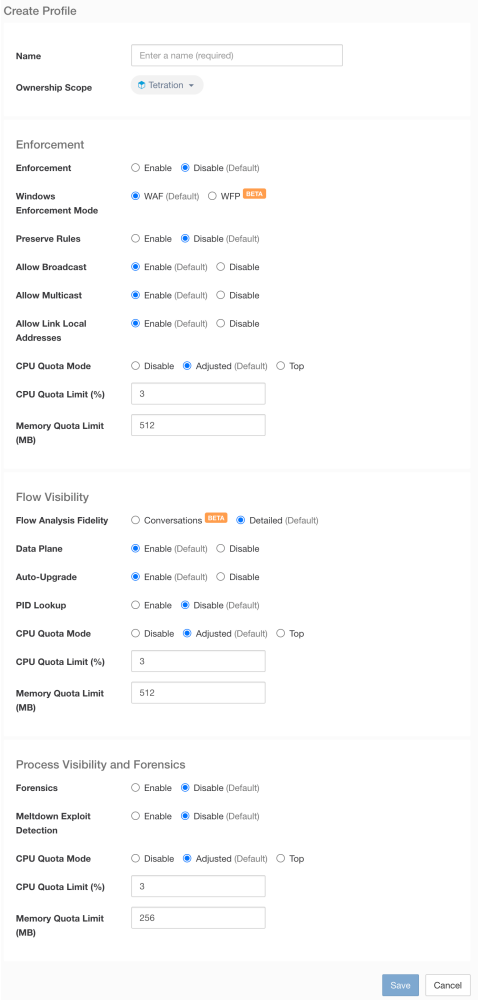


Fig. 3.7.1.2: Applying configuration profile to Agents

To configure Agent Config Intents:

- On the "Software Agent Config" page, click on "Create Intent".
- For "Apply Profile", enter profile listed under Agent Config Profiles and then select the filter.
- If filter is not already created, click on "Create new filter" to create a new filter. Enter Name, Description, Query and Scope.
- Click on Save and a new entry will be created under Agent Config Intents.

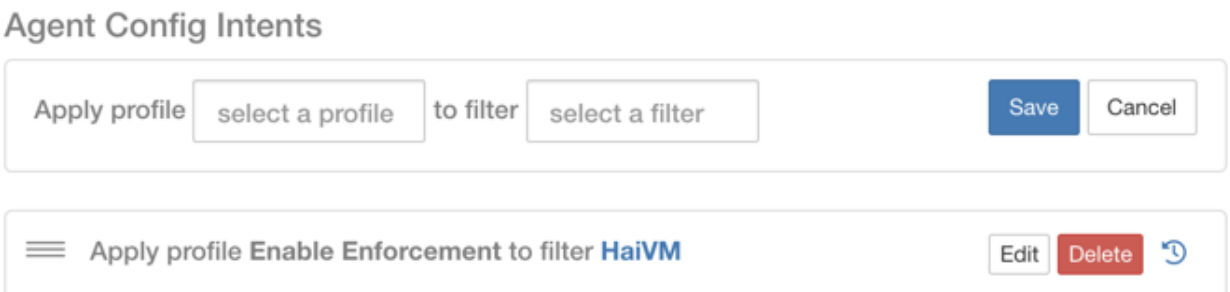


Fig. 3.7.1.3: Monitoring Agent status

Check Enforcement Agents

- On the top right corner, click on Monitoring.
- Click on Agents.
- On the Agents page, click on Enforcement Agents.
- On the Enforcement Agents page, you can check CPU Overhead, Bandwidth Overhead, Agent Health, Software Update Status, Agent Software Version Distribution, Agent OS Distribution.

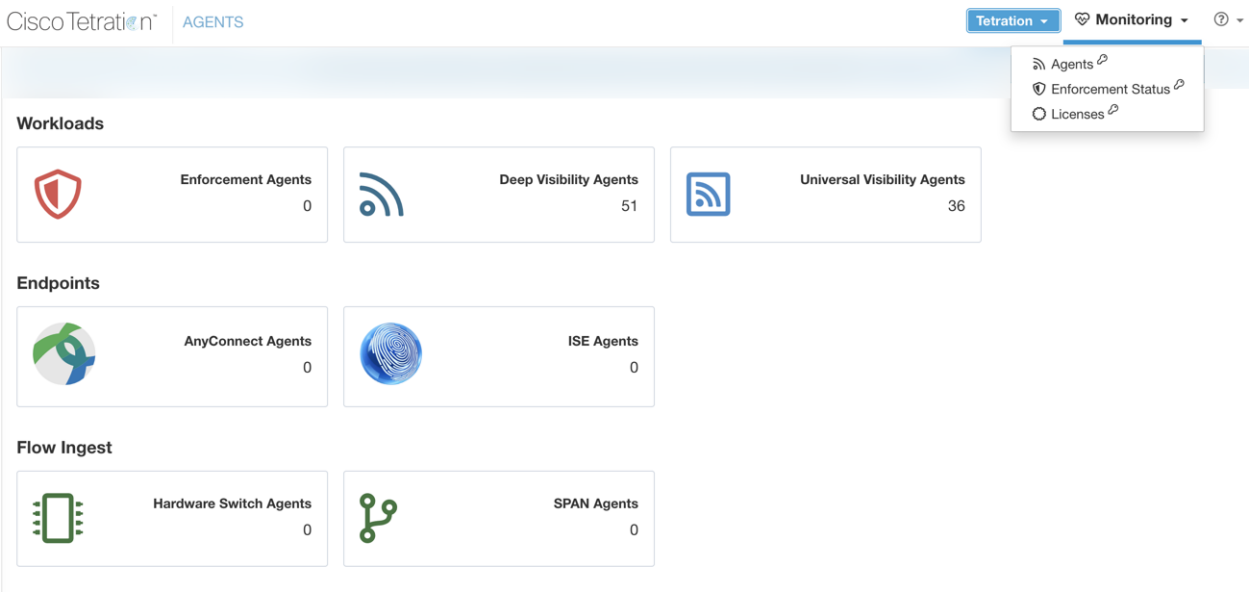


Fig. 3.7.1.4: Agents

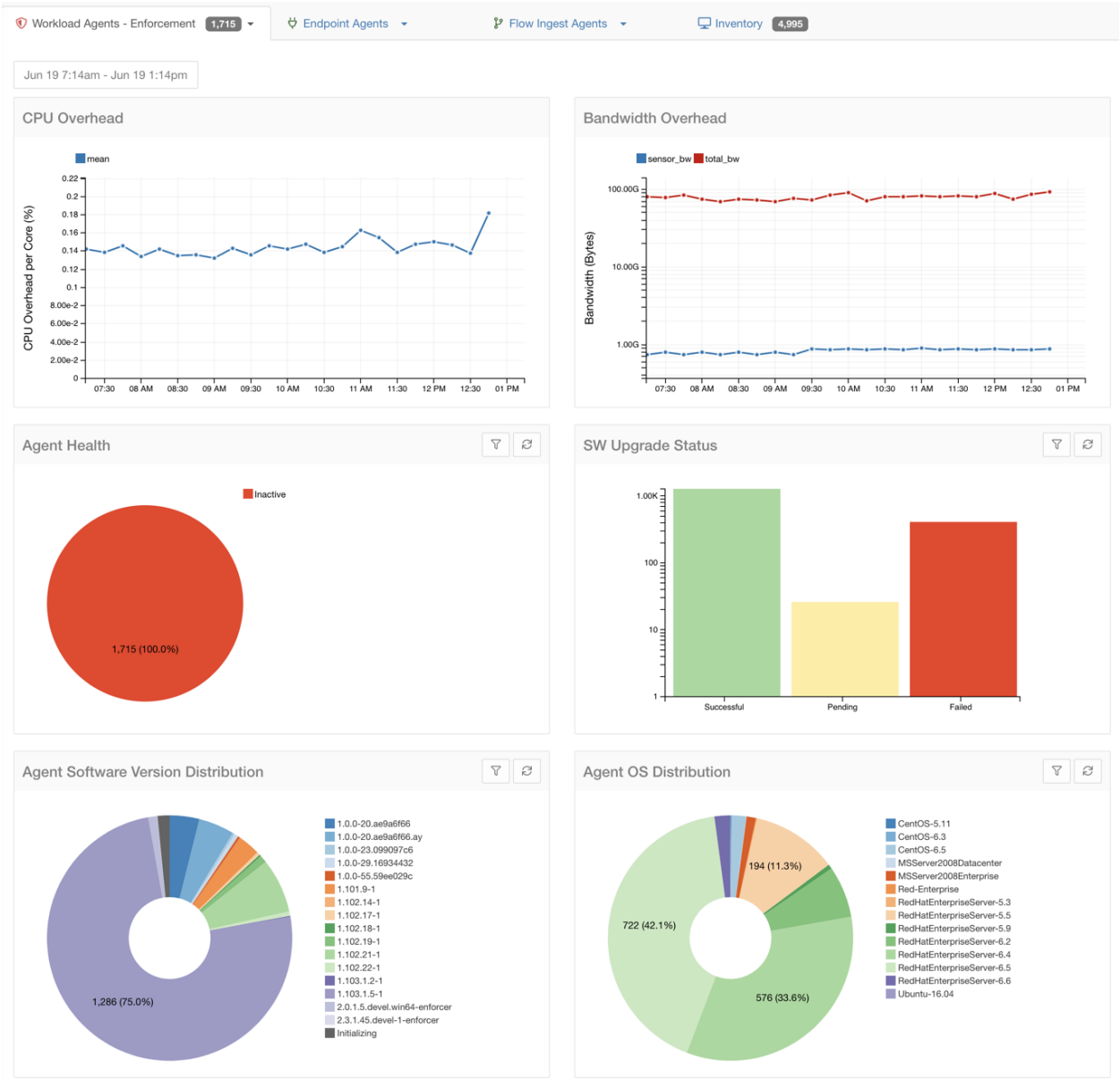


Fig. 3.7.1.5: Enforcement Agents

Check Enforcement Status

- On the top right corner, click on Monitoring.
- Click on Enforcement Status.
- On the Enforcement Agent Status page, you can check Agent Enfocement Enabled, Agent Policy Config and the list of Enforcement Agents.
- Click on one of the Enforcement Agent from the list to see the Agent details like IP address, Scopes, Inventory Type, Enforcement Groups, Experimental Groups, User Labels and Traffic Volume (Total Bytes/Total Packets). Click on IP address to view detailed Agent status as mentioned below.

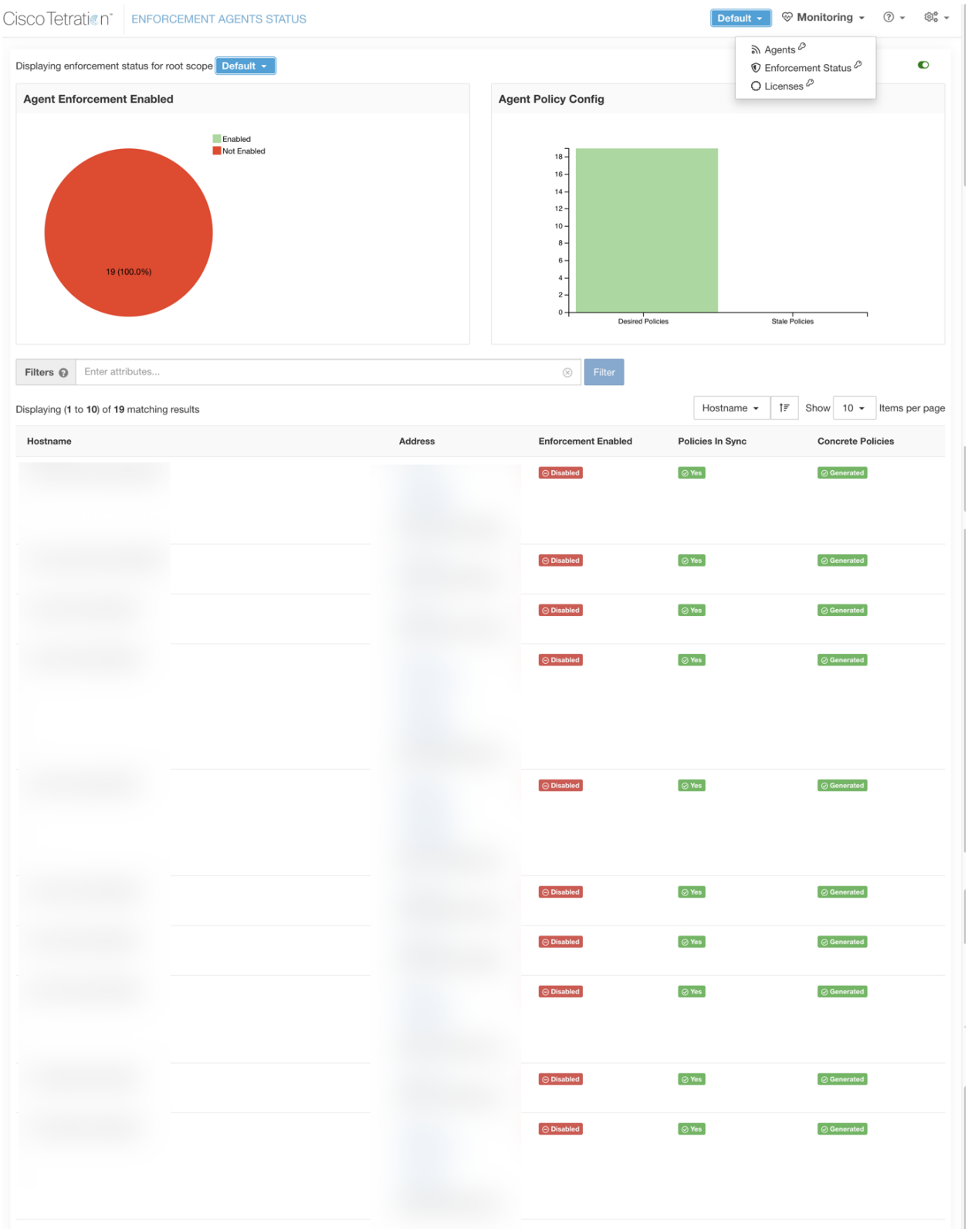


Fig. 3.7.1.6: Enforcement Status

Workload Profile (view detailed Agent status)

From the Monitoring section, follow steps to get to view detailed Agent status.

- On the Enforcement Agents page, click on Agent OS Distribution. Select an OS and click on filter image on the top right corner of the box.
- On the Software Agent List page, Agents with selected OS Distribution will be listed.
- Click on a Agent, Agent Details section will appear. Click on the IP address to go to Workload Profile page.
- On the Workload Profile page, Host Profile, Agent Profile and other Agent specific details like Bandwidth, Long-lived Processes, Packages, Process Snapshot, Configuration, Interfaces, Stats, Policies, Container Policies, etc can be seen.
- Click on Config tab to see the configuration on the endhost.
- Click on Policies tab to see the enforced policies on the endhost.

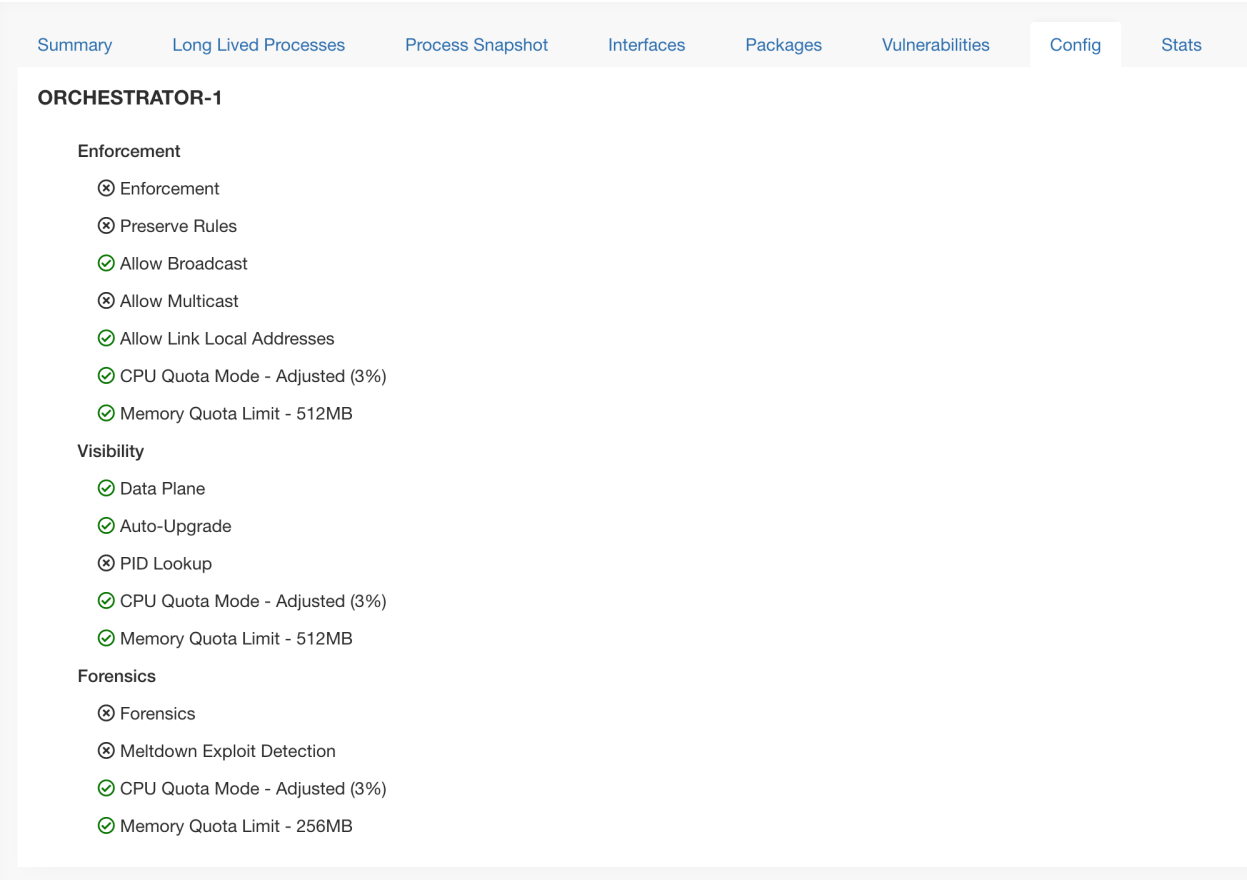
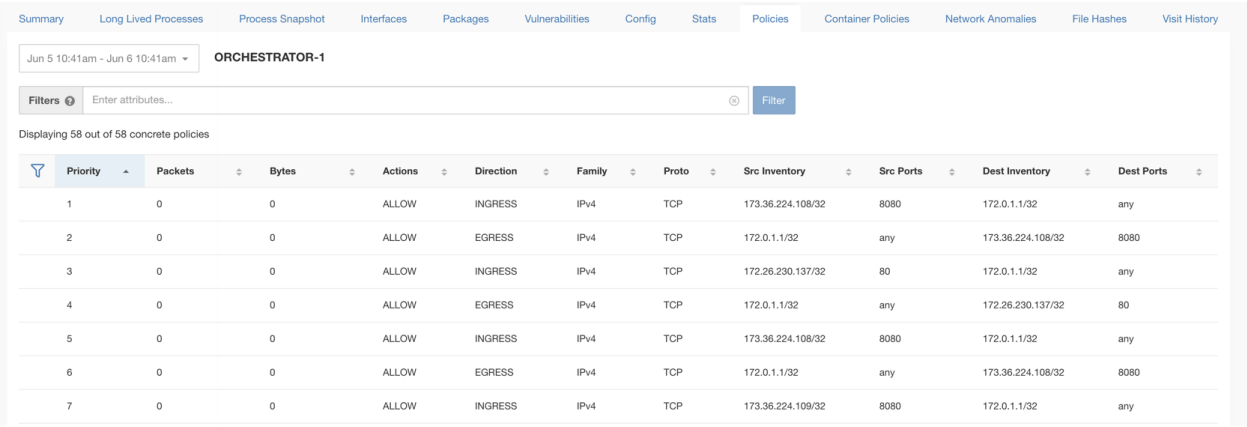


Fig. 3.7.1.7: Workload Profile - Config



Priority	Packets	Bytes	Actions	Direction	Family	Proto	Src Inventory	Src Ports	Dest Inventory	Dest Ports
1	0	0	ALLOW	INGRESS	IPv4	TCP	173.36.224.108/32	8080	172.0.1.1/32	any
2	0	0	ALLOW	EGRESS	IPv4	TCP	172.0.1.1/32	any	173.36.224.108/32	8080
3	0	0	ALLOW	INGRESS	IPv4	TCP	172.26.230.137/32	80	172.0.1.1/32	any
4	0	0	ALLOW	EGRESS	IPv4	TCP	172.0.1.1/32	any	172.26.230.137/32	80
5	0	0	ALLOW	INGRESS	IPv4	TCP	173.36.224.108/32	8080	172.0.1.1/32	any
6	0	0	ALLOW	EGRESS	IPv4	TCP	172.0.1.1/32	any	173.36.224.108/32	8080
7	0	0	ALLOW	INGRESS	IPv4	TCP	173.36.224.109/32	8080	172.0.1.1/32	any

Fig. 3.7.1.8: Workload Profile - Policies

Host IP Address change

Changing the IP address on the enforcement enabled hosts might have an impact if the host IP is seen in the host firewall rules and catch all is set to deny. In this scenario, following steps are recommended to change the host IP address:

1. On the Tetration UI, create a new Agent Config Profile with enforcement disabled.
2. Create Intent with list of hosts that need IP address change with their old and new IP address.
3. Apply the newly created Agent Config Profile to the Intent and save the Intent.
4. These select hosts should have enforcement disabled.
5. Change the IP address on these hosts.
6. On the Tetration UI, update the filters in the scope with the new IP address of these hosts.
7. Verify the IP address change from Agent Workload Profile page "Interfaces" tab. In the "Policies" tab, make sure policies are generated with new IP address.
8. Remove the Intent/Profile created above.
9. If the original Agent Config Profile for the scope had enforcement disabled, then enable enforcement.

3.7.2 Tetration Enforcement on the Linux Platform

On the Linux platform, the Tetration Enforcement Agent uses the iptables/ip6tables/ipset to enforce network policies. Once Enforcement Agent is enabled on the endhost, by default it controls and programs iptables. If IPv6 network stack is enabled then it controls the IPv6 firewall through ip6tables.

3.7.2.1 Linux iptables/ip6tables

Linux kernel has iptables and ip6tables which are used to set up, maintain and inspect the tables of IPv4 and IPv6 packet filter rules. It consists of different predefined tables. Each table contains predefined chains and can also contain user-defined chains. These chains contain set of rules and each of these rules specifies the match criteria for a packet. Predefined tables include raw, mangle, filter and nat. Predefined chains include INPUT, OUTPUT, FORWARD, PREROUTING and POSTROUTING.

Tetration Enforcement Agent programs filter table which contains rules to allow or drop packets. Filter table consists of predefine chains INPUT, OUTPUT and FORWARD. Along with these, Enforcement Agent adds custom TA chains to

categorize and manage the policies from controller. These TA chains contain Tetration rules derived from the policies along with rules generated by the Enforcement Agent. When Enfocement Agent receives platform independent rules, it parses and converts them into iptable/ip6table/ipset rules and inserts these rules into TA defined chains in the filter table. After programming the firewall, Enforement Agent monitors the firewall for any rule/policy deviation and if so, re-programs the firewall. It keeps track of the policies programmed in the firewall and reports their stats periodically to the controller. Here is an example to depict this behavior:

A typical policy in a platform independent network policy message consists of:

source set id: "test-set-1"
destination set id: "test-set-2"
source ports: 20-30
destination ports: 40-50
ip protocol: TCP
action: ALLOW
...
set_id: "test-set-1"
    ip_addr: 1.2.0.0
    prefix_length: 16
    address_family: IPv4
set_id: "test-set-2"
    ip_addr: 3.4.0.0
    prefix_length: 16
    address_family: IPv4

Along with other information. Enforcement Agent processes this policy and converts it into platform specific ipset and iptables rule:

```
ipset rule:
Name: ta_f7b05c30ffa338fc063081060bf3
Type: hash:net
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16784
References: 1
Members:
1.2.0.0/16

Name: ta_1b97bc50b3374829e11a3e020859
Type: hash:net
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16784
References: 1
Members:
3.4.0.0/16

iptables rule:
TA_INPUT -p tcp -m set --match-set ta_f7b05c30ffa338fc063081060bf3 src -m set --match-
↪set ta_1b97bc50b3374829e11a3e020859 dst -m multiport --sports 20:30 -m multiport --
↪dports 40:50 -j ACCEPT
```

3.7.2.2 Caveats

ipset kernel module

When Enforcement is enabled and Preserve Rules is disabled in the Agent Config Profile, the interested agents running on Linux hosts will make sure the ipset kernel module has a sufficiently large *max_sets* configuration. In case a change is needed, the enforcement agent reloads the ipset kernel module with a new *max_sets* value. If Preserve Rules is enabled instead, the enforcement agents will check the current ipset module *max_sets* value, but will not make any change. The current configured *max_sets* value can be found via *cat /sys/module/ip_set/parameters/max_sets*.

Host firewall backup

First time Enforcement is enabled in the Agent Config Profile, the interested agents running on Linux hosts, before taking control of the host's firewall, will store the current content of ipset and ip[6]tables in */opt/cisco/tetration/backup*. Successive disable/enable transitions of Enforcement configuration will not generate a new backup. The directory is not removed upon agent uninstallation.

3.7.3 Tetration Enforcement on the Windows Platform in WAF mode

On the Windows platform, the Tetration Enforcement Agent uses the Windows Firewall to enforce network policies.

3.7.3.1 Windows Firewall with Advanced Security

This is a native component on Windows that regulates network traffic based on the following types of settings:

- Firewall rules that regulate inbound network traffic
- Firewall rules that regulate outbound network traffic
- Firewall override rules based on authentication status of the source and destination of the network traffic
- Rules that apply to IPSec traffic and to Windows Services.

The Tetration Network Policy is programmed using Inbound and Outbound Firewall Rules.

3.7.3.2 Tetration Rules and the Windows Firewall

On the Windows platform, the Tetration Network Policy is enforced as follows:

1. Translate the platform-independent firewall rules from the Tetration Network Policy into Windows Firewall Rules.
2. Program the rules in the Windows Firewall.
3. The Windows Firewall enforces the rules.
4. Monitor the state of the Windows Firewall and its rule set: If a change is detected, report the deviation and reset the Tetration Network Policy in the Windows Firewall.

3.7.3.3 Security Profiles

Windows Firewall groups the rules based on the network the host is currently connected to. These are called Profiles and there are three such Profiles:

- Domain Profile
- Private Profile
- Public Profile

The Tetration rules are programmed into all the profiles, but only rules within active profiles are continuously monitored.

3.7.3.4 Effective Setting and Mixed-list Policies

The set of rules in the Windows Firewall is not ordered based on the precedence. When multiple rules match a packet, the most restrictive of those rules will take effect. That mean DENY rules take precedence over ALLOW rules. See this article on Microsoft TechNet for more details.

Consider the mixed-list (both allow and deny) policy example from the Enforcement Agent section:

```
1. ALLOW 1.2.3.30 tcp port 80
2. ALLOW 1.2.3.40 udp port 53
3. BLOCK 1.2.3.0/24 ip
4. ALLOW 1.2.0.0/16 ip
5. Catch-all: DROP ingress, ALLOW egress
```

When a packet headed for the host 1.2.3.30 tcp port 80 reaches the firewall, it matches all the rules above, but the most restrictive of them all—Rule 3—is the one that will be enforced and the packet will be dropped. This behavior is contrary to the expectation that the rules will be evaluated in order and Rule 1 will be the rule that is enforced and that the packet will be allowed.

This difference in behavior is expected in the Windows platform owing to the design of the Windows Firewall described above. This behavior can be observed in mixed-list policies with overlapping rules which have different rule actions. For example,

```
1. ALLOW 1.2.3.30 tcp
2. BLOCK 1.2.3.0/24 tcp
```

Interference from Other Firewalls or Policies

It is recommended to grant the Tetration Enforcement Agent full and exclusive control of the Windows Firewall in order to enforce the Tetration Network Policy as intended. The following conditions are incompatible with the reliable operation of the Enforcement Agent:

- Presence of a third party firewall. (The Windows Firewall is required to be the active firewall product on the host.)
- The Firewall is disabled for the current profiles.
- Conflicting firewall settings are deployed using Group Policy. Some of the conflicting settings are:
  - Firewall rules
  - Default inbound or outbound actions in the current profiles that differ from the catch-all rule of the policy.
  - Firewall disabled for the current profiles

3.7.3.5 Stateful enforcement

Windows Advanced Firewall is considered a **stateful** firewall, i.e. for certain protocols (such as TCP), the firewall maintains internal state tracking to detect if a new packet hitting firewall belongs to a known connection. Packets belonging to a known connection will be allowed without needing to examine firewall rules. This enables bidirectional communication without having to establish rules in both INBOUND and OUTBOUND tables.

For example, consider the following rule for a web server: **Accept all TCP connections to port 443**

The intention is clear: we want to accept all TCP connections on port 443 to the server, and allow the server to communication back to the clients. In this case, we will only insert one rule in the INBOUND table, allowing TCP connections on port 443. There won't be any rule required to be inserted in OUTBOUND table, because this is implicitly done by the Windows Advanced Firewall.

Note that the state tracking is only applicable to some protocols in which explicit **connections** are established and maintained. For other protocols, both INBOUND and OUTBOUND rules must be programmed to enable bidirectional communication.

Within Tetration enforcement context, a given concrete rule will be programmed as **stateful** when the protocol is TCP (agent will decide based on the context to insert the rule in either INBOUND or OUTBOUND table). For other protocols (including **ANY**), both INBOUND and OUTBOUND rules will be programmed.

3.7.3.6 Caveats

Host firewall backup

First time Enforcement is enabled in the Agent Config Profile, the interested agents running on Windows hosts, before taking control of the host's firewall, will export the current Windows Advanced Firewall content to *WindowsSystem32configsystemprofileAppDataRoamingtetbackup*. Successive disable/enable transitions of Enforcement configuration will not generate a new backup. The directory is not removed upon agent uninstallation.

3.7.4 Tetration Enforcement on the Windows Platform in WFP mode

On the Windows platform, Tetration Agent enforces the network policies programming WFP filters. Windows Advanced Firewall is not used to configure the network policy.

3.7.4.1 WFP(Windows Filtering Platform)

WFP, Windows Filtering Platform, is a set of APIs provided by Microsoft to configure filters for processing network traffic. Network traffic processing filters can be configured using kernel level APIs as well as User level APIs. WFP filters can be configured at various layers, Network Layer, Transport Layer, Application Layer Enforcement(ALE). Tetration WFP filters are configured at ALE layer, similar to Windows firewall rules. Each layer has a number of sublayers, ordered by weight, highest to lowest. Within each sublayer, filters are ordered by weight, highest to lowest. Network packet traverses through all the sublayers. At each sublayer, network packet traverses through the matching filters, based on weight highest to lowest and returns the action, Permit or Block. After passing through all the sublayers, packet is processed based on the action. Block action overrides Permit.

3.7.4.2 Tetration Agent WFP support

When enforcement mode is WFP, Tetration filters overrides Windows Firewall rules.

In WFP mode, Tetration Agent configures various WFP objects

- Provider - It is used for filter management. It does not affect packet filtering. It has GUID and name.
- Sublayer - Sublayer has name, guid and weight. Tetration sublayer is cofigured with the weight greater than Windows Advanced Firewall sublayer.
- Filters - Filter has name, guid, id, weight, layer id, sublayer key, action (PERMIT/BLOCK), and filter conditions. WFP filters are configured for Golden Rules, Self Rules, Policy Rules. Tetration agent also configures Port scanning prevention filters. Tetration Filters are configured with the flag, FWPM_FILTER_FLAG_CLEAR_ACTION_RIGHT. Because of this flag, Tetration Filter action cannot be overridden by Microsoft Firewall rules. For each Tetration Network policy rule, one or more WFP filters are configured based on the direction (inbound/outbound) and protocol.

For TCP inbound policy,

```
id: 14 , TCP Allow 10.195.210.184 Dir=In localport=3389
```

WFP Filters Configured

```
Filter Name:                            Tetration Rule 14
-------------------------------------------------------
EffectiveWeight:                        18446744073709551589
LayerKey:                               FWPM_LAYER_ALE_AUTH_LISTEN_V4
Action:                                 Permit
Local Port:                             3389

Filter Name:                            Tetration Rule 14
-------------------------------------------------------
EffectiveWeight:                        18446744073709551589
LayerKey:                               FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4
Action:                                 Permit
RemoteIP:                               10.195.210.184-10.195.210.184
```

Tetration agent configures **Tetration Default Inbound** filter for inbound CATCH-ALL policy. Tetration agent configures **Tetration Default Outbound** filter for outbound CATCH-ALL policy.

3.7.4.3 Tetration Agent WFP support and Windows Firewall

- Tetration Enforcement Agent **does not monitor** WAF rules or WAF profiles.
- Tetration Enforcement Agent **does not monitor** firewall states.
- Tetration Enforcement Agent **does not require** firewall state to be enabled.
- Tetration Enforcement Agent **does not conflict** with GPO policies.

3.7.4.4 Effective Setting and Mixed-list Policies

Tetraion Agent enforcement in WFP mode supports mixed-list or grey list policies. Consider the mixed-list (both allow and deny) policy example from the Enforcement Agent section:

```
1. ALLOW 1.2.3.30 tcp port 80            - wt 1000
2. BLOCK 1.2.3.0/24 ip                   - wt 998
3. ALLOW 1.2.0.0/16 ip                   - wt 997
4. Catch-all: DROP ingress, ALLOW egress - wt 996
```

When a packet headed for the host 1.2.3.30 tcp port 80 reaches the firewall, it matches rule 1. But a packet headed for the host 1.2.3.10 will be blocked because of filter 2. Packet headed for host 1.2.2.10 will be allowed by filter 3.

3.7.4.5 Stateful enforcement

Tetration WFP filters are configured at ALE layer. Network traffic is filtered for socket connect(), listen() and accept() operations. Network packets related to a L4 connection are no longer filtered once the connection is established.

3.7.4.6 Visibility of Confiured WFP filters

The configured Tetration WFP filters can be viewed using c:\program files\tetration\tetenf.exe. Supported options are

- Run 'cmd.exe' using 'Admin' privileges.
- Run c:\program files\tetration\tetenf.exe -l -f <-verbose> <-output=outfile.txt>

OR

- Run 'cmd.exe' using 'Admin' privileges.
- Run netsh wfp show filters
- Check filters.xml for configured Tetration filters

3.7.4.7 Delete Configured WFP filters

The configured Tetration WFP filters can be deleted using c:\program files\tetration\tetenf.exe. To avoid accidental deletions of filters, user needs to specify **token** in <yyyymm> format, when excuting the delete command, where yyyy is the current year and mm is the current month in the numerical form. e.g. if today's date 01/21/2021, token will be **-token=202101**

Supported options are

- Run 'cmd.exe' using 'Admin' privileges.
- To delete all Tetration filters configured Run c:\program files\tetration\tetenf.exe -d -f -all -token=<yyyymm>
- To delete all Tetration WFP objects configured Run c:\program files\tetration\tetenf.exe -d -all -token=<yyyymm>
- To delete a Tetration WFP filter by name Run c:\program files\tetration\tetenf.exe -d -name=<WFP filter name> -token=<yyyymm>

3.7.4.8 Known limitations

- "Preserve Rules" setting in Agent Config Profile has no effect when Enforcement Mode is set to WFP.

3.7.5 Tetration Enforcement on the AIX Platform

On the AIX platform, the Tetration Enforcement Agent uses IPFilter utilities to enforce network policies. Once Enforcement Agent is enabled on the endhost, by default it controls and programs the IPv4 filter table. IPv6 enforcement is not supported.

3.7.5.1 IPFilter

IPFilter package on AIX is used to provide firewall services. It is available on AIX as kernel expansion pack. It loads as kernel extension module, /usr/lib/drivers/ipf. It includes ipf, ippool, ipfstat, ipmon, ipfs and ipnat utilities that are used to program ipfilter rules and each of these rules specifies the match criteria for a packet. Please refer to IPFilter man pages on AIX for more details.

Tetration Enforcement Agent uses IPFilter to program the IPv4 filter table which contains rules to allow or drop IPv4 packets. Enforcement Agent groups these rules to categorize and manage the policies from controller. These rules include Tetration rules derived from the policies along with rules generated by the Enforcement Agent.

When Enfocement Agent receives platform independent rules, it parses and converts them into ipfilter/ippool rules and inserts these rules into filter table. After programming the firewall, Enforement Agent monitors the firewall for any rule/policy deviation and if so, re-programs the firewall. It keeps track of the policies programmed in the firewall and reports their status periodically to the controller.

A typical policy in a platform independent network policy message consists of:

source set id: "test-set-1"
destination set id: "test-set-2"

source ports: 20-30
destination ports: 40-50
ip protocol: UDP
action: ALLOW
...
set_id: "test-set-1"
ip_addr: 1.2.0.0
prefix_length: 16
address_family: IPv4
set_id: "test-set-2"
ip_addr: 5.6.0.0
prefix_length: 16
address_family: IPv4

Along with other information. Enforcement Agent processes this policy and converts it into platform specific ippool and ipfilter rule:

```
table role = ipf type = tree number = 51400
{ 1.2.0.0/16; };

table role = ipf type = tree number = 75966
{ 5.6.0.0/16; };

pass in quick proto udp from pool/51400 port 20:30 to pool/75966 port 40:50 group TA_
↪INPUT
```

3.7.5.2 Caveats

Host firewall backup

First time Enforcement is enabled in the Agent Config Profile, the interested agents running on AIX hosts, before taking control of the host's firewall, will store the current content of ippool and ipfilter into */opt/cisco/tetration/backup*. Successive disable/enable transitions of Enforcement configuration will not generate a new backup. The directory is not removed upon agent uninstallation.

3.7.5.3 Known limitations

IPv6 enforcement is not supported.

Allow policy might cause traffic disruption for existing UDP connections.

3.8 Tetration Secure Connector

In order for Tetration to import user tags or enforce policies on external orchestrators (see *External Orchestrators*), Tetration needs to establish outgoing connections to the orchestrator API servers (Vcenter, Kubernetes, F5 BIG-IP, etc.). Sometimes it is not possible to allow direct incoming connections to the orchestrators from the Tetration cluster. Secure Connector solves this issue by establishing an outgoing connection from the same network as the orchestrator to the Tetration cluster. This connection is used as a reverse tunnel to pass requests from the cluster back to the orchestrator API server.

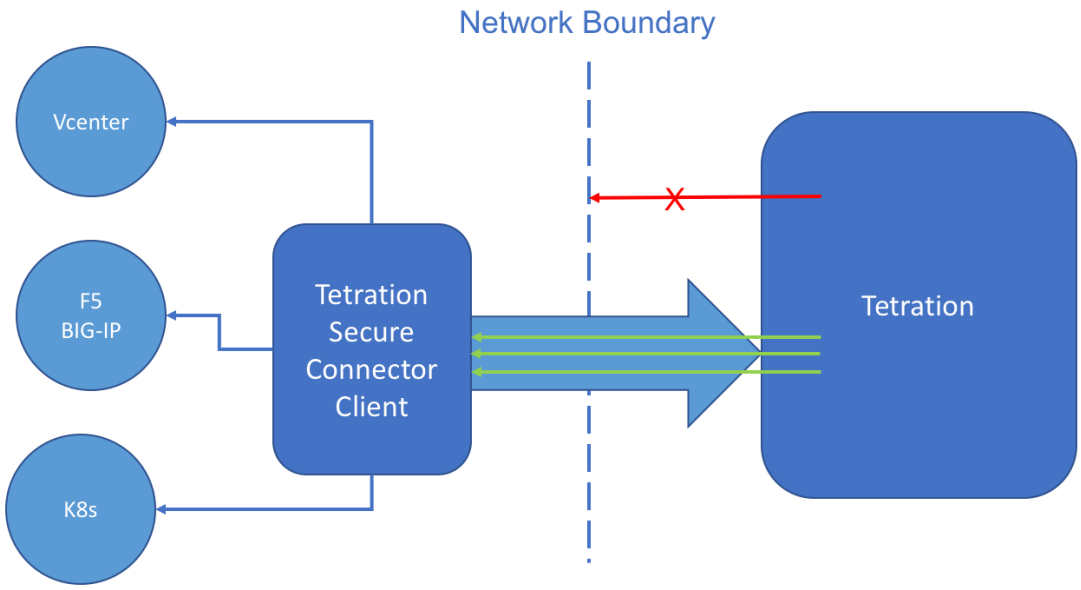


Fig. 3.8.1: Tetration Secure Connector

For each root scope, only one tunnel may be active at any time. Attempts to start additional tunnels will be rejected with an error message indicating that one is already active. The active tunnel can be used to connect to multiple orchestrators that are reachable from the network in which the client is running. A per-orchestrator configuration is used to indicate whether connections to that orchestrator should go through the Secure Connector tunnel.

All communication between the Secure Connector client and the Tetration cluster is mutually authenticated and encrypted using TLS.

For improved security, customers are advised to install the Secure Connector client on an isolated appropriately-secured machine. The machine should have firewall rules to allow outgoing connections only to the Tetration cluster and any external orchestrator API servers Tetration should be allowed to access.

For more details on installing and starting the Secure Connector client, see *Deploying the Secure Connector Client*

For more details on configuring orchestrators to use the Secure Connector tunnel, see *Create an orchestrator configuration*.

For more details on OpenAPI endpoints for the Secure connector, see *Secure Connector API endpoints*

3.8.1 Technical details

To bootstrap the tunnel, the Secure Connector client creates a public/private key pair and signs its public key certificate remotely by the server. A cryptograhic single-use time-limited token is used to secure this remote signing process and identify the root scope to which the client belongs. On the server side, each root scope has a unique certificate that the client uses to authenticate the server. These certificates are periodically rotated to ensure the continued secrecy of communication.

The Secure Connector client is internally constructed of a tunnel client and a SOCKS5 server. After the tunnel is started, the client waits for incoming tunneled connections from the Tetration Cluster. Incoming connections are handled by the SOCKS5 server and forwarded to the destination host.

Successful operation of the Secure Connector client requires:

- RHEL/CentOS 7 (x86_64)
- 2 CPU cores and 4 GB RAM
- Enough network bandwidth for handling data from the on-prem orchestrators that will use the Secure Connector
- Outgoing connectivity to the Tetration cluster on port 443 (direct or through HTTP(S) proxy)
- Outgoing connectivity to internal Orchestrator API servers (direct)

3.9 Enforcement Alerts

Note: Starting 3.5 release, Enforcement Alerts can be configured using the *Alert Configuration Model*.

Enforcement Alerts can be configured using the *Alert Configuration Model*. See *Alert Configuration Modal* for general information about the model

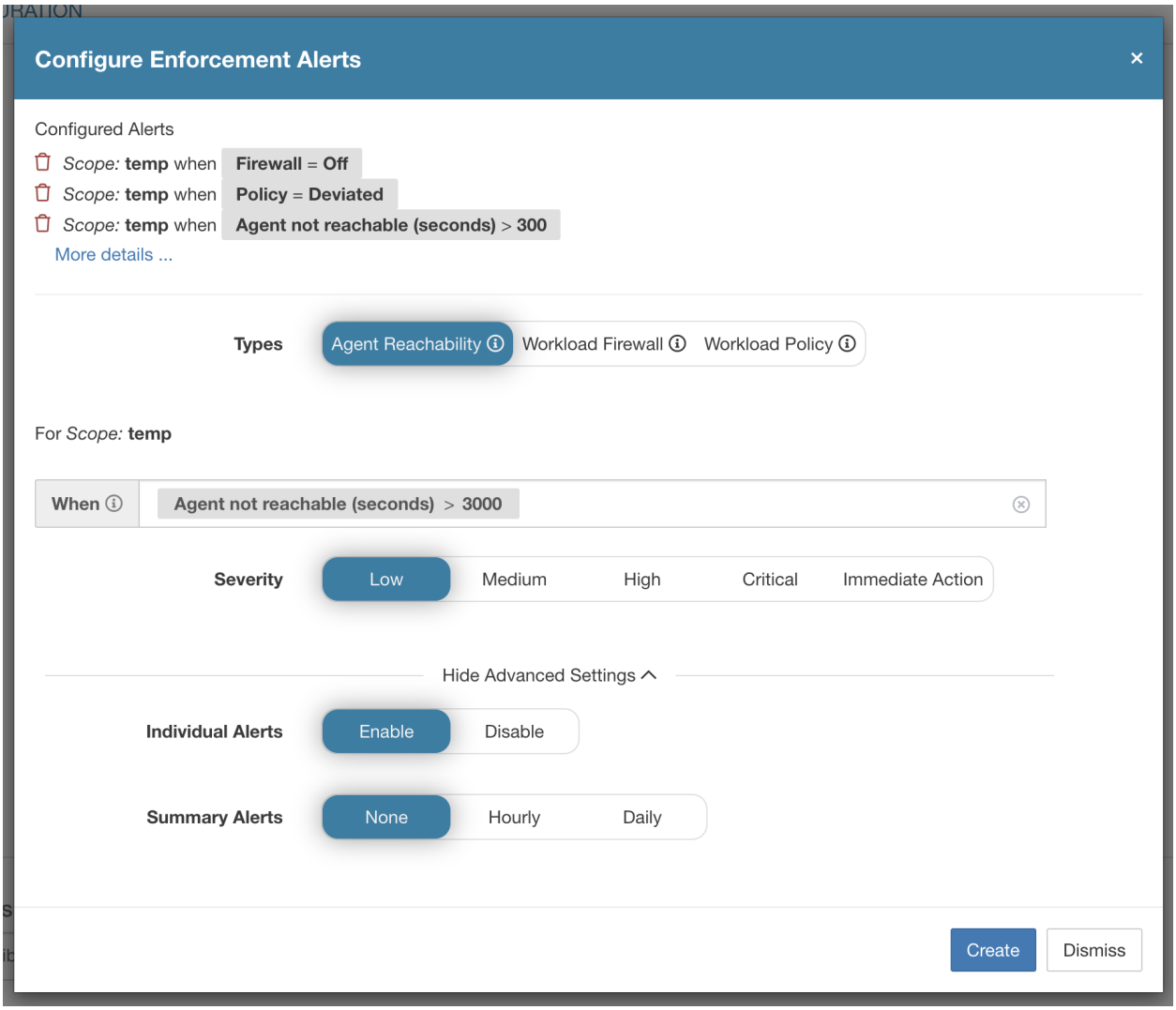


Fig. 3.9.1: Configuring Enforcement alerts.

Enforcement Alert Configuration provides the ability to configure three different types of alerts, allowing the user to set the Severity of the alert as well as other per-type configuration parameters:

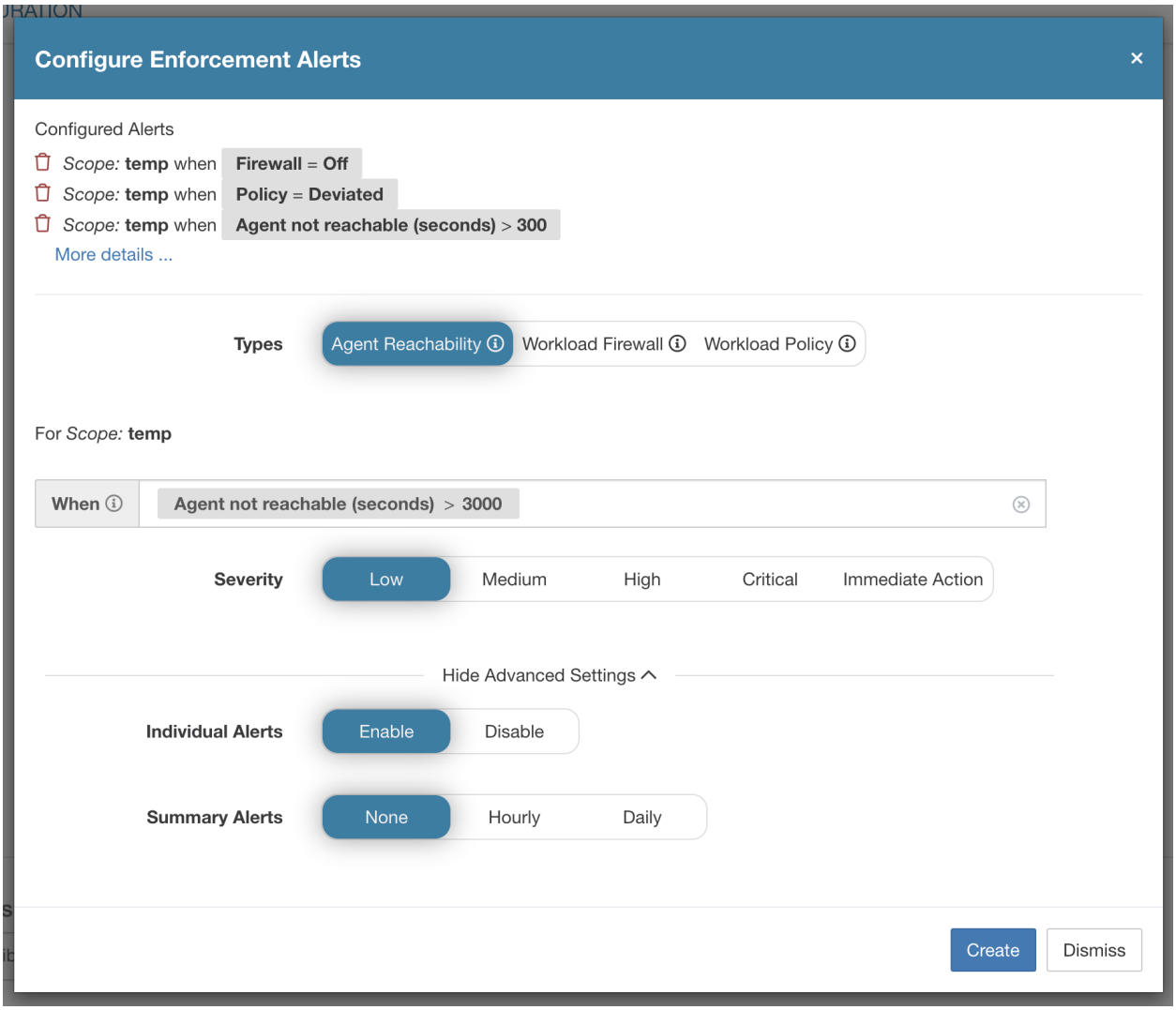


Fig. 3.9.2: Configuring Enforcement alerts when Enforcement Agent is not reachable. This alert will trigger if the enforcement agent has not communicated with the Tetration cluster for more than the configured number of seconds.

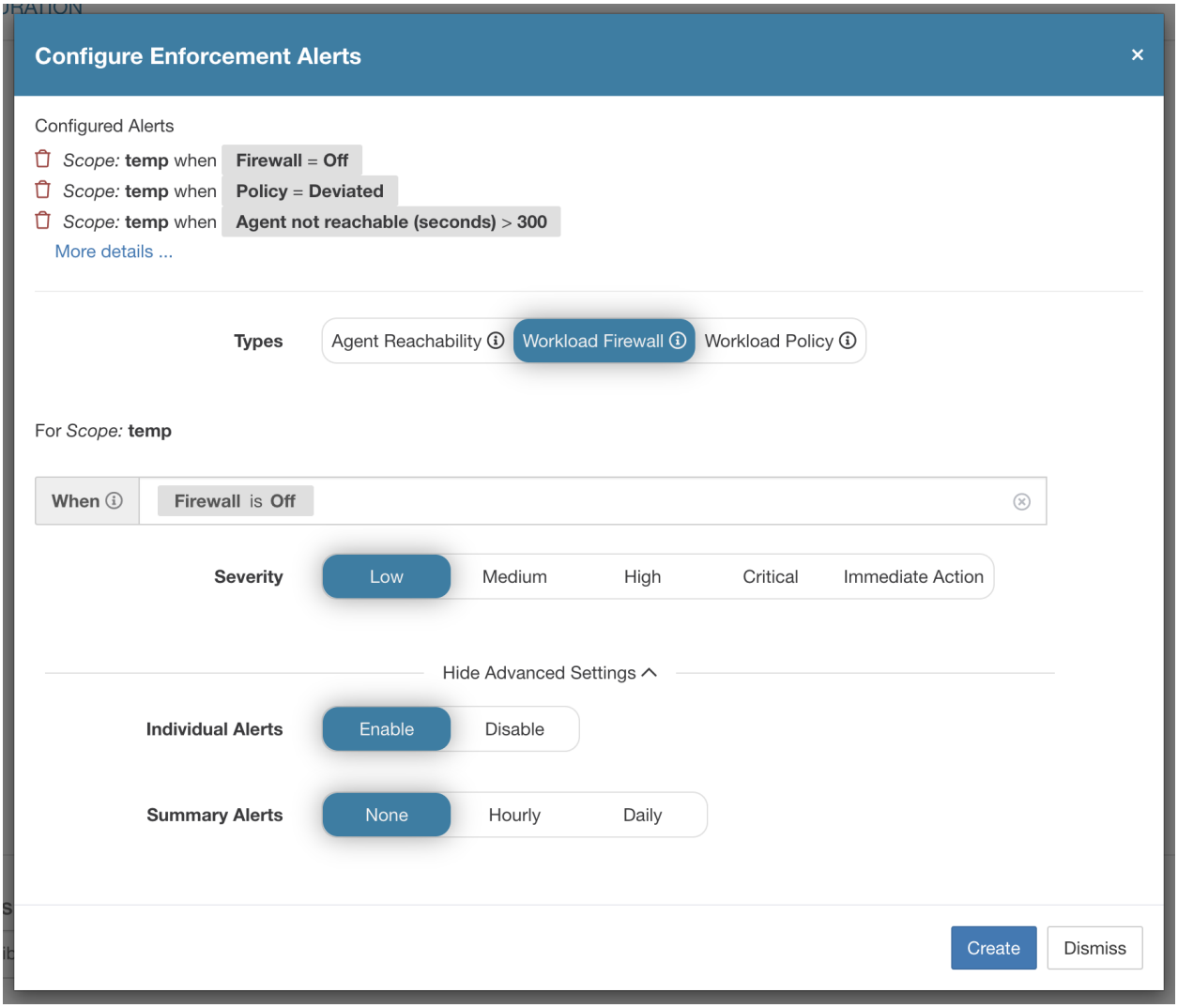


Fig. 3.9.3: Configuring Enforcement alerts to detect when the Workload firewall is off. This alert will trigger if enforcement is configured on a workload but the workload Firewall is detected to be off, since this condition will prevent Tetration Agent from enforcing traffic policies.

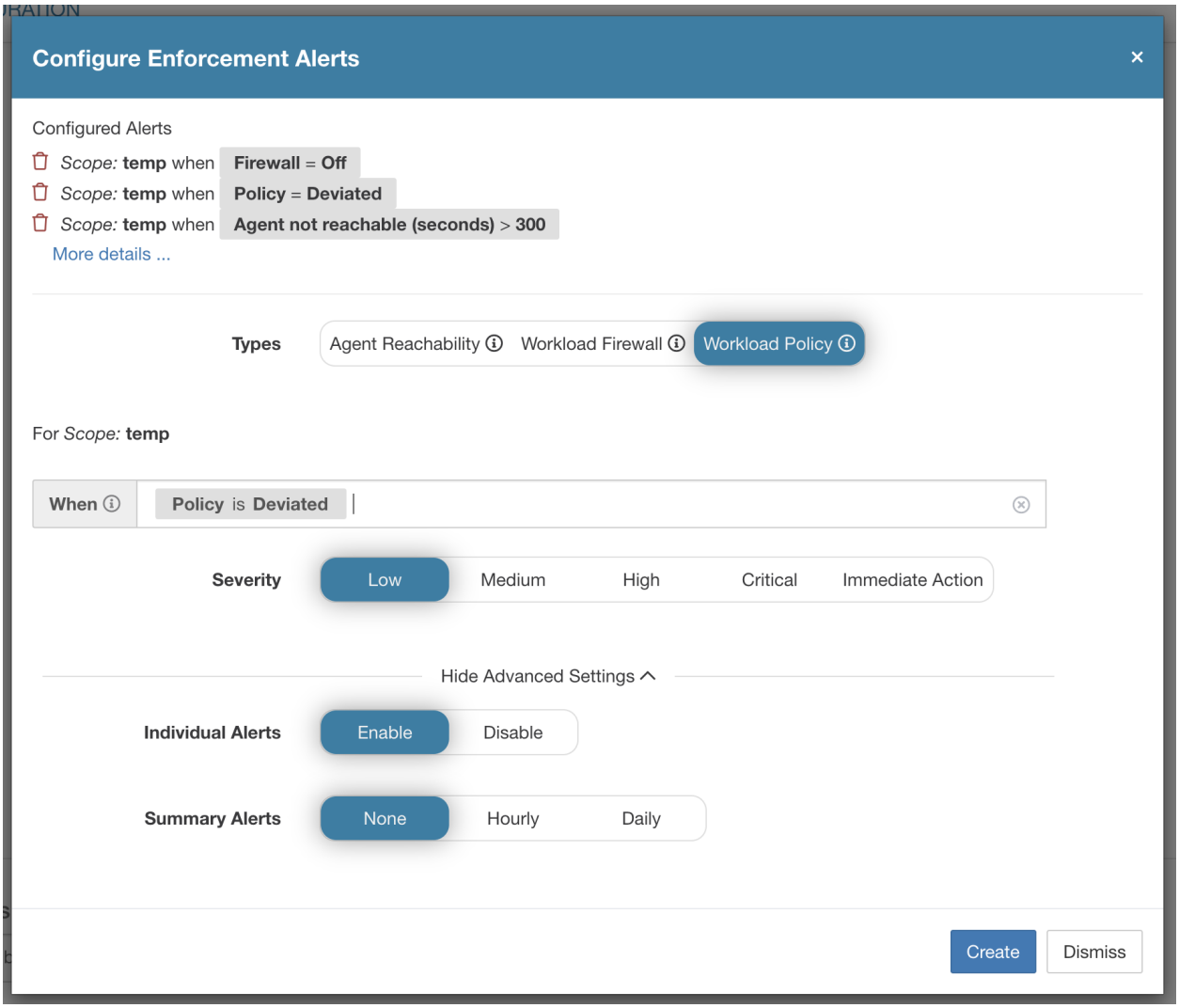


Fig. 3.9.4: Configuring Enforcement alerts when Workload policies are deviated. This alert will trigger if the workload firewall rules are deviated.

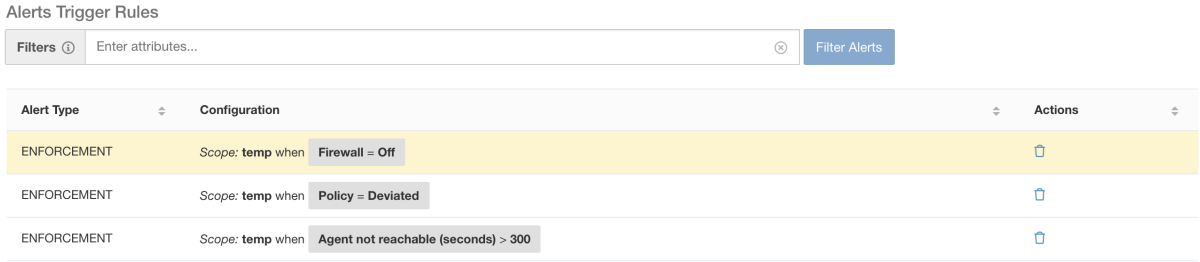


Fig. 3.9.5: Viewing configured Enforcement Alerts on the alerts configuration page.

3.9.1 Enforcement UI Alerts Details

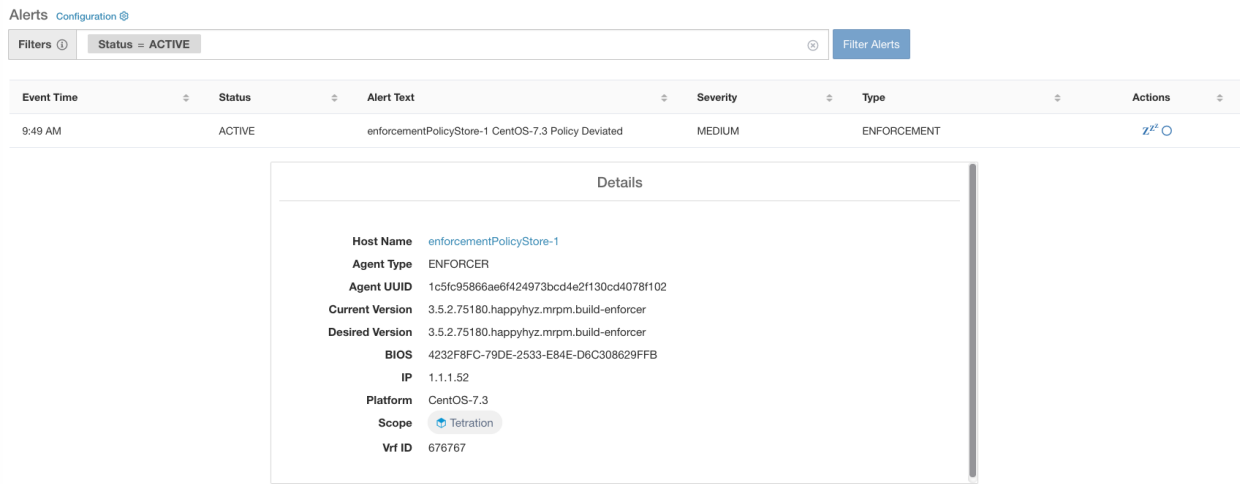


Fig. 3.9.1.1: Enforcement alert details.

3.9.2 Enforcement Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* field is structured and contains the following subfields for enforcement alerts

Field	Alert Type	Format	Explanation
AgentType	*all*	string	"ENFORCER" or "SENSOR" depending on the installed type
HostName	*all*	string	Host name on which the agent is deployed
IP	*all*	string	IP address of the node
Bios	*all*	string	BIOS UUID of the node
Platform	*all*	string	Platform/OS information of the node
CurrentVersion	*all*	string	Software version of the agent on the node
DesiredVersion	*all*	string	Software version desired for the agent
LastConfigFetchAt	*all*	integer	Unix timestamp of when the agent last sent https request

3.9.2.1 Example of alert_details for an enforcement alert

```
{
  "AgentType":"ENFORCER",
  "Bios":"72EF1142-03A2-03BC-C2F8-F600567BA320",
  "CurrentVersion":"3.5.1.1.mrpm.build.win64-enforcer",
  "DesiredVersion":"",
  "HostName":"win2k12-production-db",
  "IP":"172.26.231.193",
  "Platform":"MSServer2012R2Standard"
}
```

3.10 Sensor Alerts

Note: Starting 3.5 release, Sensor Alerts can be configured using the *Alert Configuration Model*.

Sensor Alerts can be configured using the *Alert Configuration Model*. See *Alert Configuration Modal* for general information about the model

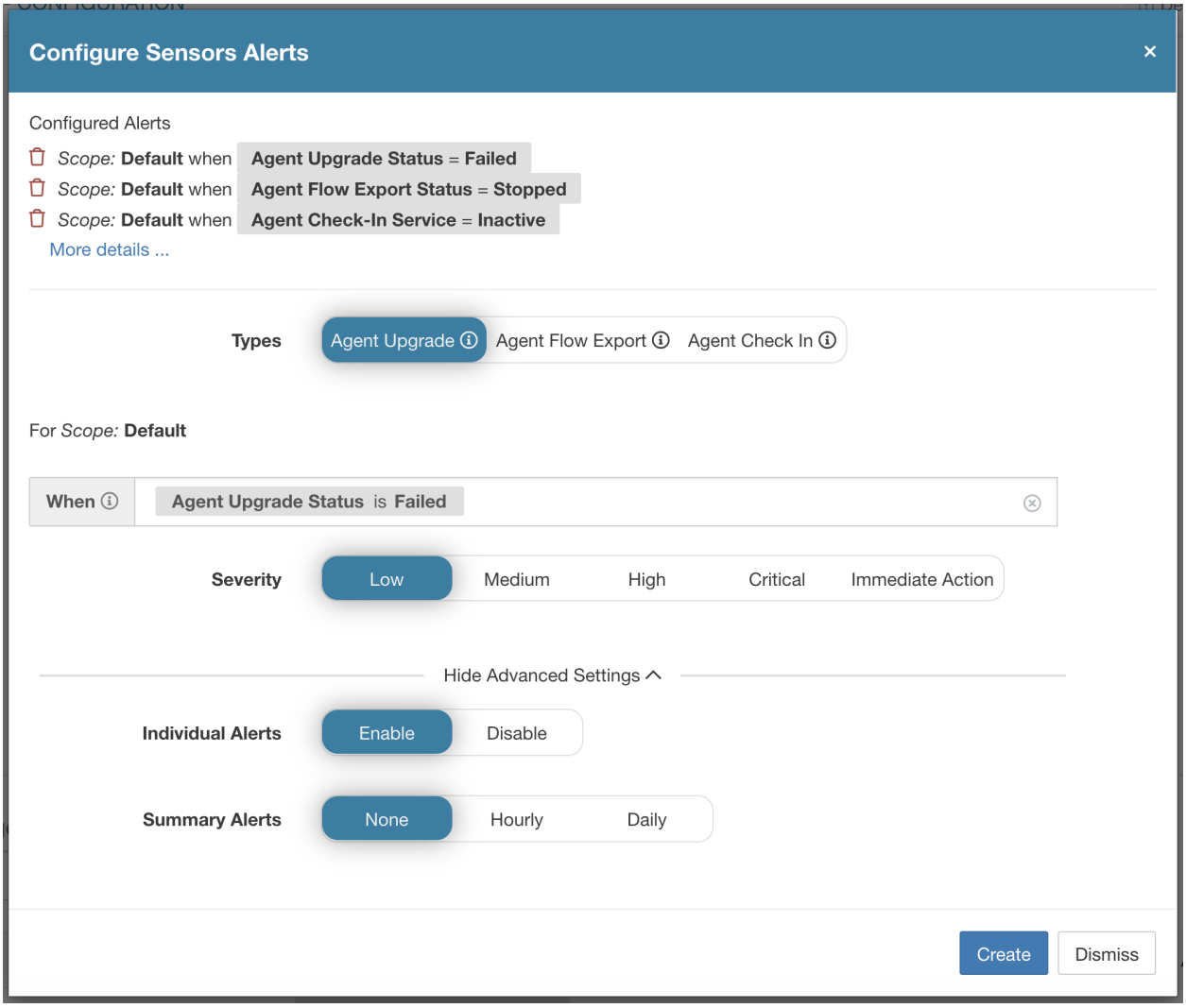


Fig. 3.10.1: Configuring Sensor alerts.

Sensor Alert Configuration provides the ability to configure three different types of alerts, allowing the user to set the Severity of the alert as well as other per-type configuration parameters:

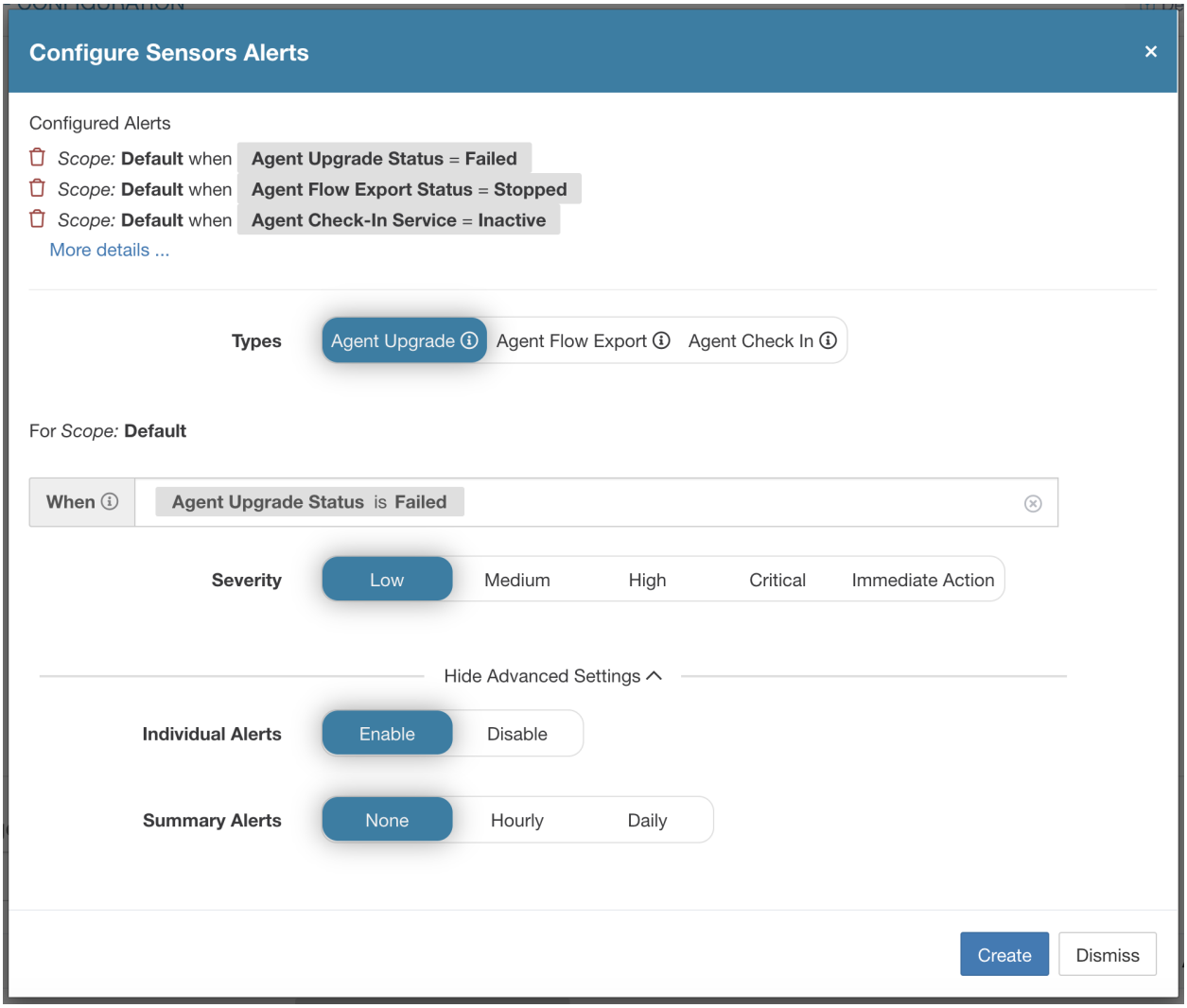


Fig. 3.10.2: Configuring Sensor alerts to report when agent failed to upgrade. This alert will trigger if agent failed to upgrade to the desired version.

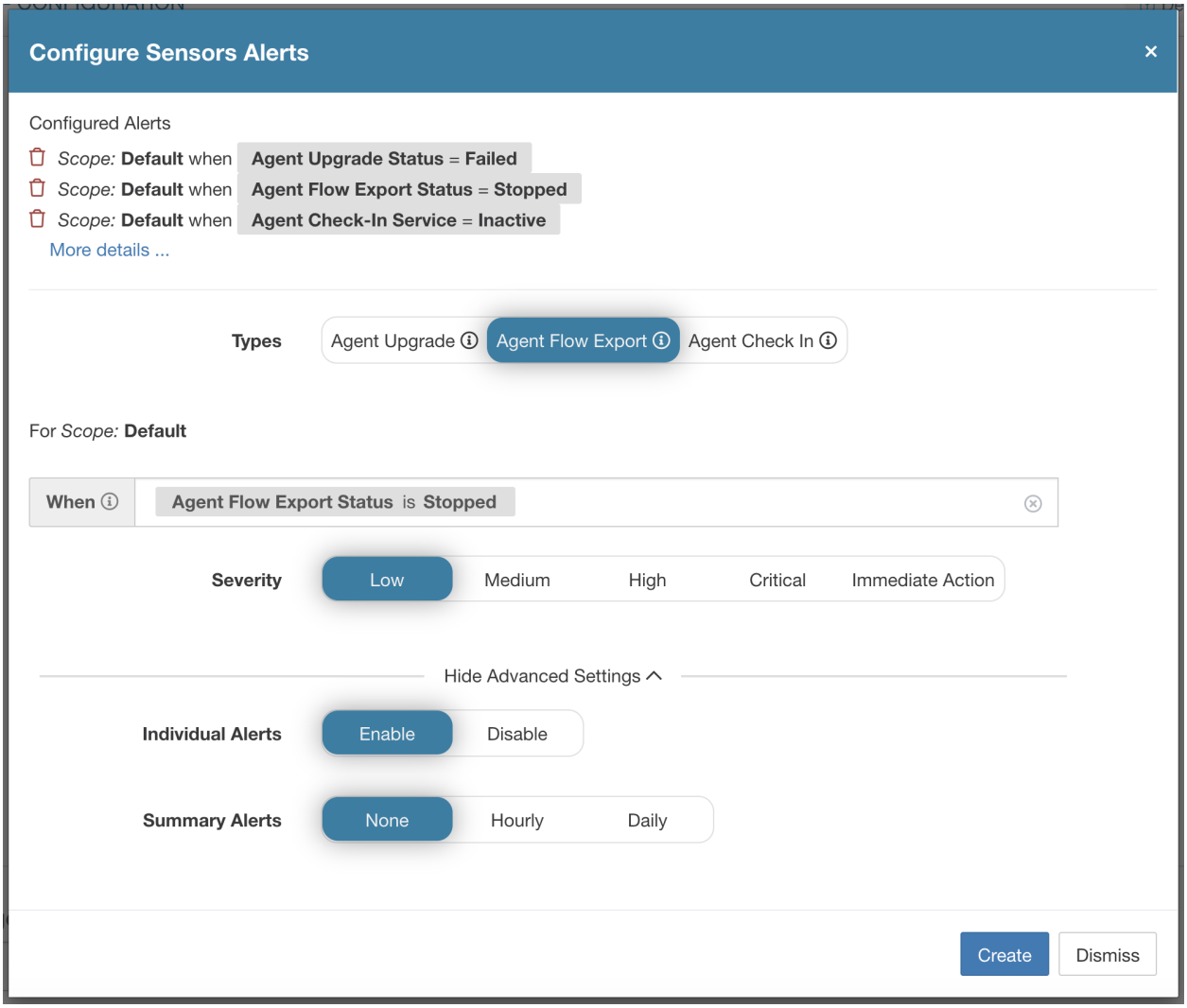


Fig. 3.10.3: Configuring Sensor alerts to detect when agent flow export has stopped. This alert will trigger if connectivity between the agent and the cluster is somewhere being blocked, therefore preventing flows and other system information from being sent or delivered.

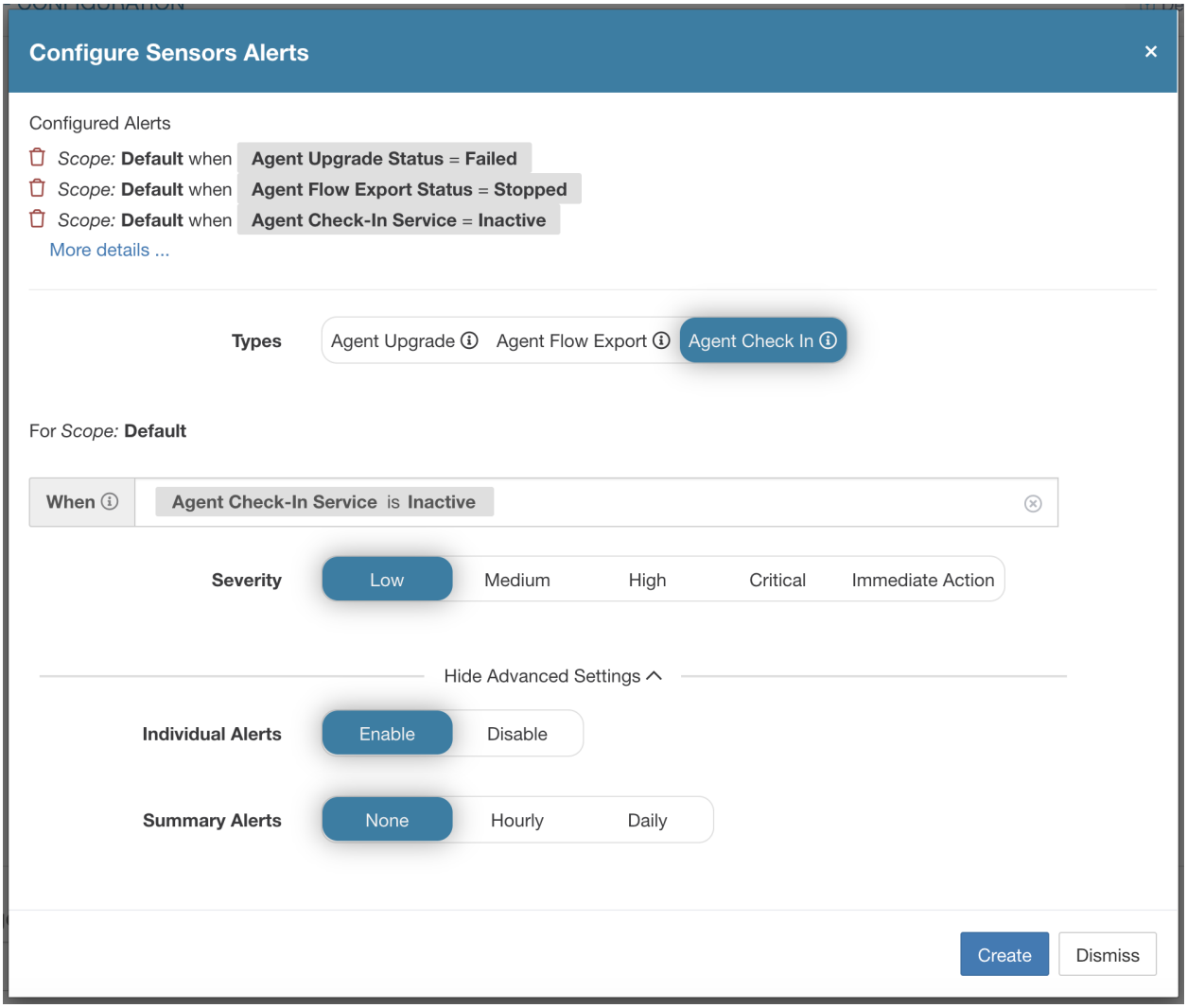


Fig. 3.10.4: Configuring Sensor alerts to detect when agent check_in has timed out. This alert will trigger if the cluster has not received a check-in request from an agent for more than 90 minutes.

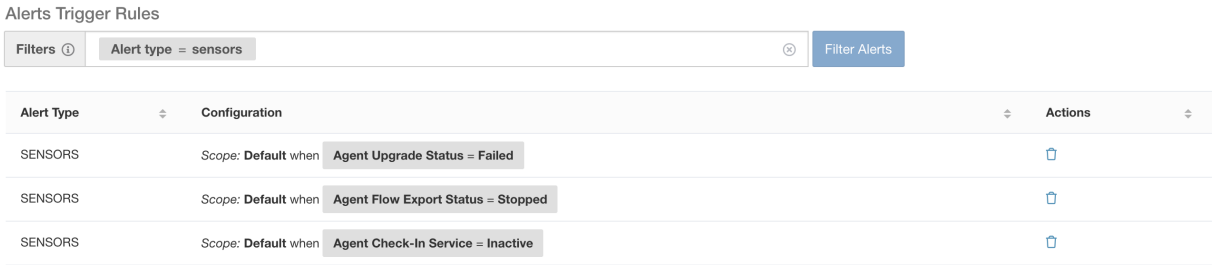


Fig. 3.10.5: Viewing configured Sensor Alerts on the alerts configuration page.

3.10.1 Sensor UI Alerts Details

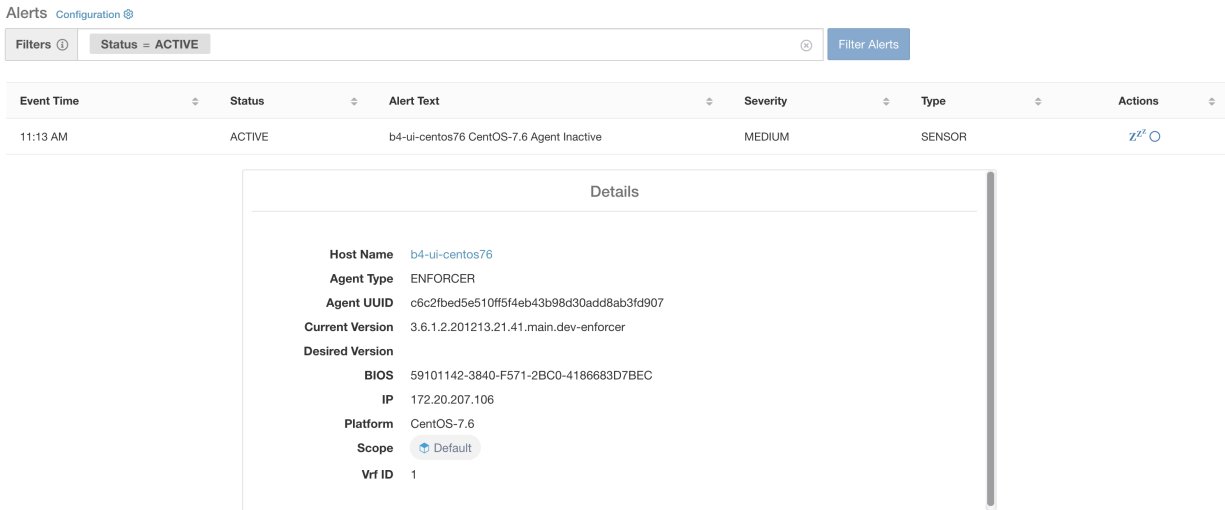


Fig. 3.10.1.1: Sensor alert details.

3.10.2 Sensor Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* field is structured and contains the following subfields for sensor alerts

Field	Alert Type	Format	Explanation
AgentType	*all*	string	"ENFORCER" or "SENSOR" depending on the installed type
HostName	*all*	string	Host name on which the agent is deployed
IP	*all*	string	IP address of the node
Bios	*all*	string	BIOS UUID of the node
Platform	*all*	string	Platform/OS information of the node
CurrentVersion	*all*	string	Software version of the agent on the node
DesiredVersion	*all*	string	Software version desired for the agent
LastConfigFetchAt	*all*	integer	Unix timestamp of when the agent last sent https request

3.10.2.1 Example of alert_details for a sensor alert

```
{
  "AgentType":"SENSOR",
  "Bios":"72EF1142-03A2-03BC-C2F8-F600567BA320",
  "CurrentVersion":"3.5.1.1.mrpm.build.win64-sensor",
  "DesiredVersion":"",
  "HostName":"win2k12-production-db",
  "IP":"172.26.231.193",
  "Platform":"MSServer2012R2Standard"
}
```

3.11 Troubleshooting Software Agents

This section lists some potential issues that the customers could possibly face during deployment and operating the software agents, methods could be used to troubleshoot the problems and some remedies that the customers could apply.

3.11.1 Agent deployment

3.11.1.1 Linux

Q: When I ran the command "rpm -Uvh tet-sensor-1.101.2-1.el6-dev.x86_64.rpm", it failed to install the agents and threw the error as follows:

> error: can't create transaction lock on /var/lib/rpm/.rpm.lock (Permission denied).

A: It seems that you don't have the right privileges to install the agents. Please either switch to root or use sudo to install the agents.

Q: What happened when running "sudo rpm -Uvh tet-sensor-1.0.0-121.1b1bb546.el6-dev.x86_64.rpm" I hit an error as follows:

> Preparing... ### [100%]
> which: no lsb_release in (/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin)
> error: %pre(tet-sensor-site-1.0.0-121.1b1bb546.x86_64) scriptlet failed, exit status 1
> error: install: %pre scriptlet failed (2), skipping tet-sensor-site-1.0.0-121.1b1bb546

A: The system does not satisfy the requirements to install the agents. In this particular case, lsb_release tool is not installed. Please refer to the section *Deploying Software Agents* for more information and install the required dependencies.

Q: When running "sudo rpm -Uvh tet-sensor-1.0.0-121.1b1bb546.el6-dev.x86_64.rpm" I hit an error as follows:

> Unsupported OS openSUSE project
> error: %pre(tet-sensor-1.101.1-1.x86_64) scriptlet failed, exit status 1
> error: tet-sensor-1.101.1-1.x86_64: install failed
> warning: %post(tet-sensor-site-1.101.1-1.x86_64) scriptlet failed, exit status 1

A: Your OS has not yet been supported to run software agents (in this particular case, "openSUSE project" is a non-supported platform). Please refer to the section *Deploying Software Agents* for more information.

Q: I have all the dependencies installed, and ran installation with proper privileges. The installation went well, no error was thrown. How do I know the agents installation really succeeded?

A: After the agents has been installed, you can run this command to verify:

> $ ps -ef | grep -e tet-sensor -e tet-engine
> root 12655 1 0 08:26 ? 00:00:00 tet-engine
> root 12659 12655 0 08:26 ? 00:00:00 tet-engine check_conf
> root 12660 12655 0 08:26 ? 00:00:00 tet-sensor -f sensor.conf

You should see 3 entries: two for tet-engine processes and one for tet-sensor process. If they are not running, then check if the following directory exists: /usr/local/tet. If it does not exist, then the installation could have failed.

3.11.1.2 Windows

Q: When I run the PowerShell agent installer script, I get one of the following errors:

1. The underlying connection was closed: An unexpected error occurred on a receive.
2. The client and server cannot communicate, because they do not possess a common algorithm

A: It is most likely because host and the server has mismatched SSL/TLS protocols configured. One can check the SSL/TLS version using the following command:

```
[Net.ServicePointManager]::SecurityProtocol
```

To set the SSL/TLS to be matching with server one can use the following command (note, this is not a permanant change, only temporary with the current PowerShell session):

```
[Net.ServicePointManager]::SecurityProtocol =
[System.Net.SecurityProtocolType]'Ssl3,Tls,Tls11,Tls12'
```

Q: When I run the MSI installer from the downloaded bundle, I get the following error:

```
This installation package could not be
opened. Verify that the package exists and
that you can access it, or contact the
application vendor to verify that this is a
valid Windows Installer package.
```

A: Make sure *C:\Windows\Installer* path exists. If running the MSI installer from the command line, make sure to not include the relative path when pointing to the msi file. Example of correct syntax:

```
msiexec /i "TetrationAgentInstaller.msi" /l*v "msi_install.log" /norestart
```

Q: I have observed that Windows Sensor software fails to upgrade if underlying NIC is Nutanix VirtIO Network Driver.

A: There is an incompatibility issue between Npcap 0.9990 and Nutanix VirtIO Network Driver version earlier than 1.1.3 and Receive Segment Coalescing is enabled.

The resolution for this is to upgrade Nutanix VirtIO Network Driver to version 1.1.3 or later.

3.11.1.3 Kubernetes

If the installer script fails during Kubernetes Daemonset Installation, there are a large number of possible reasons.

Q Is the Docker Registry serving images reachable from nodes ?

A Debug Direct or HTTPS Proxy issues with the cluster pulling images from Tetration cluster

Q Is the container runtime complaining about SSL/TLS insecure errors ?

A Verify that the Tetration HTTPS CA certificates are installed on all Kubernetes nodes in the appropriate location for the container runtime.

Q Docker Registry authentication and authorization of image downloads failures ?

A From each node, attempt to manually docker pull the images from the registry urls in the Daemonset spec using the Docker pull secrets from the secret created by the Helm Chart. If the manually image pull also fails, need to pull logs from the Tetration Cluster registryauth service to debug the issue further.

Q Is the Kubernetes cluster hosted inside the Tetration appliance heathy ?

A Check the service status page for the cluster to ensure all related services are healthy. Run the dstool snapshot from the explore page and retrieve the logs generated.

Q Are the Docker Image Builder daemons running ?

A Verify from the dstool logs that the build daemons are running.

Q Are the jobs that build Docker images failing ?

A Verify from the dstool logs that the images have not been built. Docker build pod logs can be used to debug errors during the buildkit builds. Enforcement Coordinator logs can also be used to debug the build failures further.

Q Are the jobs creating Helm Charts failing ?

A Verify from the dstool logs that the Helm Charts have not been built. Enforcement Coordinator logs will contain the output of the helm build jobs and can be used to debug the exact reason for the Helm Chart build job failures.

Q Installation bash script was corrupt ?

A Attempt to download the installation bash script again. The bash script contains binary data appended to it. If the bash script is edited in any way with a text editor or saved as a text file, special characters in the binary data may be mangled/modified by the text editor.

Q Kubernetes cluster configuration – too many variants and flavors, we support classic K8s.

A If the customer is running a variant of Kubernetes, there can be many failure modes at different stages of the deployment. Classify the failure stage - kubectl command run failure, helm command run failures, pod image download failures, pod privileged mode options rejected, pod image trust content signature failures, pod image security scan failures, pod binaries fail to run (architecture mismatch), pods run but the tetration services fail to start, tetration services start but have runtime errors due to unusual operating environment.

Q Are the Kubernetes RBAC credentials failing ?

A In order to run privileged daemonsets, we need admin privileges to the K8s cluster. Verify the the kubectl config file has its default context pointing towards the target cluster and admin-equivalent user for that cluster.

Q Busybox image available or downloadable from all cluster nodes ?

A Fix the connectivity issues and manually test that the busybox image can be downloaded. The exact version of busybox that is used in the pod spec must be available (pre-seeded) or downloadable on all cluster nodes.

Q API Server and etcd errors or a general timeout during the install ?

A Due to the instantiation of daemonset pods on all nodes in the Kubernetes cluster, the CPU/Disk/Network load on the cluster can spike suddenly. This is highly dependent on the customer specific installation details. Due to the overload, the installation process (images pulled on all nodes and written to disks) might take too long or overload the Kubernetes API server or the Tetration Docker Registry endpoint or, if configured, the proxy server temporarily. After a brief wait for image pulls on all nodes to complete and a reduction in CPU/Disk/Network load on the Kubernetes cluster nodes, retry the installation script again. API Server and etcd errors from the Kubernetes control plane indicate that the Kubernetes control plane nodes may be underprovisioned or affected by the sudden spike in activity.

Q Tetration Agent experiencing runtime issues with its operations ?

A Refer to the Linux Deep Visibility/Enforcement Agent troubleshooting section if the pods are correctly deployed and the agent has started running but is experiencing runtime issues. The troubleshooting steps are the same once the Kubernetes deployment has successfully installed and started the pods.

3.11.2 Anomaly Types

These are the most common issues encountered on the workflow when using and managing Tetration Agents.

3.11.2.1 Agent Inactivity

Agent has stopped checking to the cluster services. This can happen due to several reasons:

- The host might have been down
- The nextwork connectivity has been broken or blocked by firewall rules

- The agent service has been stopped

All platforms

- Verify the host is active and healthy
- Verify the agent service is up and running
- Verify the network connectivity to the cluster is working

3.11.2.2 Upgrade Failure

Agent upgrade has failed. This can be triggered by few cases such as:

- Not finding the package when the check in script attempts to download it - the upgrade package cannot be unpacked or the installer from the package cannot be verified.
- Installation process failing from an OS issue or dependency such as Npcap not successfully installed.

Windows

- Missing CA root certificate: *Certificate Issues*
- If agent was originally installed manually with a MSI install package, check if the Windows edition matches list of supported platforms in user guide: *Check If Platform Is Currently Supported*
- Check to make sure OS is configured correctly for Windows Installer operation: *Windows Installer Issues*
- Make sure nothing else is currently requiring Npcap services (such as Wireshark or 3rd party agents): *Npcap Issues*
- Make sure there is enough free disk space on host

Linux

- If the host OS has been upgraded since the last agent installation, verify the current release matches list of supported platforms in user guide: *Check If Platform Is Currently Supported*
- Make sure there have been no changes to the required dependencies since the last installation. You can run the agent installer script with *–no-install* option to re-verify these dependencies.
- Make sure there is enough free disk space on host

AIX

- Make sure there have been no changes to the required dependencies since the last installation. You can run the agent installer script with *–no-install* option to re-verify these dependencies.
- Make sure there is enough free disk space on host

Universal

- Universal Agents do not support automatic upgrades

3.11.2.3 Convert Failed

The current agent type mismatches desired agent type and the convert attempt has timed out. This issue can be caused by a communication issue when an agent does check_in to download the package, or wss service failed to push convert_commnad to the agent.

All Platforms

- Verify the current release and agent type matches list of supported platforms in user guide: *Check If Platform Is Currently Supported*

3.11.2.4 Convert Capability

The ability to convert the agent from one type (such as deep visibility) to another type (such as enforcement) is not available by all agents. If an agent that is not capable to do the conversion is required to convert, the anomaly will be reported.

3.11.2.5 Policy Out of Sync

The current policy (NPC) version last reported by the agent does not match the current version generated on the cluster. This can be caused by a communications error between the agent and the cluster, the agent failing to enforce the policy with the local firewall, or the agent enforcement service not running.

Windows

- If enforcement mode is WAF, verify there are no GPOs present on the host that would prevent the Firewall from being enabled, adding rules (with Preserse Rules Off) or setting default actions: *GPO Configurations*
- Verify there is connectivity between the host and the cluster: *SSL Troubleshooting*
- Verify the generated rule count is less than **2000**
- Verify the WindowsAgentEngine service is running: *sc query windowsagentengine*
- Verify there are available system resources

Linux

- Verify iptables and ipset is present with the *iptables* and *ipset* command
- Verify there is connectivity between the host and the cluster: *SSL Troubleshooting*
- Verify the tet-enforcer process is running: *ps -ef | grep tet-enforcer*

AIX

- Verify ipfilter is installed and running with the *ipf -V* command
- Verify there is connectivity between the host and the cluster: *SSL Troubleshooting*
- Verify the tet-enforcer process is running: *ps -ef | grep tet-enforcer*

3.11.2.6 Flow Export: Pcap Open

If the Tetration Agent cannot open the pcap device to caputre flows, you see errors in the Agent logs. A succesfully opened Pcap device will report as follows:

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetSen.exe.log*

```
I0609 15:25:52.354 24248 Started capture thread for device <device_name>
I0609 15:25:52.354 71912 Opening device {<device_id>}
```

Linux Log: */usr/local/tet/logs/tet-sensor.log*

```
I0610 03:24:22.354 16614 Opening device <device_name>
[2020/06/10 03:24:23:3524] NOTICE: lws_client_connect_2: <device_id>: address 172.29.
↪136.139
```

3.11.2.7 Flow Export: HTTPS Connectivity

Connectivity between the agent and the cluster is externally blocked therefore preventing flows and other system information from being delivered. This is caused by one or more configuration issues with network firewalls, SSL decryption services, or third party security agents on the host.

- If there are known firewalls or SSL decryption security devices between the agent and the cluster, make sure that communications to all Tetration collector and VIPs IP addresses are being permitted. For on-prem clusters, the list of collectors will be listed under VM Information in the Maintenance side menu. Look for collectorDatamover-*. For Tetration cloud, all the IP addresses that need to be permitted will be listed in your Portal.
- To help identify if there is SSL decryption, openssl s_client can be used to make a connection and display the returned certificate. Any additional certificate added to the chain will be rejected by the Agent's local CA. *SSL Troubleshooting*

3.11.3 Certificate Issues

3.11.3.1 Windows

Certificate Issues for MSI installer

MSI installer is signed using code signing certificate:

- Issued to: Cisco Tetration Analytics
- Issued by: Cisco Tetration Analytics

It uses timestamp certificate:

- Leaf Certificate: Symantec SHA256 Timestamping Signer - G2
- Intermediate Certificate: Symantec SHA256 Timestamping CA
- Root Certificate: VeriSign Universal Root Certification Authority

Windows Sensor Installation or upgrade will fail if digital signature of MSI installer is invalid.

Digital signature is invalid if

- *VeriSign Universal Root Certification Authority* is not a "Trusted Root Certification Authority" store
- *VeriSign Universal Root Certification Authority* is expired or revoked.

Issue 1

Installation of agent might fail with below error in the check_conf_update.log

> "TetrationAgentInstallaer.msi is not signed properly, aborting"

Resolution

- Run the command *certmgr* from command prompt
- Check *VeriSign Universal Root Certification Authority* in *Untrusted Certificates* store.
- Move it to *Trusted Root Certification Authority* store.

Issue 2

Windows Sensor upgrade fails with the following error in check_conf_update.log

> CERT_TRUST_STATUS.dwErrorStatus: 0x04000024
> CERT_TRUST_STATUS.dwInfoStatus: 0x04000024
> SignTool Error: WinVerifyTrust returned error: 0x800B010C
> A certificate was explicitly revoked by its issuer.

Resolution

- Run the command *certmgr* from command prompt
- Check *VeriSign Universal Root Certification Authority* in *Untrusted Certificates* store.
- Copy it to *Trusted Root Certification Authority* store.

Issue 3

Windows Sensor upgrade fails with the following in check_conf_update.log

> Failed to validate the upgrade package, exiting"
> "error code after running check_conf_update = 16"

OR

signtool verify /pa /v TetrationAgentInstaller.msi produces this error:

> SignTool Error: WinVerifyTrust returned error: 0x80096005
> The timestamp signature and/or certificate could not be verified or is malformed.

Resolution

- Run the command *certmgr* from command prompt
- Check *VeriSign Universal Root Certification Authority* in "Trusted Root Certification Authority" store

If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate VeriSign Universal Root Certification Authority from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificate "VeriSign Universal Root Certification Authority" under "Trusted Root Certification Authorities" and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

- Run the command *certmgr* from command prompt

- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

Certificate Issues for NPCAP installer

Applicable to Windows 2012 , Windows 2012 R2, Windows 8, Windows 8.1

NPCAP version: 0.9990

NPCAP Signing Certificate:

- Leaf Certificate: Insecure.Com LLC
- Intermediate Certificate: COMODO RSA Extended Validation Code Signing CA
- Root Certificate: COMODO RSA Certification Authority

NPCAP Timestamp certificate:

- Leaf Certificate: TIMESTAMP-SHA256-2019-10-15
- Intermediate Certificate: DigiCert SHA2 Assured ID Timestamping CA
- Root Certificate: DigiCert Assured ID Root CA

Issue 1

Windows Agent Installation might fail with below error in msi_installer.log

> CheckServiceStatus : Exception System.InvalidOperationException: Service npcap was not found on computer '.'. —> System.ComponentModel.Win32Exception: The specified service does not exist as an installed service

Resolution

- Run the command *certmgr* from command prompt
- Check "COMODO RSA Certification Authority" in "Trusted Root Certification Authority" store.
- If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate "COMODO RSA Certification Authority" from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificate "COMODO RSA Certification Authority" under "Trusted Root Certification Authorities" and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

Applicable to Windows 2008 R2

NPCAP version: 0.991

NPCAP Signing Certificate:

- Leaf Certificate: Insecure.Com LLC
- Intermediate Certificate: DigiCert EV Code Signing CA

- Root Certificate: DigiCert High Assurance EV Root CA

NPCAP Timestamp certificate:

- Leaf Certificate: DigiCert Timestamp Responder
- Intermediate Certificate: DigiCert Assured ID CA-1
- Root Certificate: VeriSign DigiCert Assured ID Root CA

Issue 1

Windows Agent Installation might fail with below error in msi_installer.log

> CheckServiceStatus : Exception System.InvalidOperationException: Service npcap was not found on computer '.'. —> System.ComponentModel.Win32Exception: The specified service does not exist as an installed service

Resolution

- Run the command *certmgr* from command prompt
- Check *DigiCert High Assurance EV Root CA* in *Trusted Root Certification Authority* store.
- If it the certificate is missing, import it from other machine.

To import the certificate, follow below steps:

First export the certificate "DigiCert High Assurance EV Root CA" from one of Working server. Follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificate "DigiCert High Assurance EV Root CA" under "Trusted Root Certification Authorities" and go to All tasksExport.
- Copy the exported certificate to the Non-working server and then import the certificate.

To import the certificate, follow below steps:

- Run the command *certmgr* from command prompt
- Right click on the certificates tab under Trusted Root Certification Authorities and go to All tasksImport.
- Select the Root certificate that you copied and add it in the store.

3.11.4 Check If Platform Is Currently Supported

3.11.4.1 Windows

- Run the command *winver.exe*
- Compare this release to what is listed here: *Supported platforms*

3.11.4.2 Linux

- Run *cat /etc/os-release*
- Compare this release to what is listed here: *Supported platforms*

3.11.4.3 AIX

- Run the command *uname -a*
- Note: The major and minor versions are reversed

```
p7-ops2> # uname -a
AIX p7-ops2 1 7 00F8AF944C00
```

- In this example, the first number after the host name is the minor and the second number is the major version, so AIX version 7.1. Compare this release to what is listed here: *Supported platforms*

3.11.5 Windows Installer Issues

- Make sure there is a *C:\Windows\Installer* directory. This is not visible in File Explorer, easiest way to verify is in a CMD session and running: *dir C:\Windows\Installer*
- Check if the *Windows Installer* service is not disabled. It must be set to *Manual*
- Check to see if there are no other errors being reported by Windows Installer. Check Windows System Event logs under Windows Logs -> Appliation -> Source *MsiInstaller*

3.11.6 Npcap Issues

Npcap is a pcap tool used for Windows Agent only.

3.11.6.1 Npcap will not upgrade (manully or via agent)

- Npcap will sometimes not uninstall correctly if a process is currently using the Npcap libraries. To check for this run the following command:

```
PS C:\Program Files\Npcap> .\NPFInstall.exe -check_dll
WindowsSensor.exe, Wireshark.exe, dumpcap.exe
```

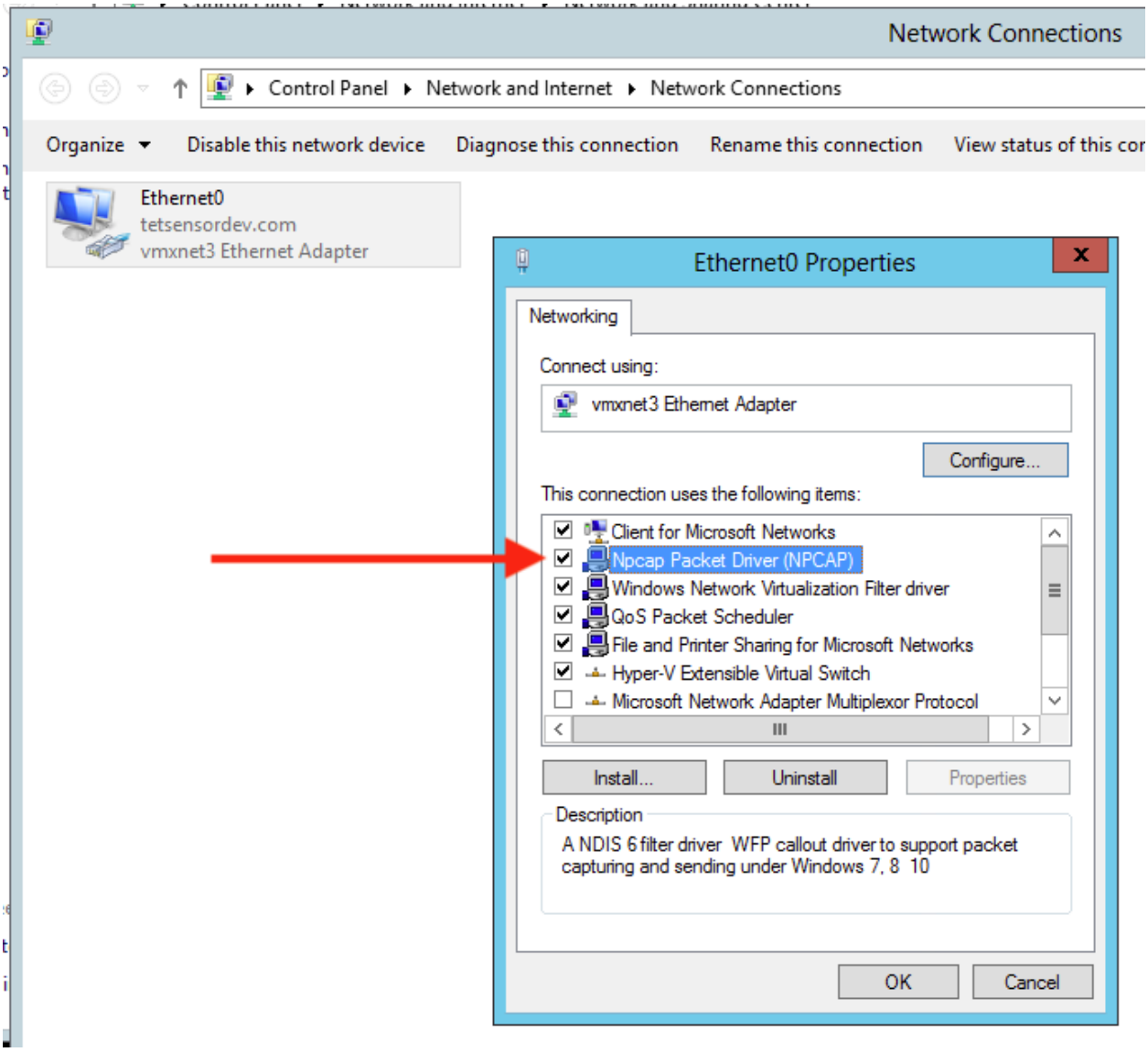
If you see processes listed, they must be stopped before the Npcap upgrade can continue. If no processes are using Npcap the above command will simply show *<NULL>*

3.11.6.2 Npcap will not install

- Check CA certificates installed on the system: Npcap Certiciates
- Check Windows Installer issues: *Windows Installer Issues*
- Verify no other user on the system is making changes to the network interfaces. This can cause a COM lock preventing NDIS driver binding.

3.11.6.3 Verify if Npcap is fully installed

1. Check Control Panel → Programs and Features to see if Npcap is listed as an installed application
2. Make sure the Npcap Packet Driver has a binding to the NIC in question (checkmark is present)



3. Check if the network driver is installed

```
C:\Windows\system32>pnputil -e | findstr Nmap
Driver package provider :   Nmap Project
```

4. Check if the driver service is installed and RUNNING

```
C:\Windows\system32>sc query npcap

SERVICE_NAME: npcap

      TYPE               : 1  KERNEL_DRIVER

      STATE              : 4  RUNNING
```

5. Check if the registry entry is there (used by Agent installer to verify npcap exists already)

```
C:\Windows\system32>reg query HKLM\software\wow6432node\npcap
HKEY_LOCAL_MACHINE\software\wow6432node\npcap
    AdminOnly    REG_DWORD    0x1
    WinPcapCompatible    REG_DWORD    0x0
    (Default)    REG_SZ    C:\Program Files\Npcap
```

6. Check if the installed Npcap program files are all there

```
C:\Windows\system32>dir "c:\program files\npcap"
 Directory of c:\program files\npcap
04/29/2020  02:42 PM    <DIR>          .
04/29/2020  02:42 PM    <DIR>          ..
01/22/2019  08:16 AM               868 CheckStatus.bat
11/29/2016  03:43 PM             1,034 DiagReport.bat
12/04/2018  11:12 PM             8,908 DiagReport.ps1
01/09/2019  09:22 PM             2,959 FixInstall.bat
04/29/2020  02:42 PM           134,240 install.log
01/11/2019  08:52 AM             9,920 LICENSE
03/14/2019  08:59 PM            10,434 npcap.cat
03/14/2019  08:57 PM             8,657 npcap.inf
03/14/2019  09:00 PM            74,040 npcap.sys
03/14/2019  08:57 PM             2,404 npcap_wfp.inf
03/14/2019  09:00 PM           270,648 NPFInstall.exe
04/29/2020  02:42 PM           107,783 NPFInstall.log
03/14/2019  09:01 PM           175,024 Uninstall.exe
              13 File(s)        806,919 bytes
               2 Dir(s)  264,417,628,160 bytes free
```

7. Check to see if the .sys driver file is in the Windows driver folder

```
C:\Windows\system32>dir "C:\Windows\System32\Drivers\npcap.sys"
Directory of C:\Windows\System32\Drivers
03/14/2019  09:00 PM            74,040 npcap.sys
           1 File(s)         74,040 bytes
```

3.11.6.4 Network Connectivity issues during NPCAP installation or upgrade

Applicable to Windows 2016 Only

If you have a 3rd party LWF (Light Weight Filter) driver (e.g. netmon) or a teaming adapter is configured in your setup, and NPCAP is installed during agent deployment, you might experience

RDP is reconnected
NetBios service is restarted
Similar network connectivity issues

This is due to a BUG in Windows 2016 OS.

3.11.6.5 OS Performance and/or stability Issues

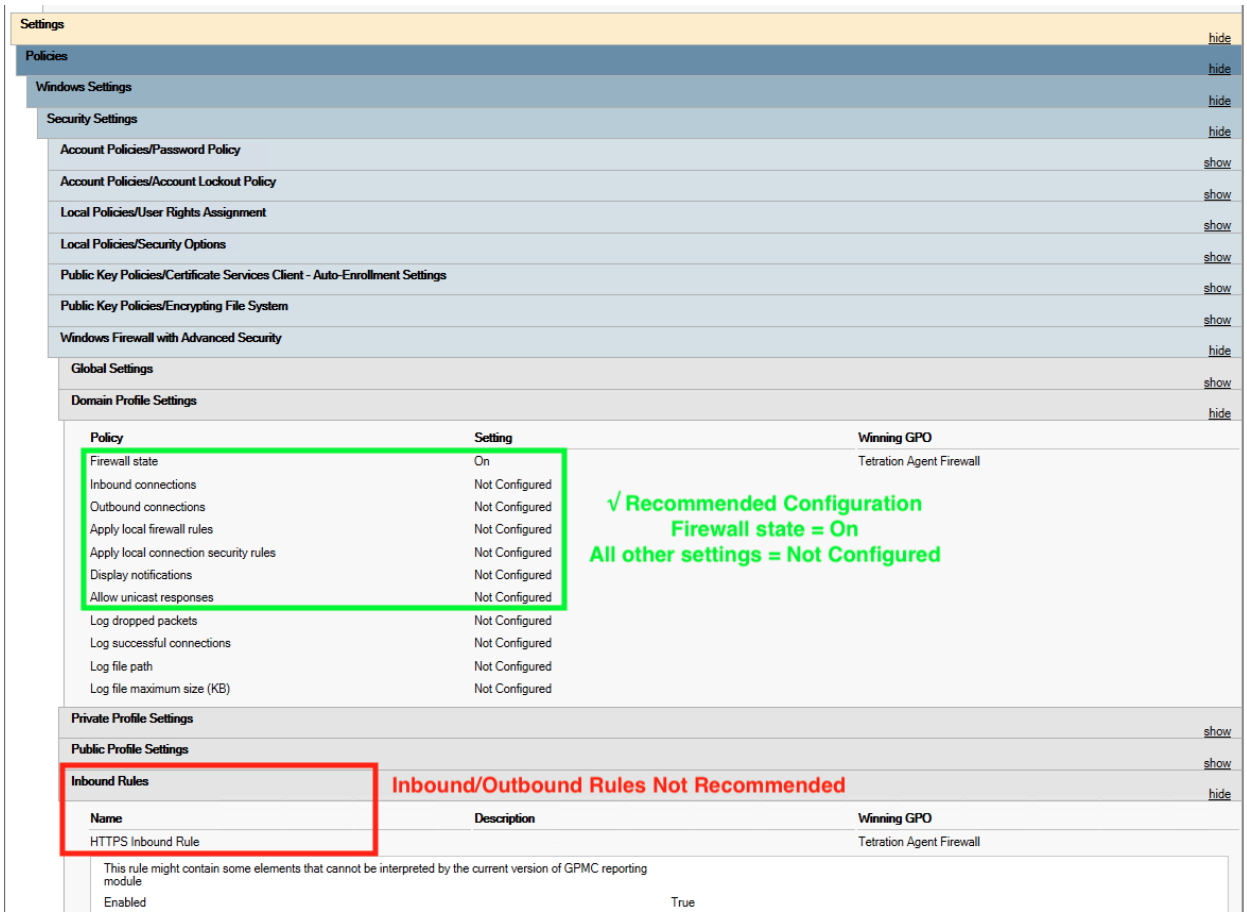
OS may experience unknown performance or stability issues if the installed NPCAP version or NPCAP configuration is not supported by Tetration Software.

Supported NPCAP Version: : 0.991 and 0.9990

3.11.7 GPO Configurations

Tetration Enforcement Agent requires only the Firewall to be enabled with either a local setting or GPO. All other GPO settings should not be set and left as "Not Configured."

- To check if a GPO setting is blocking enforcement you can check the *C:\Program Files\Cisco Tetration\Logs\TetEnf.exe.log* log and search for the following error examples:
- **Rules conflicting with "Preserve Rules=No" setting:** "There are firewall rules set in the Group Policy. Tetration agent does not have permission to remove these"
- **Firewall set to off:** "GPO has disabled firewall for DomainProfile"
- **Default Action is set:** "Group Policy has conflicting default inbound action for DomainProfile"
- To check what GPO policies are being applied to the host, run *gpresult.exe /H gpreport.html* and open the generated HTML report. In the example below *Tetration Agent Firewall* is applying a Inbound rule which will conflict with Enforcement if "Preserve Rules" is set to "No."



3.11.8 Agent To Cluster Communications

The Tetration Agent maintains connections to the cluser over mutliple channels. Depending on the type of Agent, the number of connections varies.

3.11.8.1 Types of connections

- **WSS:** Persistant socket connection over port 443 to the cluster
- **Check in:** A HTTPS call to the cluster every 15-20 minutes to check for current configurations, check for updates and to update the active state of the agent to the cluster. This also reports upgrade failures.
- **Flow export:** Persistant SSL connection over port 443 (TaaS) or 5640 (On-premise) to send flow metadata to the cluster
- **Enforcement:** Persistant SSL connection over port 443 (Taas) or 5660 (On-premise) to pull in enforcement policies and report enforcement state

3.11.8.2 Checking the connection state

The Teration UI will report either an inactive agent (no longer checking-in), no exported flows (on Agent Workloadn Profile page under Stats), or failed enforcement. Depending on the error, you can check different logs on the workload to help determine the source of the issue.

Inactive Agent

Windows Log: *C:\Program Files\Cisco Tetration\Logs\check_conf_update.log*

Linux Log: */usr/local/tet/logs/check_conf_update.log*

An HTTP response code of 304 is expected and means there is no configuration change. Error code = 2 is expected as well. Any other HTTP response code will incidate a issue talking to the WSS service on the Tetration cluster.

```
Tue 06/09/2020 17:25:25.08 check_conf_update: "curl did not return 200 code, it's 304,
↪ exiting"
Tue 06/09/2020 17:25:25.08 check_conf_update: "error code after running check_conf_
↪update = 2"
```

- **304** Expected, no config change. Successful check-in
- **401** Registration is not successful, missing Activation Key (TaaS)
- **403** Agent already registered to the cluster with same UUID
- **000** Indicates connection issue with SSL. Either curl could not reach the WSS server or there is a issue with the certificate. See SSL troubleshooting: *SSL Troubleshooting*

No exported flows

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetSen.exe.log*

Linux Log: */usr/local/tet/logs/tet-sensor.log*

The following indicates a successful connection to WSS

```
cfgserver.go:261] config server: StateConnected, wss://<config_server_ip>:443/wss/
↪<sensor_id>/forensic, proxy:
```

The following incideates a successful connection to the Collectors

```
collector.go:258] next collector: StateConnected, ssl://<collector_ip>>:5640
```

If there are errors connecting to either WSS or the Collectors, check your firewall configuration or verify if any SSL decryption is occuring between the agent and Tetration. See: *SSL Troubleshooting*

Failed to enforce policy

Windows Log: *C:\Program Files\Cisco Tetration\Logs\TetEnf.exe.log*

Linux Log: */usr/local/tet/logs/tet-enforcer.log*

```
ssl_client.cpp:341] Successfully connected to EFE server
```

If there are errors connecting to the EFE server, check your firewall configuration or verify if any SSL decryption is occuring between the agent and Tetration. See: *SSL Troubleshooting*

3.11.9 SSL Troubleshooting

3.11.9.1 Agent Communications Overview

Cisco Tetration agents use TLS to secure the TCP connections to the Tetration Cloud SaaS servers. These connections are broken down into three distinctive channels.

- Agent -> Cisco Tetration SaaS control channel over port TCP/443 (TLS) (sensorVIP)

  This is a low volume control channel that allows the agent to register with Cisco Tetration and also handles configuration pushes and software upgrade notifications.

- Agent -> Cisco Tetration SaaS flow data over TCP/443 (TLS) (collector)

  Flow data is the extracted flow metadata information; the data will be sent to 1 set of 16 IP addresses at a time. The second set of IP addresses is for standby. This is around 1 – 5% of actual server traffic.

- Agent ->Cisco Tetration SaaS enforcement data over TCP/443 (TLS) (efe)

  The enforcement data channel is a low volume control channel that is used to push the policies to the sensors and also gather enforcement statistics.

The sensor validates the the TLS certificate from from the Tetration Cloud control, data and enforcement servers against a local CA that is installed with the agent. No other CAs are used, so any other certificate sent to the agent will result in a verification failure and the agent will not connect. This will result in the agent not registering, checking-in, sending flows or receiving enforcement policies.

3.11.9.2 Configuring IP traffic for Agent Communications

A typical configuration for most will be to have a perimeter firewall and possibly a proxy between the agents (workflows) and Tetration TaaS.

Note Cisco Tetration gathers your gateway/NAT IP information during the on-boarding as well and automatically adds the information at the time of tenant creation. If you add new IP addresses or change IP addresses in the portal, the changes will require review and approval by Tetration staff.

In addition to adding your gateway/NAT IP addresses in the TaaS portal, there might be more changes required to your network to allow the traffic outbound and unmodified:

Allow outbound port 443 over TLS/HTTPS on the perimeter firewall

Configure proxy bypass and SSL/TLS bypass on the web proxy, if a decrypting web proxy is being used.

Note If you are using a transparent web proxy at the data center, you must route the specific SaaS IP address and configure the bypass rules. Sensors are connections that cannot do automatic HTTPS redirection.

The list of IPs the agents will communicate with is available on the TaaS portal. The IPs to add to your firewall outbound configuration and proxy bypass are labeled collector-n, efe-n (only if enforcement is being deployed), and sensorVIP. There are typically 17 to 33 IPs to add for agent communication, but there could more or less depending on your TaaS configuration.

3.11.9.3 Troubleshooting SSL/TLS Connections

As discussed in the previous section, it is important to configure your explicit or transparent web proxy to bypass SSL/TLS decryption for agent communications. If the bypass is not configured, these proxies might attempt to decrypt SSL/TLS traffic by sending its own certificate to the agent. Because the agent only uses its local CA to validate the certificate, these proxy certificates will cause connection failures.

Symptoms include agent failing to register to the cluster, agent not checking-in, agent not sending flows, and/or agent not getting enforcement configuration (if enforcement agent is installed).

Note Troubleshooting steps below are assuming default installation paths were used. Windows: C:Program FilesCisco Tetration Linux: /usr/local/tet. If you installed your agents in a different location, please substitute that location in the instructions.

SSL/TLS Connection issues are reported in the agent logs. To verify if there are SSL errors in the logs, run the following commands for the associated issue being observed.

Registration, check-in

Linux

```
grep "NSS error" /usr/local/tet/log/check_conf_update.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\check_conf_update.log" | select-
↪string -pattern "SSL certificate problem"
```

Flows

Most of the SSL/TLS connection issues seen are during the initial connection and registration of the agent. Sending flows relies on the registration to be complete before attempting to connect. SSL/TLS errors seen here would be the result of the sensorVIP IPs being allowed but not the collector IPs.

Linux

```
grep "SSL connect error" /usr/local/tet/log/tet-sensor.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\WindowsSensor*.log" | select-
↪string -pattern "Certificate verification error"
```

Enforcement

Linux

```
grep "Unable to validate the signing cert" /usr/local/tet/log/tet-enforcer.log
```

Windows (PowerShell)

```
get-content "C:\Program Files\Cisco Tetration\logs\WindowsSensor*.log" | select-
↪string -pattern "Handshake failed"
```

If an SSL error is seen in the log checks above you can verify what certificate is being sent to the Agents with the following commands.

Explicit Proxy - where a proxy is configured in user.cfg

Linux

```
curl -v -x http://<proxy_address>:<port> https://<sensorVIP>:443
```

Windows (PowerShell)

```
cd "C:\Program Files\Cisco Tetration"
.\curl.exe -kv -x http://<proxy_address>:<port> https://<sensorVIP>:443
```

Transparent Proxy - No user.cfg proxy configuration required. It's a proxy configured between all HTTP(S) traffic from agent to the internet.

Linux

```
openssl s_client -connect <sensorVIP from TaaS Portal>:443 -CAfile /usr/local/tet/
↪cert/ca.cert
```

Windows (PowerShell)

```
cd C:\Program Files\Cisco Tetration
.\openssl.exe s_client -connect <sensorVIP from TaaS Portal>:443 -CAfile cert\ca.cert
```

You are looking for the following in the openssl s_client respose

```
Verify return code: 0 (ok)
```

If you see an error, examine the certificate. An example certificate (chain) should include only the following cert (CN IP is an example):

Certificate chain

```
0 s:/C=US/ST=CA/L=San Jose/O=Cisco Systems, Inc./OU=Tetration, Insieme BU/CN=129.146.
↪155.109
i:/C=US/ST=CA/L=San Jose/O=Cisco Systems, Inc./OU=Tetration Analytics/CN=Customer CA
```

If you see additional certificates, then there is possibly a Web decrypting proxy between the agent and Tetration. Please contact your security or network group and verify the proxy bypass using the listed IPs from the above Configuring IP traffic for Agent Communications section have been configured.

Windows sensor installation script fails on Windows 2016 servers: Error message that might appear "The underlying connection was closed: An unexpected error occurred on a receive." Possible reason might be the SSL/TLS versions set in PowerShell.

To check the SSL/TLS versions running, run the following command:

```
[Net.ServicePointManager]::SecurityProtocol
```

If the output from the above command is:

```
Ssl3, Tls
```

Then please use the below command to change the allowed protocols and retry the installation:

```
[Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]'Ssl3,
↪Tls,Tls11,Tls12'
```

3.11.10 Agent operations

Q: I have installed the agents successfully, but I didn't see it on UI Sensor Monitoring page.

A: An agent is required to register with backend server running within cluster before it could start operating. When an agent is not shown on UI page, most likely it's because the registration has failed. There are a few things we could check to see why a registration failed:

- Check if the connection between the agent and the backend server is working properly
- Check if the curl request could be sent to backend server properly
- Check HAProxy access and backend server logs to see if the registration request made it to the server
- Check the error return from curl request in the log file

Q: The agent is installed and I could find in on UI page. However, the "SW Ver" column shows "initializing" instead of a version string.

A: After the initial agent is installed and registered with the backend server, it would take another 30 minutes for the agent to report its version.

Q: The agent is upgraded properly, but the "SW Ver" fields still show the old version after a long time (like several hours).

A: After the agent is upgraded successfully, it will try to send a curl request to report its current running version and check for new version in the same request. It is possible that the request couldn't make it to the backend, due to several reason:

- The request is timed out, couldn't get the response in time
- The network is facing problem, agent couldn't connect to backend servers

Q: I have an agent running on RHEL/CentOS-6.x and it is working properly. I am planning to upgrade the OS to RHEL/CentOS-7.x. Would the agent still work after the upgrade?

A: currently we do not support the scenario in which the OS has been upgraded, especially upgrading the major releases. In order to have the agent work after OS upgrade, do the following steps:

- Uninstall the existing agent software
- Clean up all files, including certs
- Go to UI, delete the agent entry
- Upgrade the OS to the desired version
- Install the agent software on the new OS

Q: I have an agent running on RHEL/CentOS-6.x and it is working properly. I am planning to rename the host. Would the agent still work after rename/reboot?

A: An agent identity is calculated based on the host's uniqueness, including hostname and bios-uuid. Changing hostname changes the host's indentify. It is recommended to do the following:

- Uninstall the existing agent software
- Clean up all files, including certs
- Go to UI, delete the old agent entry
- Rename the host and reboot
- Install the agent software again

Q: Universal agents fail to register with cluster with Certificate error?

A: It is most likely because hosts do not have up-to-date system time. Simply update it.

Q: On Windows host, firewall deviation was caused by adding/deleting/modifying a rule. How do I find the rule?

A: On deviation detection, agent logs the last 15 seconds of firewall events to "C:\Windows\System32\config\systemprofile\AppData\Roaming\tet\firewall_events". Rule that caused deviation will be found in the latest file created as policy_dev_<policy id>_<timestamp>.txt

CHAPTER

FOUR

INVENTORY

The **Inventory** drop-down menu options can be accessed using the top-level menu item. The options available in the menu vary depending on your role but may include **Search**, **Filters**, and **Upload**.

4.1 Scopes and Inventory

Scopes and Inventory Overview

This section provides visibility of the scope hierarchy, as well as all of the inventory it contains. Scopes categorize all of the inventory using a hierarchical structure. See *Inventory*. On the left is the scope directory user interface. Here, you can traverse down your scope hierarchy. Each scope is displayed in a scope card. The name of the scope is displayed, the number of children scopes, the inventory count, and uncategorized inventory if applicable. Clicking on a scope card will update the pane to the right to show details about that scope as well as a filterable list of all of its inventory.

Scope Design Principles

1. Inventory is matched to scope tree according to dynamic query match.
   - Queries may match against IP/Subnet or Label (preferred)
   - Tree is formed through conjunctive query at each layer
2. Scope structure may be location specific if appropriate.
   - Combined Cloud vs Data Center and Cloud Specific vs Geographic location
3. Each layer of the scope tree should represent an anchor point for:
   - Policy control
   - Role Based Access Control (RBAC)
4. Every child scope should be a subset of its parent scope
   - Ensure non-overlapping sibling scopes, see *Scope Overlap*

Note: Every organization is structured differently, and depending on your industry, require different approaches. Choose one focus in designing your scope hierarchy; location, environment, or application.

Key Features

Filtering feature for both scopes and inventory provides you with the ability to quickly traverse down the scope tree or filter the scope hierarchy and filter the inventory items of the selected scope.

Inventory count is displayed in the scopes card, providing a quick view into the amount of workloads in the scope.

4.1.1 Scopes

Scopes are a foundational element to configuration and policy in Tetration. Scopes are a collection of workloads arranged in a hierarchy. Workloads labelled to serve as attributes that build a model about where it is, its role, and its function in your environment. Scopes provide a structure to support dynamic mechanisms like identification and attributes associated with an IP that may change over time.

Scopes are used to group datacenter applications and, along with *Roles*, enable fine grained control of their management. For example, Scopes are used throughout the product to define access to *Segmentation*, *Flows* and *Filters*.

Scopes are defined hierarchically as sets of trees with the root corresponding to a **VRF**. As a result, each Scope tree hierarchy represents disjoint data that does not overlap with another Scope tree, see *Scope Overlap*

Scope Definition

Each individual Scope is defined with the attributes below:

Attribute	Description
Parent Scope	The parent of the new scope defines the tree hierarchy structure.
Name	The name to identify the scope.
Type	This is used to specify different categories of inventory. If none are applicable, or the scope contains a mix, it can be left blank.
Query	The Query defining the individual scope.

Note: Scopes should be defined in a hierarchy that mimics the application ownership hierarchy of the organization.

Note: Query may match against IP/Subnet or other Inventory attributes.

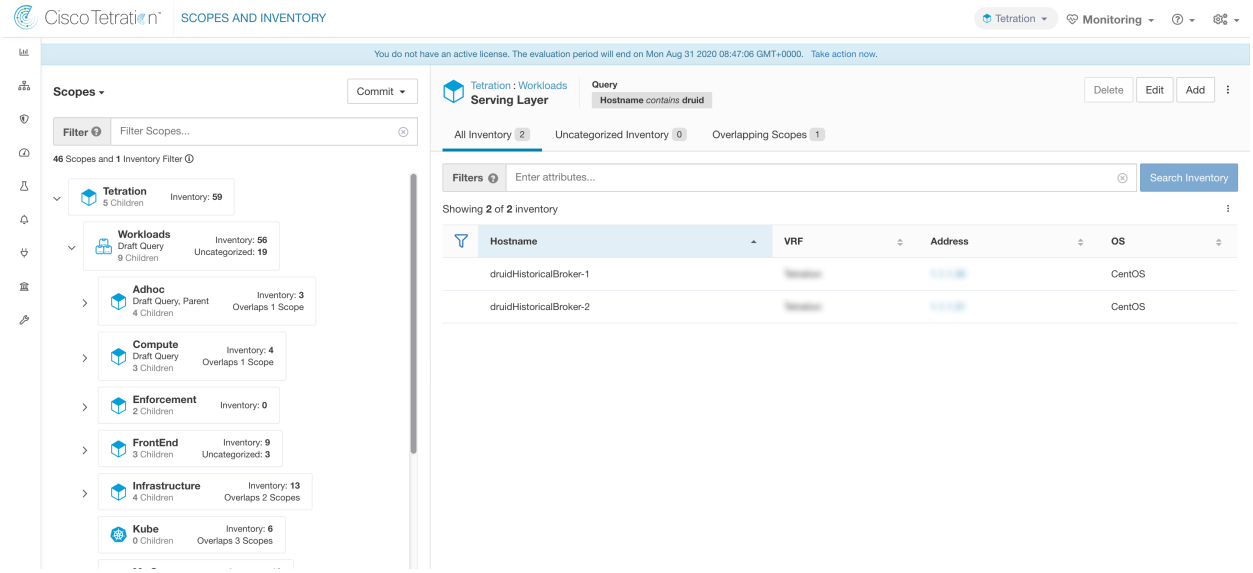


Fig. 4.1.1.1: Example of Traversing through Scope Hierarchy

The scope directory displays the scope hierarchy as well as some details of each scope (e.g. Inventory Count, number of child scopes, Workspaces). Clicking on a scope selects that scope and the details pane to the right updates with more information about that scope as well as that scope's inventory.

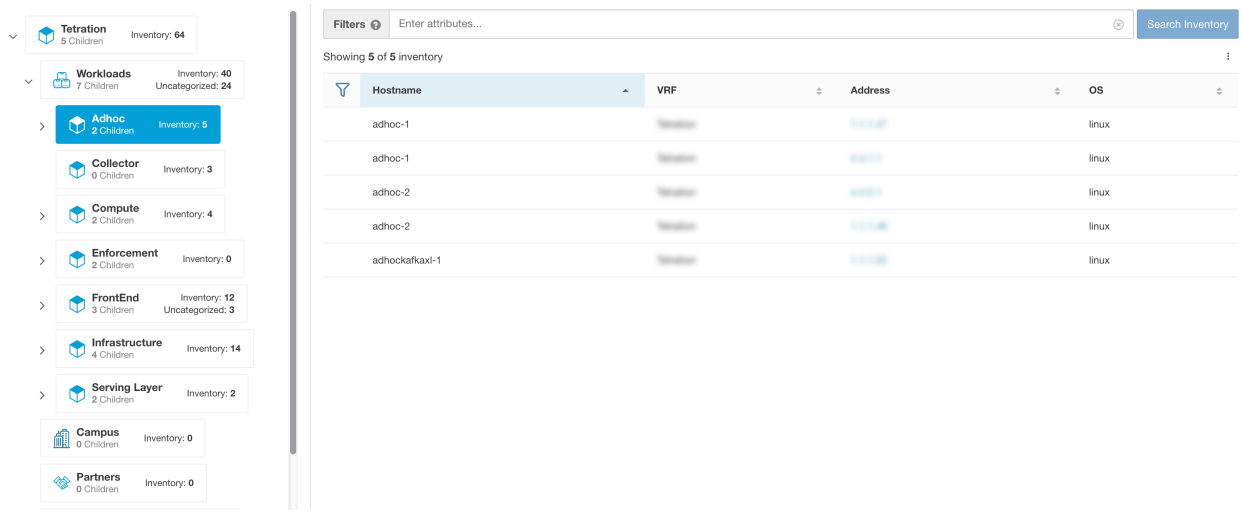


Fig. 4.1.1.2: Inventory count

4.1.1.1 Scope Filter

Users can use the Scope filter to quickly identify different scope details such as overlapping scopes and query. The filter feature is also helpful in identifying query changes, parent changes, etc.

Field	Description
Name	Filter by the name of the Scope or Inventory Filter.
Description	Filter by text appearing in the description of a scope.
Query	Filter by fields or values used in the query.
Query Change	Filter by scopes that have an uncommitted query.
Parent Change	Filter by scopes that have been moved in the draft but not committed.
Is Inventory Filter	Show Inventory Filters that are restricted to their ownership scope.
Has Workspace	Filter by scopes that have a primary workspace.
Has Enforced Workspace	Filter by scopes that have a primary workspace that is enforced.
Has Overlaps	Filter by scopes that have inventory in common with a sibling scope.
Has Invalid Query	Filter by scopes that have a query that uses invalid or unknown labels.

Examples:

Has Overlaps

Example of Scope Overlap

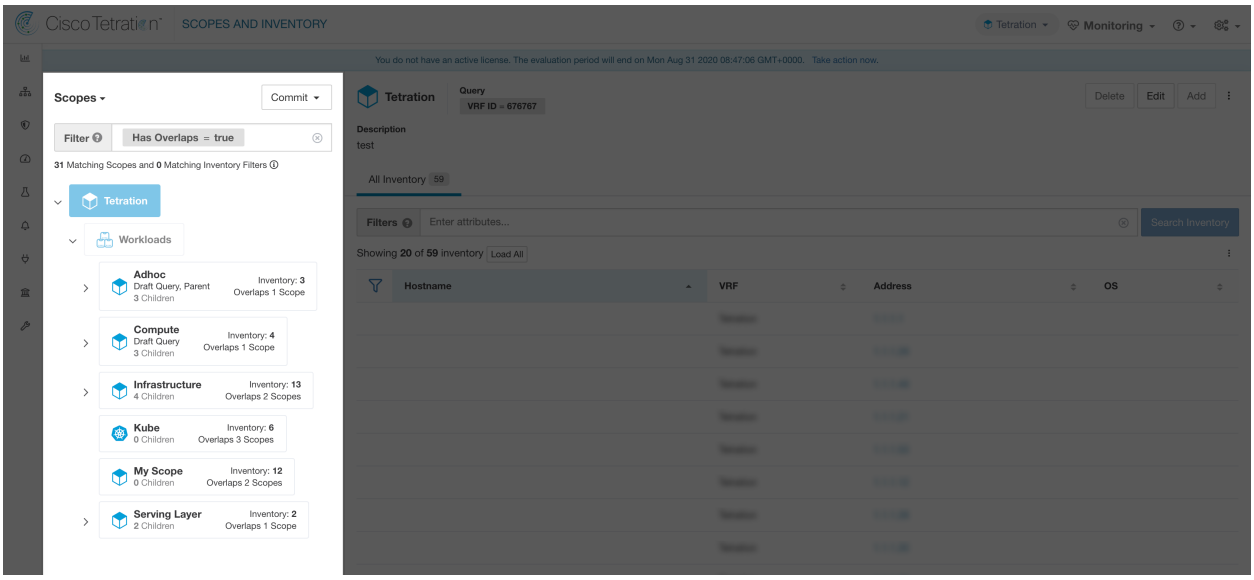


Fig. 4.1.1.1.1: Has Overlaps. For more information see *Scope Overlap*

Parent Change

Scopes that are moved in the draft but not yet committed.

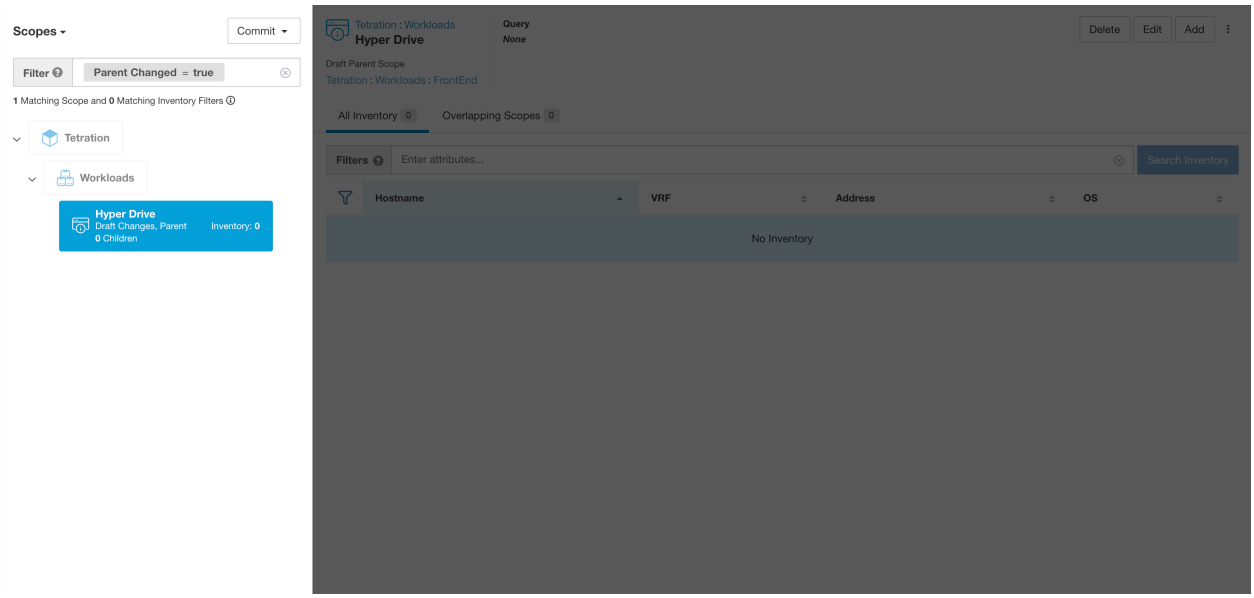


Fig. 4.1.1.1.2: Parent Change

4.1.1.2 Full Scope Queries

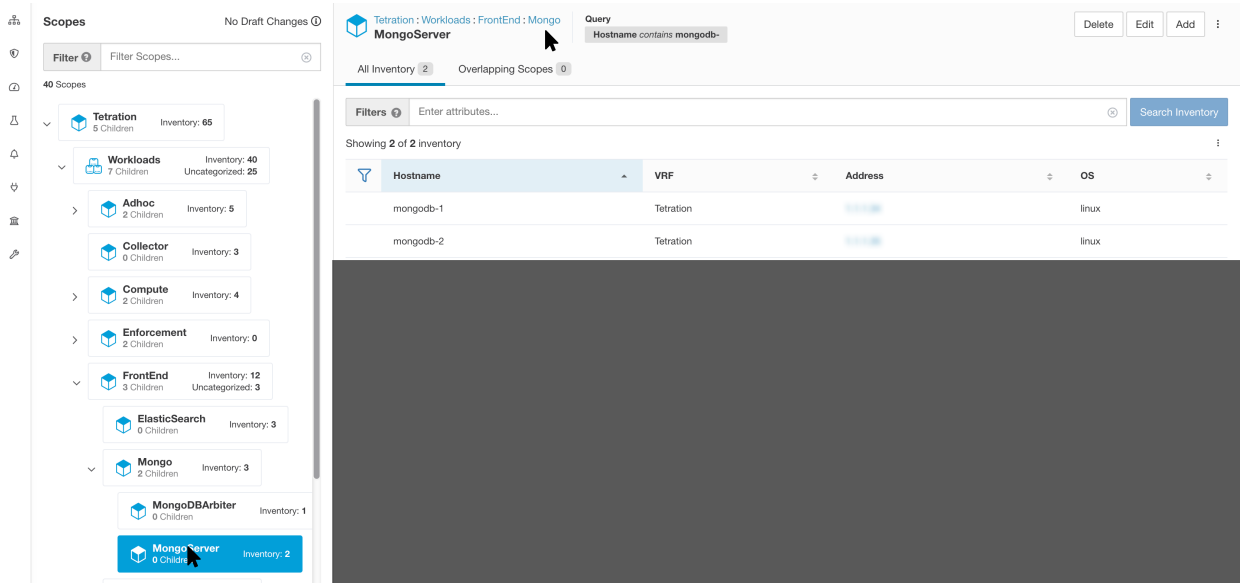


Fig. 4.1.1.2.1: Example of Scope Hierarchy

Scopes are defined hierarchically, the full query of the scope is defined as the logical 'and' of the scope along with all of its parents. Using the example above, assets assigned to the `Workloads:FrontEnd:Mongo`

Scope would match:

```
vrf_id = 676767 and (ip in 1.1.1.0/24) and (Hostname contains mongo).
```

Where vrf_id = 676767 comes from the root scope query and ip in 1.1.1.0/24 comes from the parent scope query.

Note: It is a best practice to not have overlapping queries at the same level. This removes the importance of ordering and reduces confusion. See *Scope Overlap*

4.1.1.3 Providing Access to Scopes

Users can be given Read, Write, Execute, Enforce and Owner abilities on Scopes. An overview is provided below, for complete details see *Roles*.

A User is given access to a "sub-tree". ie. the given Scope and all its children. Using the above example, a user with Read access to the `Workloads:FrontEnd` scope would, by inheritance, have read access to all the scopes under `Workloads:FrontEnd` including:

- `Workloads:FrontEnd:Mongo`
- `Workloads:FrontEnd:ElasticSearch`
- `Workloads:FrontEnd:Redis`
- etc…

It is possible to define Roles with access to multiple Scopes. For example, an "Mongo Admin" role might have Owner access to the Scopes:

- `Workloads:FrontEnd:Mongo:MongoServer`
- `Workloads:FrontEnd:Mongo:MongoDBArbiter`

Roles and Capabilities allow the users to have "horizontal" access to the Scope hierarchy.

Scope Abilities are also inherited. For example, having the Write ability on a Scope allows one to also Read that information.

4.1.1.4 Viewing Scope

Every user can view the scope tree they have access to. Users who have the Owner ability on the root scope have the ability to create, edit and delete scope in that tree. To access this view:

1. Click on the **gear menu** in the top-right corner.
2. Select **Scopes**.

Note: This is the same as accessing Inventory Search in the sidebar Visibility->Inventory Search

You can traverse through the complete scope hierarchy (up to the root) for any Scopes you have access to. This complete traversal provides context as users can create policies to any Scope. Several actions can be performed on this page:

- Click the chevron in the scope hierarchy to show that scope's children.
- Clicking on a scope card will update the pane to the right to show details about that scope as well as a filterable list of all of its inventory.

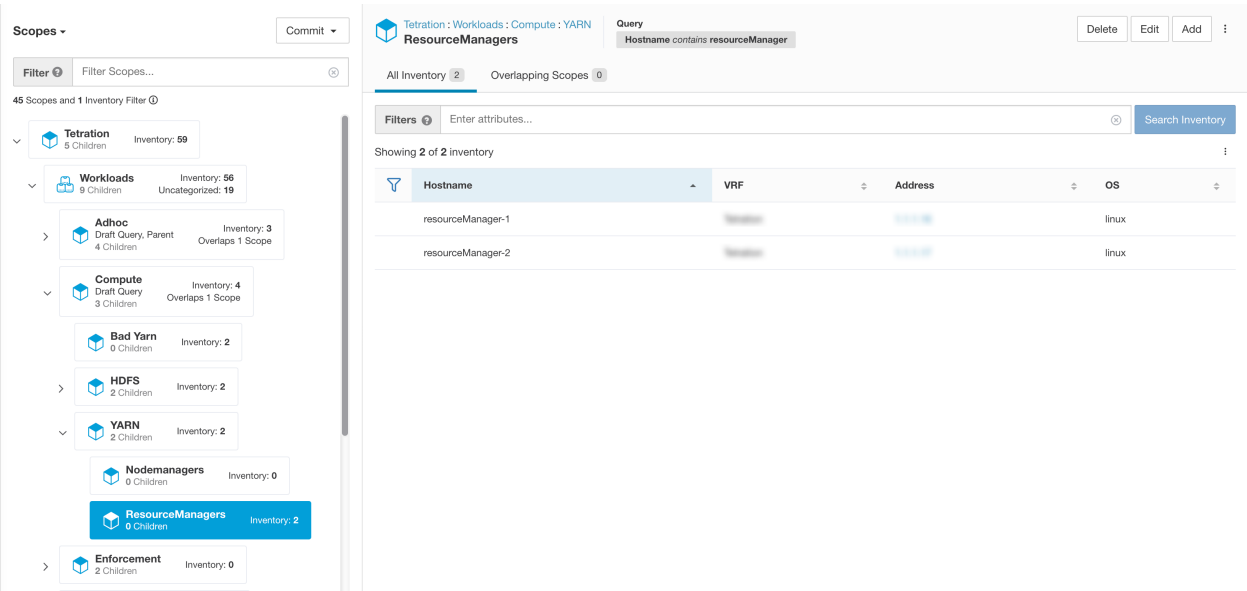


Fig. 4.1.1.4.1: Example Non-Admin View

4.1.1.5 Creating a New Scope

Child scopes are created on the **Scopes** admin page. This action requires the `SCOPE_OWNER` ability on the root scope. **Site Admins** are owners of all scopes.

Creating a child scope will impact the application inventory membership of the parent. As a result, the parent scope will be marked as having "draft changes". The changes will need to be committed and dependent structures will need to be updated. See *Commit Changes*.

1. Click on the **gear menu** in the top-right corner.
2. Select **Scopes**. The **Scopes** page is displayed showing the root Scopes corresponding to Tenants+VRFs already created on the system.
3. Select a child scope in the scope directory. You can filter the scopes first if necessary.
4. Click the **Add** button.

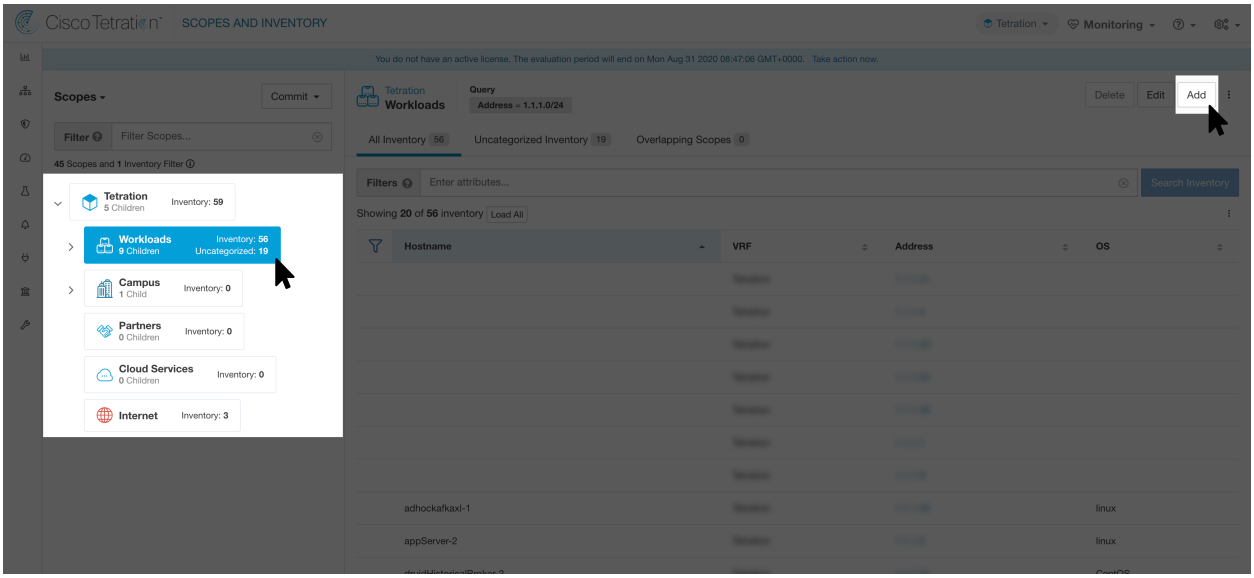


Fig. 4.1.1.5.1: Scope Add Button

5. Enter the appropriate values in the following fields:

Field	Description
Parent	The parent of the new Scope.
Name	The name to identify the Scope. Must be unique under the parent scope
Type	Select a category for the new Scope.
Query	The Query/Filter to be match the assets.

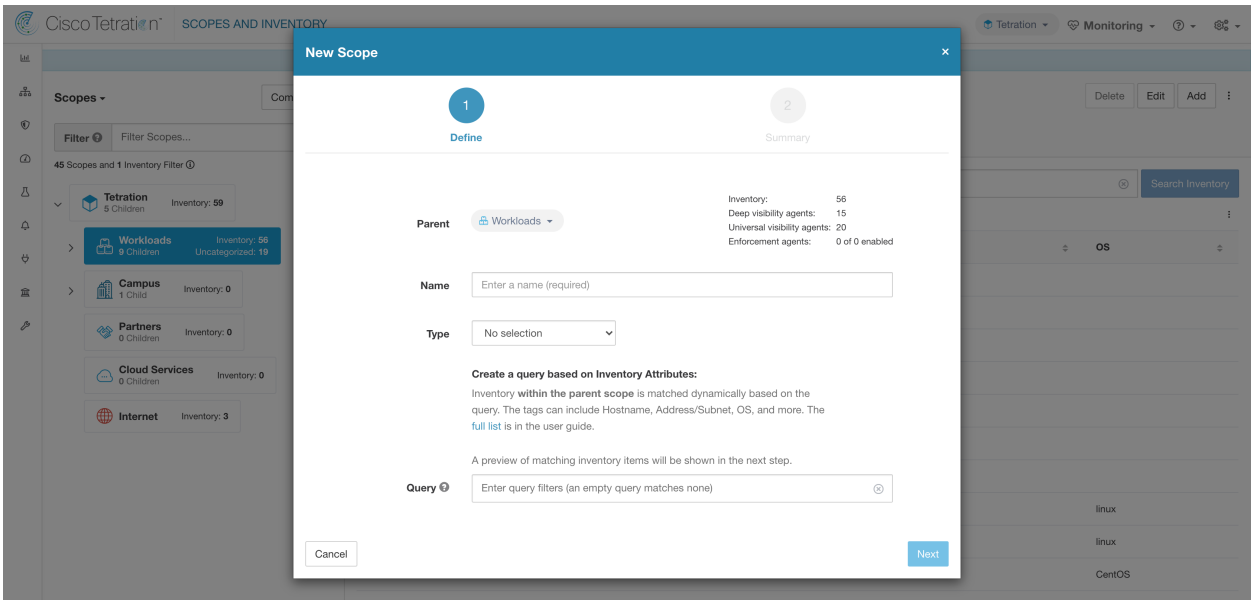


Fig. 4.1.1.5.2: Scope Create Modal

4.1.1.6 Scope Overlap

While adding scopes, it is recommended to avoid overlapping scopes. When scopes overlap, policies generated for overlapping scopes can potentially end up confusing end users. This feature proactively notifies the user if there are any overlapping scope membership, i.e., the same inventory belongs to more than one scope at the same depth in scope tree (sibling scopes). The goal is to avoid having the same workload exist in different parts of the scope tree.

To view which inventory items belong to multiple scopes, use the scope filter and enter the **Has Overlaps = true** facet.

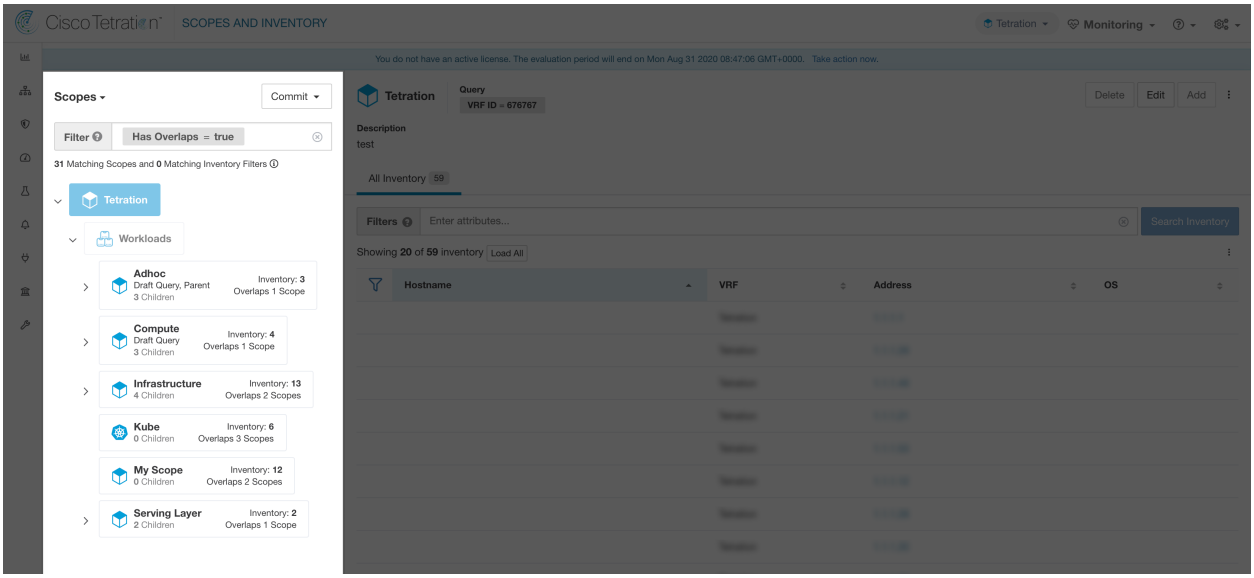


Fig. 4.1.1.6.1: Overlap facet in Scope filter

The list of overlapping scopes and the corresponding overlapping IP addresses can be viewed by traversing down the scope tree and selecting the **Overlapping Scopes** tab.

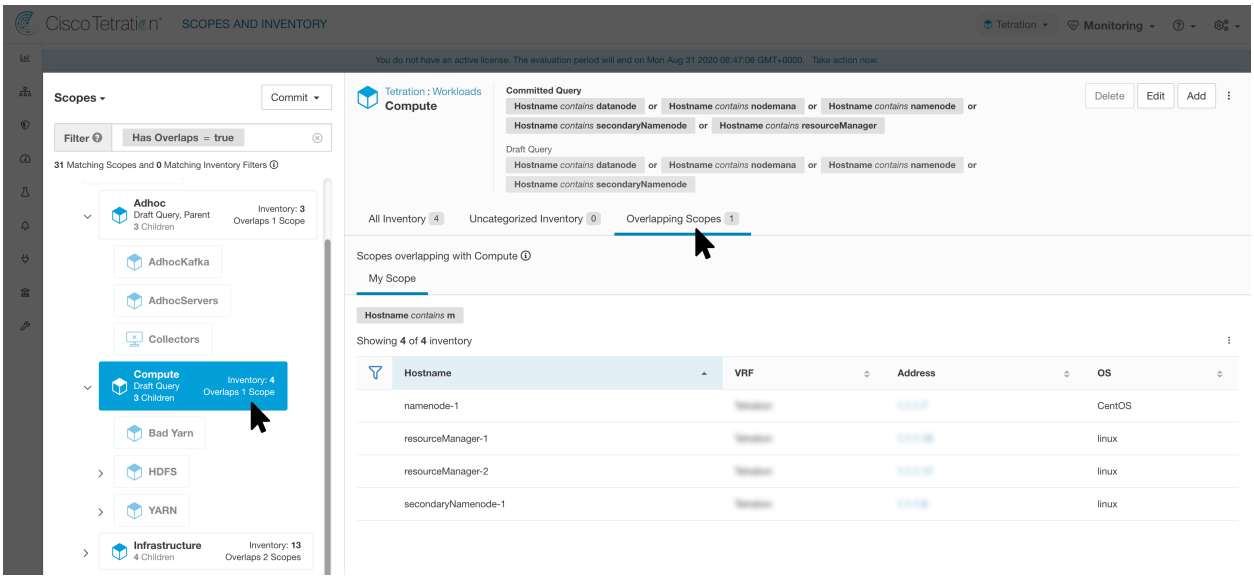


Fig. 4.1.1.6.2: Overlapping Scopes and IPs

4.1.1.7 Editing Scopes

Scopes can only be edited by users with the `SCOPE_OWNER` ability on the root scope. Site admins are owners of all scopes.

Editing a scope name

Editing a scope name happens immediately and can take several minutes depending on the number of child scopes that need to be updated.

Note: Flow searches by scope name will be impacted when changing the scope name.

Editing a scope query

When a scope's query is changed the direct parent and child scopes are impacted. Those scopes are marked as having 'draft changes' indicating changes have been made to the tree that have not been committed. Once all query updates have been completed, the user must click the **Commit Changes** button above the Scope Directory to make the change permanent. This will trigger a background task to update all of the scope queries and application 'dynamic cluster queries'.

Warning: Updating a scope query can impact application inventory membership. Changes will take effect during the **Commit Changes** process. To mitigate risks, you can compare membership changes for further impact analysis from the *Review Scope/Filter Change Impact* window.

New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

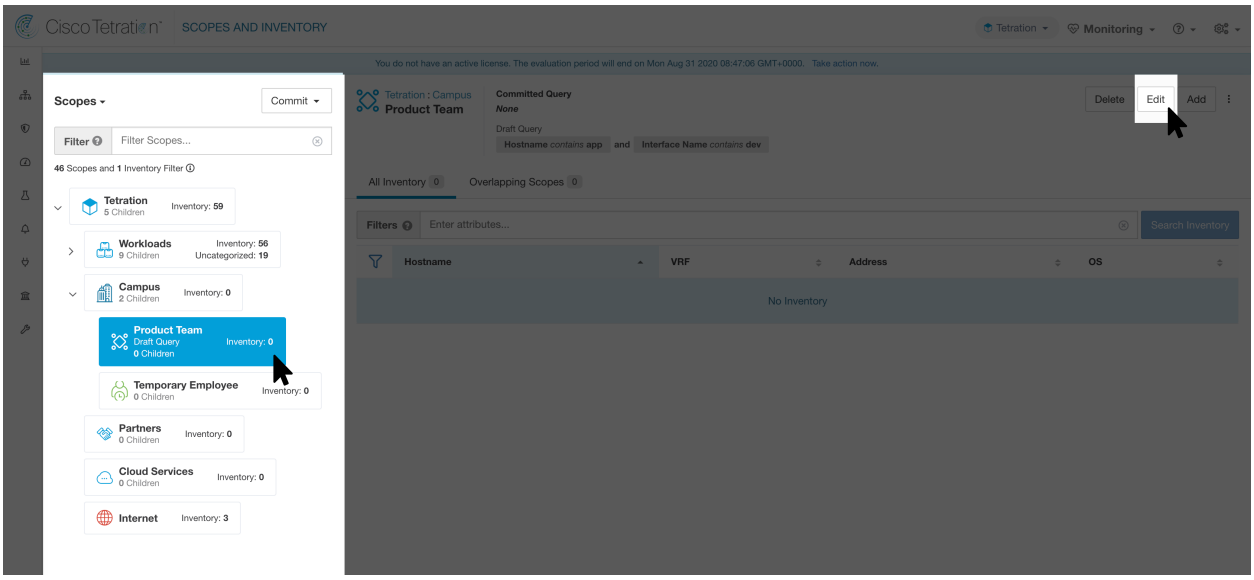


Fig. 4.1.1.7.1: Edit a Scope

To edit a scope:

1. Click on the **edit button** on the respective scope to be edited.
2. Edit the Name or Query for the selected scope.
3. Compare changes between the old and new Draft Query by following the **Review query change impact** link
4. Click on **Save**. Name gets updated right away.
5. To update the Query of all scopes, Click the **Commit Changes** button.
6. You will get a popup confirmation which states the consequences of performing scope changes. The update is processed asynchronously in a background task.
7. Click on **Save**. Depending on the number of changes this can a minute or more.

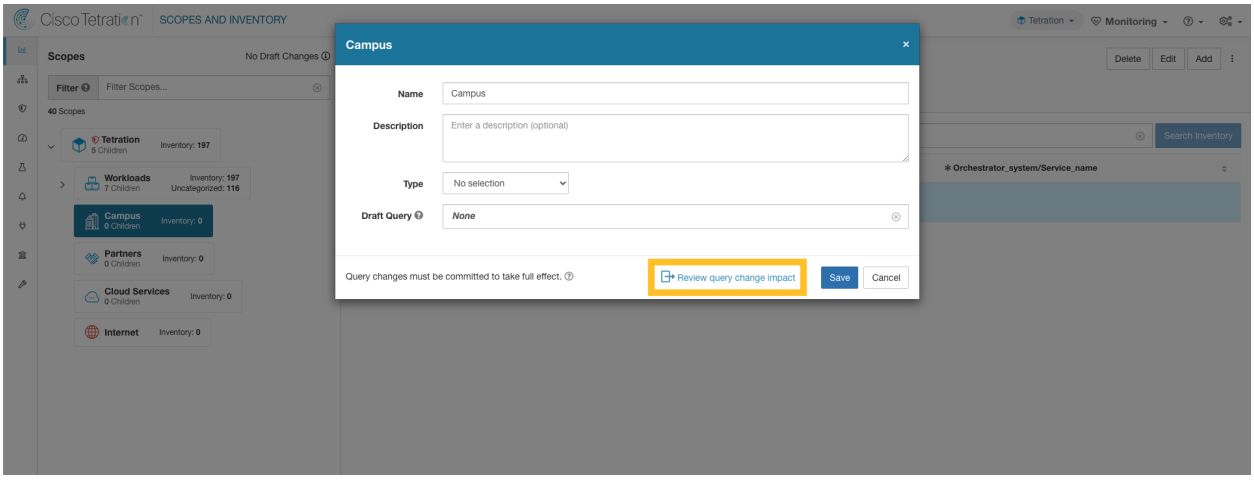


Fig. 4.1.1.7.2: Review query change impact

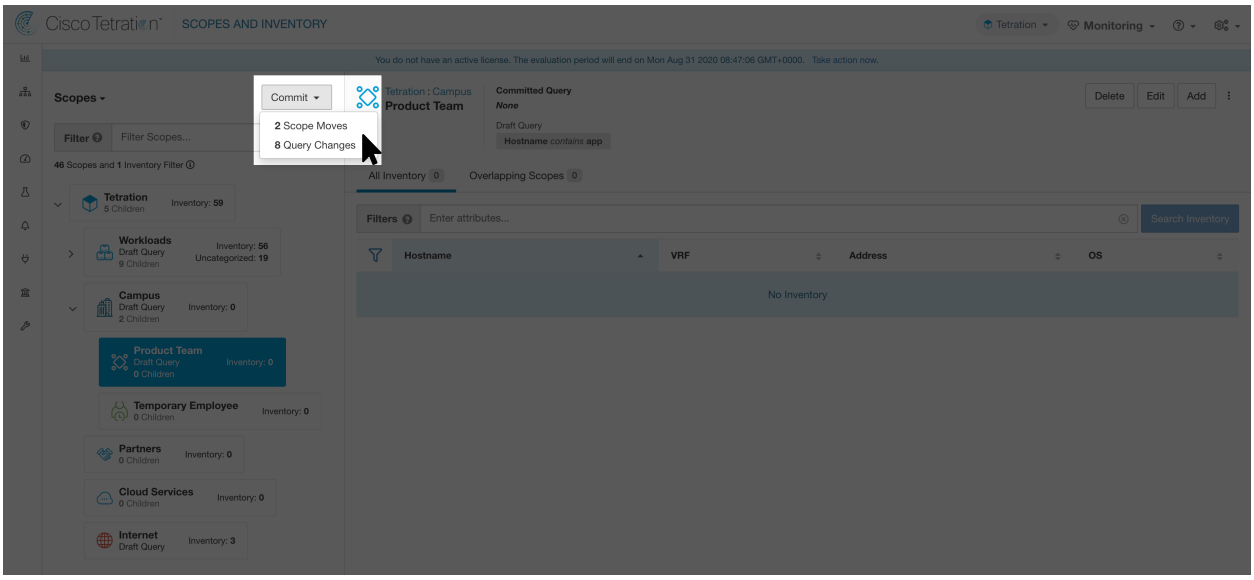


Fig. 4.1.1.7.3: Commit Changes

Editing the parent of a scope

When the parent of a scope is updated, the scope query changes. This change effects the membership of both the parent and child scopes. Similar to editing the scope query, these changes are initially saved as 'draft changes' and will not go into effect unless they are committed. The user can validate the impact of this change before committing by clicking on "Review query change impact" on the Edit Scope modal. Once validated, the changes can be committed by clicking "Commit" and accepting the "scope moves" and "query changes".

To edit the parent of a scope:

1. Click on the **edit button** on the respective scope to be edited.
2. Edit the parent for the selected scope.
3. Compare changes between the old and new Draft Query by clicking the **Review query change impact** link.
4. Click on **Save**.
5. Click on "Commit" and accept the 'scope moves' and 'query changes'. The update is processed asynchronously in a background task.
6. Depending on the number of workloads this change impacts, this can take a minute or more.

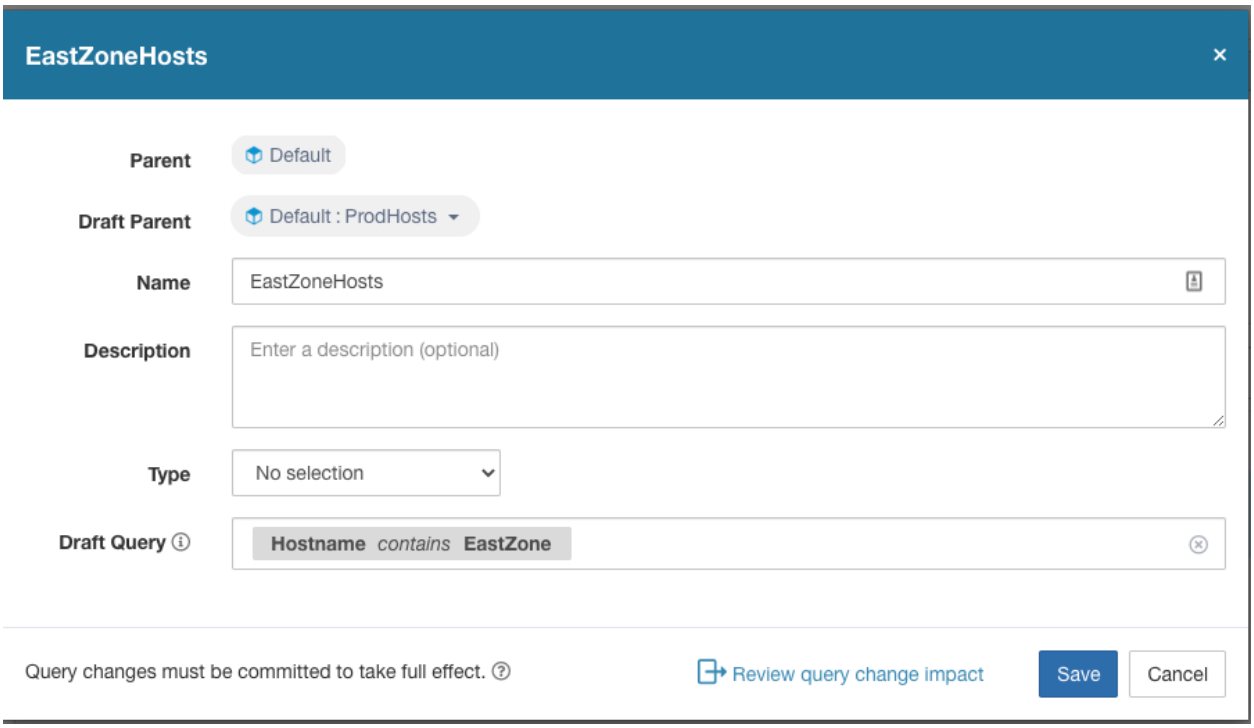


Fig. 4.1.1.7.4: Changing the parent scope from Default scope to Default:ProdHosts

4.1.1.8 Deleting Scopes

Scopes can only be deleted by users with the `SCOPE_OWNER` ability on the root scope. Site admins are owners of all scopes.

Deleting a scope will impact the application inventory membership of the parent. As a result, the parent scope will be marked as having 'draft changes'. The changes will need to be committed and dependent structures will need to be updated. See *Commit Changes*.

Scopes with dependent objects can not be deleted. An error will be returned if:

- An Application is defined on the Scope.
- There is an Inventory Filter assigned to the Scope.
- A policy exists that uses the Scope to define its consumers or providers.
- An Agent Config Intent is defined on the Scope
- An Interface Config Intent is defined on the Scope.
- A Forensics Config Intent is defined on the Scope.

To further drill down on scope dependencies, you can visit the **Dependencies** tab from the *Review Scope/Filter Change Impact* window.

These objects need to be removed before the Scope can be deleted.

1. Click on the **gear menu** in the top-right corner.
2. Select **Scopes**. The **Scopes** page is displayed.
3. Select a "scope" then click again to display child Scopes. Select the child scope you wish to delete.

4. Click the **Delete** button next to the edit and add buttons.

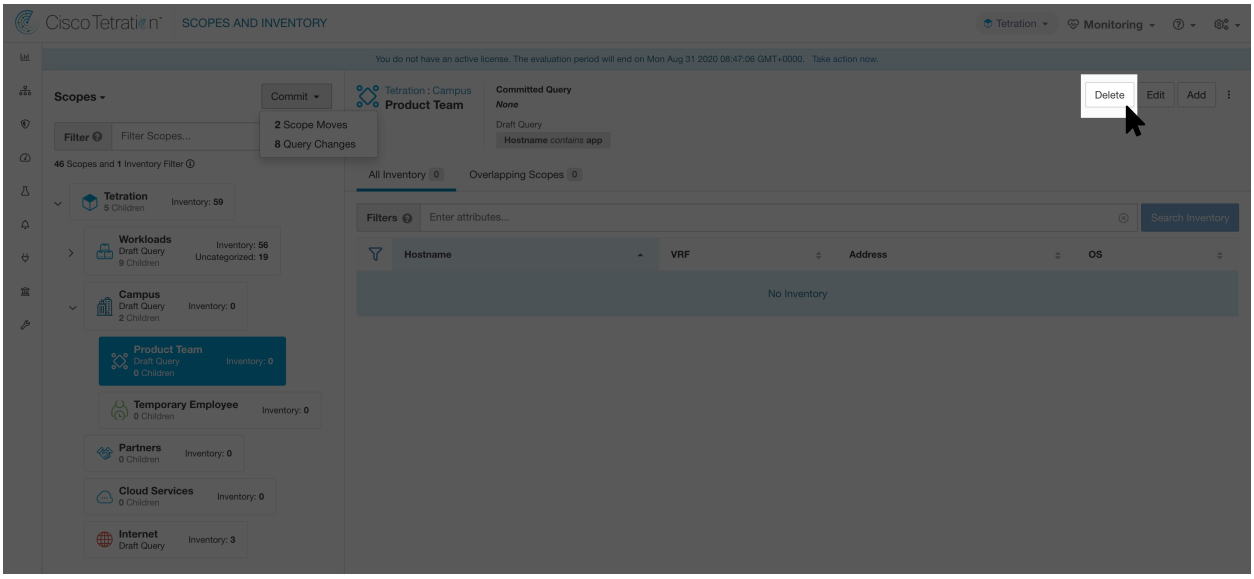


Fig. 4.1.1.8.1: Delete Scope

Note: Only Scopes without children can be deleted

Note: Root scopes must be deleted by removing the VRF from the Tenants page.

4.1.1.9 Commit Changes

A scope's application inventory query definition is defined by its query and those of its direct children. When this happens the scope is marked as having 'draft changes' and the scope's query, applications and clusters will not by changed until the **Commit Changes** background task is run. When a scope is in draft, the caution triangle is shown by the affected scopes icons, and the 'Commit Changes' button is shown on the Scopes page (top right) and should be clicked to run the **Commit Changes** background task.

Events that can mark a scope as in draft:

- query update,
- any parent's query was updated,
- direct child was added,
- direct child was deleted,
- direct child's query was updated.

Changing the name of a scope does not change the draft state of the scope.

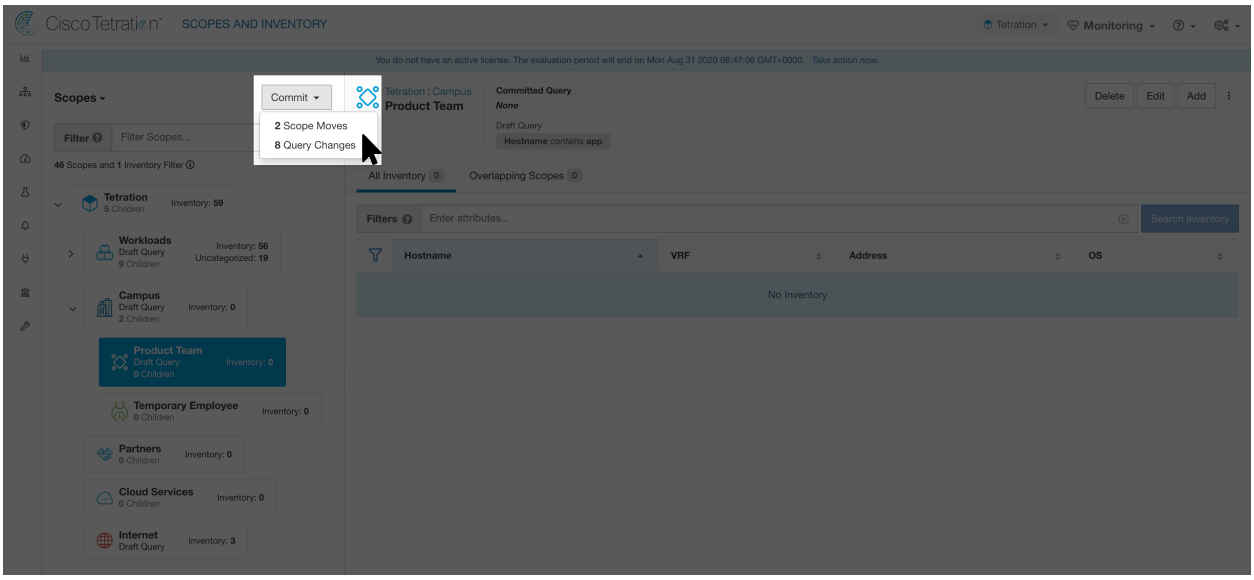


Fig. 4.1.1.9.1: Commit Changes

Note: The **Commit Changes** task is asynchronous. It usually takes several seconds but large scope trees can take several minutes.

Note: The scope update task will be completed when the root scope is no longer in draft. Refresh the page to get the latest state.

4.1.1.10 Change Log

Site Admins and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each scope by clicking change log in the overflow menu in the upper right.

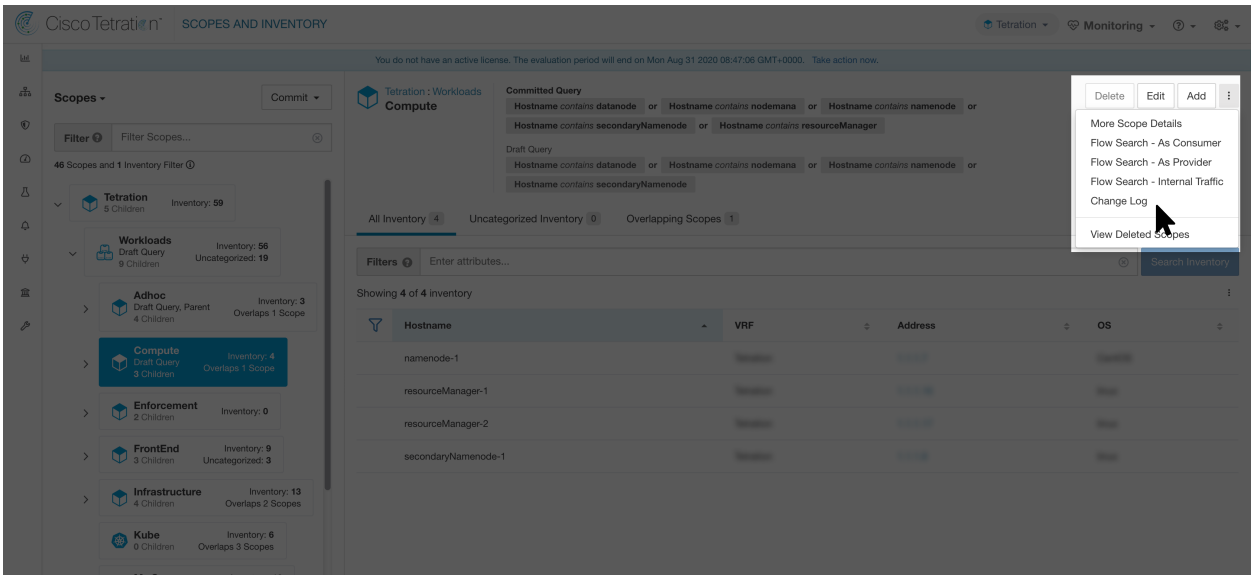


Fig. 4.1.1.10.1: Change Log

For more information on the **Change Log** see *Change Log*. Root scope owners are restricted to viewing change log entries for entities belonging to their scope.

These users can also view a list of deleted scopes by clicking on the **View Deleted Scopes** link is in the overflow menu in the upper right corner.

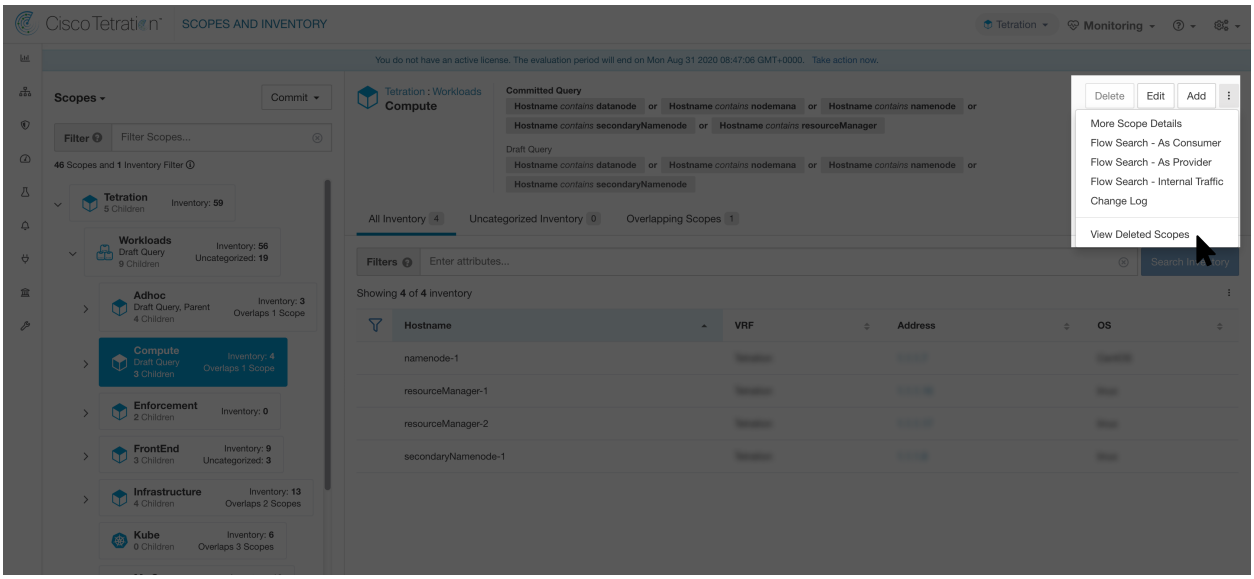


Fig. 4.1.1.10.2: View Deleted Scopes

4.1.1.11 Creating a New Tenant

Root level scopes map to VRFs that are created under *Tenants* or via the **Scopes** admin page. This action is only available to **Site Admins** and **Customer Support users**.

1. Click on the **gear menu** in the top-right corner.

2. Select **Tenants**.
3. Click the **Create New Tenant** button. The **New Tenant** modal appears.
4. Enter the appropriate values in the following fields:

Field	Description
Name	The name to identify the Scope. Must be unique under the parent Scope.
Description	An optional description.
Switch VRFs	Map multiple hardware (switch) VRFs to this Tetration tenant.

5. Click the **Create** button.

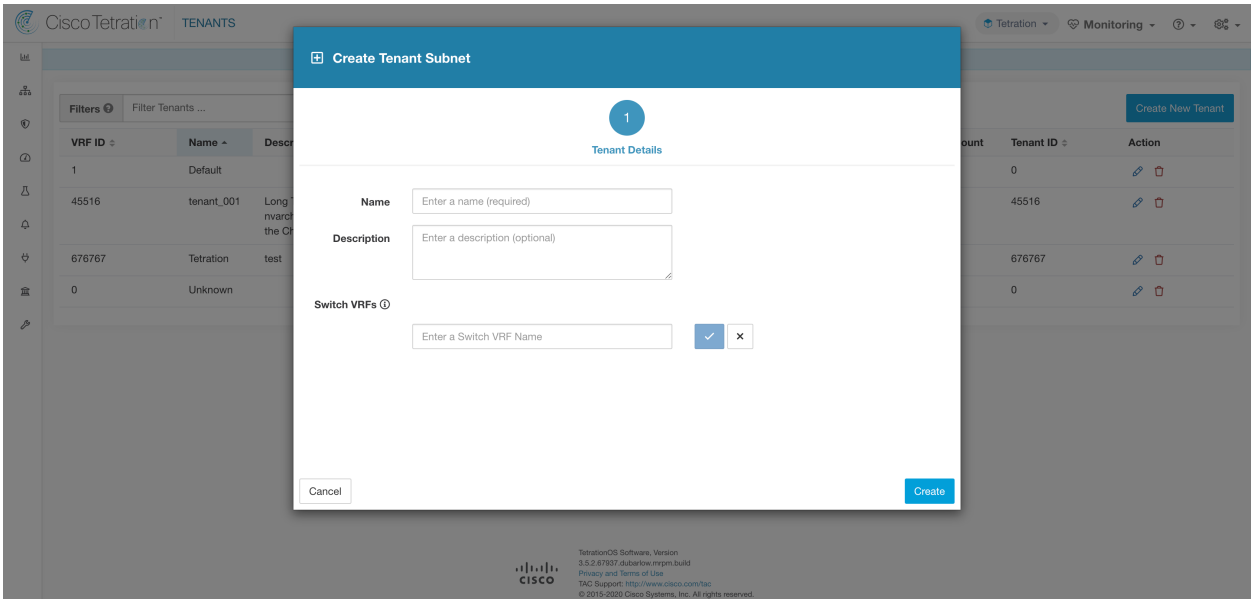


Fig. 4.1.1.11.1: Create Tenant

4.1.2 Inventory

All inventory detected on the network is searchable. You can search inventory via the sidebar: **Visibility > Inventory Search** from the top-level menu. This takes you to the **Scopes and Inventory** page. The summation of all inventory observed on the network after applying *Collection Rules* loads by default on the right side panel under the faceted input. Each inventory item is uniquely identifiable by IP and VRF and can be used for performing a search. A service inventory item is not searchable using its IP Address. Please use any of the User Labels associated to the service such as user_orchestrator_system/service_name for searching a service inventory. After a host has been found, you can view detailed information about the host on the host profile page.

Inventory Building Blocks

1. Root Scope
   - Root of the scope hierarchy under a given tenant
   - Provides a logical separation for L3 address domains
2. Scope
   - Inventory container defined by dynamic query
   - Foundation for hierarchical policy model

- Anchor point for policy, RBAC and filter configuration

3. Filter
   - Flexible construct based on dynamic inventory query
   - Anchor point for intent definition, provided services, and policy definition

Note: Includes all IP addresses from partners and anything that is communicating in your environment. Whether they have an agent on them or not, you should define what they are through label.

Label Planning Considerations

1. Source of data
   - Networks - IPAM? Routing tables? Spreadsheet?
   - Hosts - CMDB, Hypervisor, Cloud, App Owners?
2. Accuracy of data
3. How dynamic the data is and how it will be updated
   - Manual Upload?
   - API Integration?
4. Start with the basics and grow
   - Use network labels to build high-level scope structure
   - Use host labels to build more detailed scope structure at app level

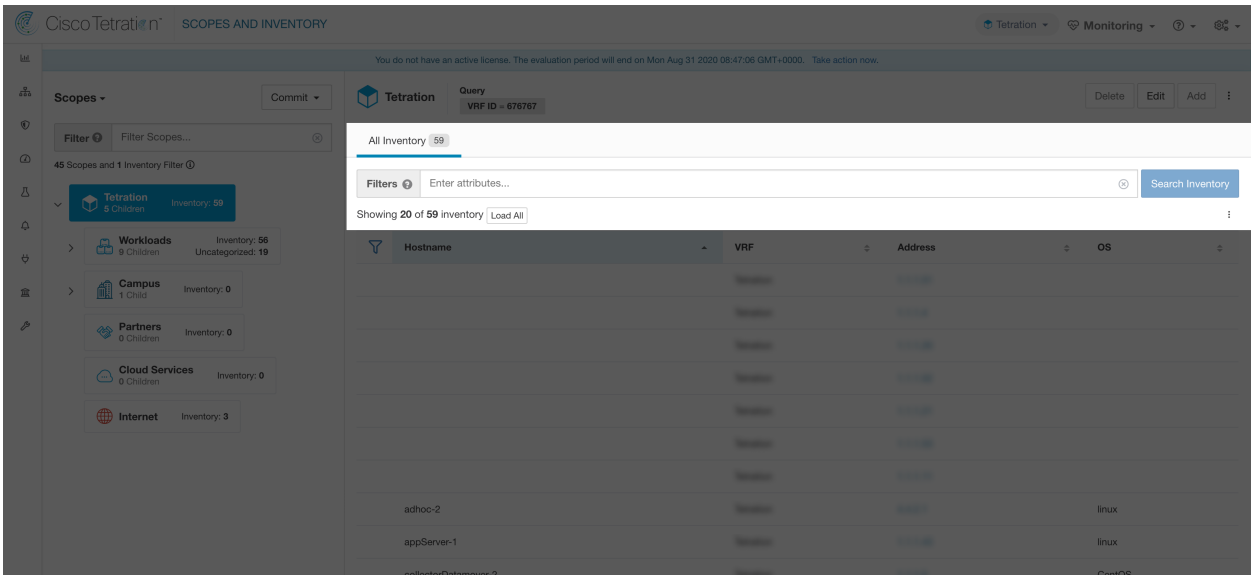


Fig. 4.1.2.1: Inventory Top-level Menu Options

4.1.2.1 Searching Inventory

Searching inventory enables you to view information about specific inventory items. The search results are grouped into four tabs: Services, Pods, Workloads and IP Addresses. Workloads consist of inventory items reported by Tetration agents, IP Addresses consist of inventory items discovered through Inventory Upload and flows, Services consist

of Kubernetes services and Load Balancers discovered through External Orchestrators and Pods consists of Kubernetes pods. The Services and Pods tabs are hidden unless a related external orchestrator is configured. There is also a mention of the inventory count next to each tab. The immediately available information in a search includes hostname, IP Address, OS, OS Version, Service Name and Pod Name. The list of displayed columns can be toggled by clicking the funnel icon in the table header. Search results are restricted to the currently selected scope shown in the scope directory. More information can be seen on the respective profile page by clicking on an item in the search results.

1. From the top-level menu, select **Visibility > Inventory Search**. The **Scopes and Inventory Search** page appears.
2. Enter the attributes in the **Filters** field for the inventory item you are looking for. The attributes include the following:

Attributes	Description
Hostname	Enter a full or partial hostname.
VRF Name	Enter a VRF name.
VRF ID	Enter a VRF ID (numeric).
Address	Enter a valid IP address or subnet (IPv4 or IPv6).
Address Type	Enter either IPv4 or IPv6.
OS	Enter an OS name (e.g. CentOS).
OS Version	Enter an OS version (e.g. 6.5).
Interface Name	Enter an interface name (e.g. eth0).
MAC	Enter a MAC address.
In Collection Rules?	Enter true or false.
Process Command Line	Enter the sub-string of a command that is running on host (Note: this facet cannot be saved as part of inventory filter)
Process Binary Hash	Enter the process hash of a command that is running on host (Note: this facet cannot be saved as part of inventory filter)
Package Info	Enter the package name optionally followed by a package version (prefixed by #)
Package CVE	Enter part of or a complete CVE ID
CVE Score v2	Enter a CVSSv2 (Common Vulnerability Scoring System) score (numeric).
CVE Score v3	Enter a CVSSv3 (Common Vulnerability Scoring System) score (numeric).
User Labels	Attributes prefixed with come from user labels.

3. Click **Search Inventory**. The results are displayed below the **Filters** field grouped into four tabs. Each tab displays a table with the relevant columns. Additional columns can be displayed by clicking on the funnel icon in the table header. If any user labels are available, they will be prefixed with and can be toggled here.

More details about each host is displayed on the **Workload Profile**, which is accessible by clicking on the IP address field of a search result row. See the *Workload Profile* for more information.

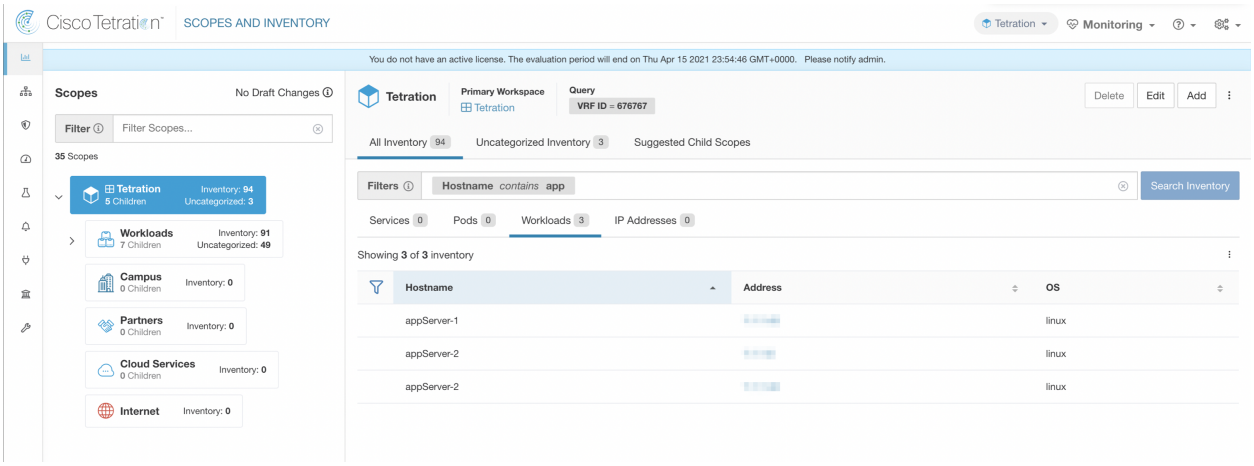


Fig. 4.1.2.1.1: Inventory Search Results

To create Inventory Filters via the sidebar: **Visibility > Inventory Filters** from the top-level menu. Click on the **Create Filter** button. A modal dialog will appear where you can give your saved filter a name.

4.1.3 Suggest Child Scopes

Suggest Child Scopes is a tool that uses machine learning algorithms (such as community detection in networks) to discover groupings that could serve as scopes. This tool is helpful when building a scope hierarchy, and facilitates the process of defining more granular child scopes for a given scope. Candidate child scopes are shown as suggestions that can then be selected and added.

A description of the algorithms at a conceptual level: A graph based on the communications among the unclaimed members of the parent scope is first created (note: unclaimed members are those that do not belong to any child scope of the parent), and the graph is preprocessed, for example the algorithms attempt to identify endpoints that communicate with sufficiently high proportion of other endpoints in the graph. Such a group of endpoints, if found, is displayed to the user as a candidate **common services** grouping. The rest of the graph is processed to detect groups that behave as **communities**, meaning roughly that the endpoints disproportionately communicate with one another more often (or on more provider ports) than to endpoints outside the group. Each such grouping may correspond to an application or a department within the organization. Such a partitioning can also lead to sparser policies among scopes.

Example:

Let 1 through 10 be individual endpoint IPs. Assume the input (communications) graph is as follows:

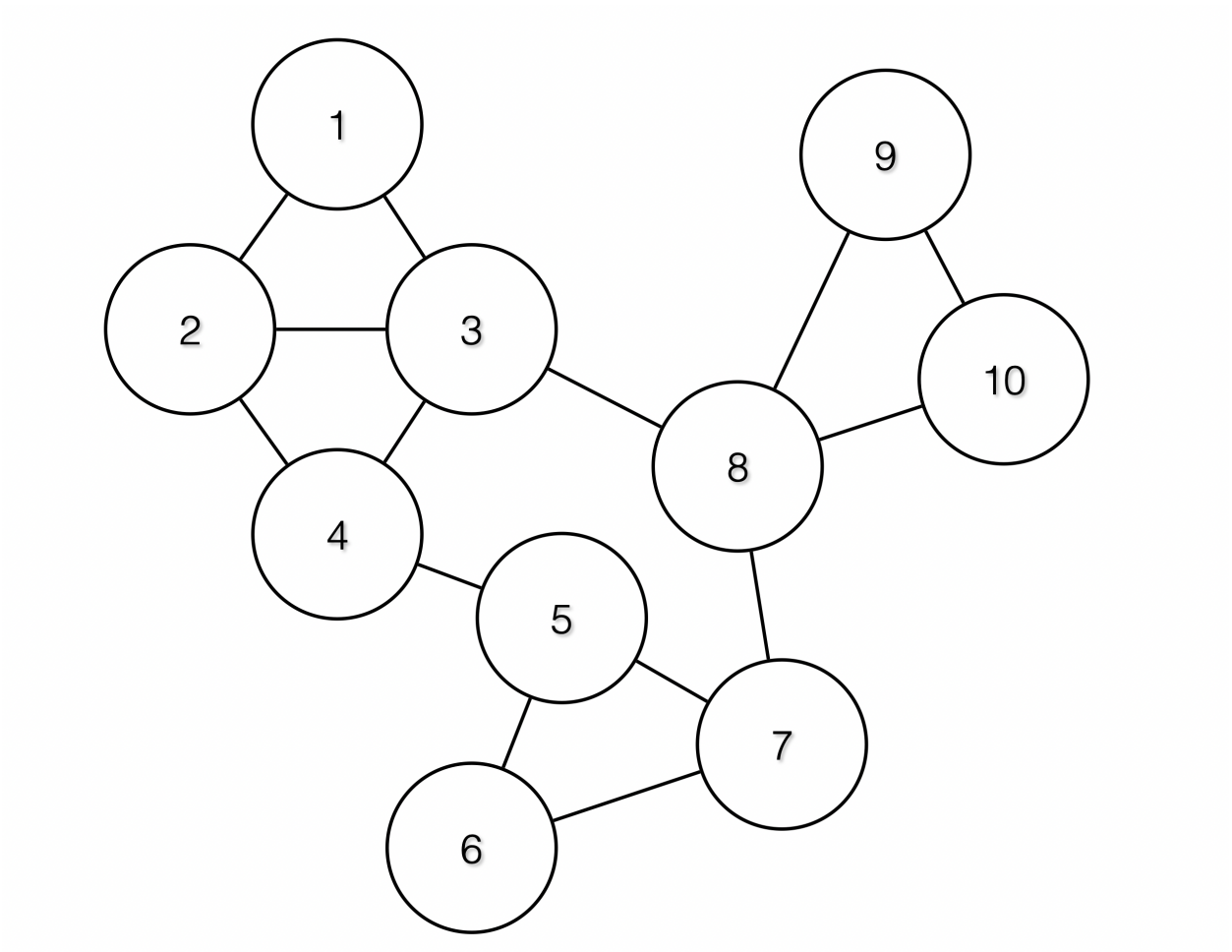


Fig. 4.1.3.1: Input graph

Then the endpoints 1 - 4, 5 - 7 and 8 - 10 will be grouped together because they have relatively high degree of communication (number of edges) among one another, and relatively low communications to other endpoints.

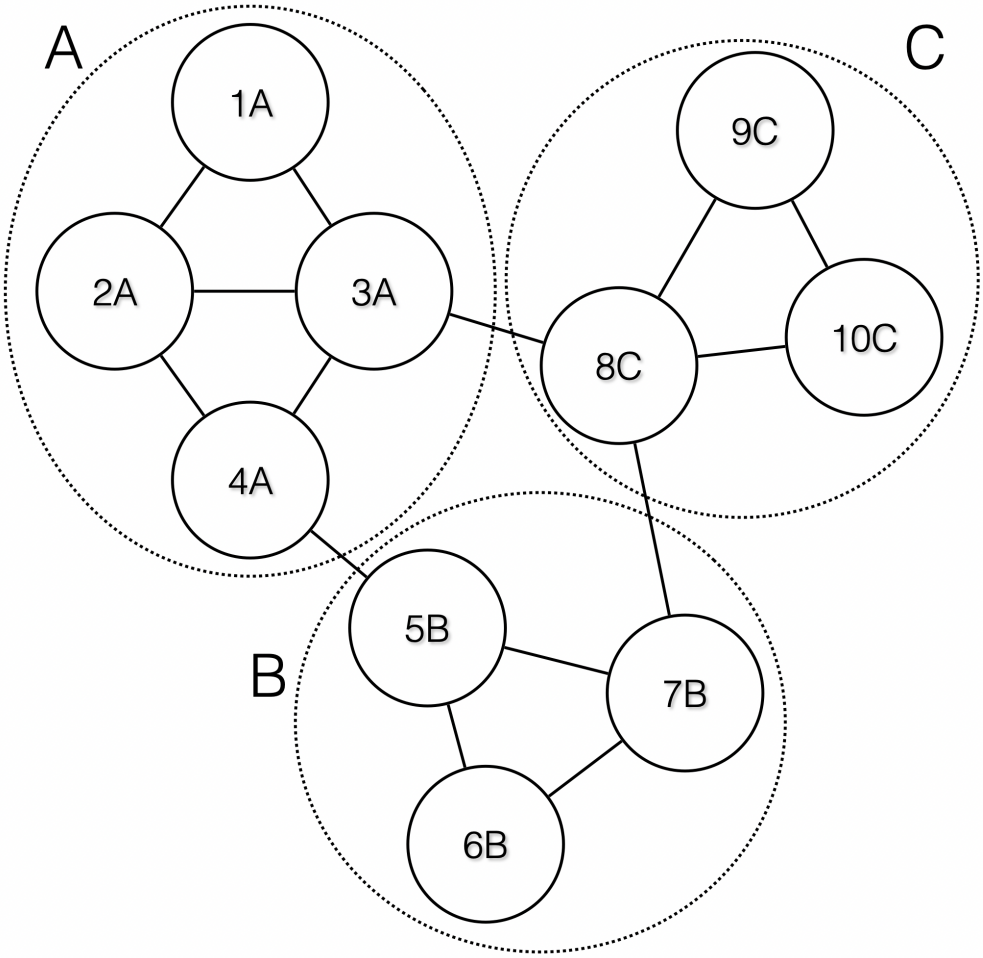


Fig. 4.1.3.2: Output groups

4.1.3.1 Steps to perform scope suggestion

To invoke scope suggestion for a desired scope user should locate on the scopes page and select it.

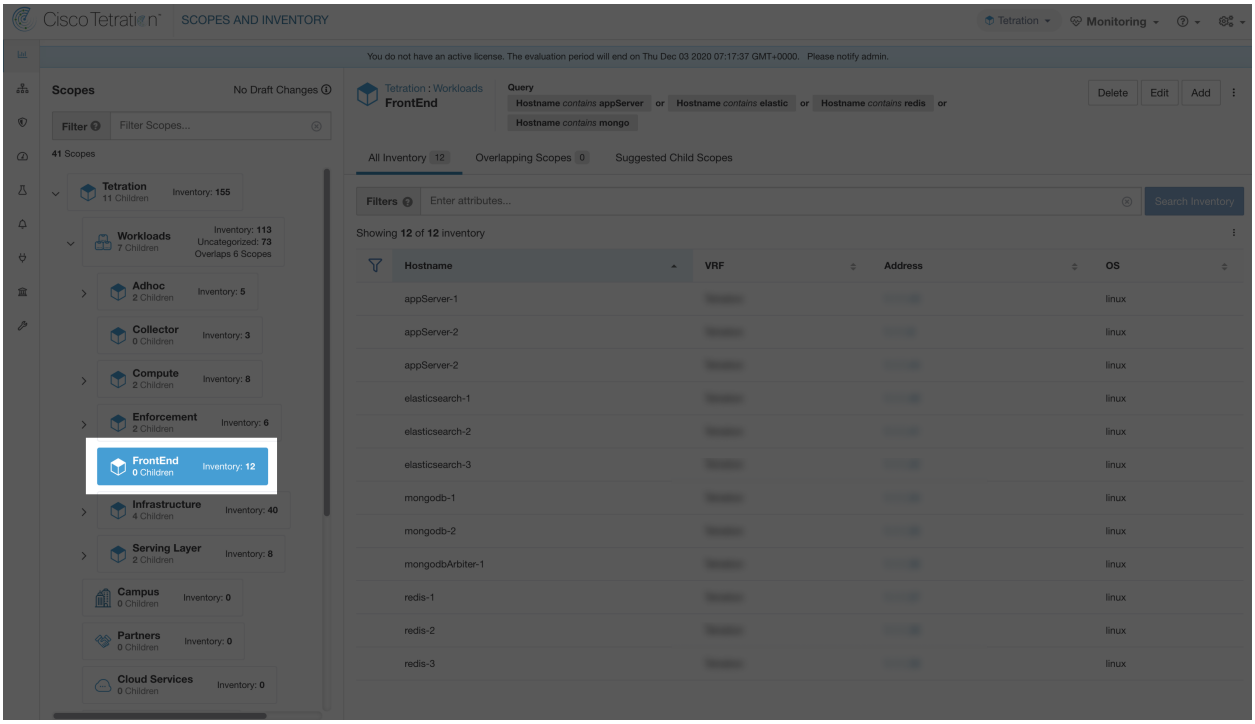


Fig. 4.1.3.1.1: Example of selecting a scope

In the window, user can browse the inventory, *uncategorized inventory items*, i.e. those items that belong to the current selected scope and that do not belong to any of the current selected scope's child scopes. Clicking on the **uncategorized inventory items** allows one to view this list.

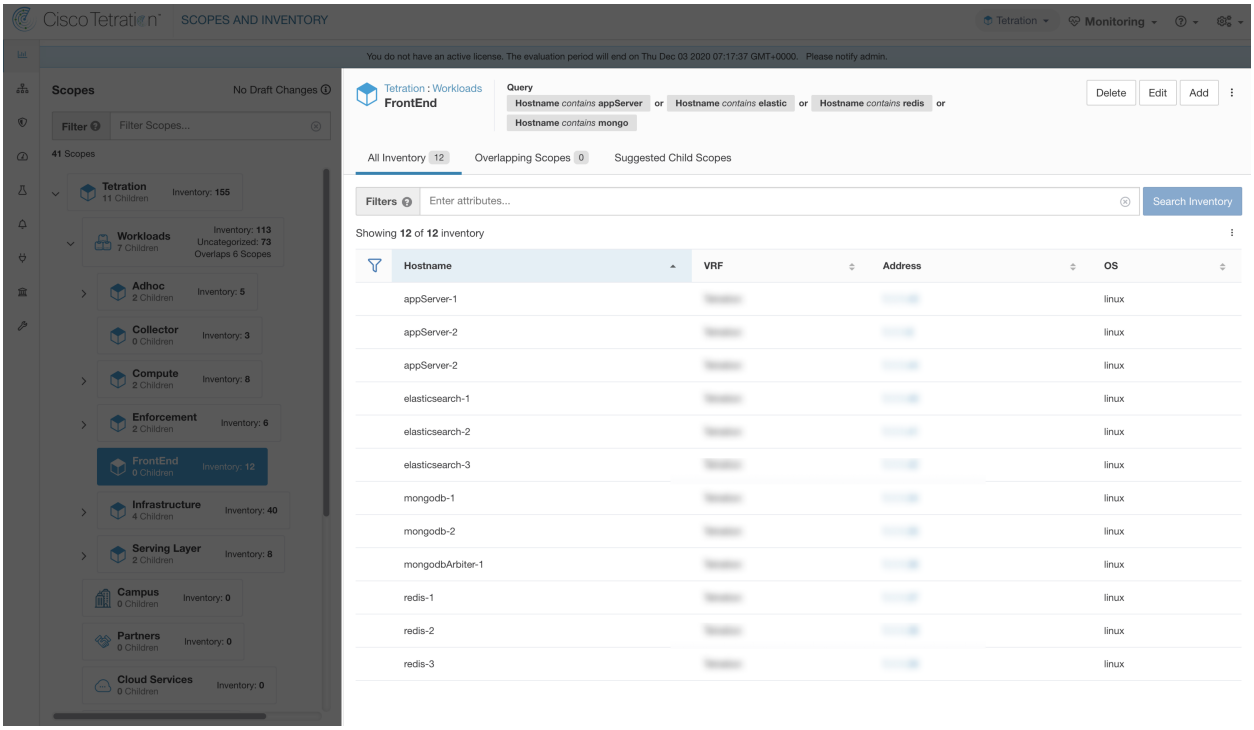


Fig. 4.1.3.1.2: Example of scope window

After selecting the scope user can click on **Suggest Child Scopes**, and click on **Start Scope Suggestion** (or click on Rerun, in case this is not the first time). Note that the input for a scope suggestion run will be the uncategorized inventory items.

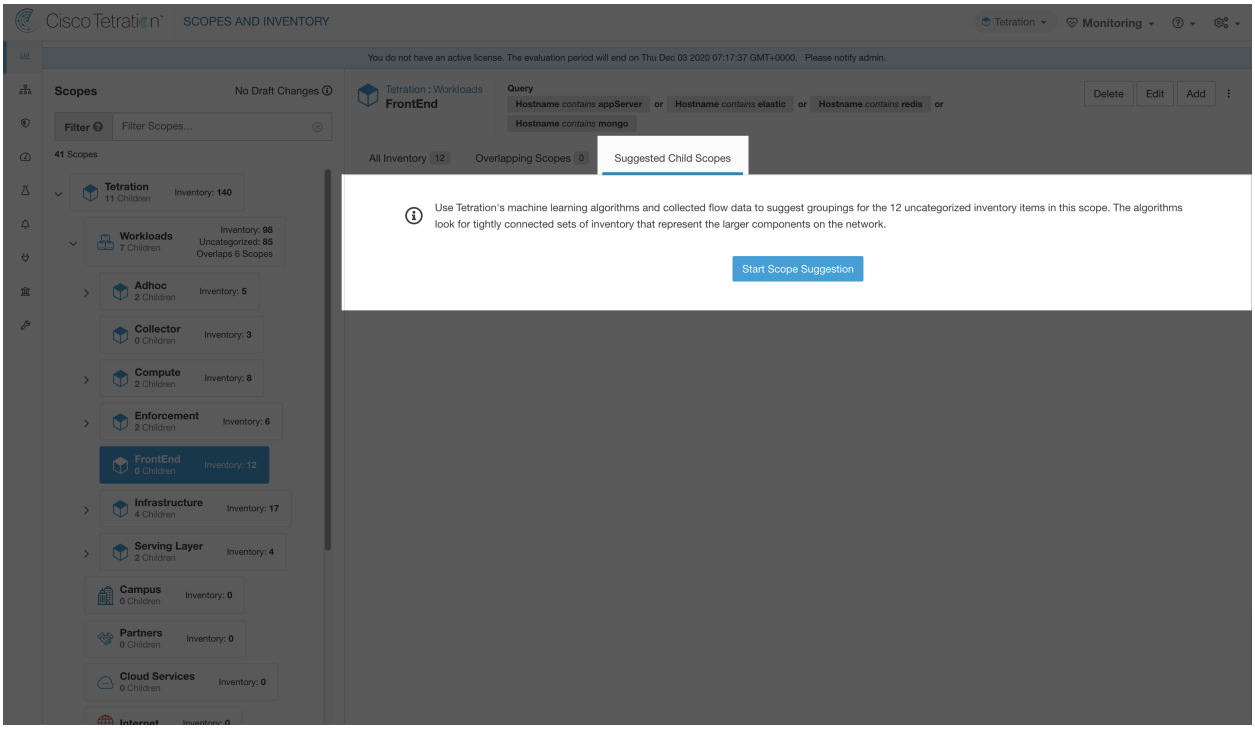


Fig. 4.1.3.1.3: Example of scope **Suggest Child Scopes** tab

User can set the date range as input for scope suggestion and click on **Suggest Scopes**. A scope suggestion run is often fast under medium overall load, and takes only a few minutes for processing ten to thousands of endpoints, with tens of thousands of conversations.

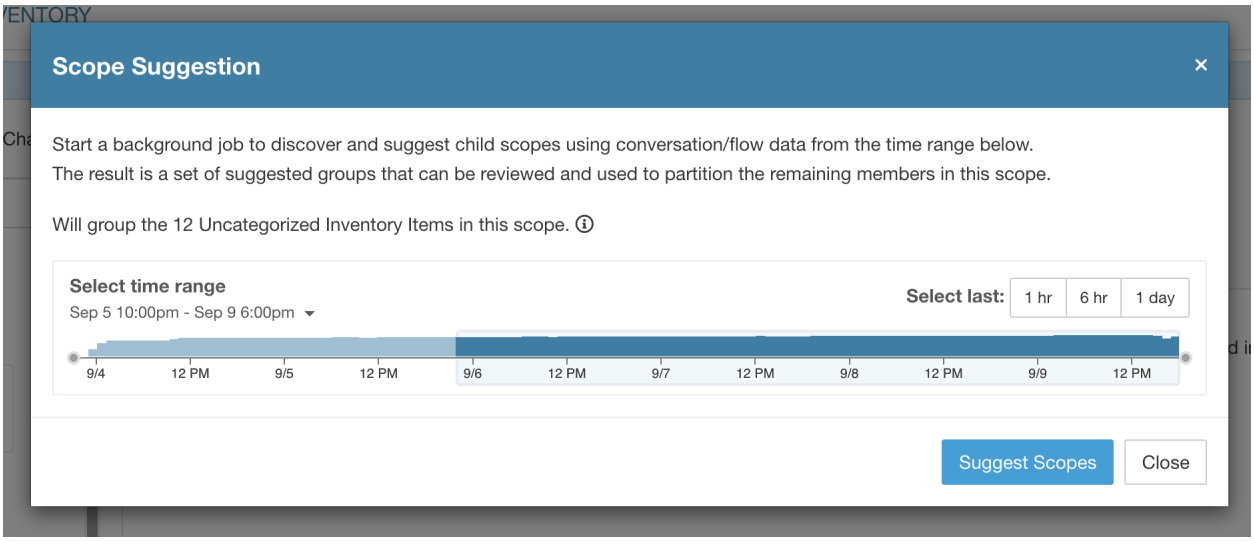


Fig. 4.1.3.1.4: Example of scope suggestion data range selector

The output is shown to the user as a list of candidates, currently up to 20 groups (shown), each accompanied with information such as group confidence (quality), a candidate scope name, and queries. Each discovered group has an associated **Group Community Confidence**, the possible values being: **Very High**, **High**, **Medium** and **Low**. This is a measure of the **Community** property of the group: the higher the confidence, the higher the community property

of the given group of endpoints (many edges inside the group, relatively few edges to outside). Currently, the subset of groups picked to be shown are selected based on the Group Community Confidence. The groups discovered can currently fall under one of these four group types:

- **Generic Group**: Any group discovered via machine learning based on the community property. Note that any group that is not explicitly designated with the special types below is a generic group.
- **Common Service**: This group consists of endpoints that communicate with much of the input inventory. These endpoints could be running some kind of shared service(s).
- **Common Service Clients**: This group consists of endpoints that only communicate with the **Common Service** group.
- **Ungrouped**: This group consists of endpoints that cannot be grouped since they don't have sufficient communications.

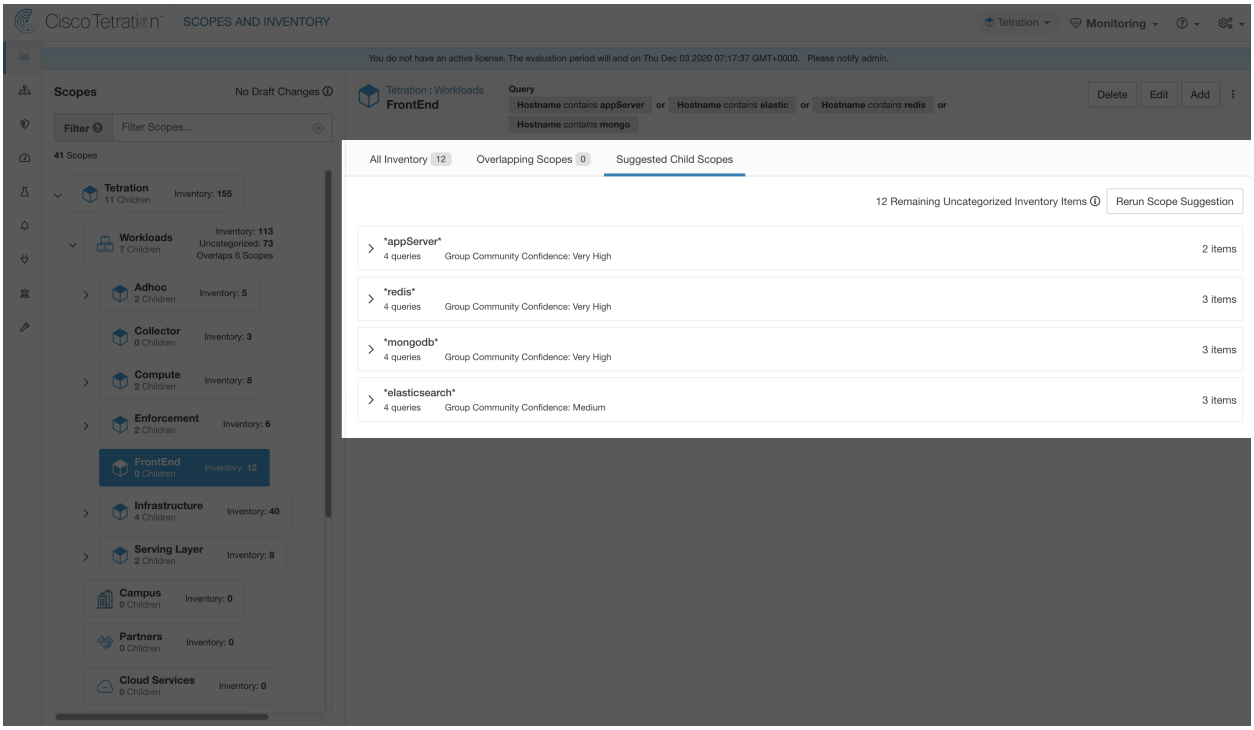


Fig. 4.1.3.1.5: Example of scope suggestion output

The user can click on a discovered group to view the list of queries generated for the selected group. The user can preview the inventory covered by the query which will closely define the discovered group. The queries consist of IP-ranges, subnets, host names and user uploaded labels. There is a confidence measure associated with each group called **Query confidence** which can have one of the following range of values **Perfect**, **Very High**, **High**, **Medium** and **Low**. For query generation, first the groups are discovered via graph processing and machine learning, then the queries are generated for each group. **Query Confidence** is a measure of how well the query can cover the endpoints. A query confidence of **Perfect** indicates that the query exactly covers the suggested (discovered) group. On the other end of the spectrum, a **Low** query confidence indicates that the query significantly misses out on exactly capturing the suggested group, which means that the query covers many **Extra IPs** (not part of the discovered group) and/or has many **Missing IPs** (not covered by the query).

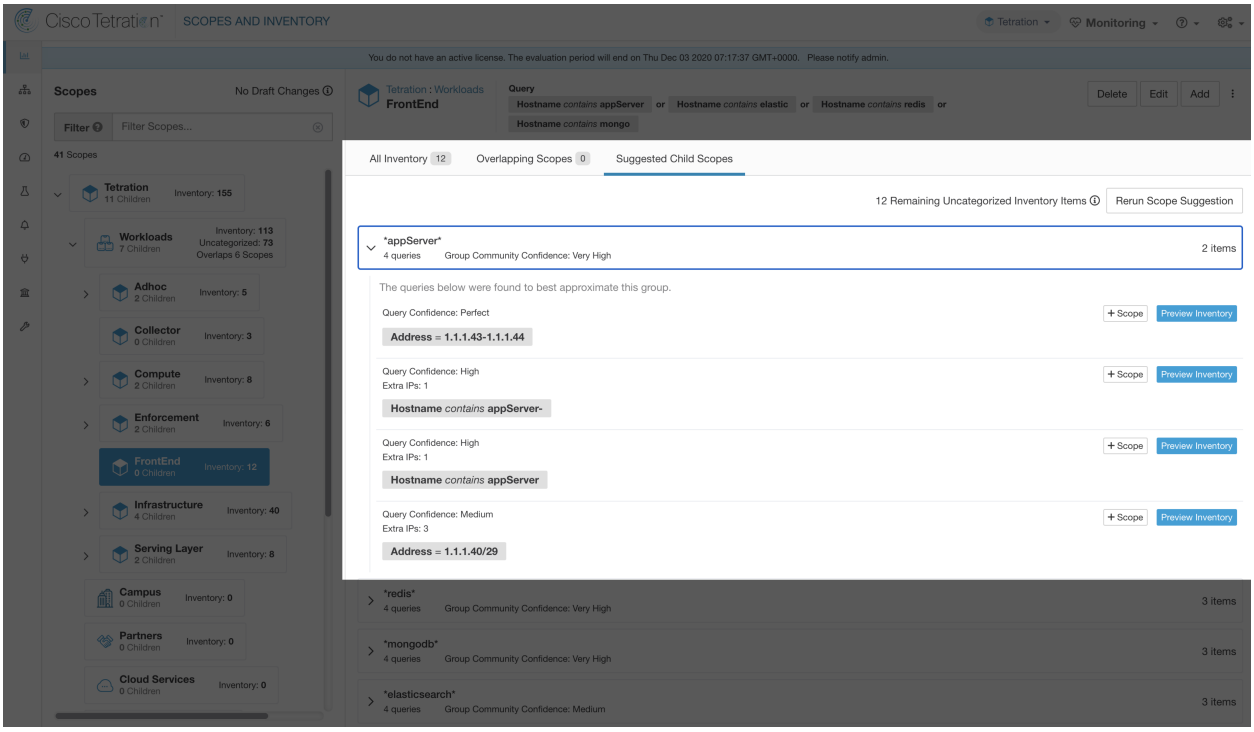


Fig. 4.1.3.1.6: Example of scope suggestion output queries

The user can click on **+ Scope** button which will take the user to an edit window where the user can edit the group name and group query. The user can examine a query, the IPs that it matches, and decide whether some IPs need to be added or removed by adjusting the query. Once satisfied, the user can then click on **Next**, to review and convert the group to a scope on the draft view canvas.

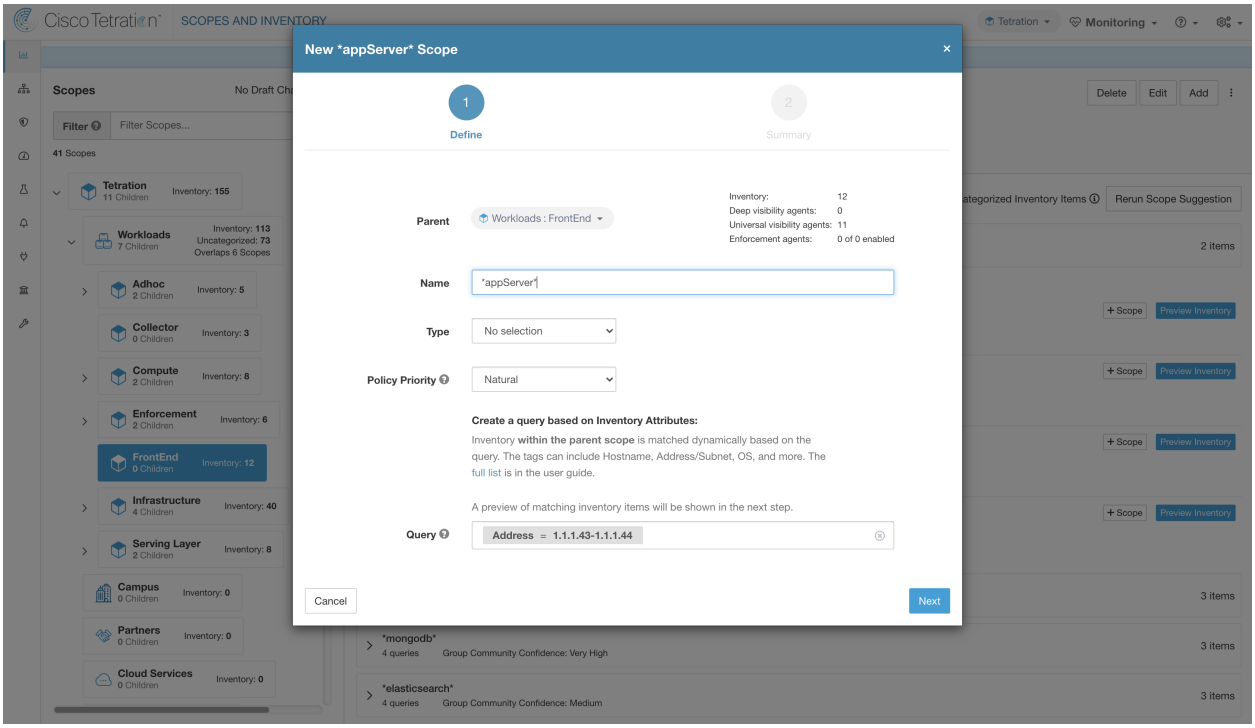


Fig. 4.1.3.1.7: Example of scope suggestion edit window

After the user has converted a suggested group to a scope, the group slot turns green and the **Uncategorized Inventory Items** count decreases.

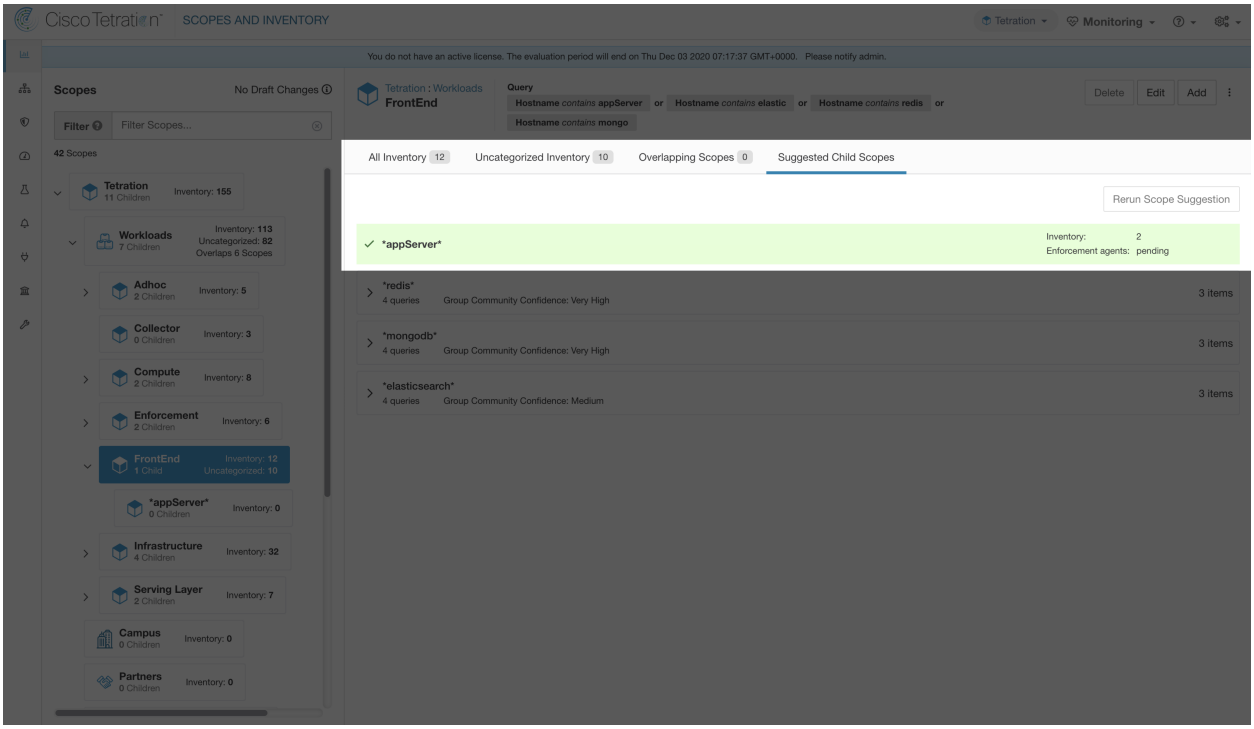


Fig. 4.1.3.1.8: Example of scope suggestion output after converting one suggested group to a scope

The user can repeat the process of scope creation from the remaining list of groups. The recommended workflow is to create one or more scopes and then re-run **scope suggestion**. A zero count for **Uncategorized Inventory Items** indicates that there is no inventory left to be further scoped (for the currently selected parent scope).

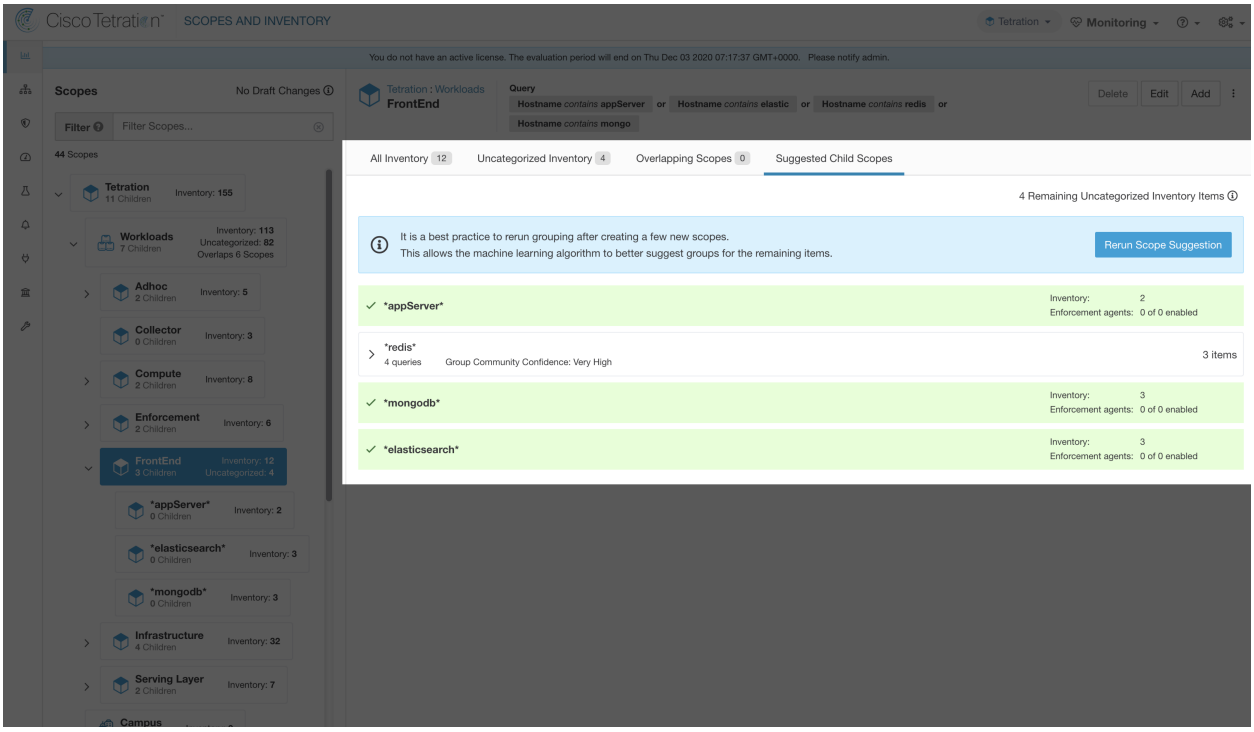


Fig. 4.1.3.1.9: Example of scope suggestion output after multiple scope creations

After the scope creation process is done (the uncategorized count is 0), user can repeat this process on the newly created child scopes in order to generate a deeper scope tree as desired.

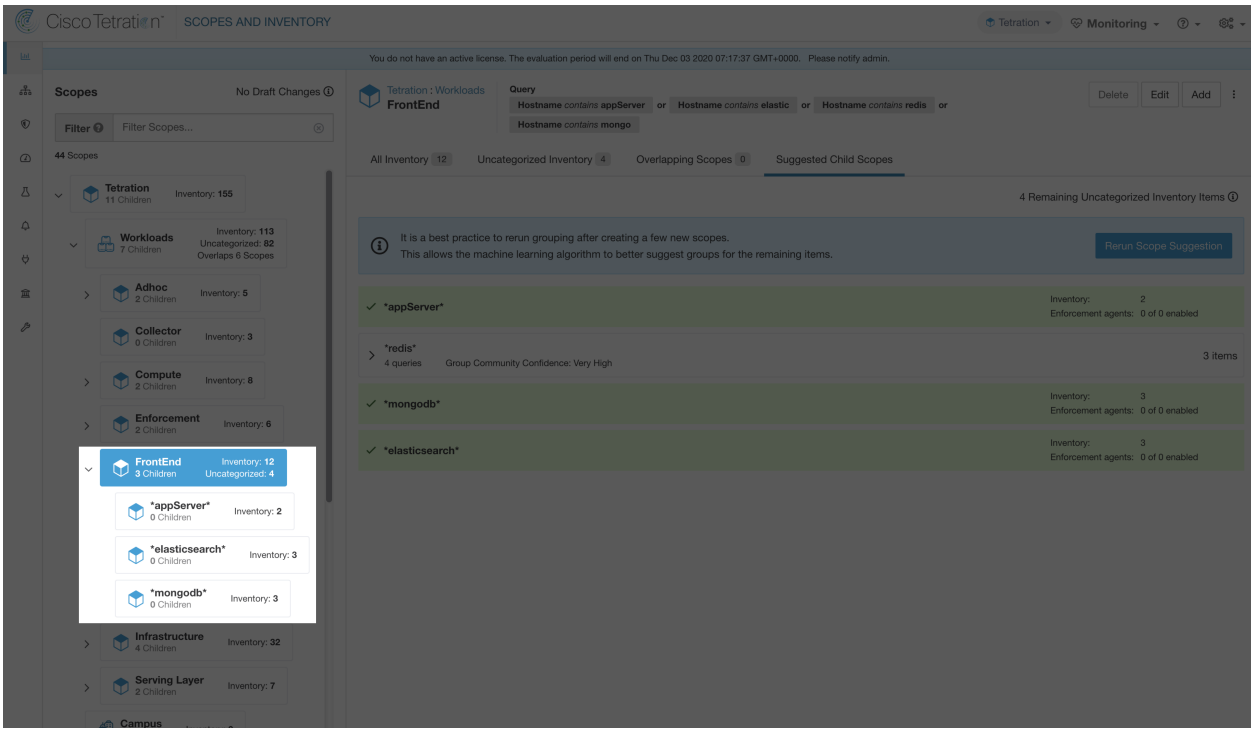


Fig. 4.1.3.1.10: Example of the scopes list after the initial scope suggestion and creation

Note: There is also a possibility that the uncategorized items in a scope do not partition well (e.g., do not form communities). In that case, the algorithm may return no groupings (an empty result).

4.2 Filters

Filters are saved inventory searches that can be used when defining policies, config intents, etc. Each filter must be associated with a scope, which is defined as the filter's ownership scope. You can view existing filters by selecting **Inventory -> Filters** from the top-level menu.

The list of filters are restricted based on the root of the currently selected scope.

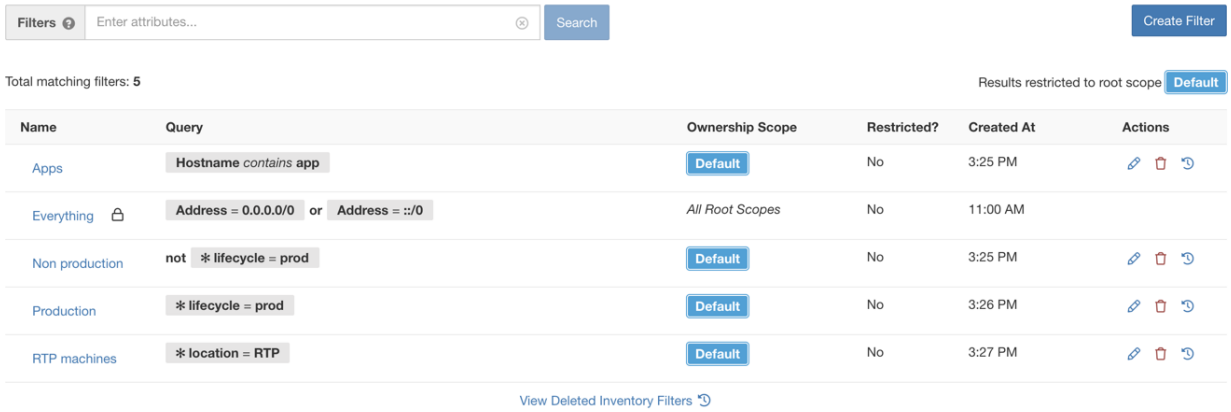


Fig. 4.2.1: Inventory filters

New filters can be created by clicking the **Create Filter** button. A modal dialog will appear where you can give your saved filter a name. The ownership scope can also be changed (it defaults to the currently selected scope). If you would like the filter query to be restricted to the ownership scope, select the **Restrict to ownership scope?** checkbox (see below for more information). Click **Save** to save the filter.

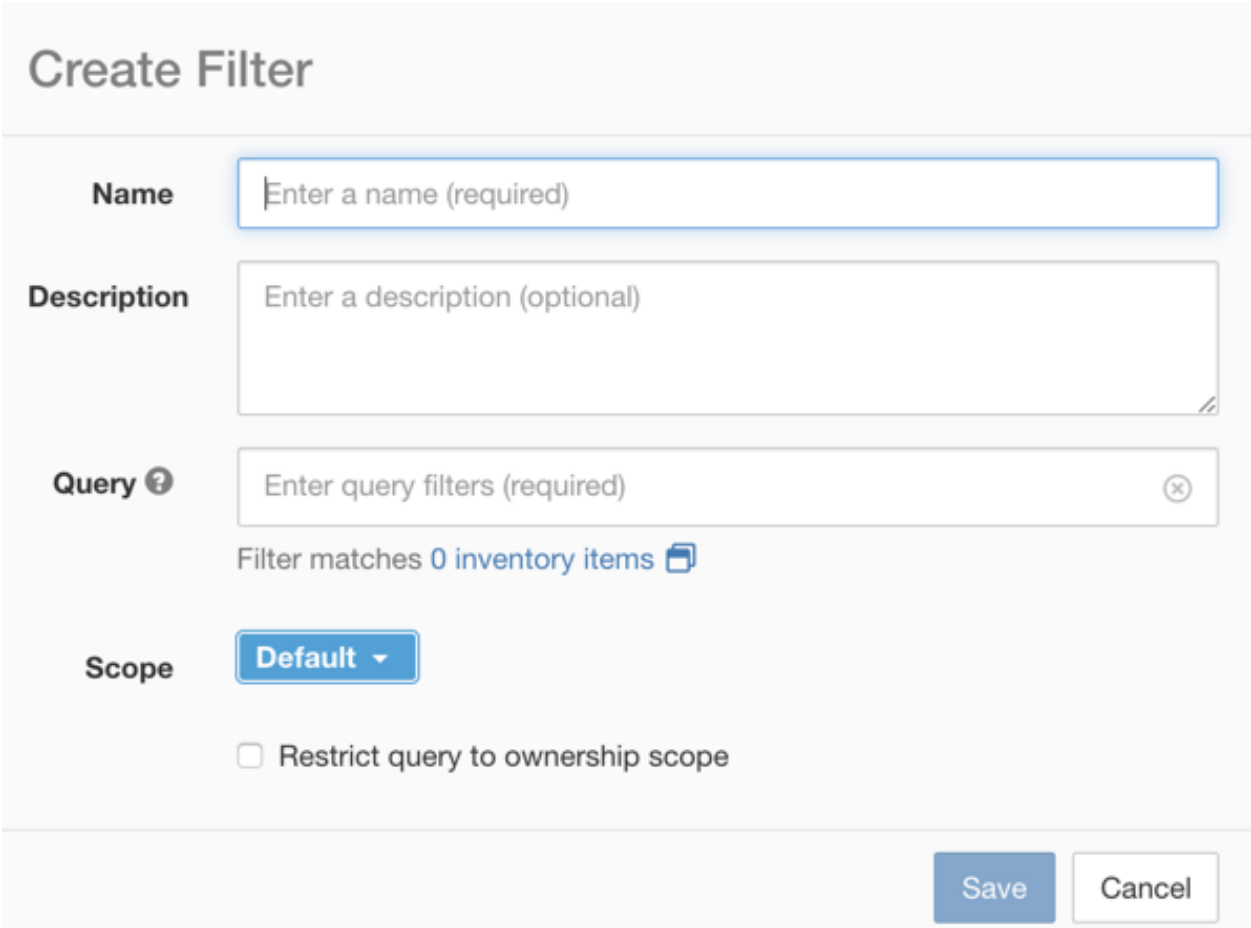


Fig. 4.2.2: Create filter modal

Existing filters can be edited or deleted by clicking the appropriate icon in the table. You can review inventory membership changes with respect to the selected parent scope by visiting the *Review Scope/Filter Change Impact* window.

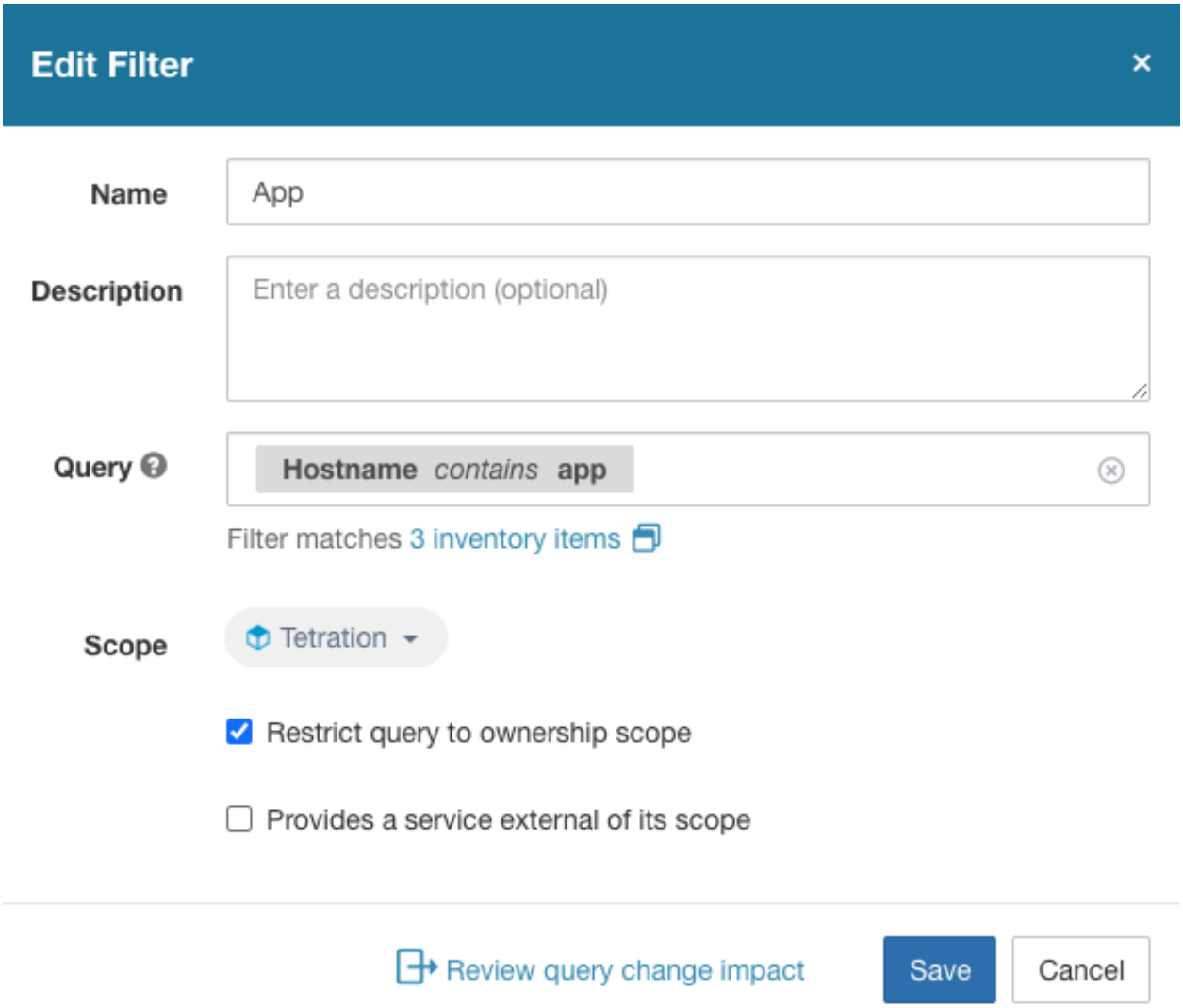


Fig. 4.2.3: Edit filter modal

4.2.1 Scope

The scope is used to determine which users can see and modify it. All users with read access within a tenant can view filters belonging to scopes within the tenant. To modify a filter, a user must have write access to the filter's scope or any of its ancestors.

Read more about *Scopes*.

4.2.2 Restrict to Ownership Scope

Whether or not the scope impacts the inventory matched by a filter is determined by the **Restrict to Ownership Scope?** checkbox.

For example, given the following structure:

1. Tenant with query `VRF ID = 3`
2. Scope within this tenant with query `hostname contains db`
3. Inventory filter with query `Platform = Linux` attached to this scope.

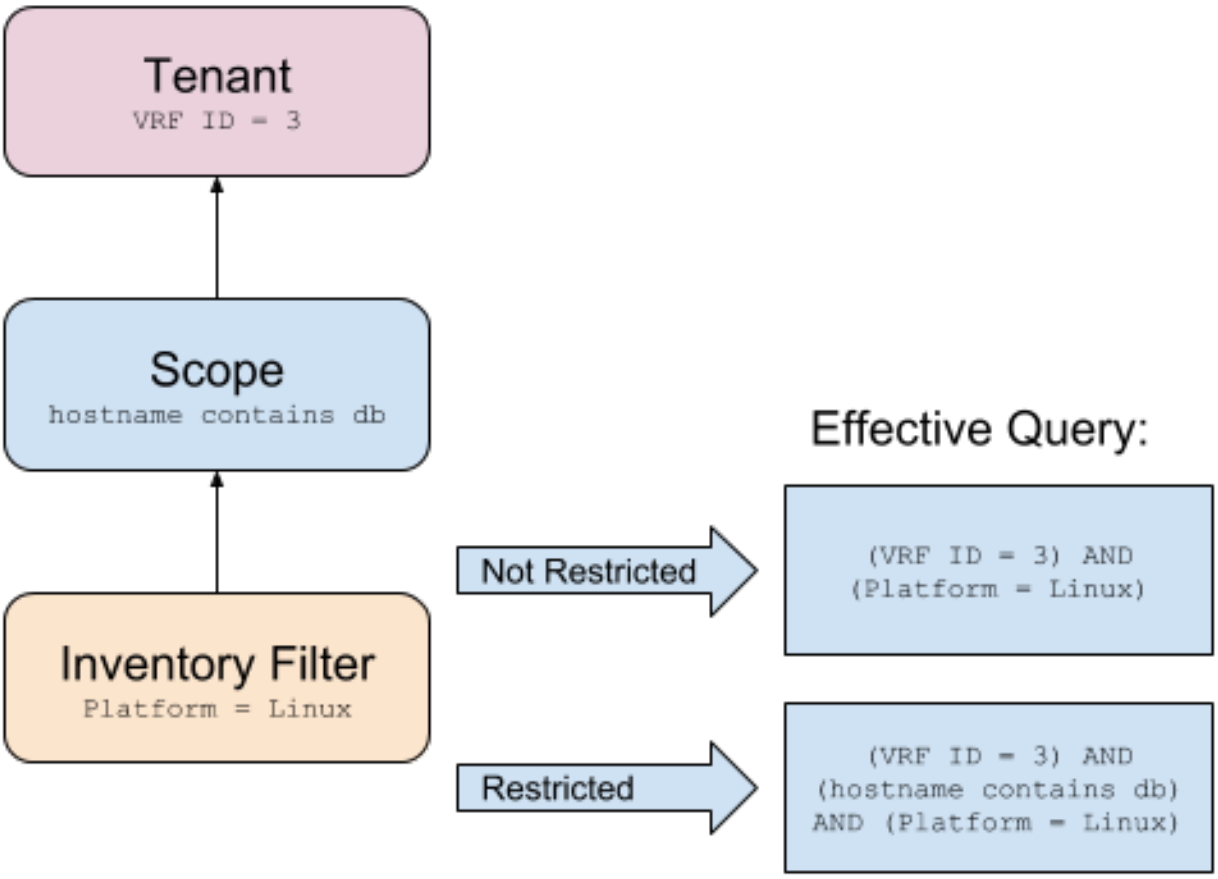


Fig. 4.2.2.1: Tenant, Scope and Inventory Filter Structure

- When **Restrict to Ownership Scope** is not checked: The filter matches all hosts within the tenant that also match the filter. The effective query would be: `(VRF ID = 3) AND (Platform = Linux)`.
- When **Restrict to Ownership Scope** is checked: The filter only matches hosts within the tenant and the scope that also match the filter. The effective query would be: `(VRF ID = 3) AND (hostname contains db) AND (Platform = Linux)`.

4.3 Review Scope/Filter Change Impact

Updating a scope query can impact application inventory membership after it gets committed. Likewise filter query change, which gets saved directly, can also impact the application inventory memberships. You can identify membership changes between the new and old queries by following the **Review query change impact** link on either Scope or Filter Edit modals. In addition, knowing the scope or filter dependencies can be helpful for impact analysis as well as

removing all neccessary objects preventing Scope deletion. Visit the **Dependencies** tab as well, to traverse the Scope Dependencies tree for further information.

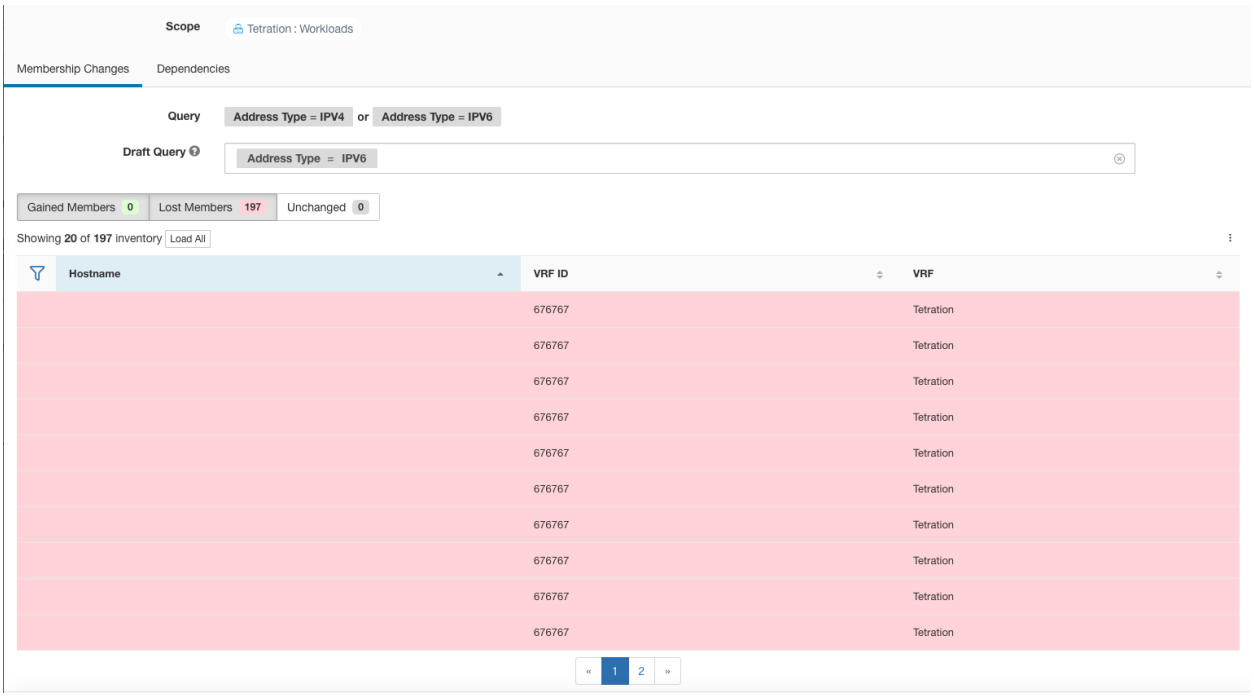


Fig. 4.3.1: Download Membership Table

4.3.1 Scope Query Change Impact Modal

Both **Membership Changes** and **Dependencies** tab can be accessed by following the link to **Review query change impact** on Scope Edit window.

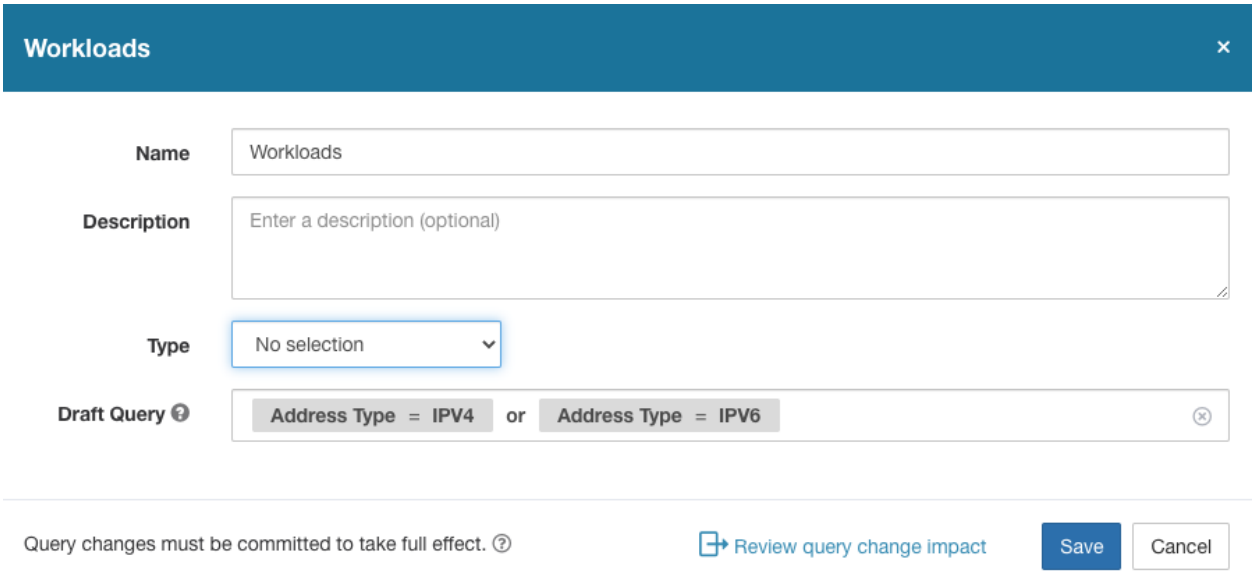


Fig. 4.3.1.1: Review query change impact

4.3.1.1 Membership Changes

The inventory table under Membership view contains all labelled columns which can be selectively displayed. Furthermore, you can download the csv or json of chosen Membership columns and rows with an additional Diff column identifying whether the inventory is **Gained**, **Lost** or **Unchanged**. Be sure that all table selection desired for download is visible to the table view.

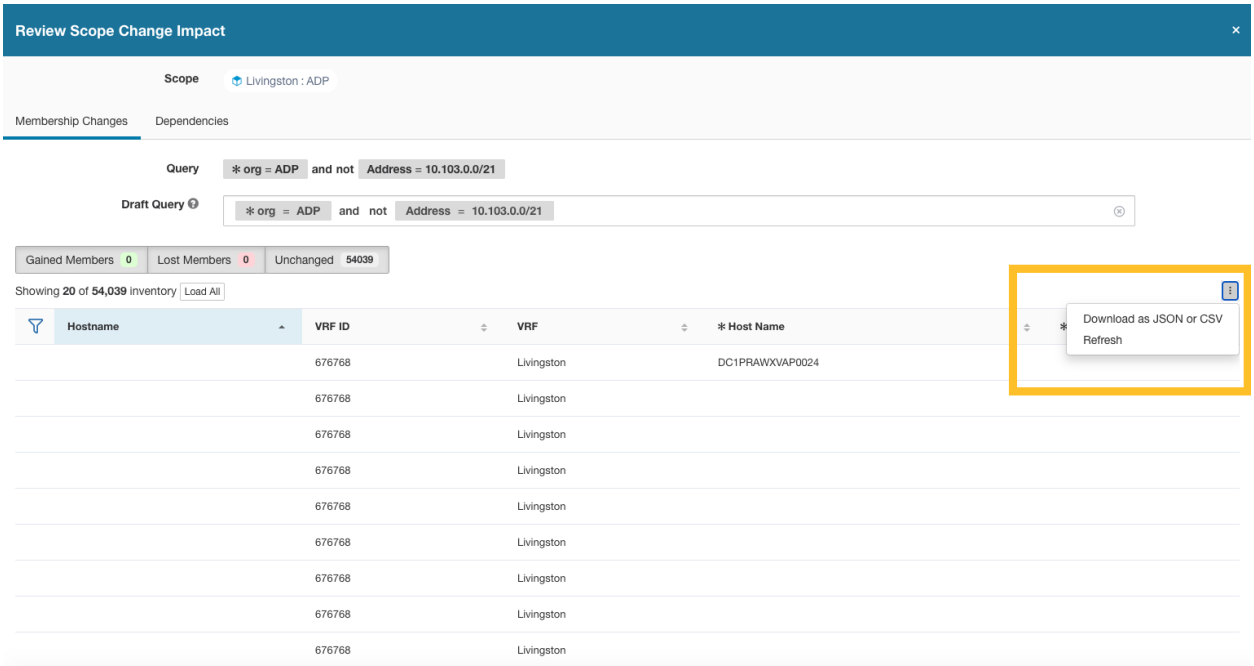


Fig. 4.3.1.1.1: Scope Membership Changes

4.3.1.2 Dependencies

You can traverse down to nested dependencies by further selecting **Review Dependencies**

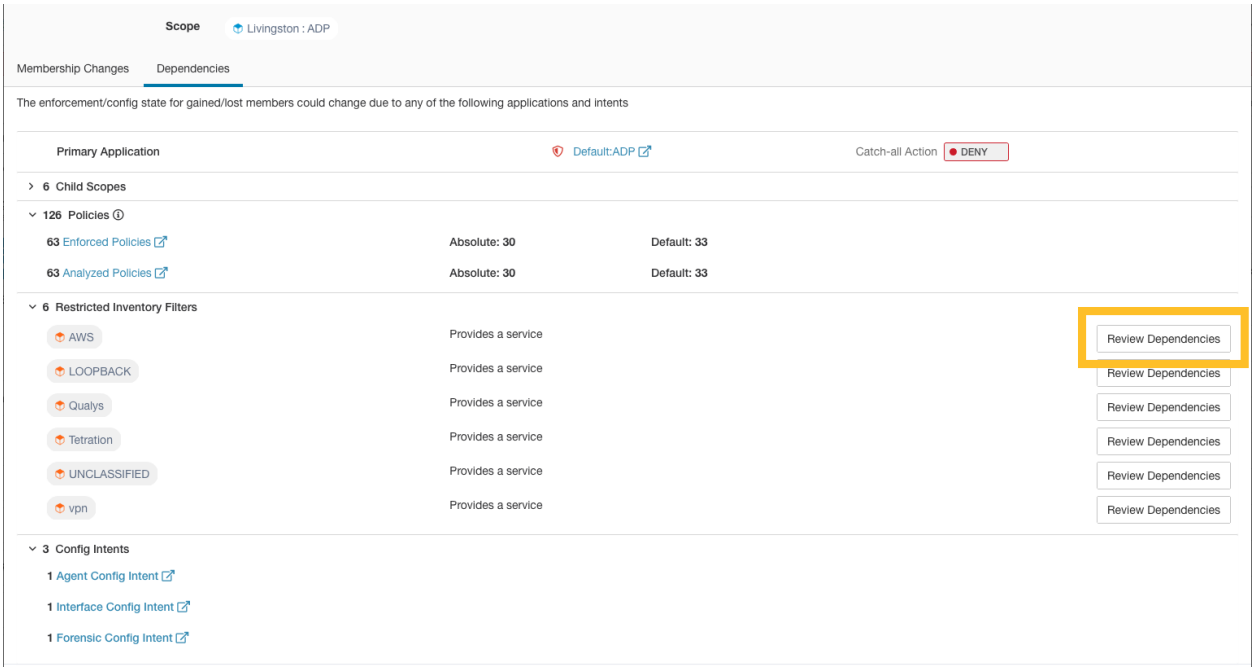


Fig. 4.3.1.2.1: Review Dependencies

You can traverse back up the dependencies tree by selecting the selected Parent link:

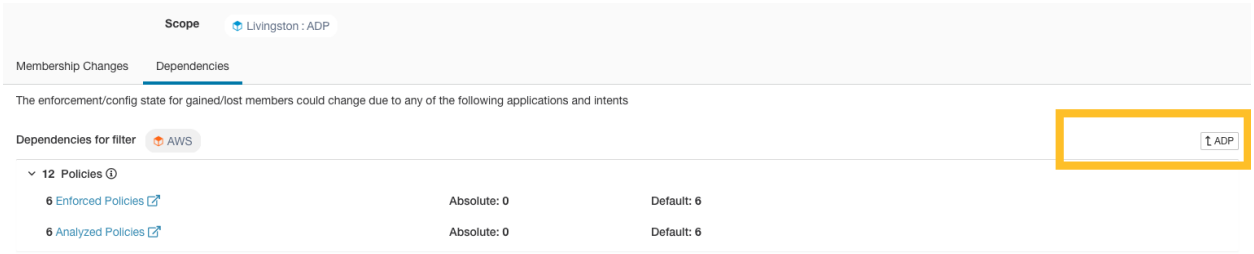


Fig. 4.3.1.2.2: Parent Link

The following are Scope Dependencies which may exist:

Type	Description
Application	Has primary and secondary application names and links to the specific workspaces under Segmentation
Child Scopes	Has names and links to child Scope Detail views. Allows drill down to lower level Dependencies
Policies	Has analyzed and enforced policies counts and links to respective Global Policy Views filtered by selected scope
Restricted Inventory Filters	Has names and links to child Filter Detail views. Allows drill down to lower level Dependencies
Config Intents	Has names and links to Agent, Interface and Forensics Config Intents views

4.3.2 Filter Query Change Impact Modal

Both **Membership Changes** and **Dependencies** tab can be accessed by following the link to **Review query change impact** on Inventory Filter Edit window.

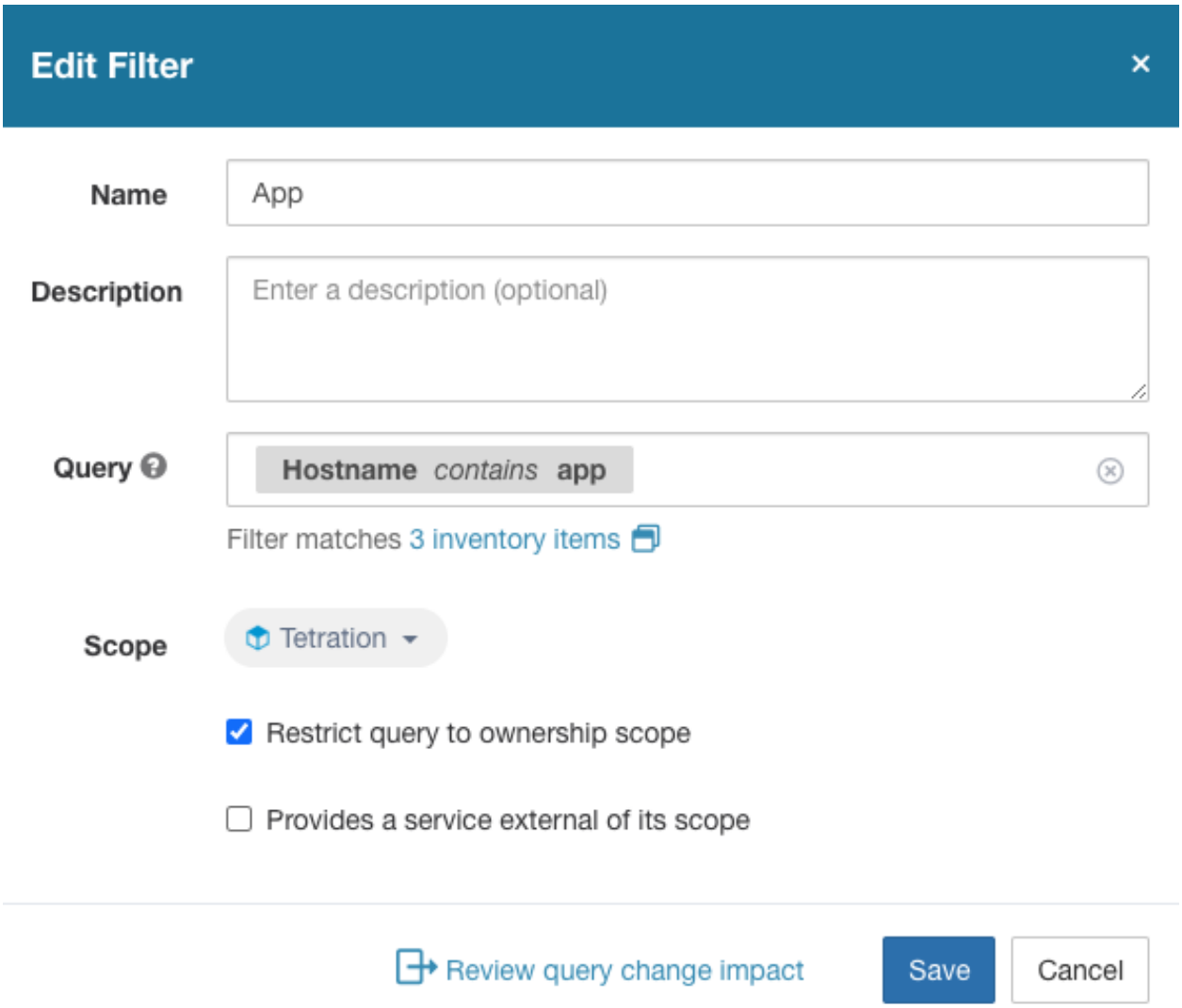


Fig. 4.3.2.1: Edit filter modal

4.3.2.1 Membership Changes

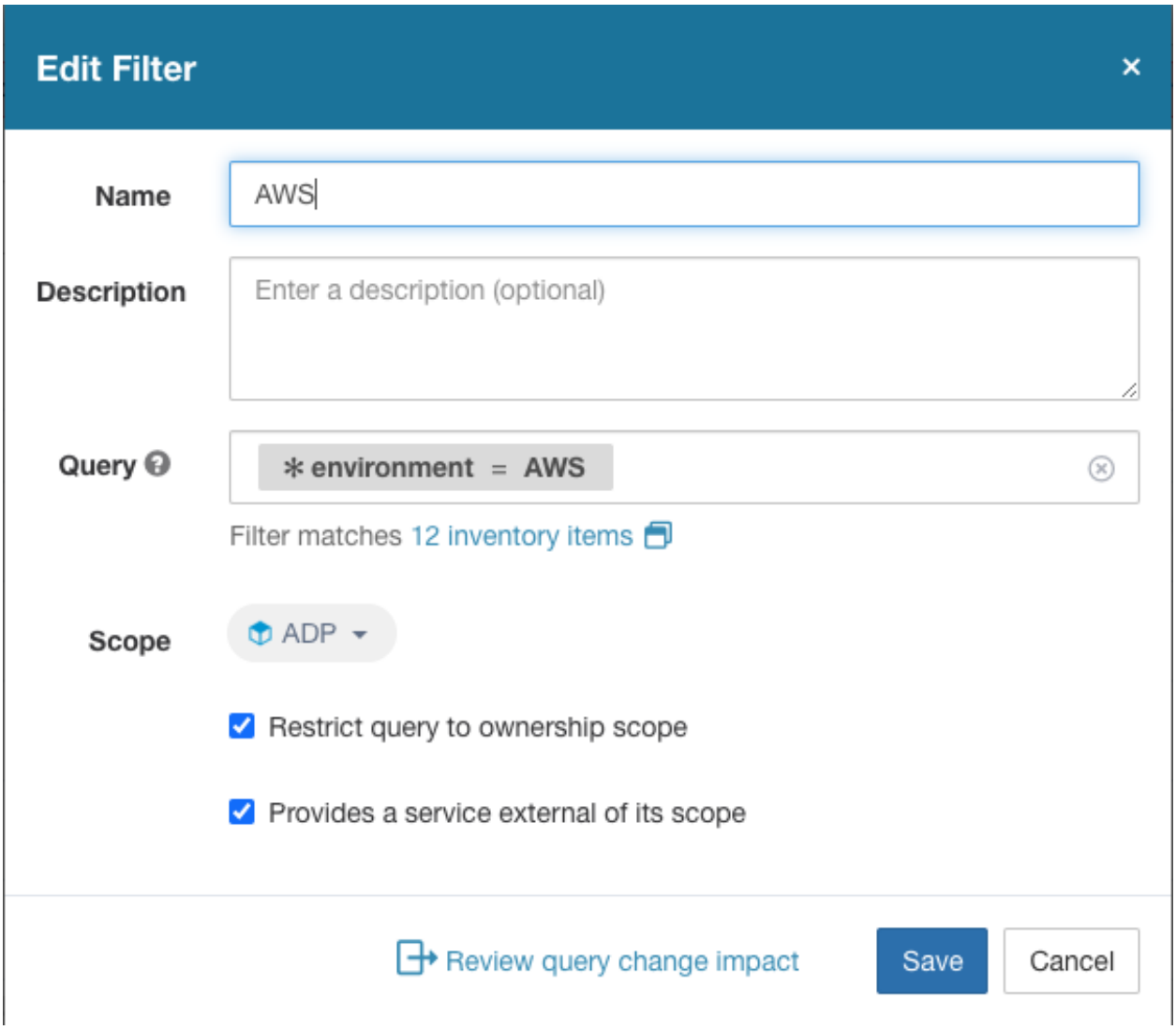


Fig. 4.3.2.1.1: Inventory Filter Membership Changes

4.3.2.2 Dependencies

The following are Filter Dependencies which may exist:

Type	Description
Policies	Has analyzed and enforced policies counts and links to respective Global Policy Views filtered by selected scope
Config Intents	Has names and links to Agent, Interface and Forensics Config Intents views

4.4 Inventory Profile

Tetration labels all inventory observed on the network and inventory profile page shows all these labels and more for a given inventory.

Note: An inventory profile page is linked from various places. One of the ways to see an inventory profile is to perform a search for inventory as described in search

From the results of inventory search, click on IP address to go to it's profile. The following labels are available for the inventory

4.4.1 Scopes

List of scopes that the inventory belongs to.

4.4.2 Inventory Type

- **Flow Learnt** inventory is registered based on the observed flows and *Collection Rules*.
- **Labeled** inventory is uploaded by the user via inventory upload utility.
- **Agent** inventory is reported by the Tetration software agent installed on a host.

4.4.3 User Labels

The list of user uploaded attributes for this inventory. See *User Labels* for more details.

4.4.4 Experimental Groups

The experimental groups is a list of cluster or user-defined inventory filters that are used for policy live analysis.

4.4.5 Enforcement Groups

The enforcement groups is a list of cluster or user-defined inventory filters that are used for policy enforcement. They can be different from experimental groups depending on the versions of policies being analyzed and/or enforced in the system.

4.4.6 Bandwidth Chart

This chart shows detailed time series data for traffic bytes and packets occurred between the period indicated by **Time Picker** at the top-left corner.

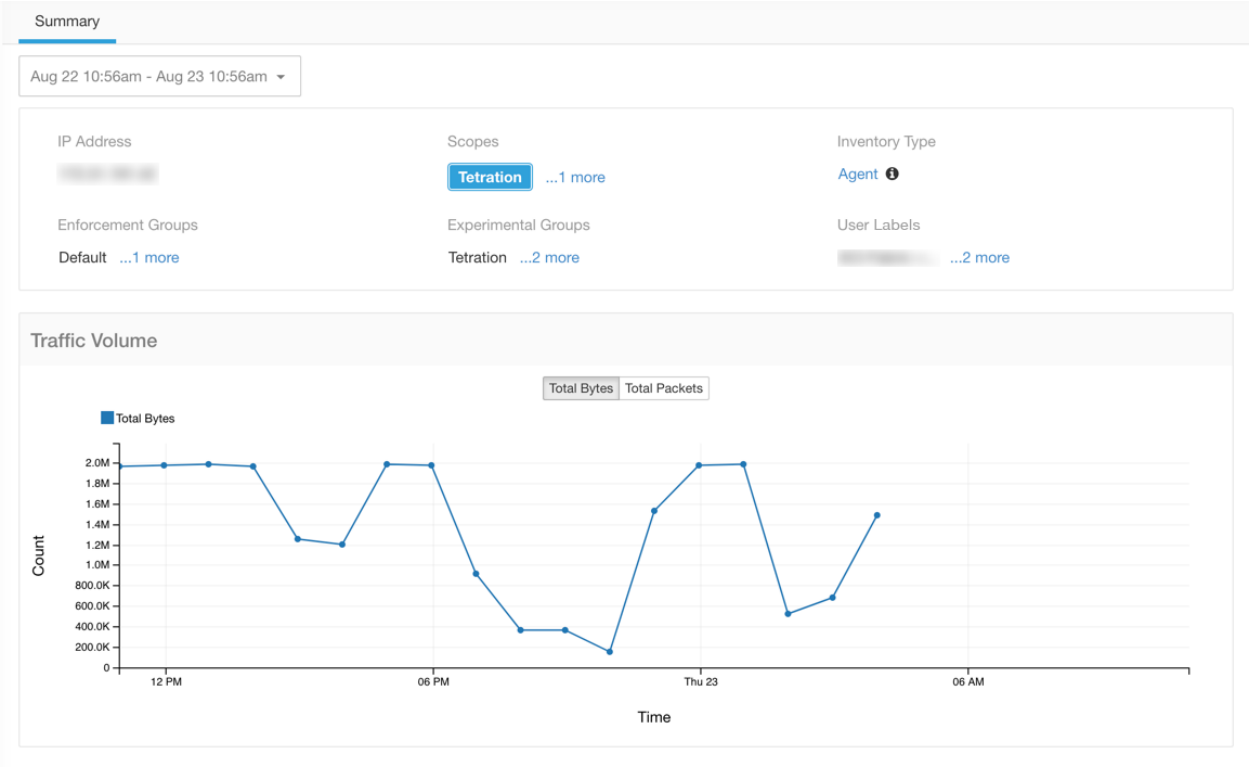


Fig. 4.4.6.1: Inventory Profile

4.4.7 Geo Chart

This chart shows aggregated inbound/outbound geo data for traffic observed weekly. You can adjust the time range to retreive data from custom time intervals.

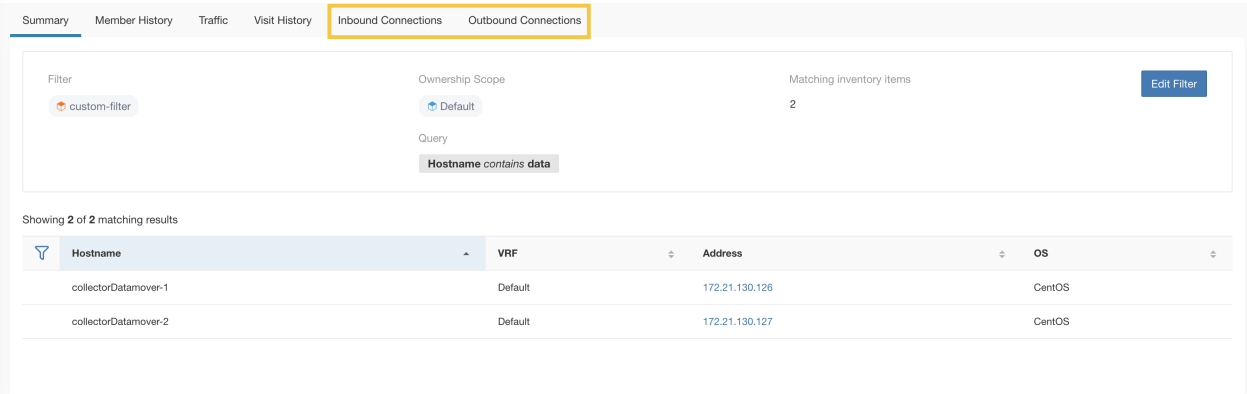


Fig. 4.4.7.1: Accessing Geo Tabs

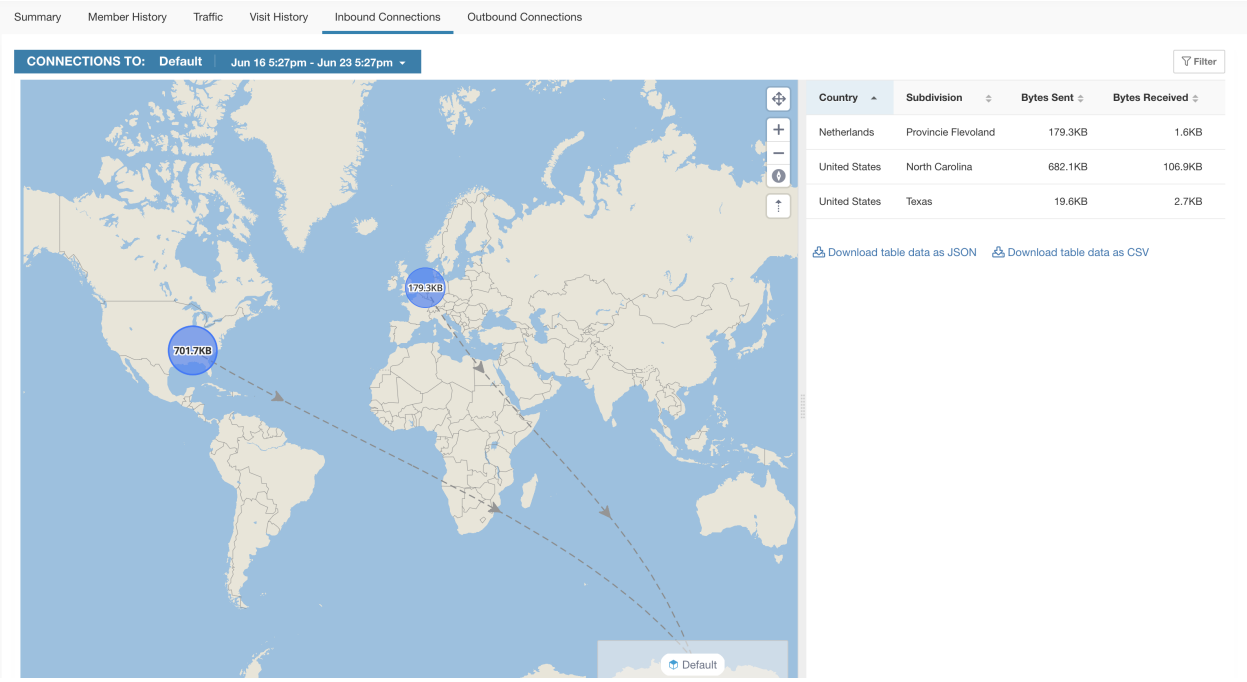


Fig. 4.4.7.2: Geo Chart Details

4.5 Workload Profile

Workload profile displays detailed information about a host where Tetration software agent is installed. This section explains how to view a workload profile and the information it contains.

Note: A workload profile page is linked from various places. One of the ways to see a workload profile is to perform a search for host as described in search

---

From the results of inventory search, click on IP address of the host to go to it's profile. Based on the type of agent installed on the host, the following tabs are available on the page. Note that you may end up on inventory profile page if Tetration software agent is not installed on the host that this inventory belongs to.

4.5.1 Summary Tab

This tab includes the enforcement and experimental groups, scopes that the host belongs to. The experimental groups are inventory filters that are used for policy live analysis, while the enforcement groups are the filters that are used for policy enforcement. They can be different depending on the versions of policies being analyzed and/or enforced in the system.

The status information of the host software agent such as it's type, OS platform, agent version and last check-in time are also shown in the summary. See *Software Agent Config* for more details. This tab also shows detailed time series data for traffic bytes and packets occurred between the period indicated by the **Time Picker** at the top-left corner.

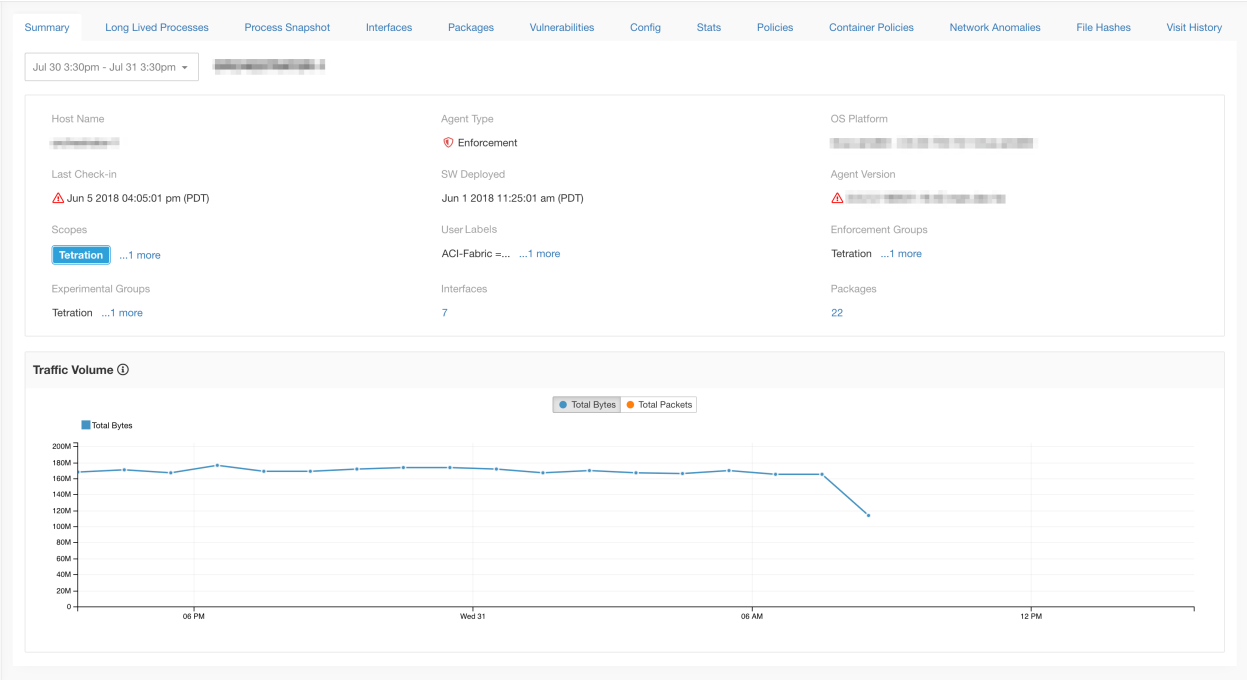


Fig. 4.5.1.1: Workload Bandwidth

For users with root scope owner privileges, summary page also includes a section to collect and download agent logs for deep visibility and enforcement agents (versions 3.3 or later) within that root scope. Also note that this feature is not available for agents running on platforms AIX and SUSE Linux Enterprise Server (s390x-Linux on IBM Z architectures). Use "Initiate Log Collection" button to collect logs from the agent and then logs will be available for download in a few minutes. If the download fails, please retry collection of logs and then attempt download again.

Fig. 4.5.1.2: Agent Logs

4.5.2 Interfaces tab

This tab shows details about the network interfaces installed on the host. It is available for all types of software agents.

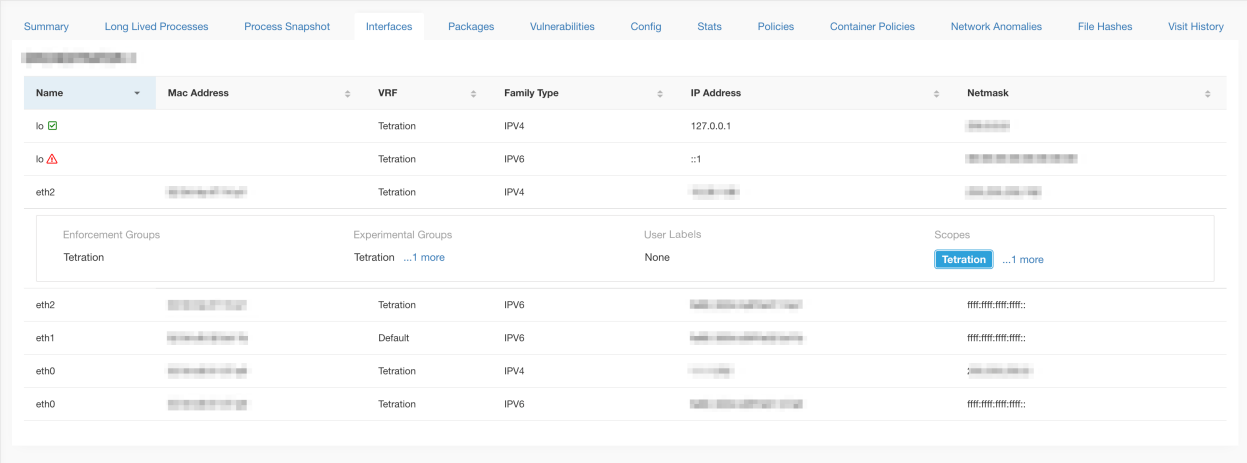


Fig. 4.5.2.1: Workload Interface List

4.5.3 Process List Tab

This tab shows list of processes running on the host. A filter is also available to narrow down the list of processes based on the attributes of a process shown in table header below.

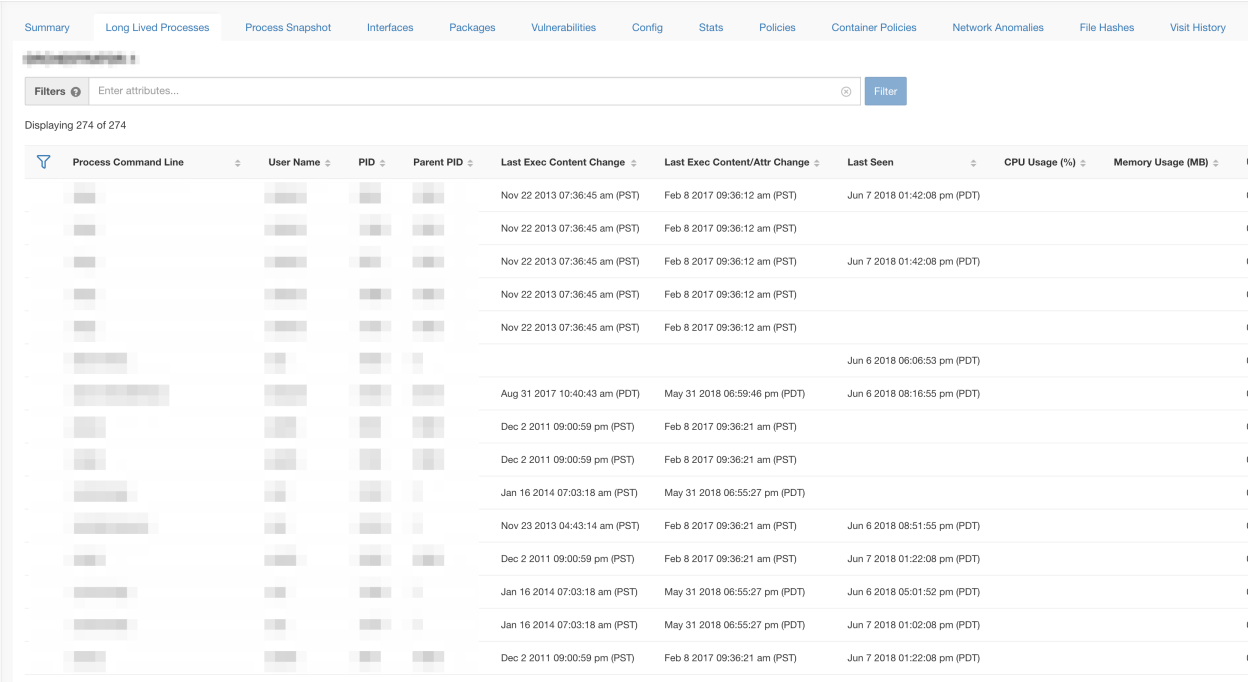


Fig. 4.5.3.1: Workload Process List

Attribute Descriptions:

Attribute	Description
Last Exec Content Change	Similar to mtime in linux. It is the timestamp when only the file content changes
Last Exec Content/Attr Change	Similar to ctime in linux. It is the timestamp when either the file content or attribute changes
Last Seen	Last time when the process is observed. Available when the process is dead
CPU Usage	CPU usage trend by the process in the past hour
Memory Usage	Memory usage trend by the process in the past hour
Process Binary Hash	SHA256 hash of the process binary in hex string, also known as process hash for short. Not available for kernel processes
Anomaly Score	Process hash (anomaly) score. See *Process hash anomaly detection* for more information
Verdict	Verdict of the process hash (either Malicious or Benign). The verdict is determined based on whether the process hash belongs to any user-defined hash list or known threat-intelligence hash database. See *Process hash anomaly detection* for more information.
Verdict Source	Source of the verdict. The verdict source can be either User Defined, or Tetration Cloud, or NIST. This attribute is known as Hash DB Source in previous releases. See *Process hash anomaly detection* for more information

4.5.4 Process Snapshot Tab

This tab shows searchable process tree observed on the workload.

Fig. 4.5.4.1: Workload Process Snapshot

4.5.5 Software Packages Tab

This tab shows list of packages installed on the host. Users can selectively view software packages based on package attributes in the table header.

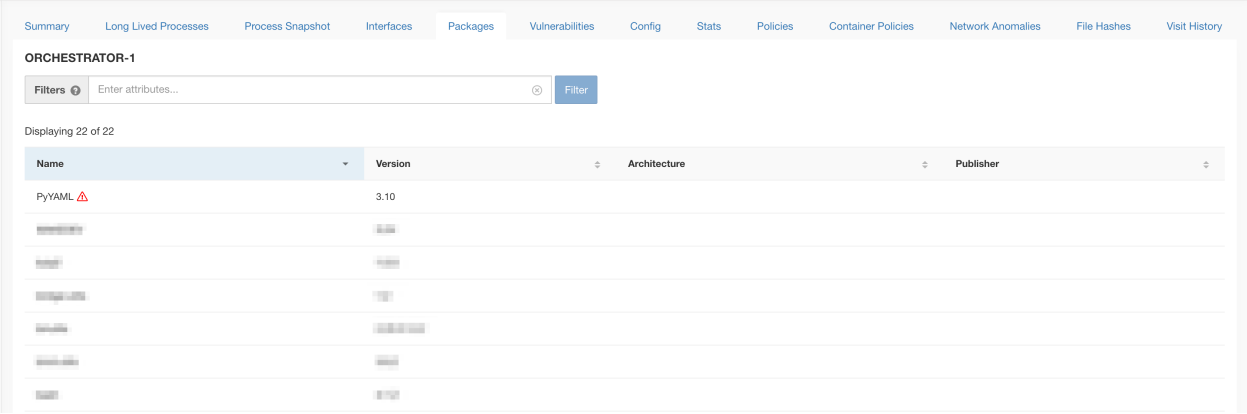


Fig. 4.5.5.1: Software Packages List

4.5.6 Vulnerabilities Tab

This tab shows searchable vulnerabilities observed on the workload based on the Common Vulnerabilities and Exposures (CVE) system. See *Vulnerability data visibility*

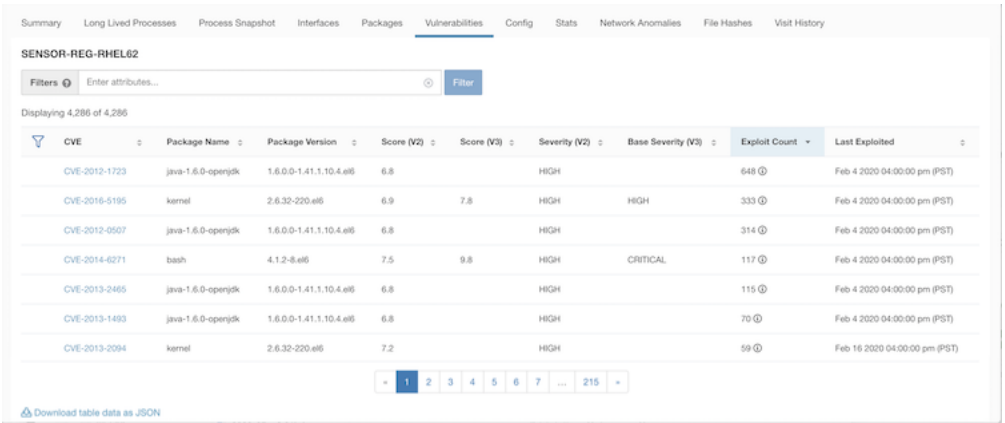


Fig. 4.5.6.1: Vulnerabilities Tab

4.5.7 Agent Configuration Tab

This tab shows software agent settings. It is only available for Deep Visibility and Enforcement Agents. These settings can be modified using Agent Configuration Intents via the agent config page. See *Software Agent Config*

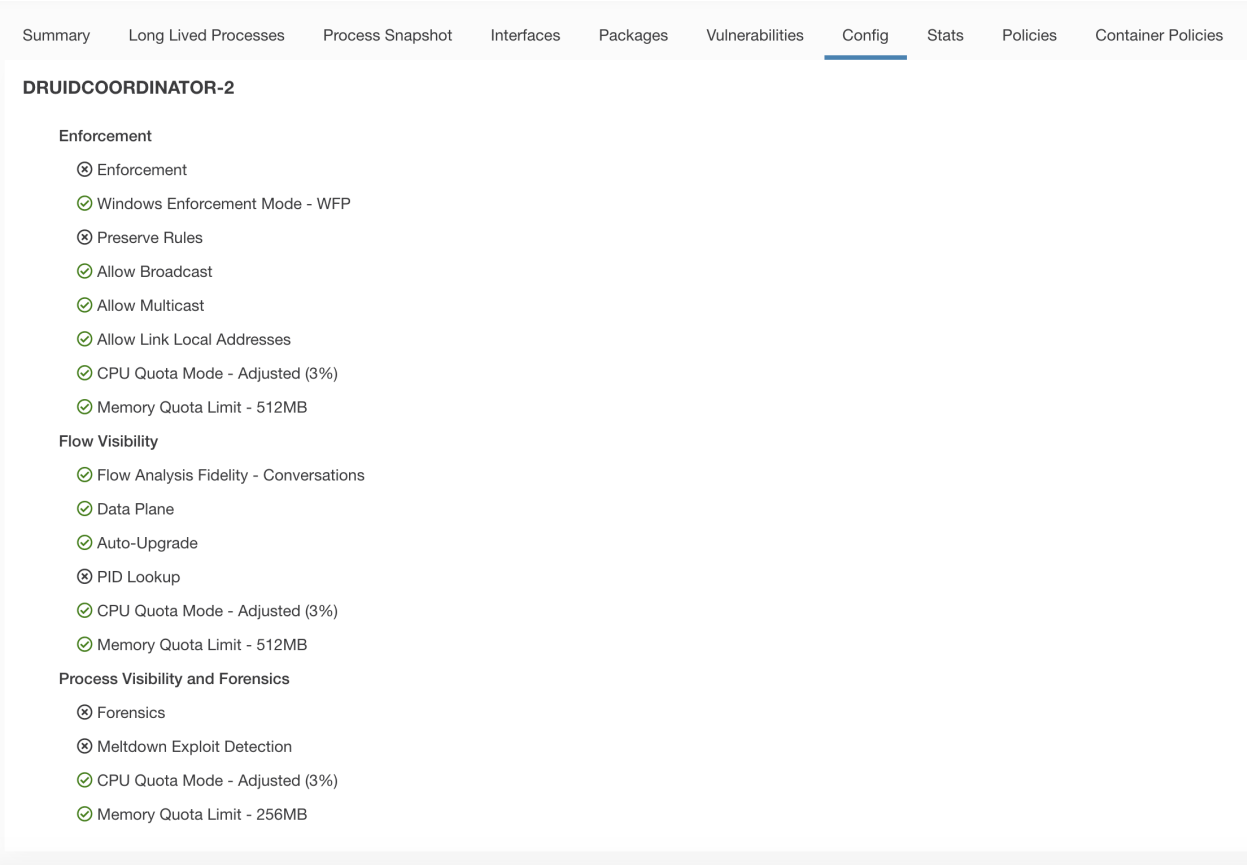


Fig. 4.5.7.1: Applied Workload Configuration

4.5.8 Agent Statistics Tab

This tab shows statistics about the Tetration agent installed on the host. It is only available for Deep Visibility and Enforcement Agents.

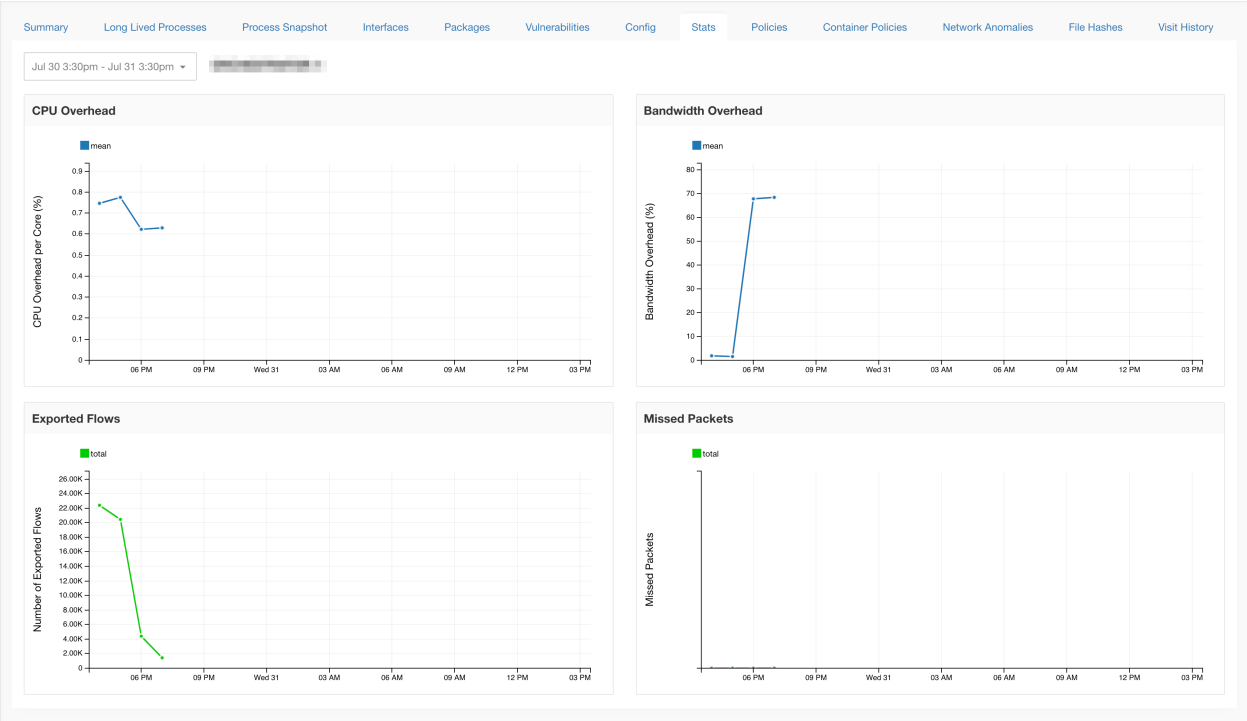
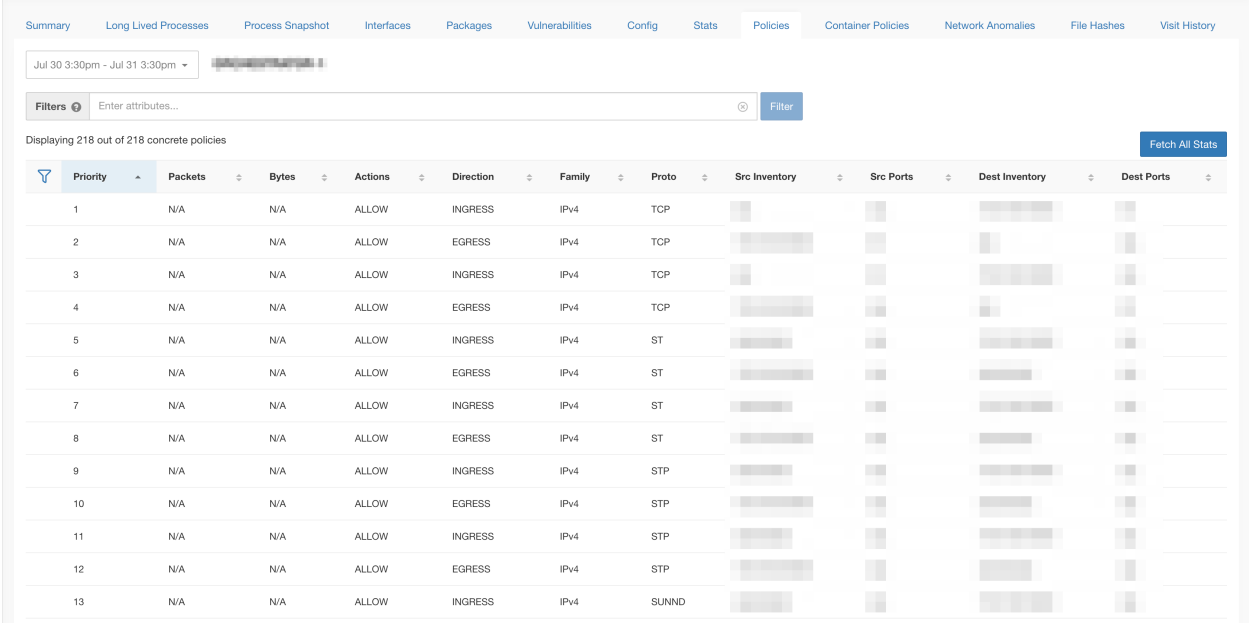


Fig. 4.5.8.1: Agent Statistics

4.5.9 Policies Tab

This tab shows Tetration concrete enforcement policies applied on the host. Each row in this table corresponds to a firewall rule implemented on the host. Each policy row can be further expanded to display the logical intent from which this concrete policy derived. Packet and byte count time series view is also available for each rule. A filter is also available in this tab to narrow the list of enforced policies based on attributes of a policy shown in table header below. This tab is only available for Enforcement Agents.



Priority	Packets	Bytes	Actions	Direction	Family	Proto	Src Inventory	Src Ports	Dest Inventory	Dest Ports
1	N/A	N/A	ALLOW	INGRESS	IPv4	TCP				
2	N/A	N/A	ALLOW	EGRESS	IPv4	TCP				
3	N/A	N/A	ALLOW	INGRESS	IPv4	TCP				
4	N/A	N/A	ALLOW	EGRESS	IPv4	TCP				
5	N/A	N/A	ALLOW	INGRESS	IPv4	ST				
6	N/A	N/A	ALLOW	EGRESS	IPv4	ST				
7	N/A	N/A	ALLOW	INGRESS	IPv4	ST				
8	N/A	N/A	ALLOW	EGRESS	IPv4	ST				
9	N/A	N/A	ALLOW	INGRESS	IPv4	STP				
10	N/A	N/A	ALLOW	EGRESS	IPv4	STP				
11	N/A	N/A	ALLOW	INGRESS	IPv4	STP				
12	N/A	N/A	ALLOW	EGRESS	IPv4	STP				
13	N/A	N/A	ALLOW	INGRESS	IPv4	SUNND				

Fig. 4.5.9.1: Concrete Policy List

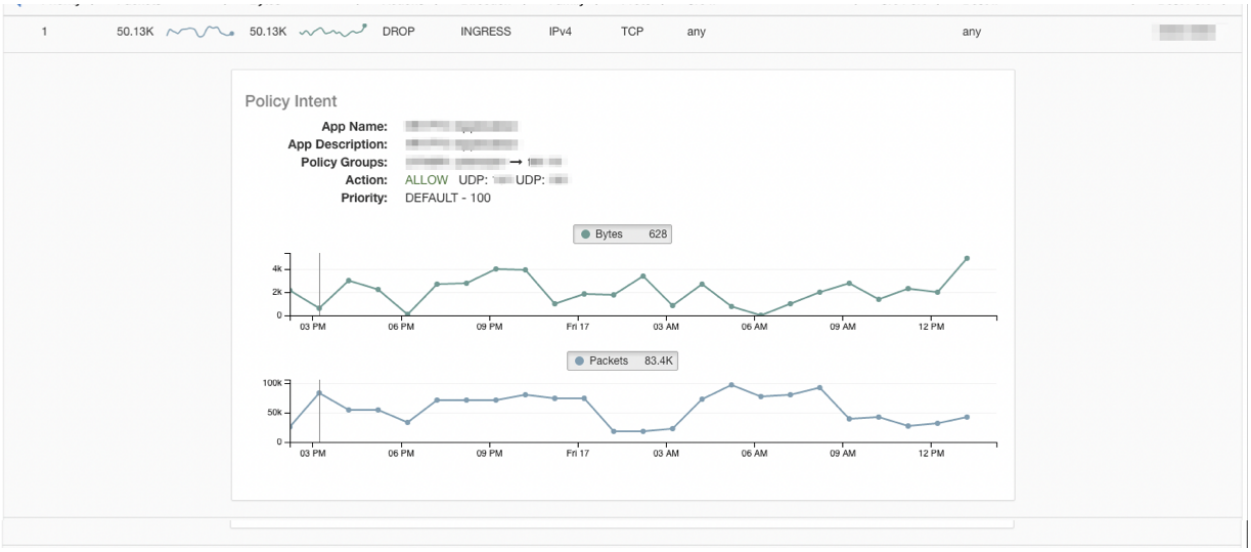


Fig. 4.5.9.2: Concrete Policy Row

4.5.10 Container Policies Tab

This tab shows Tetration concrete enforcement policies applied on the containers. Each row in this table corresponds to a firewall rule implemented on the container pod.

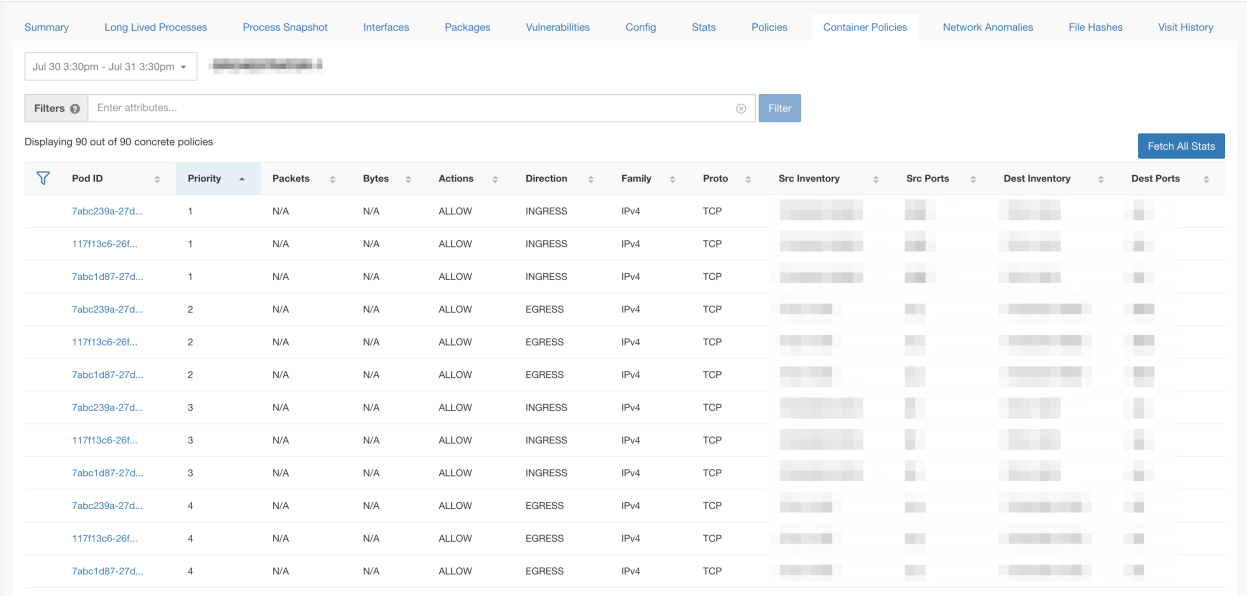


Fig. 4.5.10.1: Container Concrete Policy List

4.5.11 Network Anomalies Tab

This tab helps to identify the events with large data movements in or out of this workload. See *PCR-based Network Anomaly detection* for more information.

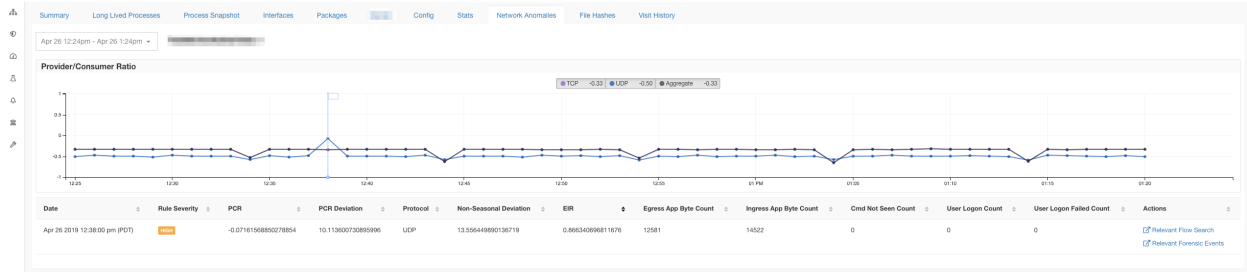


Fig. 4.5.11.1: Workload Network Anomalies

4.5.12 File Hashes Tab

This tab detects process hash anomalies by assessing the consistency of process binary hashes across the system. See *Process hash anomaly detection* for more info.

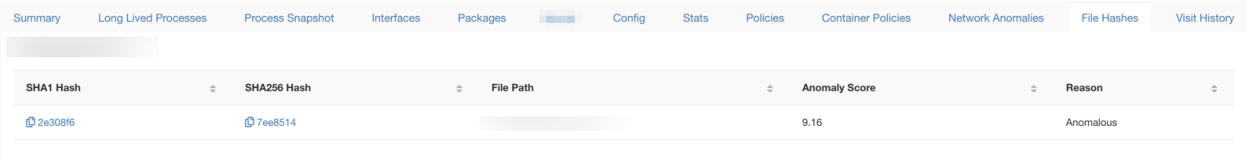


Fig. 4.5.12.1: Workload File Hashes

4.5.13 Visit History Tab

This tab shows visited domains and resolvers used for Domain Name Service(DNS)), Geo/Autonomous System Number(ASN) in last 24 hours at host ordered by bytes sent. It is useful for spotting anomalies so that appropriate action can be taken to prevent malicious activity.

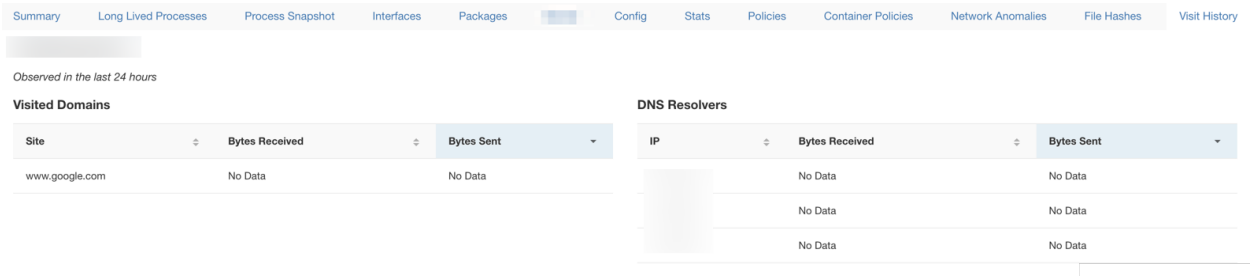


Fig. 4.5.13.1: Workload Visit History

4.6 Software Packages

The **Software Packages** feature set allows viewing packages installed on hosts and the vulnerabilities affecting them. Specifically, it allows to:

- View packages registered with the following package managers:
  - Linux: Redhat Package Manager (RPM) and Debian Package Manager (dpkg)
  - Windows: Windows Registry Service
- View Common Vulnerabilities and Exposures (CVEs) affecting packages installed on a host.
- Define inventory filters using the package name and version.

4.6.1 Packages Tab

To view packages installed on a host, navigate to the packages tab on the workload profile *Workload Profile* page.

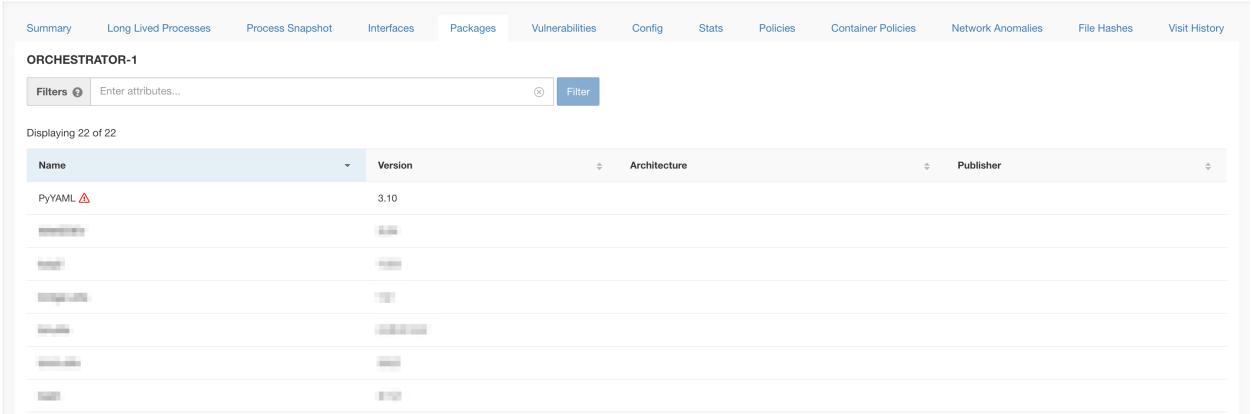


Fig. 4.6.1.1: Workload profile packages

4.6.2 Common Vulnerabilities and Exposures (CVEs)

In addition to displaying packages under the packages tab, we display common vulnerabilities affecting them along with their severity. Each vulnerability contains a link to the Nation Vulnerability Database (NVD) which provides more information on the specific vulnerability. In addition to displaying the CVE ID, we also display the impact score (on a scale of 10), indicative of the severity of the vulnerability.

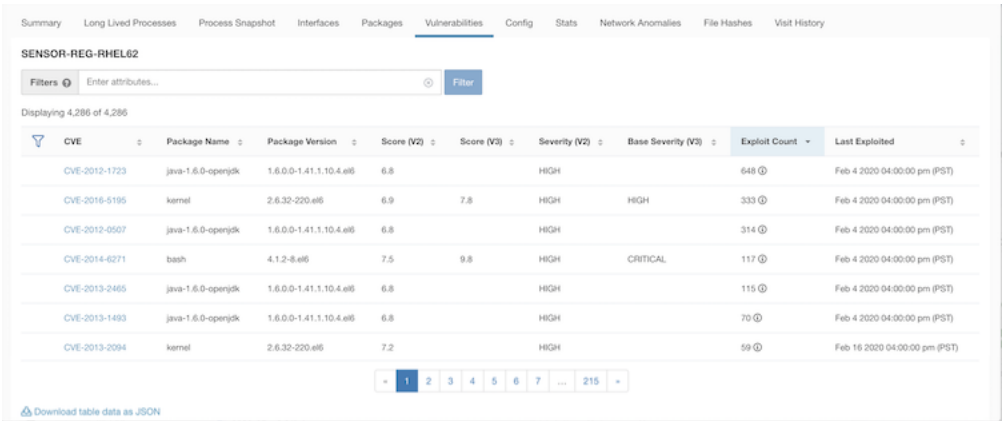


Fig. 4.6.2.1: Workload profile packages CVE

4.6.3 Windows Packages and CVEs

Following section lists the behavior of Windows agent with regards to reporting package information to Tetration.

- Windows applications, PowerShell, IE are reported as packages. .net framework is also reported as a package.
- Other Windows applications like notepad.exe, cmd.exe, mstsc.exe etc. are not reported.
- Windows server configured roles and features are reported as packages but the version may be incorrect. For example: If the DNS server is configured, reported version will either 0 or 8.
- Windows agent reports 3rd party products installed using MSI installer or exe installer:
  - For MSI installers, MSI APIs are used to retrieve package information e.g. version, publisher, package name.
  - If the exe installer is used to install the package, package information is retrieved from the registry.
  - Package installer fields like version, publisher are optional. If version is missing, the package will not be reported.
  - If a product is extracted from zip file or installed as an app, it will not be reported in the package list.

4.6.4 Inventory Filters

Package related information can be searched by defining an inventory filter with the package name and version (optional).

The syntax for this filter is as follows: PackageName#PackageVersion

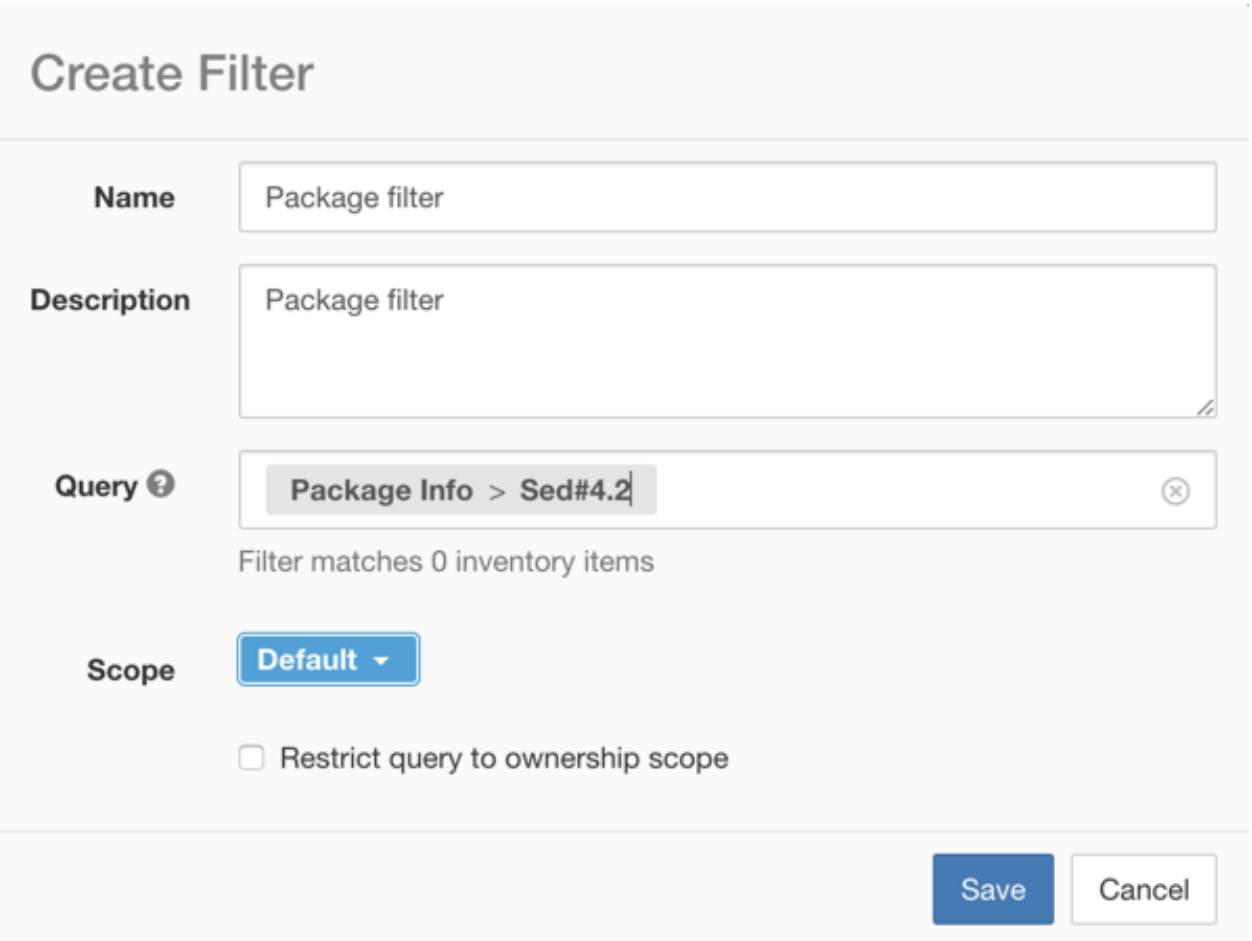


Fig. 4.6.4.1: Inventory package

The following operations are supported:

- Equality - returns hosts with packages matching PackageName and the PackageVersion (if provided).
- Inequality - returns hosts with packages matching PackageName but not the PackageVersion (if provided).
- Greater Than - returns hosts with packages matching PackageName and with version greater than PackageVersion.
- Greater Than or Equal To - returns hosts with packages matching PackageName and with version greater than or equal to PackageVersion.
- Less Than - returns hosts with packages matching PackageName and with version less than PackageVersion.
- Less Than or Equal To - returns hosts with packages matching PackageName and with version less than or equal to PackageVersion.

4.7 Vulnerability data visibility

The **Vulnerability data visibility** feature allows for detecting and viewing vulnerabilities affecting packages and processes on a host. Inventory filters can be defined using:

```
- CVE IDs.
- CVSS v2 and v3 scores.
- CVSS v2 access vector and access complexity.
- CVSS v3 attack vector, attack complexity, and privilege required.
```

4.7.1 Workload Profile Page

Vulnerability related information affecting packages and processes on a system is displayed on the *Workload Profile* page.

4.7.1.1 Packages Tab

The packages tab lists packages installed on a host and vulnerabilities affecting them.

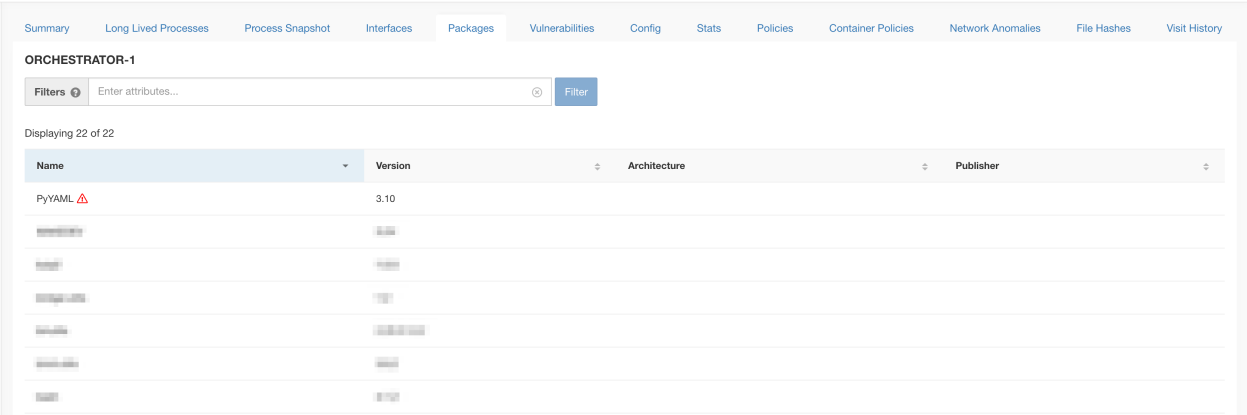


Fig. 4.7.1.1.1: Workload profile packages

4.7.1.2 Process List Tab

Long-lived processes along with their vulnerabilities are displayed under the process list tab.

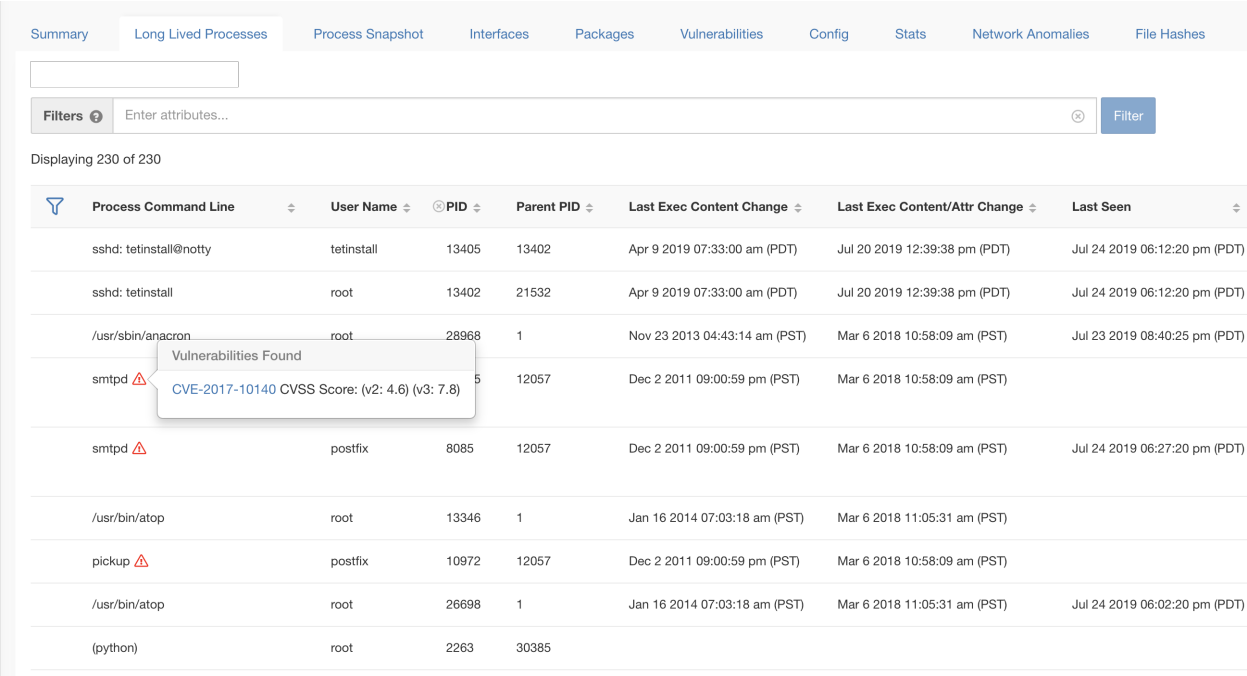


Fig. 4.7.1.2.1: Workload profile process list

4.7.1.3 Process Snapshot Tab

Vulnerability information is displayed for all processes in the process tree under the process snapshot tab.

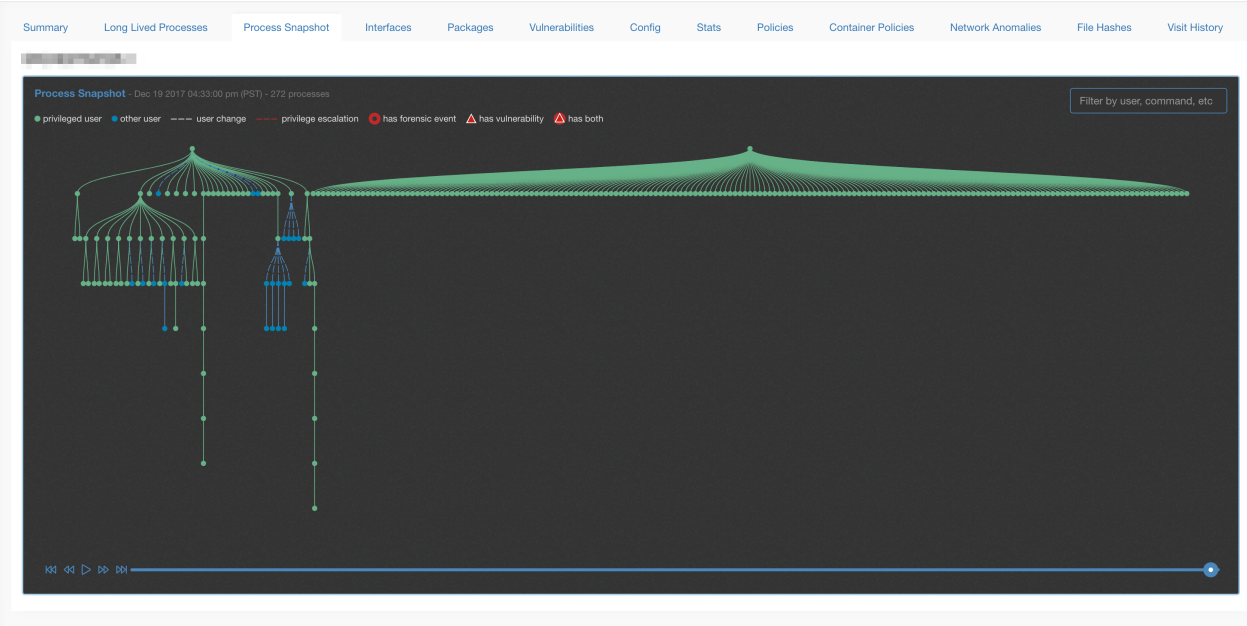


Fig. 4.7.1.3.1: Workload profile process snapshot tab

4.7.1.4 Vulnerabilities Tab

The vulnerability tab shows a list of vulnerabilities observed on the workload.

For each CVE, besides basic impact metrics, exploit information based on our threat intelligence is displayed:

- Exploit Count: number of times CVE was seen exploited in the wild in the last year
- Last Exploited: last time CVE was seen exploited in the wild by our threat intelligence

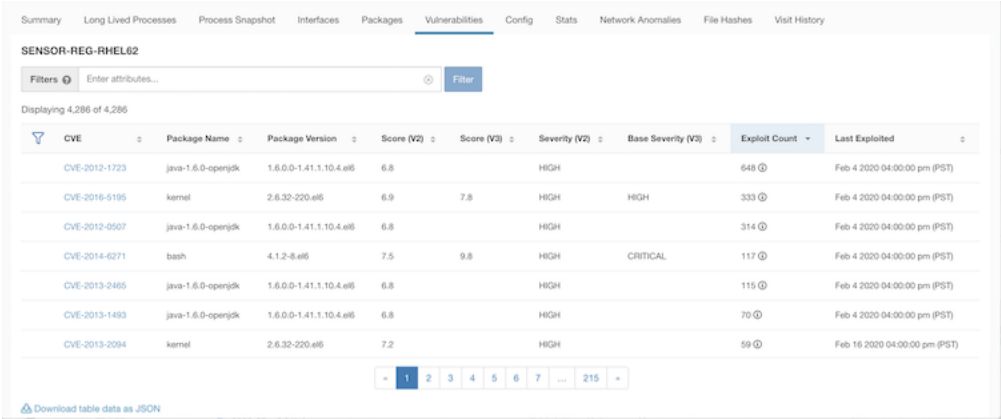


Fig. 4.7.1.4.1: Workload profile vulnerabilities tab

4.7.2 Inventory Filters

The following types of inventory filters can be defined to identify hosts with vulnerable packages:

4.7.2.1 CVE ID based filter

This filter allows searching for hosts affected by a specific CVE or any CVE.

To search for a host affected by a specific CVE, provide the CVE ID in the format: CVE-XXXX-XXXX

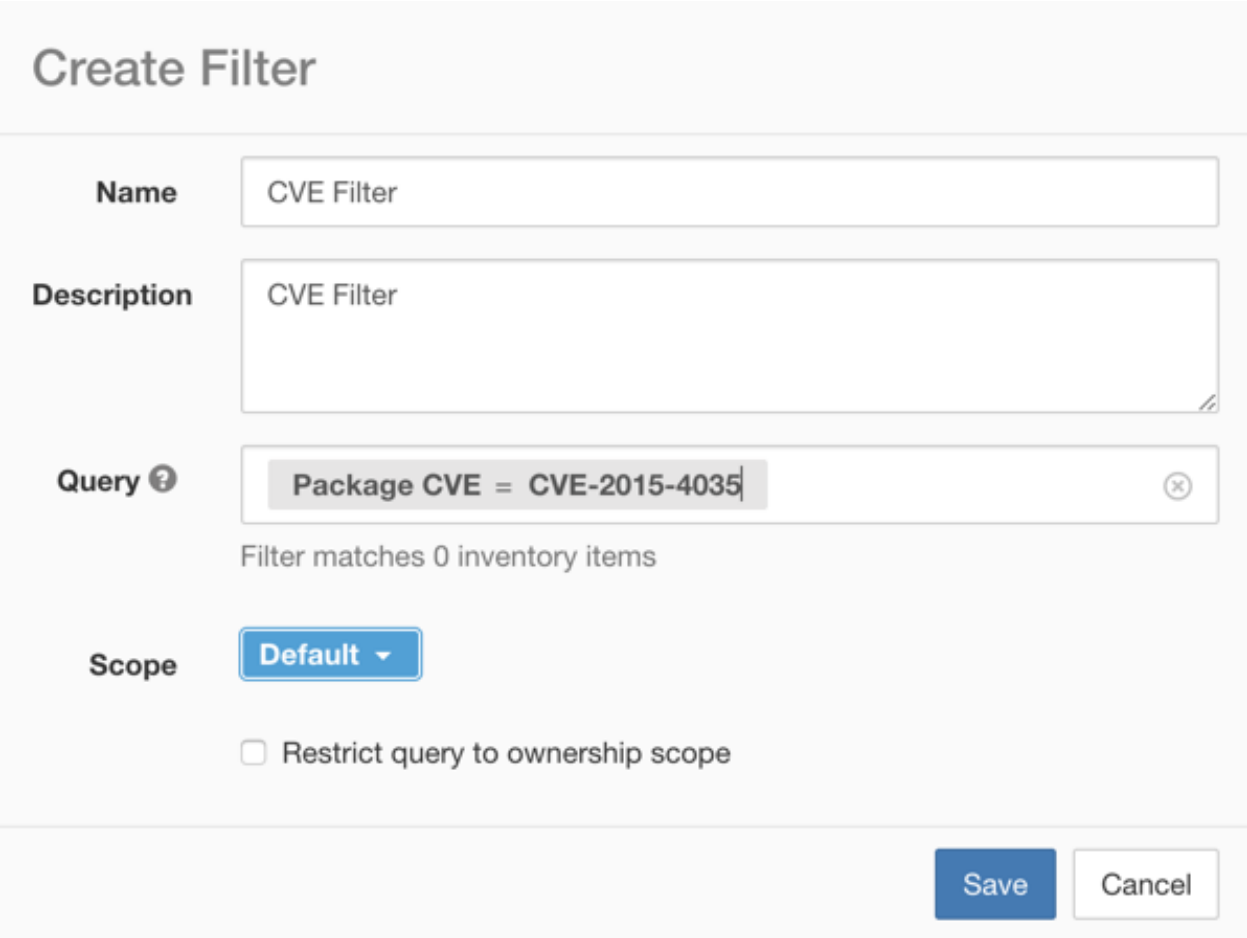


Fig. 4.7.2.1.1: Inventory filter CVE

The following operations are supported:

- Equality - returns hosts with packages affected by a CVE ID.
- Inequality - returns hosts with packages not affected by a CVE ID.
- Contains - returns hosts with packages affected by a CVE present in the input string (entering "cve" will return hosts affected by a CVE).
- Doesn't contain - returns hosts with packages not affected by a CVE present in the input string (entering "cve" will return hosts not affected by a CVE).

4.7.2.2 CVSS (Common Vulnerability Scoring System) impact score based filter

This filter allows searching for hosts that have CVE with the specified CVSSv2 or CVSSv3 impact score. To search for hosts which have any CVE with impact score (v2 or v3), user can provide the score in numeric format

To search for hosts which have CVE with CVSSv2 impact score greater than 7.5

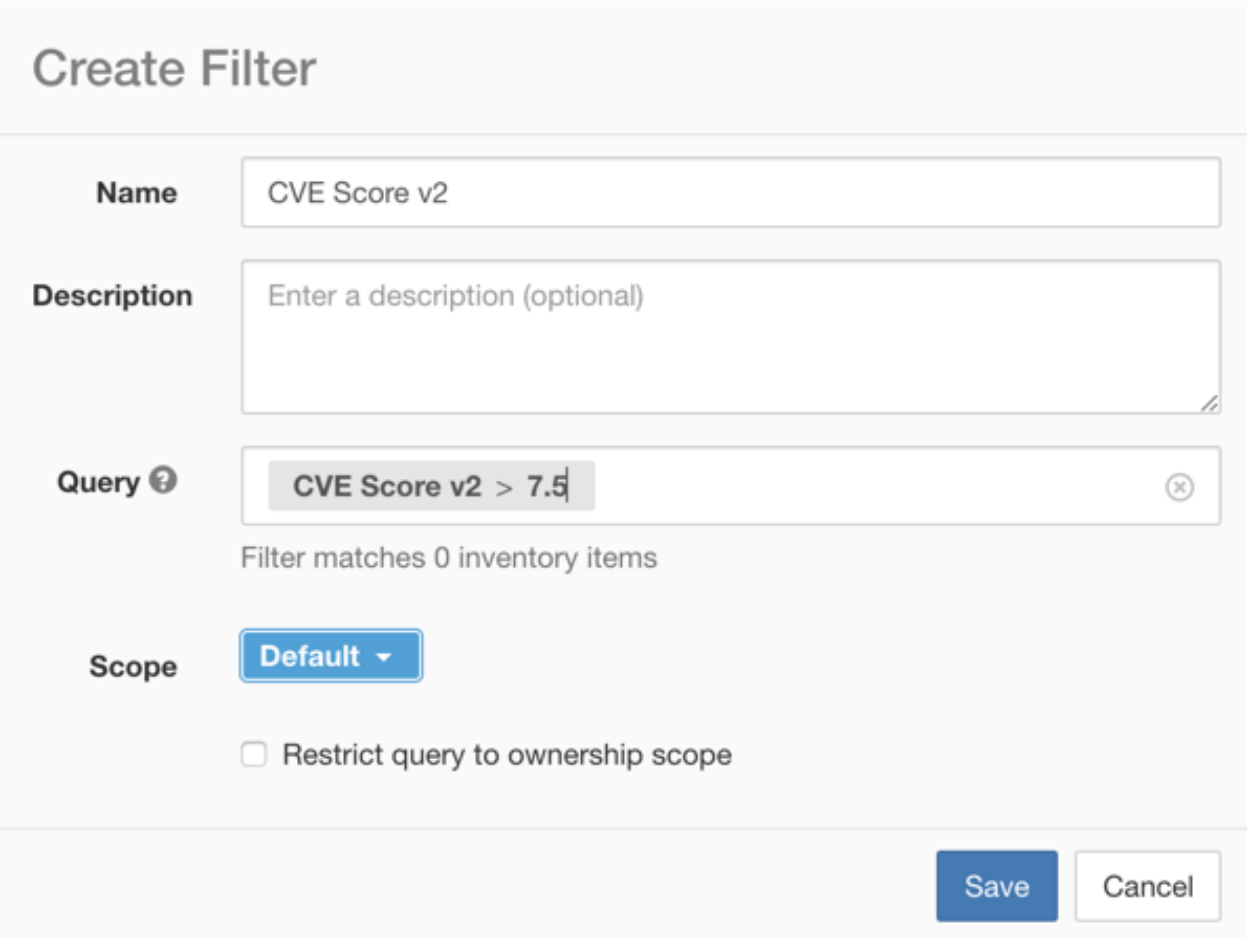


Fig. 4.7.2.2.1: Inventory filter CVSS

The following operations are supported:

- Equality - returns hosts which have CVE with the specified CVSSv2 or CVSSv3 impact scores.
- Inequality - returns hosts which don't have CVE with the specified CVSSv2 or CVSSv3 impact scores.
- Greater Than - returns hosts which have CVE with CVSSv2 or CVSSv3 impact scores greater than the specified CVSSv2 or CVSSv3 impact scores respectively.
- Greater Than or Equal To - returns hosts which have CVE with CVSSv2 or CVSSv3 impact scores greater than or equal to the specified CVSSv2 or CVSSv3 impact scores respectively.
- Less Than - returns hosts which have CVE with CVSSv2 or CVSSv3 impact scores less than the specified CVSSv2 or CVSSv3 impact scores respectively.
- Less Than or Equal To - returns hosts which have CVE with CVSSv2 or CVSSv3 impact scores less than or equal to the specified CVSSv2 or CVSSv3 impact scores respectively.

4.7.2.3 CVSSv2 based filters

Inventory filters can be created using access vectors and access complexities to identify vulnerable hosts. These filters support the following types of operations:

- Equality - returns hosts with packages affected by vulnerabilities matching the filter.

- Inequality - returns hosts with packages not affected by vulnerabilities matching the filter.

Access Vector

Access vector reflects how the vulnerability is exploited. The farther the attacker can get from the vulnerable system, the higher the base score. The table below lists different access vectors with their access requirements:

Value	Type of access
LOCAL	Physical or local (shell).
ADJACENT_NETWORK	Broadcast or collision.
NETWORK	Remotely exploitable.

Access Complexity

This metric measures the complexity in exploiting a vulnerability once the attacker is able to access the target system. The base score is inversely proportional to the access complexity. The different types of access complexities are as follows:

Value	Description
HIGH	Specialized access conditions exist.
MEDIUM	Access conditions are somewhat specialized.
LOW	Specialized access conditions do not exist.

4.7.2.4 CVSSv3 based filters

Attack vectors, attack complexities, and privilege required to influence the CVSSv3 score and can be used in inventory filters. These filters support the following operations:

- Equality - returns hosts with packages affected by vulnerabilities matching the filter.
- Inequality - returns hosts with packages not affected by vulnerabilities matching the filter.

Attack Vector

This metric reflects the context by which vulnerability exploitation is possible. The farther an attacker can get from the vulnerable component, the higher the base score. The table below lists different attack vectors with their access requirements:

Value	Type of access
LOCAL	Local (keyboard, console) or remote (SSH).
PHYSICAL	Physical access is needed.
ADJACENT_NETWORK	Broadcast or collision.
NETWORK	Remotely exploitable.

Attack Complexity

This metric describes the conditions that must exist in order to exploit the vulnerability. The base score is greatest for least complex attacks. The different types of access complexities are as follows:

Value	Description
HIGH	Significant effort needed in setting up and executing the attack.
LOW	Specialized access conditions do not exist.

Privileges Required

This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability. The base score is highest when privileges aren't needed to carry out an attack. The different values of privilege required are as follows:

Value	Privileges required
HIGH	Privileges providing significant control over the vulnerable component.
LOW	Low privileges that grant access to non-sensitive resources.
NONE	Privileges aren't needed to carry out an attack.

4.8 User Labels

In addition to the attributes discovered by Tetration agents running on inventory items, user labels can be used to add custom attributes to items. These labels enable creating inventory filters and scopes with higher precision and more flexibility.

Tetration supports four methods for adding user labels:

- Manual import from user-uploaded Comma Separate Value (CSV) files
- Manual assignment via the UI
- Automated import via *Connectors for Endpoints*
- Threat data based labels. See *Lookout Annotation* for more information.

In addition to the above methods Tetration *external orchestrators* provides an automated import method for orchestrator generated and user labels.

All user labels are prefixed by * in the UI (*user_* in OpenAPI). In addition, labels automatically imported from external orchestrators are prefixed with *orchestrator_*. For connector imported labels refer to details in *Connectors for Endpoints*, but may include labels prefixed with *ldap_*. For threat data based labels refer to details in *Lookout Annotation*; these are prefixed by *TA_*.

For example, an label with a key of *department* imported from user-uploaded CSV files will appear in the UI as * *department*, and in OpenAPI as *user_department*. A label with a key of *location* imported from an external orchestrator will appear in the UI as * *orchestrator_location*, and in OpenAPI as *user_orchestrator_location*.

4.8.1 Importing User Labels

Custom labels can be uploaded or manually assigned to associate user-defined data with specific hosts. This user-defined data will be used to annotate associated **Flows** and **Inventory**.

Before You Begin

You must be **Site Admin**, **Customer Support** or a root **scope owner** to upload, download, assign or search labels within a root scope.

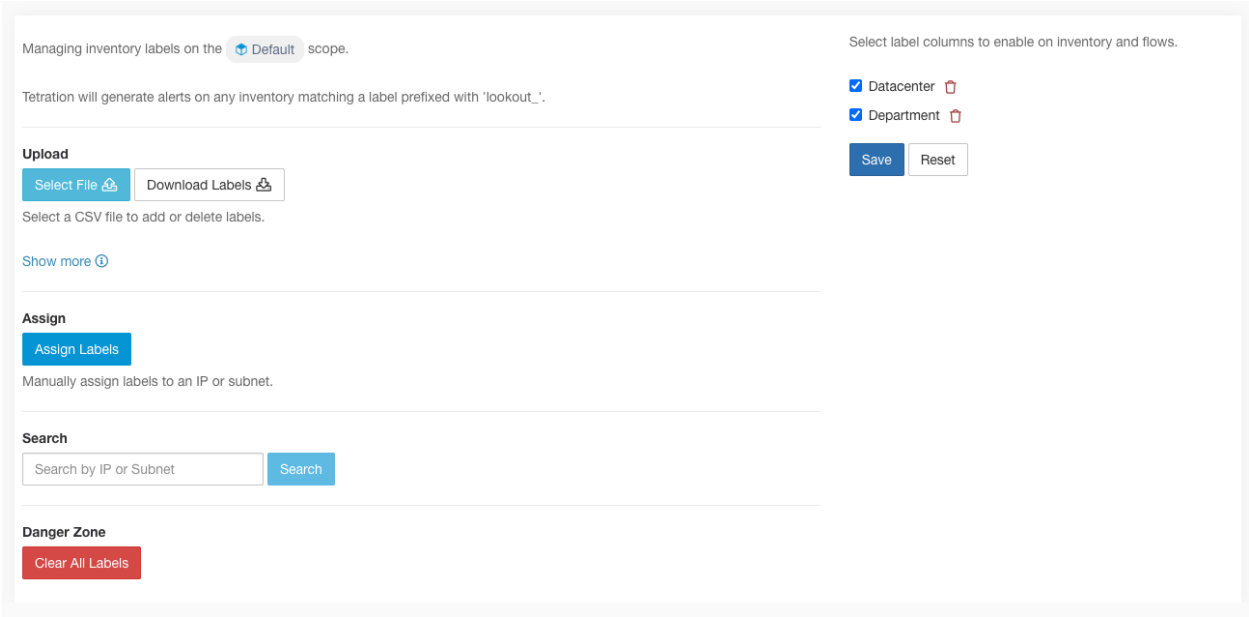


Fig. 4.8.1.1: Inventory Upload

4.8.1.1 Label limits

The limits on the number of IPv4/IPv6 addresses/subnets that can be labelled across all root scopes are as follows:

Platform	IP Address count	Subnet count
39RU Cluster	1.5 million	200 thousand
8RU Cluster	500 thousand	50 thousand
Tetration-V	70 thousand	7 thousand

On Tetration Cloud, we allow 6,000 IPv4/IPv6 addresses and 120 subnets to be labelled for every 100 licenses purchased.

Upload

This section explains how users with **Site Admin**, **Customer Support** or a root **scope owner** role can upload labels.

1. All uploaded files must follow the same schema. Click the **Show More** link for access to sample file. The uploaded files must include a label key that is IP.
2. Click **Select File**. A file dialog will appear when you can select the CSV file you would like to upload.
3. Select the operation, either Add or Delete. Add appends labels to new and existing addresses/subnets. Conflicts are resolved by selecting newer labels over existing ones. For example, if labels for an address in the database are *{"foo": "1", "bar": "2"}* and the CSV file contains *{"z": "1", "bar": "3"}*, *add* sets labels for this address to *{"foo": "1", "z": "1", "bar": "3"}*. Delete is used to remove labels for an address/subnet.
4. Click **Upload**.

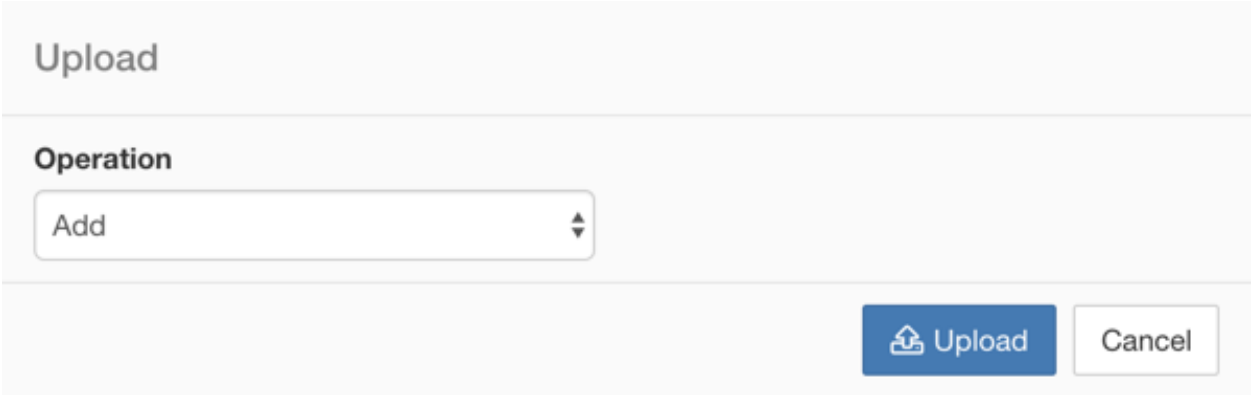


Fig. 4.8.1.1.1: Upload Modal

Note: To use non-English characters in labels, the uploaded csv file must be in UTF-8 format.

Assign

This section explains how users with **Site Admin**, **Customer Support** or a root **scope owner** role can manually assign labels.

Labels can be manually assigned to a given IP or subnet by clicking the **Assign Labels** button.

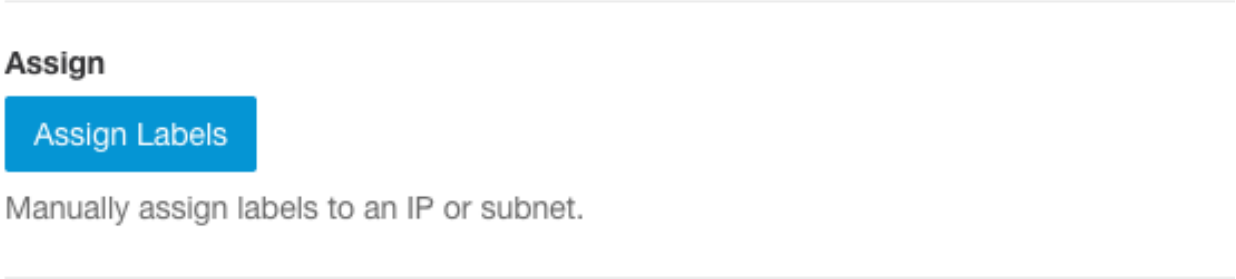


Fig. 4.8.1.1.2: Assign Labels button

1. Enter the IP or subnet. Click **Next**.
2. Existing labels will be shown and can be edited.
3. To add a new label, click **Add Label**. Enter the desired label name and value and click the checkmark. Then click **Next**.
4. Review changes and click **Assign** to commit them.

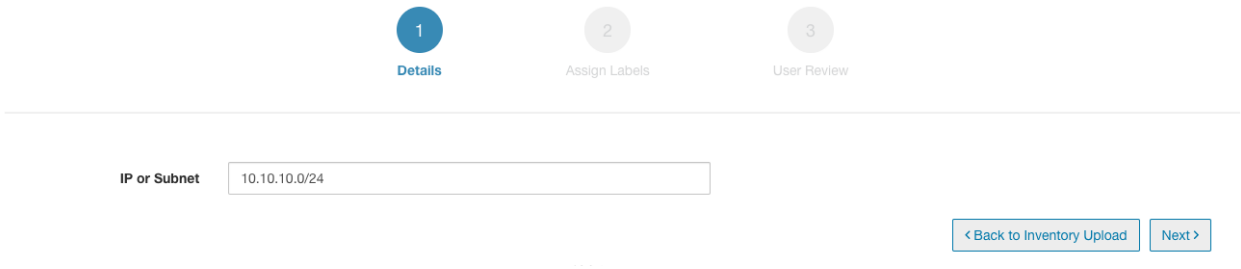


Fig. 4.8.1.1.3: Enter IP or subnet

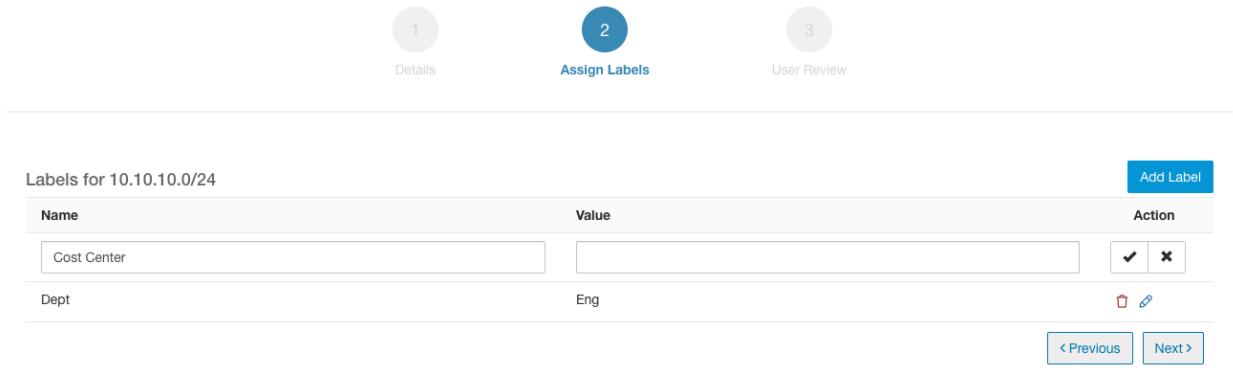


Fig. 4.8.1.1.4: Add and edit labels

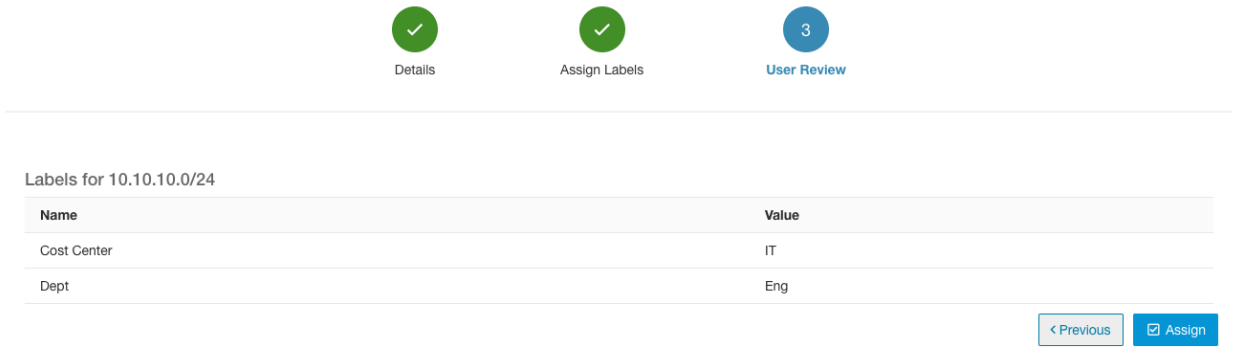


Fig. 4.8.1.1.5: Review and assign

Search

This section explains how users with **Site Admin**, **Customer Support** or a root **scope owner** role can search label definitions by entering an IP or subnet. The associated labels can be edited by clicking on the matching IP or subnet.

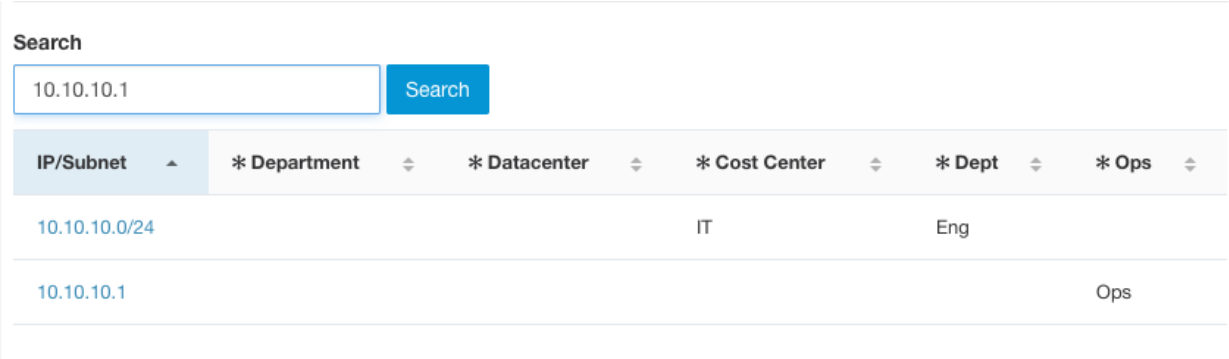


Fig. 4.8.1.1.6: Search by IP or subnet

Download

This section explains how users with **Site Admin**, **Customer Support** or a root **scope owner** role can download previously defined labels belonging to a root scope by clicking the **Download Labels** button.

Label Key Schema

Guidelines governing column names

- There must be one column with a header "IP" in the label key schema. Additionally, there must be at least one other column with attributes for the IP address.
- The column "VRF" has special significance in the label schema. If provided, it should match the root scope to which the labels are uploaded. It's mandatory when uploading the CSV file using the *scope independent API*.
- Column names should contain only ASCII characters and must be limited to 200 characters.
- Column names cannot be prefixed with "orchestrator_", "TA_", nor "LDAP_" since these can conflict with labels from internal applications.
- The CSV file should not contain duplicate column names.

Guidelines governing column values

- Values are limited to 255 characters
- **Addresses appearing under the "IP" column should conform to the following format::**
  - IPv4 addresses can be of the format "x.x.x.x" and "x.x.x.x/32".
  - IPv4 subnets should be of the format "x.x.x.x/<netmask>", where netmask is an integer between 0 and 31.
  - IPv6 addresses in the Long format ("x:x:x:x:x:x:x:x" or "x:x:x:x:x:x:x:x/128") and the Canonical format ("x:x::x" or "x:x::x/128") are supported.
  - IPv6 subnets in the Long format ("x:x:x:x:x:x:x:x/<netmask>") and the Canonical format ("x:x::x/<netmask>") are supported. Netmask must be an integer between 0 and 127.

The order of the columns does not matter. The first 32 user-defined columns will automatically be enabled for label. If more than 32 columns are uploaded, up to 32 can be enabled using the checkboxes on the right-side of the page.

Delete Columns

Columns can be deleted by clicking the "TrashCan" icon appearing near the column name on the right hand side of the page.

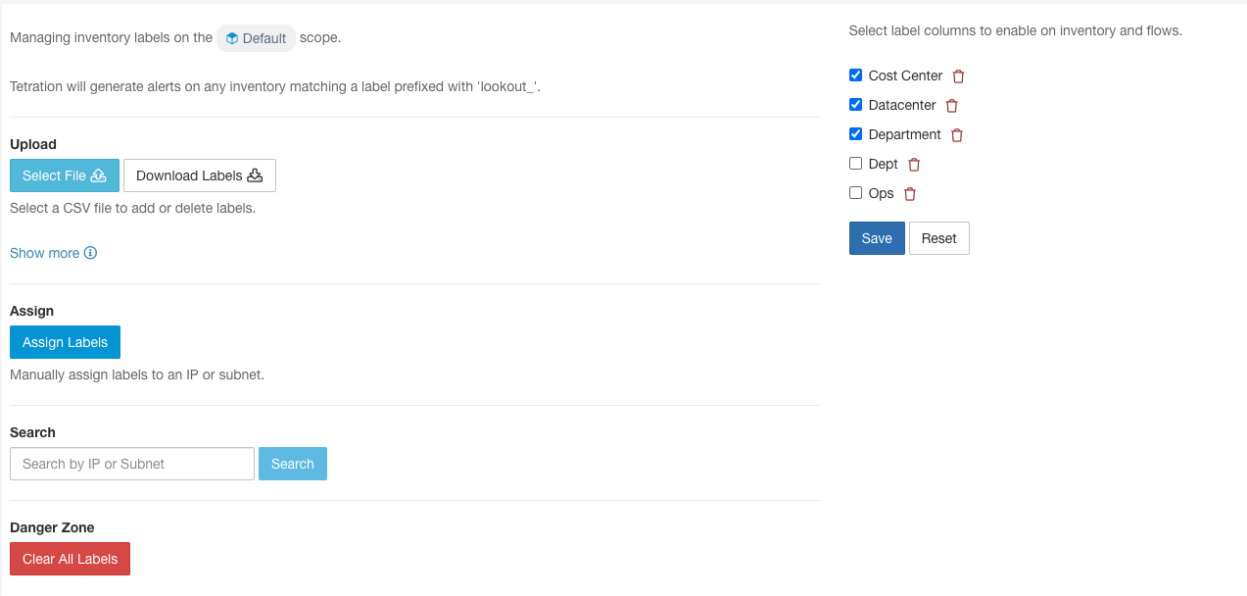


Fig. 4.8.1.1.7: Delete Columns

Clear Labels

> **Warning:** One way to change the schema is to click the **Clear Labels** button. *Proceed with caution.* This action will clear all existing labels which will impact all dependent **Filters** and **Scopes**. *Please ensure these labels are not used.* This action cannot be undone.

Subnet based Label inheritance

Subnet based label inheritance is supported in release 3.1.x and above. Smaller subnets and addresses inherit labels from larger

- the label is missing from the list of labels for the smaller subnet/address.
- the label value for the smaller subnet/address is empty.

Consider the following example,

IP	name	purpose	environment	spirit-animal
10.0.0.1	server-1	webtraffic	production	
10.0.0.2				frog
10.0.0.3				eagle
10.0.0.0/24	web-vlan		integration	
10.0.0.0/16		webtraffic		badger
10.0.0.0/8			test	bear

In release 3.0.x and below, the labels for IP address *10.0.0.3* are *{"spirit-animal": "eagle"}*.

With release 3.1.x and above, the labels for IP address *10.0.0.3* are *{"name": "web-vlan", "purpose": "webtraffic", "environment": "integration", "spirit-animal": "eagle"}*.

4.9 External Orchestrators

For deployments where an authorized system of record exists with labels for workloads, we provide a way for automatically importing the labels through external orchestrator integrations. Any modifications in the system of record will be learnt automatically by Tetration and used for updating label of the inventory table.

The picture below shows an example of inventory search using the orchestrator generated label *orchestrator_system/os_image*:

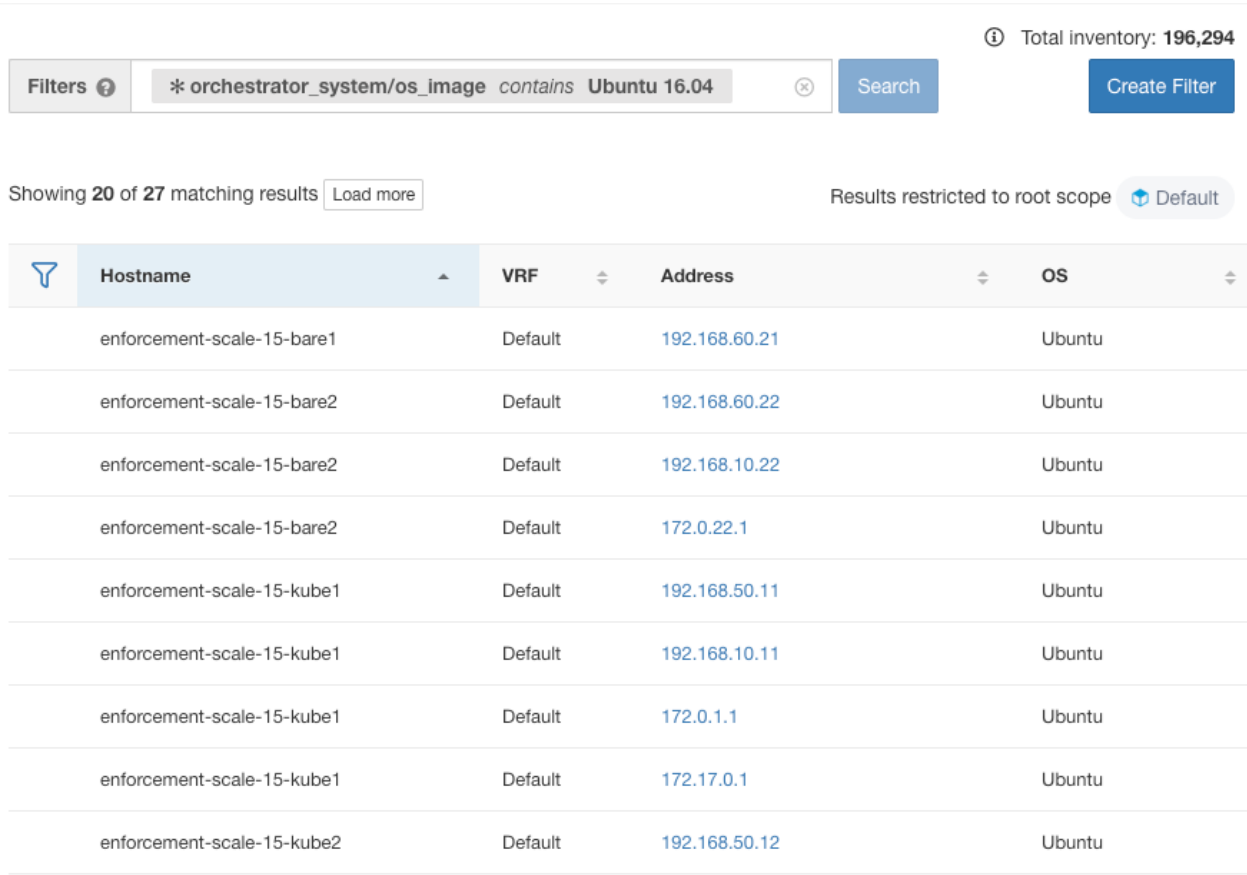


Fig. 4.9.1: Example inventory search with orchestrator generated labels

Furthermore, the imported labels allow the user to define a logical policy like

allow traffic from consumer hr_department to provider employee_db

Instead of specifying the members of the consumer and provider workload groups, we can define the logical policy using the labels as shown in picture below. Note that this allows the membership of the consumer and provider groups to be dynamically modified without the need to modify the logical policy. As workloads are added and removed from

the fleet, Tetration gets notified through the external orchestrators. This enables Tetration to evaluate the membership of the consumer group *hr_department* and the provider group *employee_db*.

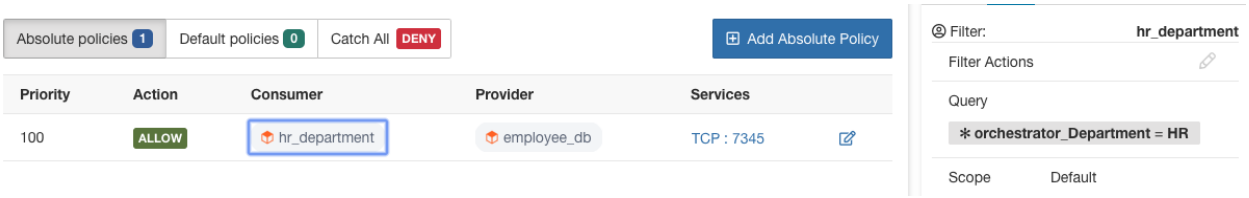


Fig. 4.9.2: Example policy with labels

The following table shows the currently supported external orchestrators:

Type	Description/When to use
VMWare vCenter	This allows Tetration to import virtual machine data such as host name, IP address and labels from a Vcenter server. The generated labels can be used to create Tetration scopes and enforcement policies.
Amazon Web Services	This allows Tetration to import data of EC2 server instances such as host name, IP address and labels from the given AWS account. The generated labels are useful to create Tetration scopes and policies.
Kubernetes/OpenShift	This allows Tetration to import Kubernetes' entities such as nodes, pods, services and and labels. These labels can be used within Tetration to define scopes and policies.
DNS	This allows Tetration to import A/AAAA and CNAME records from a DNS server via zone transfer and produces DNS names as labels, which are useful in defining Tetration scopes and policies.
Infoblox	This allows Tetration to import networks, hosts and A/AAAA records with extensible attributes from an Infoblox appliance with IPAM/DNS enabled. The imported extensible attributes can be used as labels in Tetration scopes and policies.
F5 BIG-IP	This allows Tetration to read virtual server configurations from the given F5 load balancer and generate labels for the provided services, which can be used to define enforcement policies in Tetration. The policy enforcement feature will translate them into F5 policy rules via F5 REST API.
Citrix Netscaler	This allows Tetration to read virtual server configurations from the given Netscaler load balancer and generate labels for the provided services, which can be used to define enforcement policies in Tetration. The policy enforcement feature will translate them into Netscaler ACLs via its REST API.
Cisco FMC *(BETA)*	This allows Tetration to deploy policies to all FTDs (Firepower Threat Defense) registered to the given FMC (Firepower Management Center) using the FMC's REST API.

4.9.1 List External Orchestrators

The main page for Tetration external orchestrators can be reached by selecting **Visibility > External Orchestrators** from the menu on the left pane. It shows the existing external orchestrators and provides functions to modify and delete them as well as to create new external orchestrators:

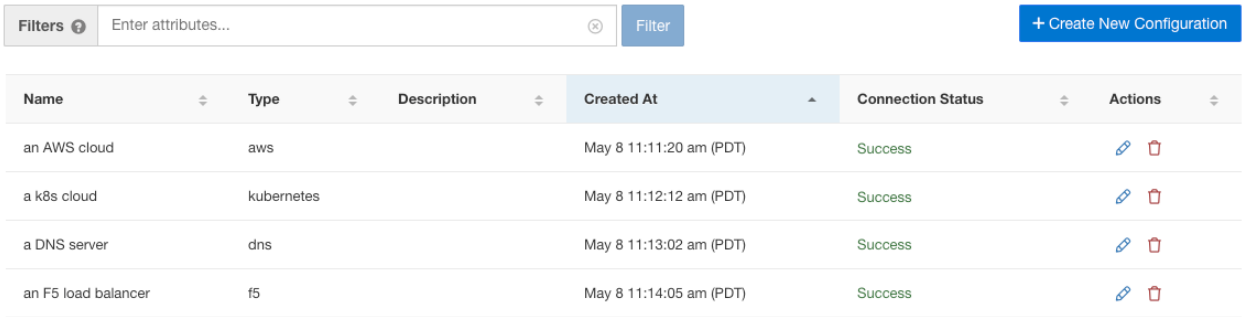


Fig. 4.9.1.1: External orchestrators' main page

Each row shows a short version of the external orchestrator with its *Name*, *Type*, *Description*, *Created at* and *Connection Status*. The latter one tells if a connection to the given external data source could be made successfully. In case of *Failure* you can click on the respective row to get more details:

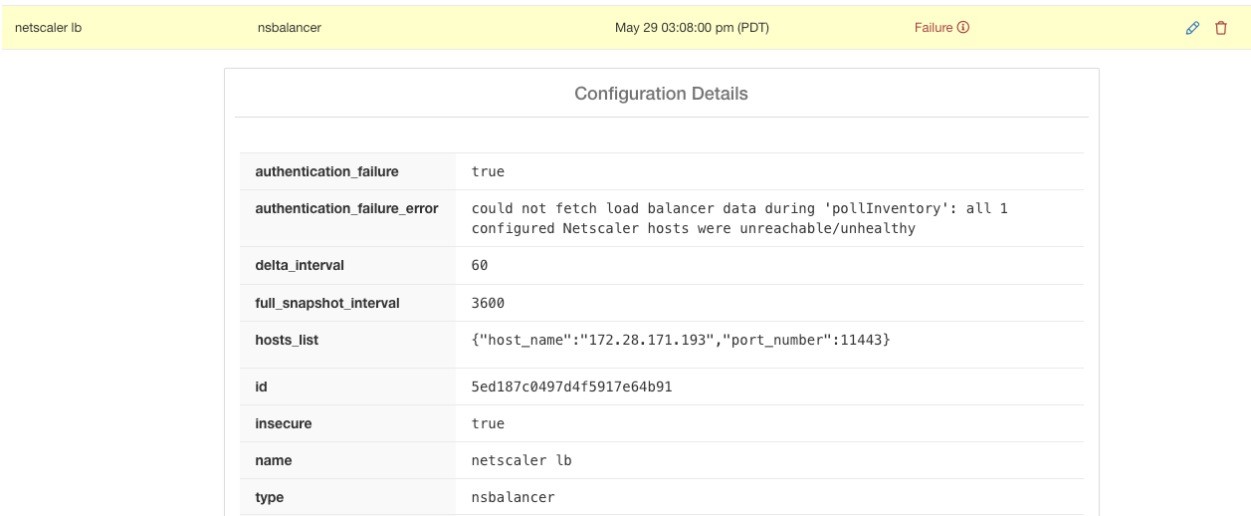


Fig. 4.9.1.2: Example External Orchestrator Authentication Failure

4.9.2 Create External Orchestrator

A new external orchestrator can be created by clicking the **Create New Configuration** button in the external orchestrators main page. This leads to a modal dialog, where you can enter a name and choose an external orchestrator type. The picture below shows the basic configuration page:

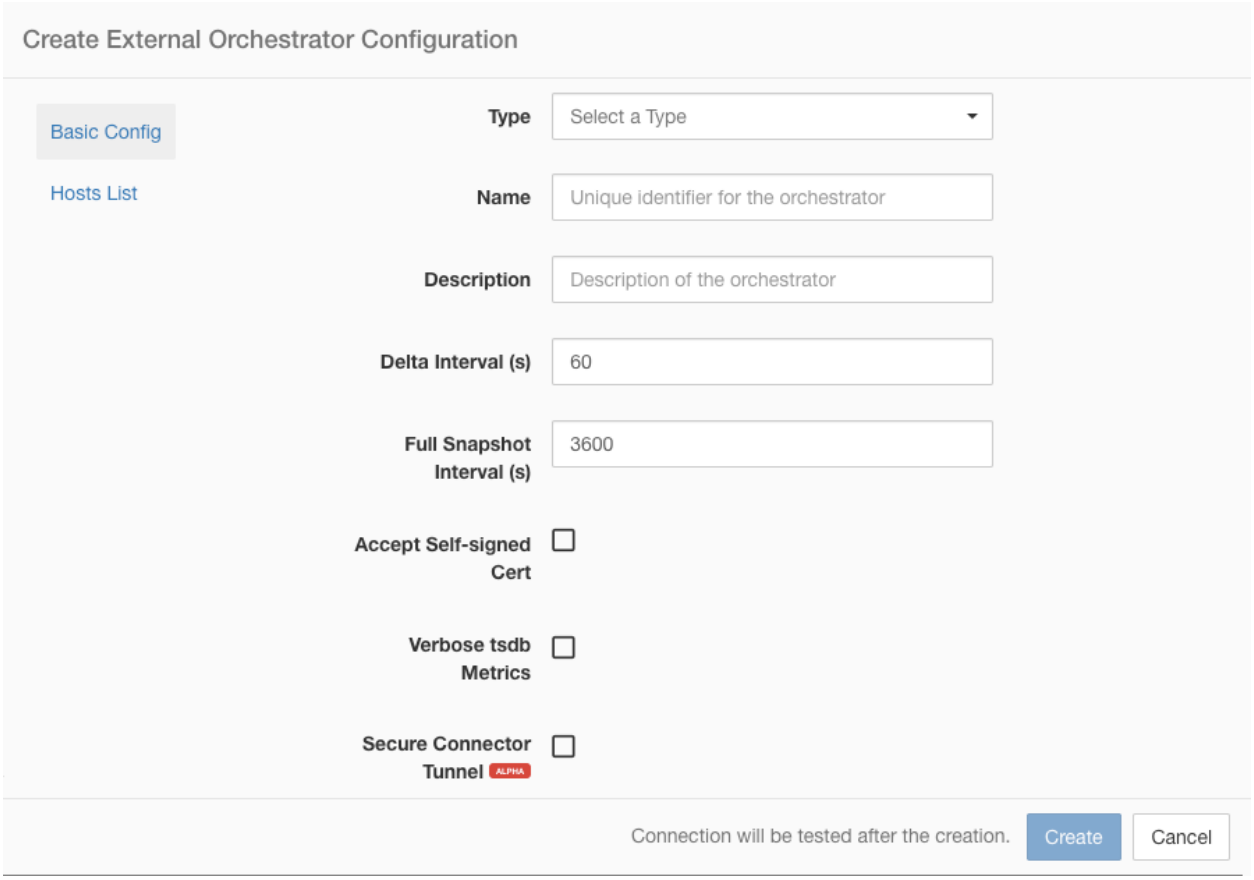


Fig. 4.9.2.1: Create External Orchestrator Configuration

The following table describes the common fields for external orchestrators. Depending on the selected type the *Basic Config* page requires additional parameters to be given. These will be covered by the respective section of the individual external orchestrators below.

Common Field	Required	Description
Type	Yes	Select one of the shown drop down list for supported external orchestrators: AWS, Vcenter, Kubernetes, F5 BIG-IP, Citrix Netscaler, Infoblox and DNS.
Name	Yes	Name of the external orchestrator, which must be unique for the active tenant.
Description	No	Description of the external orchestrator.
Full Snapshot Interval (s)	Yes	Interval in seconds the external orchestrator will try to import the full snapshot of configuration from the selected *Type*.
Accept Self-signed Cert	No	Check this option to accept self-signed server certificates for the HTTPS connection used by Tetration to retrieve configuration data from the selected *Type*. Default is not to allow self-signed server certificates.
Secure Connector Tunnel	No	Check this option to set connections to the Tetration cluster to be tunneled through a *Tetration Secure Connector* tunnel.

Note: The fields *Delta interval* and *Verbose TSDB Metrics* as shown in the picture above are optional and applicable

only for certain external orchestrators, which are explained in the respective description below.

Except for the external orchestrator type *AWS*, the *Hosts List* must be given. It specifies the network address(es) of the external data source from which the external orchestrator will fetch data and generate labels. This can be done by clicking on the tab *Hosts List* on the left hand side, which is shown in the following picture:

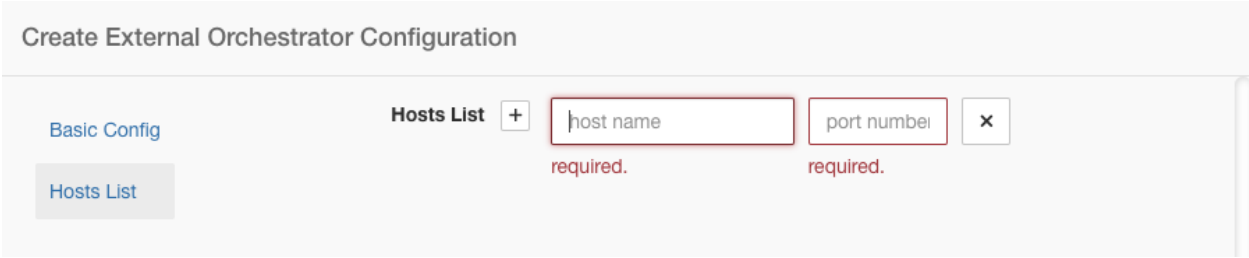


Fig. 4.9.2.2: External Orchestrator's Hosts List

In order to add new host list entry click the plus sign. Each row must contain a valid DNS host name, IPv4 or IPv6 address and a port number. Depending on the selected external orchestrator type you can enter multiple hosts for high availability or redundancy purpose. Please refer to the respective description for the chosen external orchestrator below for more details.

Click the **Create** button to create the new external orchestrator, whose configuration details can be viewed by clicking on the respective row in the list view:

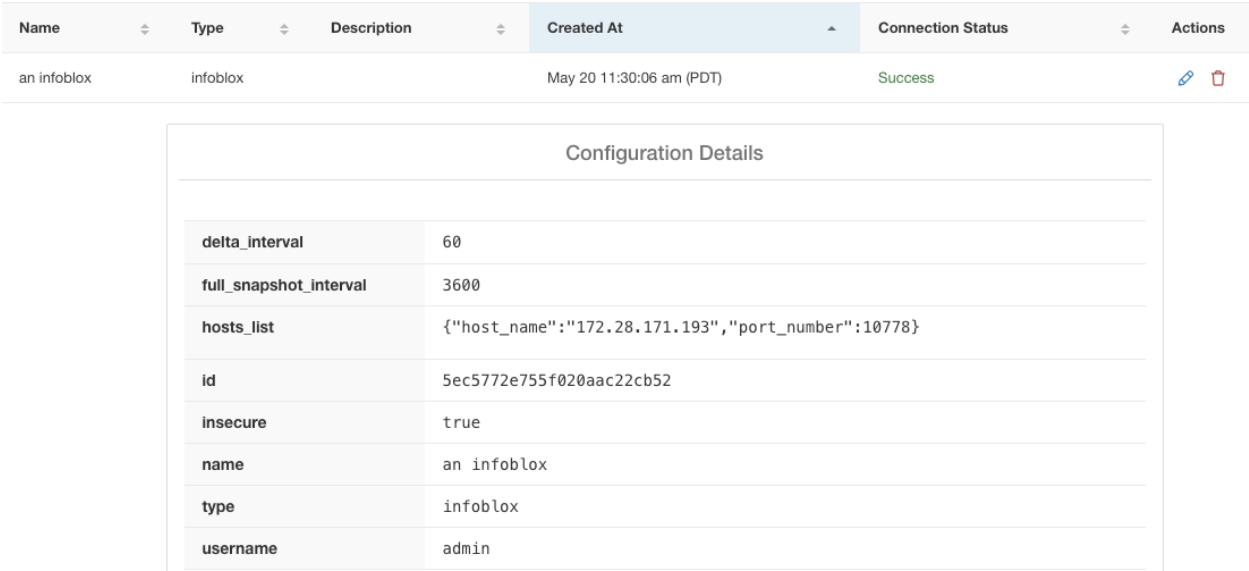


Fig. 4.9.2.3: External Orchestrator's Configuration Details

Note: Since the first full snapshot pull from an external orchestrator is an asynchronous operation, expect about one minute for the connection status field to be updated.

4.9.3 Edit External Orchestrator

Click the pencil button on the right hand side of an external orchestrator row as shown below to open a modal dialog similar to the one for creating an external orchestrator, where the configuration can be modified.

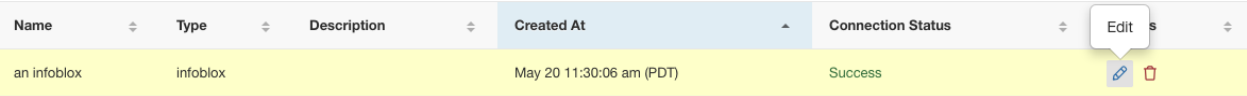


Fig. 4.9.3.1: Edit External Orchestrator

Note:

- The **Type** field is not editable.
- If a configuration uses keys/certificates for authentication, the keys and certificates have to be provided every time the configuration is updated.
- Since the configuration changes of an external orchestrator is an asynchronous operation, expect about one minute for the connection status field to be updated and to confirm the correctness of entered changes.

Click the **Update** button to save the changes made to the configuration.

4.9.4 Delete External Orchestrator

In order to delete an external orchestrator click the trash bin button as shown below:

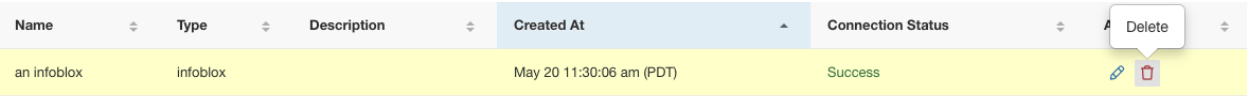


Fig. 4.9.4.1: Delete External Orchestrator

4.9.5 Orchestrator Generated Labels

These labels add metadata to the orchestrator learned inventories. This metadata facilitates in different kinds of filters both for visibility and policies.

For example, if a user wants to create a inventory filter encapsulating all inventories belonging to certain orchestrator, this can be done using the cluster_name.

4.9.6 Amazon Web Services

Tetration supports automated ingestion of inventory live from an AWS region. When an external orchestrator configuration is added for type "aws", the Tetration appliance will connect to the AWS endpoint and fetch the metadata for all the instances in running/stopped state.

4.9.6.1 Prerequisites

- Security tokens (access key and secret key) used should have the right kind of IAM privileges to allow fetching of orchestrator information.

4.9.6.2 Configuration fields

Attribute	Description
ID	Unique identifier for the orchestrator.
Name	User-specified name of the orchestrator.
Type	Type of orchestrator - (*aws* in this case)
Description	A brief description of the orchestrator.
AWS Access Key ID	ACCESS KEY associated with the account for which orchestrator config is being created.
AWS Secret Access key	SECRET KEY associated with the account for which orchestrator config is being created. Please note that SECRET KEY has to be re-entered every time the config is edited.
AWS Region	The Region in which workload has been deployed. If a workload is spread across multiple regions, a separate config is required for every region. Please refer to the link below for correct *region* values. :ref: https://docs.aws.amazon.com/general/latest/gr/rande.html.
Accept Self-signed Cert	Is automatically marked true for AWS. User cannot edit it.
Full Snapshot Interval	Full snapshot interval in seconds. Orchestrator Inventory manager will perform a full refresh poll from the orchestrator.
Delta Snapshot Interval	Delta snapshot interval in seconds. Orchestrator Inventory manager will only fetch incremental updates from the orchestrator.
Hosts List	AWS orchestrator type doesn't require hosts list. The endpoint for AWS will be derived from *AWS Region* field above. This field should be left empty.
Verbose TSDB metrics	If enabled, tsdb metrics for each individual orchestrator will be reported. Else an aggregation of all orchestrator metrics will be reported.
Secure Connector Tunnel	Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel.

4.9.6.3 Workflow

- Configure an AWS orchestrator filled with the configuration fields above.

4.9.6.4 Orchestrator generated labels

Tetration adds the following labels to all the AWS instances.

Key	Value
orchestrator_system/orch_type	aws
orchestrator_system/cluster_name	*<Cluster_name is the name given by the user for this orchestrator's configuration>*
orchestrator_system/cluster_id	*<UUID of the orchestrator's configuration in |product|>*

4.9.6.5 Instance-specific labels

The following labels are instance specific.

Key	Value
orchestrator_system/workload_type	vm
orchestrator_system/machine_id	*<InstanceID assigned by AWS>*
orchestrator_system/machine_name	*<PublicDNS(FQDN) given to this node by AWS>*
orchestrator_'<AWS Tag Key>'	*<AWS Tag Value>*

4.9.6.6 Troubleshooting

- Confusion between AWS Region and Availability Zone.

  Both these values are interrelated and should not be confused. For example us-west-1 might be the region and availability zone can be either of us-west-1a or us-west-1b etc. While configuring orchestrator, *Region* should be used. Refer to https://docs.aws.amazon.com/general/latest/gr/rande.html for all regions.

- Connectivity/Credentials issue after updating the orchestrator config.

  Customers must re-submit the *AWS Secret Key* every time the config gets updated.

4.9.7 Kubernetes/OpenShift

Tetration supports automated ingestion of inventory live from a Kubernetes cluster. When an external orchestrator configuration is added for a Kubernetes/OpenShift cluster, Tetration connects to the cluster's API server and tracks the status of nodes, pods and services in that cluster. For each object type, Tetration imports all Kubernetes labels and labels associated with the object. Label keys are imported as is, and label keys are prefixed with *annotation/*. All values are imported as is.

In addition to importing the labels defined for Kubernetes/OpenShift objects, Tetration also generates a number of labels that facilitate the use of these objects in inventory filters. These additional labels are especially useful in defining scopes and policies. If enforcement is enabled on the Kubernetes nodes (enforcement agents are installed and the configuration profile enables enforcement on these agents), enforcement policies will be installed in both the nodes as well as inside the pod namespaces using the information ingested about the Kubernetes entities via this integration.

Tetration supports configuration of the following managed kubernetes services as external orchestrator:

- Amazon Elastic Kubernetes Service(EKS): Amazon EKS gives users the flexibility to start, run, and scale Kubernetes applications in the AWS cloud or on-premises. It is a fully managed service that offers high availability, security and integration with AWS services like IAM, VPC, STS, etc.

4.9.7.1 Prerequisites

- Secure Connector tunnel, if needed for connectivity.
- Kubernetes 1.[12-18]

4.9.7.2 Configuration fields

The following configuration fields pertain to Kubernetes Orchestrator configuration in the Orchestrator Object.

Field	Description
Name	User specified name of the orchestrator.
Description	User specified description of the orchestrator.
Delta Interval	Interval (in seconds) to check the Kubernetes endpoint for changes
Full Snapshot Interval	Interval (in seconds) to perform a full snapshot of Kubernetes data
Username	Username for the orchestration endpoint.
Password	Password for the orchestration endpoint.
Certificate	Client certificate used for authentication.
Key	Key corresponding to client certificate.
Auth Token	Opaque authentication token (bearer token).
CA Certificate	CA Certificate to validate orchestration endpoint.
Accept Self-Signed Cert	Checkbox to disable strictSSL checking of the Kubernetes API server certificate
Verbose TSDB Metrics	Maintain per Kubernetesorchestrator metrics - if set to False, only Tetration cluster-wide metrics are maintained.
Secureconnector Tunnel	Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel
Hosts List	Array of { "host_name", port_number} pairs that specify how Tetration must connect to the orchestrator
K8s manager type	Manager type for the kubernetes cluster(None for Vanilla/Openshift kubernetes deployments)
AWS cluster name	Name of the orchestrator as specified at time of creation of cluster(EKS only)
AWS Access ID	ACCESS KEY associated with the account for which orchestrator config is being created(EKS only)
AWS Secret Access Key	SECRET KEY associated with the account for which orchestrator config is being created. Please note that SECRET KEY has to be re-entered every time the config is edited.(EKS only)
AWS Region	The Region in which workload has been deployed. If a workload is spread across multiple regions, a separate config is required for every region. Please refer to the link below for correct *region* values. :ref: https://docs.aws.amazon.com/general/latest/gr/rande.html. (EKS only)
AWS Assume Role ARN	Amazon resource number of the role to assume while connecting to the orchestrator. :ref: https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html (EKS only)

4.9.7.3 Orchestrator Golden Rules

The golden rules object attributes are described below. These golden rules allow a concise specification of rules necessary for the Kubernetes cluster to stay functional once enforcement is enabled on the Kubernetes cluster nodes.

Attribute	Description
Kubelet Port	Kubelet node-local API port
Services	Array of Kubernetes Services objects

The kubelet port is necessary to create policies to allow traffic from the Kubernetes management daemons to kubelets

such as for live logs, execs of pods in interactive mode etc. Vital connectivity between the various kubernetes services and daemons is specified as a series of services - each entry in the services array has the following structure

- Description: A string that describes the service
- Addresses: A list of service endpoint addresses of the format <IP>:<port>/<protocol>.
- Consumed By: A list of consumers of the endpoints (allowed values are Pods or Nodes)

Note: If **kubernetes** is chosen as the type, Golden Rules configuration will be allowed.

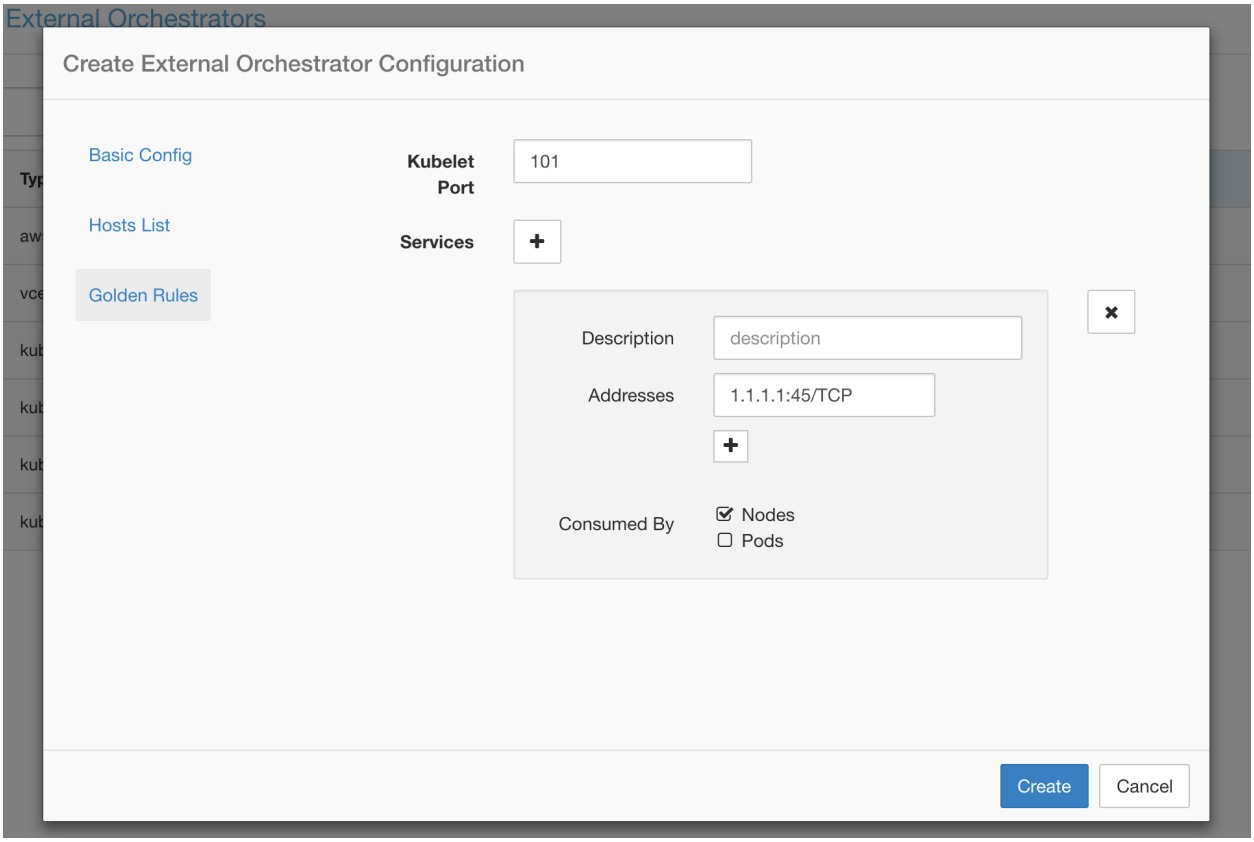


Fig. 4.9.7.3.1: Create Golden Rules Configuration for Kubernetes Type

4.9.7.4 Workflow

- Configure Secure Connector tunnel, if needed, for connectivity from the Tetration cluster to a Kubernetes API server (or servers).
- Configure a Kubernetes orchestrator filled with the configuration fields above.
- Configure the Golden Rules for the Kubernetes orchestrator.

4.9.7.5 Orchestrator-generated labels

4.9.7.6 Generated labels for all resources

Tetration adds the following labels to all the nodes, pods and services retrieved from the Kubernetes/OpenShift API server.

Key	Value
orchestrator_system/orch_type	kubernetes
orchestrator_system/cluster_id	*<UUID of the cluster's configuration in |product|>*
orchestrator_system/cluster_name	*<Name given to this cluster's configuration>*
orchestrator_system/namespace	*<The Kuberentes/OpenShift namespace of this item>*

4.9.7.7 Node-specific labels

The following labels are generated for nodes only.

Key	Value
orchestrator_system/workload_type	machine
orchestrator_system/machine_id	*<UUID assigned by Kubernetes/OpenShift>*
orchestrator_system/machine_name	*<Name given to this node>*
orchestrator_system/kubelet_version	*<Version of the kubelet running on this node>*
orchestrator_system/container_runtime_version	*<The container runtime version running on this node>*

4.9.7.8 Pod-specific labels

The following labels are generated for pods only.

Key	Value
orchestrator_system/workload_type	pod
orchestrator_system/pod_id	*<UUID assigned by Kubernetes/OpenShift>*
orchestrator_system/pod_name	*<Name given to this pod>*
orchestrator_system/hostnetwork	*<true|false> reflecting whether the pod is running in the host network*
orchestrator_system/machine_name	*<Name of the node the pod is running on>*
orchestrator_system/service_endpoint	*[List of service names this pod is providing]*

4.9.7.9 Service-specific labels

The following labels are generated for services only.

Key	Value
orchestrator_system/workload_type	service
orchestrator_system/service_name	*<Name given to this service>*

Tip: Filtering items using **orchestrator_system/service_name** is not the same as using **orchestrator_system/service_endpoint**.

For example, using the filter **orchestrator_system/service_name = web** selects all *services* with the name **web** while **orchestrator_system/service_endpoint = web** selects all *pods* that provide a service with the name **web**.

4.9.7.10 Example

The following example shows a partial YAML representation of a Kubernetes node and the corresponding labels imported by Tetration.

```
- apiVersion: v1
kind: Node
metadata:
  annotations:
    node.alpha.kubernetes.io/ttl: "0"
    volumes.kubernetes.io/controller-managed-attach-detach: "true"
  labels:
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/os: linux
    kubernetes.io/hostname: k8s-controller
```

Imported label keys
orchestrator_beta.kubernetes.io/arch
orchestrator_beta.kubernetes.io/os
orchestrator_kubernetes.io/hostname
orchestrator_annotation/node.alpha.kubernetes.io/ttl
orchestrator_annotation/volumes.kubernetes.io/controller-managed-attach-detach
orchestrator_system/orch_type
orchestrator_system/cluster_id
orchestrator_system/cluster_name
orchestrator_system/namespace
orchestrator_system/workload_type
orchestrator_system/machine_id
orchestrator_system/machine_name
orchestrator_system/kubelet_version
orchestrator_system/container_runtime_version

4.9.7.11 Kubernetes RBAC Resource Considerations

The Kubernetes client attempts to GET/LIST/WATCH the following resources. It is highly recommended NOT to configure the admin key/cert or an admin service account.

The provided Kubernetes authentication credentials should have a minimum set of privileges to the following resources:

Resources	Kubernetes Verbs
endpoints	[get list watch]
namespaces	[get list watch]
nodes	[get list watch]
pods	[get list watch]
services	[get list watch]
ingresses	[get list watch]

Essentially, you can create a special service account on your Kubernetes server with these minimal privileges. An example sequence of kubectl commands is below that will facilitate the creation of this serviceaccount. Note the use of the clusterrole (not role) and clusterrolebindings (not rolebindings) - these are cluster-wide roles and not per namespace. Using a role/rolebinding will not work as Tetration attempts to retrieve data from all namespaces.

```
$ kubectl create serviceaccount tetration.read.only
$ kubectl create clusterrole tetration.read.only --verb=get,list,watch
--resource=endpoints,namespaces,nodes,pods,services,ingresses
$ kubectl create clusterrolebinding tetration.read.only
--clusterrole=tetration.read.only --serviceaccount=default:tetration.read.only
```

To retrieve the authtoken secret from the serviceaccount (used in the Auth Token field in the GUI) and decode from base64, you can retrieve the name of the secret by listing the serviceaccount with yaml output.

```
$ kubectl get serviceaccount -o yaml tetration.read.only
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: 2020-xx-xxT19:59:57Z
  name: tetration.read.only
  namespace: default
  resourceVersion: "991"
  selfLink: /api/v1/namespaces/default/serviceaccounts/e2e.minimal
  uid: ce23da52-a11d-11ea-a990-525400d58002
secrets:
- name: tetration.read.only-token-vmvmz
```

Listing the secret in yaml output mode will yield the token but in Base64 format (which is standard Kubernetes procedure for secret data). Tetration does not accept the token in this format, you must decode it from Base64.

```
$ kubectl get secret -o yaml tetration.read.only-token-vmvmz
apiVersion: v1
data:
  ca.crt: ...
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVX....HRfZ2JwMVZR
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: tetration.read.only
    kubernetes.io/service-account.uid: ce23da52-a11d-11ea-a990-525400d58002
  creationTimestamp: 2020-05-28T19:59:57Z
  name: tetration.read.only-token-vmvmz
  namespace: default
  resourceVersion: "990"
  selfLink: /api/v1/namespaces/default/secrets/tetration.read.only-token-vmvmz
  uid: ce24f40c-a11d-11ea-a990-525400d58002
type: kubernetes.io/service-account-token
```

To list the secret and output only the `.data.token` field and decode from base 64 encoding in one command, the following command that use the `--template` option is helpful.

```
$ kubectl get secret tetration.read.only-token-vmvmz
--template "{% raw %}{{ .data.token }}{% endraw %}" | base64 -d
```

This authtoken can be used for configuring a Kubernetes orchestrator in the Tetration UI instead of username/password or key/cert.

EKS specific RBAC considerations

User credentials and AssumeRole (if applicable) must be configured with minimum set of privileges. The user/role must be specified in the aws-auth.yaml config map. aws-auth.yaml can be edited using the following command.

```
$ kubectl edit configmap -n kube-system aws-auth
```

If AssumeRole is not used, the user must be added to the "mapUsers" section of the aws-auth.yaml with appropriate group. If AssumeRole ARN is specified, the role must be added to the "mapRoles" section of the aws-auth.yaml. A sample aws-auth.yaml with AssumeRole is provided below.

```
apiVersion: v1
data:
  mapAccounts: |
    []
  mapRoles: |
    - "groups":
      - "system:bootstrappers"
      - "system:nodes"
      "rolearn": "arn:aws:iam::938996165657:role/eks-cluster-
↪20210114181445234700000000a"
      "username": "system:node:{{EC2PrivateDNSName}}"
    - "rolearn": arn:aws:iam::938996165657:role/BasicPrivilegesRole
      "username": tetration.read.only-user
      "groups":
        - tetration.read.only

  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2021-01-14T18:14:47Z"
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:mapAccounts: {}
        f:mapRoles: {}
        f:mapUsers: {}
    manager: HashiCorp
    operation: Update
    time: "2021-01-14T18:14:47Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "829"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: 6c5a3ac7-58c7-4c57-a9c9-cad701110569
```

4.9.7.12 Policy Enforcement on Kubernetes Nginx Ingress controller running in Hostnetwork mode

Tetration will enforce policies both at the nginx ingress controller and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.

Note: If the ingress controller is not running in hostnetwork mode please refer IngressControllerAPI

Note: IBM-ICP uses Kubernetes Nginx Ingress controller by default and runs on control plane nodes in hostnetwork mode.

Following are the steps to enforce policy using Kubernetes Nginx Ingress controller.

1. Create an external orchestrator for Kubernetes/OpenShift as described here.

```
➜  ~
➜  ~ k8s get ingress
NAME           HOSTS   ADDRESS         PORTS   AGE
test-ingress   *       192.168.60.100  80      7s
➜  ~
```

2. Create an ingress object in the Kubernetes cluster. A snapshot of the yaml file used to create ingress object is provided in the following picture.

```
▶ k8s get ingress
NAME                    HOSTS   ADDRESS         PORTS   AGE
svc-ce2e-teeksitlbiwlc  *       192.168.10.13   80      74s
```

```
~
▶ k8s get ingress -o yaml
apiVersion: v1
items:
- apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    annotations:
      virtual-server.f5.com/ip: 192.168.10.13
      virtual-server.f5.com/partition: k8scluster
    creationTimestamp: "2020-06-26T21:31:01Z"
    generation: 1
    labels:
      e2e-test: "yes"
    name: svc-ce2e-teeksitlbiwlc
    namespace: default
    resourceVersion: "1074475"
    selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/svc-ce2e-teeksitlbiwlc
    uid: 5526b4a3-b7f4-11ea-aa09-525400d58002
  spec:
    backend:
      serviceName: svc-ce2e-teeksitlbiwlc
      servicePort: 80
  status:
    loadBalancer:
      ingress:
      - ip: 192.168.10.13
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

3. Deploy Kubernets Nginx Ingress controller in the Kubernetes cluster. IBM-ICP Ingress controller pods are running on control plane nodes by default.

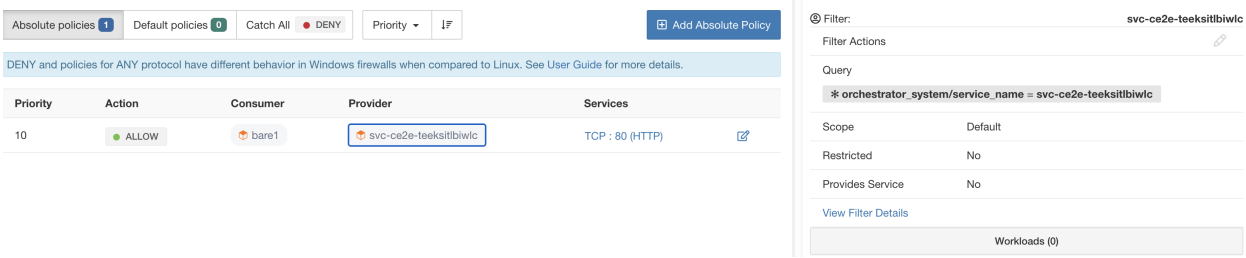
```
~
▶ k8s get pods -o wide -n ingress-nginx
NAME                                        READY   STATUS    RESTARTS   AGE     IP              NODE                          NOMINATED NODE
nginx-ingress-controller-6bc9c6745c-scfzs   1/1     Running   0          2m11s   192.168.10.13   enforcement-scale-16-kube3   <none>

~
▶ k8s get node enforcement-scale-16-kube3 -o wide
NAME                          STATUS   ROLES    AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
enforcement-scale-16-kube3   Ready    <none>   7d5h   v1.12.3   192.168.10.13   <none>        Ubuntu 16.04.5 LTS   4.4.0-139-generic   docker://18.6.1
```

4. Create a backend service which will be accessed by the consumers outside the cluster. In the example provided below we have created a simple *svc-ce2e-teeksitlbiwlc* (http-echo) service.

```
~
▶ k8s get svc svc-ce2e-teeksitlbiwlc
NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
svc-ce2e-teeksitlbiwlc   ClusterIP   10.102.30.231   <none>        80/TCP    6m11s
```

5. Create a policy between external consumer and backend service. Enforce the policy using *Policy Enforcement* tab.



6. In case of Nginx ingress controller Tetration software will apply the appropriate allow/drop rule where the source will be consumer specified in the above step and destination will be corresponding Ingress controller pod IP. In case of backend pods, Tetration software will apply the appropriate allow/drop rule where the source will be Ingress pod and destination will be the backend pod IP.

4.9.7.13 Policy Enforcement on Kubernetes Nginx/Haproxy Ingress controller running as Deployment/Daemonset

Tetration will enforce policies both at the ingress controller and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.

Following are the steps to enforce policies on Ingress controller.

1. Create/Update an external orchestrator for Kubernetes/OpenShift using OpenAPI. See *Orchestrators* for information on creating the external orchestrator using OpenAPI. Add information of Ingress Controllers for External Orchestrator config.
2. Create an ingress object in the Kubernetes cluster.
3. Deploy Ingress controller in the Kubernetes cluster.
4. Create a backend service which will be accessed by the consumers outside the cluster.
5. Create a policy between external consumer and backend service. Enforce the policy using *Policy Enforcement* tab.
6. In case of Ingress controllers Tetration software will apply the appropriate allow/drop rule where the source will be consumer specified in the above step and destination will be corresponding Ingress controller pod IP. In case of backend pods, Tetration software will apply the appropriate allow/drop rule where the source will be Ingress pod and destination will be the backend pod IP.

4.9.7.14 Troubleshooting

- Client key/certificate Credentials parsing/mismatch

  These must be supplied in PEM format and be the correct entry from the kubectl.conf file. We have encountered customers pasting CA certs into client cert fields, as well as keys and certs not matching each other.

- gcloud credentials instead of GKE credentials

  Customers using GKE under the gcloud CLI mistakenly provide the gcloud credentials when the GKE cluster credentials are needed.

- Kubernetes cluster version unsupported

  Using an incompatible version of Kubernetes may result in failures. Verify that the Kubernetes version is in the supported versions list.

- Credentials have insufficient privileges

  verify that the authtoken or user or client key/cert used has all the privileges listed in the table above.

- Kubernetes inventory keeps flipping around

  The hosts_list field specifies a pool of API servers for the same Kubernetes cluster - you cannot use this to configure multiple Kubernetes clusters. Tetration will probe for aliveness and randomly select one of these endpoints to connect to and retrieve the Kubernetes inventory information. No load balancing is performed here, nor is there a guarantee of evenly distributing load across these endpoints. If these are different clusters, the Kubernetes inventory will keep flipping between them, depending on which cluster's API server we connect to.

- Multiple authorization methods

  Multiple authorization methods may be filled in during configuration (username/password, authtoken, client key/certificate) and will be used in the client connection established with the API server. The standard Kubernetes rules for valid simultaneous authorization methods apply here.

- SSL Certificate validation fails

  If the Kubernetes API endpoint is behind a NAT or load balancer, then the DN in the SSL certificate generated on the kube control plane nodes may mismatch with the IP address configured in Tetration. This will cause an SSL validation failure even if the CA certificate is provided and is valid. The Insecure knob bypasses strict server SSL certificate validation and will help workaround this issue but can lead to MITM issues. The correct fix for this is to change the CA certificate to provide SAN (Subject Alternative Name) entries for all DNS/IP entries that can be used to connect to the Kubernetes cluster.

4.9.8 VMWare vCenter

vCenter integration allows user to fetch bare metal and VM attributes from configured vCenter.

When an external orchestrator configuration is added for type "Vcenter", Tetration fetches bare metal and VM attributes for all the bare metals and VM's controlled by that vCenter instance. Tetration will import the following attributes of a bare metal/VM:- a) Hostname b) IP addresses c) BIOS UUID d) Categories/Labels.

A new inventory will be created in Tetration with the above bare metal/VM attributes, if the inventory is not present in the appliance. If the inventory is already present in the appliance (created by Tetration visibility sensor running on the bare metal/VM), the existing inventory will be labelled with the fetched bare metal/VM Categories/Labels list.

4.9.8.1 Prerequisites

- Secure Connector Tunnel, if needed for connectivity.
- vCenter version supported is 6.5+

4.9.8.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

- **Hosts List** is an array of hostname/ip and port pairs pointing to the vCenter server from which bare metal/VM attributes will be fetched.

4.9.8.3 Workflow

- First, the user must verify that the vCenter server is reachable on that IP/Port from the Tetration cluster.
- For TaaS or in cases where the vCenter server is not directly reachable, the user must configure a secure connector tunnel to provide connectivity.

4.9.8.4 Orchestrator generated labels

Tetration adds the following labels to all the VM's learnt from vCenter server.

Key	Value
orchestrator_system/orch_type	Vcenter
orchestrator_system/cluster_name	*<Name given to this cluster's configuration>*
orchestrator_system/cluster_id	*<UUID of the cluster's configuration in |product|>*

4.9.8.5 Instance-specific labels

The following labels are instance specific.

Key	Value
orchestrator_system/workload_type	vm
orchestrator_system/machine_id	*BIOS UUID of bare metal/VM*
orchestrator_system/machine_name	*Hostname of the bare metal/VM*
orchestrator_'<Category Name>'	*<Tag Value>*

4.9.8.6 Caveats

- When an external orchestrator configuration is added for vCenter, Tetration software will connect to the vCenter server specified in the hosts list. After the connection to the server is successful, Tetration software will import hostnames, IP addresses and Category/Labels for all the bare metals and Virtual Machines present in the vCenter server. In order to import hostnames and IP addresses of the bare metals and VM's, VM tools must be installed on all the bare metals and VM's. If VM tools is not installed for a given bare metal/Virtual Machine, Tetration software will not display Category/Labels for that particular bare metal/VM.
- Tetration software doesn't import Custom attributes of the bare metal/VM.
- It is recommended to set **Delta** interval timer to more than 10 min so as to reduce the load on the vCenter server. Any change in the inventory/labels on the vCenter server will have a propagation delay of at least 10 min, once the above mentioned timer is modified.

4.9.8.7 Troubleshooting

- Connectivity Issues

  In case, Tetration appliance is not able to connect/reach the vCenter server, **Connection Status** tab of the External orchestrator will display the failure status along with the appropriate error if any.

- Tetration software health check.

  Please check the **MAINTENANCE/Service Status** page to see if any service is down. Please check if **OrchestratorInventoryManager** is up and running.

4.9.9 DNS

The DNS Integration allows Tetration to annotate known inventory with DNS information such as hostnames from CNAME and A/AAAA records.

When an external orchestrator configuration is added for type "dns", the Tetration appliance will attempt to connect to the DNS server(s) and perform a zone transfer download of DNS records. These records (only A/AAAA and CNAME records) will be parsed and used to enrich inventory in the Tetration pipelines (as belonging to the Tenant under which the orchestrator is configured) with a single multi-value label called "orchestrator_system/dns_name", whose value will be the DNS entries that point (directly or indirectly) to that IP address.

4.9.9.1 Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- Supported DNS Servers: BIND9, servers supporting AXFR (RFC 5936), Microsoft Windows Server 2016

4.9.9.2 Configuration fields

- **DNS zones** is an array of strings, each of which represents a DNS zone to be transferred from the DNS server. All dns zones must have a trailing period (".") character.
- **Hosts List** is an array of hostname/ip and port pairs pointing to the DNS server(s) from which to fetch the DNS records. Multiple DNS servers may be configured here for HA purposes only. High Availability behavior across multiple DNS servers specified in the hosts_list is "first healthy server" and will favor the earlier entries in the hosts_list. Zones cannot be split across the DNS servers.

4.9.9.3 Workflow

- First, the user must verify that the DNS server is reachable on that IP/Port from the Tetration cluster.
- For TaaS or in cases where the DNS server is not directly reachable, the user must configure a secure connector tunnel to provide connectivity.
- Configure the correct DNS Zone Transfers ACLs/configuration on the DNS server. Refer to the documentation for the particular DNS server software for more information.

4.9.9.4 Generated labels

orchestrator_system/dns_name -> a multi-value field whose values are all the CNAME and A/AAAA hostnames pointing to that IP.

4.9.9.5 Caveats

- The DNS orchestrator feed is a *metadata feed* - IP addresses learnt from a DNS zone transfer will not create inventory items in Tetration, rather, labels for an existing IP address will be updated with the new DNS metadata. DNS data for unknown IPs is silently discarded. In order to annotate DNS metadata to IPs not learnt from any sensor or via any other orchestrator integrations, IPs must be uploaded via the CMDB bulk upload mechanism to create inventory entries for them. Subnets learnt from CMDB uploads do not create inventory entries.
- Only CNAME and A/AAAA records from the DNS server are processed. CNAME records will be processed to their ultimate IPv4/IPv6 records via the A/AAAA records they point to. Only a single level of deferencing is supported (i.e. chains of CNAME -> CNAME -> A/AAAA or longer are not deferenced) as long as the

CNAME points to an A/AAAA record from that same orchestrator. CNAME deferencing across different DNS orchestrators is not supported.

4.9.9.6 Troubleshooting

- Connectivity Issues

  Tetration will attempt to connect to the provided ip/hostname and port number using a TCP connection originating from one of the Tetration appliance servers or from the cloud in the case of TaaS or from the VM hosting the Tetration Secure Connector VPN tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic.

- DNS AXFR Privilege Issues

  In addition, most DNS servers (BIND9 or Windows DNS or Infoblox) require additional configuration when client IPs attempt DNS zone transfers (AXFR requests as per the DNS protocol opcodes) as these are more resource intensive and privileged as compared to simple DNS requests to resolve individual DNS records. These errors typically show up as AXFR refused with reason code 5 (REFUSED).

  Thus, any manual testing to establish that the DNS server is configured correctly must not depend on succesful hostname lookups but rather they must test AXFR requests specifically (using a tool such as dig).

  Any failure to perform an AXFR zone transfer from the DNS server will be reported in the "authentication_failure_error" field by Tetration appliance.

  Also, note that Tetration will attempt zone transfers from all configured DNS zones and all must succeed in order for the DNS data to be injected into the Tetration label database.

- Inventory Hostname fields are not populated by DNS Field 'hostname' is always learnt from the Tetration sensor. If the inventory was uploaded via CMDB upload and not from the sensor, it may be missing the hostname. All data from the DNS orchestrator workflow only shows up under the "orchestrator_system/dns_name" label and will never populate the hostname field.

4.9.9.7 Behavior of Full/Delta polling for DNS Orchestrators

Default Full Snapshot Interval is 24 hours

Default Delta Snapshot Interval is 60 minutes

These are also the minimum allowed values for these timers.

DNS Records may rarely change. So, for optimal fetching behaviour, at every delta snapshot interval, Tetration will check if the serial numbers of any of the DNS zones has changed from the previous interval. If no zones have changed, no action is needed.

If any zones have changed, we will perform a zone transfer from all configured DNS zones (not just the single zone that has changed).

Every full snapshot interval, Tetration will perform zone transfer downloads from all zones and inject into the label database regardless of whether the zone serial numbers have changed.

4.9.9.8 Unsupported Features

Warning:

- DNAME aliasing and lookups are not supported.

- Incremental Zone Transfers (IXFR) are not supported.

4.9.10 Infoblox

The Infoblox integration allows Tetration to import Infoblox subnets, hosts (*record:host*) and A/AAAA records into Tetration inventory database. The extensible attribute names and values are imported as is and can be used as Tetration labels to define scopes and enforcement policies.

Note: Only Infoblox objects with extensible attributes are considered, ie. those without any extensible attributes attached will be excluded from the import.

Below picture shows an example of generated labels for a host object imported from Infoblox with the extensible attribute *Department*:

```
1. orchestrator_Department = AES789
2. orchestrator_system/cluster_id = [redacted]
3. orchestrator_system/cluster_name = scale13-ib
4. orchestrator_system/machine_id =
   record:host/[redacted]:client8/%20
5. orchestrator_system/machine_name = client8
6. orchestrator_system/orch_type = infoblox
```

Fig. 4.9.10.1: Example Infoblox labels

4.9.10.1 Prerequisites

- Infoblox REST API endpoint supporting WAPI version 2.6, 2.6.1, 2.7, 2.7.1 (recommended)

4.9.10.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

Common Field	Required	Description
Hosts List	Yes	The hosts list denotes one Infoblox grid, ie. more than one grid members with REST API access can be added, and the external orchestrator will switch over to the next one in the list in case of connection errors. If you want to import labels from another Infoblox grid, please create a new external orchestrator for it.

4.9.10.3 Workflow

- First, the user must verify that the Infoblox REST API endpoint is reachable from the Tetration cluster.
- For TaaS or in cases, where the Infoblox server is not directly reachable, the user must configure a *Secure Connector tunnel* to provide connectivity.

- Create an external orchestrator with type *Infoblox*. Depending on the volume of Infoblox data, ie. the number of subnets, hosts and A/AAAA records it can take up to one hour for the first full snapshot is available in Tetration.

4.9.10.4 Orchestrator generated labels

Tetration adds the following system labels to all objects retrieved from Infoblox.

Key	Value
orchestrator_system/orch_type	infoblox
orchestrator_system/cluster_id	<UUID of the external orchestrator in Tetration
orchestrator_system/cluster_name	<Name given to this external orchestrator>
orchestrator_system/machine_id	<Infoblox object reference/identifier>
orchestrator_system/machine_name	<Infoblox host (DNS) name>

4.9.10.5 Generated labels

All Infoblox extensible attributes will be imported as Tetration labels with the prefix *orchestrator_*. For instance, a host with an extensible attribute called *Department* can be addressed in Tetration inventory search as *orchestrator_Department*.

Key	Value
orchestrator_<extensible attribute>	<value(s) of the extensible attribute as retrieved from Infoblox>

4.9.10.6 Caveats

- The maximal number of subnets that can be imported from Infoblox is 50000.
- The maximal number of hosts and A/AAAA records that can be imported from Infoblox is 400000 in total.

4.9.10.7 Troubleshooting

- Connectivity issue Tetration will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Tetration appliance servers or from the cloud in the case of TaaS or from the VM hosting the Tetration Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges to send REST API requests to the Infoblox appliance.
- Not all expected objects are imported Tetration imports only subnets, hosts and A/AAAA records with attached extensible attributes. Note there is a limit number objects that can be imported from Infoblox, see *Caveats*.
- Could not find subnets in inventory It is not possible to use inventory search to find Infoblox subnets as Tetration inventory by design includes only IP addresses, ie. hosts and A/AAAA records.
- Could not find a host or A/AAAA record Tetration imports all extensible attributes as retrieved from Infoblox. Remember to add the prefix *orchestrator_* to the extensible attribute name in eg. inventory search. Note subnets extensible attributes, if not marked as inherited in Infoblox, are not part of hosts and hence not searchable in Tetration.

4.9.11 F5 BIG-IP

The F5 BIG-IP integration allows Tetration to import the *Virtual Servers* from an F5 BIG-IP load balancer appliance and to derive service inventories. A service inventory corresponds to an F5 BIG-IP virtual server, whose service is characterized by the *VIP* (virtual IP address), protocol and port. Once imported into Tetration this service inventory will have labels such as *service_name*, which can be used in inventory search as well as to create Tetration scopes and policies.

A big benefit of this feature is the enforcement of policies in that the *external orchestrator for F5 BIG-IP* translates Tetration policies to security rules assigned to the virtual server and deploys them to the F5 BIG-IP load balancer via its REST API.

4.9.11.1 Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- F5 BIG-IP REST API endpoint version 12.1.1

4.9.11.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

Field	Required	Description
Hosts List	Yes	This specifies the REST API endpoint for F5 BIG-IP load balancer. If High Availability is configured for F5 BIG-IP, please enter also the other member node and the external orchestrator will switch over if it fails to communicate with the current node. If you want to import labels from another F5 BIG-IP load balancer, you need to create a new external orchestrator.
Enable Enforcement	No	Default value is false (unchecked). If checked, this allows Tetration *policy enforcement* to deploy security policy rules to the corresponding F5 BIG-IP load balancer. Note the given credentials must have write access for the F5 BIG-IP REST API.
Route Domain	No	Default value is 0 (zero). The route domain specifies which virtual server are to be considered by the external orchestrator. This is determined by the list of partitions assigned to the given route domain, and only the virtual servers defined in those partitions will be imported in Tetration.

4.9.11.3 Workflow

- First, the user must verify that the F5 BIG-IP REST API endpoint is reachable from Tetration.
- For TaaS or in cases, where the F5 BIG-IP appliance is not directly reachable, the user must configure a *Secure Connector tunnel* to provide connectivity.
- Create an external orchestrator with type *F5 BIG-IP*.
- Depending on the *delta interval* value it might take up to 60 seconds (default delta interval) for the first full snapshot of F5 BIG-IP virtual servers to complete. Thereafter the generated labels can be used to create Tetration scopes and enforcement policies.

4.9.11.4 Orchestrator generated labels

Tetration adds the following system labels for an external orchestrator for *F5 BIG-IP*:

Key	Value
orchestrator_system/orch_type	f5
orchestrator_system/cluster_id	<UUID of the external orchestrator>
orchestrator_system/cluster_name	<Name given to this external orchestrator>
orchestrator_system/workload_type	service
orchestrator_system/namespace	<Partition the virtual server belongs to>
orchestrator_system/service_name	<Name of the F5 BIG-IP virtual server>

4.9.11.5 Generated labels

For each virtual server the external orchestrator will generate the following labels:

Key	Value
orchestrator_annotation/snat_address	<Virtual servers SNAT address>

4.9.11.6 Policy enforcement

This feature enables Tetration to translate logical policies with provider groups that match labelled *F5 BIG-IP* virtual servers into *F5-BIGIP* security policy rules and deploys them to the load balancer appliance using its REST API. As mentioned above any assignment of existing security policy to the respective *F5-BIGP* virtual server will be replaced by a new assignment pointing to Tetration generated security policy. All security policies created by the user will not be manipulated or removed from *F5-BIGIP* policy list.

By default, the field *Enable Enforcement* is not checked, ie. disabled, in the dialog *Create Orchestrator* as shown in the picture below:

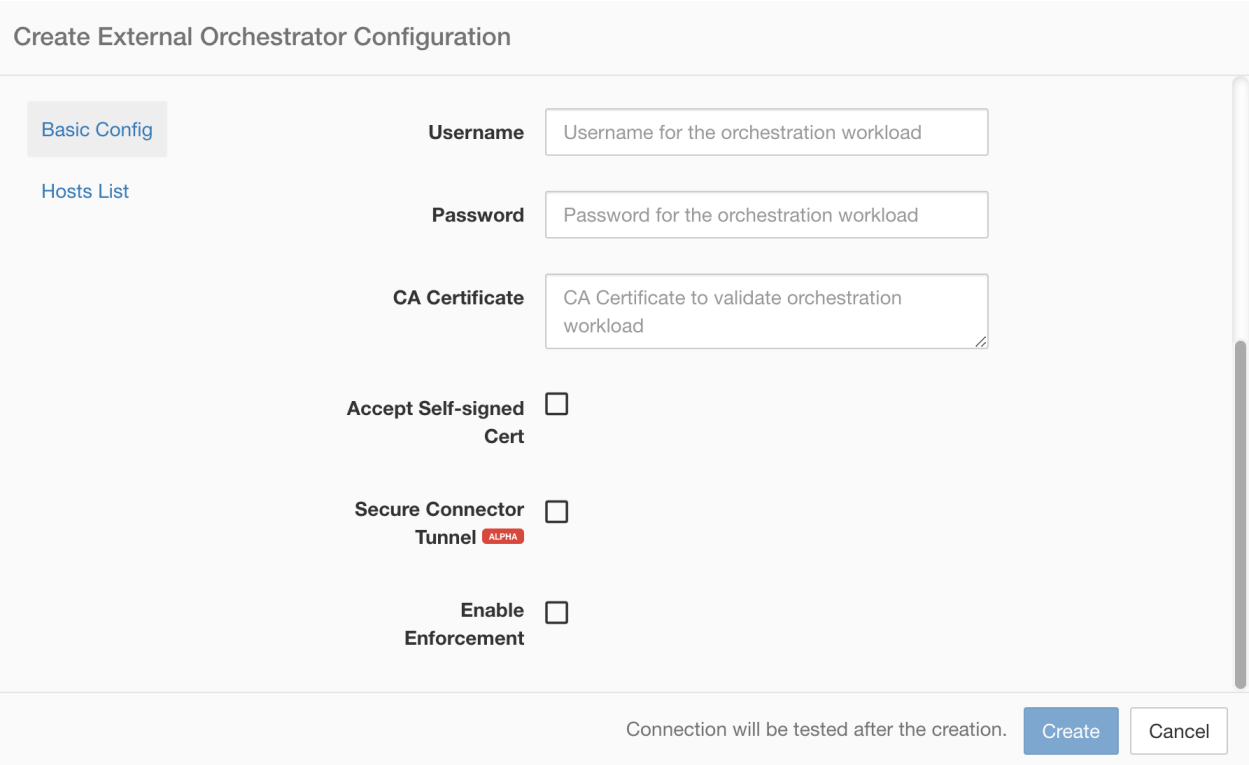


Fig. 4.9.11.6.1: Configuration Option *Enable Enforcement*

Just click on the designated check box to enable enforcement for the orchestrator. This option can be modified any time as needed.

Enable enforcement for the orchestrator, regardless whether it is done by creating or editing the orchestrators configuration, will not deploy the current logical policies to the load balancer appliance immediately. This task is performed as part of the workspace policy enforcement to be triggered by the user as shown in the following picture or due to any updates of inventories. However, disable enforcement for the orchestrator will cause all deployed security policy rules being removed from the *F5-BIGP* load balancer immediately.

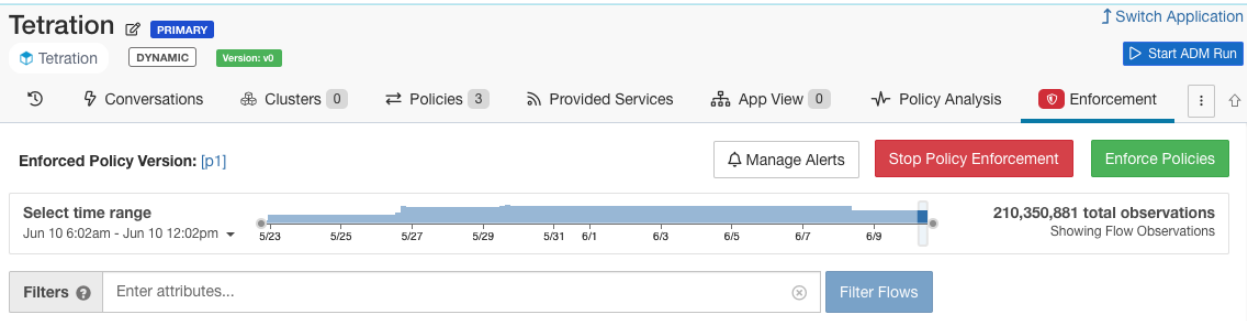


Fig. 4.9.11.6.2: Workspace Policy Enforcement

Note:

- The orchestrator for *F5 BIG-IP* also detects any deviation of security policy rules and replaces it with Tetration policies, ie. any policy changes towards the virtual servers should be done with Tetration only.

- When policy enforcement is stopped or the external orchestrator is deleted, the security policy for virtual servers will become empty as all Tetration policies will be removed from *F5 BIG-IP* load balancer.

The OpenAPI *Policy enforcement status* for external orchestrator can be used to retrieve the status of Tetration policy enforcement to the load balancer appliance associated with the external orchestrator. This helps to verify if the deployment of security policy rules to the *F5-BIGIP* appliance has succeeded or failed.

4.9.11.7 Policy enforcement for F5 ingress controller

Tetration will enforce policies both at the *F5 BIG-IP* load balancer and at the backend pods when the pods are exposed to the external clients using Kubernetes ingress object.

Following are the steps to enforce policy using F5 ingress controller.

1. Create an external orchestrator for *F5 BIG-IP* load balancer as described above.
2. Create an external orchestrator for Kubernetes/OpenShift as described here.

```
➜  ~
➜  ~ k8s get ingress
NAME           HOSTS   ADDRESS         PORTS   AGE
test-ingress   *       192.168.60.100  80      7s
➜  ~
```

3. Create an ingress object in the Kubernetes cluster. A snapshot of the yaml file used to create ingress object is provided in the following picture.

```
➜  ~
➜  ~ k8s get ingress test-ingress -o yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    virtual-server.f5.com/ip: 192.168.60.100
    virtual-server.f5.com/partition: k8scluster
  creationTimestamp: "2019-07-26T18:34:39Z"
  generation: 1
  name: test-ingress
  namespace: default
  resourceVersion: "8310"
  selfLink: /apis/extensions/v1beta1/namespaces/default/ingresses/test-ingress
  uid: 06f8a705-afd4-11e9-97fb-525400d58002
spec:
  backend:
    serviceName: nginx
    servicePort: 80
status:
  loadBalancer:
    ingress:
    - ip: 192.168.60.100
➜  ~
```

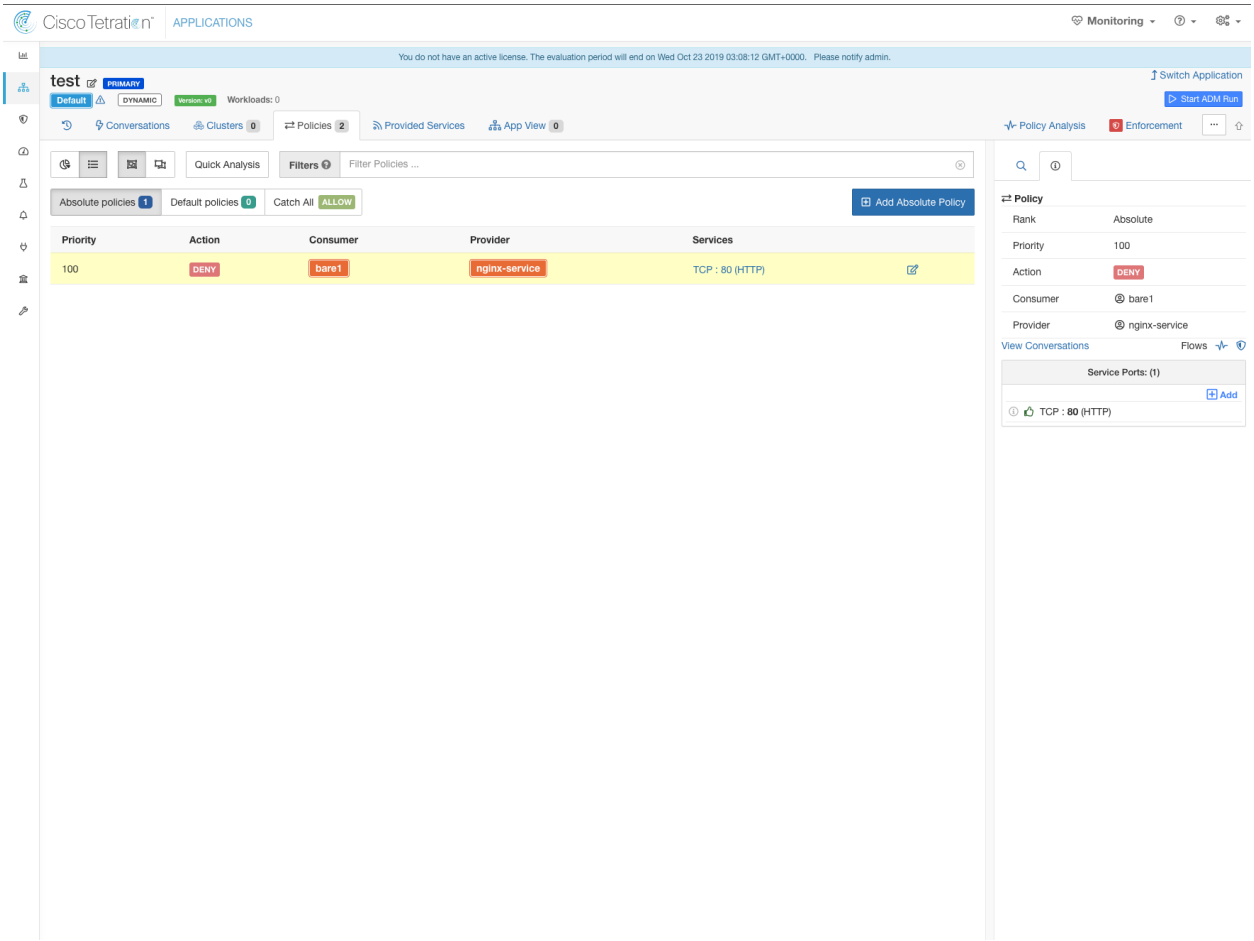
4. Deploy F5 ingress controller pod in the Kubernetes cluster.

```
➜  ~ k8s get deploy -n kube-system
NAME                     DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns                  2         2         2            2           31m
k8s-bigip-ctlr-cluster   1         1         1            1           5m20s
➜  ~
```

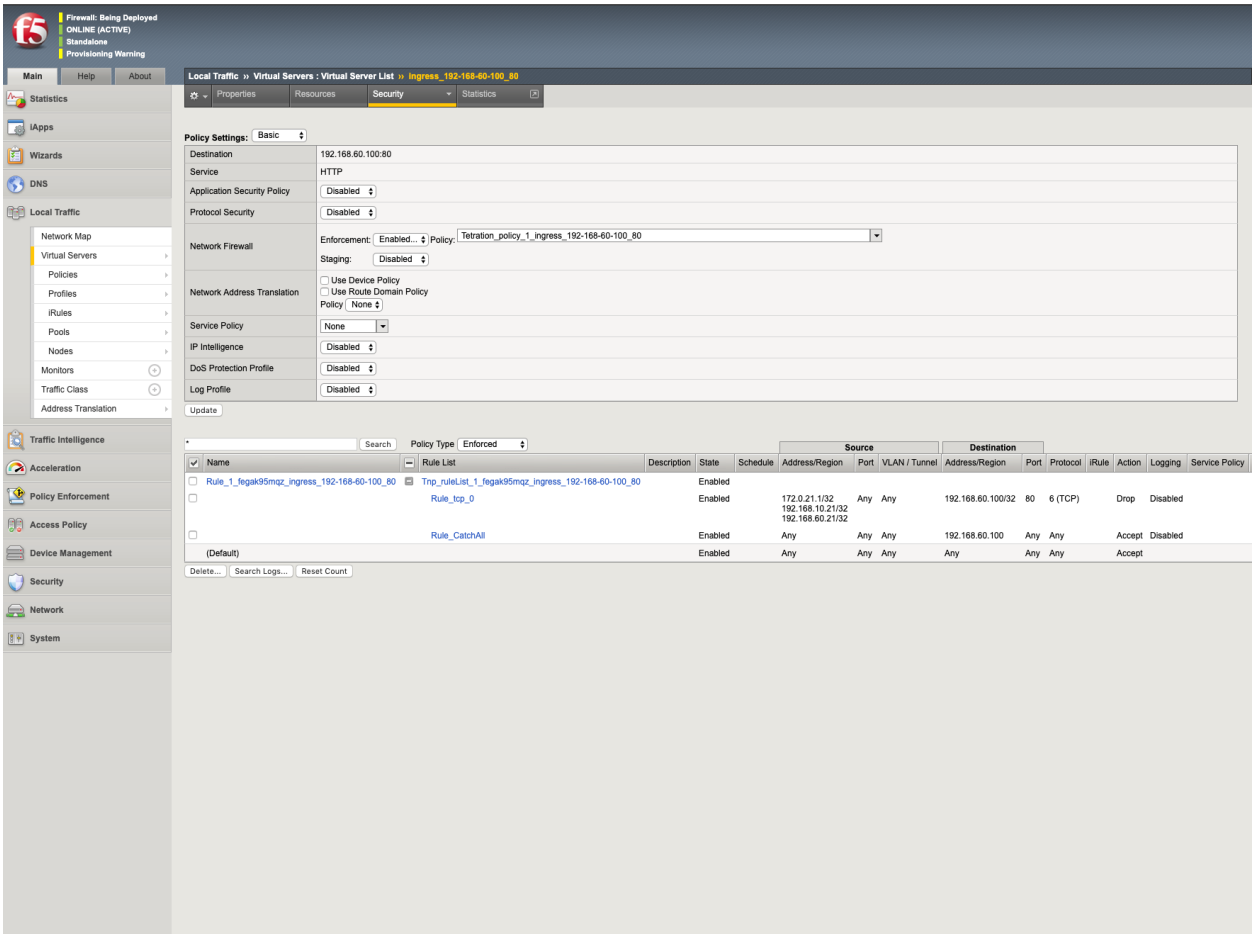
5. Create a backend service which will be accessed by the consumers outside the cluster. In the example provided below we have created a *nginx* service.

```
➜  ~
➜  ~ k8s get deploy
NAME    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx   1         1         1            0           5s
➜  ~
```

6. Create a policy between external consumer and backend service. Enforce the policy using *Policy Enforcement* tab.



7. Check the policies on *F5 BIG-IP* load balancer and backend pods. In case of F5 load balancer Tetration will apply the appropriate allow/drop rule where the source will be the consumer specified in step 6 and the destination will be VIP [VIP for the ingress virtual service for F5]. In case of backend pods, Tetration will apply the appropriate allow/drop rule where the source will the SNIP [in case SNAT pool is enabled] or F5 IP [auto map enabled] and destination will be backend pod IP.



4.9.11.8 Caveats

- During upgrade Tetration from a version prior to 3.3.1 to version 3.3.1 or later all existing external orchestrator for *F5 BIG-IP* will have the default *route domain* (zero) assigned. In case the route domain in F5 is set to a different value please update the *route domain* field in external orchestrator configuration page accordingly!
- Starting with Tetration 3.4 the enforcement virtual appliance for *F5 BIG-IP* is **deprecated**. Please destroy the created VM as it is no longer supported.
- Enable enforcement is disabled for all *F5 BIG-IP* orchestrators created with Tetration version prior 3.4. See _f5_policy_enf for more details.
- During deployment phase of *F5 BIG-IP* HA mode, please enable the *configuration sync* option. This ensures the external orchestrator can fetch the latest list of virtual servers from the currently connected host.
- In case of *F5 BIG-IP* HA deployment mode, if *Auto-Map* is configured instead of SNAT pool for Address translation, please ensure that the *Primary BIG-IP* is configured with the floating *Self IP* address.
- Only VIP specified as a single address is supported, ie. VIP given as a subnet is not supported.

4.9.11.9 Troubleshooting

- Connectivity issue Tetration will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Tetration appliance servers or from the cloud in the case of *TaaS* or from the VM hosting the Tetration Secure Connector tunnel service. In order to correctly establish this

connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges with read and write access to send REST API requests to the *F5 BIG-IP* appliance.

- Security rules not found In case no security rules for a defined virtual server are found, after policy enforcement was performed, please make sure the corresponding virtual server is enabled, ie. its availability/status must be *available/enabled*.

4.9.12 Citrix Netscaler

The Citrix Netscaler integration allows Tetration to import the *Load Balancing Virtual Servers* from a Netscaler load balancer appliance and to derive service inventories. A service inventory corresponds to a Netscaler service provided by a virtual server and has labels such as *service_name*, which can be used in inventory search and to create Tetration scopes and policies.

A big benefit of this feature is the enforcement of policies in that the *external orchestrator for Citrix Netscaler* translates Tetration policies to Netscaler ACLs rules and deploys them to the Netscaler load balancer via its REST API.

4.9.12.1 Prerequisites

- Secure Connector Tunnel, if needed for connectivity
- Netscaler REST API endpoint version 12.0.57.19

4.9.12.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

Common Field	Required	Description
Hosts List	Yes	This specifies the REST API endpoint for Citrix Netscaler load balancer. If High Availability is configured, please enter also the other member node and the external orchestrator will switch over if it fails to communicate with the current node. If you want to import labels from another Citrix Netscaler load balancer, you need to create a new external orchestrator.
Enable Enforcement	No	Default value is false (unchecked). If checked, this allows Tetration *policy enforcement* to deploy ACL rules to the corresponding Citrix Netscaler load balancer. Note the given credentials must have write access for the Citrix Netscaler REST API.

4.9.12.3 Workflow

- First, the user must verify that the Netscaler REST API endpoint is reachable from the Tetration cluster.
- For TaaS or in cases, where the Netscaler appliance is not directly reachable, the user must configure a *Secure Connector tunnel* to provide connectivity.
- Create an external orchestrator with type *Citrix Netscaler*.
- Depending on the *delta interval* value it might take up to 60 seconds (default delta interval) for the first full snapshot of Netscaler virtual servers to complete. Thereafter the generated labels can be used to create Tetration scopes and enforcement policies.
- Enforce policies from Tetration to deploy Netscaler ACL rules.

4.9.12.4 Orchestrator generated labels

Tetration adds the following system labels for an external orchestrator for *Citrix Netscaler*:

Key	Value
orchestrator_system/orch_type	nsbalancer
orchestrator_system/cluster_id	<UUID of the external orchestrator>
orchestrator_system/cluster_name	<Name given to this external orchestrator>
orchestrator_system/workload_type	service
orchestrator_system/service_name	<Name of the load balancing virtual server>

4.9.12.5 Generated labels

For each load balancing virtual server the external orchestrator will generate the following labels:

Key	Value
orchestrator_annotation/snat_address	<Virtual servers SNAT address>

4.9.12.6 Policy enforcement

This feature enables Tetration to translate logical policies with provider groups that match labelled *Citrix Netscaler* virtual servers into *Citrix Netscaler* ACL rules and deploys them to the load balancer appliance using its REST API. As mentioned above all existing ACL rules will be replaced by Tetration generated policy rules.

By default, the field *Enable Enforcement* is not checked, ie. disabled, in the dialog *Create Orchestrator* as shown in the picture below:

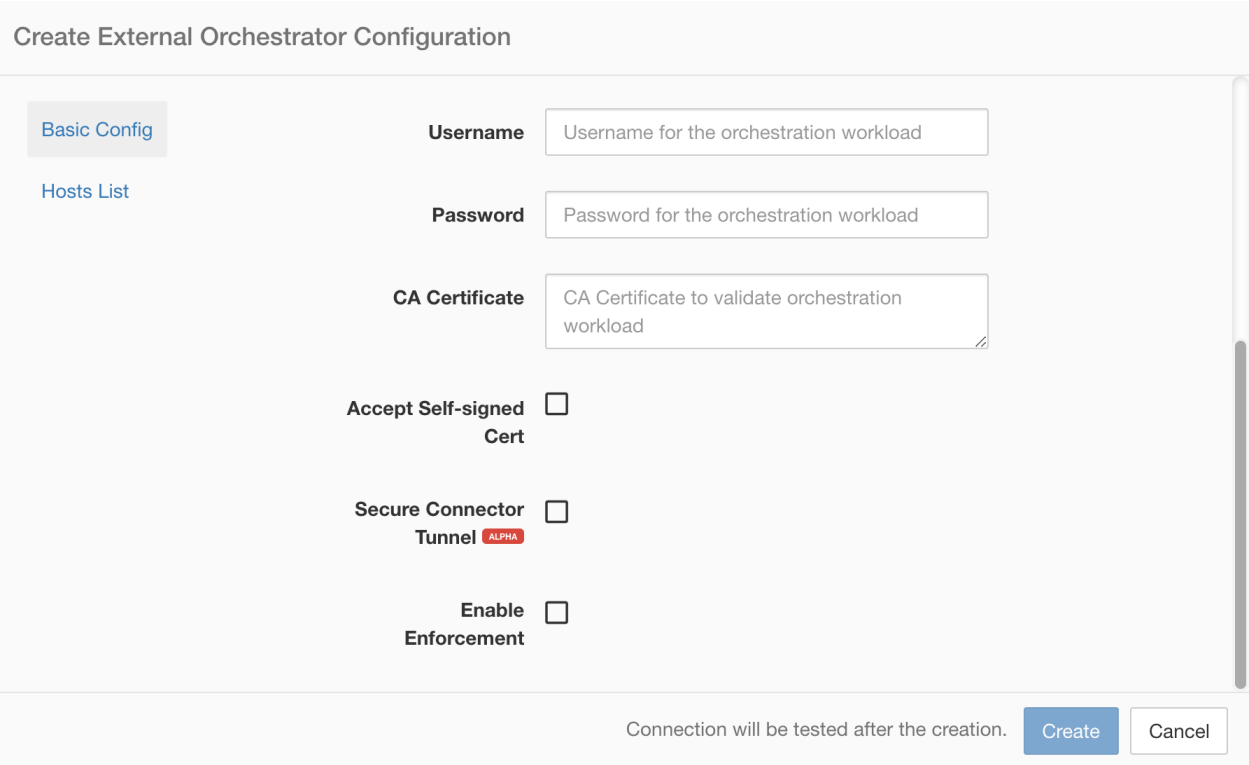


Fig. 4.9.12.6.1: Configuration Option *Enable Enforcement*

Just click on the designated check box to enable enforcement for the orchestrator. This option can be modified any time as needed.

Enable enforcement for the orchestrator, regardless whether it is done by creating or editing the orchestrators configuration, will not deploy the current logical policies to the load balancer appliance immediately. This task is performed as part of the workspace policy enforcement to be triggered by the user as shown in the following picture or due to any updates of inventories. However, disable enforcement for the orchestrator will cause all deployed ACL rules being removed from the *Citrix Netscaler* load balancer immediately.

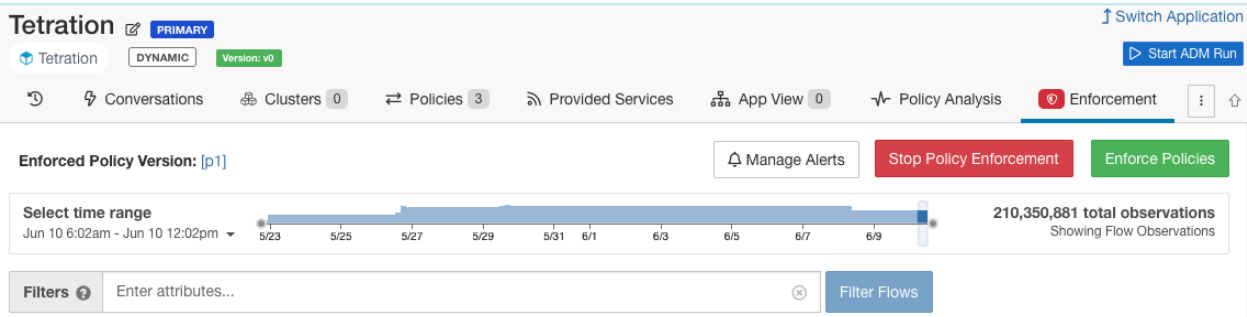


Fig. 4.9.12.6.2: Workspace Policy Enforcement

Note:

- The orchestrator for *Citrix Netscaler* also detects any deviation of ACL rules and replaces it with Tetration policies, ie. any policy changes towards the load balancing virtual servers should be done with Tetration only.

- When policy enforcement is stopped or the external orchestrator is deleted, the ACLs will become empty as all Tetration policies will be removed from *Citrix Netscaler* load balancer.

The OpenAPI *Policy enforcement status* for external orchestrator can be used to retrieve the status of Tetration policy enforcement to the load balancer appliance associated with the external orchestrator. This helps to verify if the deployment of ACL rules to the *Citrix Netscaler* appliance has succeeded or failed.

4.9.12.7 Caveats

- Starting with Tetration 3.4 the enforcement virtual appliance for *Citrix Netscaler* is **deprecated**. Please destroy the created VM as it is no longer supported.
- Enable enforcement is disabled for all *Citrix Netscaler* orchestrators created with Tetration version prior 3.4. See _ns_policy_enf for more details.
- If enforcement is enabled, the Tetration policies will always be deployed to the global list of ACLs, ie. partition *default*.
- Only VIP specified as a single address is supported, ie. VIP given as an address pattern is not supported.
- Visibility for the detected services (*Citrix Netscaler* virtual servers) is not supported.

4.9.12.8 Troubleshooting

- Connectivity issue Tetration will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Tetration appliance servers or from the cloud in the case of *TaaS* or from the VM hosting the Tetration Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges with read and write access to send REST API requests to the *Citrix Netscaler* appliance.
- ACL rules not found In case no ACL rules are found, after policy enforcement was performed, please make sure the corresponding virtual server is enabled, ie. its status must be *up*.

4.9.13 TAXII

The TAXII (Trusted Automated Exchange of Intelligence Information) Integration allows Tetration to ingest threat intelligence data feeds from security vendors to annotate network flows and process hashes with STIX (Structured Threat Information Expression) indicators such as malicious IPs, malicious hashes.

When an external orchestrator configuration is added for type "taxii", the Tetration appliance will attempt to connect to the TAXII server(s) and poll STIX data feed collections. The STIX data feeds (only IPs and binary hashes indicators) will be parsed and used to annotate network flows and process hashes in the Tetration pipelines (as belonging to the Tenant under which the orchestrator is configured).

Network flows with either provider or consumer addresses matched imported malicious IPs will be tagged with multi-value label "orchestrator_malicious_ip_by_<vendor name>" where <vendor name> is the user orchestrator configuration input TAXII vendor, and the label value is "Yes".

The ingested STIX binary hash indicators will be used to annotate workload process hashes, which will be displayed (if matched) in the Security Dashboard / Process Hash Score Details and in the Workload Profile / File Hashes.

4.9.13.1 Prerequisites

- Secure Connector Tunnel, if needed for connectivity

- Supported TAXII Servers: 1.0
- Supported TAXII feeds with STIX verion: 1.x

4.9.13.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

Common Field	Required	Description
Name	Yes	User specified name of the orchestrator.
Description	Yes	User specified description of the orchestrator.
Vendor	Yes	The vendor provides intelligence data feeds.
Full Snapshot Interval	Yes	The interval (in seconds) to perform a full snapshot of the TAXII feed. (Default: 1 day)
Poll Url	Yes	The polling full URL path to poll data.
Collection	Yes	The TAXII feed collection name to be polled.
Poll Days	Yes	The number of earlier days threat data to poll from TAXII feed.
Username		Username for authentication.
Password		Password for authentication.
Certificate		Client certificate used for authentication.
Key		Key corresponding to client certificate.
CA Certificate		CA Certificate to validate orchestration endpoint.
Accept Self-Signed Cert		Checkbox to disable strictSSL checking of the TAXII API server certificate
Secureconnector Tunnel		Tunnel connections to this orchestrator's hosts through the Secure Connector tunnel.
Hosts List	Yes	The hostname/ip and port pairs pointing to the TAXII server(s).

4.9.13.3 Workflow

- First, the user must verify that the TAXII server is reachable on that IP/Port from the Tetration cluster.
- Configure the correct TAXII server with the poll path and TAXII feed name.

4.9.13.4 Generated labels

Key	Value
orchestrator_system/orch_type	*TAXII*
orchestrator_system/cluster_id	UUID of the cluster's configuration in Tetration.
orchestrator_system/cluster_name	Name given to this cluster's configuration>.
orchestrator_malicious_ip_by_ <vendor>	*Yes* if the flow provider/consumer address matches the imported TAXII malicious IPs data.

4.9.13.5 Caveats

- The TAXII integration is supported only on on-premise Tetration.
- Only IPs and hashes indicators from TAXII feeds are ingested.

- Maximum number of ingested IPs is 100K (most recently updated) per TAXII feed.
- Maximum number of ingested hashes is 500K (most recently updated) for all TAXII feeds.
- Only TAXII feeds with STIX version 1.x are supported.

4.9.13.6 Troubleshooting

- Connectivity Issues

  The Tetration will attempt to connect to the provided poll URL path from one of the Tetration appliance servers or from the VM hosting the Tetration Secure Connector VPN tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic.

4.9.13.7 Behavior of Full polling for TAXII Orchestrators

Default Full Snapshot Interval is 24 hours

Every full snapshot interval, Tetration will perform pulling TAXII feeds of IPs and hashes up to the above limits into the label database.

4.9.14 Cisco FMC

The FMC (Firepower Management Center) integration allows Tetration to deploy policies to all FTDs (Firepower Threat Defense) registered to the given FMC using the FMC REST API.

Note: The FMC external orchestrator does not generate any user annotations.

When policy enforcement is performed, the FMC external orchestrator will convert all Tetration policies into FMC prefilter policy rules and deploy them to the FTDs. There are three groups of prefilter rules generated by this task in the order listed below:

- Golden rules with the prefix *Tetrul_golden_* to allow network traffic between the Tetration sensor/enforcer agents installed behind the FTDs, if any, and Tetration cluster.
- Policy rules with the prefix *Tetrul_* as defined in application workspaces.
- Scopes catch all rules with the prefix *Tetrul_ca_* in the same order as defined in *policy order*. Note these catch all rules are created only if the corresponding application workspace is enforced.

The following picture shows an example of prefilter rules generated by Tetration:

#	Name	Rule Type	Source Interface Obje...	Destination Interface Obje...	Source Networks	Destination Networks	Source Port	Destination Port	VLAN Tag	Action	Tunnel Zone
1	Tetrul_golden_1	Prefilter	any	any	172.21.130.181	any	TCP (6):443	any	any	Fastpath	na
2	Tetrul_golden_2	Prefilter	any	any	any	172.21.130.181	any	TCP (6):443	any	Fastpath	na
3	Tetrul_golden_3	Prefilter	any	any	172.21.130.182 172.21.130.183	any	TCP (6):5640 TCP (6):5660	any	any	Fastpath	na
4	Tetrul_golden_4	Prefilter	any	any	any	172.21.130.182 172.21.130.183	any	TCP (6):5640 TCP (6):5660	any	Fastpath	na
5	Tetrul_5	Prefilter	any	any	192.170.60.10	192.170.50.10	any	TCP (6):22	any	Fastpath	na
6	Tetrul_6	Prefilter	any	any	192.170.60.0-1!	192.170.50.0-1!	any	TCP (6):8081	any	Fastpath	na
7	Tetrul_ca_7	Prefilter	any	any	192.170.50.0-1! 192.170.60.0-1!	any	any	any	any	Block	na
8	Tetrul_ca_8	Prefilter	any	any	any	192.170.50.0-1! 192.170.60.0-1!	any	any	any	Block	na
9	my rule two	Prefilter	any	any	any	10.10.10.10	any	any	any	Analyze	na
10	my rule one	Prefilter	any	any	any	8.8.8.8	any	any	any	Analyze	na

Non-tunneled traffic is allowed Default Action: Tunnel Traffic Analyze all tunnel traffic

Fig. 4.9.14.1: Example FMC prefilter rules generated by Tetration

The rules numbered 1-4, 5-6 and 7-8 are the golden rules, Tetration policy rules and scope catch all rules generated by the FMC external orchestrator. The remaining prefilter rules 9-10 are preserved as they were created by the users. For more details please see *Configuration fields* below.

Note: The initial policy deployment of a large set of policies (10000 or more) may last over several minutes. On an average the policy deployment to FMC should take about 1.5-2 minutes to finish. The actual response time depends on the resource configuration of FMC and FTDs as well as the number of policy rules to be installed on the FTDs.

4.9.14.1 Prerequisites

- All platforms available for FMC and FTD version 6.6 and 6.7 are supported.
- The FTDs must be registered to FMC as Tetration uses the unified FMC REST API to program the policy rules on FTDs. That means standalone FTDs are not supported by the FMC external orchestrator.
- Supported FTD modes: routed and transparent.
- It's strongly recommended to create a user prefilter policy and assign it to all FTDs that should receive Tetration policies. The FMC default prefilter policy is read-only, which prevents Tetration to perform any updates in it.
- The given FMC user credentials must have *Administrator* role in domains that are assigned to the FTDs, access and prefilter policies so that the FMC external orchestrator is able to update these entities during the policy deployment.

4.9.14.2 Configuration fields

Beside the common configuration fields as described in *Create External Orchestrator* the following fields can be configured:

Common Field	Required	Description
Hosts List	Yes	This specifies the REST API endpoint for FMC. If high availability is enabled, the second/standby FMC endpoint should be added here. The FMC external orchestrator will switch over to the new active FMC when it detects a role switching performed on FMC.
Enable Enforcement	No	This option is pre-selected in the UI. If selected, this allows Tetration *policy enforcement* to deploy prefilter rules to the FTDs assigned to the given FMC. If this option is changed to unchecked, any deployed Tetration policies will be removed from the FTD.
Enforcement Mode	No	Default value is "merge". Other accepted value: "override". In "merge" mode Tetration policy rules will always be inserted before any existing prefilter rules created by the user. The "override" mode forces all existing prefilter rules, if any, to be replaced by Tetration policy rules.

4.9.14.3 Workflow

- First, the user must verify that the FMC REST API endpoint is reachable from the Tetration cluster.
- For TaaS or in cases, where the FMC appliance is not directly reachable, the user must configure a *Secure Connector tunnel* to provide connectivity.
- Create an external orchestrator with type *FMC* with the option *Enable Enforcement* selected.

4.9.14.4 Orchestrator generated labels

None

4.9.14.5 Generated labels

None

4.9.14.6 Caveats

- FTDs must be assigned to an FMC in order to retrieve Tetration policies as the policy deployment uses the unified FMC REST API.
- Tetration policy rules are deployed to FMC prefilter policies, which may interfere with access policy rules managed by the users. For instance, the deployed prefilter rules may block packets that should be allowed by the access policy.
- Tetration prefilter rules use the allow action *FASTPATH* so that further packet inspection is not possible. This approach makes sure that the allowed traffic as defined in Tetration policies is not blocked by any access policy rule or default action.
- All Tetration policies will be deployed to all FTDs assigned to the given FMC.
- If the configuration option *Enforcement Mode* is *merge*, avoid naming any user rules with the prefix *Tetrul_* as they will be wiped out during the reconciliation of Tetration policy.
- A policy deployment that requires deleting a large set (a few of thousands) of policies can exceed over several minutes for completion.

- If high availability is enabled, it can take up to 4 minutes for FMC policy enforcer to switch to the new active node and recover with policy enforcement. During this time any policy enforcement to the non-active FMC will fail.

4.9.14.7 Troubleshooting

- Connectivity issue Tetration will attempt to connect to the provided IP/hostname and port number using an HTTPS connection originating from one of the Tetration appliance servers or from the cloud in the case of TaaS or from the VM hosting the Tetration Secure Connector tunnel service. In order to correctly establish this connection, firewalls must be configured to permit this traffic. Also, make sure the given credentials are correct and have privileges to send REST API requests to the FMC appliance.
- No Tetration rules are deployed to FTDs First verify that Tetration is able to communicate with FMC by checking the external orchestrator's status field. Also make sure that the field *Enable Enforcement* is selected. Another reason could be that the FTD still has the default prefilter policy assigned. Also make sure that the given user credentials have *Administrator* privileges for the domains, which the FTD, access and prefilter policy belong to.

4.10 Service Profile

Tetration provides visibility of all Kubernetes services and other Load Balancers ingested through *External Orchestrators*. Service profile page shows the details for a given service.

Note: Service profile page is linked from various places. One of the ways to see a service profile is to perform a search for service as described in search

From the results of search, click on a Service Name under the Services tab to go to its profile. The following labels are available for the service

4.10.1 Header

Header consists of

- **Orchestrator Name:** Name of the external orchestrator which reported this service.
- **Orchestrator Type:** Type of the external orchestrator.
- **Namespace:** Namespace of the service.
- **Service Type:** Type of the service. Possible values include ClusterIP, NodePort and LoadBalancer.

4.10.2 IP and Ports

This table lists all the possible IP and port combinations through which this service is accessible. For services of type NodePort, this table shows both ClusterIP:Port and NodeIp:NodePort association.

4.10.3 User Labels

The list of user uploaded and orchestrator system generated labels for this service. See *External Orchestrators* for more details.

4.10.4 Scopes

List of scopes that the service belongs to.

4.11 Pod Profile

Tetration provides visibility of all Kubernetes pods ingested through Kubernetes *External Orchestrators*. Pod profile page shows the details for a given pod.

Note: Pod profile page is linked from various places. One of the ways to see a pod profile is to perform a search for pod as described in search

From the results of search, click on a Pod Name under the Pods tab to go to its profile. The following labels are available for the pod

4.11.1 Header

Header consists of

- **Orchestrator Name:** Name of the external orchestrator which reported this pod.
- **Orchestrator Type:** Type of the external orchestrator.
- **Namespace:** Namespace of the pod.
- **IP Address:** Pod's IP Address.

4.11.2 User Labels

The list of user uploaded and orchestrator system generated labels for this pod. See *External Orchestrators* for more details.

4.11.3 Scopes

List of scopes that the pod belongs to.

CHAPTER

FIVE

FLOWS

The **Flows** option in the top-level menu takes you to the Flow Search page. This page provides the means for quickly filtering and drilling down into the flows corpus. The basic unit is a "Flow Observation" which is a per-minute aggregation of each unique flow. The two sides of the flow are called "Consumer" and "Provider", the Consumer is the side that initiated the flow, and the Provider is responding to the Consumer (e.g. "Client" and "Server" respectively). Each observation tracks the number of packets, bytes, and other metrics in each direction for that flow for that minute interval. In addition to quickly filtering, the flows can be explored visually with the "Explore Observations" button. The resulting list of flows observations can be clicked to view details of that flow, including latency, packets, and bytes over the lifetime of that flow.

> **Warning:** For hosts instrumented with Deep Visibility Agents or Enforcement Agents, Tetration is able to correlate flow data against the process that provides or consumes the flow. As a result, full command line arguments, which may include **sensitive information such as database or API credentials**, used to launch the process are available for analysis and display.

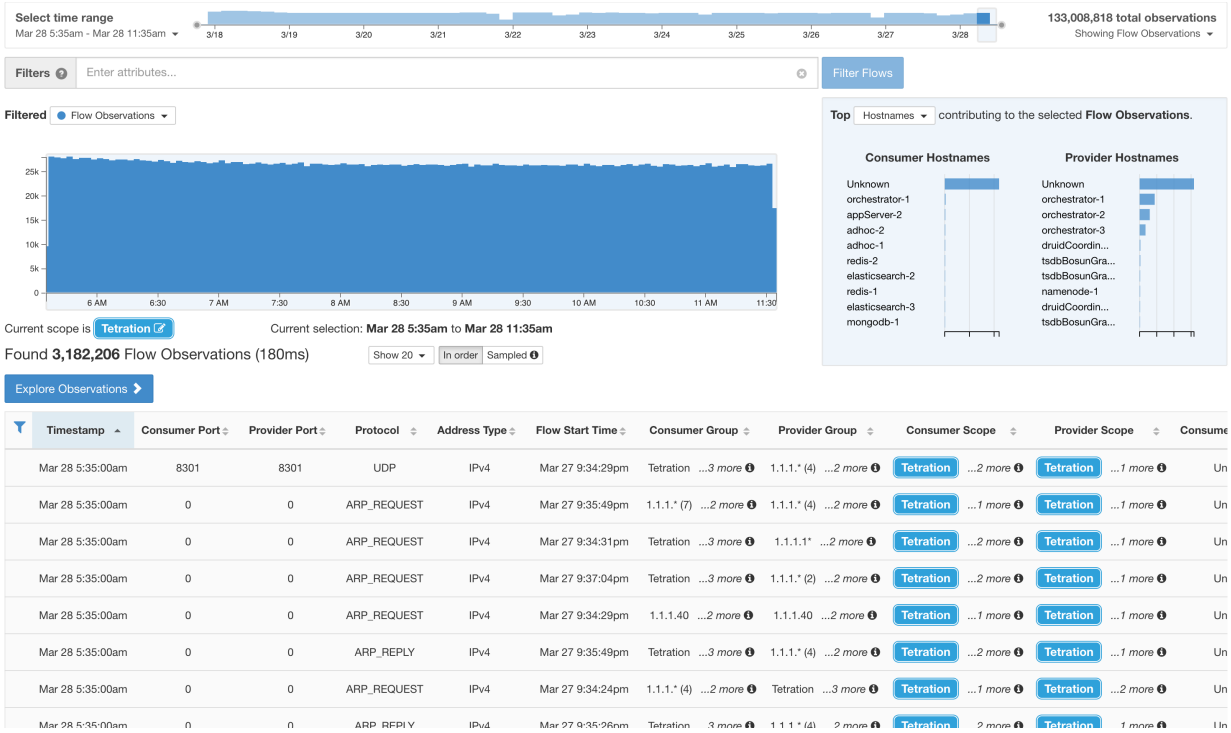


Fig. 5.1: Flows Overview

5.1 Corpus Selector

Fig. 5.1.1: Corpus Selector

This is the unfiltered summary timeseries data for the current **Scope** for the entire corpus. The purpose of this component is to allow you to know what date range is being viewed, and easily change that date range by dragging within the component. The data in the chart is there in case it's useful for deciding which time range to select. You can select different metrics to be shown, by default the count of **flow observations** is shown.

The Corpus Selector can currently support selecting up to *approximately* **2 billion flow observations**.

5.2 Columns and Filters

Fig. 5.2.1: Filter input

This is where you define filters to narrow-down the search results. All of the possible dimensions can be found by clicking on the **(?)** icon next to the word **Filters**. For any User Labels data, those columns will also be available for the appropriate intervals. This input also supports **and**, **or**, **not**, and **parenthesis** keywords, use these to express more complex filters. For example, a direction-agnostic filter between IP *1.1.1.1* and *2.2.2.2* can be written:

> *Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1*

And to additionally filter on Protocol = TCP:

> *(Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1) and Protocol = TCP*

The filter input also supports "," and "-" for Port, Consumer Address and Provider Address, by translating "-" into range queries. The following are examples of a valid filter:

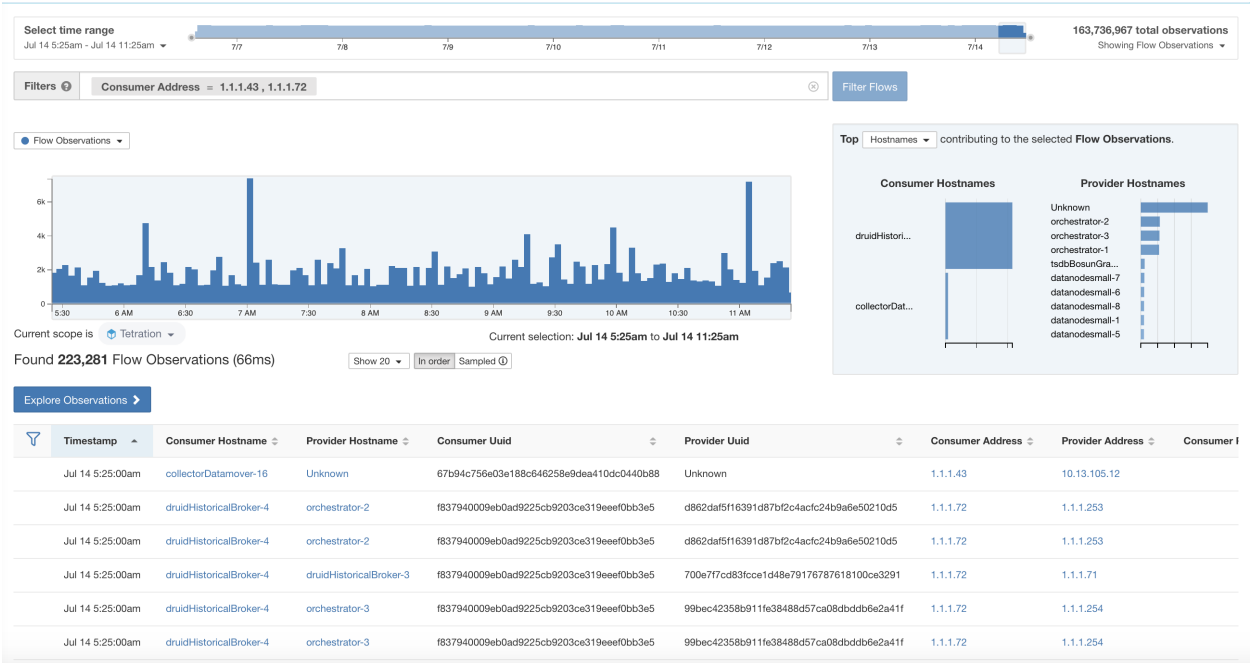


Fig. 5.2.2: Example: Filter input supports “,” for Consumer Address

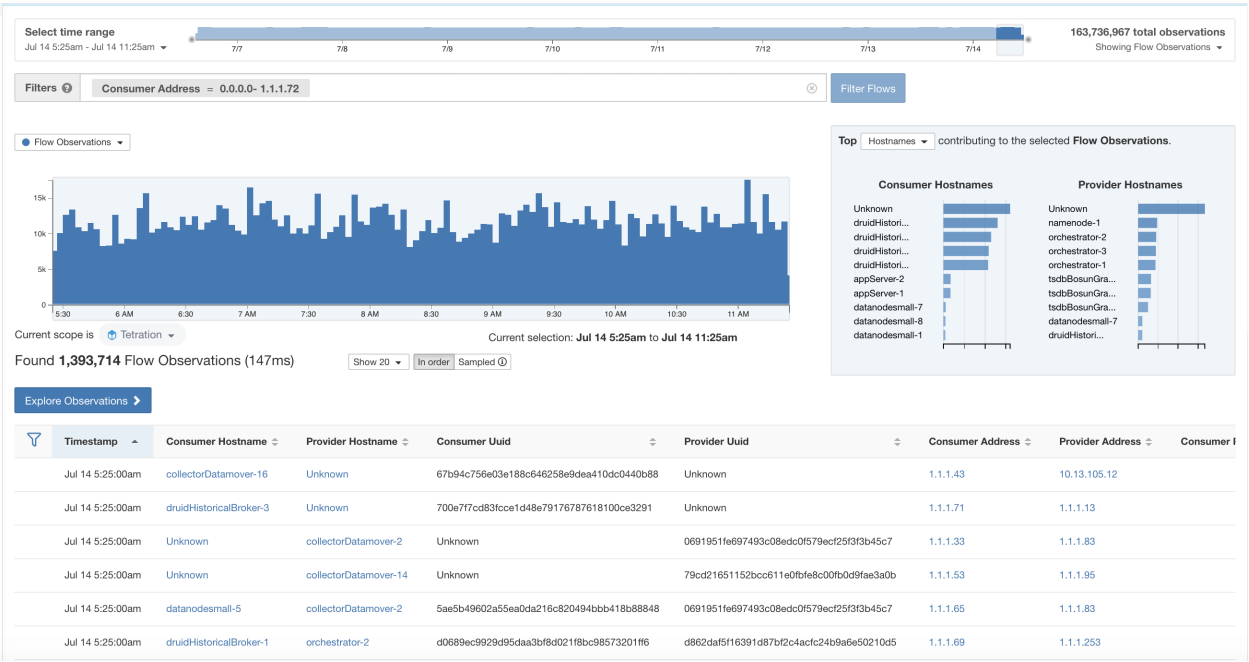


Fig. 5.2.3: Example: Filter input supports range query for Consumer Address

Available Columns and Filters:

Columns (names exposed in API)	Description
Consumer Address (*src_address*)	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24). Matches flow observations whose consumer address overlaps with provided IP Address or subnet.
Provider Address (*dst_address*)	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24) Matches flow observations whose provider address overlaps with provided ip address or subnet.
Consumer Hostname (*src_hostname*)	Matches flows whose consumer hostname overlaps with provided hostname.
Provider Hostname (*dst_hostname*)	Matches flows whose provider hostname overlaps with provided hostname.
Consumer Enforcement Group (*src_enforcement_epg_name*)	The Consumer Enforcement Group is the name of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the consumer.
Provider Enforcement Group (*dst_enforcement_epg_name*)	The Provider Enforcement Group is the name of the filter (Scope, Inventory Filter or Cluster) in the enforced policies that matches the provider.
Consumer Analysis Group	The Consumer Analysis Group is the name of the filter (Scope, Inventory Filter or Cluster) in the analyzed policies that matches the consumer.
Provider Analysis Group	The Provider Analysis Group is the name of the filter (Scope, Inventory Filter or Cluster) in the analyzed policies that matches the provider.
Consumer Scope (*src_scope_name*)	Matches flows whose consumer belongs to the specified Scope.
Provider Scope (*dst_scope_name*)	Matches flows whose provider belongs to the specified Scope.
Consumer Port (*src_port*)	Matches flows whose Consumer port overlaps with provided port.
Provider Port (*dst_port*)	Matches flows whose Provider port overlaps with provided port.
Consumer Country (*src_country*)	Matches flows whose Consumer country overlaps with provided country.
Provider Country (*dst_country*)	Matches flows whose Provider country overlaps with provided country.
Consumer Subdivision (*src_subdivision*)	Matches flows whose Consumer subdivision overlaps with provided subdivision (state).
Provider Subdivision (*dst_subdivision*)	Matches flows whose Provider subdivision overlaps with provided subdivision (state).
Consumer Autonomous System Organization (*src_autonomous_system_organization*)	Matches flows whose Consumer autonomous system organization overlaps with provided autonomous system organization (ASO).
Provider Autonomous System Organization (*dst_autonomous_system_organization*)	Matches flows whose Provider autonomous system organization overlaps with provided autonomous system organization (ASO).
Protocol (*proto*)	Filter flow observations by Protocol type (TCP, UDP, ICMP).
Address Type (*key_type*)	Filter flow observations by Address type (IPv4, IPv6, DHCPv4).
Fwd TCP Flags	Filter flow observations by flags (SYN, ACK, ECHO).
Rev TCP Flags	Filter flow observations by flags (SYN, ACK, ECHO).
Fwd Process UID (*fwd_process_owner*)	Filter flow observations by process owner UID (root, admin, yarn, mapred).
Rev Process UID (*rev_process_owner*)	Filter flow observations by process owner UID (root, admin, yarn, mapred).
Fwd Process (*fwd_process_string*)	Filter flow observations by process (java, hadoop, nginx). See *Process String Visibility Warning*
Rev Process (*rev_process_string*)	Filter flow observations by process (java, hadoop, nginx). See *Process String Visibility Warning*
Consumer In Collection Rules?	Match only internal Consumers.
Provider In Collection Rules?	Match only internal Providers.

Continued on next page

Table 5.2.1 – continued from previous page

Columns (names exposed in API)	Description
SRTT Available	Matches flows which have SRTT measurements available using the values 'true' or 'false'. (This is equivalent to SRTT > 0).
Bytes	Filter flow observations by Byte traffic bucket. Matches flows which Byte traffic bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).
Packets	Filter flow observations by Packet traffic bucket. Matches flows which Packet traffic bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).
Flow Duration (µs)	Filter flow observations by Flow Duration bucket. Matches flows which Flow Duration bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024).
Data Duration (µs)	Filter flow observations by Data Duration bucket. Matches flows which Data Duration bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024).
SRTT (µs) (*srtt_dim_usec*)	Filter flow observations by SRTT bucket. Matches flows which SRTT bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).
Fwd Packet Retransmissions (*fwd_tcp_pkts_retransmitted*)	Filter flow observations by Packet Retransmissions bucket. Matches flows which Packet Retransmissions bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).
Rev Packet Retransmissions (*rev_tcp_pkts_retransmitted*)	Filter flow observations by Packet Retransmissions bucket. Matches flows which Packet Retransmissions bucket values are =, <, > (bucketed by powers of 2 (0, 2, 64, 1024)).
TCP Handshake (*fwd_tcp_handshake_usec*)	Filter flow observations by TCP Handshake bucket. Matches flows which TCP Handshake bucket values are =, <, > e.g. '[10µs - 25µs)'. See *Visibility Warning*
TCP Performance	Matches flows which have one of the following TCP Performance events: 'App Limited', 'Consumer App Limited', 'Provider App Limited', 'Network Limited'. See *Visibility Warning*
Fwd TCP Bottleneck (*fwd_tcp_bottleneck*)	Matches flows which have one of the following TCP Bottleneck events: 'App', 'Network', 'Both', 'None' See *Visibility Warning*
Rev TCP Bottleneck (*rev_tcp_bottleneck*)	Matches flows which have one of the following TCP Bottleneck events: 'App', 'Network', 'Both', 'None' See *Visibility Warning*
Fwd Congestion Window Reduced	Matches flows which have Congestion Window Reduced using the values 'true' or 'false'. See *Visibility Warning*
Rev Congestion Window Reduced	Matches flows which have Congestion Window Reduced using the values 'true or 'false'. See *Visibility Warning*
Fwd MSS Changed	Matches flows which have Maximum Segment Size Changed using the values 'true' or 'false'. See *Visibility Warning*
Rev MSS Changed	Matches flows which have Maximum Segment Size Changed using the values 'true' or 'false'. See *Visibility Warning*
Fwd TCP Rcv Window Zero?	Matches flows which have TCP Receive Window Zero using the values 'true' or 'false'. See *Visibility Warning*
Rev TCP Rcv Window Zero?	Matches flows which have TCP Receive Window Zero using the values 'true' or 'false'. See *Visibility Warning*
Fwd Fabric Path	Filter flow observations that go through a particular fabric link in the forward direction. e.g. 'leaf1(eth1/2)->spine(eth1/1)'. Optionally include 'class', 'drops', or 'latency'. e.g. 'leaf1(eth1/2)->spine(eth1/1) latency:[1µs - 10µs)'. See *Visibility Warning*

Continued on next page

Table 5.2.1 – continued from previous page

Columns (names exposed in API)	Description
Rev Fabric Path	Filter flow observations that go through a particular fabric link in the reverse direction. e.g. 'spine(eth1/1)->leaf(eth1/2)'. Optionally include 'class', 'drops', or 'latency'. e.g. 'spine(eth1/1)->leaf(eth1/2) latency:[1µs - 10µs)'. See *Visibility Warning*
Fwd Burst Indicator	Filter flow observations by the number of bursts observed during the minute in the forward direction. See *Burst detection*
Rev Burst Indicator	Filter flow observations by the number of bursts observed during the minute in the reverse direction. See *Burst detection*
Fwd Max Burst Size (KB)	Filter flow observations by the size of the maximum burst (in kilobyte) observed during the minute in the forward direction. See *Burst detection*
Fwd Rev Burst Size (KB)	Filter flow observations by the size of the maximum burst (in kilobyte) observed during the minute in the reverse direction. See *Burst detection*
User Labels (*user_* prefix)	Attributes prefixed with come from user labels.

Note: Because flow data is labelled with User Labels only at ingestion time, User Labels will not appear right away after enabling them. It may take a few minutes before the labels start appearing in Flow Search. Also, the available User Labels will be different depending on which part of the **Corpus Selector** you have selected, since the enabled Labels might have been changed at various times.

5.3 Filtered Timeseries

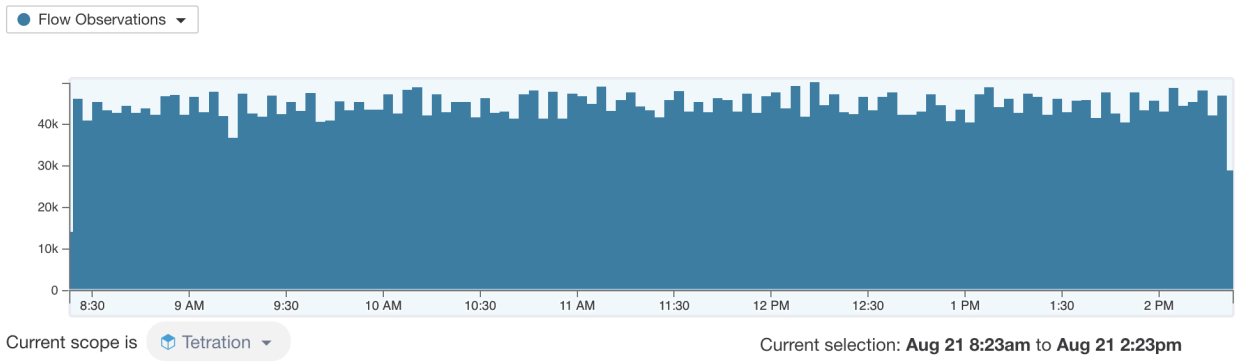


Fig. 5.3.1: Filtered Timeseries

This component displays the aggregated totals of various metrics for the interval selected (the selection made in the above *Corpus Selector*). Use the dropdown to change which metric is being displayed.

Fig. 5.3.2: Timeseries dropdown

Further-narrowing of the selected interval can also be done in this component. Simply click the area of the chart that you'd like to focus on, and the Top N Charts and the data below will all be updated to include only data from that selected interval.

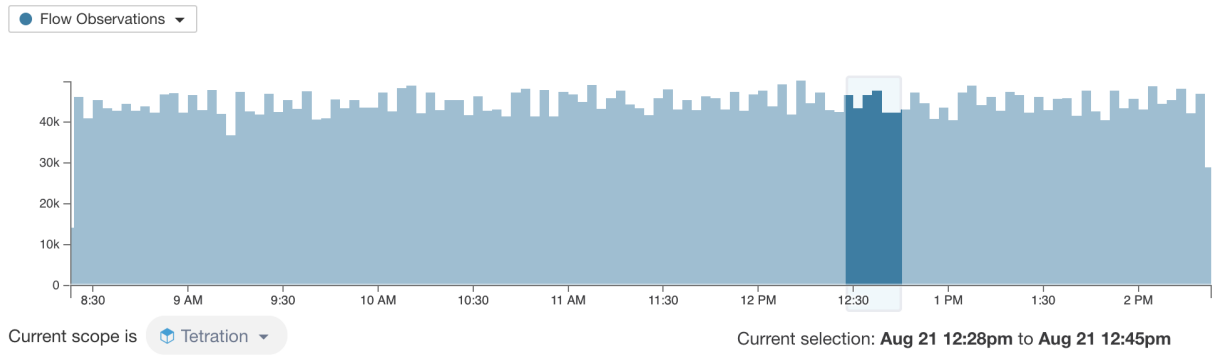


Fig. 5.3.3: Timeseries with selection

5.4 Top N Charts

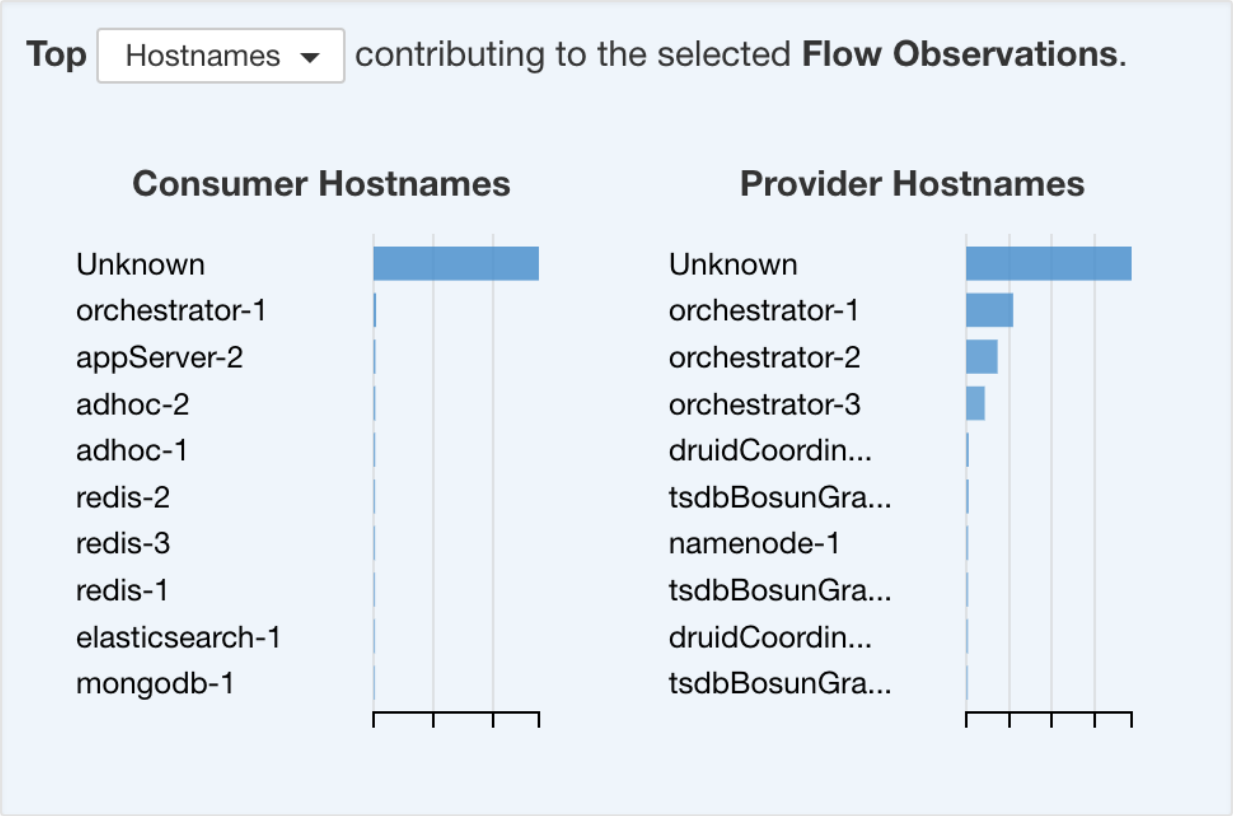


Fig. 5.4.1: Top N Charts

These charts display the Top N values that contribute to the selection in the Filtered Timeseries chart to the left. Selecting a peak in Flow Observations in the timeseries chart, and hostnames in the Top N charts, will display the list of hostnames (Consumer and Provider) that contribute the most to those flow observations. Also, if the timeseries chart is set to display SRTT, then the Top Hostnames will display those that contribute most to that selected SRTT.

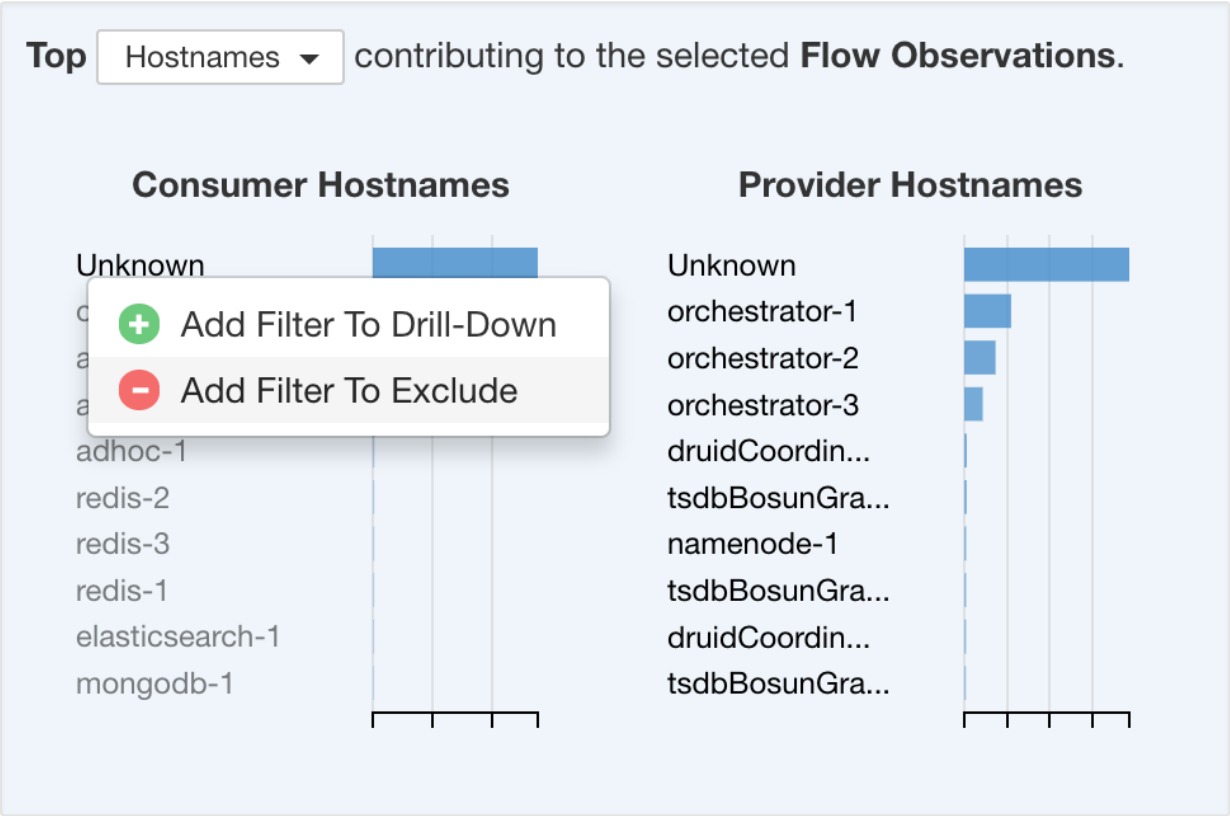
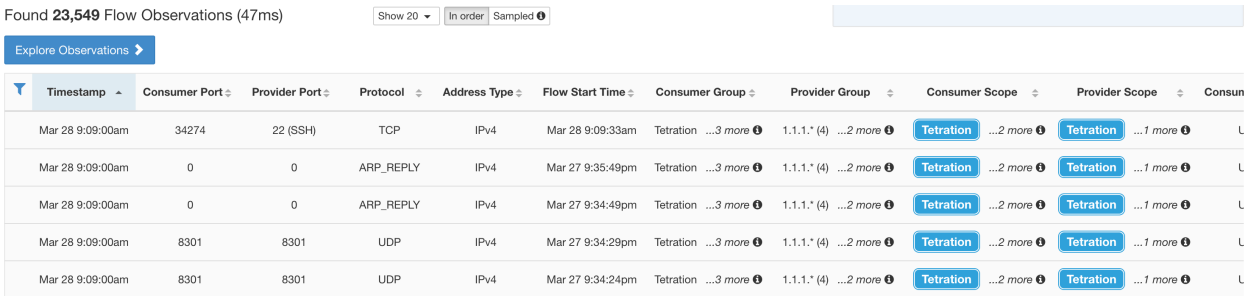


Fig. 5.4.2: Drill-down/Exclude

Clicking on any of the items in the Top N charts will show a menu that allows you to either "Drill-down" or "Exclude" that value. Clicking "Drill-down" will add a filter that will confine the results to just that value. Clicking "Exclude" will add a filter that will exclude that value from the results.

Note: After clicking "Drill-down" or "Exclude", the **Filter** button must be pressed in order for the filter to take effect. This is so that multiple "Exclude" actions can be taken quickly without having the page repeatedly update in the middle.

5.5 Observations List



This is the list of actual **Flow Observations** that match the filters and selections in the page above. By default, 20 will be loaded starting from the beginning of the interval. It's possible to increase the number that are loaded by using the dropdown. It's also possible to load a random set of flow observations from the selected interval by using **Sampled** rather than **In order**. The **Sampled** setting is useful for getting a more representative set of flow observations from the selected interval rather than loading them sequentially from the beginning of the interval.

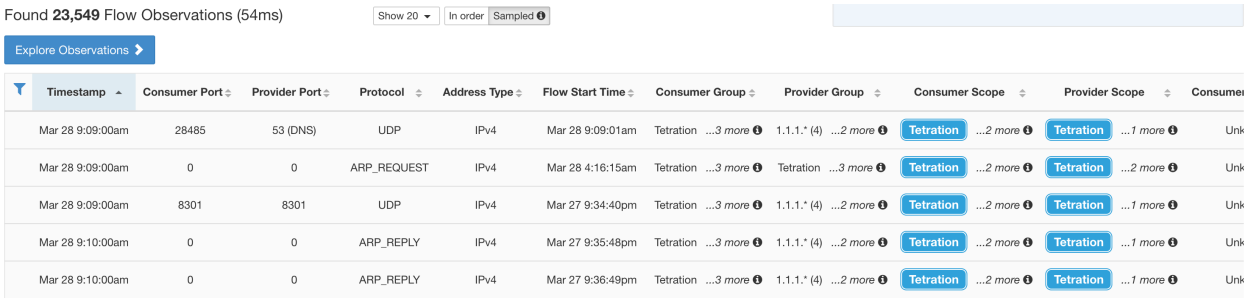


Fig. 5.5.1: Sampled

5.5.1 Flow Details

Clicking on any of the rows will expand the **Flow Details** section below that row. This will display a summary of the flow as well as charts for various metrics for the lifetime of that flow. For long-lived flows, a summary chart will be displayed at the bottom that will allow you to choose different intervals fow which to view timeseries data.

Fig. 5.5.1.1: Flow details

For flows labelled with Fabric Path information, **Fwd/Rev Fabric Latency** and **SRTT** will be available. Time-series charts for other metrics such as **Fwd/Rev Burst Indicators** and **Fwd/Rev Burst+drop Indicators** may be displayed if available. See *Visibility Warning*.

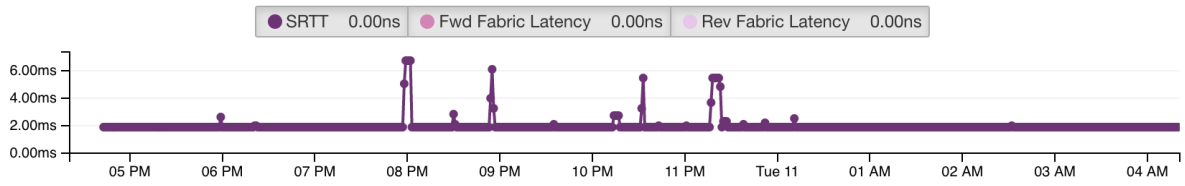


Fig. 5.5.1.2: Latency

In addition, details about the **Fwd/Rev Fabric Path** will be available. Each link can be clicked, toggling **Latency** and **Drop Indicators** timeseries charts (when none-zero). Clicking on **Fwd** or **Rev** navigates to the *Fabric Path Overlay* page drill-down for the flow.

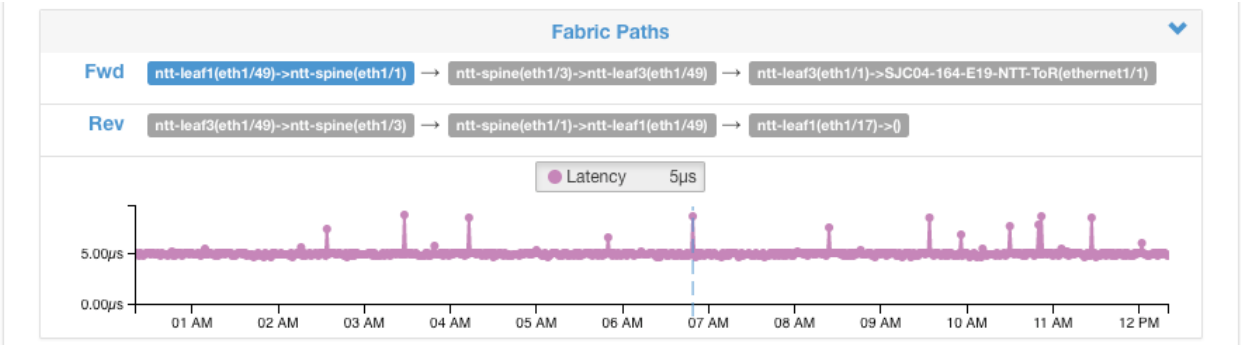


Fig. 5.5.1.3: Fabric paths

5.6 Explore Observations

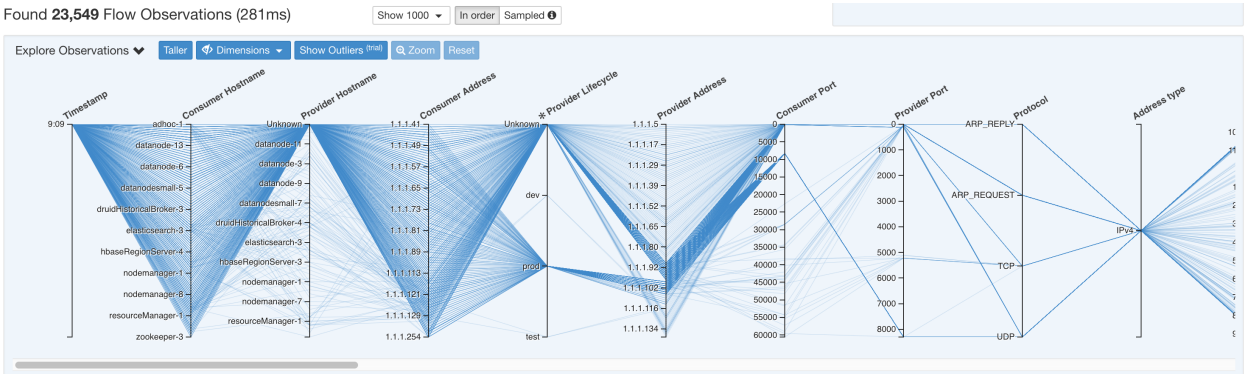


Fig. 5.6.1: Explore Observations

Clicking on the blue **Explore Observations** button will enable a chart view that allows quick exploration of the high-dimensional data (this is called a "Parallel Coordinates" chart). A bit overwhelming at first, this chart can become very useful when enabling only the dimensions you're interested in (by unchecking items in the **Dimensions** dropdown), and when rearranging the order of the dimensions. A single line in this chart represents a single observation, and

where that line intersects with the various axes indicates the value of that observation for that dimension. This can become more clear when hovering over the list of observations below the chart to see the highlighted line representing that observation in the chart:

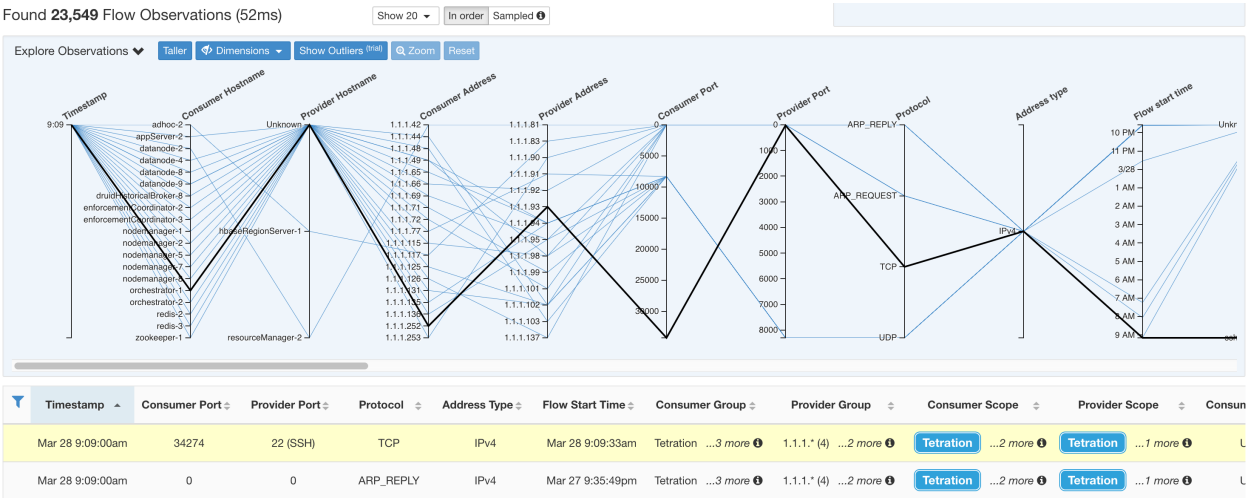


Fig. 5.6.2: Flow Observation hovered

Due to the high-dimensional nature of the flow data, this chart is quite wide by default, and will require scrolling right to see the entire chart. For this reason it's useful to disable all but the dimensions you are interested in.

Sampling vs. In-Order

It's recommended that Explore Observations be done with **sampling** enabled, and with a larger number of flows. This will allow you to see more of the variety of flows that comprise the selected interval. So, if you've selected 2 million flow observations in the timeseries chart above, loading a sample of 1000 will taken uniformly from throughout the interval, whereas loading flows **In-order** will load the first 1000 flow observations from the very beginning of the interval:

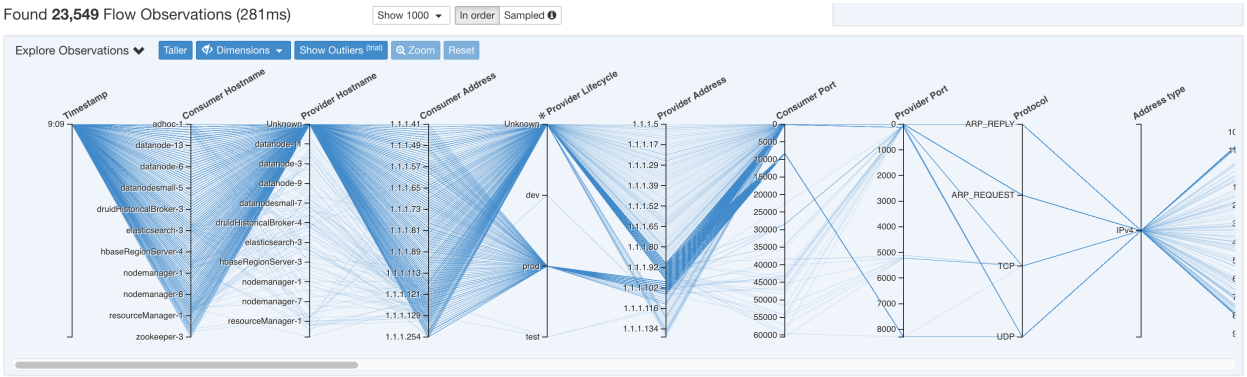


Fig. 5.6.3: 1000 In-order

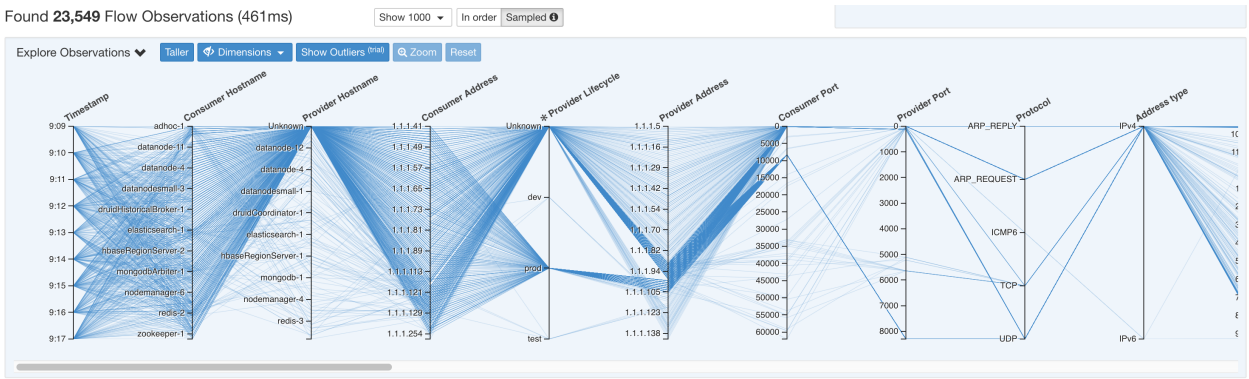


Fig. 5.6.4: vs. 1000 sampled

Notice how the **Timestamp** for all of the in-order observations is from 9:09 and how the observations are evenly distributed through the selected interval in the sampled version.

Filtering

Dragging the cursor along any of the axes will create a selection that will show only observations that match that selection. Click again on the axis to remove the selection at any time. Selections can be made on any number of axes at a time. The list of observations will update to show only the selected observations:

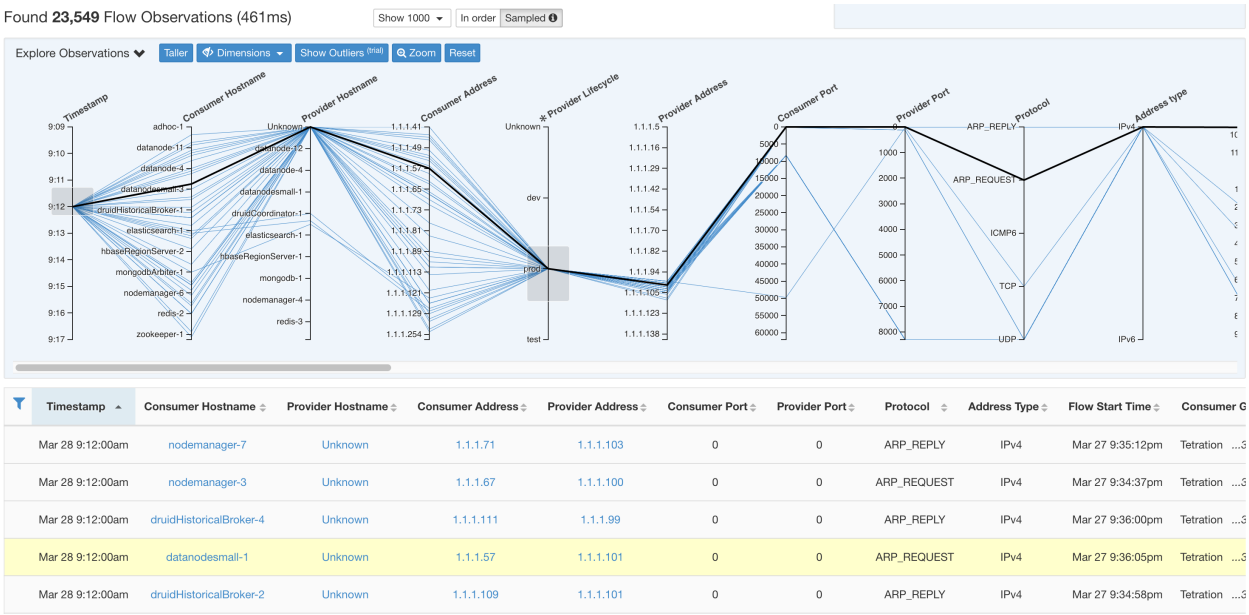


Fig. 5.6.5: Explore with selection

5.7 Client Server Classification

Flow direction (client/server or provider/consumer classification) is important for visibility, mapping applications (ADM), policy generation and enforcement. Every unicast flow has a client and a server classification.

For example, if there are clients (192.168.1.1-192.168.1.3) accessing a web server (192.168.2.1) using https, typically source port is an ephemeral port in the range 1025-65535 and destination port is 443.

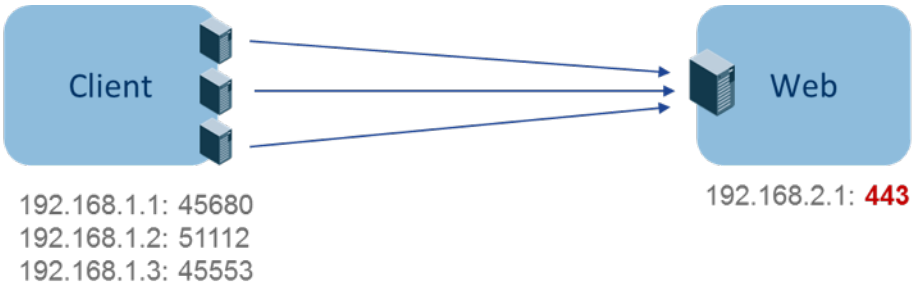


Fig. 5.7.1: Client Server Classification

The accurate client server direction is:

- Client: 192.168.1.1-3
- Server: 192.168.2.1
- Services: TCP port 443

Policies generated (by ADM) are shown in the figure below (with left endpoints grouped):

Fig. 5.7.2: Policies generated

Now, if the client - server direction decision is reversed (an inaccurate classification), that is:

- Client: 192.168.2.1
- Server: 192.168.1.1-3
- Services: the list of ephemeral ports (45680, 51112, 45553)

then, in the above inaccurate classification, policies generated may be as shown in the figure below:

Fig. 5.7.3: Inaccurate classification

This consumes more resources in terms of policy enforcement. In addition, depending on how you enforce the policy, even though 192.168.1.1-3 uses these ephemeral ports, they cannot access 192.168.2.1. For example, if you use

Tetration software sensor enforcement, the enforcement policy for Client to Web above (ESTAB) does not match with traffic generated by Client destined to Web (NEW, ESTAB).

Timestamps and TCP flags are used in Tetration to determine the client-server direction. If there are no TCP flags information (SYN, SYN/ACK) because, for example, the packets could be UDP/ICMP or a HW sensor is used that does not support direction signals, then user-defined override rules, timestamps, and other heuristics are used to infer the flow direction. Heuristics by definition do not guarantee 100% accuracy. Client-server accuracy is a function of the type of sensor used and the conditions in which sensors are used. The user can use Tetration's REST-API (OpenAPI) to insert client-server override rules to identify the server ports for those flow types that Tetration gets the direction wrong. Then allow Tetration to process new flow data captured with those rules in place, and then generate the policies over the time duration when the flow direction were fixed. For more details on the API to specify override rules, refer to: *Client Server configuration*. Note that users can also define their own manual policies and examine/remove the undesired policies. See *Policies* in particular *Ungrouped Policy Table View* in ADM.

5.7.1 Sensor type recommendation

Deep visibility or Enforcement Software agents provide best signals to Tetration client server classification algorithms. It is strongly encouraged to consider deploying deep visibility or enforcement agents. These agents get all the necessary signals to drive the correct client-server classification. If deployment of deep visibility or enforcement agents is not possible for some workloads it is recommended to use ERSPAN sensors and stopping there for policy generation or ADM. Other flavors of sensors like Universal sensors or hardware sensors help with visibility but require a lot of manual work from the operator for policy generation and ADM. Tetration will assist as best as it can and we are continuously improving our heuristics algorithms based on feedback.

When the correct client server direction information is not available, Tetration uses user defined overrides or heuristics to infer what the direction may have been. Heuristics by definition do not guarantee 100% accuracy. The accuracy drops with type of sensor used and the condition in which was used.

The following is the recommended order for client-server decision for policy generation use cases:

- **Deep visibility or enforcement agents**: For best results, use Software Sensors (Deep Visibility or Enforcement agents). Traffic flows started before the Sensor was started would be processed by heuristics discussed below.
- **ADC Sensors like F5/Citrix/… agents**: These agents gather the client server state from the ADC devices and stream that source of truth to Tetration.
- **ERSPAN sensors**: With ERSPAN sensor, user needs to take care of providing full visibility of the traffic to and from the workload in question, and make sure the ERSPAN sensor sees all the spanned traffic. The ERSPAN sensor must also not be over subscribed, so that its visibility is not impaired of the network communication of the workload. Furthermmore, user must ensure that packet drops for ERSPAN sensors are kept to the minimum. The operator will not see process information with the network flow information for ADM computation.
- **Universal Sensors**: Universal Sensors take periodic snapshots so their visibility is limited to what was the system snapshot. Because they look at the OS state, they usually have flow direction right, but the snapshots can cause them to lose visibility into short flows.

While using any sensors listed below, user has to sign up for lot more manual work on policy analysis and generate exception rules. Tetration will use extensive use of heuristics, which by definition are not 100% accurate.

- **Nexus 9k FX2 Sensors**: The FX2 and superior switches provide client server direction support signals but can get overwhelmed if the flow activity is high in the deployments. We strongly recommend using collection rules to filter the state that goes into the switch flow tables, so that only the interesting flows are captured. As the filter rules and flows are recorded by the switch ASIC which has limited table sizes for both data structures, the operator has to take care while defining collection rules and tracking how many flows are recorded by the ASIC. When there is no direction support signals, Tetration has to fall back to heuristics, which in the rare cases can be incorrect, and thus could require more manual work on behalf of the end user – like defining exception rules for Tetration.

- **Nexus 9k FX Sensors**: The FX series of switches do not provide the direction support signals. As a result, Tetration algorithms have to fall back to heuristics, which in the rare case if incorrect require more manual work on behalf of the end user – like defining exception rules for Tetration.
- **Nexus 9k EX Sensors**: The EX series of switches do not provide the direction support signals and have smaller flow tables than the FX and FX2 series. In such scenarios, Tetration has to fall back to heuristics, which in the rare case if incorrect requires more manual work on behalf of the end user – like defining exception rules for Tetration. In addition Tetration sees lesser flow information or unidirectional flows.
- **Netflow Sensor and AWS VPC flow logs (Cloudwatch logs)**: Netflow and AWS VPC Flow logs are sampled and aggregated flow data. The aggregation and sampling lose client server direction information. This impacts ADM and policy generation results and makes the problem harder. Netflow/AWS VPC flow logs are excellent for high level visibility. Tetration has to fall back to heuristics, which in some cases if incorrect require more manual work on behalf of the operator – like defining exception rules for Tetration. Netflow and VPC flow logs also miss some of the short flows and the signal quality depends on the device producing Netflow/VPC logs. We recommend using Netflow with Tetration for specialized use cases like stitching flows through L3/L4 NAT devices like Application Delivery Controllers (or Server Load Balancers) to provide Tetration visibility into which flow is related to which other flow.

More details of the Client Server direction analysis follow.

5.7.2 Identifying Producers (aka Servers) and Consumers (aka Clients) for a flow

There are multiple ways (often heuristics) that are used to detect servers:

- If sensor sees the SYN handshake, it can figure out who the server is.
- Based on time - the initiator of a connection is deemed client.
- Degree model - a server will typically have many clients talking to it. In contrast, the degree for client port is expected to be far less.

The priority order is SYN_ANALYSIS/NETSTAT > USER_CONFIG > DEGREE_MODEL.

The thinking behind giving SYN_ANALYSIS higher priority over user config is that config can get stale, and that sensor has the best vantage point to establish ground truth. DEGREE_MODEL is where learning/heuristics come into play, and the accuracy cannot be 100% guaranteed.

It is possible that our heuristics for client server detection can go wrong - in spite of our best intentions and continuous algorithmic refinements that we make in this area. For those scenarios, the OpenAPI interface can be used to punch well known server ports. These configs are not applied to past flows, and only affect markings on flows from that point on (i.e., going forward). It is intended as a last resort fallback, rather than the normal modus operandi.

We also make it a point to not keep flipping the client server marking for the full duration of a given flow (even if we get it wrong, and when our internal models have changed - which they do over time, as more flow patterns are observed/analyzed). Higher/equal priority updates are allowed to override lower priority ones (we will flip client server for the existing flows as well). In other words, the stickiness of marking "for the lifetime of a flow" only applies to degree model based marking.

CHAPTER

SIX

SEGMENTATION

Applications in Tetration are containers for defining policies or generating policy suggestions from Tetration as well as segmentation (policy enforcement). Applications play a central role in many Tetration features including policy enforcement, policy compliance and visibility.

Application Dependency Mapping (**ADM**) is a functionality in Cisco Tetration that helps provide insight into the kind of complex applications that run in a datacenter.

Furthermore, ADM enables network admins to build tight network security policies based on various signals such as network flows, processes and other side information like load balancer configs and route labels. Not only can these policies be exported in various formats for consumption by different enforcement engines, but Tetration can also verify policies against ongoing traffic in near realtime.

The following figure shows the overview of Applications workflow cycle.

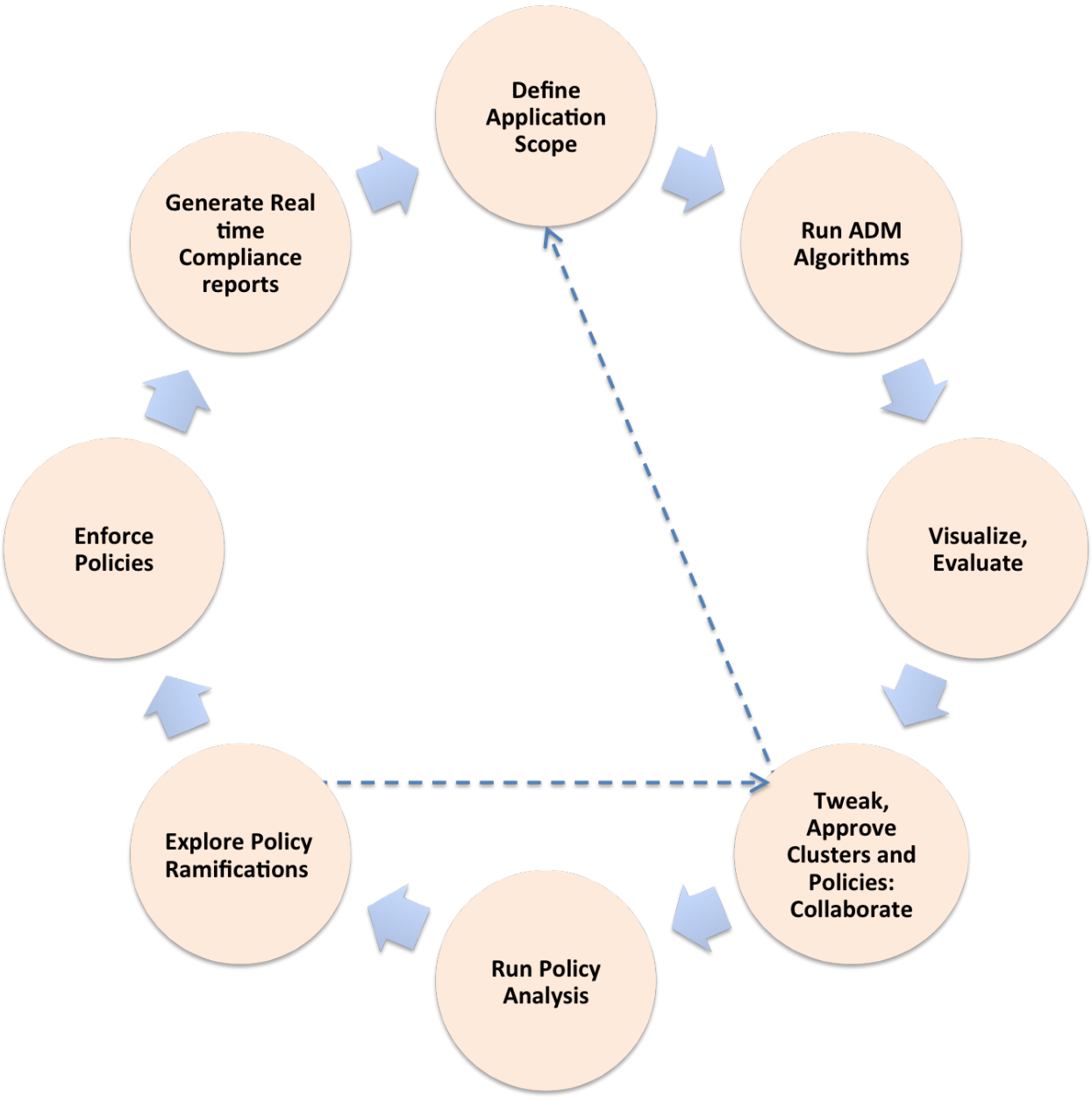


Fig. 6.1: Application management workflow cycle

Application related pages are accessible via the toplevel navigation bar.

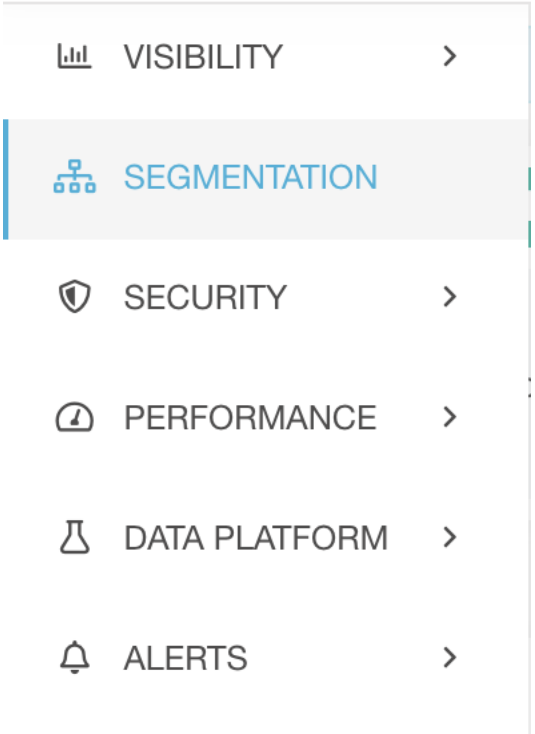


Fig. 6.2: Navigation Bar

6.1 Release Changes

For those familiar with ADM, this section highlights the major changes in recent releases.

Changes in Release 3.3.

- A new capability to generate policies for an entire scope sub-tree has been added, and the option can be selected in the Advanced Configurations section of the ADM run page. See *Deep policy generation* for more details.
- The Applications section is now renamed to Segmentation.
- Applications (Segmentation) landing page is new, offering an overview of not just application workspaces, but also all policies (analyzed or enforced). The page also provides buttons for various functionality such as adding a policy or creating a new filter. As before, clicking on the Applications menu toggles between the most-recently viewed workspace and the overviews page.
- User Defined policies have been migrated to **Approved Policies**. See the *approved policies* section for more information.
- Static mode application workspaces are deprecated. All new workspaces are will be in dynamic mode. The main differentiator of a dynamic mode workspace is the ability of clusters to have dynamic queries and not be limited to a static set of ip addresses. All workspaces will be upgraded to dynamic mode in the next release.
- Published (p*) versions are limited to 100 total. Once this limit is reached users will need to delete old versions using the UI or API.
- New options to *compress policies*, *generate only policies and skip clustering* , *enable service discovery on agent* and *auto accept outgoing policy connectors* upon an ADM run (Advanced Configurations).

Changes in Release 3.1.

- Changes to Policy and Conversations views: A confidence column as a function of client-server confidence is provided (see the new *Ungrouped Policy Table View*). Links to flow search, to streamline flowsearch, are also provided.
- Manual policies defined in a workspace lead to exclusion of flows for policy generation, so redundant policies are not generated by the subsequent ADM run. Excluded flows (based on exclusion filters or policies), are marked in the Conversations View (*Conversations*).
- To designate a cluster as a provided service, the cluster should now be *promoted* to an inventory filter, and in the process one selects the provided service status. This is done in the Clusters page (click on a cluster, promote button is in the right window next to approved). The provided services page now only lists the current provided services (candidates are not shown anymore).
- *Reorder naturally* button for ADM run configuration (child-first, post-order) and order By Number.
- Ancestral user-defined policies: When ADM is ran on a workspace, the user defined policies in the primary latest workspaces of parent and ancestral scopes are excluded from policy generation.
- Confidence score and links to view conversations, quick analysis, and enforcement in *policy side view*.

Changes in Release 2.3.

- A Conversation table view was added, see *Conversations* for more details.
- Imported policies are validated more strictly, missing filters and incorrect property values will return an error.
- It is now possible to delete Policy (p*) versions if they are not being actively analyzed or enforced.
- To help resolve Application delete restrictions it is possible to delete policies from old ADM (v*) versions. For example, a policy referencing a cluster in an "undeletable" application created via a Provided Service.

Highlights of the major changes in Release 2.x compared to previous releases:

- ADM can now generate candidate queries for each cluster in the workspace, based on uploaded user labels, hostnames, and IP addresses. The user has the option of selecting one of the queries, editing it, and associating it with the cluster. A cluster that is associated with a query is **dynamic** in the sense that its membership can change over time: more or fewer workloads can match the query as the inventory changes over time. Approved clusters can now overlap in membership due to user edits of the associated queries. See *Making Changes to Clusters* for associating/editing a query.
- The ADM (Applications) UI currently provides two modes: a **static mode** (same as past releases), and a **dynamic mode**. Queries are generated for clusters *only in the dynamic mode.* The zone view is no longer available in the dynamic mode (the chord-chart view of allow policies was removed in 2.1.1.x in the dynamic mode, but is back). The clusters view has changed to support query to cluster association. Currently, the user has to choose the mode when the workspace is created.
- Policies that have been published for analysis or enforcement can be compared to see additions, deletions, and other changes
- [beyond ADM] *Scopes* can be changed once defined, that is, the queries associated with scopes can be edited. This facilitates initial experimentation with the scope hierarchy and allows for future flexibility as an organization's structure evolves.
- Semantics of inventory (member workload) ownership: children of a parent scope can now overlap in member workloads, and thus for workspace creation, no precedence order needs to be specified among child scopes (in the past, a workload belonged to one scope exclusively). A parent scope gets lowest priority: as long as one child scope owns the inventory, the parent workspace will not own the workload (the workload will not appear in its workspace).
- Note that the changes in membership, eg when a new child scope is added, may take a bit of time to reflect in UI and may also require an explicit commit, in particular in case of a change to the scope definition.

Application workspaces are the containers for defining, analyzing and enforcing policies for a particular application. Each application workspace provides an isolated environment, allowing experimentation with no effect on other workspaces. Many visibility tools are provided to help analyze a set of networked applications, and their interactions with external applications in other scopes.

Application workspaces are meant to be used by multiple users from the same team as shared documents. The level of access to an application workspace can be defined via roles defined on application scopes.

6.2 Navigating to Applications

As a first-time user, clicking on Segmentation tab on the top level navigation bar takes you to the workspaces page where you can view existing or create new Application workspaces.

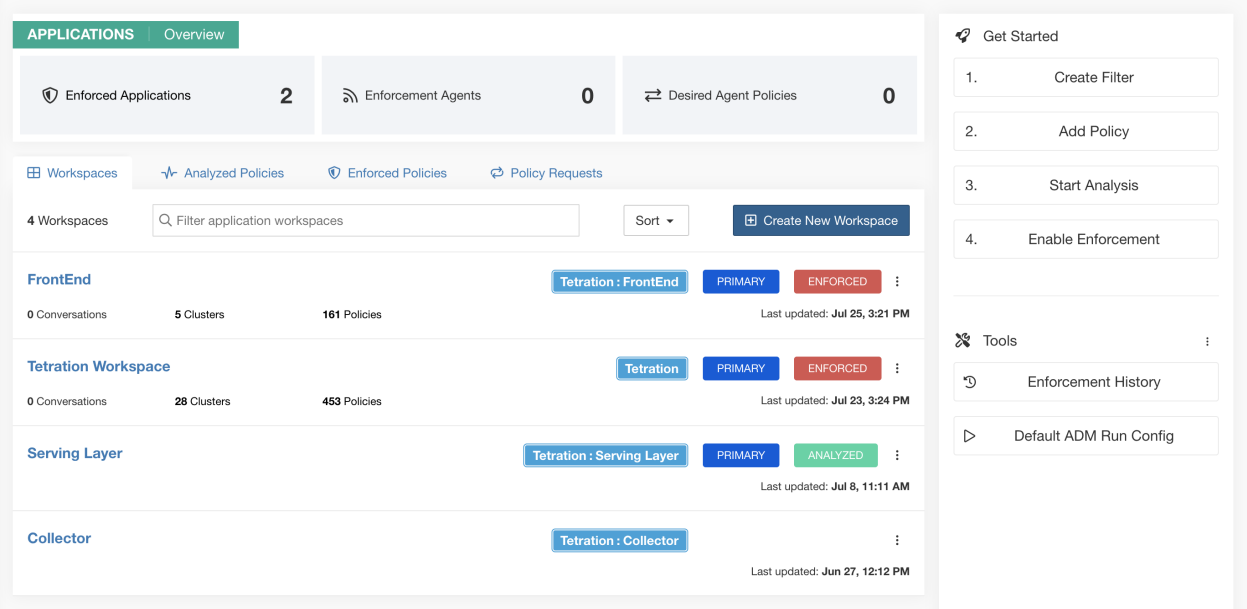


Fig. 6.2.1: Applications Page

Alternatively, clicking on **Switch Application** link in application header navigates to workspace management page. Clicking on the Segmentation tab on the top level navigation bar while viewing a particular application will also navigate to the application workspace management page.

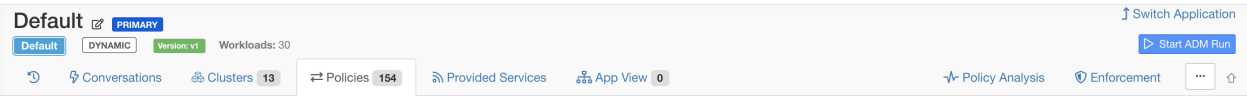


Fig. 6.2.2: Switch to the Applications Page

6.3 Creating Application Workspaces

To create a new application workspace click the "Create New Workspace" button. A modal will appear, fill in the form and click the **Create** button.

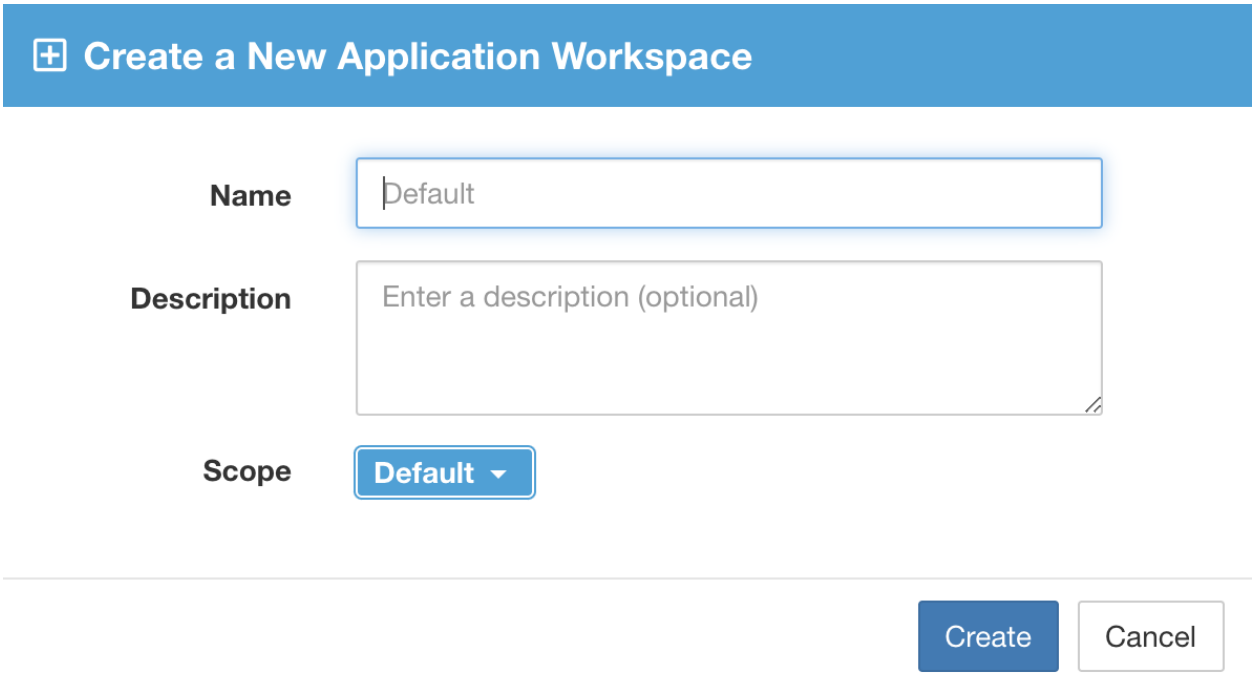


Fig. 6.3.1: Creating Application Workspace modal

The following table illustrates the Required and Optional fields. Field descriptions:

Field Name	Definition
Name	Workspace (application) name
Description	(optional) Workspace description for future reference
Scope	Specifies the application scope (*Scopes*) which determines the set of workloads that can be affected by the policies for this application. Furthermore, roles and access control for this application are defined via the scope.

6.4 Analyzed and Enforced Policies

The **Analyzed Policies** and **Enforced Polices** tabs provide a global view of the analyzed and enforced policies respectively. The view can be used to validate the order and priority of policies in parent applications.

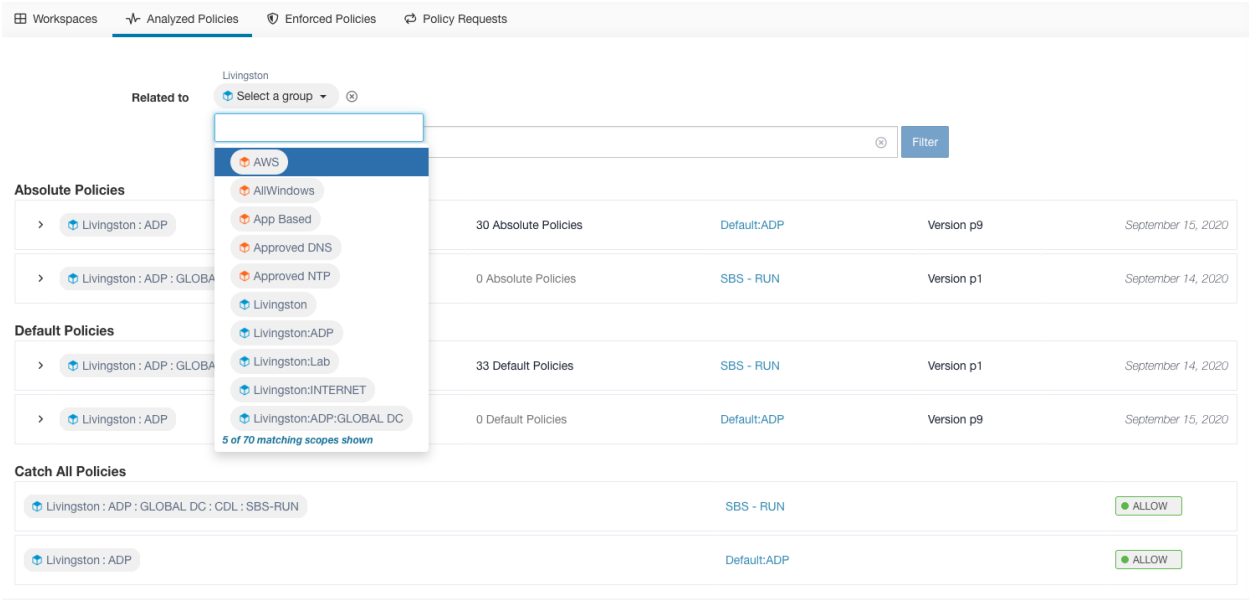


Fig. 6.4.1: List of enforced policies in their policy priority order.

It is possible to first select a scope or filter under the same root scope and limit the list of policies to only those which includes the selected scope or filter as a consumer or provider. On top of this, the list of policlies can be further filtered by additional fields, for example, "port = 80" or "Action = DENY".

Available filters:

Filter Name	Definition
Port	Policy port to match, e.g. 80.
Protocol	Policy protocol to match, e.g. TCP.
Approved	Matches policies that have been marked as *Approved*.
External?	If the policy crosses Application/Scope boundaries.
Action	Policy action: Allow or Deny

6.5 Policy Requests

Each time a policy is created in a primary application, when the provider is a service from another primary application with an associated workspace, and given that the policy doesn't exist already for that application (e.g. that policy or a more general policy may have already been created manually or via a prior ADM run), a *Policy Request* is delivered as a notification to the provider application.

Under the **Policy Requests** tab, in the Applications Overview page, the request counts for all primary applications are shown in one place. Additionally the count of "Auto Created" policies is shown. This is the number of policies created by *Auto-pilot Rules* since the last Published (p*) version was created for that application.

6.6 Enforcement History

Enforcement History provides a list of changes to the list of enforced workspaces and their version. Each section defines an event and a summary of what has changed. Clicking the event provides more information about all the policies that were enforced at that time.

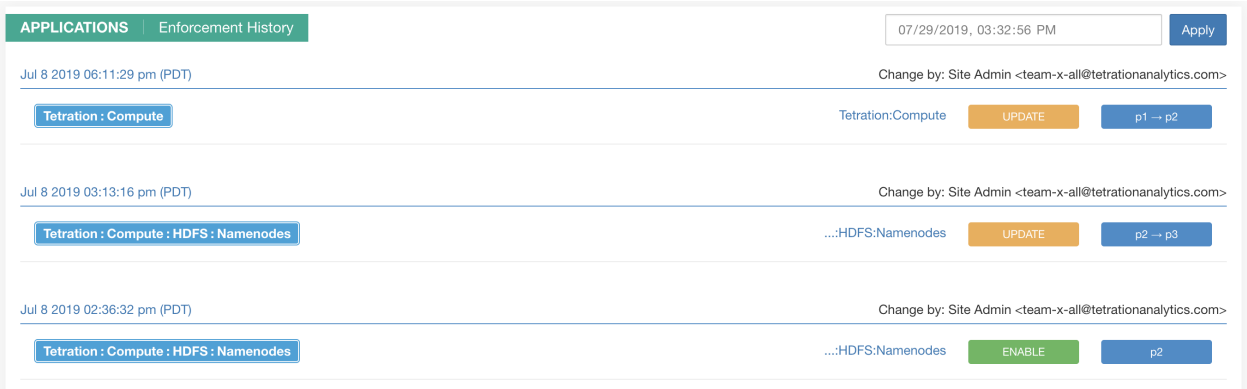


Fig. 6.6.1: Enforcement history view

6.7 Deleting Application Workspaces

Application workspaces can be deleted from the Application Overview page by clicking the menu icon next to the application and selecting "Delete Workspace." Only secondary (non-primary) applications can be deleted.

It is possible for a Cluster in an Application to be referenced by a Policy in another application as a result of a Provided Service. In this case the dependent application can not be deleted and a list of the dependencies will be returned. This information can be used to fix the dependency.

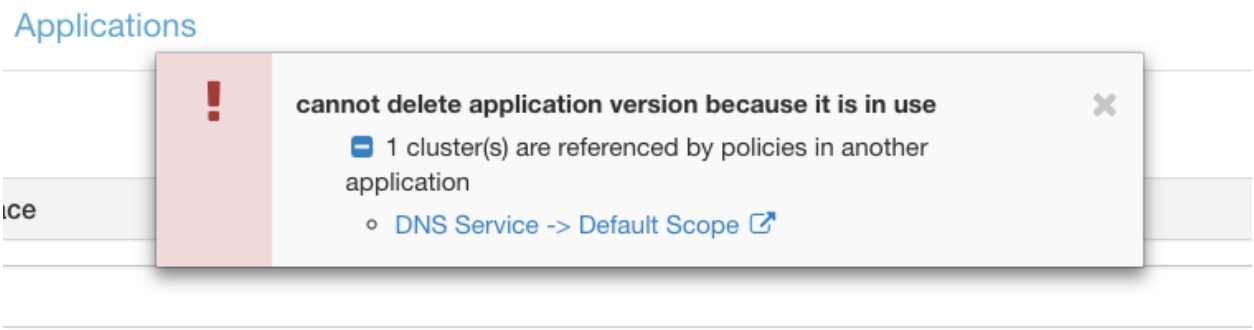


Fig. 6.7.1: List of items preventing the deletion of the application

In rare conditions there may be a cross dependency where Application A depends on a cluster in Application B and a Application B depends on a cluster in Application A. In this case the individual policies or Published (p*) Versions will need to be deleted. The "delete restrictions" error will provide links to all the policies so this can be accomplished.

6.8 Switching Applications

Click on the name of any of the existing application to view or edit that application workspace. The current active application workspace is highlighted in the list.

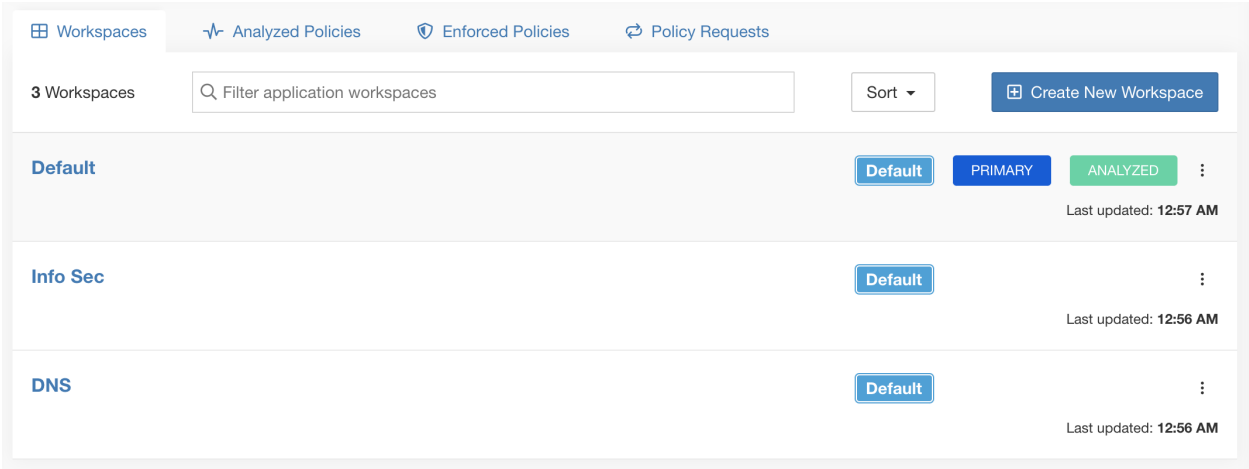


Fig. 6.8.1: Switching Application

6.9 Primary Applications

You can create many application workspaces for a given scope. However, only one of those application workspaces can be promoted to become the **Primary** application for that scope. Many of the more advanced features like policy enforcement, live policy compliance reporting, collaborative security policy definition are only available for primary applications.

The main motivation for the notion of a primary application is to have a single source of truth for the policies that need to be enforced/analyzed without confusing conflicts with other applications from the same scope. Moreover, secondary applications facilitate experimenting with Tetration policy discovery workflows as a **staging** ground without the fear of disrupting existing applications.

There are two ways to make an application primary/secondary. One is by clicking on the secondary/primary label on the application header. The second is on the Application Overview page by clicking the menu icon next to the application and selecting "Toggle Primary."

Please note that many features (tabs) appear as the application is promoted to primary status:

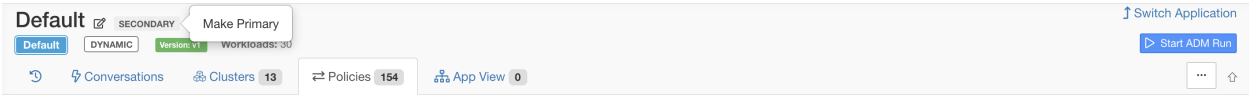


Fig. 6.9.1: Making a Primary Application

6.10 Policy Priorities

Policy priority ordering can be accessed by clicking the menu icon next to "Tools" and selecting **Policy Order**. Since changing policy priorities can affect enforcement results on all applications, this feature is limited to users with very high privilege roles such as site admin.

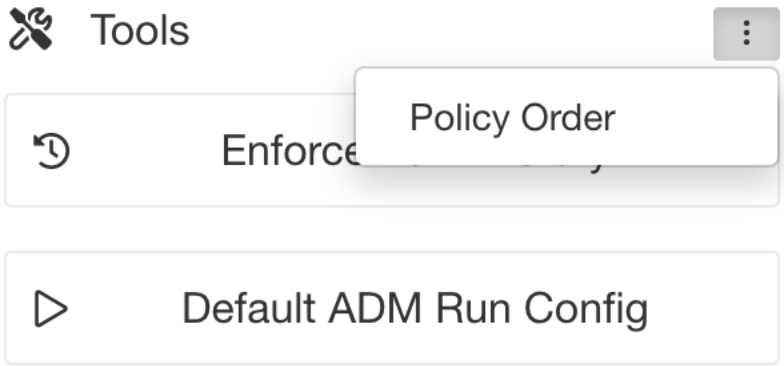


Fig. 6.10.1: Navigating to Policy Priorities page

Once on the Policy Order page, you can see the list of all scopes and their corresponding primary applications according to the current policy priority. There are several ways to reorder the scopes:

1. Dragging the rows up and down.
2. Selecting "By Number" to set a number for each scope to be used for sorting. This can be easier for large lists.
3. Selecting "Reorder Naturally" which does a pre-order tree traversal in which parents are always first. This is the recommended order and any deviation from this should be done with care.

It is very rare that the scope policy priority order needs to be changed, one should always want a parent first ordering so they can take advantage of the hierarchical structure of scopes. However, if sibling scopes are overlapping (not recommended, update scope queries first), it may be necessary to reorder sibling scopes and their children.

NOTE: Changing policy priorities while policy enforcement via Tetration agents is in progress, could change the firewall rules on the hosts for which policies are enforced.

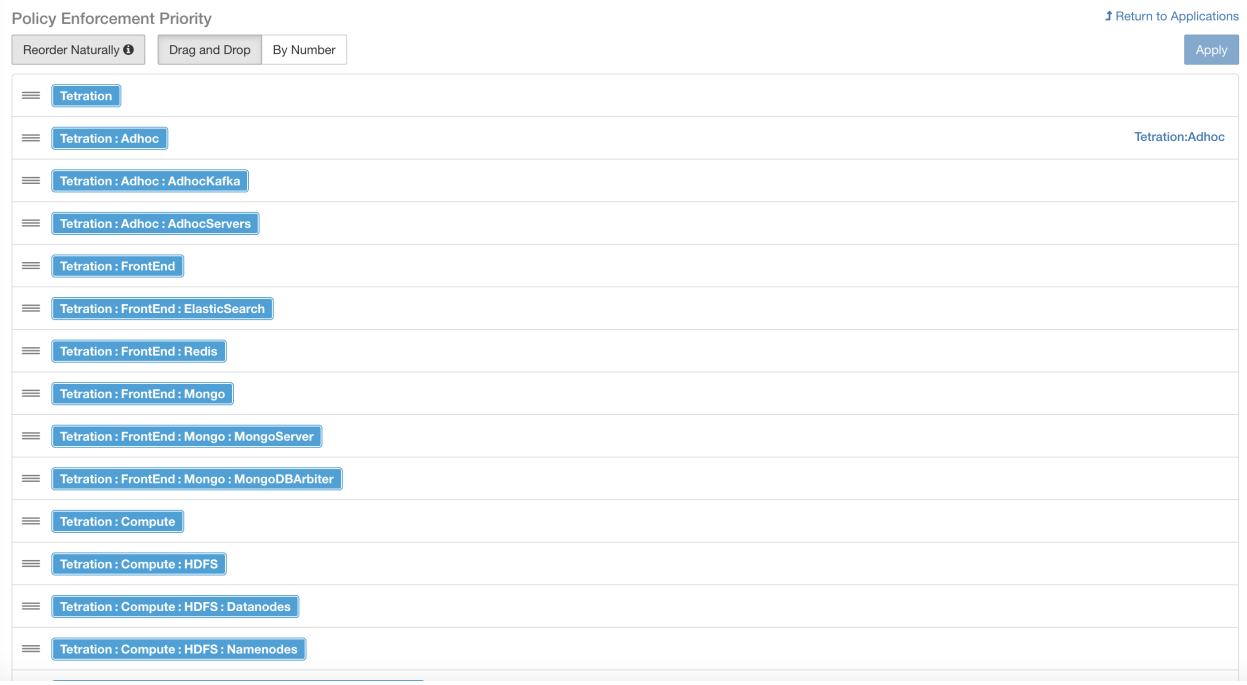


Fig. 6.10.2: Setting Policy Priorities for Scopes

See *Semantics and Viewing* to learn more about policy sorting logic and how policy priorities on scopes translate to ordering of individual policy intents.

6.11 Default ADM Run Config

Default ADM Run Config can be accessed from the "Application Workspaces" page by clicking the menu icon next to "Tools" and selecting **Default ADM Run Config**.

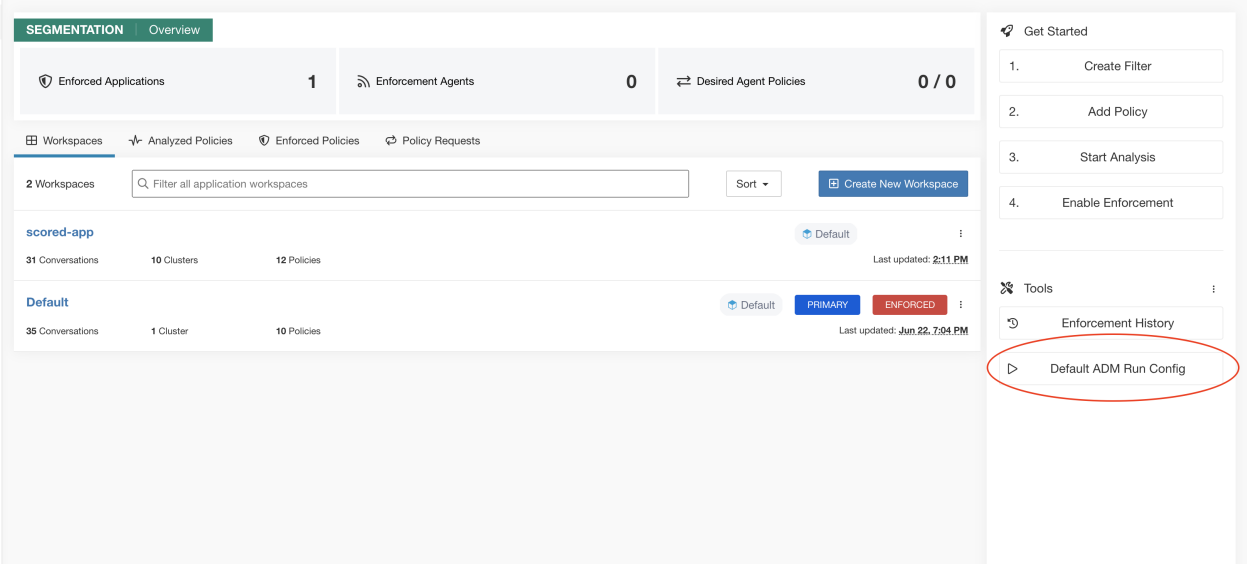


Fig. 6.11.1: Navigating to Default ADM Run Config page

Once on the Default ADM Run Config page, you can see the **External Dependences**, **Advanced Config** and **Default Exclusion Filters** sections. The user can set the default ADM run configuration for the whole root scope. Once a default configuration is set it will be used to preset the options on the ADM run config page.

Notes:

The defined External Dependencies will be used over those of the previous run. "Advanced Configuration" options will use the previous run if available. In paticular, you have the option to use or ignore default exclusion filters in combination with the exclusion filters defined for each application. Those options are controlled by the checked boxes below:

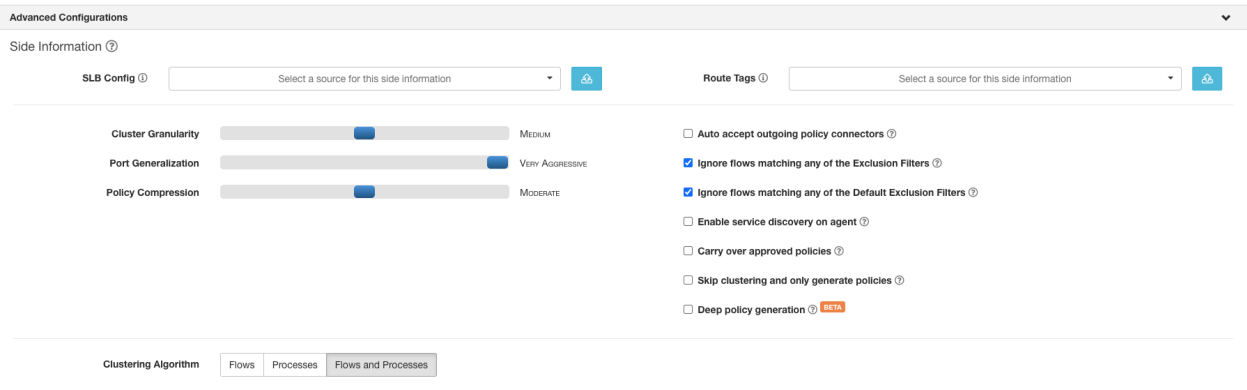


Fig. 6.11.2: Ignore flows matching...

6.11.1 Default Exclusion Filters

Exclusion Filters help you fine-tune ADM run results and policy generation by excluding certain flows from the ADM run input. This results in different allow policies and possibly different clustering results (Note: all conversations

remain viewable in the Conversations View). For example, in order to disallow certain protocols like ICMP in the final allow list model, you just need to create one exclusion rule with a protocol field set to ICMP.

You can make a single global Exclusion filters list available for all application workspaces within a tenant. You can configure these "Default Exclusion Filters" by navigating to "Default ADM Run Config" under the main segmentation page. This list can be used in combination with the workspace specific Exclusion Filters list in "Advanced Configurations".

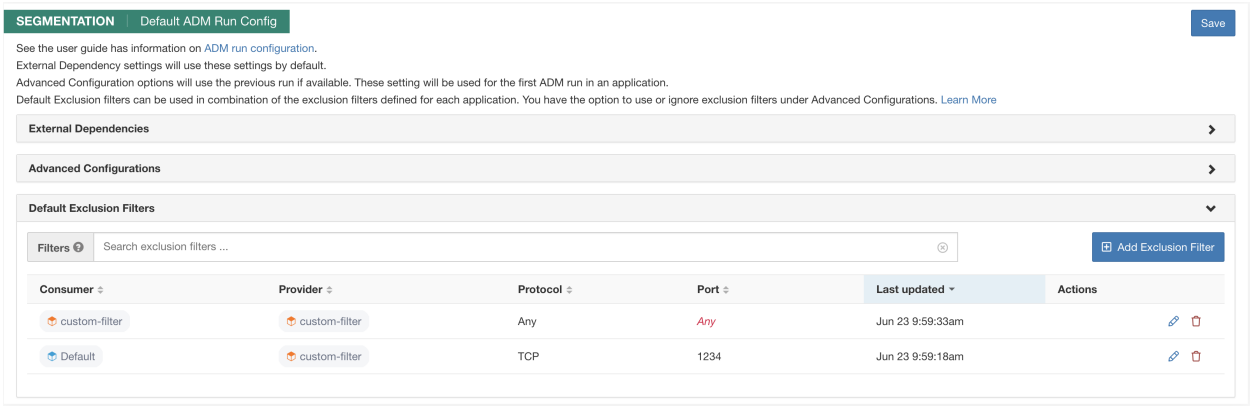


Fig. 6.11.1.1: Default Exclusion Filters

Once on the Default Exclusion Filters section of the page, click on the **Create Exclusion Filter** button to add a new filter to the table. There are four fields to configure, but they are not all required. Any empty field will be treated as a wildcard for matching flows. The available fields are:

- **Consumer**: Matches conversations where the consumer address is a member of the selected cluster/filter/scope. You can specify any arbitrary address space by creating a new custom filter.
- **Provider**: Matches conversations where the provider address is a member of the selected cluster/filter/scope. You can specify any arbitrary address space by creating a new custom filter.
- **Protocol**: Matches conversations with specified protocol.
- **Port**: Matches conversations with provider (server) port matching the specified port, or port range. Port ranges can be defined using a dash separator, e.g. "100-200"

Any conversation that matches all the fields of any exclusion filter will be discarded for the purposes of policy creation and clustering. Click on the **Edit** button to change an existing exclusion filter, and the **delete** button to delete one. These buttons are only visible when the row is hovered by the mouse pointer.

In addition, exclusion filters can be created specific to each workspace. One way to access this list is to click on top right of any ADM page on the '...' icon (by the Enforcement icon) and select exclusion filters. On the ADM run page under Advanced Configurations, one can also click on the exclusion filters link. Visit *Exclusion Filters* for more details

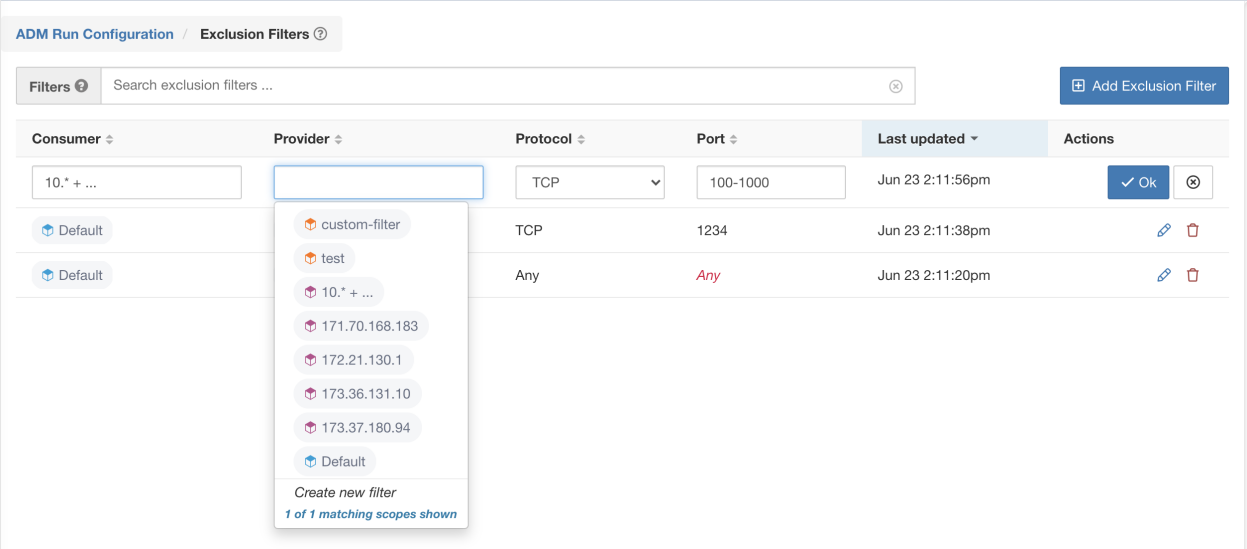


Fig. 6.11.1.2: Exclusion Filters for a workspace

6.12 ADM Concepts

The main concepts behind the application dependency mapping (ADM) tool are clusters and policies, and what running ADM entails.

Cluster: A cluster is a set of workloads (its members). The ADM clustering algorithm generates a partitional clustering of the non-approved workloads that belong to a workspace. The user can improve this grouping by editing the query. This makes it possible that clusters associated with queries may overlap. An **approved cluster** is a cluster that has been explicitly approved by the user, and its workloads are referred to as *approved* workloads. A user approves a cluster to tell ADM not to change the cluster upon ADM reruns: The query associated with an approved cluster is not changed upon reruns. Note that the memberships of approved clusters can change only if the members of the workspace changes. See *Re-running ADM Algorithms* for additional information. A cluster may also be promoted to a **provided service**, which makes collaboration across multiple workspaces easier (can lead to more secure finer-grained policies). See *Collaboration Among Applications*.

Policy: Also known as Cluster Edges. ADM generates a (directed) edge between two clusters if it observes at least one conversation among the member workloads of the clusters (in the time period input to the ADM run). These cluster edges translate to ALLOW policies. Users can modify and define their own allow as well as deny (block list) policies, and a rich set of features are available for prioritizing policies. See *Policies* as well as *Conversations* for further information.

Workload: A workload is an IP. Workloads participate in conversations (can be the end of a conversation).

> **Target Workload:** Any workload that falls within the scope of an ADM workspace, according to parent-child priority, is a target workload or member workload of the workspace. See *Member (Target) Workloads*. Upon running ADM, target workloads are clustered based on their network communications (by default) or processes running on them, or a combination of both signals, unless they are already in approved clusters.
>
> **External Workload:** Any workload that is not a target workload. Such workloads are an end of a conversation with a target workload.

> **Warning:** Static mode is deprecated and all static workspaces will be upgraded to dynamic mode in the next release.

Dynamic vs Static UI Modes: All new workspaces are in dynamic mode. However, existing workspaces before this change may still be in static mode.

In static mode, clusters are lists of literal addresses, that is those workloads that belong to the cluster. If the cluster is approved, this list (or special type of query) does not change. In the dynamic mode, a user can associate a cluster with a more general query. An example query can be hostname containing the substring 'HR'. In the future, if more hosts are added to the workspace with hostname containing HR, the cluster expands to contain them automatically.

In dynamic mode, ADM will examine the hostnames and all the user uploaded labels (See uploads) associated with workloads. For each cluster, ADM will generates a short list of candidate queries based on the hostnames and these user uploaded labels. From these queries, the user can select one, possibly edit it, and associate it with the cluster. Note that, in certain cases, when ADM can not formulate simple enough queries based on the hostnames and user uploaded labels, no (alternate) queries are suggested.

Another difference between two modes is that the zone view is no longer available in the dynamic mode, and the clusters view has changed to support query to cluster association, and query editing.

Port (Interval) Generalization

Some applications such as Hadoop use and change many server ports in some interval, for instance in 32000 to 61000. ADM attempts to detect such behavior for each workload, using the workload's server port usages in the observed flows: by observing only a fraction of total possible ports (but numerous ports, eg 100s), ADM may 'generalize' that any port in, say 32000 to 61000, could be used as a server port by the workload. Ports that fall within intervals are replaced with such intervals (when certain criteria on minimum observed counts are met). This results in fewer cluster edges and more compact policies. Interval estimation is important for computing accurate policies: without sufficient generalization many legitimate future flows would be dropped if the policy is enforced. By merging numerous ports into one or a few intervals, the rendering time of the UI is sped up significantly as well. A knob in advanced ADM Run settings allows the user to control the degree of port generalization including disabling it.

Allow Policy: An *ALLOW* policy is a rule that specifies what communication (in terms of attributes such as service ports and protocol, and client/server roles) is allowed between two ends (workloads, clusters, scopes, inventory filters). A *block list* or *DENY* policy has an opposite meaning: what kind of communication is not permitted (should be dropped). See policy *Semantics and Viewing*. ADM runs automatically generate ALLOW policies, and users can manually modify these or add their own policies.

6.13 Navigation

6.13.1 Header

The Application header serves two main purposes:

1. Provide high level context about the application workspace and most recent run by showing the name, and high level stats about the application like number of clusters, workloads and app views.

2. Quick navigation among several views designed to simplify examination and consuming ADM analysis results.

The following figure is annotated with some of the features of the header:

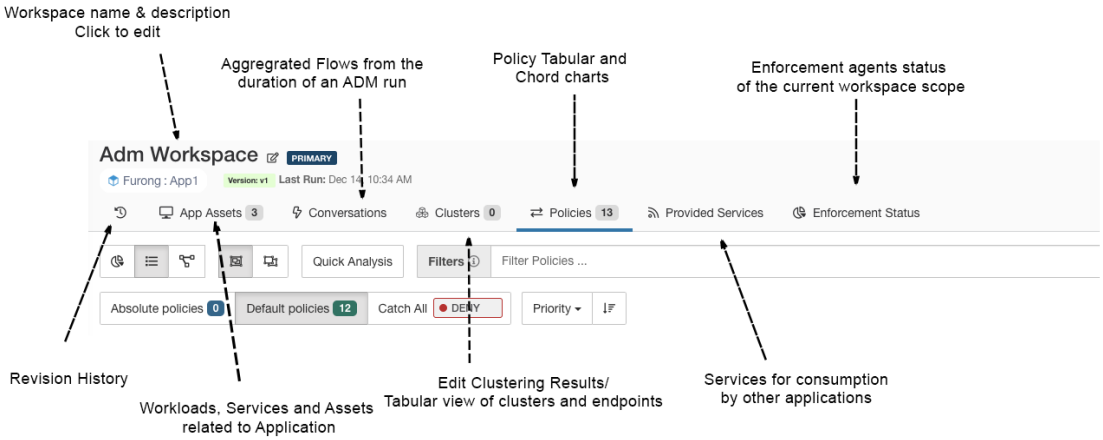


Fig. 6.13.1.1: Application Header with labels

6.13.2 Side Panel

Side panel feature is shared across many different application pages. Side panel typically comes with two main tabs: Info & Search.

The **info** tab provides context for many of the complex charts by showing more details about selected objects. Controls within the info tab allow for easy navigation to other views to help users get more insight about certain aspects of hosts or applications.

The **search** tab is the simplest way to find any relevant workload, cluster, or policy in a workspace. A search is defined using a set of **filters**. Multiple filters will be treated as logical ANDs. For IP addresses and numeric values, logical ORs can be indicated using a comma: 'port: 80,443'. Range queries are also supported for number values: 'port: 3000-3999'.

Available filters:

Filters	Description
Name	Enter a cluster or workload name. Performs a case-sensitive substring search.
Description	Searches cluster descriptions.
Approved	Matches approved clusters using the values 'true' or 'false'.
Address	Enter a subnet or IP address using CIDR notation (eg. 10.11.12.0/24). Will match workloads or clusters which overlap this subnet.
Supernet	Enter a subnet using CIDR notation (eg. 10.11.12.0/24) to match clusters whose workloads are fully contained in this subnet.
Process	Searches workload processes using a case-sensitive substring search.
Process UID	Searches workload process usernames.
Port	Searches both workload provider port and policy port.
Protocol	Searches both workload provider protocol and policy protocol.
Consumer Name	Matches a policy's consumer cluster name. Performs a case-sensitive substring match.
Provider Name	Matches a policy's provider cluster name. Performs a case-sensitive substring match.
Consumer Address	Matches policies whose consumer address overlaps with the provided IP or subnet.
Provider Address	Matches policies whose provider address overlaps with the provided IP or subnet.

The following figures illustrate the search functionality of the side panel:

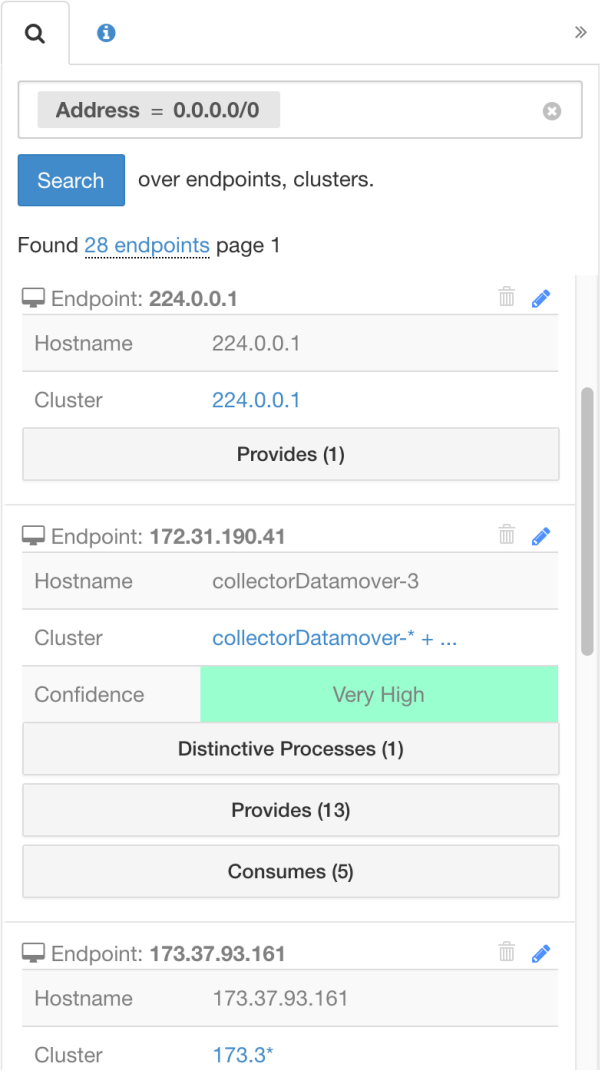


Fig. 6.13.2.1: Search Functionality of Side Panel

To filter by a specific type, click the result total and select the type from the dropdown. A type filter will be added and the search will be rerun.

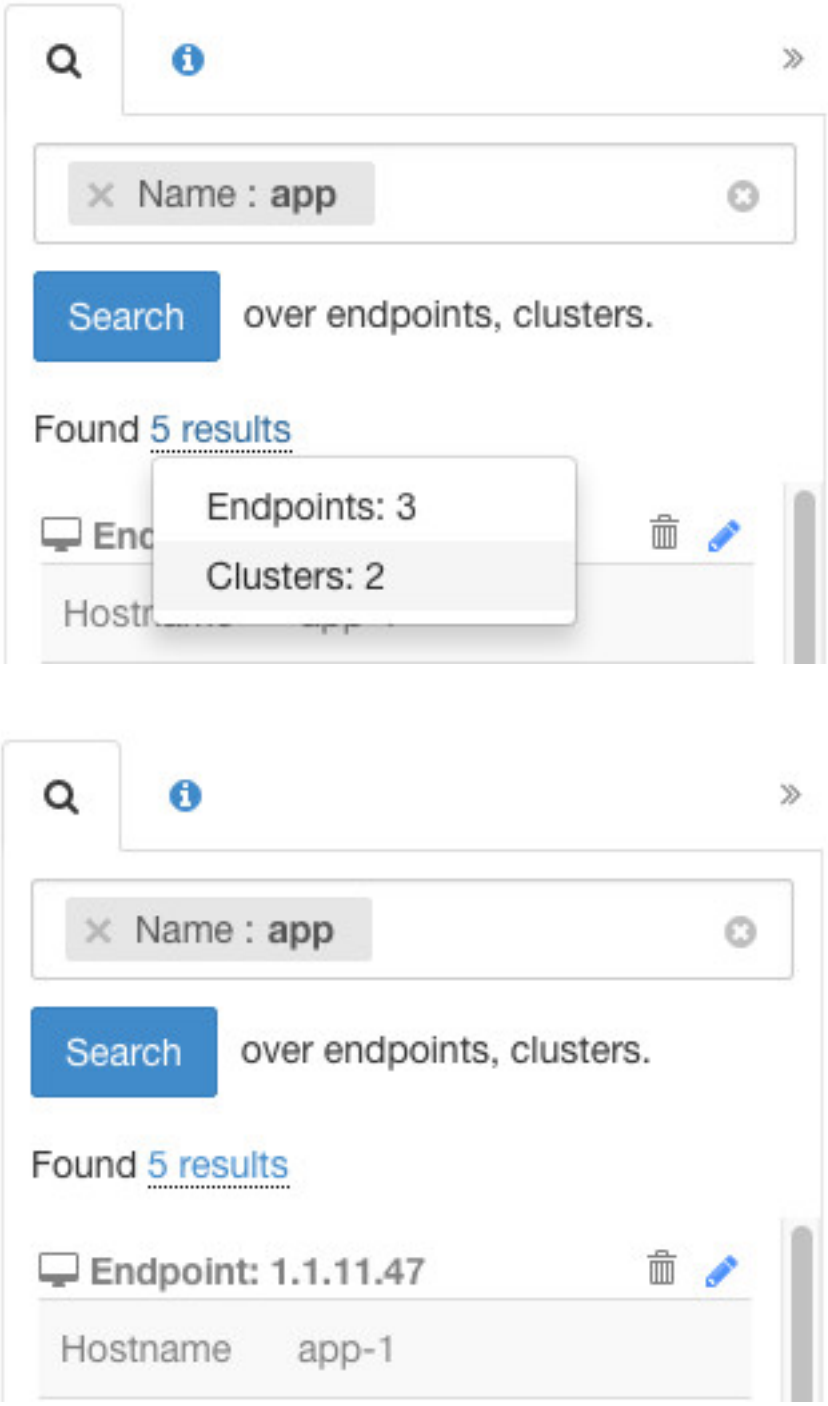


Fig. 6.13.2.2: Filtering results by a specific type

The figure below shows the side panel providing context for a selection for one of the charts (policy view). This is a common behavior across many charts.

NOTE: You can resize the side panel by dragging the edge.

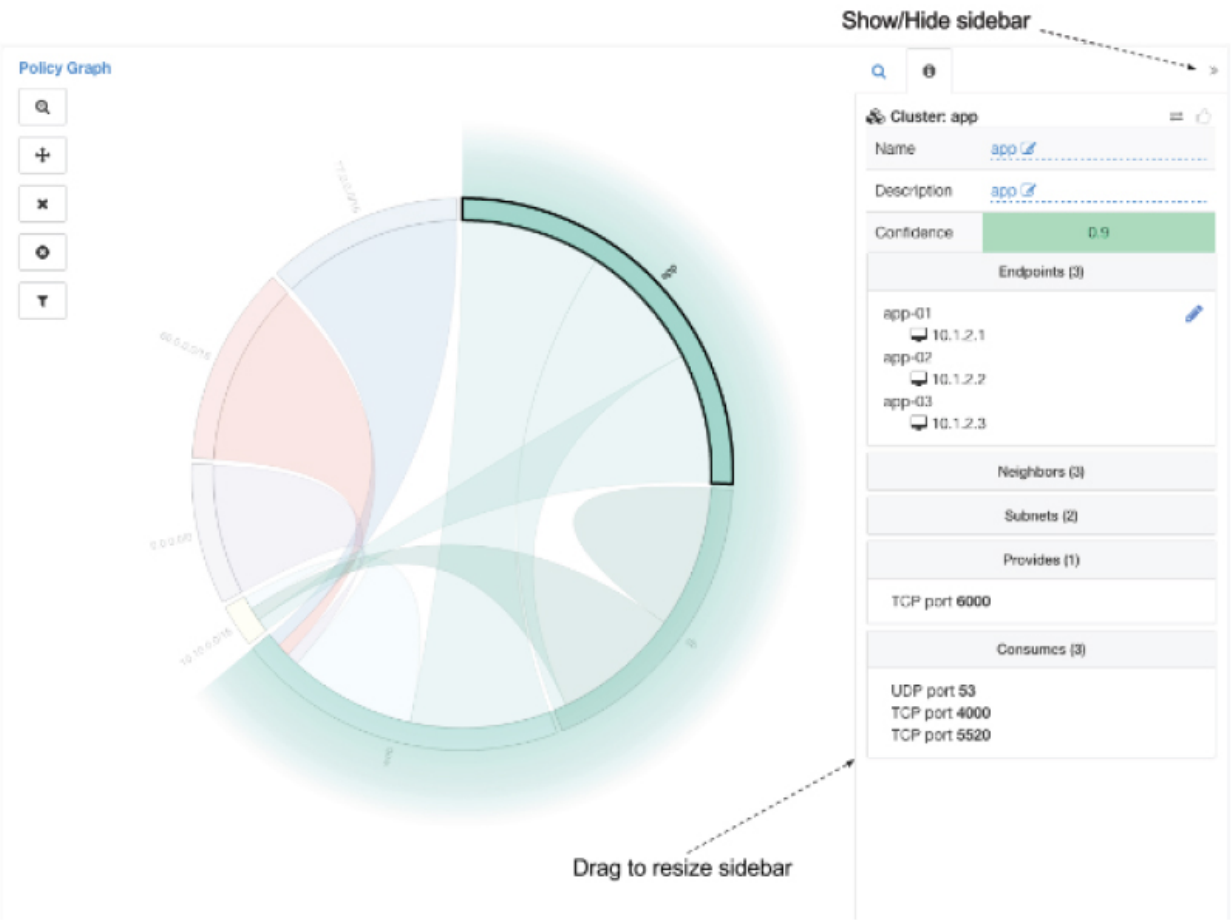


Fig. 6.13.2.3: Policy View

6.14 Running ADM

An ADM run groups similar workloads of a workspace into clusters and generates (allow list) security policies among the clusters. To initiate a run (or a rerun), the user selects the time range to gather the data on the workloads (for computing similarities and policies), and may change other run parameters (the run configuration) and then launches a run. The user can then explore and modify/approve the results, and do subsequent runs (reruns).

6.14.1 ADM Run Configuration

Click on the **Start ADM Run** button on application header section, to navigate to **ADM Run Configuration** page.

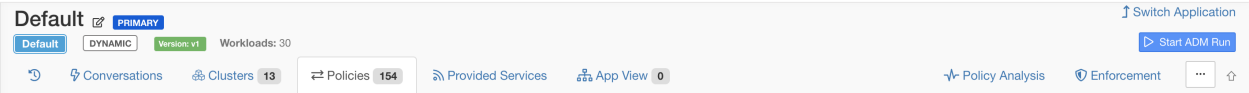


Fig. 6.14.1.1: Navigating to ADM Run Configuration

6.14.1.1 Basic ADM Run Configuration

The minimum requirement to submit a run is to select a date range. Effectively, the user is asking to group the member workloads that are similar into clusters and generate security policies based on the observations in the specified date range. The ADM algorithms use all the available signals to decide whether workloads are logically running the same set of services and should be the grouped together, and infers a set of allow policies based on successfully network activities.

Steps to submit an **ADM run** are as follows:

1. Select date range using date pickers
2. Submit ADM Run

Flow summary data used by ADM runs is currently computed every 6 hours. Thus, upon initial deployment of the Tetration appliance, ADM is not runnable until such data is available.

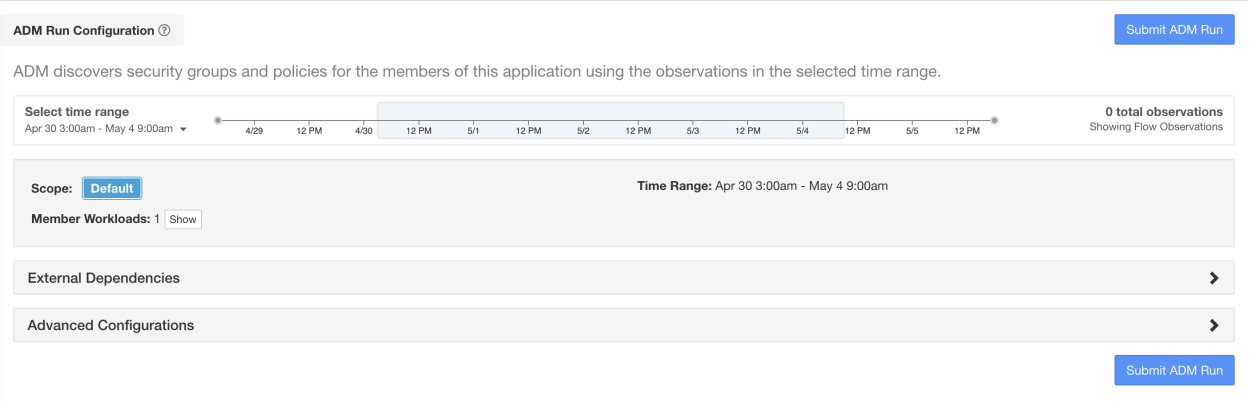


Fig. 6.14.1.1.1: Submitting ADM Run

6.14.1.2 Member (Target) Workloads

For every application, ADM algorithms are run on the **member workloads** of that application to infer policies relevant to that application. A **member workload** for an application is an IP address that belongs to that application as defined by its scope and the parent-child priority semantics (children always take precedence over parent in terms of ownership of workloads, but child scopes can overlap). Furthermore, ADM only follows the **latest definitions** of the application scopes when analyzing conversations. That means in determining which conversations to analyze in the ADM run, which are those conversations in which at least one end is a member workload in the time range selected, workload membership is based on the most current inventory information (scope definitions), regardless of any changes in workload membership prior to the time of the ADM run.

A note on parent-child priority: The sub-scopes (children) of any particular scope are by definition fully overlapping with, and have higher priority than the parent scope. Therefore, the exclusive members of the parent scope is limited to workloads that are *not already claimed* by the children. If all of the members of the parent scope are claimed by its children, it means that the scope is cleanly partitioned into separated applications. In this case, there is no need to run

ADM algorithms on the application of the parent scope, since ADM runs on children's workspaces would infer all the necessary policies to secure each application.

Limits: Note that a maximum of 5000 member workloads is currently recommended for an ADM run, and the number of conversations (which is computed early during an ADM run) should not exceed 10 million (except for deep policy generation mode, in which the limits are 25000 workloads matching the scope query, with 20 million conversations), otherwise the ADM run may fail. The limits are imposed for efficiency (such as to keep the clustering time to within a few hours) as well as UI response time and other user experience considerations. Therefore, the user should break larger scopes into smaller child scopes as necessary.

You can view the member workloads before submitting an ADM run by clicking on the **show** button next to member workload count:

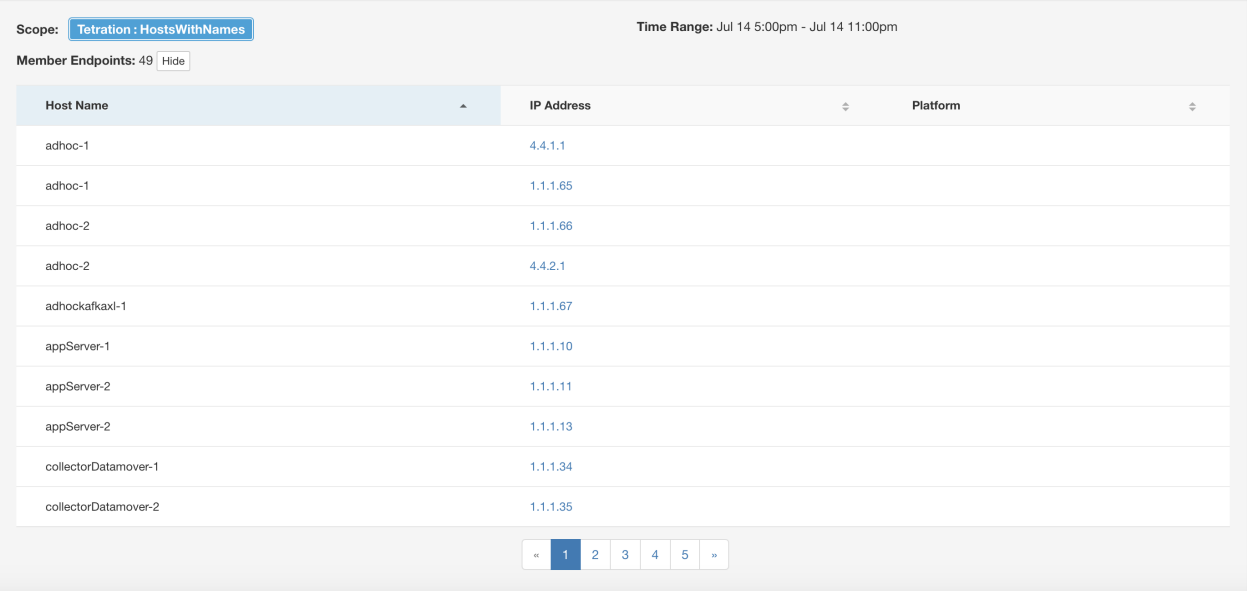


Fig. 6.14.1.2.1: View Member Workloads

6.14.1.3 ADM Run Progress

ADM run progress is always visible in the header. Navigating to other applications, does not affect the progress. You can abort the run while in progress using the **abort** button.

Once the run is complete, a message is displayed. If successful, **Click to see results** navigates to a different view showing the changes before and after the run. If ADM run fails, it is indicated with a different message and perhaps a reason.

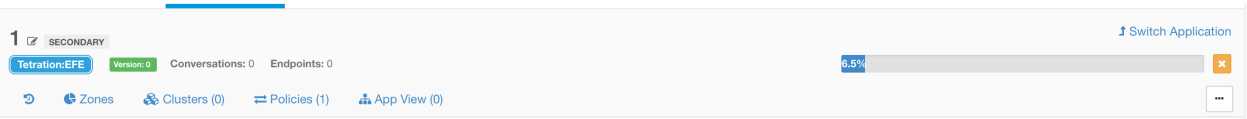


Fig. 6.14.1.3.1: ADM Run Progress

6.14.1.4 External Dependencies

External Dependencies configuration provides a powerful way to manage the granularity of the ADM generated policies to/from other applications.

ADM algorithms discover allow policies based on conversations among member workloads of an application as well as conversations of member workloads and other workloads that belong to other applications. Given an observation of communication to an external workload, users have a choice to direct ADM to generate specific or refined policies (more secure), or coarse policies to higher scopes, which may generalize better (i.e more likely to allow legitimate flows that were not seen in the time range of conversations given to ADM). Therefore the granularity of the policies generated by ADM algorithms can be fine tuned via the scope ranking in the External Dependencies configuration.

Given an external workload that is communicating with a member of the application, ADM *resolves* the external workload to the scope (or finer grain cluster/filter) based on the ordering specified in External Dependencies configuration. The first scope, cluster or custom filter, that matches the workload will be used to generate the allow policy, where the matching order is determined by the top-down ranking shown in the External Dependencies display. As a result, defining scopes to include the correct endpoints/workloads, as well as carefully configuring an appropriate ADM External Dependencies list is crucial for ADM to generate quality allow policies.

You can view the ranked list of all scopes (from the same tenant) in the External Dependencies list:

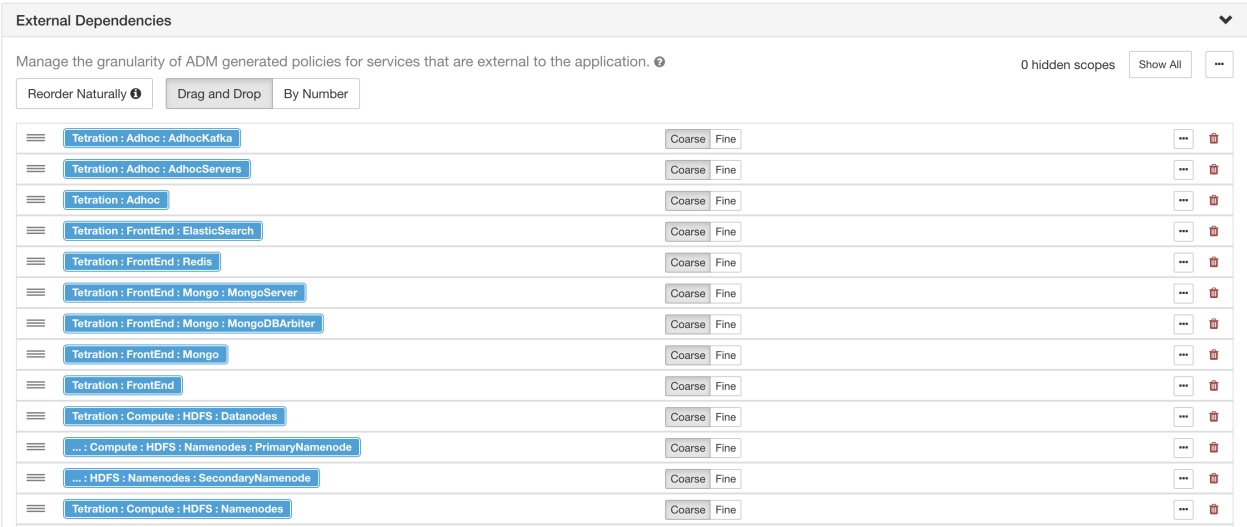


Fig. 6.14.1.4.1: Default External Dependencies

You can remove and rearrange the list to generate polices at a desired granularity. For example, removing all Company:RTP sub-scopes will help generate wide policies to the whole Company:RTP scope, but not its individual components, while maintaining the higher granularity for Company:SJC scope. Furthermore, you can click on the **Fine** button next to any scope and see if there are finer grain candidates defined under that scope. There is also an option to **Reorder Naturally** which will order the external dependencies in a child-first, post-order manner. This is useful when new child scopes are created by a user, which by default are added to the bottom of the list. The order may also be changed via the **Drag and Drop** option or via the **By Number** option. In the By Number option, the external dependencies will be assigned priority values in multiples of 10. These priorities can be adjusted with values and changes the order. Once numbers are modified, click **View** to update the list order and reassign multiples of 10 to each of the priorities.

Note that on an ADM run, you can reuse the changes you made to the list on the last ADM run by clicking on "Previous Config" on top right of the list. You can also make a single global list, available for all application workspaces, by going to the main Segmentation page, and clicking on "Default ADM Run Config" on the right. Later in any workspace, click on the "Default Config" button, next to "Previous Config" button, to use that default global list on every subsequent ADM run. Or, after obtaining the default list, you can modify it as desired (for that workspace only), and then use the customized version on subsequent runs by clicking "Previous Config" once.

TIPS: By default, the root scope is configured as the lowest entry in the External Dependencies list, so that ADM always generates policies to more specific scopes whenever possible. Initially, to view relatively few coarse-grained

policies, the user can place the root scope on the top of external dependencies (via drag and drop or via numbering). This way, after an ADM run, the user will see all external policies of the application connecting to only one scope, the root scope (as every external workload maps to the root scope). The resulting number of generated policies will be smaller and easier to examine and comprehend. Furthermore, the user can also bundle the internal workloads, i.e. all workloads of the application, into one cluster, approve the cluster and run ADM. Again, this results in a reduced set of policies, as no clustering (sub-partitioning of the application/scope) takes place, so the user can view policies that are either internal (connect to internal workloads), or external (connect an internal to an external workload). Subsequently, the user can view progressively more refined policies by unbundling internal workloads and/or placing one or a few external scopes of interest above the root. The user should examine the ADM generated policies carefully, when policies involving root scope is created. Since it will essentially allow all traffic to or from the entire networks. It is especially important, when the rootscope is placed low in the External Dependencies list and it is not the user's intention to generate coarse policies. Such policies may **not** have been resulted from some network-wide application traffic in or out of the workspace scope. Rather they can be triggered by a few external endpoints who failed to receive finer scopes or inventory filters assignments beyond simply the rootscope. While auditing these policies, the user should examine the associated conversations (See *Conversations*) to identify these endpoints and subsequently categorize them into finer scopes or inventory filters, in order to avoid loose root scope level policies.

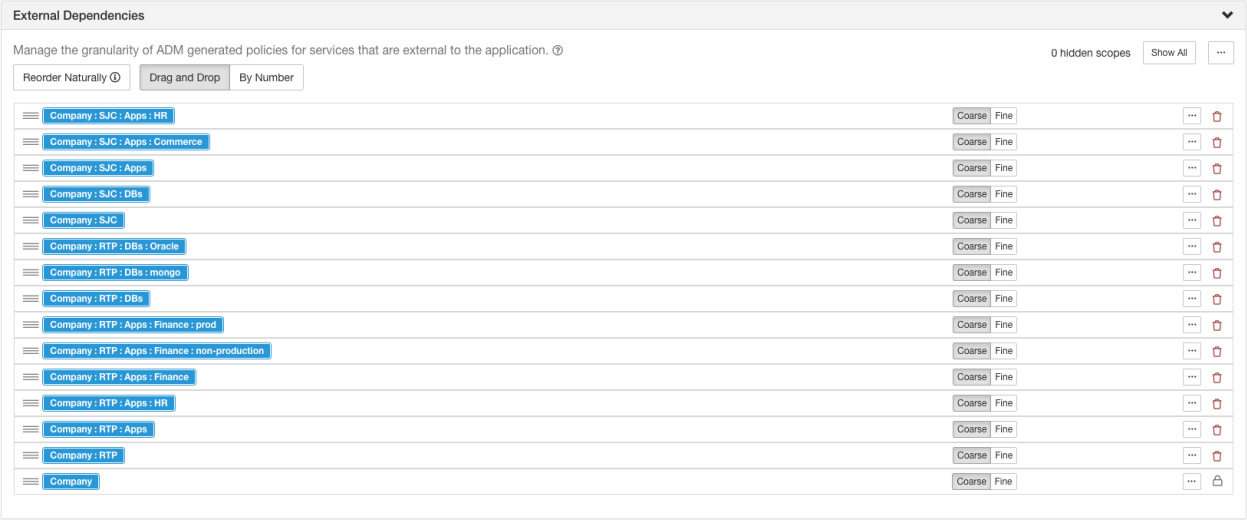


Fig. 6.14.1.4.2: Reorder naturally

Note: Only Inventory Filters that are restricted to a scope and marked as **providing a service** can be used for fine-grained external policy generation. See *Collaboration Among Applications*.

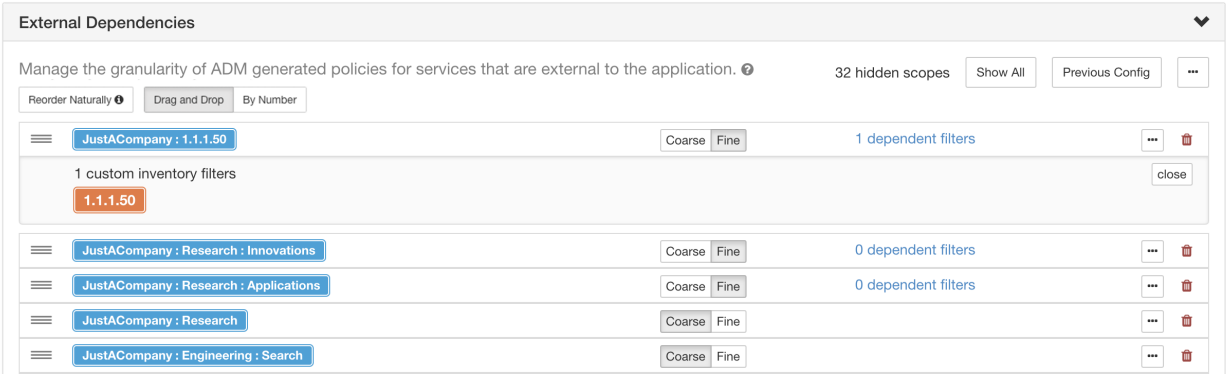


Fig. 6.14.1.4.3: Fine-tuning External Dependencies

6.14.1.5 Advanced ADM Run Configurations

Advanced run config allows us to upload and select additional side information to be used in conjunction with other realtime metrics for ADM analysis. Extra controls are also provided for advanced users to help the ADM algorithms adapt to a particular environments requirements, as described below.

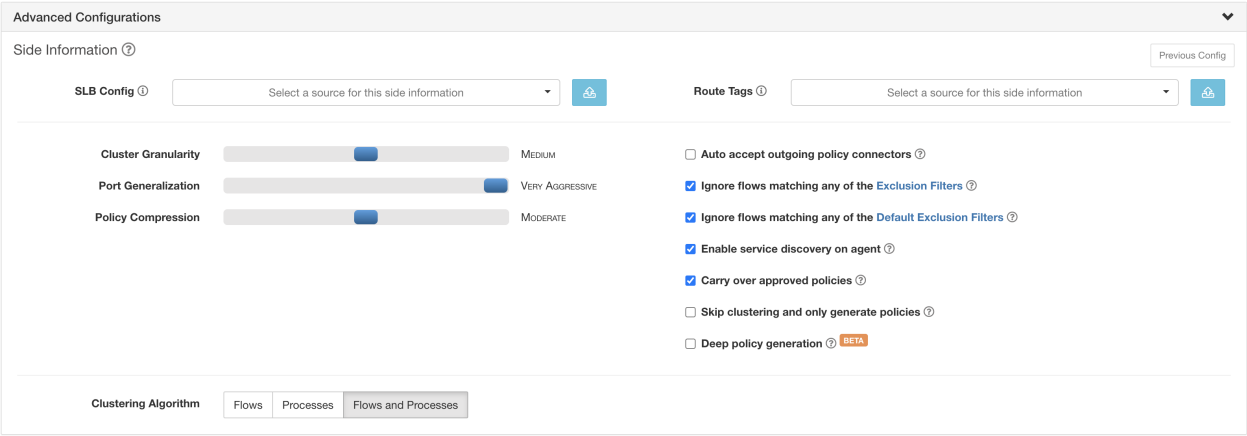


Fig. 6.14.1.5.1: Advanced ADM Run Configurations

Side Information

The following table describes the three types of side information, that is supported currently:

Currently Supported Side Information

Side Information	Description
Load balancer (SLB) configurations	Uploading loadbalancer config is allowed in three formats such as **F5 BigIP**, **Citrix Netscaler**, **HAProxy** and **Normalized JSON**. **Normalized JSON** is a simple schema with basic information on Virtual IPs (VIP) and backend IPs. It is the responsibility of the user to convert any unsupported load balancer config into the normalized schema. See *Retrieving LoadBalancer Configurations* for more info.
Route Labels	List of provisioned subnets/routes from the routers to help partition hosts based on pre-provisioned set of subnets. The clustering results generated by ADM algorithm never spans the subnet boundaries as defined by the sideinfo. The results can be modified by the user after the ADM run is complete.

NOTES:

- Click on the **i** button to download an example sideinfo file in JSON format. Additionally, you can click on the **download** or **trash** icon next to each row inside the dropdown menu, to view or delete previously uploaded sideinfo.

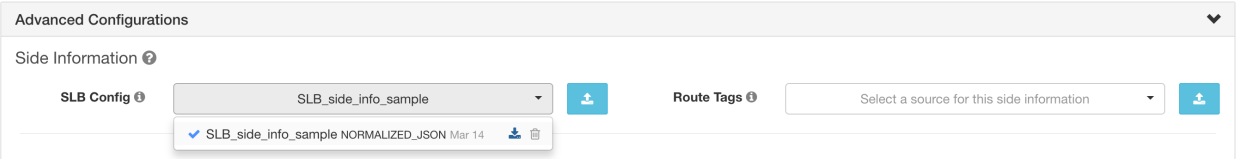


Fig. 6.14.1.5.2: **Side Information**

- Clusters do not span partition boundaries, meaning a cluster computed by ADM does not contain target workloads from two different partitions. Partition are computed from the uploaded side information (SLB, Routes, etc). However, the user can freely move targets from one cluster to another, eg via changing cluster query definitions (manual cluster editing), or disable the upload of any side info.

Clustering Granularity

Clustering Granularity allows the user to have a control on the size of the generated clusters by ADM algorithms. **Fine** results in more but smaller clusters, and **Coarse** results in fewer but larger clusters.

NOTE: You may not observe a significant change in the results due to many other signals that our algorithms take into account. For example, if there is a very high confidence in the generated clusters, changing this control will make little change in the results.

Input to Clustering

Advanced user can choose the main source of data for clustering algorithms, that is, live network flows, or running processes, or both.

Port Generalization

This knob controls the level of statistical significance required when performing port generalization, i.e., replacing numerous ports, being used as server ports on a single workload, with a port interval (see *ADM Concepts* for more on the semantics of port generalization). In the extreme left, port generalization is disabled. Note that if disabled, the ADM run time and/or ADM UI rendering time may be slowed substantially, in case many server ports are used by the workloads. As the knob is placed to the right to the more aggressive generalization settings, less evidence is required to create port-intervals and also the criterion for replacing original policies (involving single ports) with port-intervals is relaxed.

Carry over Approved Policies

When this flag is set, all the policies that are marked as approved by the user via UI or OpenAPI will be preserved. This helps users to not have to re-define a particular broad DENY rule that should take effect regardless of the allow policies that are discovered by ADM algorithms.

Enable service discovery on agent

When this flag is set, ephemeral port-range information regarding services present on the agent node are reported. Policies are then generated based on the reported port-range information.

Example:

- Windows Active Directory Domain Server uses default Windows ephemeral port-range **49152-65535** to serve few requests. When this flag is set this port range information is reported by the agent and policies are generated based on this information.

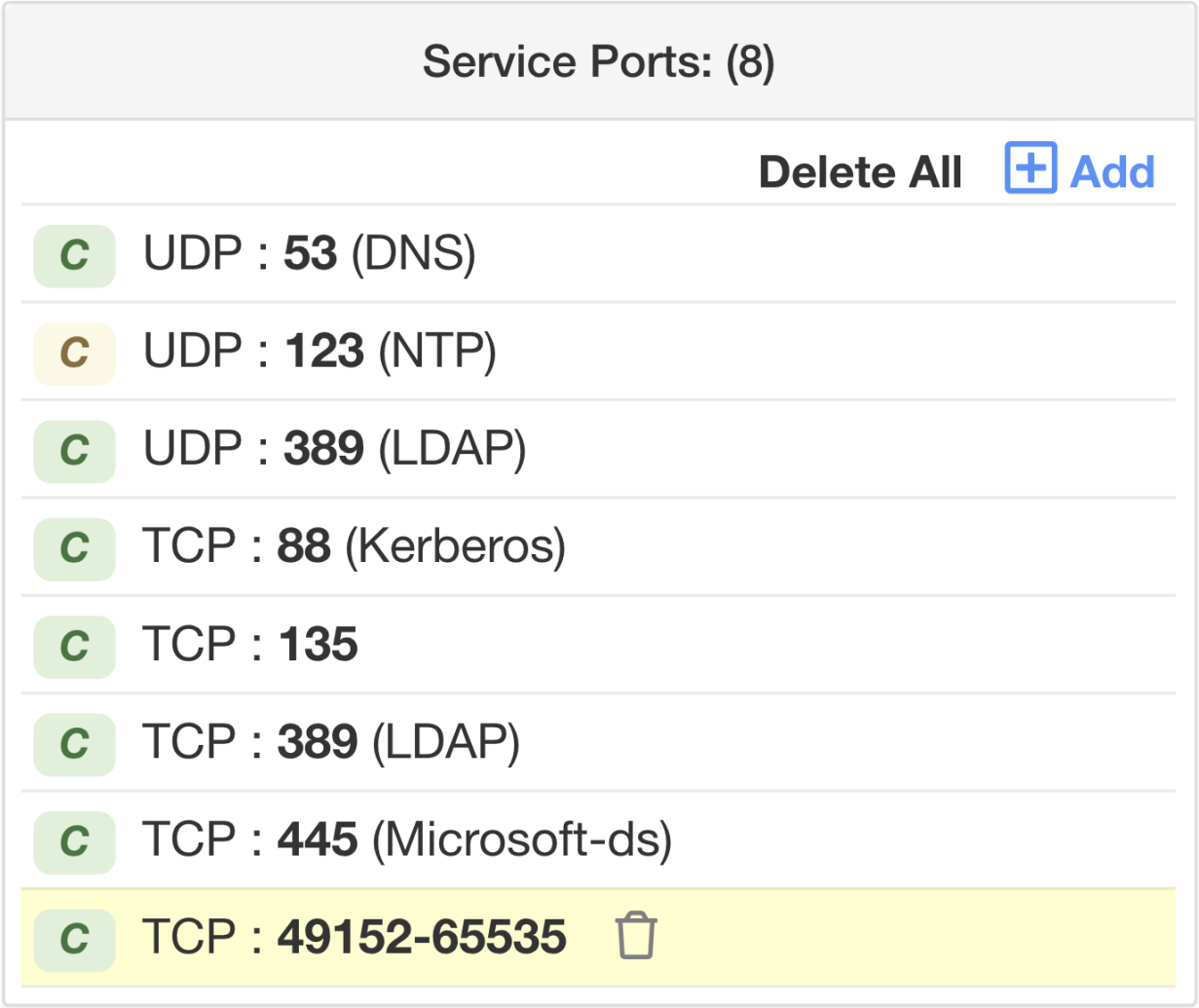


Fig. 6.14.1.5.3: Service discovery enabled on the agent

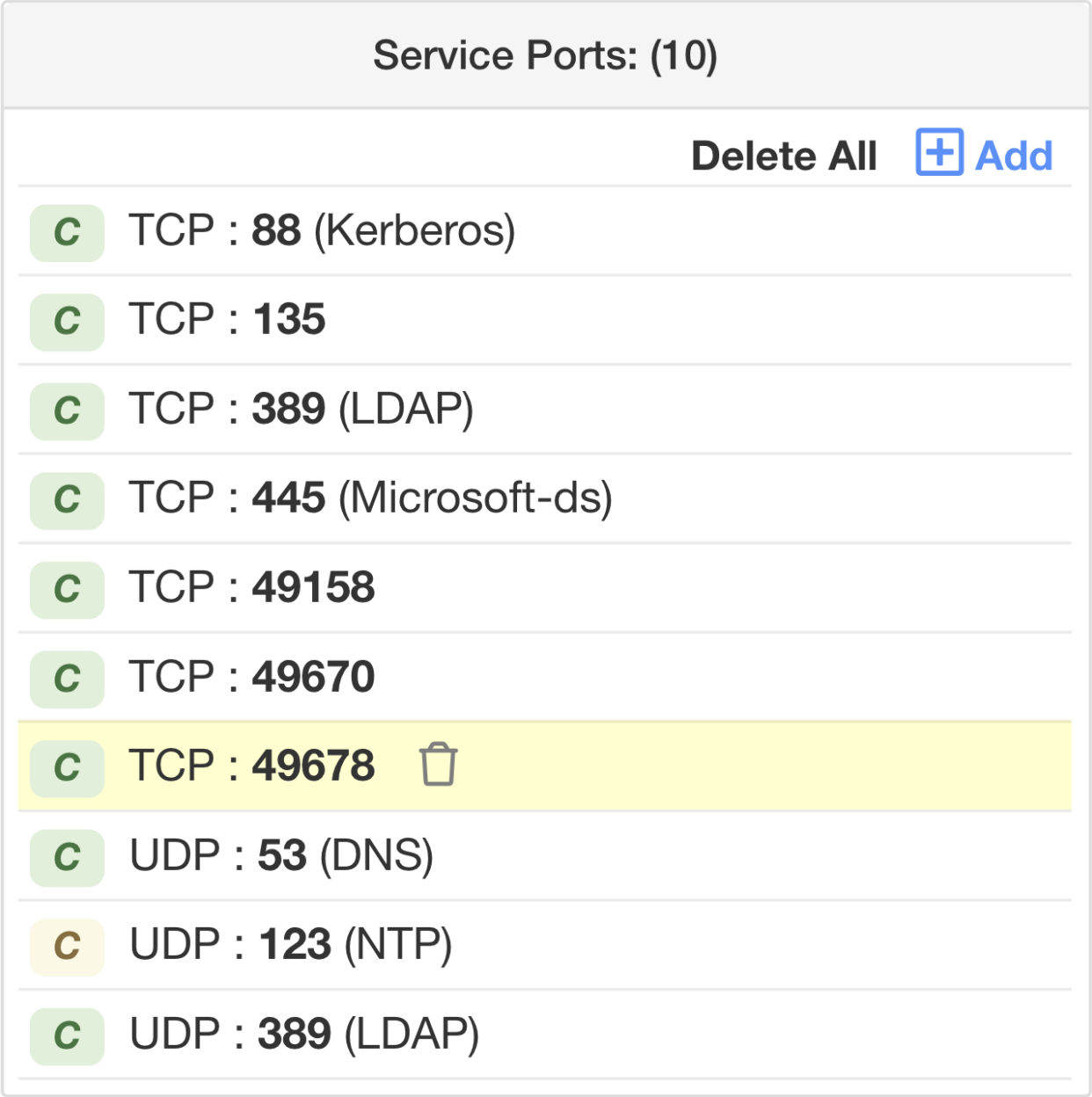


Fig. 6.14.1.5.4: Service discovery not enabled on the agent

Policy Compression

When policy compression is enabled, policies that are sufficiently frequent, i.e. they use the same provider port, among the generated clusters inside a workspace may be 'factored out' to the parent, that is, replaced with one or more policies applicable to the entire parent scope. For example, if all or almost all clusters in the workspace provide the same port to the same consumer, then all those policies are replaced with one policy from the parent scope, meaning that the parent scope is allowed to provide the consumer on that port. So policy compression can reduce the number of policies significantly and reduce clutter, and it may also lead to allowing legitimate future flows that could have been dropped (accurate generalization). The more aggressive the compression knob setting, the smaller is the required threshold on policy frequency in order to replace with a parent policy.

With Deep policy generation option selected:

This knob can be used to alter the level of aggressiveness in *Hierarchical policy compression*.

Note: Currently, the ADM conversations page does not support showing the conversations that led to a compressed policy (the user may need to disable compression or use flow search).

Skip clustering and only generate policies

No new clusters are generated, and policies are generated from any existing approved clusters or inventory filters and otherwise involve the entire application scope (in effect, treating the entire scope as a single cluster). This option can result in substantially fewer (but coarser) policies.

Deep policy generation

This option is useful specially when one is interested in global policy generation (for example, at a single scope, or a few scopes, at or near the top of the scope hierarchy), and for generating coarse policies among scopes. When this option is selected note that the limits are increased to a maximum of 25000 scope workloads (see below) and the number of conversations to 20 million.

In this mode, only policies among the scopes of the scope tree are generated (clustering is skipped). For generating policies, two aspects need to be addressed: 1) the set of conversations used for policy generation, and 2) the (scope) label that each end of the conversation is assigned.

All conversations where at least one endpoint is a target endpoint are used for policy generation (unless, of course, the conversation is excluded by a filter). However, to allow policy generation for an entire subtree of scopes, the definition of target endpoint is relaxed and is different from classic ADM runs: an endpoint is a target endpoint here if it is in the scope of the application (matches the scope query), **irrespective** of whether the endpoint also belongs to a subscope. Note that in the typical/classic ADM run (to fascilitate RBAC), an endpoint is NOT considered a target endpoint if it's also claimed by (matches) a subscope. With this relaxation, one can generate policies for an entire subtree.

For the 2nd aspect, all endpoints, whether target or not, are assigned the highest matching scope label according to the top-down order given in the external dependencies list. Thus policies generated may involve scopes at various levels of the scopes tree (to the desired granularity). All the policies generated will reside in the workspace in which the deep policy generation is issued (even if the policy involves only sub-scopes or ancestor scopes).

Note: This option is only available for root scope owners.

Note:

Currently, the number of workloads shown in ADM UI is count of those not claimed by a subscope, which is useful for standard ADM runs, and thus may be lower than the total number of target workloads on which deep policy generation is applied (see description above).

Hierarchical policy compression

Policy compression can also be done for *deep policy generation*. The *policy compression* knob can be used to alter the level of aggressiveness in hierarchical policy compression. An example of hierarchical policy compression is illustrated below.

- Let A, B, C and D be scopes part of a scope tree, where "C" and "B" are the child scopes of "B". Let "C" → "A" be a TCP "ALLOW" policy on port 5520 and "D" → "A" be TCP "ALLOW" policy on port 5520.

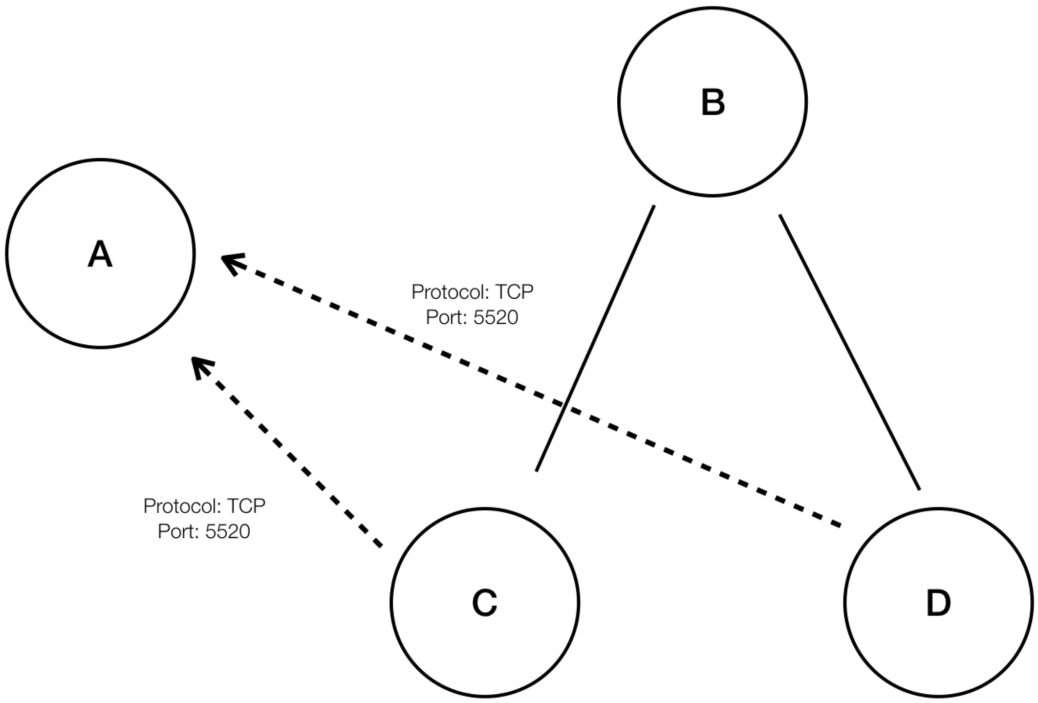


Fig. 6.14.1.5.5: Before hierarchical policy compression

- With hierarchical policy compression if a sufficiently large group child scopes involves in policies sharing the same port, protocol and destination or source, these policies will be replaced by a generalized policy that connects the parent scope to the common source or destination. In the above mentioned case "C" and "D" are child scopes of "B" and the policies "C" → "A" and "D" → "A" share the same destination, port and protocol. Since 100% of child scopes of "B" contain the similar policy the policy will be promoted to be "B" → "A", resulting in the following. Furthermore, hierarchical compression can be repeated so a generalized policy can go all the way to the root of the subtree on which deep policy generation is invoked.

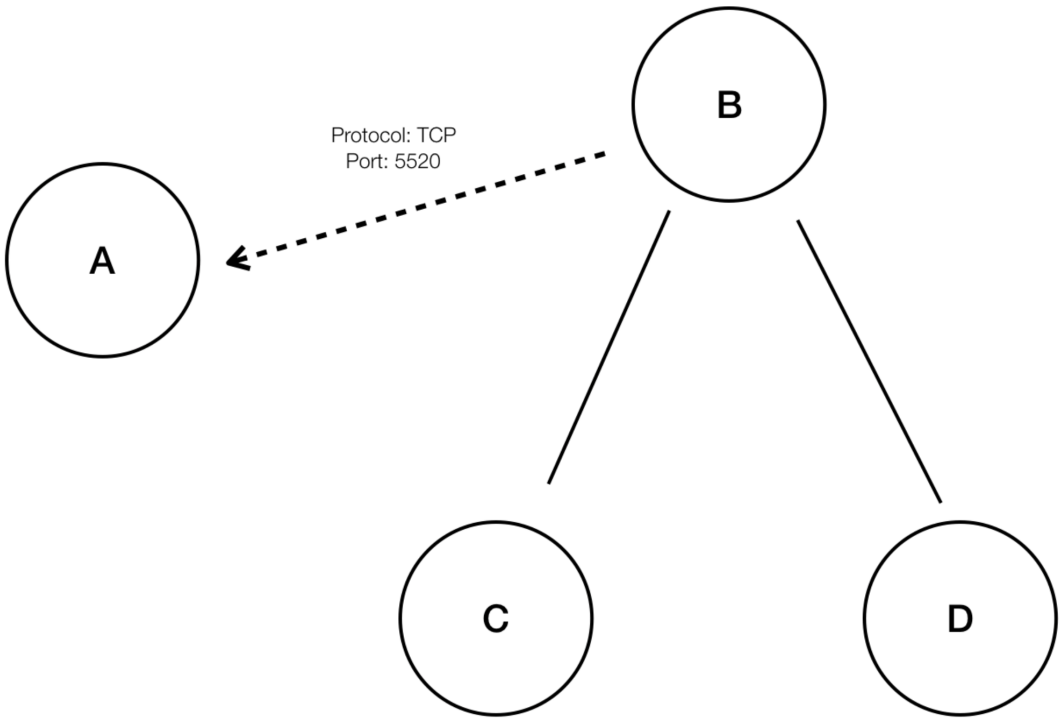


Fig. 6.14.1.5.6: After hierarchical policy compression

- The policy compression knob allows the user to tune the aggressiveness of such compression, by changing the minimum required proportion of the policy-sharing child scopes (usually measured as the fraction of total number of child scopes) to trigger the compression. When disabled, each policy is generated between highest priority scopes based on the External Dependencies list. Subsequently, if the user chooses to impose the naturally ordered External Dependencies list, the policies generated will be the most granular policies among scopes.

Enable redundant policy removal

This option is only available when *Deep policy generation* is selected.

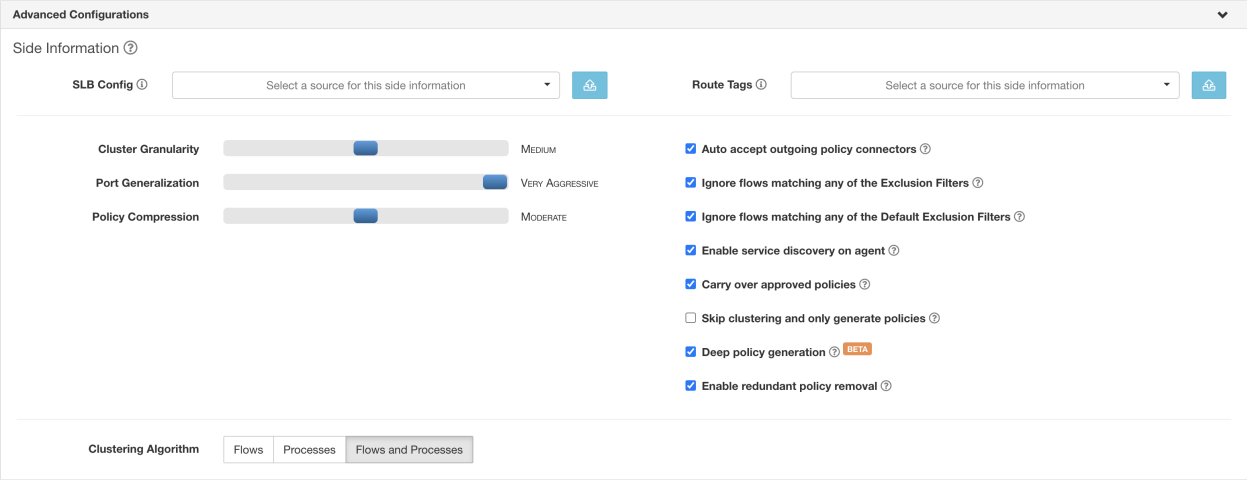


Fig. 6.14.1.5.7: When Deep policy generation option is selected

This option enables/disables removal of redundant granular policies.

Example:

- Let Root, A, B, C, A1 and A2 be scopes part of a scope tree. Let the following be the policies:
  1. "Root" → "Root"
  2. "B" → "Root"
  3. "C" → "Root"
  4. "A1" → "Root"

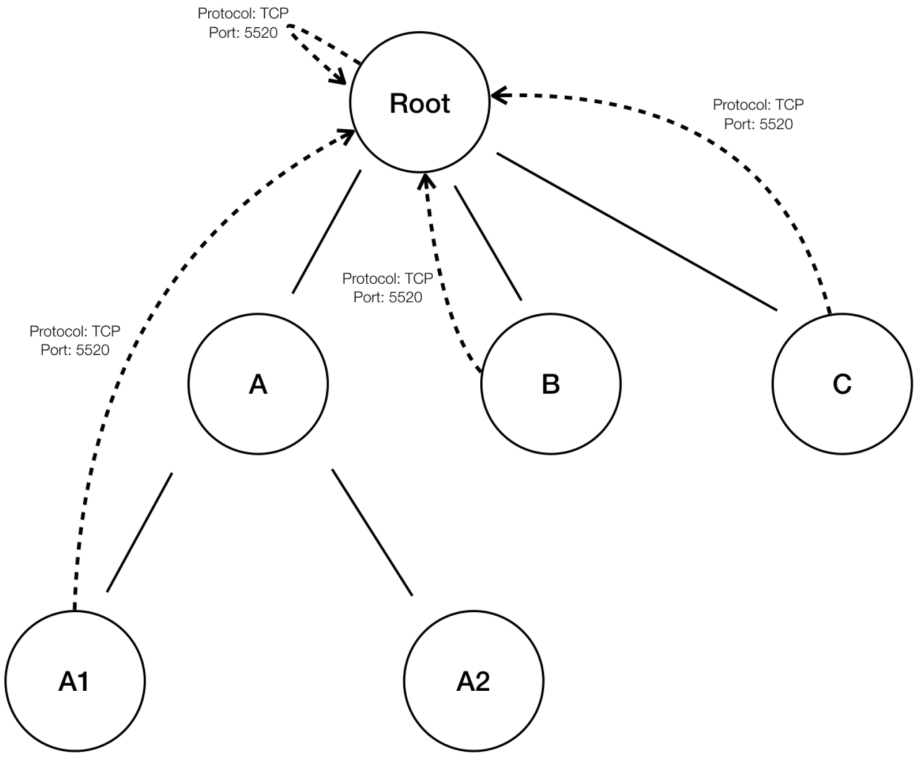


Fig. 6.14.1.5.8: Before removal of redundant policies

- The policies "B" → "Root", "C" → "Root" and "A1" → "Root" are redundant as the policy "Root" → "Root" covers these policies. The remove redundant policies feature will check and remove such policies resulting in only one policy "Root" → "Root" as follows.

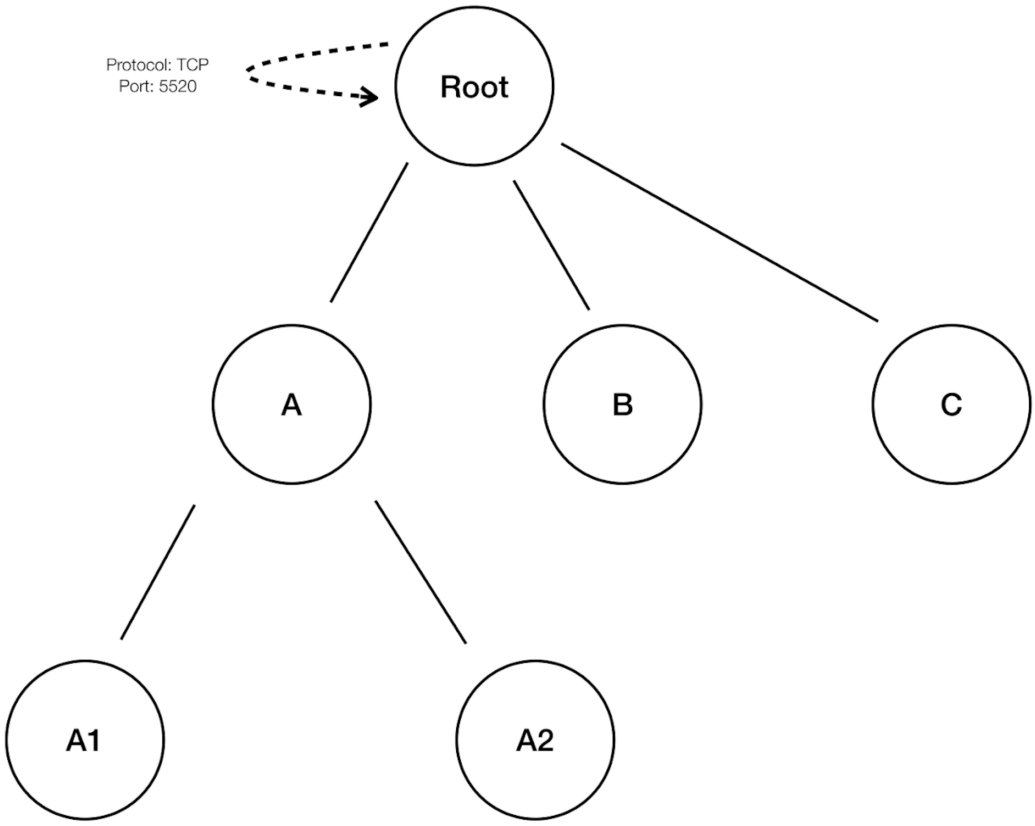


Fig. 6.14.1.5.9: After removal of redundant policies

Redundant policy removal can be very useful in maintaining a succinct set of interpretable policies. The reduced policy set contains the minimal number of policies at the chosen compression level to cover all the workload traffic. However, the user should always audit the policy through policy analysis and examine the corresponding conversations to evaluate the tightness of the resulting policies. This is especially important when there exists traffic to or from endpoints that are not categorized into finer scopes or inventory filters. Such endpoints may trigger the generation of coarser policies than intended, such as policies involving the rootscope. If at the same time, redundant policy removal is enabled, more granular policies will be removed and will not be presented to the user. To diagnose the source of (compressed) policies and to view finer level policies, turn off policy compression and redundant policy removal. Also note that currently in this release, the ADM conversations page may fail to show the conversations that lead to a compressed/generalized policy, so to get around this, one can turn off compression and redundant policy removal, so the one can easier find the conversations that lead to the generated policies.

TIPS Since deep policy generation discovers all policies for the scope subtree rooted at the workspace scope, these policies will cover all the legal traffic seen by ADM for all the workloads under the subtree. When analyzing these policies using tools such as Policy Analysis (See *Policies*), the user is advised to turn off Policy Analysis in all the workspaces associated with the subscopes. This way, the policies (if any) residing in the subscope workspaces (usually receive a high priority due to more specific scope definition) will not take priority and interfere with the results. However, exceptions apply when the policies in the subscope workspaces are configured to cover different sets of traffic that usually involve finer inventory filters or clusters specific to the subscopes.

Auto accept outgoing policy connectors

Any outgoing policy requests created during the ADM run will be auto accepted. If this option is selected as part of the Default ADM run config policy requests created manually will be auto accepted as well. See *policy requests* for more info.

Note: This option is only available for root scope owners.

Exclusion Filters

This option provides the flexibility to ignore all conversations matching any of the user defined exclusion filters (if any). This is particularly useful when ADM run is automatically generating allow policies for an undesired set of flows. Using this option you can guide the algorithms to ignore certain kinds of flows. Click on the **Exclusion Filters** link to navigate to the exclusions filter configuration page, where you can add/delete and update the filters using subnet, port and protocol filters. See *Exclusion Filters* for more info.

6.14.2 Retrieving LoadBalancer Configurations

Below are the instructions for retrieving supported load balancer configuration files in a format that can be directly uploaded by Tetration ADM tool. Note that all files must be encoded as ASCII.

6.14.2.1 Citrix Netscaler

Concatenate the output of `show run` in your console and upload the file to the tool.

See Sample config file

6.14.2.2 F5 BigIP

Upload the `bigip.conf` file to the tool. If you have a file with a `.UCS` extension, please untar the archive and upload only the `bigip.conf` file within the configuration dump. If there are multiple files, concatenate them and upload.

See Sample config file

6.14.2.3 HAProxy

Upload your `haproxy.cfg` file to the tool. The path is typically `/etc/haproxy/haproxy.cfg`.

See Sample config file

6.14.2.4 Normalized JSON

If you find the above options limiting, please convert your configs to the following JSON schema and upload them directly to the tool. The example JSON file can be directly downloaded by clicking the **i** icon next to SLB Config in Advanced Run Configurations.

See Sample config file

6.14.3 Exclusion Filters

Exclusion Filters help you fine-tune ADM run results and policy generation by excluding certain flows from the ADM run input. This results in different allow policies and possibly different clustering results (Note: all conversations remain viewable in the Conversations View). For example, in order to disallow certain protocols like ICMP in the final allow list model, you just need to create one exclusion rule with a protocol field set to ICMP.

Exclusion filters can be created automatically whenever a policy is deleted (the choice is given to the user). Or they can be created manually in the Exclusion Filters page. One way to access the filters page, to view or create filters, is to click on top right of any ADM page on the '. . . ' icon (by the Enforcement icon) and select exclusion filters. On the ADM run page under Advanced Configurations, one can also click on the exclusion filters link. Note that there is a limit of 100 exclusion filters per workspace.

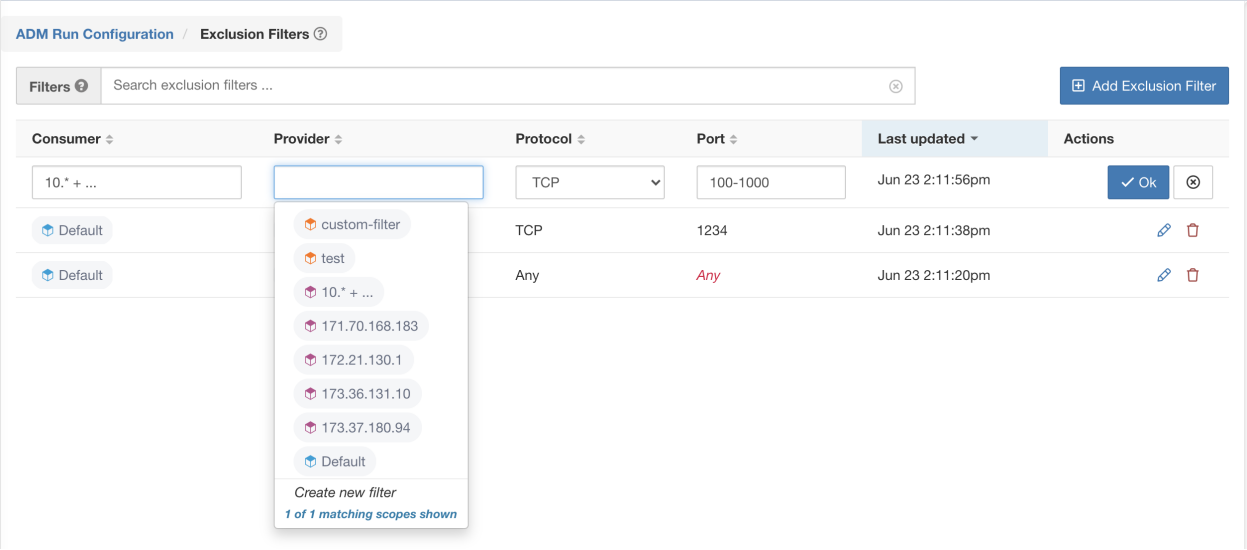


Fig. 6.14.3.1: Exclusion Filters Workflow

Once on the page, click on the **Create Exclusion Filter** button to add a new filter to the table. There are four fields to configure, but they are not all required. Any empty field will be treated as a wildcard for matching flows. The available fields are:

- **Consumer**: Matches conversations where the consumer address is a member of the selected cluster/filter/scope. You can specify any arbitrary address space by creating a new custom filter.
- **Provider**: Matches conversations where the provider address is a member of the selected cluster/filter/scope. You can specify any arbitrary address space by creating a new custom filter.
- **Protocol**: Matches conversations with specified protocol.
- **Port**: Matches conversations with provider (server) port matching the specified port, or port range. Port ranges can be defined using a dash separator, e.g. "100-200"

Any conversation that matches all the fields of any exclusion filter will be discarded for the purposes of policy creation and clustering. Click on the **Edit** button to change an existing exclusion filter, and the **delete** button to delete one. These buttons are only visible when the row is hovered by the mouse pointer.

You can make a single global Exclusion filters list available for all application workspaces within a tenant. You can configure these "Default Exclusion Filters" by navigating to "Default ADM Run Config" under the main segmen-

tation page. This list can be used in combination with the workspace specific Exclusion Filters list in "Advanced Configurations". The interface for managing each list is the same.

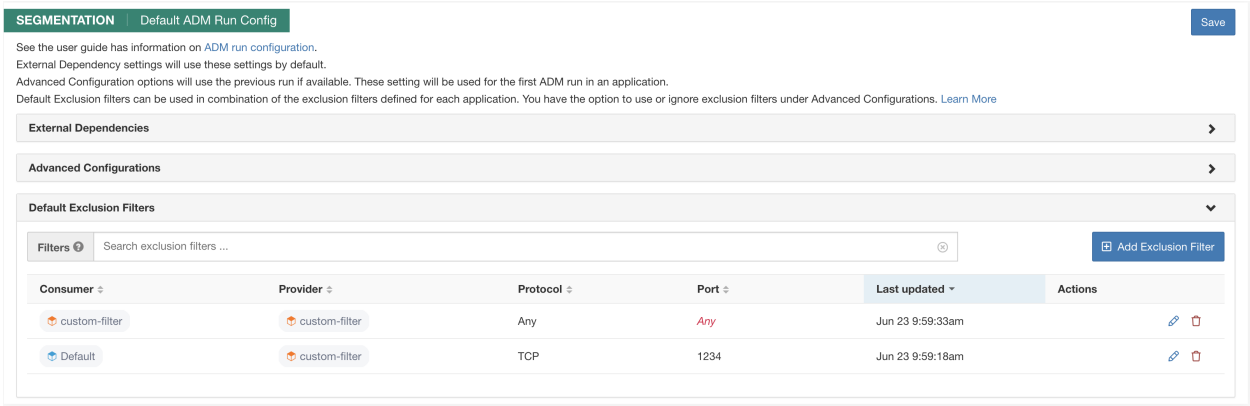


Fig. 6.14.3.2: Default Exclusion Filters

Notes:

1. Make sure any scope changes are committed before the ADM run, otherwise the filters may not match (exclude) the flows. See commit scope updates.

2. Conversations that match exclusion filters are excluded for the purposes of policy generation and clustering, but are kept in the Conversations View with a red 'excluded' icon (shown for visibility, see Table View in *Conversations*). Likewise, workloads of the workspace incident on such conversations remain viewable as well.

3. An exclusion filter that uses a cluster or a filter definition from a workspace is effective currently only if the workspace is primary (otherwise, its cluster definitions are not visible to the label system, and any matching conversations are not excluded).

4. Modifications to exclusion rules are trackable via the history (see *History & Diff*). However, in releases prior to 3.0, the exclusion rules themselves were not scoped by workspace versions. Therefore, later editions to exclusion filters would be available even when switched back to an older revision of the workspace. Starting with 3.0, exclusion filters are also versioned.

6.14.4 Re-running ADM Algorithms

At any point during the lifespan of ADM workspace, you can rerun ADM algorithms by navigating to ADM run configuration page. The main reason to rerun algorithms is to include additional information that was not initially taken into account in the previous run. For example, one might:

1. Increase the timespan of flows used to generate ADM clusters and policies.

2. Change side information or other run configurations.

3. Edit and approve a few clusters, which can improve the clustering of others upon rerun.

In order to trigger an ADM rerun, navigate to run configuration page, change configuration and click on **Submit ADM Run** button.

6.14.4.1 Effects of ADM re-run

Rerunning ADM on an existing workspace may change the contents of the clusters and policies in the workspace. If a host is no longer in the scope of the workspace, upon a subsequent ADM run, that host will not appear in any cluster: if it were in an approved cluster, it will no longer appear in that cluster. Even with the same set of member workloads but with a different timeframe or configuration, running the clustering algorithms may result in different clusters.

Application views (*App Views*) may also get affected by ADM reruns. In the event that content of a cluster changes due to an ADM run, our algorithms take a best effort approach to match the new clusters with the old ones. For example, if one or two members of a cluster with 10 workloads have changed, we consider it the same cluster and application view will remain unchanged. In this scenario the application view will refer to the new cluster and newly generated policies, not the old one as a reference for nodes and edges respectively. However, if the contents of a cluster is significantly changed, for example, a cluster of 10 workloads is split into two clusters of size 5, we consider the old cluster deleted and two new clusters added. In this case, the application view may not show the right graph, and needs to be edited by the user to reflect the correct set of dependencies.

There are use cases where an ADM rerun might be necessary, but the contents of certain clusters should not change. For example, users might have edited and fixed the contents and created application views, and now they need to add new targets to the workspace and cluster them without affecting the existing policies. In this case, the user has the option to **Approve** a cluster as shown below. Approving a cluster is like freezing the cluster contents and attributes in the current state. ADM algorithms always guarantee to keep the approved clusters intact.

NOTES:

- Approving clusters and rerunning ADM may improve the clustering of the remaining target workloads.

6.14.4.2 Approving Clusters

Make sure the cluster of interest is shown on the side panel. You can accomplish this via searching for the cluster, or clicking on the desired cluster on the chart in any of the views.

Then click on the **thumbs-up** icon on the top-right corner of the cluster info on the side panel as illustrated below. The icon will change color to indicate that the cluster is approved by the user and will be unchanged by ADM algorithms, You may remove the approval by clicking on the same icon again.

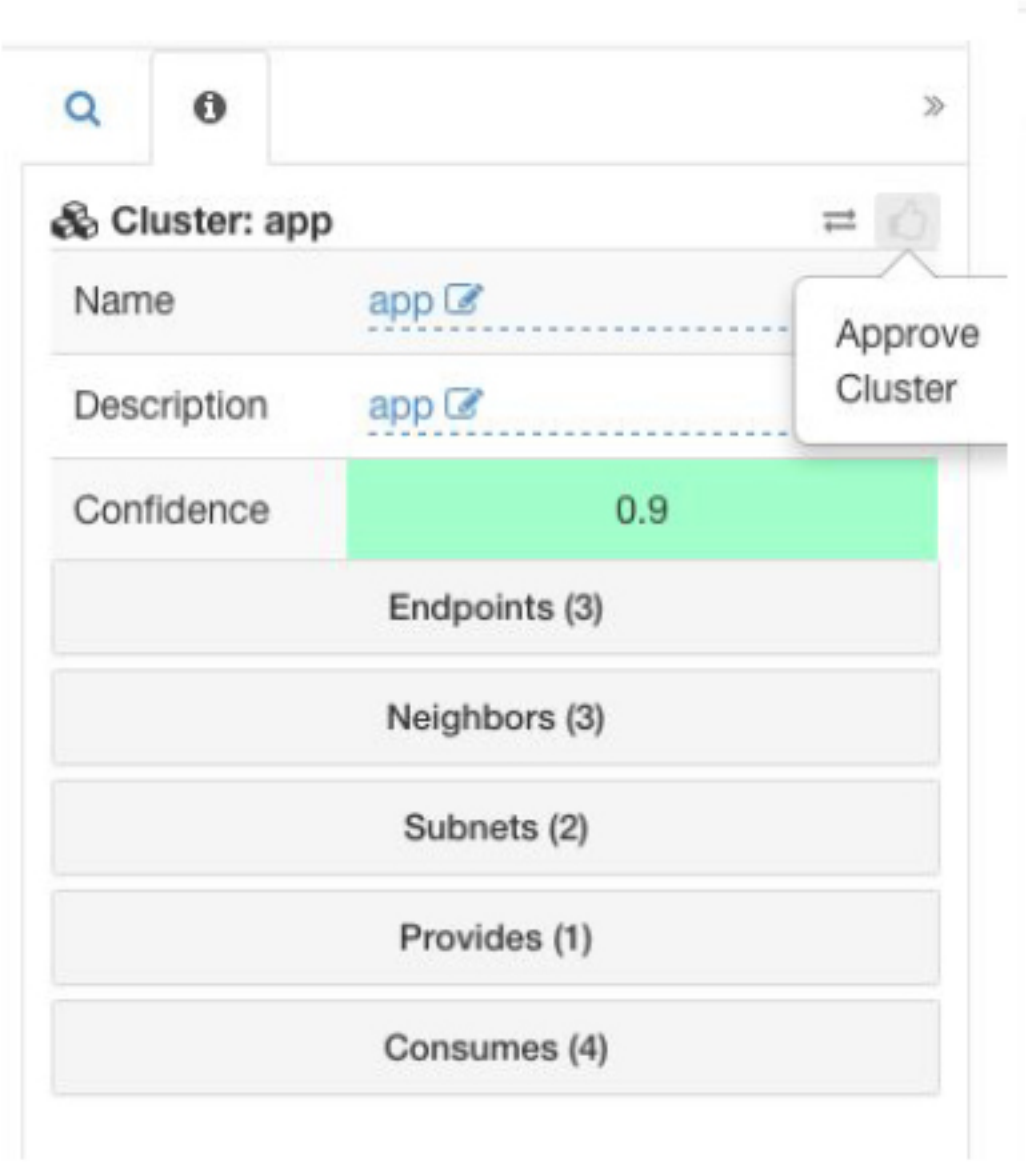


Fig. 6.14.4.2.1: Approving Clusters

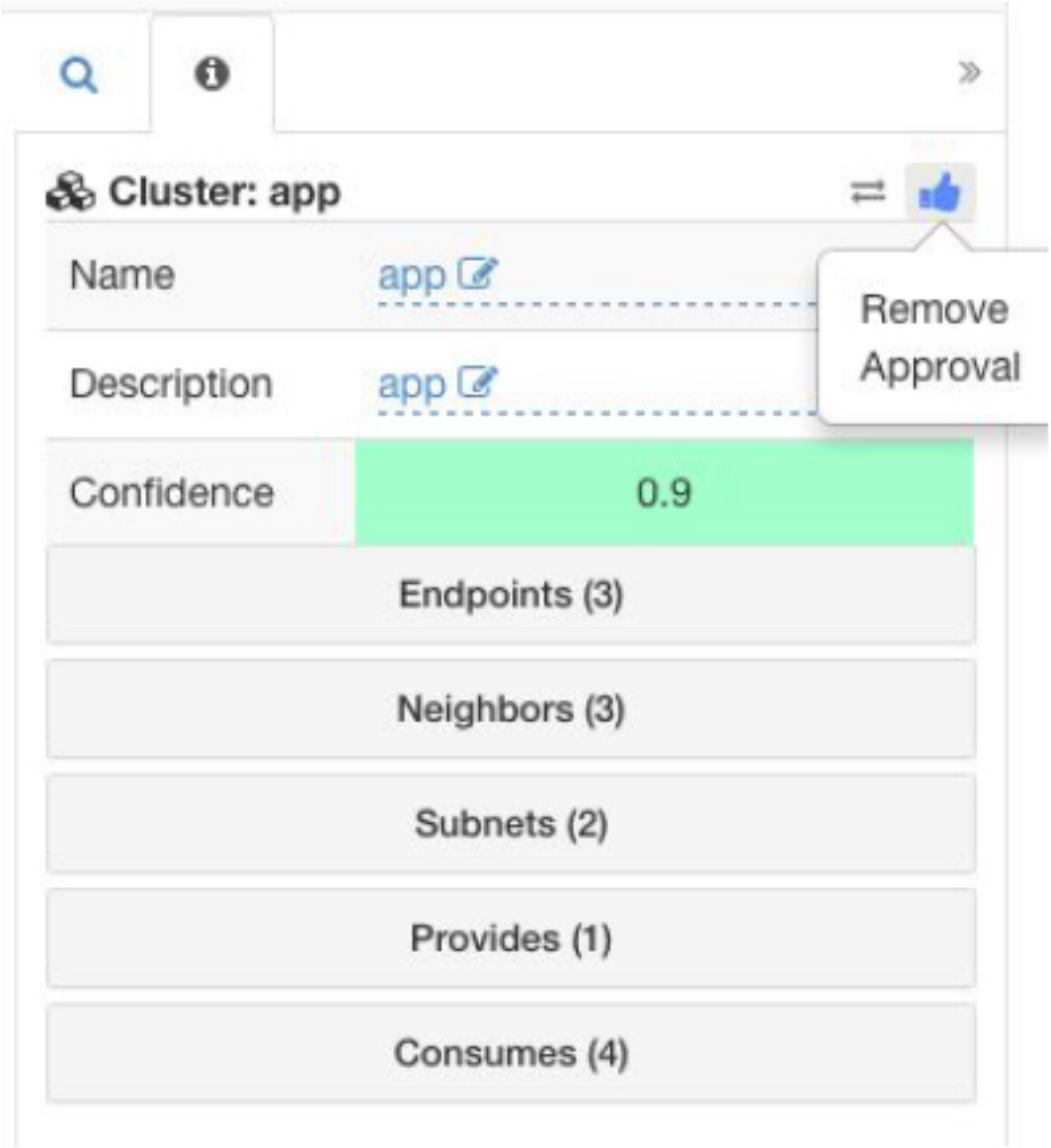


Fig. 6.14.4.2.2: Approving Clusters

6.15 Clusters

A cluster is a set of workloads (its members). The ADM clustering algorithm generates a partitional clustering of the non-approved workloads that belong to a workspace.

ADM algorithms try to find the best way to group workloads together based on the signals observed in the timeframe specified as part of run configurations. However, due to incomplete or conflicting information the results may not completely match the expectation of all users.

In the following sections, we describe a few workflows to edit, enhance and approve the clustering results. Note that one can change/approve clusters only in the latest version of a workspace (see *History & Diff*). Click on the

clusters box in the ADM header in order to browse and edit clusters. Note that approved clusters, or those promoted to inventory filters, are not changed upon ADM reruns.

6.15.1 Cluster Confidence

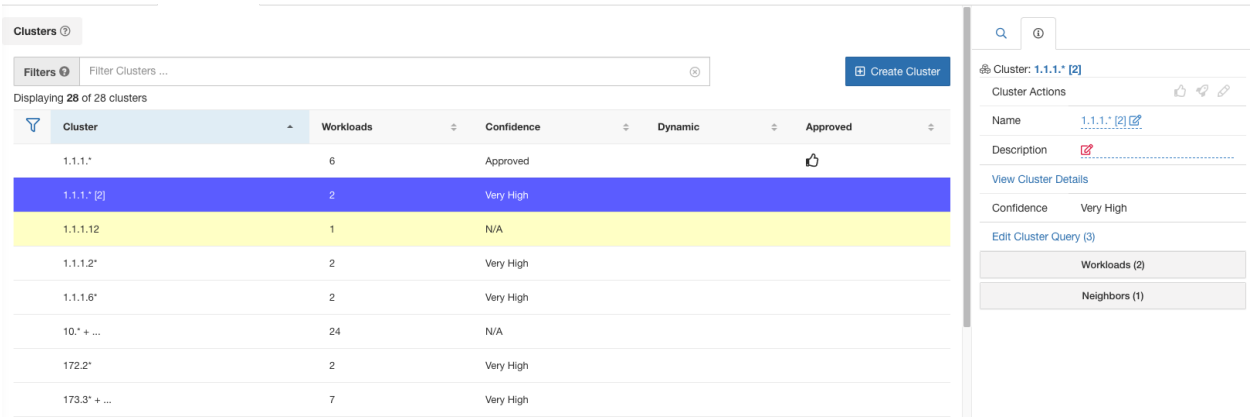
The confidence or quality score of a cluster, indicated by color, helps the user in assessing the quality of a cluster and thus indicate clusters that could be improved. The confidence for a cluster is the average of the confidences for member workloads. In general, the more similar a workload is to other members of the cluster it was assigned, and the more dissimilar it is to the workloads of the closest (most similar) alternative cluster the higher the confidence for that workload. When flows are used for clustering, two workloads are similar when they have a similar pattern of conversations (such as similar sets of neighbors in the conversation graph, i.e., similar sets of consumer and provider workloads and ports).

NOTES:

- The confidence is not computed (undefined) in several cases. It is not computed for singleton clusters (a cluster with one member), approved clusters, and target workloads for which no communication was observed (or no process information is available, if process-based clustering was chosen). In case of singleton clusters, similarity among workloads inside the cluster is undefined (this is required to compute confidence).
- Clusters do not span partition boundaries (such as subnet boundaries, see route labels in Advanced ADM Run Configurations). However, in computing confidence and alternate cluster, such boundaries are ignored. Rationale: this is to signal to the user the potential existence of workloads or clusters that behave very similarly even though they are in different subnets.
- After editing clusters, the confidence scores may become inaccurate as they are NOT recomputed (unless an ADM rerun is done).

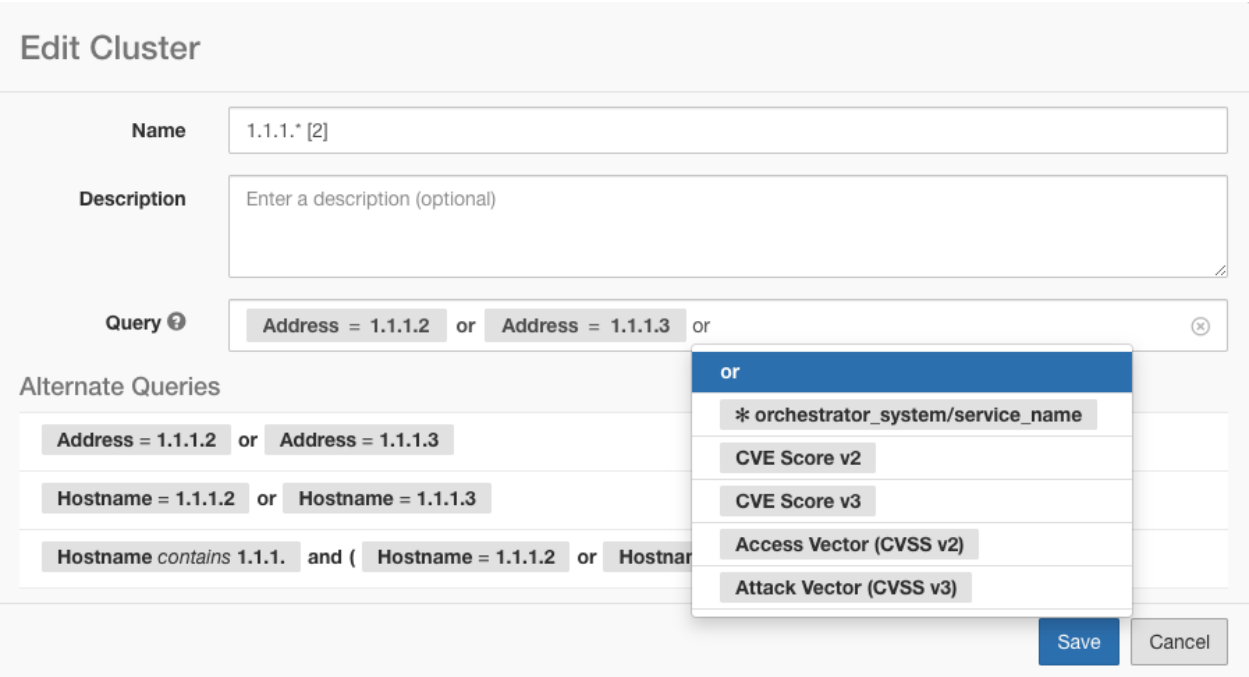
6.15.2 Dynamic Mode

If Dynamic Mode is selected for the workspace, one table of clusters is shown in the clusters view, where one can rank the clusters based on a column (such as name, the number of workloads, or confidence). For each cluster, by clicking on its row, the user can view further cluster information such as description, suggested or approved queries, and the member workloads in the right panel. Several of these fields are editable.



6.15.3 Making Changes to Clusters

In the Dynamic Mode, an ADM run creates one or more candidate queries for each cluster. To change a cluster (e.g. change the members of a cluster or select/change its query) user can select/edit the cluster's query, as shown below. One can add or remove explicit addresses, or pick another query from the list of alternatives provided and edit that query. A cluster's query can be any query filter expressed in terms of addresses, hostnames, and user uploaded labels. After query selection and possible editing is done, click save. Note that once the SAVE button is clicked, the cluster is automatically marked approved, the approved thumbs-up icon turns blue (whether or not a change was made). The approved icon can be toggled to change the approved status as desired. See *ADM Concepts* for the semantics of approved clusters.



NOTE: When a cluster's membership is changed, a rerun of ADM may be necessary to get an updated policy accurately reflecting the changes in flows among the changed clusters. This is because cluster memberships may have changed (such as new nodes added to a cluster). A similar situation can occur if the scope corresponding to the workspace is edited or in general when workspace membership changes. Similarly, cluster confidence scores may no longer be accurate with changes to cluster memberships. In all these cases, an ADM rerun is useful to get updated policies and cluster confidence scores (updated confidences on unapproved clusters).

6.15.4 Creating or Deleting Clusters

Click the **Create Cluster** button on the clusters page to create a new empty cluster. Alternatively, you can also create a cluster from the new ADM landing page by clicking on **Create Filter** button in Get Started sidebar and selecting Clusters in the modal.

Fig. 6.15.4.1: Creating a new Cluster

The new user defined cluster will show up on the side panel to be renamed, if necessary.

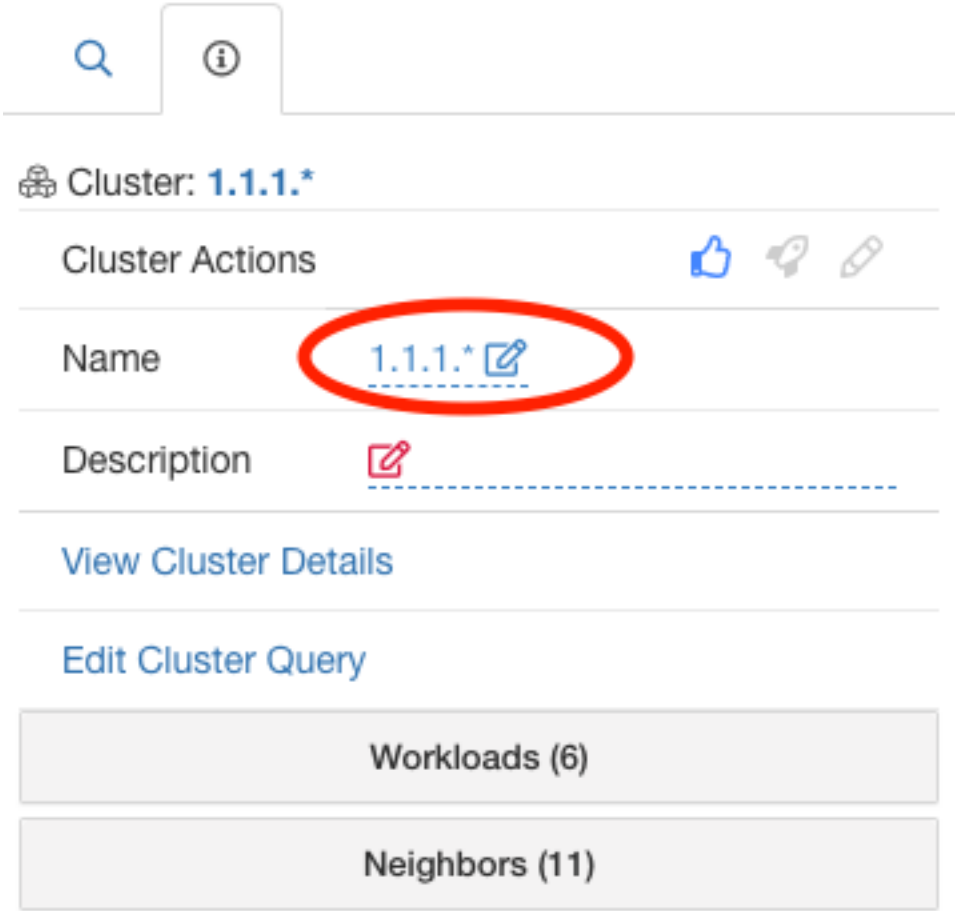


Fig. 6.15.4.2: Renaming a Cluster

An empty cluster may be deleted by selecting the cluster in any of the views so that the details show up on the side panel and clicking the trash button on the header of cluster detail view. See figure above.

6.16 Policies

Network security policies are the building block for many powerful features of Cisco Tetration. They provide a simple and intuitive mechanism for both application owners and security teams to define the necessary intents to secure assets and applications within datacenters.

Notes: The quality of policies depend on the quality of input data. Our algorithms work better with certain sensors (which provide more signals) than others. It is strongly recommended to use software sensors deployed in the workload in order to get the most robust policies. The workload is the best place of visibility as the ADM algorithms can work both on the flow and process data to perform clustering and to generate policies. Even while using just flow data for ADM runs, process info is still available to the user enabling them to better understand what flows are associated

with what process while evaluating ADM clusters. In addition, information collected by software sensors provides visibility of unused L4 ports. Unused ports are the ones for which no communication was seen for the interval over which an ADM run was selected. This information can be used to open up policies for communication on those ports OR to close those applications binding to the unused ports, thereby reducing the attack surface of the workload.

Policies rely on client/server, or the flow direction, being correctly identified, and we use different techniques to determine flow directions. See *Client Server Classification* for further details. In some limited cases, flow direction classification can be incorrect which may impact the generated policy or ADM results. A confidence indicator is provided for each policy, and policies can be ranked by confidence, which helps quickly identify the relatively low-confidence (possibly incorrect) policies. See the *ungrouped policies* view.

6.16.1 Semantics and Viewing

We support any mixture of block list/allow list (deny/allow) security models for different applications, letting application owners define very fine-grained policies to secure their applications while simultaneously allowing the security teams to enforce their guidelines and best practices on wide sets of applications. By taking into account the scope of security policies, we can guarantee that an application owner cannot negatively affect workloads that are not under their control, thus democratizing the tedious process of defining and maintaining security policies.

In order to better understand how security policies take effect in a dynamic and collaborative environment, let us define a few basic components of any policy:

Security Policy Property	Description
Consumer	Represents the client or the initiator of the connection. We allow for any filter on the inventory to be used dynamically to define the set of IP addresses that should be taken into consideration as the consumers (clients) of a service. Any cluster, user defined filter or scope can be used as the consumer of a policy.
Provider	Represents the server or the recipient of the connection. We allow for any filter on the inventory to be used dynamically to define the set of IP addresses that should be taken into consideration as the providers (servers) of a service. Any cluster, user defined filter or scope can be used as the provider of a policy.
Service	The service made available by the provider that should be permitted or blocked. This means the server (listening) port and IP protocol. All policies are *bidirectional*. A policy could apply (allow or deny) to either/both directions, from consumer to provider or the reverse direction.
Action	ALLOW or DENY: Whether we should allow or drop traffic from consumer to provider on the given service port/protocol.
Rank	Absolute or Default: Whether we are allowing the policy to be overridden by other lower priority applications (Default), or it should take effect even though it contradicts the app-specific policies defined by app owners. Generally, app owners use very fine-grained Default policies while security teams use broad Absolute policies to protect different zones, enforce best practices or quarantine a specific application.
Priority	Specifies the relative order of policies in a specific rank in a given application workspace. The absolute values of the priorities matter only to the extent that they determine the relative order of the policies. Among policies of the same category (Absolute or Default) in the same workspace, a policy with a smaller priority number takes precedence in the policy list over a policy with higher priority number.

6.16.1.1 Policy Scopes

In addition to the above attributes, the effect of each security policy is limited by the *scope* of the application workspace under which it is defined. The scope of each policy defines the set of all inventory items (workloads) that the security policy can potentially affect. Consider a simple example with three scopes **Apps**, **Apps:HR** and **Apps:Commerce**, where **Apps:HR** and **Apps:Commerce** contain possibly overlapping subsets of the items in **Apps**. Assume the owner of the **Apps** scope defines the following policy:

```
DENY PROD -> NON-PROD on TCP port 8000  (Absolute)
```

where PROD and NON-PROD are filters specifying all production and non-production hosts, respectively. Since this policy is defined under the primary workspace under *Apps* scope, it will affect all PROD/NON-PROD hosts (including ones that belong to *Apps:HR* or *Apps:Commerce* scope).

Now consider the case where the exact same policy is defined under the workspace with *Apps:HR* scope. In this scenario, the policy can only affect PROD/NON-PROD hosts under *Apps:HR* scope. More precisely, this policy will result in inbound rules on NON-PROD HR hosts (if any) denying connections on TCP port 8000 from **any** PROD host, and outbound rules on PROD HR hosts (if any) dropping connection requests to **any** NON-PROD host.

Note: It is important to note that consumer or provider inventory filters specified in a policy serve following purposes:

- these filters or groups specify the set of IP addresses that will get used in the firewall rules installed on the workloads.
- furthermore, these filters specify the workloads or Tetration agents that will receive policy or firewall rules.

As a concrete example, say provider filter in a policy with action ALLOW includes all inventory in the subnet 1.1.1.0/24. When this policy gets installed on a (say) Linux workload with Tetration enforcement agent and having IP address 1.1.1.2, the firewall rules look like:

1. For incoming traffic firewall rules allow traffic destined to 1.1.1.2 specifically and not to the whole subnet 1.1.1.0/24.
2. For outgoing traffic firewall rules allow traffic sourced from 1.1.1.2 specifically and not from the whole subnet 1.1.1.0/24.

Above is the default behavior of how firewall rules get programmed on the workloads. There can be special instances where user(s) may need to separate the two purposes of filters in a policy. That is, user(s) may need to specify the group of IP addresses that policy uses in the firewall rules which is different from the workloads that the policy gets installed to. For such scenarios, the policy model allows specifying *effective provider* and *effective consumer* – we will get into more details about these advanced options in section on Effective Consumer and Provider for a policy.

6.16.1.2 Policy Side View

The Policy Side View can be accessed after clicking on the services for a policy. Information about the policy such as rank, priority, action, consumer, provider, and service ports are available for viewing. After ADM runs, a policy confidence mark is added next to each service. Above the list of service ports, there are links for quick access to the conversations, quick analysis, and enforcement associated with the policy.

6.16.1.3 Approved Policies

Policies may be manually added or edited through the Policies tab, as shown below. Such policies are approved by default. Approved policies are shown with a thumbs-up icon next to the protocol type in the policy side view. The approved state can be toggled by clicking the thumbs-up icon. Policies may also be uploaded and those policies are approved by default unless explicitly marked as `approved: false`

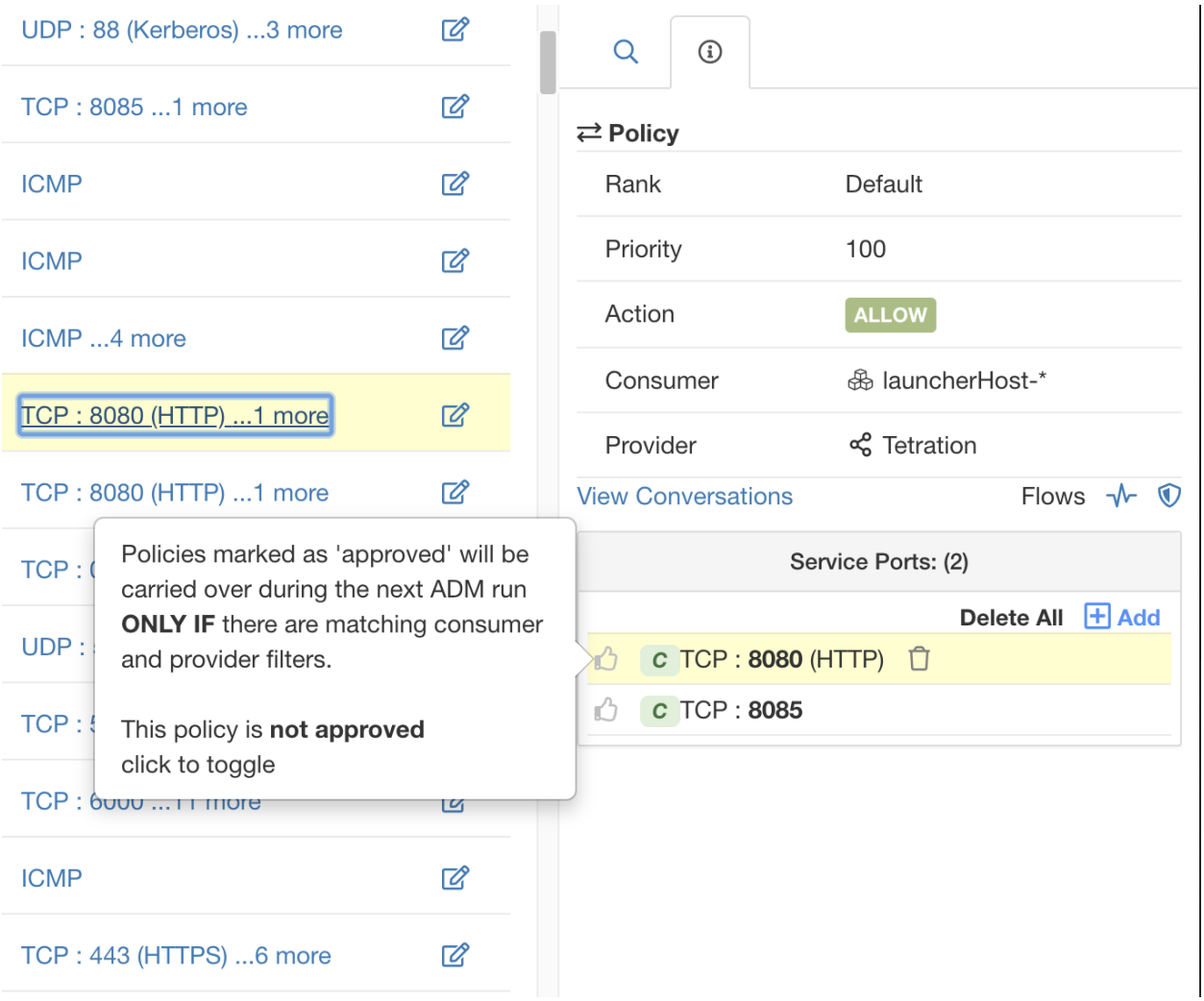


Fig. 6.16.1.3.1: Approved Policies

Briefly, in the following 2 ways, approved policies behave differently (and further explained below):

1. An approved policy can persist upon ADM runs, that is, it may remain in subsequent versions of the application workspace, but this is not gauaranteed.
2. An approved policy will prevent subsequent ADM runs from generating policies that are 'covered' by the approved policy.

Persisting an approved policy is often desired, since the user does not have to add the same policy upon an ADM rerun. Upon a run, an approved policy often persists, but note that this is not guaranteed, since approving a policy does not automatically lead to approving the clusters involved (if any). If either end of the policy is a non-approved cluster, and upon the ADM run, no newly generated cluster has sufficiently high overlap with such cluster, the approve policy won't persist. In all other cases, when *both* ends are any of: approved cluster, inventory filter or external scope, or a cluster that doesn't significantly change membership, the approved policy is preserved (but note that the cluster memberships may have changed in the last case). Therefore, if an approved policy involves an unapproved cluster, and if the user wants to preserve the approved policy, upon an ADM run, we strongly recommend that they also explicitly approve the cluster(s), at each end of the policy.

Approved (manually created) policies are often general policies and it is desired that, upon ADM runs, no redundant policies, that is policies that are already covered by them, be generated. Therefore, upon an ADM run, an approved

policy may also prevent generating policies that are already covered by it. The process to achieve this is briefly as follows. Upon an ADM run, any conversations that match the criteria for an existing approved policy will be excluded from the policy generation. This omission prevents redundant policies covering the same conversations from being generated. This is called **approved policy exclusion**. This process differs from the exclusion filters (See *Exclusion Filters*), in which matching filters, instead of policies are defined by the user. Exclusion filters prevent matching conversations from being visible to all parts of ADM runs. On the other hand, approved policies only exclude conversations from inducing policies in ADM run analysis, allowing these conversations to be considered in ADM's clustering analysis and cluster generation.

From the conversations view (See *Conversations*), the user can tell which conversations are excluded by existing approved policies from ADM policy generations, by filtering conversations with the excluded flag. The user can also explore which existing approved policies result in the exclusion of these conversations in the policy side view, by clicking the exclusion icon next to the conversation.

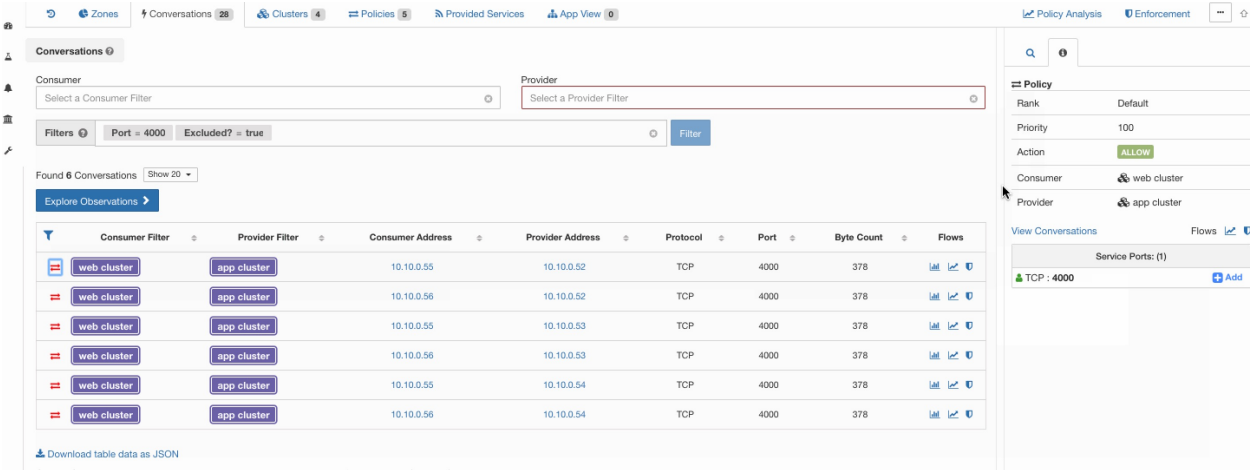


Fig. 6.16.1.3.2: Manually Adding or Editing Policies in list view

Explore approved policies excluded conversations

Approved policies in primary workspaces of a scope will also propagate to workspaces of child scopes and descendants. As a result, in an ADM run of a workspace, the policies that participate in the approved policy exclusion process do not only include the approved policies in this particular workspace, but also include the approved policies in the latest versions of primary workspaces of parent and ancestral scopes of the scope the workspace.

Other than manually input policy from policy side view page, any policies generated from accepting policy results from another workspace (See *Collaboration Among Applications* for details) are also considered approved policies.

6.16.1.4 Policy Global Ordering & Conflict Resolution

Given the very flexible, dynamic and distributed nature of the security policy intents, conflicts can arise between different policies defined under different scopes. More specifically, conflicts arise for workloads (inventory items) that belong to multiple scopes, such as parent/child or or overlapping sibling scopes, with contradicting policy intents (*i.e.*, when scopes overlap and have contradicting policies). It is not feasible to resolve such conflicts manually due to the dynamic nature of scope membership; workloads can enter and leave scopes as their properties change. Therefore, a global order is defined, as described below, for all policies according to the scope under which they are defined. For each workload, the list of relevant policies (according to consumer/provider/scope) is identified and sorted by the global order. The decision to permit or drop a flow is made based on the *first* matching policy in the sorted list.

By understanding the global ordering scheme of security policies, network admins can define the correct scopes and their priorities to apply the overall desired policies on workloads. Within each scope, application owners maintain

their ability to enforce fine-grained policies on their respective workloads.

A global network policy consists of:

1. A number of scopes ordered by priority (highest priority first).
2. Each scope has at most one primary application with absolute policies, default policies and a catch-all action.
3. Each group of absolute or default policies within each application is sorted according their local priorities (highest first).

The global order of policies is defined as follows:

1. Groups of absolute policies from the primary applications of all scopes (arranged from highest to lowest priority).
2. Groups of default policies from the primary applications all scopes (arranged from lowest to highest priority).
3. Catch-all policies from all scopes (arranged from lowest to highest priority).

Note that the scope order applies to groups of policies in category 1 and 2, rather than individual policies. Within each group, individual policies with with lower policy priority numbers taking precedence.

For a specific workload, first the subset of scopes it belongs to is determined, then the above order is applied. The catch-all policy from the lowest priority (enforced) workspace to which this workload belongs is the applicable catch-all (but an absolute or default policy may override). For a given flow on that workload, the action of the highest matching policy is applied.

Notes:

- An application should have either Absolute or Default policies defined. If both are missing, the application is ignored. The application's catch-all policy *will not* be included in the global order.
- If a workload has two or more interfaces, in overlapping or disjoint scopes, the catch-all policy of the lowest priority workspace with enforcement enabled will be applicable (among all the applicable catch-all policies).
- The order of Default policies in the global order is the reverse of the scope priorities. This provides the flexibility for network and security admins to define broad policies for all scopes securing the perimeter of all applications including those that do not have policy enforcement enabled. At the same time application owners who have enabled enforcement on their scopes have the ability to override these default policies.

We expand our previous three-scope example to illustrate this ordering scheme. Assume the three scopes are assigned the following priorities (See *Navigating to Applications* for instruction on how to change scope priorities):

1. Apps
2. Apps:HR
3. Apps:Commerce

Each of these scopes has at most one primary application with absolute policies, default policies and a catch-all action. Each group of absolute or default policies within each application is sorted according their local priorities.

The global ordering of the policies will be as follows:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:Commerce Absolute policies
4. Apps:Commerce Default policies
5. Apps:HR Default policies
6. Apps Default policies

7. Apps:Commerce Catch-all
8. Apps:HR Catch-all
9. Apps Catch-all

A workload that belongs to the *Apps* scope will receive only the following policies in the given order:

1. Apps Absolute policies that match the workload
2. Apps Default policies
3. Apps Catch-all

A workload that belongs to the *Apps* and *Apps:Commerce* scopes will receive only the following policies in the given order:

1. Apps Absolute policies
2. Apps:Commerce Absolute policies
3. Apps:Commerce Default policies
4. Apps Default policies
5. Apps:Commerce Catch-all

A workload that belongs to the *Apps* and *Apps:HR* scopes will receive only the following policies in the given order:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:HR Default policies
4. Apps Default policies
5. Apps:HR Catch-all

A workload that belongs to all three *Apps*, *Apps:HR* and *Apps:Commerce* scopes will receive the following policies in the given order:

1. Apps Absolute policies
2. Apps:HR Absolute policies
3. Apps:Commerce Absolute policies
4. Apps:Commerce Default policies
5. Apps:HR Default policies
6. Apps Default policies
7. Apps:Commerce Catch-all

Note that the relative ordering of the *Apps:HR* and *Apps:Commerce* scopes only matters if the two scopes overlap, *i.e.*, there are workloads that belong to both scopes. This is because policies are always defined under a scope. A workload belonging to one scope only will not be affected by policies from the other scope, thus the order does not matter.

6.16.1.5 Grouped Policy Table View

Policy table (list) view provides a simple way to view, edit and understand policies for a given application. Click on the list icon to navigate to the policy list page.

You can see three tabs separating Absolute, Default and Catch-all policies. All policies are grouped by consumer/provider/action for more concise viewing. You can examine aspects such as services (all the ports) by clicking

on the entry in the Services column. Once clicked, on the right panel, one can view the full list of ports, and can click on 'view conversations' to view the conversations that generated the policy (see *Conversations*).

You can edit each of these policies as well as the catch-all action, by clicking on the edit icon next to them. Adding new services (ports) to an existing row is accomplished by clicking on the service column and then on the **ADD** button on the side panel:

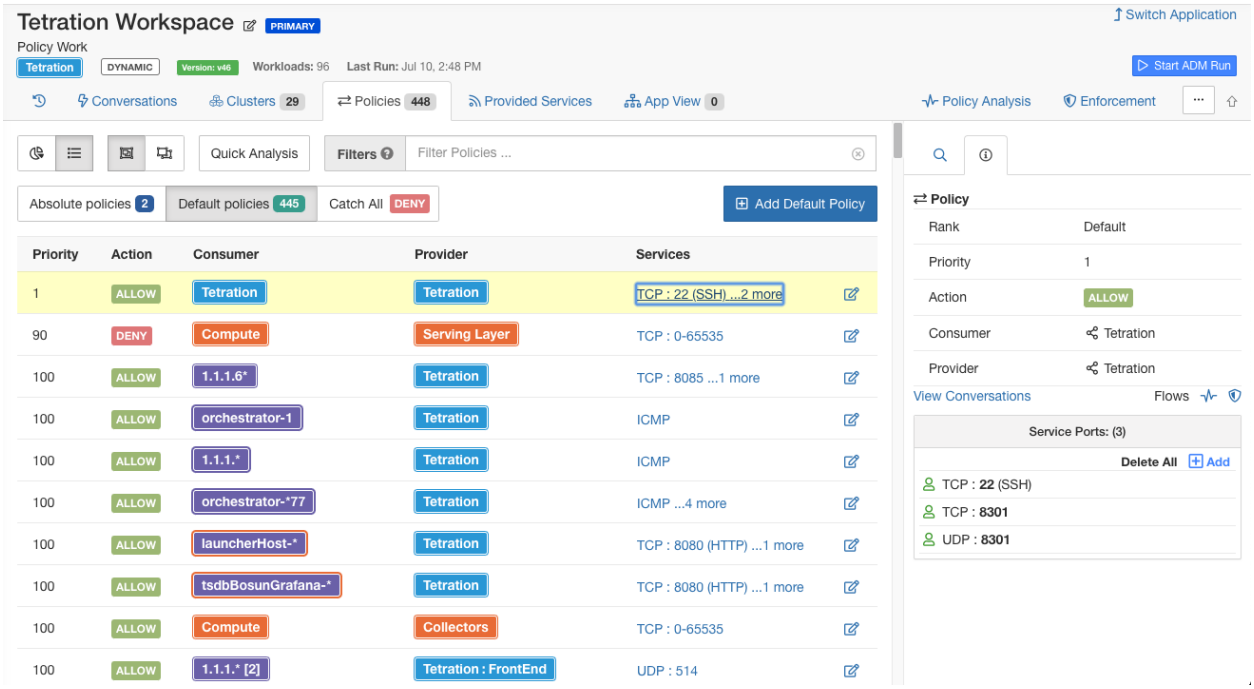


Fig. 6.16.1.5.1: Adding a new service

Click on **Add Absolute Policy** or **Add Default Policy** to create a new pair of consumer/provider with specific action and priority number. When selecting the consumer/provider, you can type the name of a cluster (of the current application) or a filter or a scope (from the same tenant) and get suggestions about existing filters. If no matching filter is found you can create a new one in the same page by clicking on the **Create New** item in the drop-down and defining the filter in the dialog.

After the creation of a new row, the service column indicates **inactive**, which means that there are no services defined yet. Click on the *inactive* hyper-link to view the policy on the side-panel. Then you can add/remove services to the policy as described above.

The following animation demonstrates a few of the mentioned workflows in action:

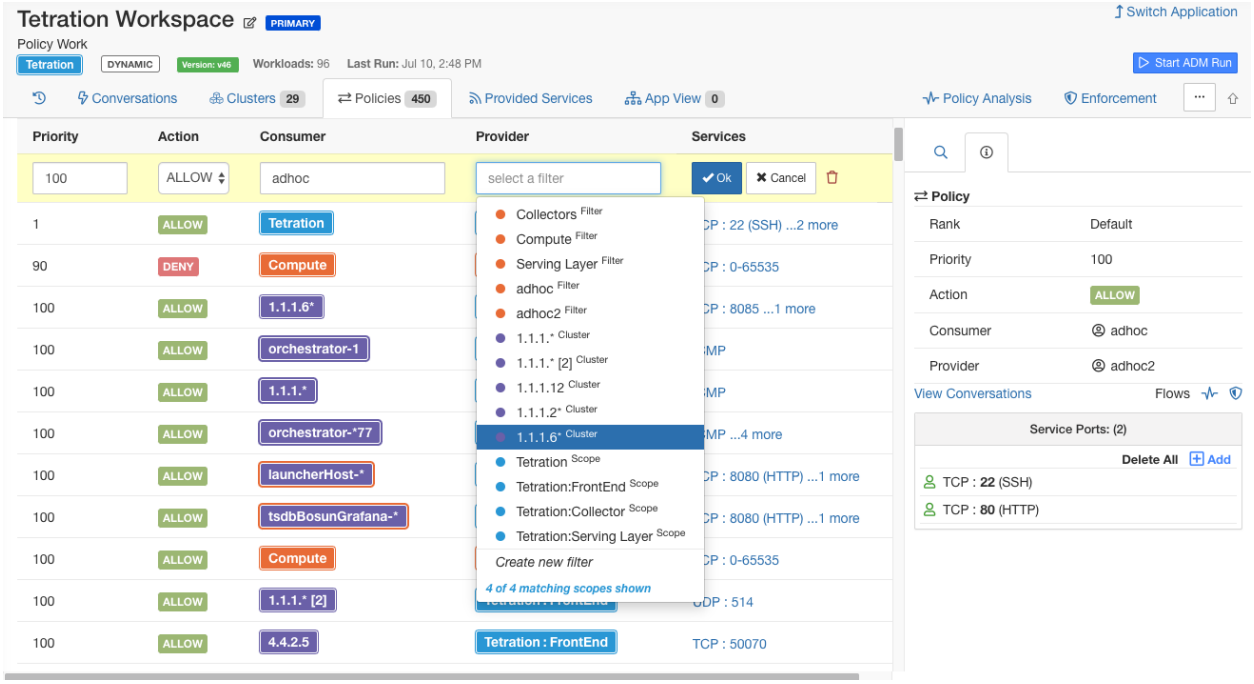


Fig. 6.16.1.5.2: Editing Policies in list view

Removal of Redundant Policies On subsequent ADM runs approved policies in primary workspaces will remove matching conversations for policy generation, so redundant policies are not generated. Note this, as the case for exclusion filters, this functionality may not work perfectly on non-primary workspaces if the policy uses a Cluster filter defined in the workspace. Cluster filters from a non-primary workspaces are not active, and will not match any flows, thus redundant policies may still get generated in non-primary workspaces upon ADM runs.

6.16.1.6 Ungrouped Policy Table View

Rows in this ungrouped list view are differentiated by port (port-range) in addition to consumer/provider/action. Thus one can search or filter the rows easily based on ports. In particular, one can view policy confidences (or confidence on the server port classification). The confidence of a policy is determined by the confidence in the client-server decisions made for the conversation(s) that led to the creation of the policy (See *Client Server Classification*). The user can use this view to rank by confidence and examine relatively low-confidence policies and possibly remove and replace them if deemed incorrect (assuming write access). See animation. NOTE: This view currently does not provide a means to edit or add a policy (deletion only).

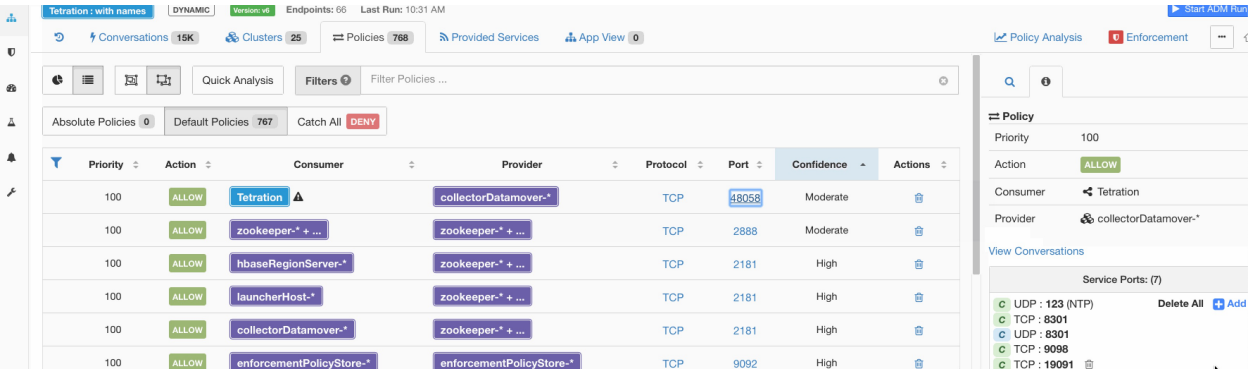


Fig. 6.16.1.6.1: Ungrouped list view of policies

6.16.1.7 Policy Visual Representation

Policy visual representation provides a graphical view of the policies. Click on the graph icon located to the right of list icon to navigate to the policy visual representation page.

The graphical view consists of nodes and edges. The nodes on the canvas represent the consumers and providers of a policy. The consumers and providers here can be a Cluster (purple), Inventory filter (orange) or Scope (blue). User can view membership of the consumer/provider by double clicking on the nodes. An edge on the graph represents one or more policies between the consumer and provider. The policy edges are grouped by consumer and provider for more concise viewing. The user can examine all the aspects of a policies such as services (ports), action (Allow/Deny) and protocol between a consumer and provider by clicking on the edge in the graph.

To create a policy edge hover the cursor on the consumer until you see a "+" sign and then hold and drag the edge on to the provider. On the Policies popup fill in the port and protocol info click "Add", this creates a Default policy. To create an Absolute policy click on "Show Advanced options" and mark the policy as Absolute. Policies can also be added, updated and deleted by double clicking on the policy edge and updating the info on the Policies popup.

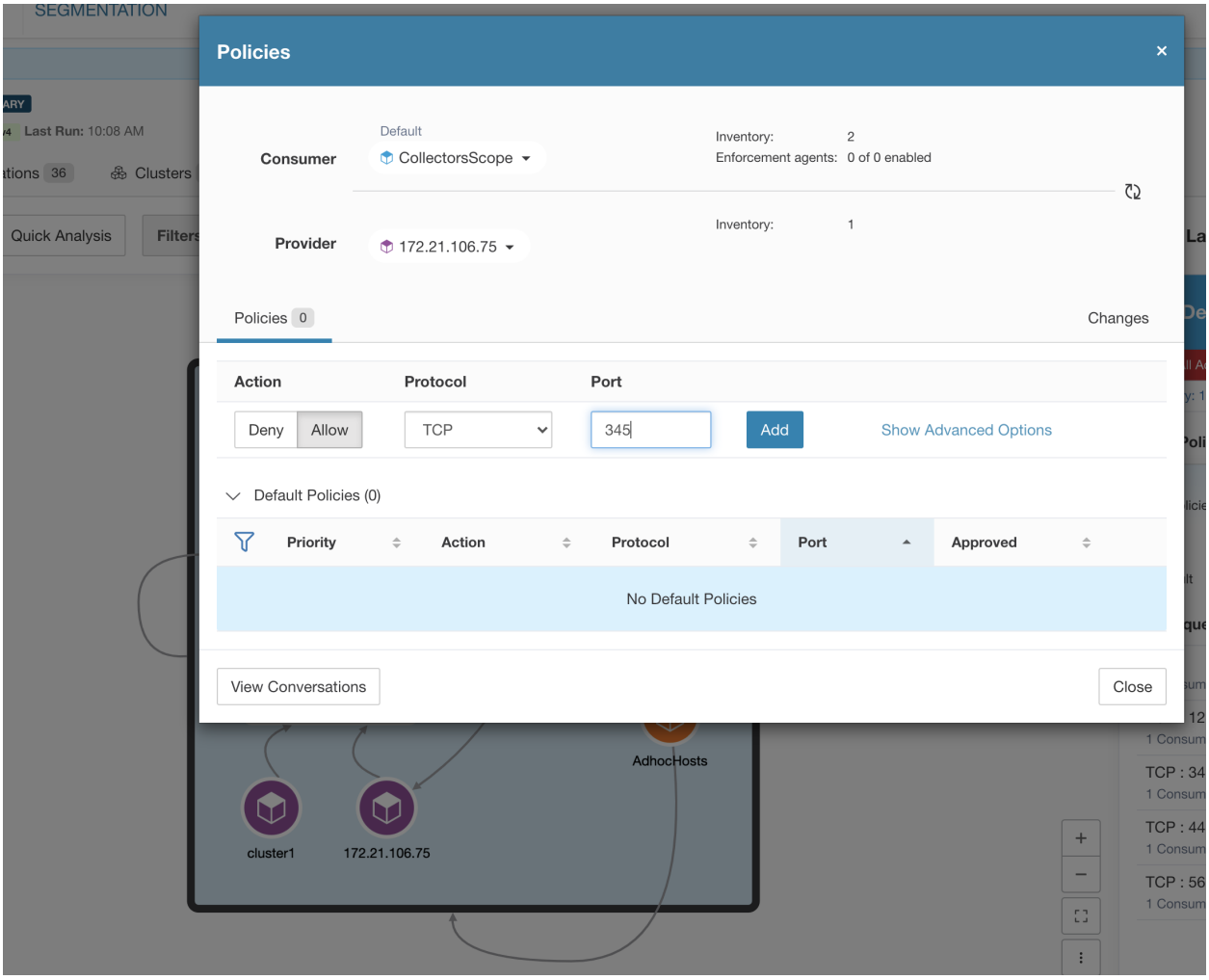


Fig. 6.16.1.7.1: Policy creation in graphical view.

By default the right-hand panel shows all the policies that are generated to and from the scope of the application. The user can select a node on the canvas and view the policies entering and leaving the node on the panel. The user can filter the policies by drilling down the different tabs present on the panel. On the first layer of filtering on the panel the user can filter internal and external policies, on the second layer the user can filter policies based on policy rank (Absolute/Default) and so on. For example to view all the Default policies with TCP protocol entering or leaving the 'dev' scope, the user can click on 'dev' scope on the canvas and further filter the desired set of policies using different tabs on the panel.

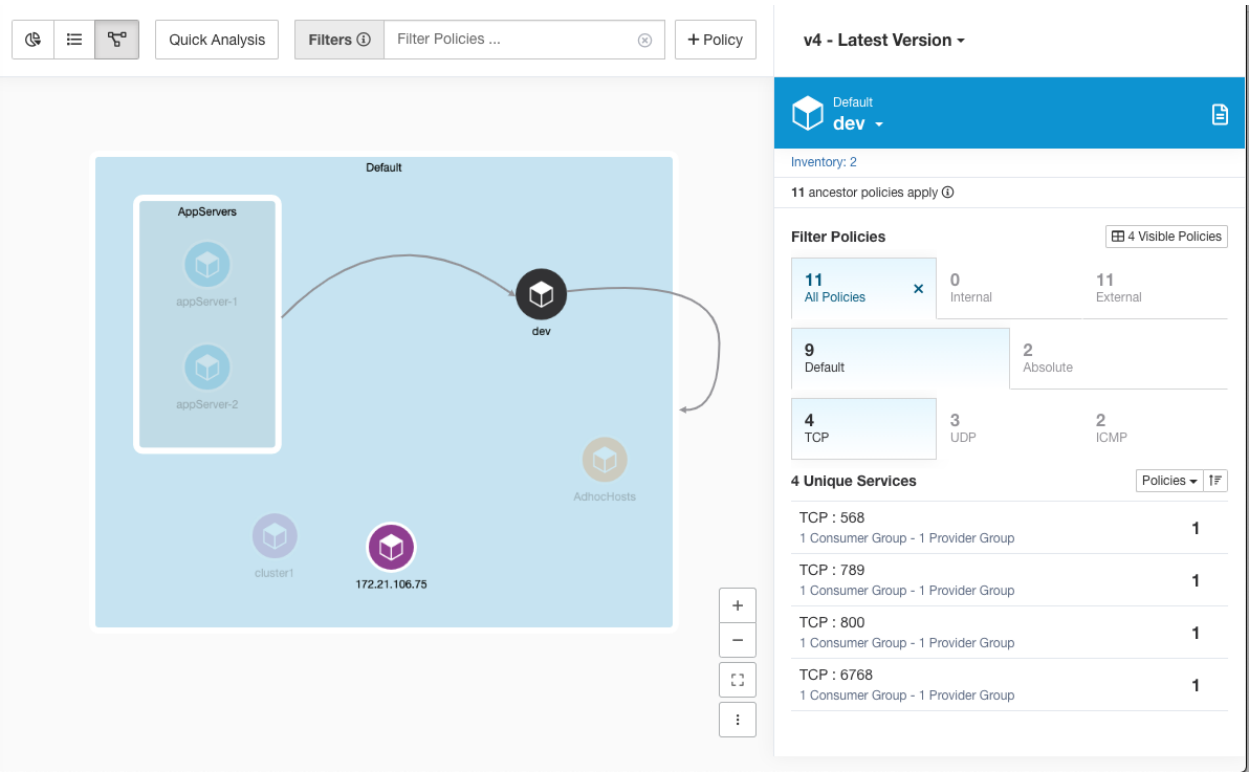


Fig. 6.16.1.7.2: Filtering policies in graphical view.

6.16.1.8 Policy Chord View

Policy chord view provides a top-level graph view of all *ALLOW* policies in one chord chart with various ways to drill down and filter information.

The following figure shows some of the basic concepts of the policy chord chart. The arcs around the circumference of the chart represent clusters or partitions (group of clusters). Expanded partitions show up as a glow around all of the member clusters.

The chords represent a group of all policy intents between a pair of clusters, filters or scopes. If a chord starts or ends at a partition, it represent the union of all policies from all the clusters inside that partition.

The chord represents bidirectional set of policies. The thickness of a chord on each side is proportional to the number of services consumed by the corresponding cluster or partition.

TIPS:

- You can use the edit clusters view (see *Clusters*) to get a quick tabular view of clusters and their content. Use the policy view when you want to see the communications (the edges or policies).

NOTES:

- Double click on a partition arc to expand/collapse that partition.
- Single click on any of the chart elements, i.e., partition, cluster, or policy selects or deselects that element. Moreover, the side panel gets updated with context information about the latest clicked element.
- Double click on the canvas outside the chart to reset the chart to its original state.

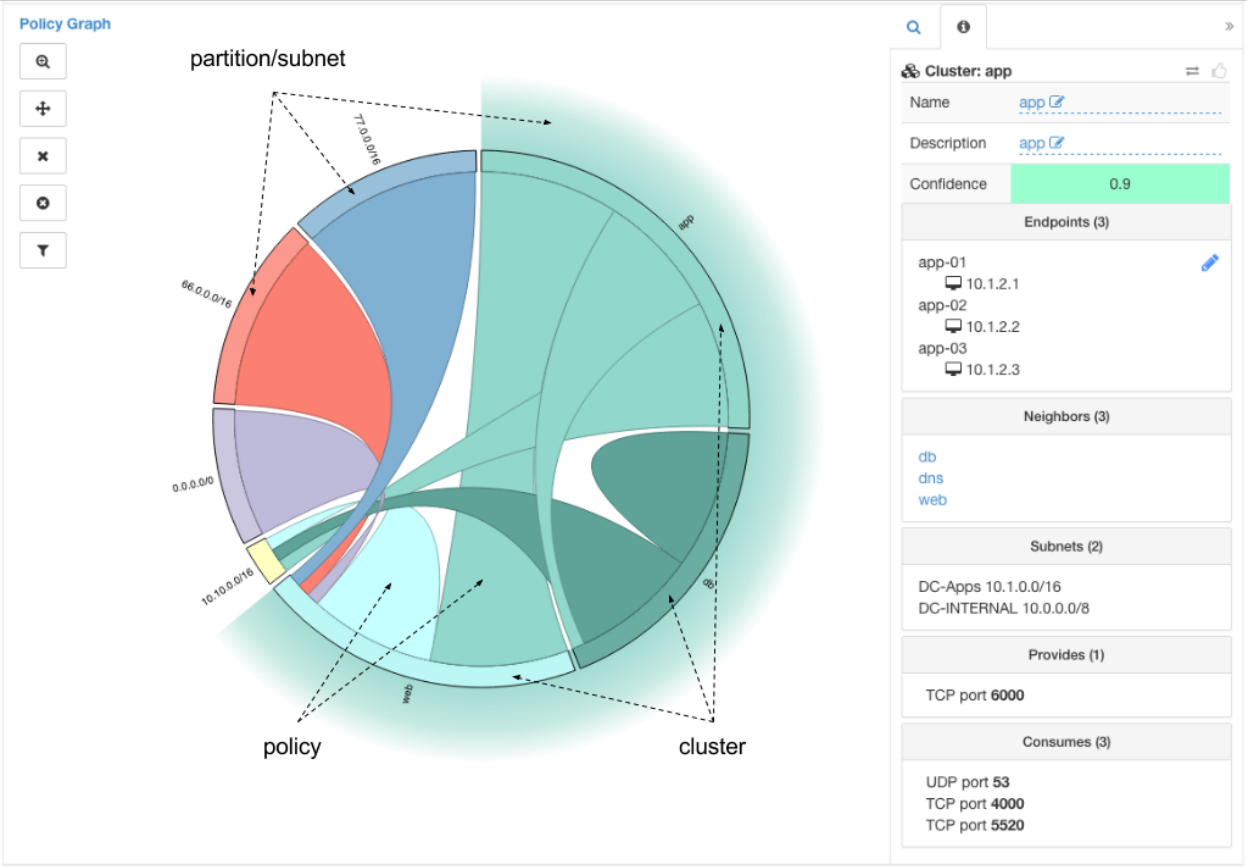


Fig. 6.16.1.8.1: Policy Chord View

Chord Chart Toolbar

The set of controls or the toolbar on the top-left corner of the policy view page is designed to simplify interaction with large and complex charts by allowing the user to focus on a subset of the clusters and policies.

The **Filter** button helps filter out the policies by port and protocol. Green colored button indicates that the filter is active. Simply click **disable** to remove any filtering. See example below:

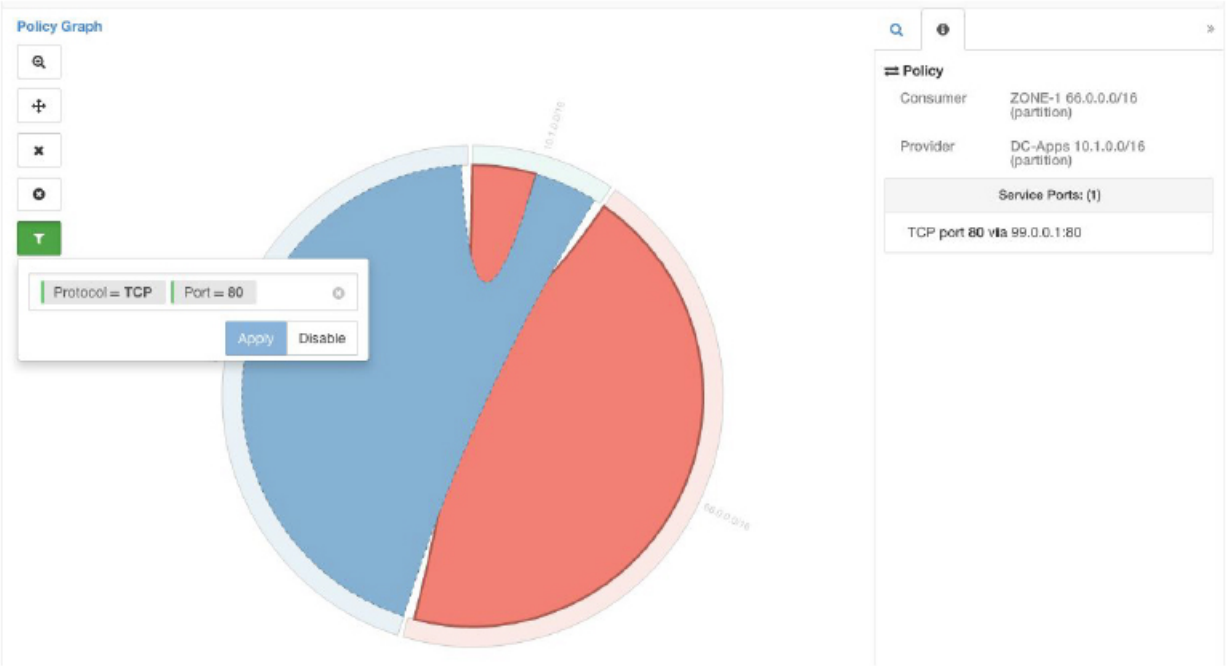


Fig. 6.16.1.8.2: Chord Chart Toolbar

The **Show Cluster Detail** button helps drill down into the content of selected clusters and observe conversations/connections of the hosts inside the clusters.

NOTE: At least one cluster (not partition) needs to be selected to have the ability to drill down into a cluster. The rest of the controls as their names indicate help remove unwanted clusters or policies or limit the chart to only one or more neighbors of a cluster.

6.16.1.9 Quick Analysis

Quick analysis enables testing a hypothetical flow against all the policies in the current application workspace as well as all other relevant policies from other applications. Quick analysis is available only on **Primary** application workspaces to facilitate debugging and experimentation with different security policies, without the need to publish the workspace.

Note: Tetration software versions prior to 2.0.2.x allowed quick analysis on both primary and secondary workspaces. This feature has been simplified to run only on primary workspaces.

Click on the **Quick Analysis** button to view the dialog. Enter the Consumer (client) IP, Provider (server) IP, port and protocol for the hypothetical flow, then click on **Find Matching Policies** button.

A policy decision will be shown indicating whether the hypothetical flow would be allowed or denied given the policy definitions in the latest version of the workspace and all other policies from relevant workspaces that are already pushed for live policy analysis.

At the bottom of the dialog, we show the matching outbound and inbound policies policies separately, and in their globally sorted order. It is only the very first row on either side that has any effect. For a connection to successfully get established, we need both the top outbound rule on consumer and the top inbound rule on the provider side to be ALLOW rules.

Showing all other matching policies in their order, provides a valuable debugging tool to help sort out issues in policy definitions when a certain policy seems to not be taking any effect. You can add, update, or delete policies from the workspace, and repeat the analysis immediately without the need for publishing the workspaces.

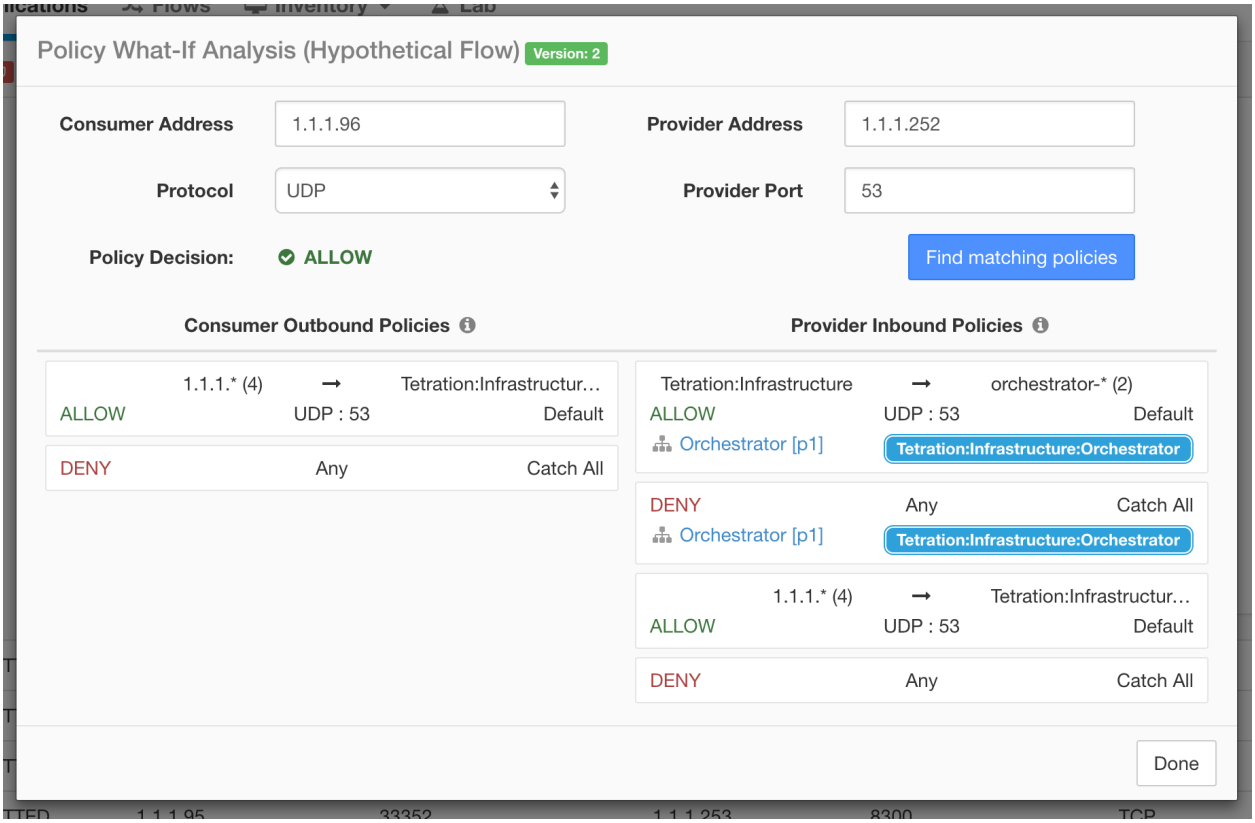


Fig. 6.16.1.9.1: Quick Policy Analysis

6.16.2 Live Analysis

Policy analysis is an important part of generating security policies for applications under a allow list model. Once the set of network security policies generated by ADM is reviewed and approved by the user, and before pushing the policies to the enforcement engine, the user must try to get answers to a few questions:

1. What would be the impact of the policies on an existing application if the policies start getting enforced now?
2. Could we have prevented a previously known security attack/risk via enforcing the new set of policies?
3. Is the network enforcement engine implementing the policy intentions correctly?
4. How much is the average network usage or other telemetry data associated with each security rule?

The first question is of particular interest, since the flow observations used by ADM algorithms to generate the allow list modeled policies may not fully capture all of the active components of the application.

This might be caused by picking a small duration to run ADM algorithms. Hence, pushing the new policies without an analysis check may break the application.

Policy analysis is provided to cross examine the policies generated by ADM and enhanced by the users against live traffic in the network. The first step of the policy analysis workflow is to **Enable policy analysis** on the Application

workspace to allow its policies to be cross examined with the ongoing flows in the network. It is possible to publish each workspace individually, but not all workspaces need to be published.

6.16.2.1 Enable Policies

Once the user has verified the results of the ADM algorithms in a workspace, they can start the analysis by clicking on **Enable Policy Analysis** on the ADM workspace. To **Enable Policy Analysis** follow these steps:

1. Toggle the application to **Primary** by clicking "Secondary" next to the application name in the header.
2. Navigate to the **Policy Analysis** tab.
3. Click the **Start Policy Analysis** button on the right.

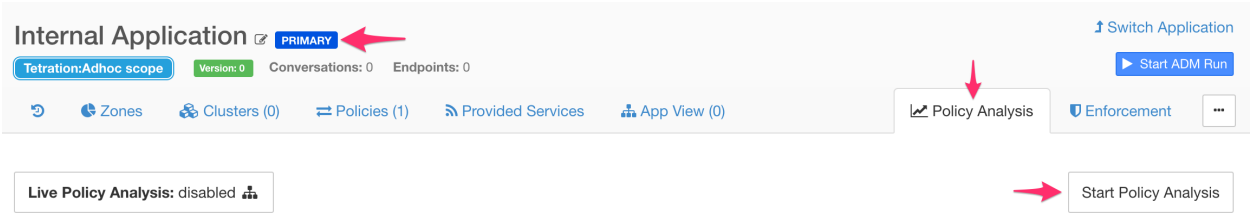


Fig. 6.16.2.1.1: Enable Policy Analysis

6.16.2.2 Analysis without Policies

The flows into, out of, and within the Scope of the Application may be affected by policies published in other workspaces. If policy analysis is not enabled on this Application the flows will be marked with those of the other published Applications in the system.

Note: If no applications have published policies, the timeseries chart will be empty.

Disable Live Policy Analysis

Disabling the published policies does not affect the contents of the workspace. It only removes the policies from the policy analysis tool. Other policies may now have priority over some flows and they will be marked accordingly.

Fig. 6.16.2.2.1: Disable Live Policy Analysis

6.16.2.3 Policy Analysis Overview

The Policy Analysis page shows the results of cross-checking published policies against live network traffic. The policy analysis tool classifies all the flows traveling into, out of and within the Scope of the Application into three categories:

1. **Permitted**: Flow was allowed by the network, and also by the policy group.
2. **Escaped**: Flow was allowed by the network, but should have been dropped according to the policy group.
3. **Rejected**: Flow was dropped by the network, and also by the policy group.

In the following screenshot you can see an overview of the page. There are permitted flows up to about 12pm. New policies were then published by another application (published on this application would create a label flag), causing flows to be marked as escaped.

Fig. 6.16.2.3.1: Overview of Policy Analysis

You may filter the flow information presented in this page via a faceted filter bar similarly to the flow search page. Clicking on the "Filter Flows" button updates all the charts accordingly. Hovering on the chart shows the percentage of the aggregate observed flows at that timestamp.

Moreover, clicking on that timestamp reveals a list of all filtered flows in a table below for further analysis.

You may choose to limit the interactions to one of the three categories by selecting/deselecting the categories at the top of the time series charts.

Similar to the flow search page, there is a Top N chart on the right. This shows the top Hostnames, Addresses, Ports, etc. contributing to the data shown in the timeseries chart on the left. An example use case may be limiting the timeseries chart to just escaped flows and selecting "Ports" in the Top N chart to see the top ports contributing to escaped flows (See details below).

6.16.3 Policy Analysis Details For Advanced Users

Flow Disposition

In policy live analysis, to decide on whether a flow is **Permitted**, **Escaped**, or **Rejected**, we have to first determine the **Disposition** of the flow from the network perspective. Each flow will receive an **ALLOWED**, **DROPPED** or **PENDING** disposition, derived from the signals and observations given by hardware agents or Tetration software agents (applies to only deep visibility agents and enforcement agents which captures

real-time flow data). There are a number of scenarios based on the agent configurations along the path of the flow and the flow types.

First, regardless of flow types, if any agent (hardware agent, deep visibility agent or enforcement agent), along the path of a flow reports that the flow is DROPPED, the flow will receive a DROPPED disposition.

When there is no DROP reported by any agents along the path of the flow, We consider the case of bidirectional flows and unidirectional flows separately. When bidirectional flows are observed, we look at flows in pairs (forward and

reverse) based on their source, destination ports and protocol, as well as their timings. The same cannot be done for unidirectional flows.

For bidirectional flows, if there are deep visibility or enforcement agents installed and data plane enabled on both ends, a forward flow will receive an ALLOWED disposition if both the source and the destination agent report that the flow is observed. Otherwise, the forward flow will get a PENDING disposition. If there is only one deep visibility or enforcement agent installed on either the source or the destination side, then the forward flow will received an ALLOWED disposition if and only if the agent observes subsequent reverse flow within a **60** seconds window. Otherwise a PENDING status will be assigned to the forward flow. The disposition of the reverse part of the bidirectional flow follows the same logic except that now the source and the destination is reversed. For example, in the case where only one side has an agent, whether a reverse flow disposition is PENDING or ALLOWED depends on the observation and timing of its subsequent forward flow based on the same logic.

Note that we assume firewalls implement silent drop. If a reject message is sent on the **same** flow (e.g. rejecting a TCP SYN with RST + ACK), a reverse flow will be detected, and the previous forward flow will be marked as ALLOWED. However if the reject message is sent on a different flow (e.g. rejecting a TCP SYN with an ICMP message), the forward flow will remain as PENDING.

For a unidirectional flow, the flow will be considered DROPPED if it is reported as DROPPED by any agent as in the case of bidirectional flows. However, since there is no matching reverse flow, the flow will have PENDING disposition status, if both agents observe the flow.

Violation Types

The flow dispositions are checked against the policies being analyzed to determine the final violation types.

A flow's violation type will be

- **Permitted**, if its disposition is ALLOWED or PENDING, and its deciding policy action is ALLOW,

- **Escaped**, if its disposition is ALLOWED, and its deciding policy action is DENY,
- **Rejected**, if its disposition is DROPPED or PENDING, and its deciding policy action is DENY,

Note that since version 3.4, Tetration policy analysis no longer reports the **Misdropped** flow category. The Tetration system will only assign DROPPED status to flows whose relevant agents explicitly report their DROPPED status. When there is no explicitly report of dropping for agents, Tetration no longer infers whether a flow is dropped, rather such a flow will receive PENDING status.

When disposition is PENDING, the policy will be given the benefit of doubt. That is,

- if disposition is PENDING and policy action is DENY then violation type is set to Rejected.
- if disposition is PENDING and policy action is ALLOW then violation type is set to Permitted.

For a bidirectional flow, if the policy violation types of forward and reverse part of the flow agree, only a single type is shown in the policy analysis or enforcement analysis page. Otherwise, forward and reverse are shown separately, such as PERMITTED:REJECTED.

Next we provide a few example scenarios for flow violation types, based on disposition and violation logic.

1. Packets dropped at the source-side enforcement

- In this case, the source side tetration egress agent will report that the flow

is DROPPED.

2. Packets leave the source.

- If there is only a deep visibility or enforcement agent on the source side, the flow will be

reported as ALLOWED by the egress agent if a reverse packet is also oberved in by the agent in a 60 seconds window. - If there is a deep visibility agent on both the source and the destination side, the flow will be given a DROPPED disposition status, if and only if the ingress agent reports that the flow is DROPPED. Otherwise, the flow will be reported as ALLOWED.

3. Flow packets received at the destination, but no reverse traffic. - The flow will get a PENDING status, if there is no destination side agent. Otherwise, it will be assigned ALLOWED status.

A Deep Dive Into Diagnosis Using Policy Analysis

From the definitions of the three violation types, it is easy to see that **Escaped** flows require some special attention as their actual flow dispositions differ from the intended actions of the currently analyzed policies. Enforcing currently analyzed policies will potentially block these flows. If some of these flows are important flows for the normal operations of certain applications, blocking the flows may adversely affect the performance or functionalities of those applications.

Therefore, it is critical to examine this category of policy results in analysis in order to guarantee that enforcing the latest policies do not create unintended enforcement results. Next we highlight a number of most commonly used filters (and explanations) when drilling into specific flows when conducting diagnosis on policy results.

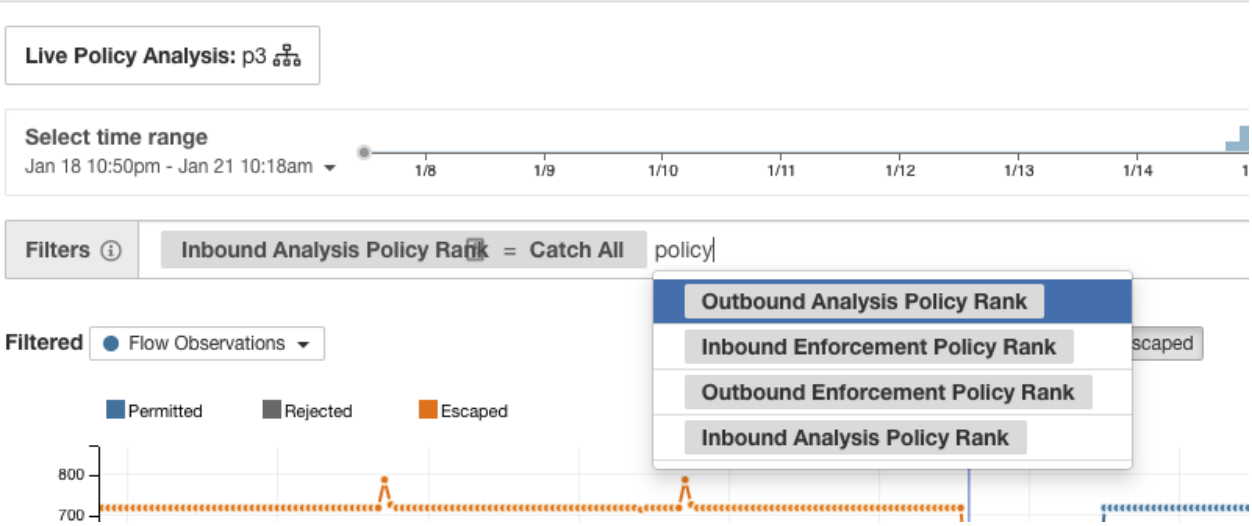
1. Checking only *ESCAPED FLOWS*, or *REJECTED FLOWS*

We can click and select flows with different violation types to only focus on the specific types of flows on the policy analysis page.

2. Identifying catch-all policy matched flows with inbound and outbound policy ranks

It is important to understand what flows are matched to catch-all policies, especially in an allow-list policy model. If these flows are legitimate but do not have explicit allow policies configured for them, the user may want to add appropriate explicit policies in the corresponding inbound or outbound scopes. On the other hand, if they are suspicious flows, we want to quickly identify them and further investigate their details.

To focus on these flows, we can apply filters based on the *catch-all* value of **inbound_policy_rank** or **outbound_policy_rank**, depending whether we are looking at the inbound, outbound or both sides, shown below.



3. Filtering out TCP flows with RST: *Fwd flags does not contain RST, Rev flags does not contain RST*

Some escaped TCP flows have RST flags set. These flows are reseted by either their consumers or providers. They are essentially unestablished connections without data exchange, but may be reported as ALLOWED because the agents see their handshaking packets. Since they do not have established connections to begin with, they will not be affected when currently analyzed polices are enforced. Filtering out TCP flows that have RST flag on either side allows us to focus on more meaningful and important escaped flows whose established connection will get blocked by the currently analyzed policies.

4. *address type = IPv4, address type != IPv6*

Focus only on IPv4 flows if most of the traffic are using IPv4. It is also helpful to filter out *link-local* address.

5. *top Hostnames, top Ports, top Addresses, top Scopes*

Selecting *Hostname, Ports* or *Addresses* from the TopN feature window helps the user quickly survey the landscapes of the analyzed flows. We can usually combined these with other filters to drill-down to a particular type of traffic when diagnosing policies. It helps us to priorize which flows to focus on in the next step of diagnosis.

6. *Consumer Hostname contains {something}, Provider Hostname contains {something}, Provider Port = {some port number}, Protocol = TCP Protocol != ICMP*

Once we have an idea about the top candidates of the targeted flows regarding their hostnames, port and etc, we can choose to drill down the flows by either applying drill-down filters directly from values given in the top N query window or manually entering relevant filters into the flow search filters bar.

7. Check individual flows and quick analysis

Finally, we are able to focus on a specific flow to examine its policy result by clicking the row corresponding to the flow. Pay attention to the policies matched to the flow and the scopes of both the consumer and the provider addresses. If the policy action does not match your intended action, you need to create appropriate policies in workspaces associated with the consumer's and/or the provider's scopes to change the policy action.

The figure below shows an example workflow of narrowing down escaped flows using some of the highlighted filtering. The search input also supports "," and "-" for Port, Consumer Address and Provider Address, by translating "-" into range queries.

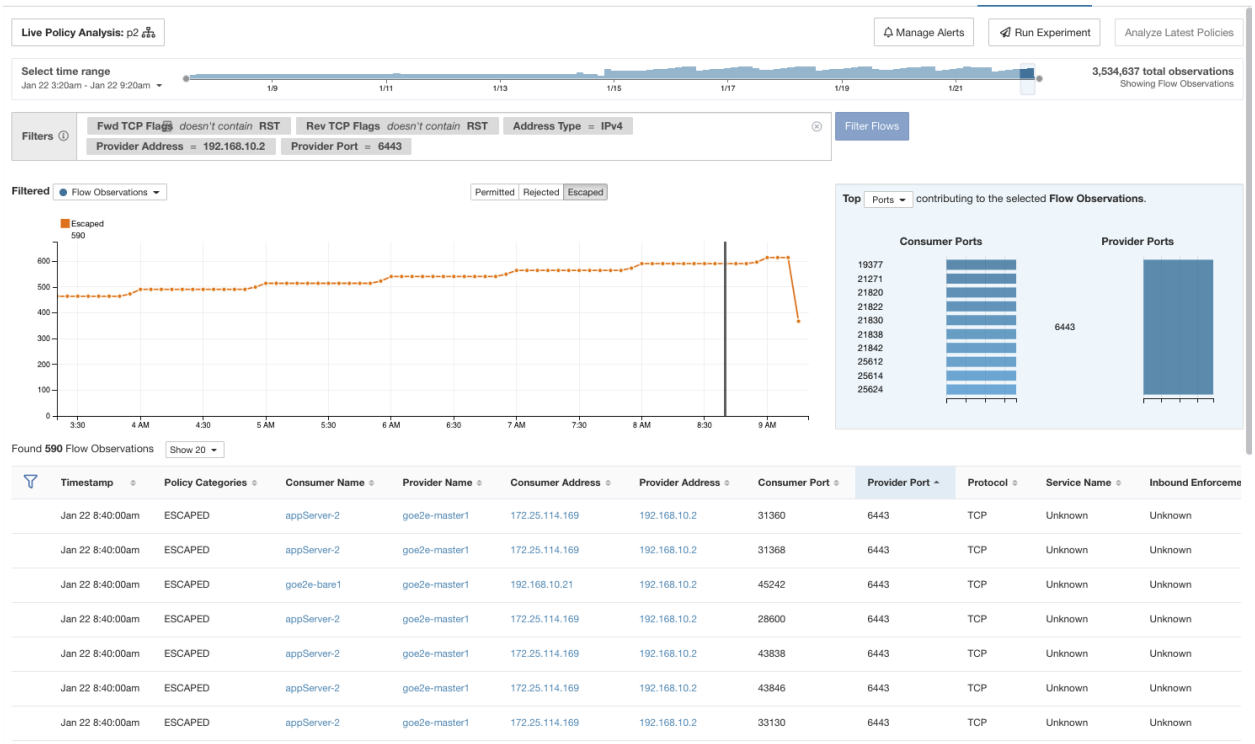


Fig. 6.16.3.1: Policy analysis diagnosis example

6.16.3.1 Analyze Current Policies

Modifications to the Application workspace are not automatically synced to the policy analysis tool, but the workspace can be republished any number of times by clicking on **Analyze Latest Policies** to reflect the changes.

The act of publishing a given workspace takes a snapshot of all the clusters and policies defined in that workspace for further analysis. We refer to these snapshots as the **Policy Analysis Versions** and they start with the letter 'p', for example, 'p1'

6.16.3.2 Policy label flags

All of the published policy versions are available for examination on the policy analysis timeseries chart via **Policy Label Flags**. If we click on the flag it navigates us to the particular policy analysis version on the *Semantics and Viewing* page.

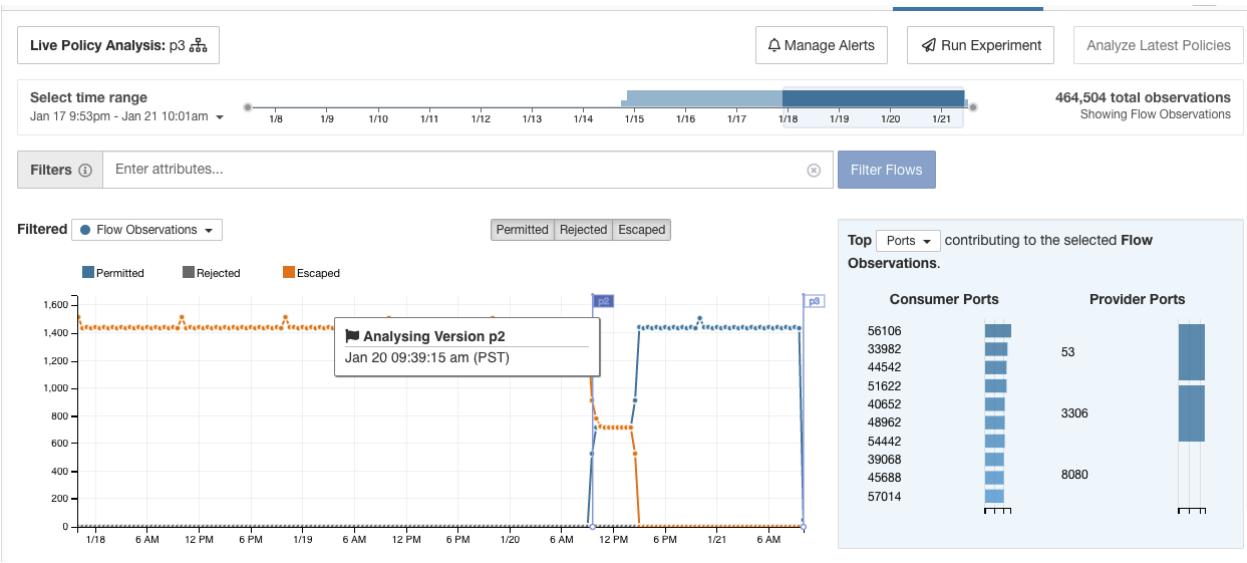


Fig. 6.16.3.2.1: Policy label flags

A timeseries chart with policy labels shows the changes in the policy group over time. This helps keep track of changes in published policies in case multiple users make changes to the original Application workspace and publish those policies for analysis.

Note: The policy live analysis results are currently not available from the *Data Platform*. Only enforcement policy results are available in Tetration Data Platform.

6.16.3.3 Policy Experiments

The default behavior for the analysis of published policies is to mark live network traffic according to the rules defined in the policy group. However, certain short-lived flows (like a known attack) may never occur in the network. In order to verify the hypothetical network security behavior under the published policies, you can create backdated policy experiments. In other words, the policy experiments help address the question "What if I had this set of network policies at the time of an attack?"

There are two steps to run a policy experiment:

1. Click on the **Run Experiment** button on the right corner of the policy analysis page.
2. In the new dialog select a name and a duration for the policy experiment.

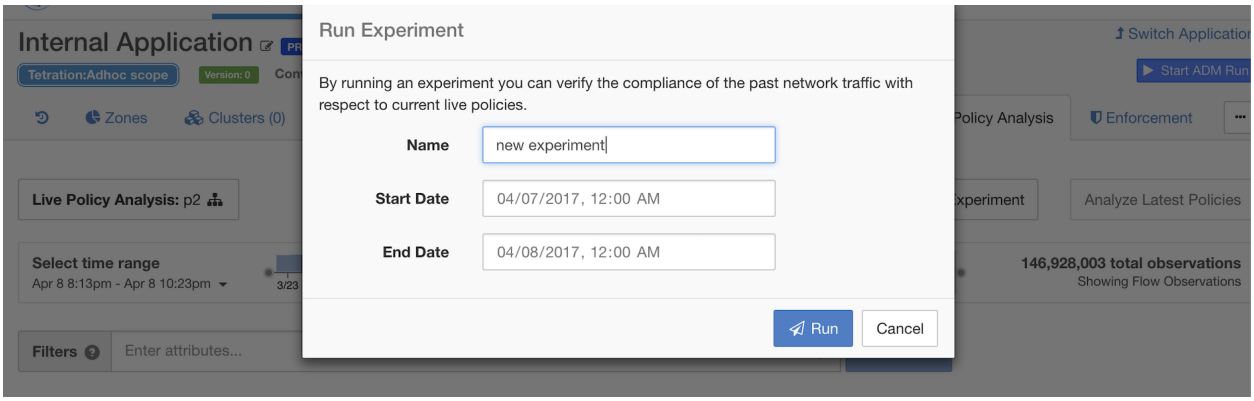


Fig. 6.16.3.3.1: Run Experiment Form

This will start a new policy analysis job which goes back in time and reanalyzes all the flows in the selected duration against the selected published policy.

This job may take a few minutes, depending on the selected duration. The progress is shown in the policy selector menu. Once the results are ready to be presented, you should be able to select the policy experiment like any other published policy and the time series charts showing different flow categories will get updated accordingly.

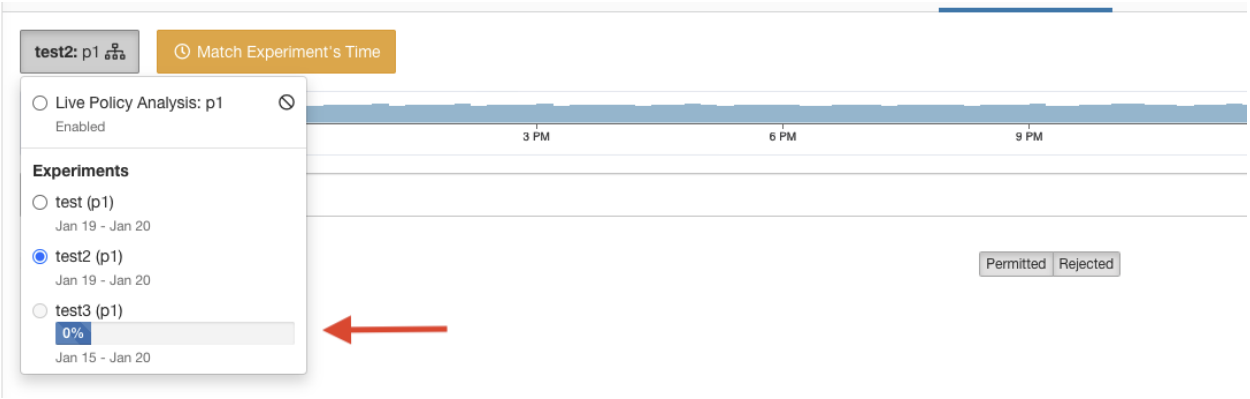


Fig. 6.16.3.3.2: Experiment

NOTE: If you cannot see any flows when selecting a policy experiment, it might be due to time range mismatch, e.g., the current time range of the charts is the past 1 hour, but the experiment duration is 6 hours in the past. In order to reset the time range to the duration of the experiment, click on the clock icon next to the policy selector.

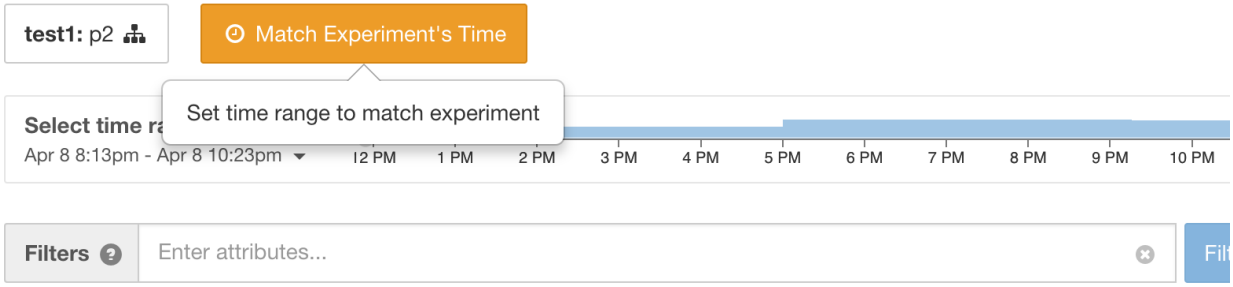


Fig. 6.16.3.3.3: Match time range

6.16.3.4 Activity logs of Policy Analysis

All application users may view activity logs associated with changes done on the policy analysis page in the ADM history (see *History & Diff*).

1. Enable policy analysis

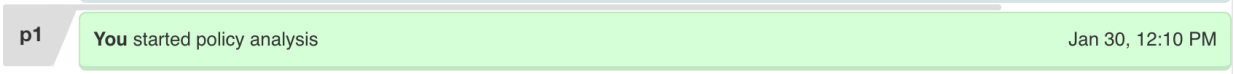


Fig. 6.16.3.4.1: Enable policy analysis

2. Disable policy analysis

Fig. 6.16.3.4.2: Disable policy analysis

3. Update policy analysis

Fig. 6.16.3.4.3: Update policy analysis

6.16.4 Enforcement

Policy Enforcement is similar to *Live Analysis*, except the policies are pushed to the assets in the Scope of the Application and **new firewall rules are written**.

Note: Please familiarize yourself with the concepts in *Live Analysis* before continuing.

Warning: When using this feature **new host firewall rules will be inserted** and any existing rules will be deleted on the relevant hosts.

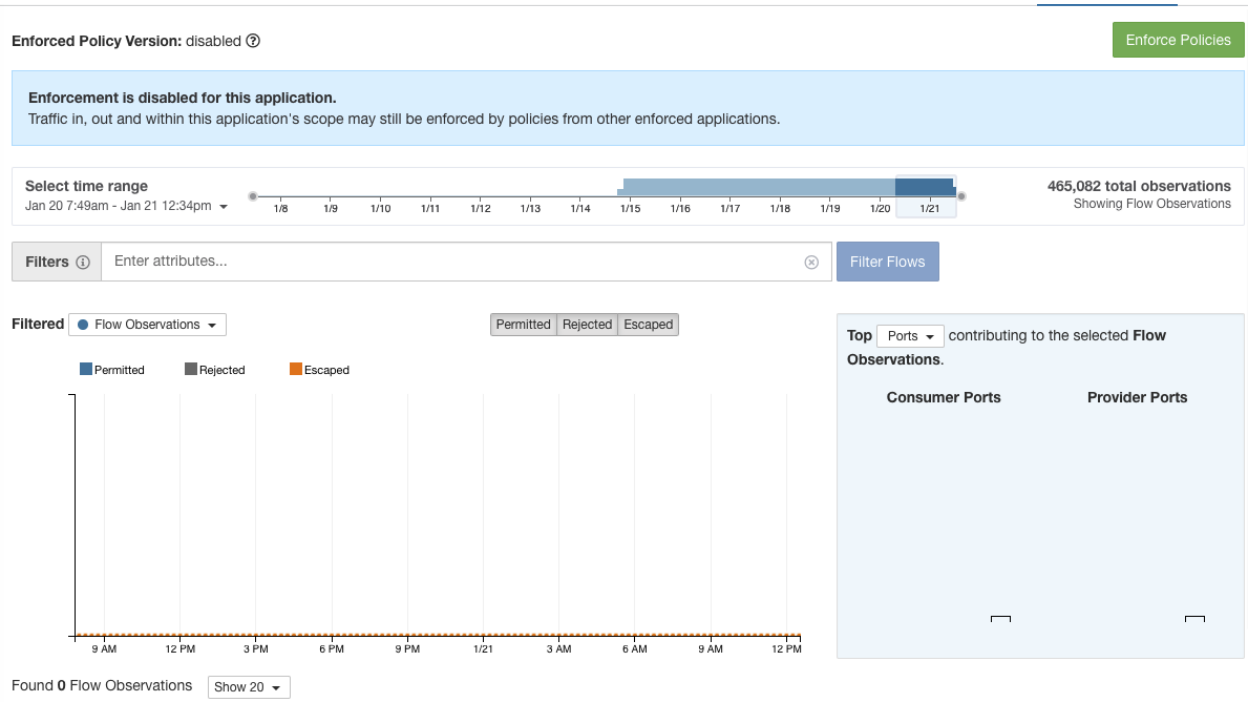


Fig. 6.16.4.1: The Policy Enforcement page with enforcement disabled

6.16.4.1 Enable Policy Enforcement

Policy Enforcement requires users to have the Enforce ability or higher on the Application's Scope. Users with other abilities on the Scope can still view this page but will not be able to enforce (or disable) new policies. For more information about Abilities see *Roles*.

Before Policy Enforcement is enable on an Application, the Policy Enforcement page will show data about how flows are being enforced by policies created in other Applications. For example, a broad "Prod should not talk with Non-Prod hosts" policy may exist in an enforced Application of a parent Scope that is impacting traffic within this Application's Scope.

To enable Policy Enforcement:

1. Ensure the Application is "Primary" for its Scope.
2. Verify the policies are correct using the *Live Analysis* tool.
3. Ensure you have the "Enforce" ability on the Scope of this Application.
4. Navigate to the Policy Enforcement page by clicking the **Policy Enforcement** tab on the right of the header.
5. Click the green **Enforce Policies** button.
6. Inspect the impact of the enforcement and accept the warning indicating that new firewall rules will be written to the hosts.

At this point new firewall rules will be pushed to the assets assigned to the Scope of the Application. The catch-all rule is applied to the workload globally, so may affect interfaces out of scope. A Label Flag will be created at the time of enforcement. See screenshot below.

Note: If no new information is being shown in the Enforcement charts, make sure the correct time range is selected.

Note: It is best practice to analyze policies before enforcing them.

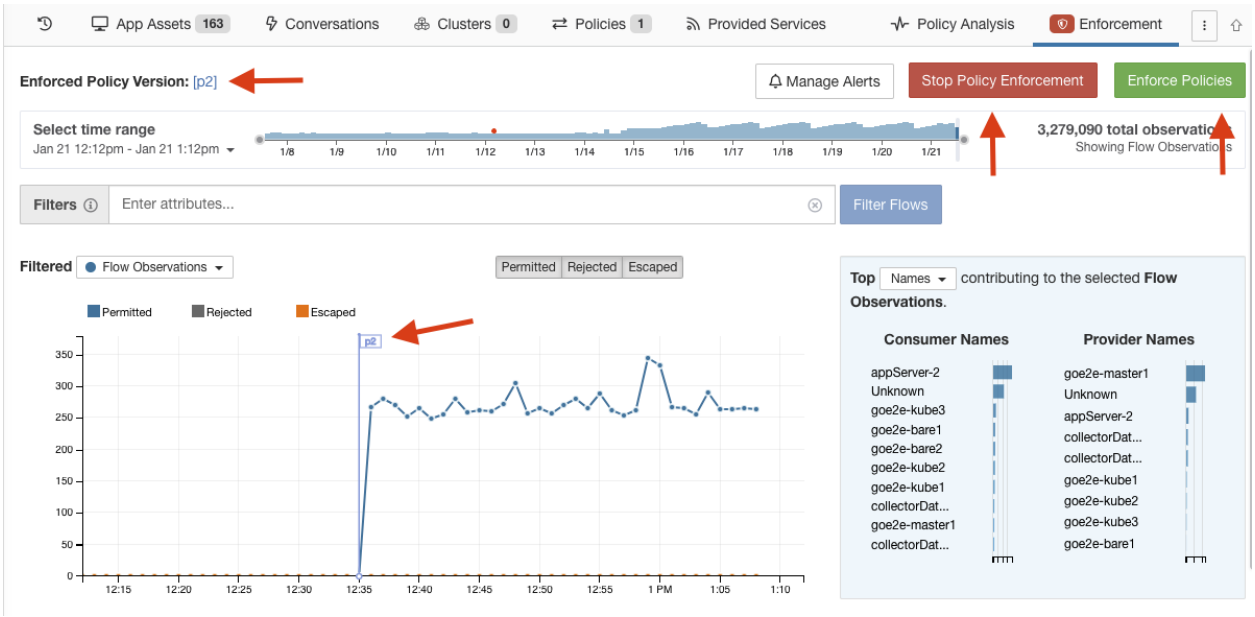


Fig. 6.16.4.1.1: The Policy Enforcement page with enforcement enabled

6.16.4.2 Policy Enforcement Wizard

Policy enforcement wizard brings visibility and predictability into enforced policies before they are implemented on the workloads. It provides a mechanism for selecting policy changes to be enforced (or rollback) and review the potentially impacted workloads within the application workspace.

There are 4 steps in the policy enforcement wizard:

1. Select Policy Updates

You can select which version of policies to be enforced on the application workloads. The difference between the currently enforced policies and policies in the selected version is displayed. If the Latest Version is selected, you have the ability to select a subset of the changes to be enforced. If a previous version is selected (rollback scenario), policy change selection is not allowed. Similarly to the *Policy Diff*, you have the ability to filter and review the policy changes and download them as CSV.

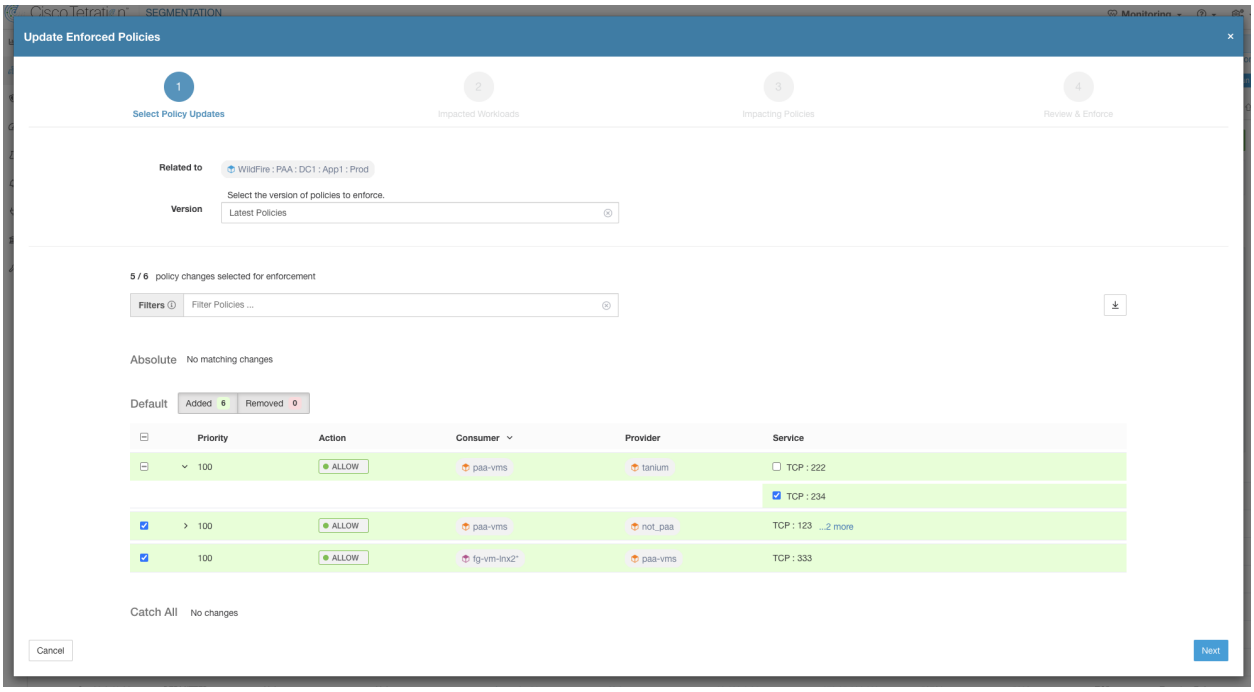


Fig. 6.16.4.2.1: Select policy changes for the enforcement

2. Impacted Workloads

This step shows the impacted workloads that will be affected by the new firewall rules generated from the selected policy changes. The result comes from searching all the workloads that have enforcement agents within the union of the consumers/providers of the selected policy changes. Note that potentially impacted workloads cannot exceed all workloads within the application's scope. However, the actual impacted workloads might be smaller due to other factors such as agent config intents.

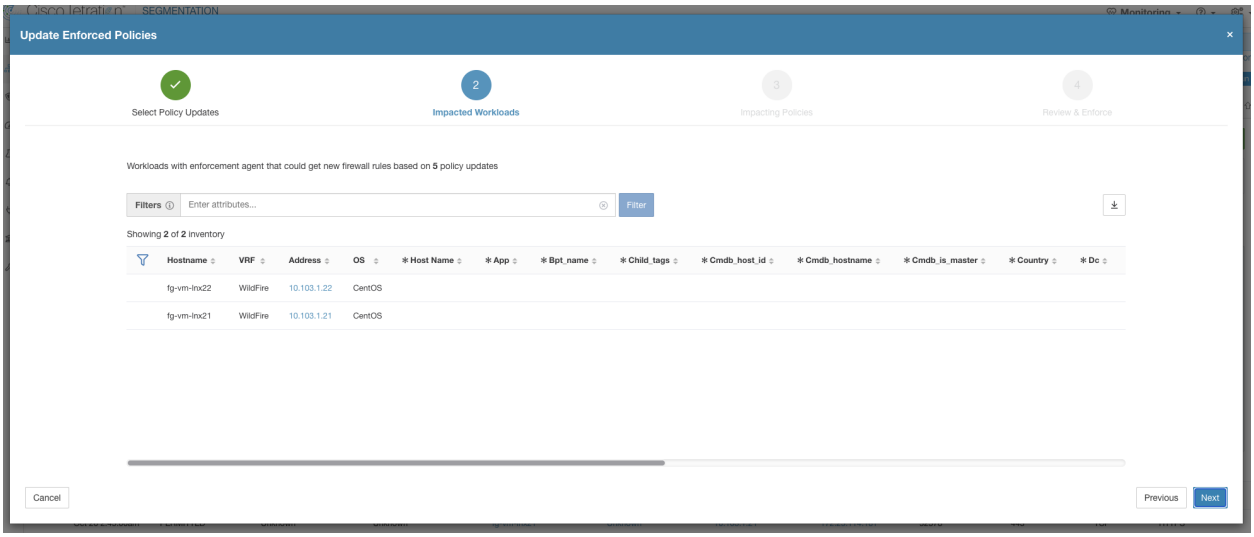


Fig. 6.16.4.2.2: List of impacted Workloads

Please refer to the *Inventory* for more details on viewing, filtering, and downloading inventory items.

3. Impacting Policies

Policies from the ancestor workspaces may have an impact on the workloads in the current application workspace. Therefore, users should make sure the desired allow policies from ancestor workspaces are enforced.

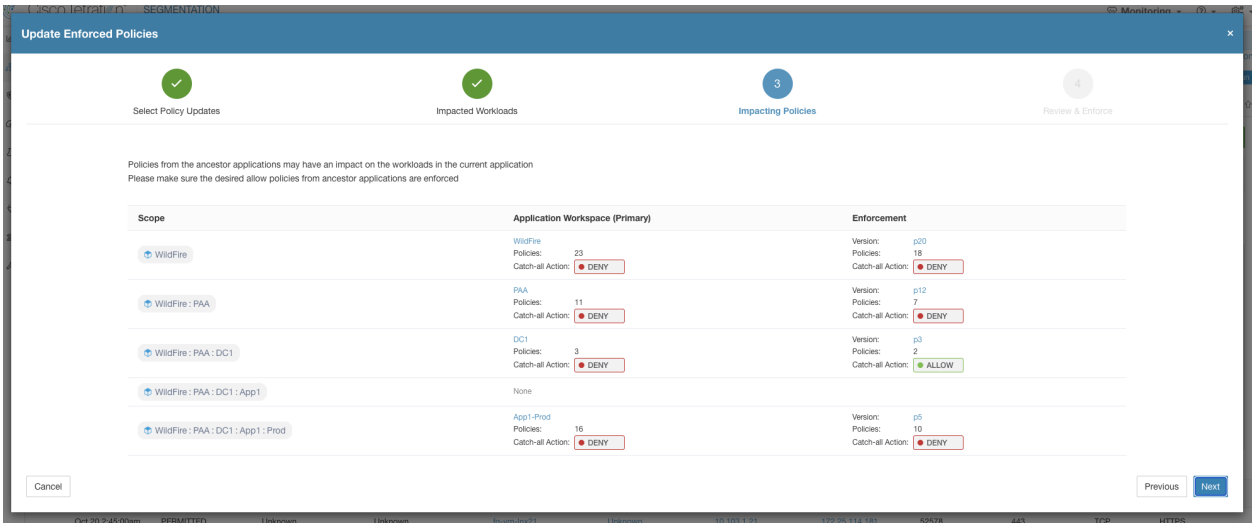


Fig. 6.16.4.2.3: List of ancestor workspaces and enforced versions

4. Review & Accept

This final step provides a summary of policy changes to be enforced, the number of potentially impacted workloads, and the catch-all action that will be enforced. Once the *Accept and Enforce* button is pushed, policy intents will be used to calculate new firewall rules that will be configured on the relevant workloads.

You will have the option to provide a name, description, and reason for action for the newly enforced policies for future reference. Note that in the case of rollback, only setting reason for action is allowed as name and description for a past version cannot be changed.

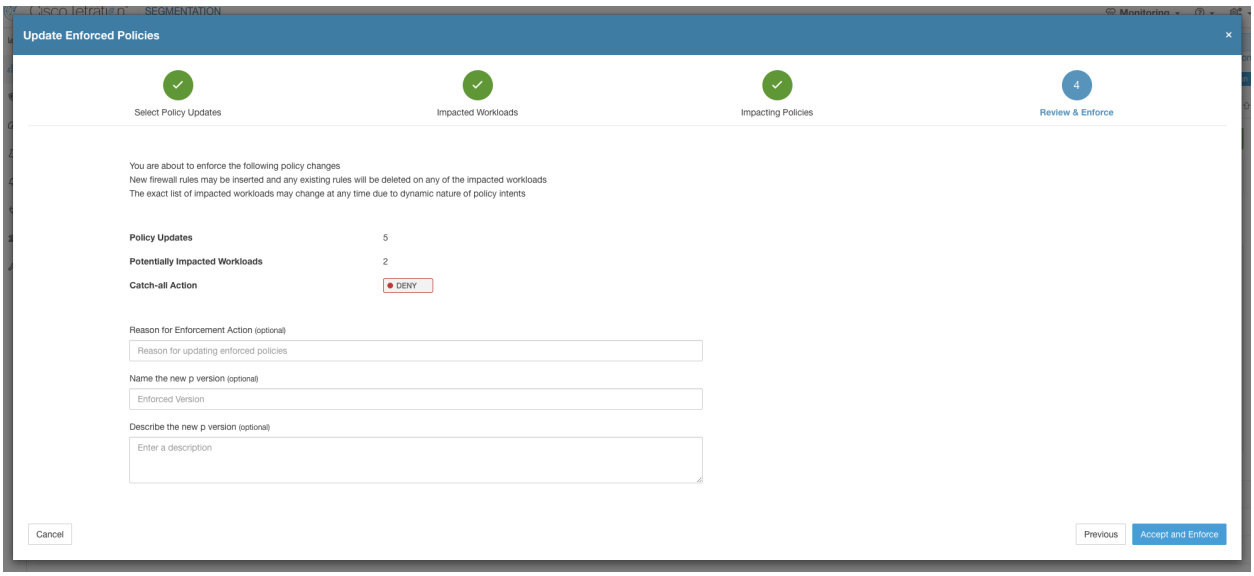


Fig. 6.16.4.2.4: Review the summary and enforce policy changes

6.16.4.3 Viewing Enforced Policies

A new Application version is created when policies are enforced. This version is of the form 'p*' and can be viewed similar to other Application versions. The currently enforced 'p*' version is listed on the left. For example "Enforced Policy Version: [p1]" in the screenshot above.

Clicking the "[p1]" or on a Label Flag in the timeseries chart will switch the Application to that version and show the *Semantics and Viewing*.

6.16.4.4 Enforcing New Policies

Once policies have been published for enforcement it is possible to publish new (improved) policies. This can be done by clicking the **Enforce Latest Policies** button in the upper right of the page. See the screenshot above.

6.16.4.5 Disabling Policy Enforcement

To disable policy enforcement, navigate to the Policy Enforcement page and click the red **Stop Policy Enforcement** button. This will write new firewall rules to assets in the Application's Scope based on other Applications that are enforced. A Label Flag with an 'x' will be created on the timeseries chart. See the screenshot above.

6.16.4.6 Effective Consumer or Effective Provider for a policy

Tetration exposes couple of advanced options in the policy model called effective consumer and effective provider of a policy. To understand these options, it is important to understand the meaning of consumer or provider filter in a policy inside an application workspace. The consumer or provider filters in a policy govern the set of IP addresses that get used in the installed firewall rules as well as the set of workloads with Tetration agents that receive the policy. When Tetration agent receives a policy, the firewall rules are written specific to that workload. This is best illustrated with the following example:

Consider an ALLOW policy with provider filter specifying 1.1.1.0/24 subnet. When this policy is programmed on a Linux workload with IP address 1.1.1.2, host firewall rules look like the following:

1. For incoming traffic firewall rules allow traffic destined to 1.1.1.2 specifically and not to the whole subnet 1.1.1.0/24.
2. For outgoing traffic firewall rules allow traffic sourced from 1.1.1.2 specifically and not from the whole subnet 1.1.1.0/24 (to prevent spoofing).

As a corollary, any agent workloads belonging to the application workspace that do not have IP address within 1.1.1.0/24 subnet will not receive the above firewall rules. However, there can be instances where user(s) need to specify a group of IP addresses that the policy uses in the firewall rules that is different from the workloads that receive the policy. This is where user(s) can use advanced policy options to specify effective consumer and / or effective provider.

We will use an example of configuring policies for a fleet of workloads behind a virtual IP (VIP), similar to keepalived or windows failover clustering solutions, to illustrate the use of this feature.

Consider a fleet of workloads with IP addresses (*172.21.95.5* and *172.21.95.7*) that provide a service sitting behind a VIP - *6.6.6.6*. This VIP is a floating VIP and only one workload owns the VIP at any point in time – goal is to program firewall rules on all the workloads in this cluster to allow traffic to *6.6.6.6*.

In this setup, we have a scope and a corresponding workspace that comprise of the cluster of workloads (*172.21.95.5* and *172.21.95.7*) as well as the VIP (*6.6.6.6*).

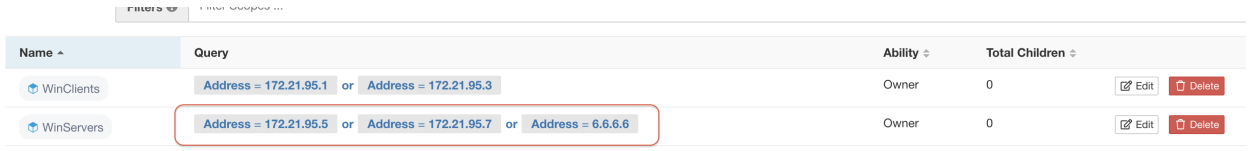


Fig. 6.16.4.6.1: Scopes including VIP and cluster of workloads

VIP is exposed in this application as a provided service as shown below:

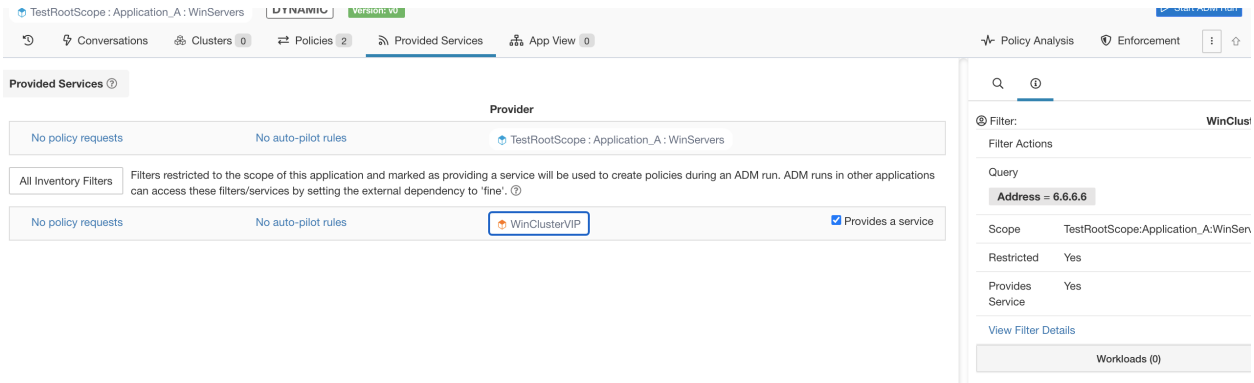


Fig. 6.16.4.6.2: VIP exposed as a provided service

If we were to add a policy from the clients of this service to the service VIP, then (by default) firewall rules allowing traffic to the VIP will only get programmed on the workload that owns the VIP. The issue with this approach is that in case of a failover event, it may take some time for the new workload that subsequently owns the service VIP to get the right firewall rules and application traffic could get disrupted for a brief while.

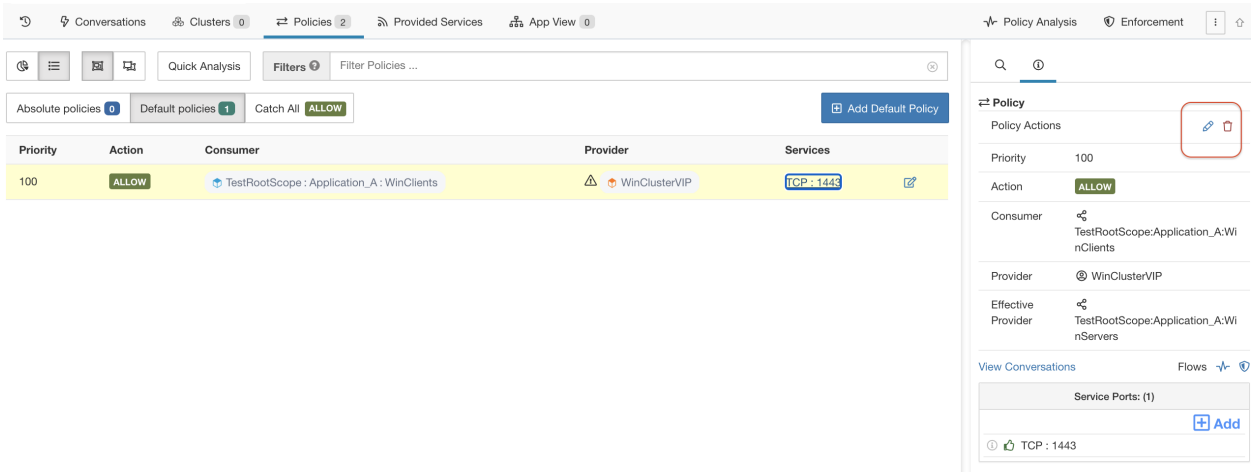


Fig. 6.16.4.6.3: Policy allowing traffic from clients to service VIP

In such scenarios, user(s) can click on the Edit button on top right side of the policy to go to advanced policy options. There are two options available in that widget – Effective Consumer and Effective Provider. For our use case, we set Effective Provider to include the group of workloads where firewall rules allowing traffic to the service VIP need to be programmed – does not matter if any of these workloads own the VIP or not.

When Effective Provider is set, we can see on the workloads that firewall rules allowing traffic to 6.6.6.6 are programmed even when workload does not own the VIP. When all workloads backing the service can be programmed with these rules, we will not see any application traffic disruption during a failover event because the new primary workload (that owns the VIP) will have the necessary firewall rules programmed.

```
$
$ hostname -I | awk '{print $1}'        IP Address of
172.21.95.7                             the server
$                                       part of cluster
$
$ sudo iptables -n --list TA_INPUT   <— Ingress rules
Chain TA_INPUT (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            match-set ta_6c6b4133313438ff5429ca8c14b6 src match-set ta_ac2618d307e4e7dbb76b96c0df3f dst mul
tiport dports 1443 ctstate NEW,ESTABLISHED /* PolicyId=DEFAULT:100:ALLOW:5ed53fe8497d4f26444d50b3:5ed5435b497d4f26414d50b1:6 */
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
$
$
$ sudo iptables -n --list TA_OUTPUT   <— Egress rules
Chain TA_OUTPUT (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0            match-set ta_ac2618d307e4e7dbb76b96c0df3f src match-set ta_6c6b4133313438ff5429ca8c14b6 dst mul
tiport sports 1443 ctstate ESTABLISHED /* PolicyId=DEFAULT:100:ALLOW:5ed53fe8497d4f26444d50b3:5ed5435b497d4f26414d50b1:6 */
RETURN     all  --  0.0.0.0/0            0.0.0.0/0
$
$
$ sudo ipset list ta_ac2618d307e4e7dbb76b96c0df3f
Name: ta_ac2618d307e4e7dbb76b96c0df3f
Type: hash:net
Revision: 3
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16816
References: 2
Members:
6.6.6.6              <— VIP
$ sudo ipset list ta_6c6b4133313438ff5429ca8c14b6
Name: ta_6c6b4133313438ff5429ca8c14b6
Type: hash:net
Revision: 3
Header: family inet hashsize 1024 maxelem 65536
Size in memory: 16848
References: 2
Members:
172.21.95.1
172.21.95.3       <— Client IPs
$
```

Fig. 6.16.4.6.4: Firewall rules on the host allowing traffic to service VIP

6.16.4.7 Enforcement on Containers

Tetration supports enforcing policies inside container workloads managed by Kubernetes and OpenShift. This requires an external orchestrator configuration to be added for Kubernetes/OpenShift API server and Enforcement agents to be used on one of the supported platforms. See *External Orchestrators* and *Deploying Software Agents* for more details.

> **Attention: Agents running on Kubernetes/OpenShift hosts have to be configured to preserve existing rules.**
>
> In order for the Enforcement agent not to interfere with iptables rules added by Kubernetes, the agent has to be configured with a profile that has the *Preserve Rules* option enabled. See *Creating an Agent Config Profile*

When enforcing policies on containers, Tetration allows Kubernetes/OpenShift service abstractions to be used as providers. Internally, the policies for service abstractions are transformed into rules for the provider pods and the nodes they are running on. This transformation depends on the type of the Kubernetes/OpenShift service, and it is dynamically updated whenever changes are received from the API server.

The following example illustrates the flexibility made possible by this feature. Consider the following policy which allows traffic from all hosts and pods with the label *environment* = *prod* to a Kubernetes service of type *NodePort* with the name *db* which exposes TCP port 27017 on a set of pods.

Consumer	Provider	Protocol/Port	Action
environment = prod OR orchestrator_environment = prod	orchestrator_system/service_name = db	TCP 27017	Allow

This policy would result in the following firewall rules:

- On hosts and pods annotated with *environment = prod*, allow outgoing connections to all Kubernetes nodes of the cluster to which the service belongs. This rule uses the node port assigned to this service by Kubernetes.
- On pods with the label *environment = prod*, allow outgoing connections to the ClusterIP assigned to this service by Kubernetes. This rule uses the port exposed by the service (TCP 27017).
- On Kubernetes nodes of the cluster to which the service belongs, allow outgoing connections to the provider pods. This rules uses the target port exposed by the service (TCP 27017).
- On pods providing the service db, all incoming connections from all kubernetes nodes and consumer hosts and pods. This rule uses the target port exposed by the service (TCP 27017).

Changes to the type of the service, ports and set of provider pods will immediately be picked up by Tetration rule generator and used to update the generated firewall rules.

Warning: Policies including Kubernetes/OpenShift items need to be designed carefully to avoid conflicting with the internal operation of the kubernetes cluster.

Kubernetes/OpenShift items imported by Tetration include the pods and services constituting the kubernetes cluster (e.g. pods in the kube-system namespace). This allows precise policies to be defined to secure the kubernetes cluster itself, but it also means that badly designed policies can affect the operation of the cluster.

6.16.4.8 Pausing policy update

To prevent rule update in all enforcement endpoints, go to the *Enforcement Status* to pause or un-pause. This feature is reserved for site admin and customer support.

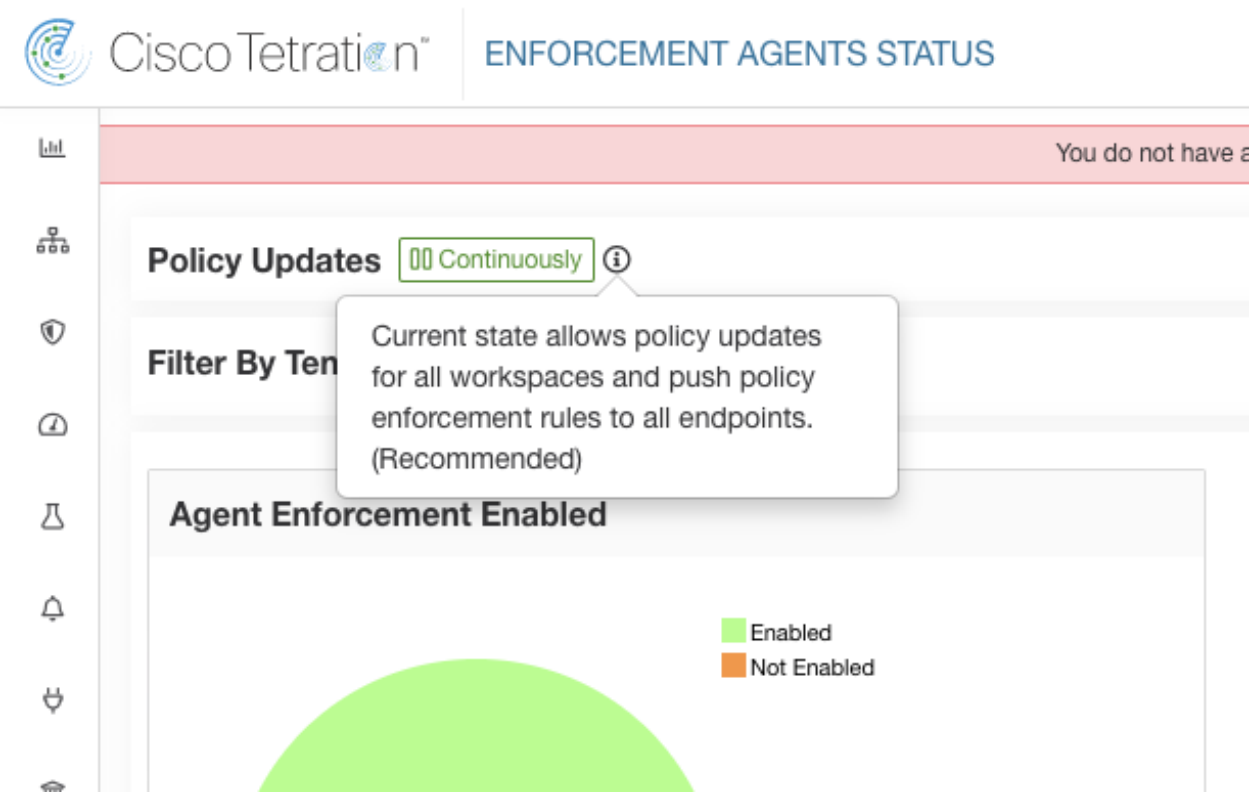


Fig. 6.16.4.8.1: Rule update continuously to

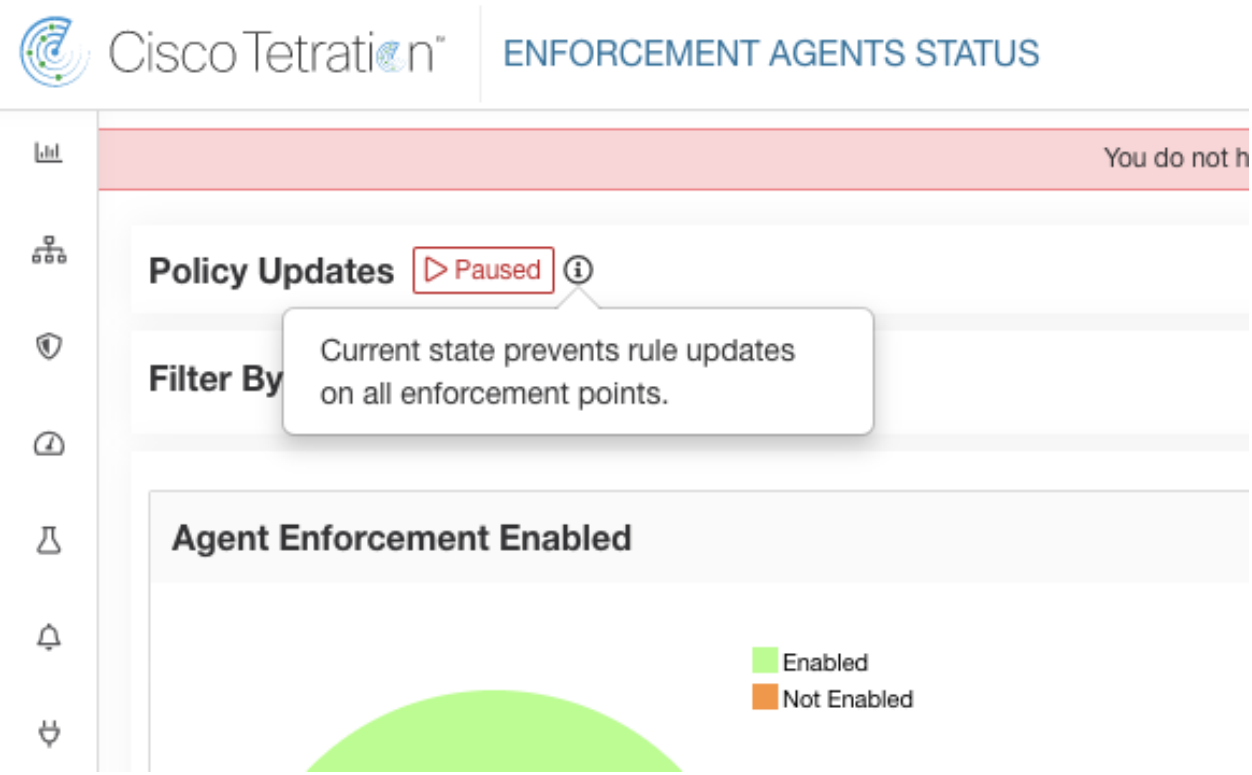


Fig. 6.16.4.8.2: Rule update paused

6.16.5 Collaboration Among Applications

The provided services page is a **collaboration tool** to help application owners build tight security policies *across* applications with inter-dependencies.

For example, consider an Authentication application that consists of multiple tiers and services. This Authentication application serves as infrastructure for many other applications which require access to a certain set of auth servers on a certain port.

Once a dependent (consumer) application e.g. HR creates a policy to consume the auth service from the authentication application, it only affects the outbound rules of the HR machines. This is due to the scope of the policy being limited to the HR application. In case both ends are analyzing or enforcing policies, both ends need to allow such flows. In this scenario, the HR owner (the consumer) needs the owner of the Authentication application (the provider) to create a policy opening up access to auth servers on the correct port. To create a policy that allows the flow from the provider side, the owner of Authentication workspace can manually accept the connector request(s) (connection requests) that are sent to it (see below), set up auto-pilot rules to accept such, or run ADM on her workspace (with appropriate time range so the corresponding flows are seen) and publish the policy.

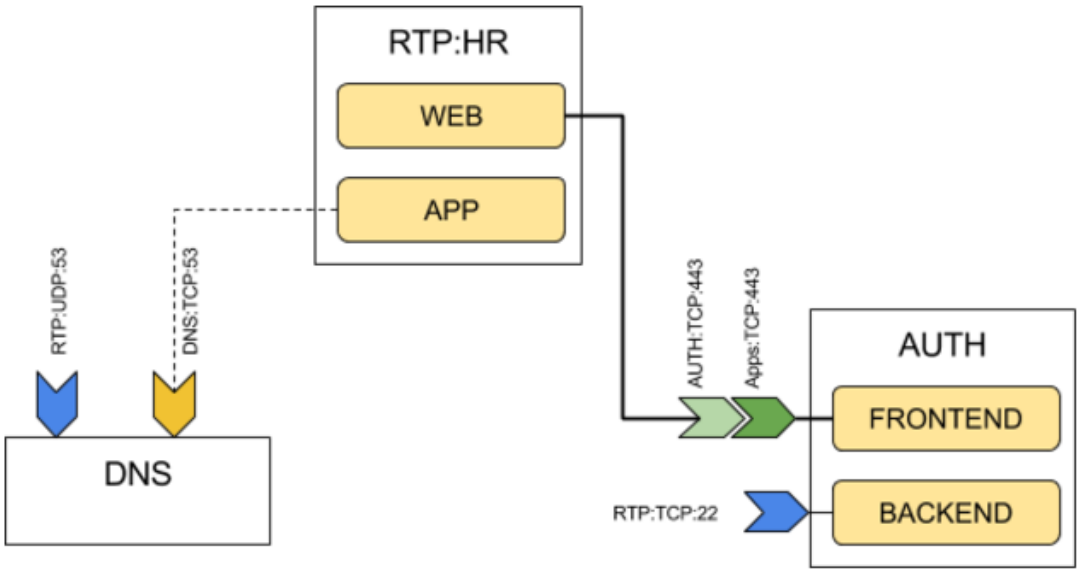


Fig. 6.16.5.1: Providing Services to External Applications

The **Provided Services** page facilitates these interactions allowing for app owners to collaboratively build security policies that only grant access to dependent applications (that is, fine-grained policies).

The provided services page shows a list of current connection requests to the application, indicating the (external) consumer application making the request(s), and which internal provided service (possibly the whole scope) the request is being made to.

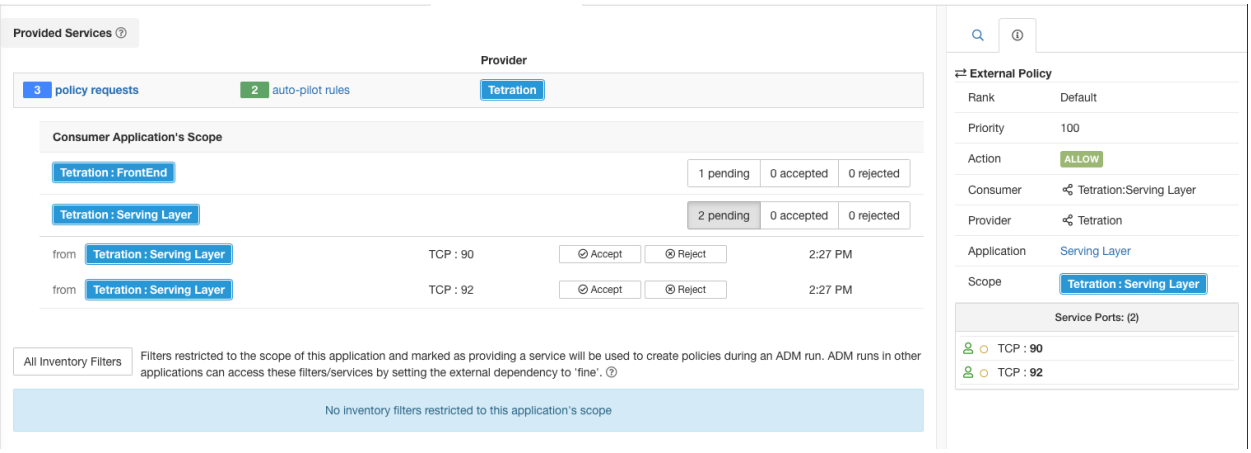


Fig. 6.16.5.2: Connection Requests from External (consumer) Applications to Provided Services

Notes:

- The provided services page is only available to primary applications. This is to ensure that isolated experiments on secondary applications do not create notifications on other primary applications.
- The small number next to the Provided Services tab represents the total number of pending policy requests as

described below.

- If an external scope does not have a workspace, no requests are sent (for example, this could be the case for the root scope, or any scope defined for workloads outside the organization). If an external scope has not published any policy, policy analysis and enforcement are carried out on the consumer end only.
- **No connection requests for consumers**: If a consumer workspace is analyzing or enforcing policies, it has to explicitly include policies that allow all its legitimate consuming flows, either through ADM runs or explicit manually crafted policies (no connection requests from external provider workspaces are generated to it).

6.16.5.1 Provided Services

Services that are marked as *provides a service* are for consumption by other applications (in other/external scopes). In other words, the application owner is presenting which services, within his/her scope, are to be used by dependent applications.

Check the **provides a service** box to mark a filter as public and providing a service. The user can also promote a cluster to an inventory filter and in the process make it a provided service (and to make it accessible/visible to other scopes). The benefit of promoting inventory filters and clusters to provided services is that makes it possible to create and manage finer grain or tighter (more secure) policies among applications. Otherwise, external or inter-scope policies would be limited to the higher (coarser) granularity of scopes. Note also that making a service with existing policy requests private, does not affect the state of existing policy requests. It may just avoid future policy requests from ADM runs.

Provided services can be used as candidates for **External Dependencies** when performing ADM runs (for policy generation) on other applications. The *dependent filters* shown on external dependencies panel in the ADM run page, when clicking on the **Fine** button next to a scope, are based on filters that are restricted to their scope and marked as providing a service by the external scopes. Thus, if owner of workspace A wants to generate policies, upon an ADM run, to the provided services in workspace B, owner of A should choose **Fine** next to scope B. Otherwise, only policies to the whole scope of B would be generated, and the owner of B may reject those policy requests.

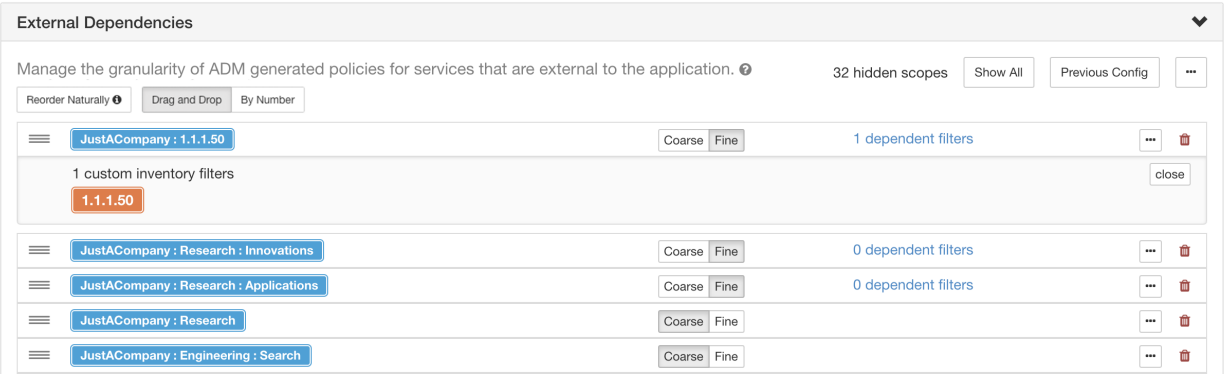


Fig. 6.16.5.1.1: Provided Services as Fine-grained External Dependencies in ADM run page.

Note: The scope of an application is always public, since other applications from the same tenant can always create policies with scopes as the provider. This is consistent with the coarse granularity for external dependencies when configuring ADM runs.

6.16.5.2 Policy Requests

Each time a policy is created in a primary application where the provider is from another (external) primary application, unless a published policy that allows corresponding flows exists in that application, a policy request is delivered as a

notification to the provider application. Seeing the notifications helps the owner of the provider application to open up necessary services for other dependent applications as the applications evolve.

The following conditions should hold at the time of policy creation for a policy request to be sent:

1. The original (consumer) policy must be created in a primary application (in the consumer's application)
2. The policy must have ALLOW action
3. Provider of the policy must be in another primary application (e.g., an external scope or a provided service)
4. There is no existing matching policy under the provider application

In the following example, the FrontEnd app is creating two policies on TCP port 22 and UDP Port 514 from **FrontEnd** scope to **Tetration** scope. The Serving Layer app is creating two policies on TCP port 90 and UDP Port 92 from **ServingLayer** scope to **Tetration** scope.

Two policy requests are immediately sent to the Tetration Workspace (primary application with Tetration scope), and the policies on FrontEnd App and ServingLayer app are shown with a pending status.

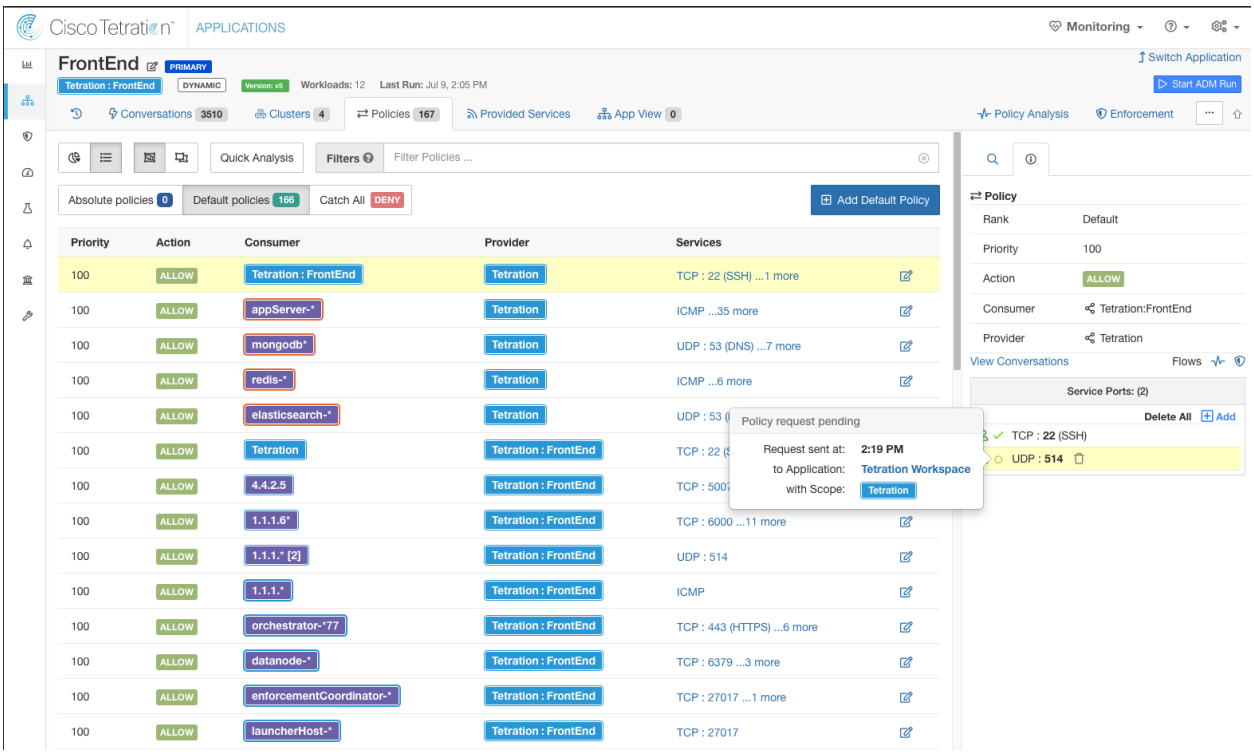


Fig. 6.16.5.2.1: Policies created in consumer application and status shows as Pending (FrontEnd)

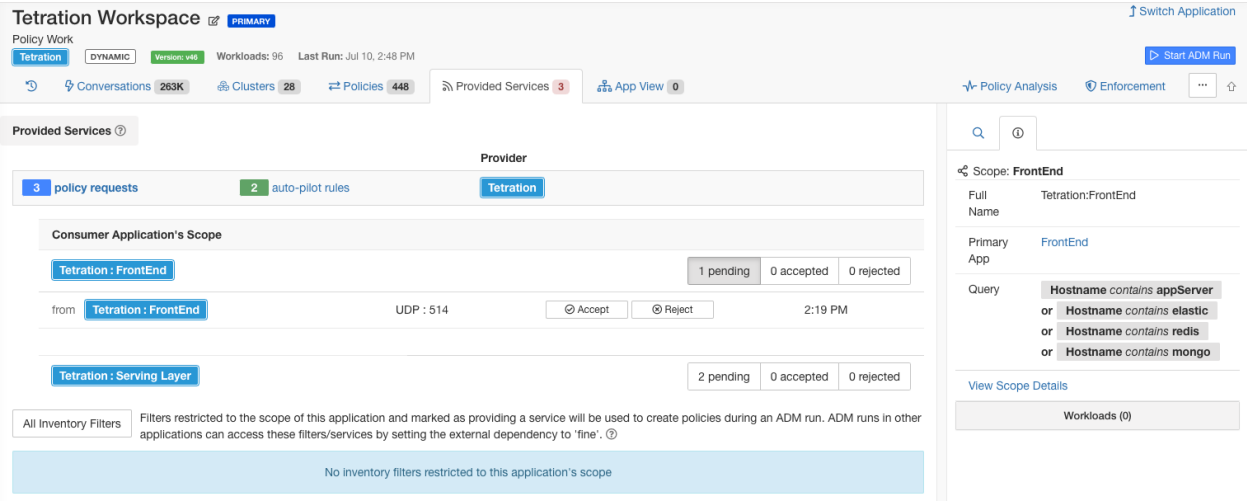


Fig. 6.16.5.2.2: Pending Policy Requests on provider application (Tetration Workspace)

6.16.5.3 Accepting/Rejecting Policy Requests

To accept or reject a policy request, click on the **Accept**, or **Reject** button next to each policy request. Alternatively, you can perform these operations in bulk for each service using **Accept All** and **Reject All** buttons.

Accepting a policy request on a service is equivalent to creating a policy from the requested filter as the consumer to the service as the provider. Additionally, upon accepting a policy request, the original policy from the consumer application (FrontEnd App and Serving Layer) will be marked as accepted (see figures below)

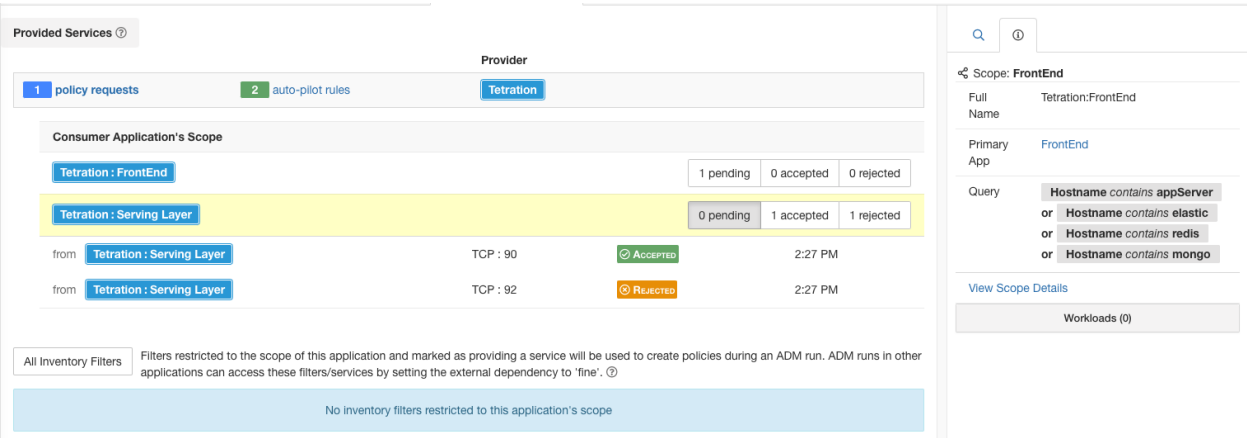


Fig. 6.16.5.3.1: Accepting/Rejecting policy requests

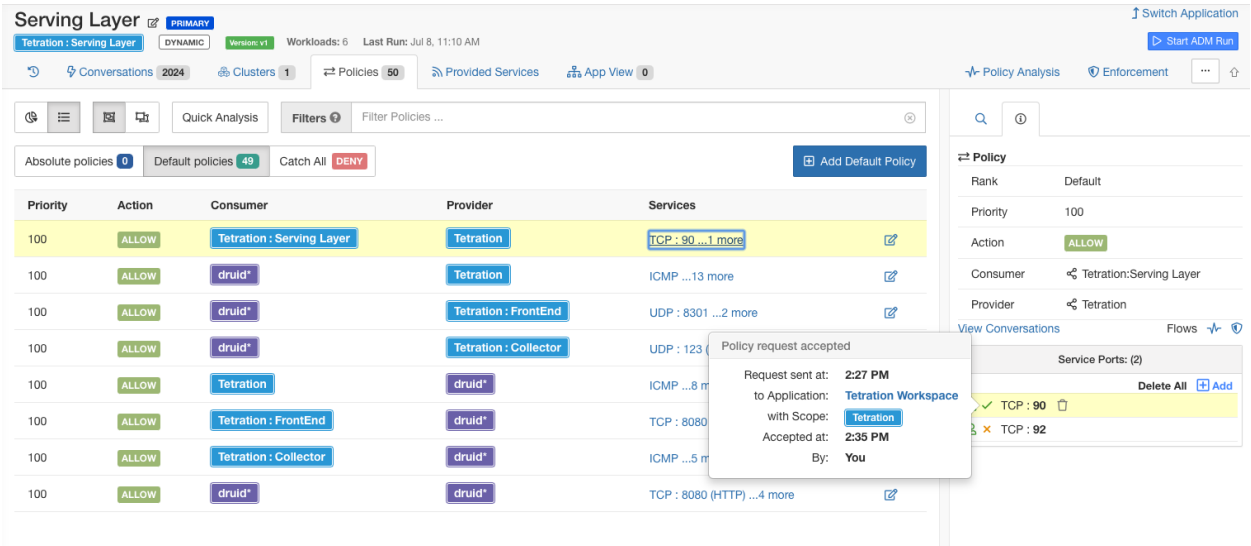


Fig. 6.16.5.3.2: Policy status shown as Accepted

The new policy created on the provider application e.g. Tetration is marked with a **plus** icon indicating that this policy was created due to an external policy request.

Note: If the original policy on the consumer side is deleted after the policy request is accepted, the policy on provider side will not be deleted. However, the tooltip next to the policy shows the original policy as deleted with the timestamp of the event:

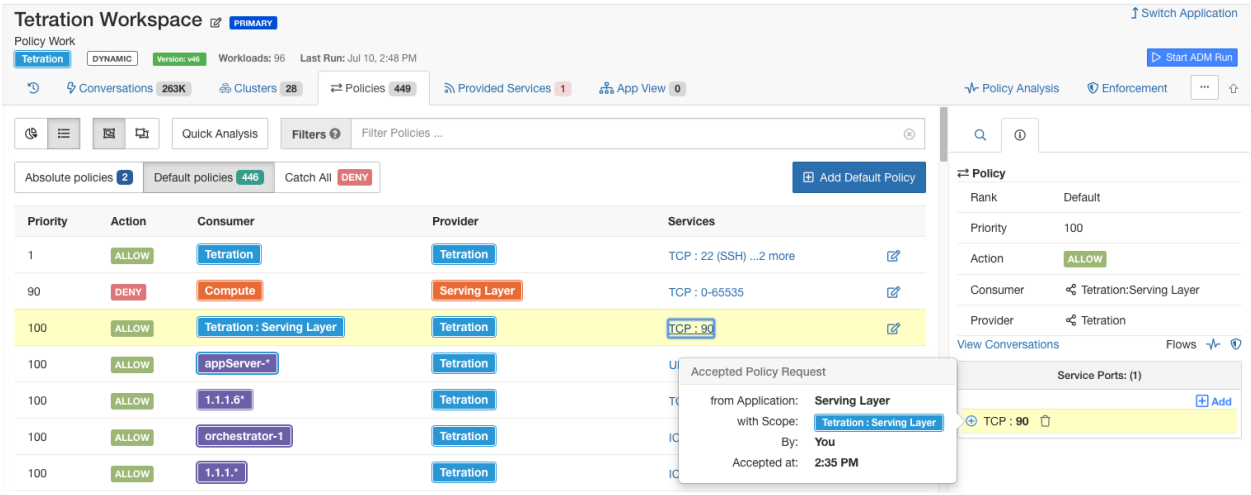


Fig. 6.16.5.3.3: Provider side policy, created by accepting a policy request

Rejecting a policy request does not create or update any policies. The original policy from the consumer application (Serving Layer App) will be marked as rejected, but the policy remains in effect, i.e., outbound traffic still will be allowed. The tooltip next to the reject policy has information about the provider application, the user that rejected the policy request as well as the time of the rejection.

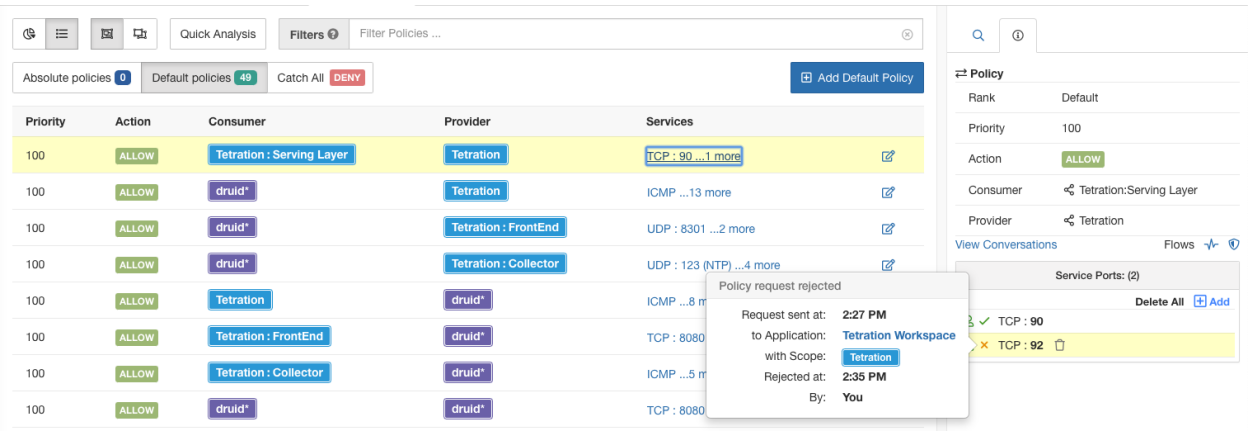


Fig. 6.16.5.3.4: Policy status shown as Rejected

6.16.5.4 Resolved Policy Requests

If the first 3 conditions for creating a policy request are met, but there is a matching existing policy on the provider application, the policy created on the consumer application will be marked as resolved indicating that the provider application is already allowing the traffic through the requested port.

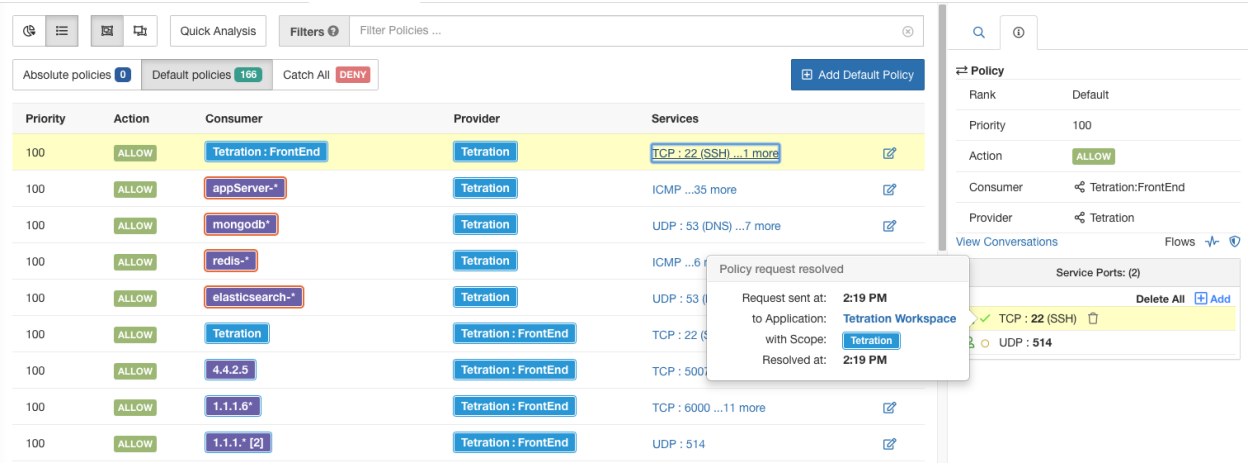


Fig. 6.16.5.4.1: Policy status shown as Resolved

6.16.5.5 Auto-pilot Rules

Infrastructure applications that provide services to many other applications in a datacenter are prone to a flood of policy requests from other applications.

Auto-pilot rules are designed to limit the manual steps necessary to accept or reject large number of policy requests by pre-provisioning a set of simple rules that will be used to automatically accept/reject policy requests with a certain pattern.

Note: Auto-pilot rules must be provisioned before the policy requests are delivered. Creating auto-pilot rules does not automatically result in accepting/rejecting the currently pending policy requests.

Click on the **auto-pilot rules** link on the service row to open/close the auto-pilot panel.

Click on the **New Auto-pilot Rule** button to create a new rule. Each autopilot rule specifies whether we should automatically accept/reject a policy request from a certain scope on a particular port range.

Note that any policy request where the consumer is guaranteed to be contained in the specified scope will be a match for auto-pilot rule. For example, any sub-scope, filter restricted to the scope or sub-scopes, and any cluster in a primary application of the specified scope or sub-scope will be a match as well.

In the example below, we create a new auto-pilot rule to reject TCP policy requests in port range 1-200 from any consumer contained in Tetration:Adhoc to the provider service Tetration

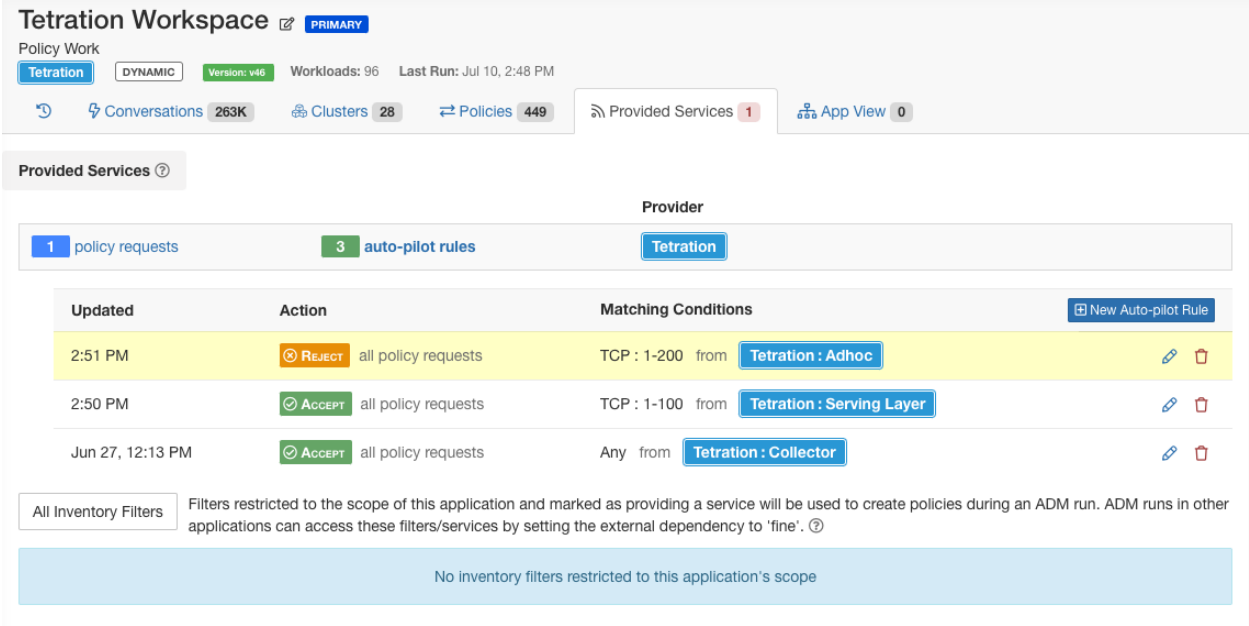


Fig. 6.16.5.5.1: Creating/Updating Auto-pilot rules

Then we create a new policy in application workspace *FrontEnd App* on TCP port 23. Since the policy is a match for the auto-pilot rule, it will be automatically rejected. The status and reason for policy rejection is indicated on the tooltip next to the rejected policy.

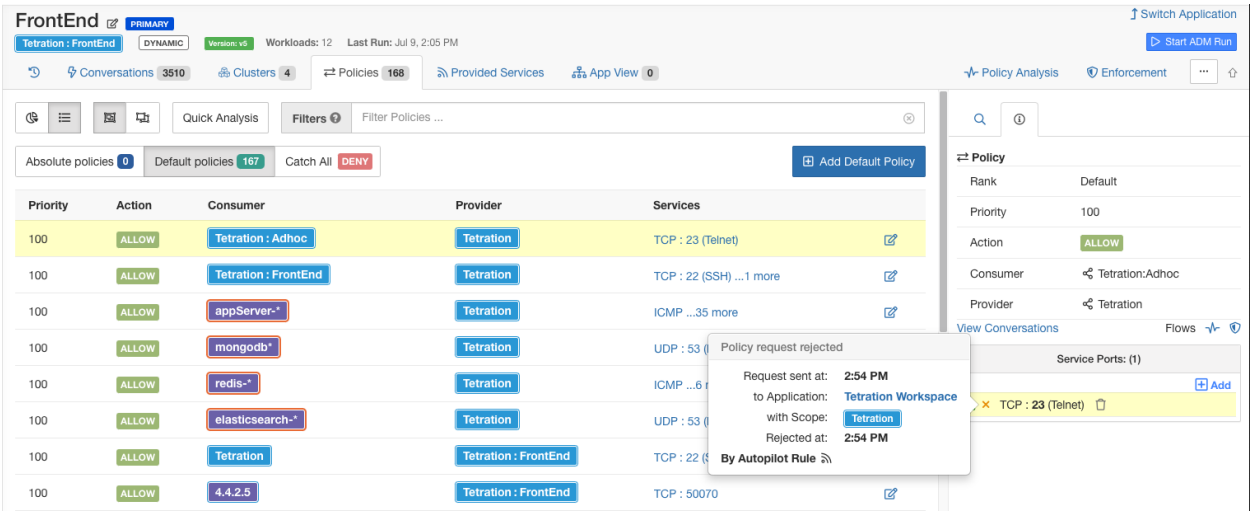


Fig. 6.16.5.5.2: Policy automatically getting rejected by Auto-pilot rule

6.16.5.6 Auto Accept Policy Connectors

Auto accept outgoing policy connectors option allows users to auto accept any policy connector request created as part of an ADM run, manual policy creation or application import.

Note: This option is only available for root scope owners.

In order to set this option click on **Default ADM Run Config** button on the Applications list page.

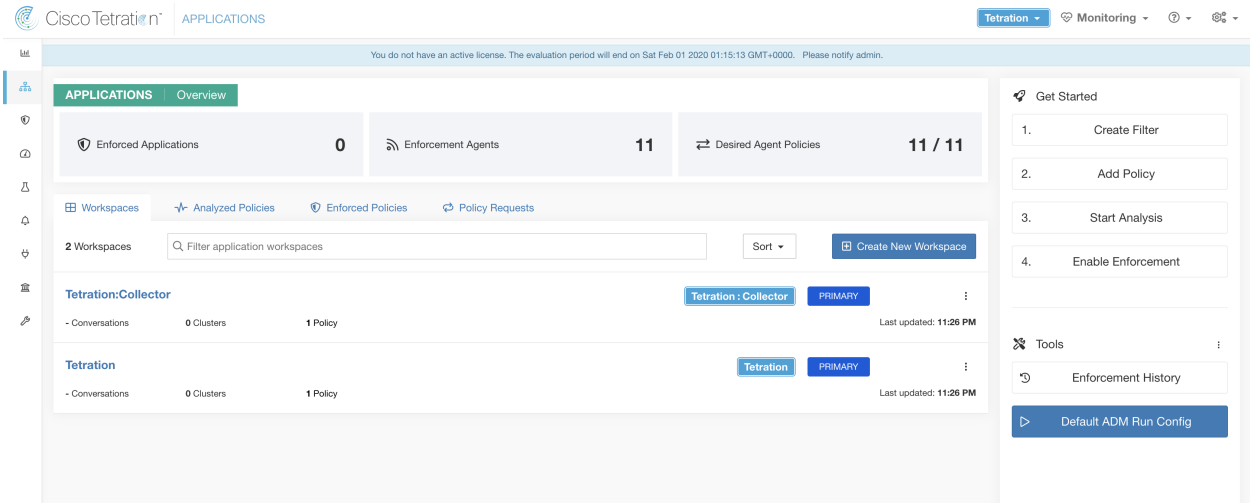


Fig. 6.16.5.6.1: Default ADM Run Config

Select the **Auto accept outgoing policy connectors** option and click on the **save** button.

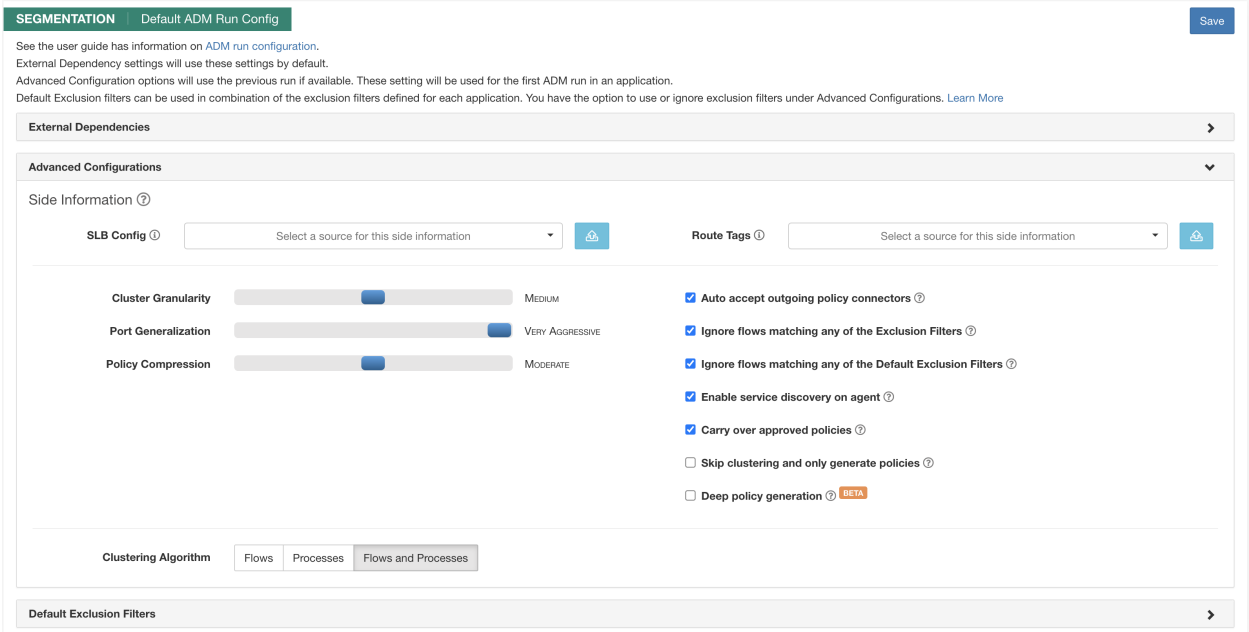


Fig. 6.16.5.6.2: Select Auto accept outgoing policy connectors option

Once this option is set any policy request created in the root scope will be auto accepted.

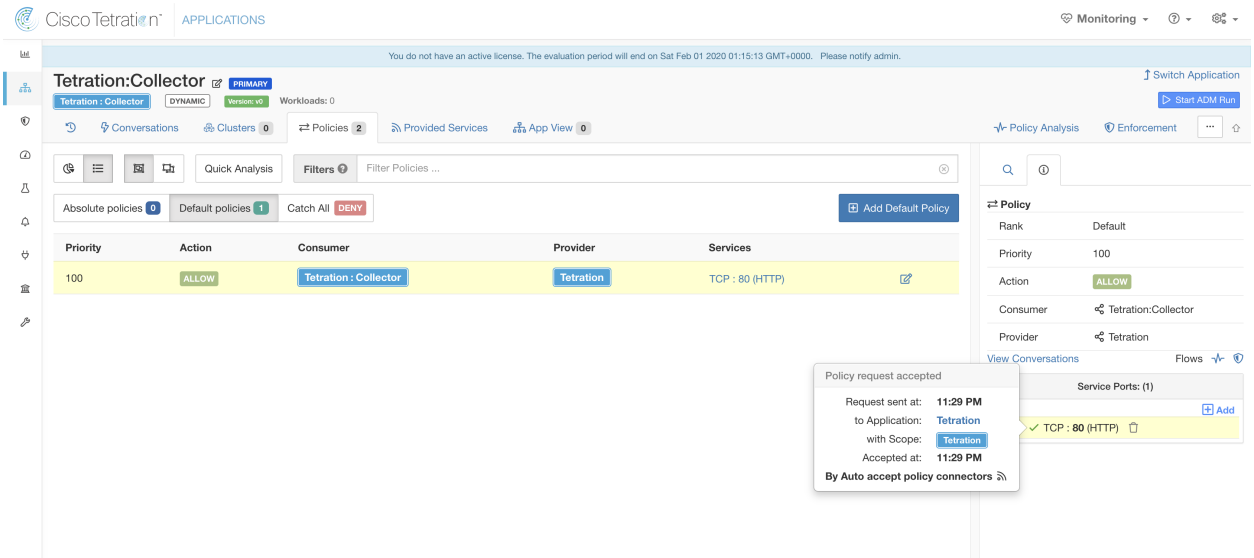


Fig. 6.16.5.6.3: Policy automatically getting accepted by Auto accept policy connectors

6.16.6 Policies Publisher

Policies Publisher is an advanced Tetration feature allowing third party vendors to implement their own enforcement algorithms optimized for network appliances such as load balancers or firewalls. This feature is realized by publishing defined policies to a Kafka instance residing within Tetration cluster and by providing customers with Kafka client certificates, which allows third party vendor code to retrieve policies from Kafka and to translate them into their network appliances configuration appropriately.

This section aims to describe the procedure third party vendors, in short users in the following, need to perform in order to exploit the *Policies Publisher* feature with Java on Linux.

6.16.6.1 Prerequisites

Linux system with eg. Ubuntu 16.04 with following software packages installed:

- Java 8 JDK
- Apache Kafka Clients: kafka-clients-1.0.0.jar
- Protocol Buffers Core: protobuf-java-3.4.1.jar
- Apache Log4j: log4j-1.2.17.jar
- Simple Logging Facade for Java: slf4j-api-1.7.25.jar, slf4j-log4j12-1.7.25.jar
- Snappy compressor/decompressor for Java: snappy-java-1.1.4.jar

6.16.6.2 Getting Kafka client certificates

- Create a user role with capability *"Owner"* and assign it to a user account of choice:

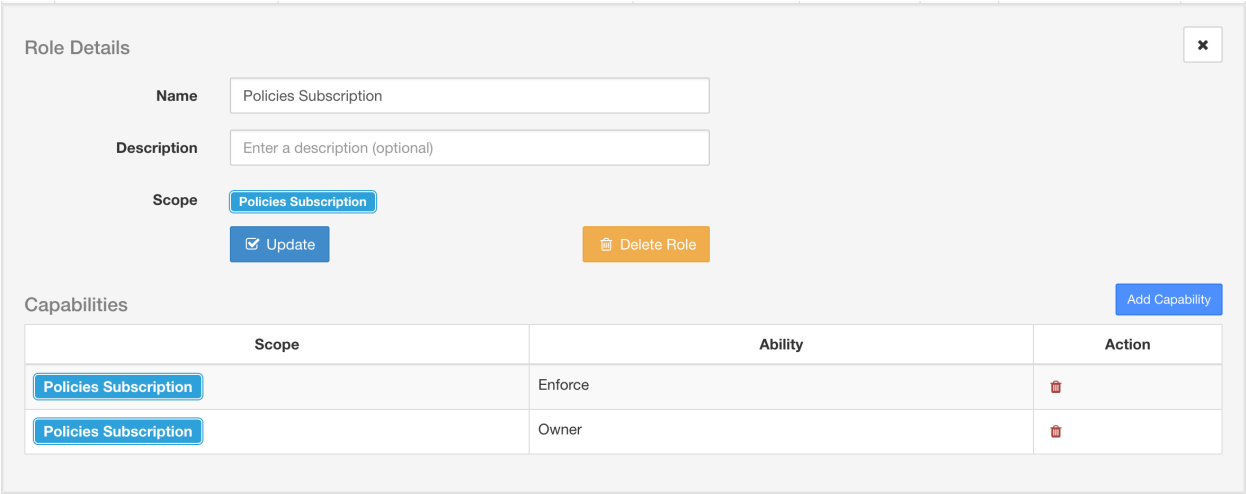


Fig. 6.16.6.2.1: User role configuration to receive policies from Kafka

- Perform policies enforcement as described in *Enforcement*. This first step is necessary as it will create a Kafka topic associated with active scope.
- Navigate to sub-menu *"Applications" - "Data Platform Admin"*:

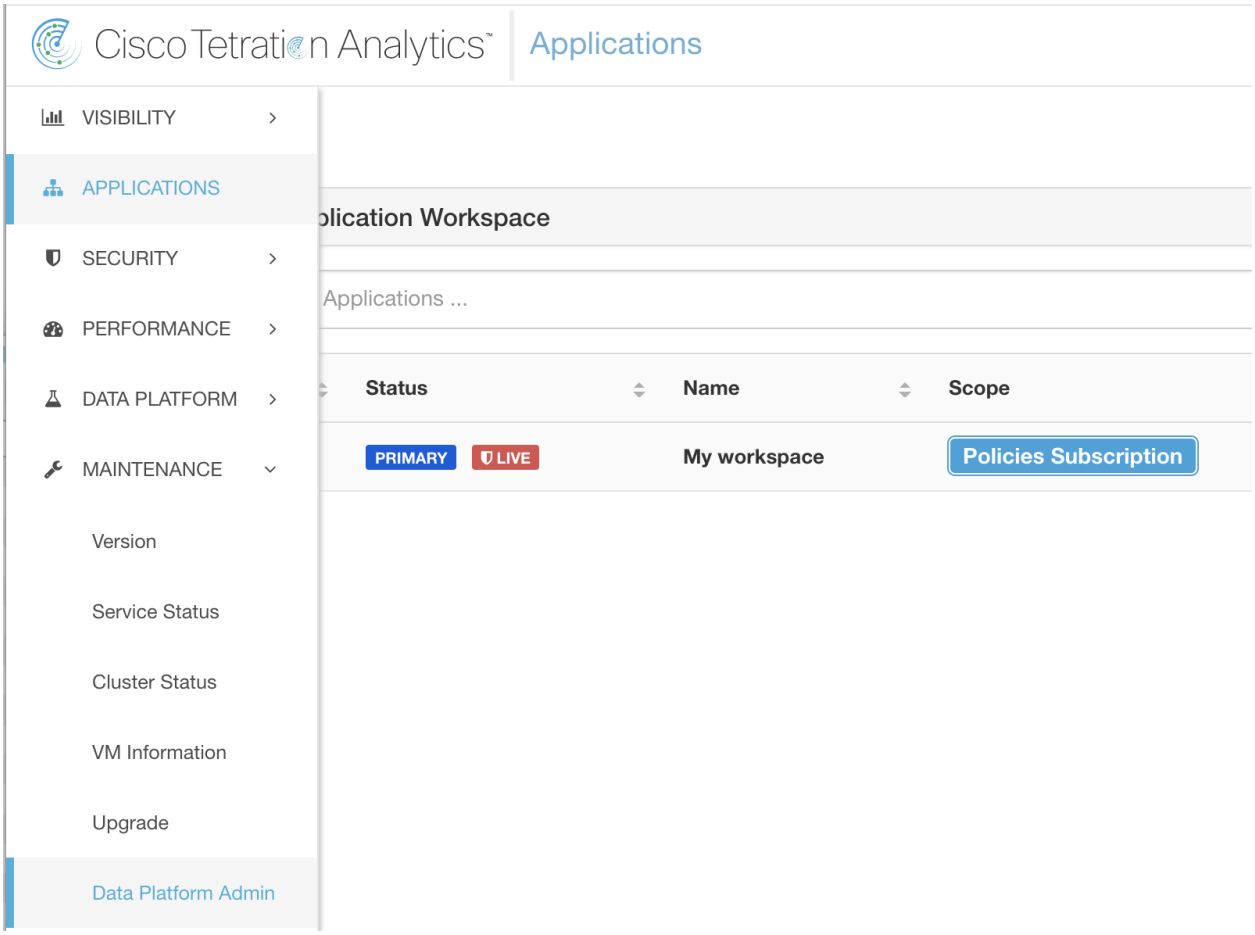


Fig. 6.16.6.2.2: Sub-menu Data Platform Admin

- Select the tab *"Data Taps"* and download Kafka client certificates by clicking on the download button under column *"Actions"*. Make sure to select the *Java Keystore* format in the download dialog.

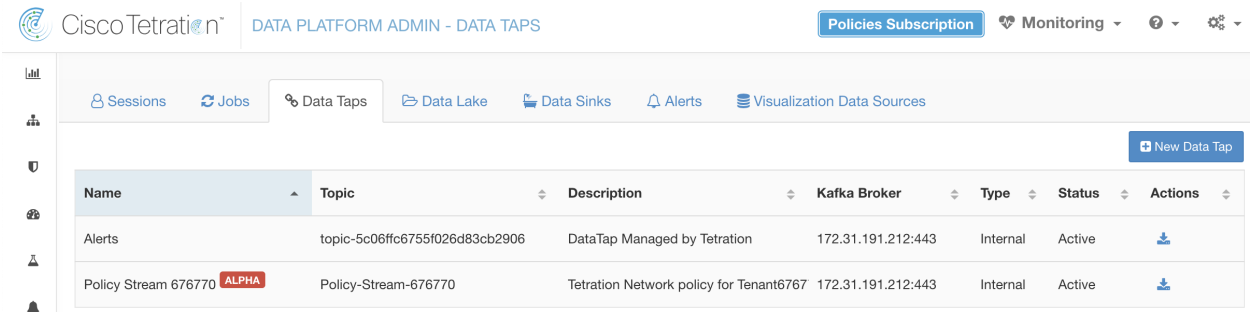


Fig. 6.16.6.2.3: Data Taps view

- The downloaded clients certificates file usually has a name like *Policy-Stream-10-Policies-Subscription.jks.tar.gz*. Create a directory and unpack it underneath the created directory as below:

```
mkdir Policy-Stream-10-Policies-Subscription
tar -C Policy-Stream-10-Policies-Subscription -zxf Policy-Stream-10-Policies-
↪Subscription.jks.tar.gz
```

6.16.6.3 Protobuf definition file

The network policies exposed by Tetration backend to Kafka are encoded in Google Protocol Buffers format. Refer to this guide for instructions how to download and install it on your Linux system.

The proto file of Tetration network policy can be downloaded from here.

6.16.6.4 Data Model of Tetration Network Policy

Picture below shows a simplified UML diagram of Tetration entities exposed to Kafka:

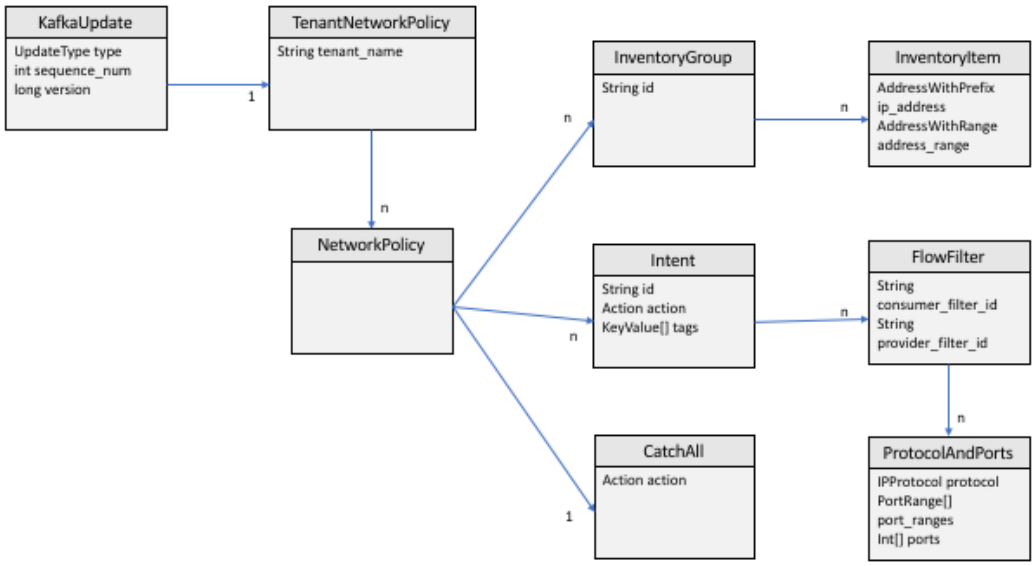


Fig. 6.16.6.4.1: Data Model of Tetration Network Policy

A *Tetration Network Policy* as modeled in protobuf consists of a list of *InventoryGroups*, a list of *Intents* and a *CatchAll* policy. Each policy contains all the items belonging to one root scope. An *InventoryGroup* contains a list of *InventoryItems*, which represent Tetration entities such as servers or appliances by specifying their network address, be it a singular network address, subnet or address range. An *Intent* describes action (allow or deny) to be taken when a network flow matches with the given consumer's *InventoryGroup*, provider's *InventoryGroup* and network protocols and ports. The *CatchAll* represents the catch-all action defined for the root scope inside Tetration. If no application workspace with enforcement enabled exists for the root scope, a default policy of *ALLOW* is written to the produced policy.

When an enforcement is triggered by the users or by a change of inventory groups, Tetration backend sends a full snapshot of defined network policies to Kafka as a sequence of messages represented as *KafkaUpdates*. Refer to *KafkaUpdate*'s comments in *tetration_network_policy.proto* file for details how to reconstruct those messages to a full snapshot as well as how to handle error conditions.

In case *KafkaUpdate* message size is greater than 10MB, Tetration backend will split this message into multiple fragments, each of size 10MB. In case of multiple fragments, only the first fragment will have the *ScopeInfo* field of *TenantNetworkPolicy*. The *ScopeInfo* will be set to nil in the remaining fragments of *KafkaUpdate* message.

6.16.6.5 Reference Implementation of Tetration Network Policies client

Please refer to this tnp-enforcement-client in Java for a reference implementation and instructions how to compile and run a demo client.

This implementation provides common code to read network policies from Tetration policy stream via Kafka only. Vendor specific code to program the actual policies to a network device can be plugged in by implementing the required interface PolicyEnforcementClient.

6.17 Conversations

A conversation is defined as a service provided by one host on a particular port and consumed by another host. Such a conversation is materialized from many flows over different times. ADM algorithms take all such flows, ignore the ephemeral/client ports and de-duplicate them to generate the conversation graph. For any given conversation between host A and host B on server (provider) port N, there has been at least one flow observation from A to B on port N in the timeframe for which the **ADM run** has been performed.

Note that client/server classification affects the ADM conversation view – it dictates which port is dropped (is deemed ephemeral) in the aggregation: See *Client Server Classification*.

6.17.1 Conversations Table View

The Conversations Table view provides a simple way to view aggregated flows from the duration of an ADM run where the consumer port is removed and there is only one record for all time. While policies go from filter to filter, conversations are from ip address to ip address.

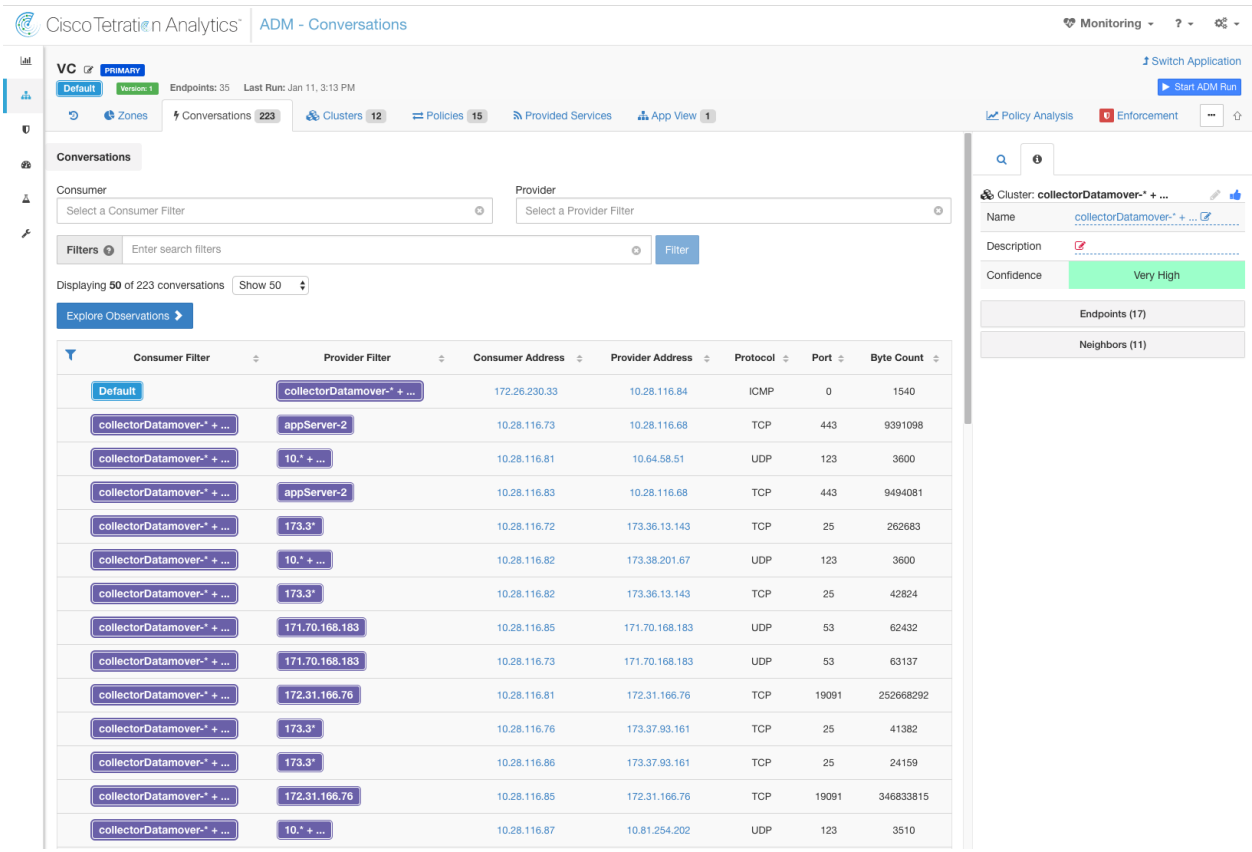


Fig. 6.17.1.1: Conversations Table View

6.17.1.1 Choosing Consumer or Provider

Consumer and Provider can be selected by a typeahead dropdown selector which allows us to choose, Filter, Scope and Clusters as shown in the example below. It displays all conversations between the chosen Consumer and Provider. Note: to delete an existing filter, click on the 'x' icon (erasing the filter may not work).

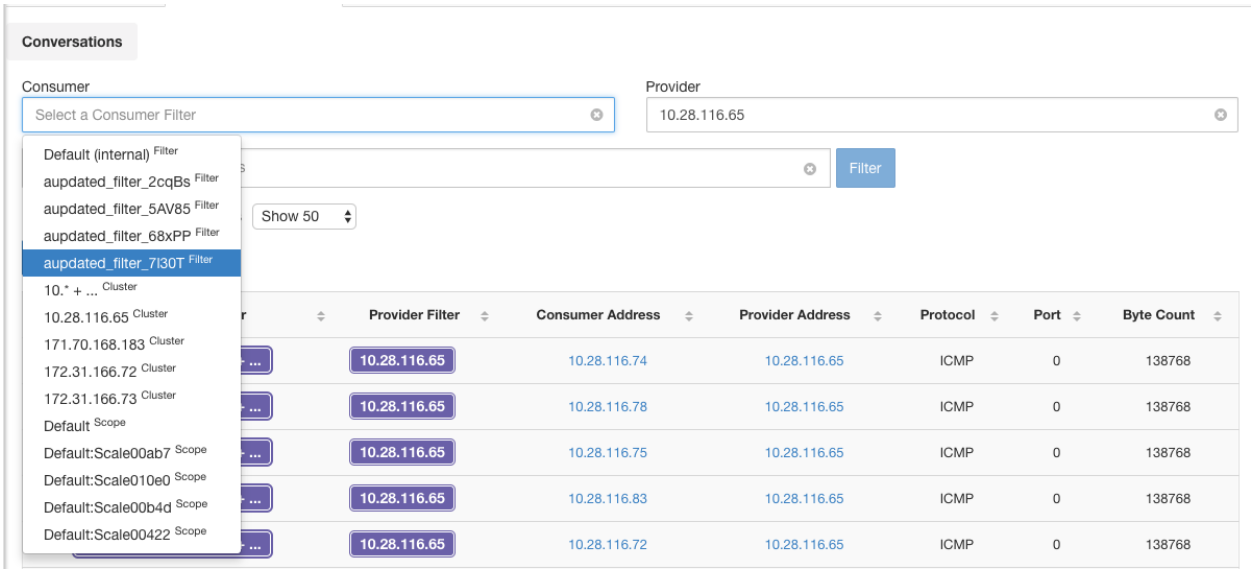


Fig. 6.17.1.1.1: Choosing Consumer or Provider

6.17.1.2 Conversation Filters

Fig. 6.17.1.2.1: Conversation Filters

This is where you define filters to narrow-down the search results. All of the possible dimensions can be found by clicking on the (?) icon next to the word Filters. For any User Labels data, those columns will also be available for the appropriate intervals. This input also supports and, or, not, and parenthesis keywords, use these to express more complex filters. For example, a direction-agnostic filter between IP 1.1.1.1 and 2.2.2.2 can be written:

Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1 And to additionally filter on Protocol = TCP:

(Consumer Address = 1.1.1.1 and Provider Address = 2.2.2.2 or Consumer Address = 2.2.2.2 and Provider Address = 1.1.1.1) and Protocol = TCP

The filter input also supports "," and "-" for Port, Consumer Address and Provider Address, by translating "-" into range queries. The following are examples of a valid filter:

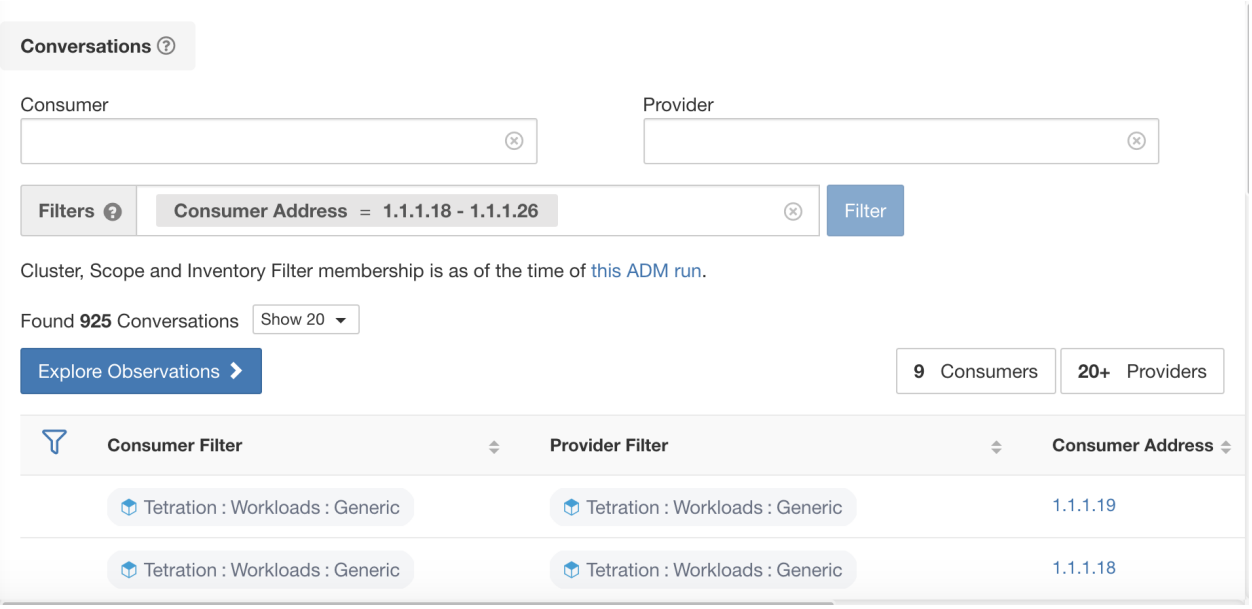


Fig. 6.17.1.2.2: Example: Filter input supports range query for Consumer Address

Available filters:

Filters	Description
Consumer Address	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24). Matches conversation flow observations whose consumer address overlaps with provided IP Address or subnet.
Provider Address	Enter a subnet or IP Address using CIDR notation (eg. 10.11.12.0/24) Matches conversation flow observations whose provider address overlaps with provided ip address or subnet.
Port	Matches conversation flow observations whose port overlaps with provided port.
Protocol	Filter conversation flow observations by Protocol type (TCP, UDP, ICMP).
Address Type	Filter conversation flow observations by Address type (IPv4, IPv6, DHCPv4).
Confidence	Indicated the confidence in the direction of flow. Possible values: High, Very High, Moderate.
Excluded?	Match conversations excluded by an exclusion filter or approved policy.
Excluded By	Match conversations excluded by a specific filter. Possible values: Exclustion Filter, Policy.

6.17.2 Explore Observations

Clicking on the Explore Observations button will enable a chart view that allows quick exploration of the high-dimensional data via a "Parallel Coordinates" chart. A bit overwhelming at first, this chart can be very useful when enabling only the dimensions you're interested in (by unchecking items in the Dimensions dropdown), and when re-arranging the order of the dimensions. A single line in this chart represents a single observation, and where that line intersects with the various axes indicates the value of that observation for that dimension. This can become more clear when hovering over the list of observations below the chart to see the highlighted line representing that observation in the chart:

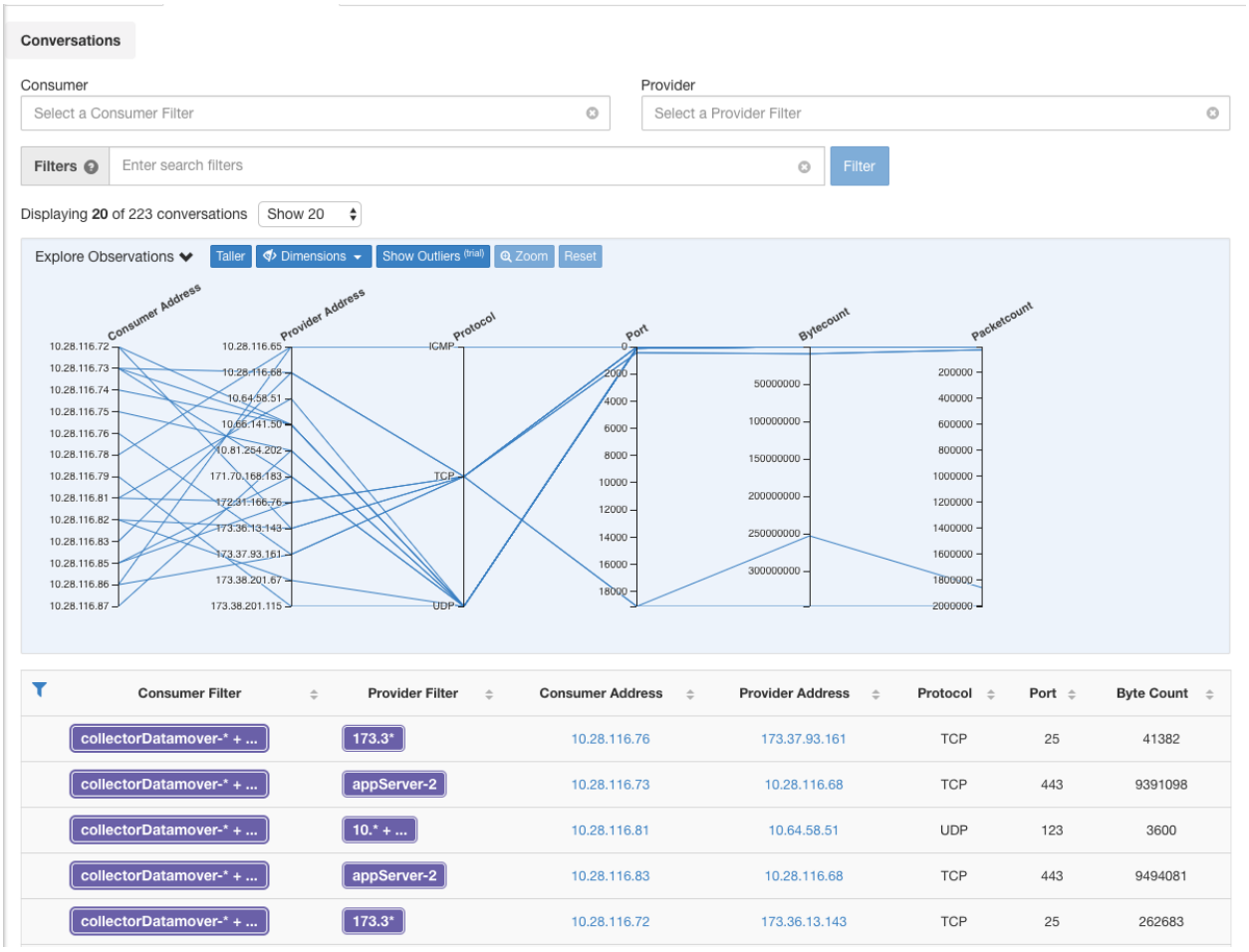


Fig. 6.17.2.1: Explore Observations

6.17.2.1 Conversation Observation hovered

Due to the high-dimensional nature of the conversations data, this chart is quite wide by default, and will require scrolling right to see the entire chart. For this reason it's useful to disable all but the dimensions you are interested in. Hover state in Explore Conversations is provided to map (hover) each conversation with the table list view.

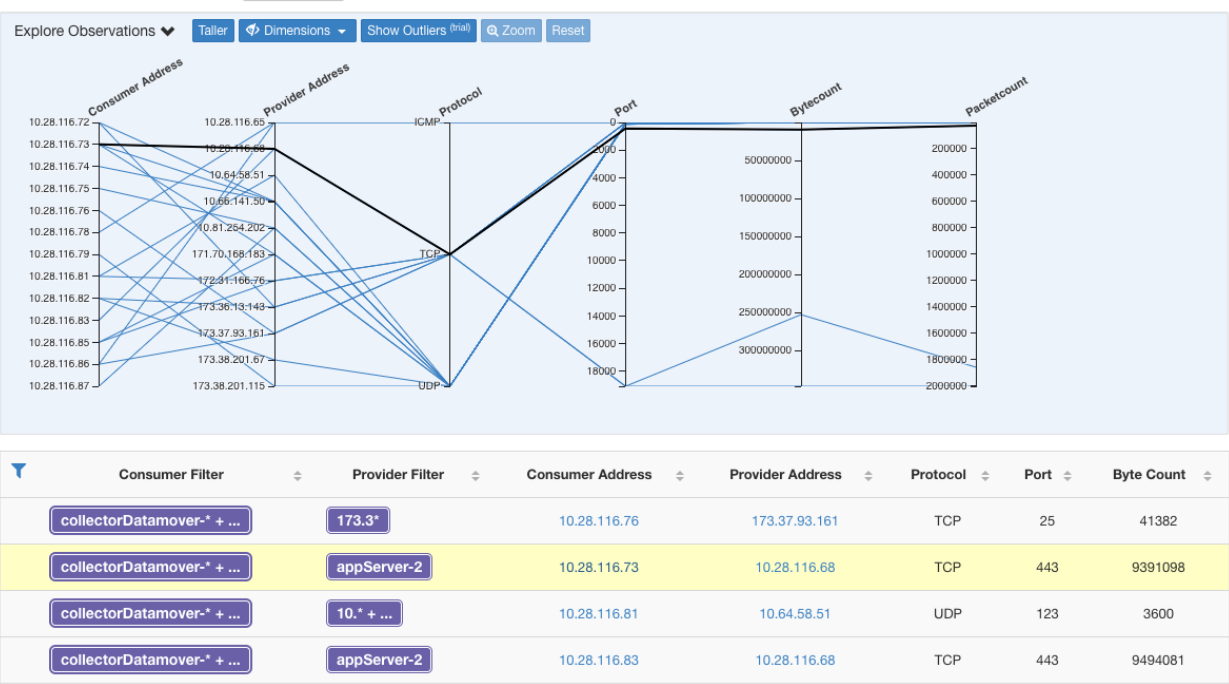


Fig. 6.17.2.1.1: Conversation Observation hovered

6.17.2.2 Filtering

Dragging the cursor along any of the axes will create a selection that will show only observations that match that selection. Click again on the axis to remove the selection at any time. Selections can be made on any number of axes at a time. The list of observations will update to show only the selected conversations

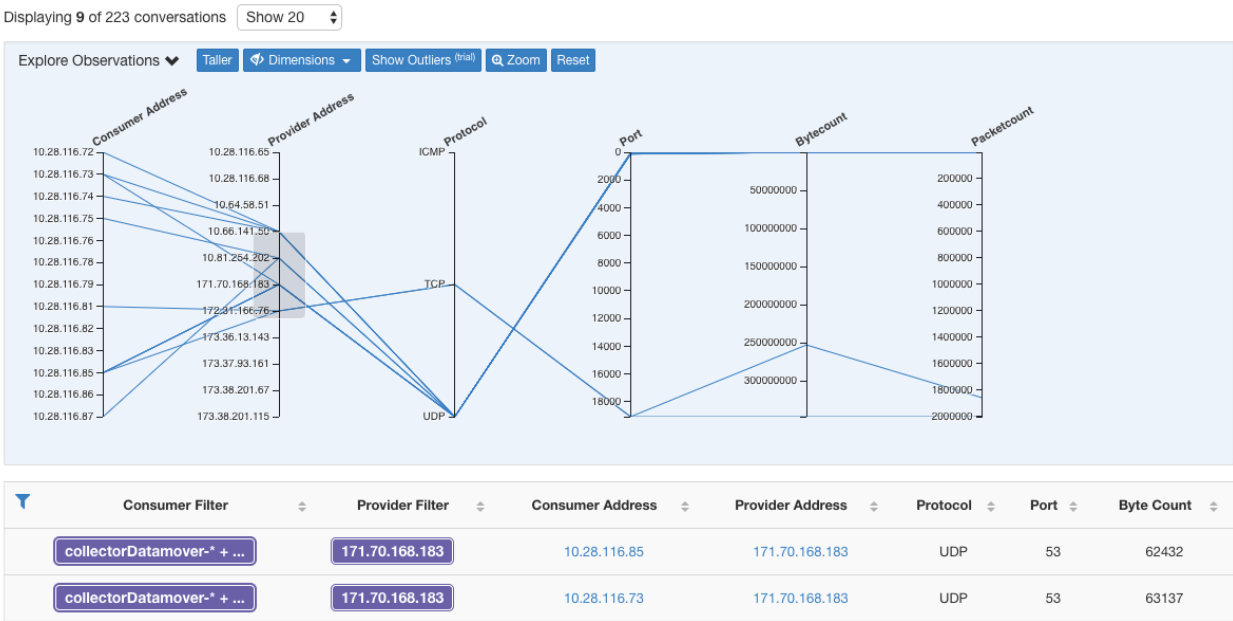


Fig. 6.17.2.2.1: Filtering

6.17.3 Conversations Chart View

Conversation chart view has a very similar look and feel to the policy view page, except that instead of partitions/clusters/policies, it focuses on clusters/workloads/conversations. As illustrated in the figure below, the outer arcs at the high level represent clusters and can be expanded to show the member hosts/workloads as inner arcs. The chords represent the conversations or connections.

The controls and side panel on conversation view behave similarly to the policy view, except for the fact that the side panel information also show detailed information about selected workloads such as consumed/provided services as well as link to parent cluster and process information, if available.

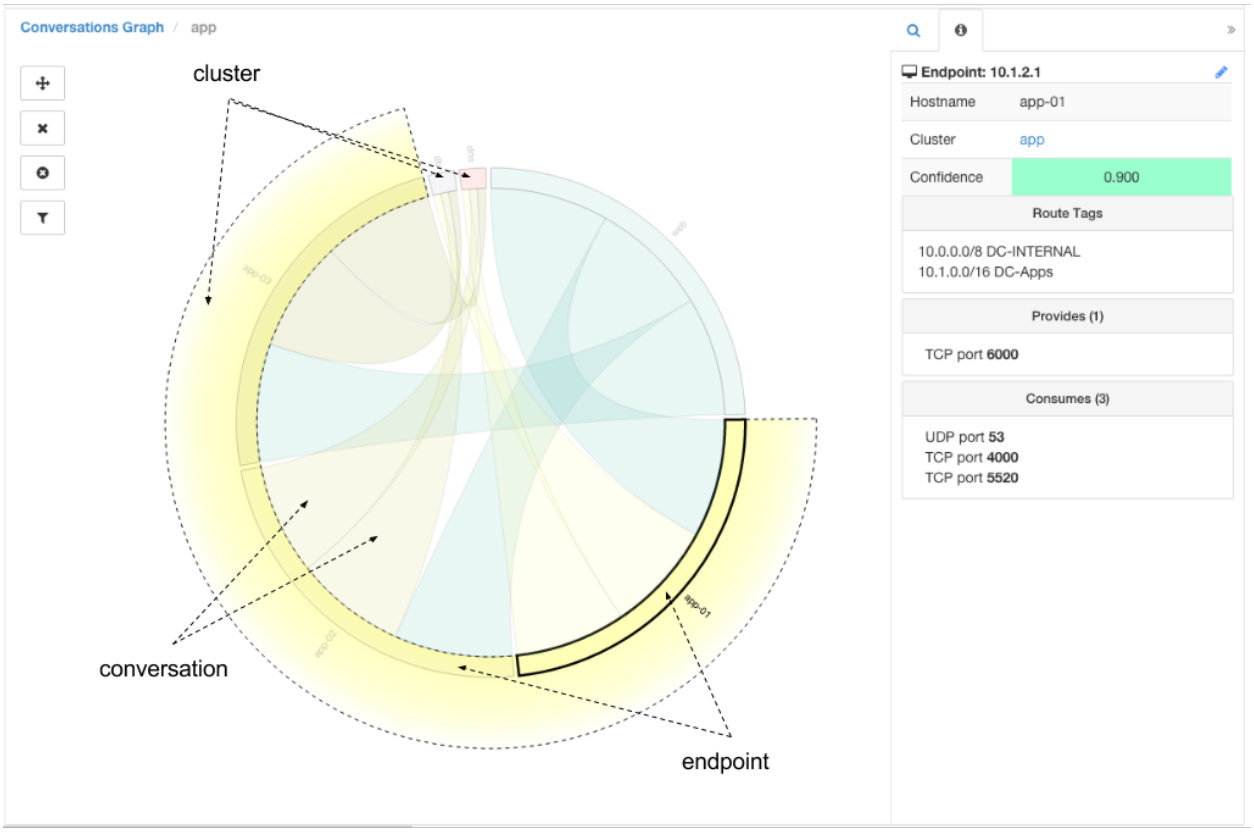


Fig. 6.17.3.1: Conversations Chart View

6.17.4 Top Consumers/Providers of Conversations

The number of top Consumers or Providers based on total conversations reflecting chosen filters can be seen from two buttons on top of the Conversations table. Click on each one to see a dialog containing a table with the Conversation Count column along with each Consumer/Provider's Address, Hostname and other User Annotated columns.

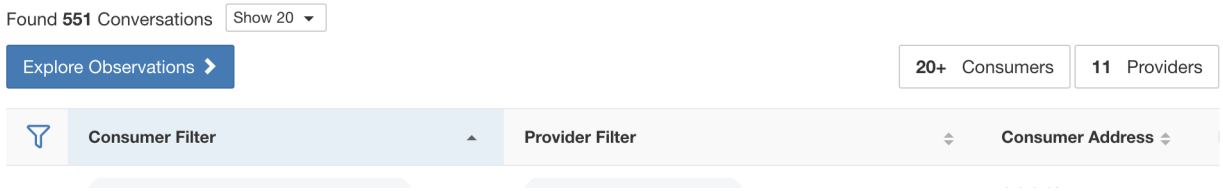


Fig. 6.17.4.1: Above the conversations table

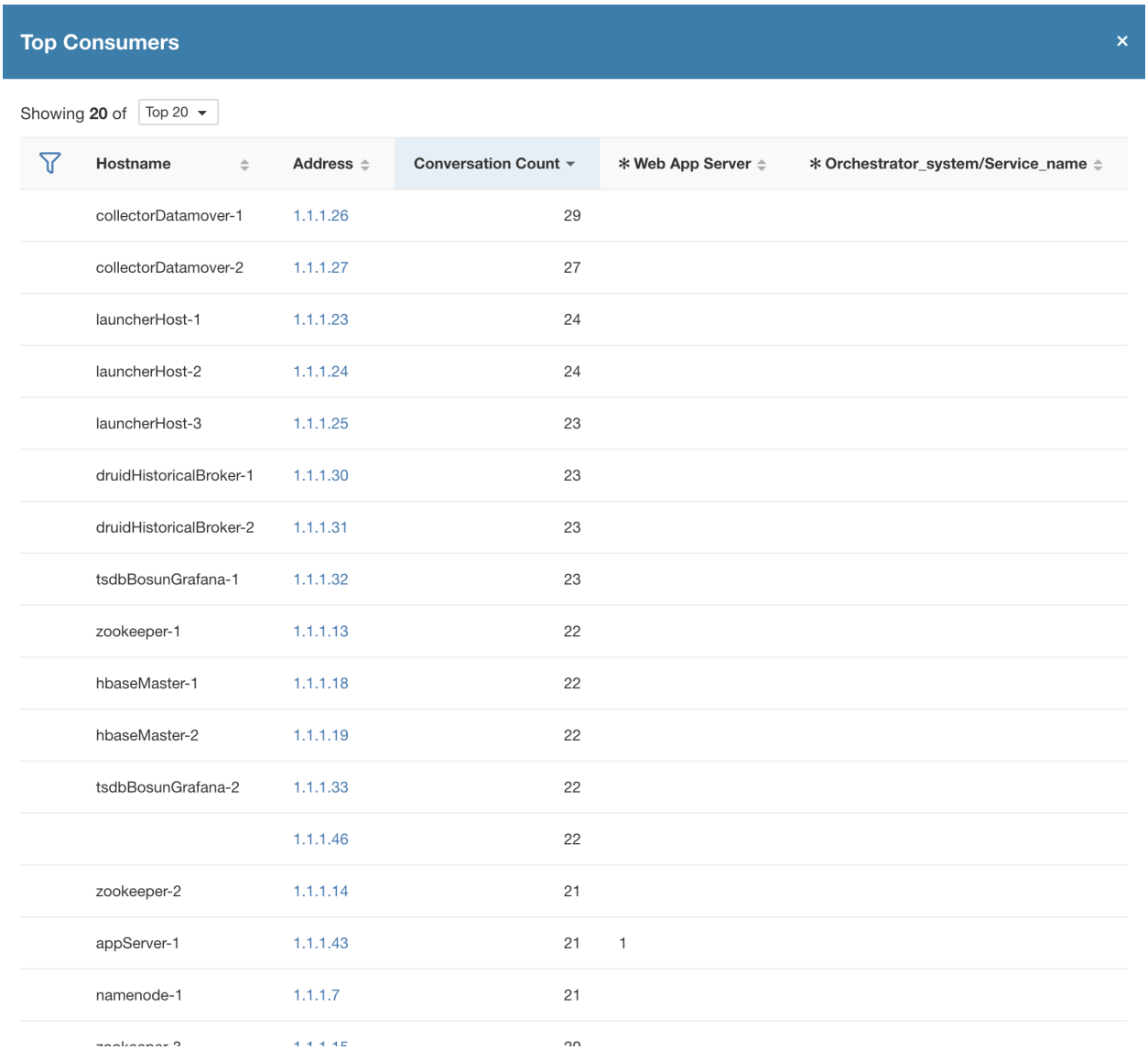


Fig. 6.17.4.2: Top Consumers Modal

Top Providers ×

Showing **11** of Top 20

Hostname	Address	Conversation Count	* Web App Server	* Orchestrator_system/Service_name
appServer-2	1.1.1.44	72		
appServer-1	1.1.1.43	65	1	
mongodb-1	1.1.1.34	56		
redis-1	1.1.1.37	54		
mongodb-2	1.1.1.35	52		
mongodbArbiter-1	1.1.1.36	51		
redis-3	1.1.1.39	51		
redis-2	1.1.1.38	43		
elasticsearch-1	1.1.1.40	37		
elasticsearch-2	1.1.1.41	36		
elasticsearch-3	1.1.1.42	34		

Download table data as JSON Download table data as CSV

Close

Fig. 6.17.4.3: Top Providers Modal

6.18 Misc Functions

6.18.1 App Views

Application Views play a central role in ADM Feature usability and help bridge the gap between the network and application teams. In other words, application views provide a bottom up way of exploring ADM algorithm results with the goal of gaining insight into multitier datacenter applications like a web application. There could be thousands of such applications running in a datacenter. The application view helps users to focus on particular one and share their view with other users.

Similar to ADM workspace workflows, the application list view provides a way to create new application views and view the existing ones by clicking on the tabular view.

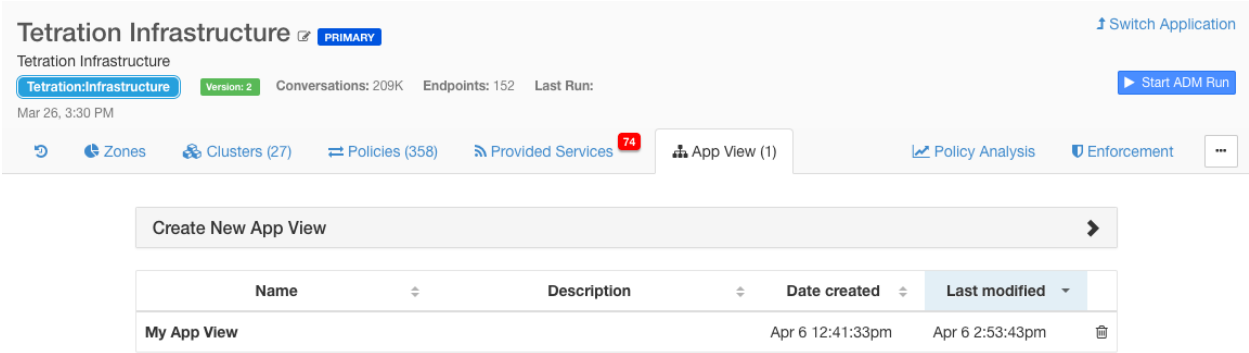


Fig. 6.18.1.1: App View list

6.18.1.1 Building Application View Layout

Upon creation of a new application view, an empty canvas with the list of nodes (clusters, user defined filters and Scopes) is presented. The user can choose to **pin** certain nodes to the canvas and start exploring their neighbors in the sense of network policies. Note that this page shows an extra tab on the right side panel with the list of all nodes.

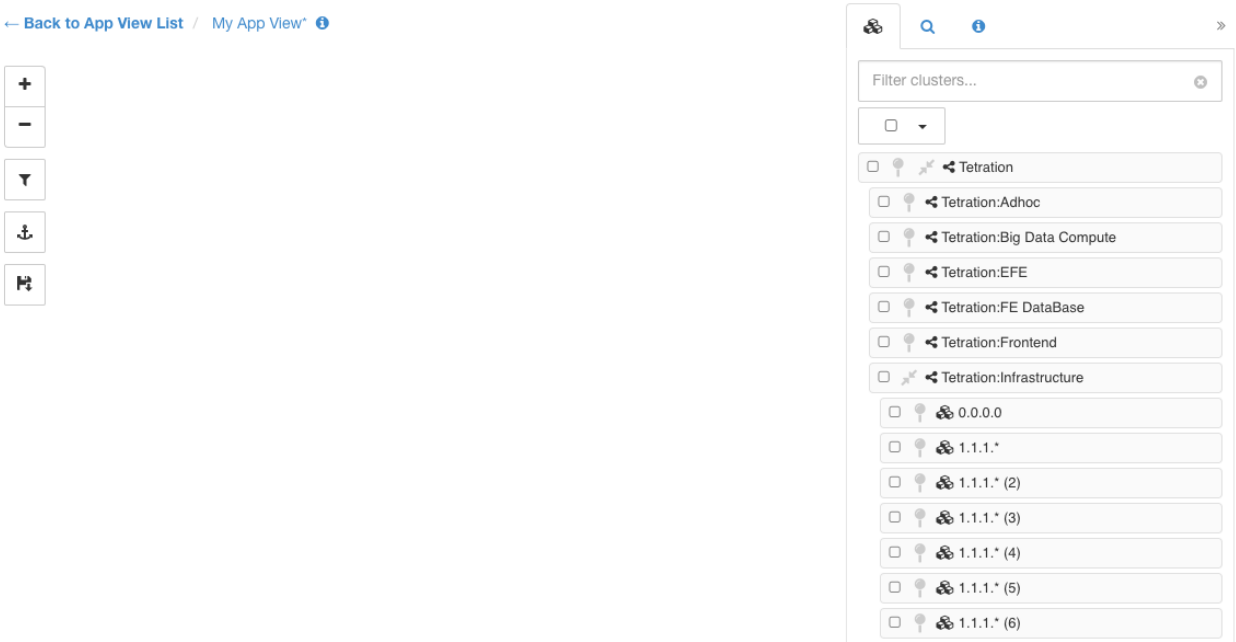


Fig. 6.18.1.1.1: App View empty canvas

The tools on the left provide the ability to:

- Zoom in
- Zoom out
- Filter visible policies
- Anchor selected node positions
- Save App View state, make a copy or export node/policy data

6.18.1.2 Adding Nodes to Application View

Click on the **pin** button next to each item to add that node to the canvas, and double click on any node on the canvas to **show** or **hide** its neighbors.

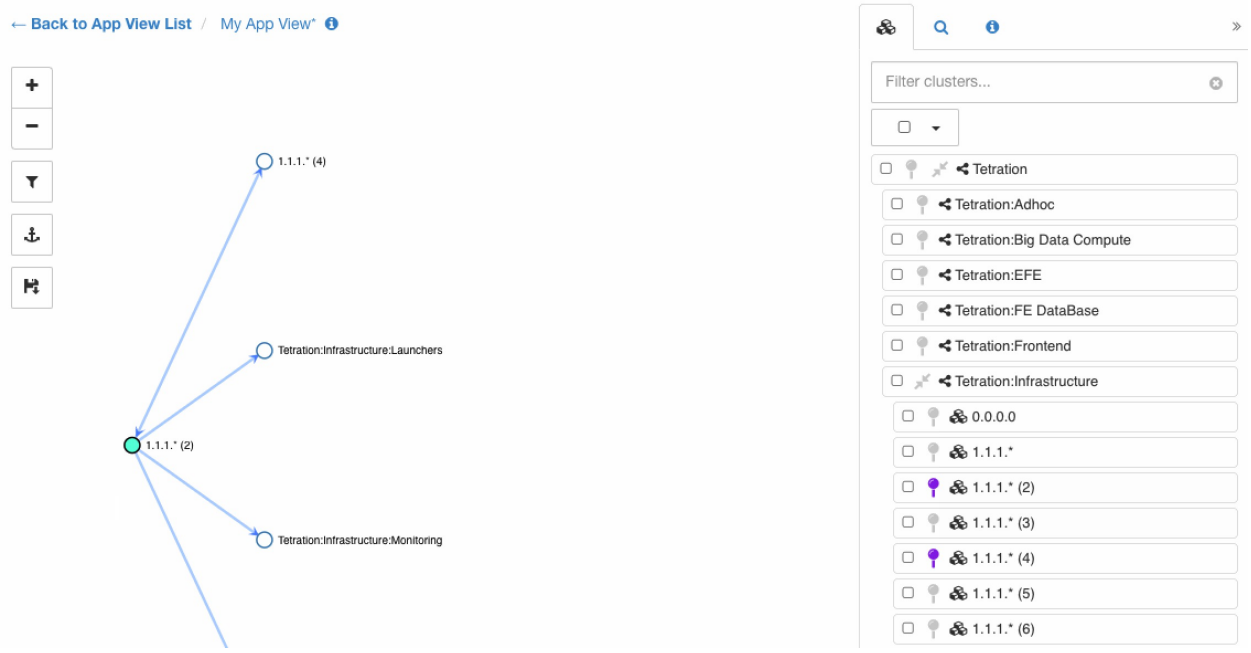


Fig. 6.18.1.2.1: App View pinning and expanding node

6.18.1.3 Adjusting Application View Layout

Note: An edge between two nodes represents the set of network policies between the nodes. If one or more of the conversations between the nodes is going through a load balancer (defined as part of side information uploaded under Advanced ADM run Config), a load balancer icon is shown over the edge. More information can be seen by hovering or clicking over the presented elements.

We can move the nodes to any position to achieve the desired layout. In that case the user's choice will be honored and an anchor icon appears on the node. To reset the position of the **anchored** node, click on the **anchor** button on the toolbar.

The following figure shows a fully expanded graph of our example multi-tier app.

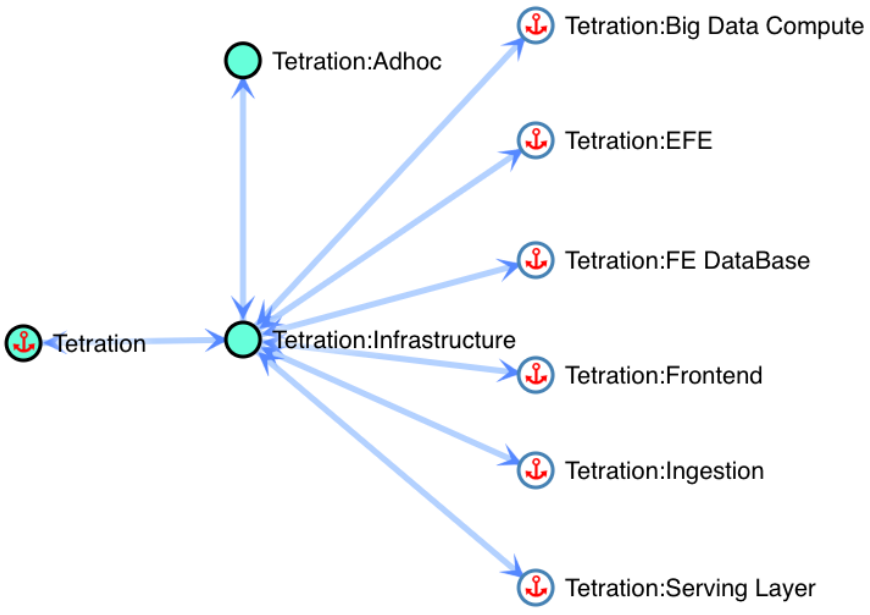


Fig. 6.18.1.3.1: App View layout with anchored nodes

6.18.1.4 Example Multi-tier Application

Click on the save button of the toolbar to save the current layout. This way other users can view the exact same layout of this particular application.

Note: You can move multiple selected nodes all at once by holding the **SHIFT** key and dragging any of the selected nodes.

6.18.1.5 Expanding and Collapsing Scopes

Expand or **collapse** Scopes using the double-arrow icons within App View. When collapsing, all descendant nodes and their policies will be rolled up into the collapsed Scope. This may create edges between the collapsed Scope and other nodes, even if there isn't a policy directly connecting the two nodes because one of the collapsed Scope's children has a policy to that node. The rolled-up edges will be reflected in the App View export, along with aggregated port list.

In new App Views, all Scopes are expanded by default.

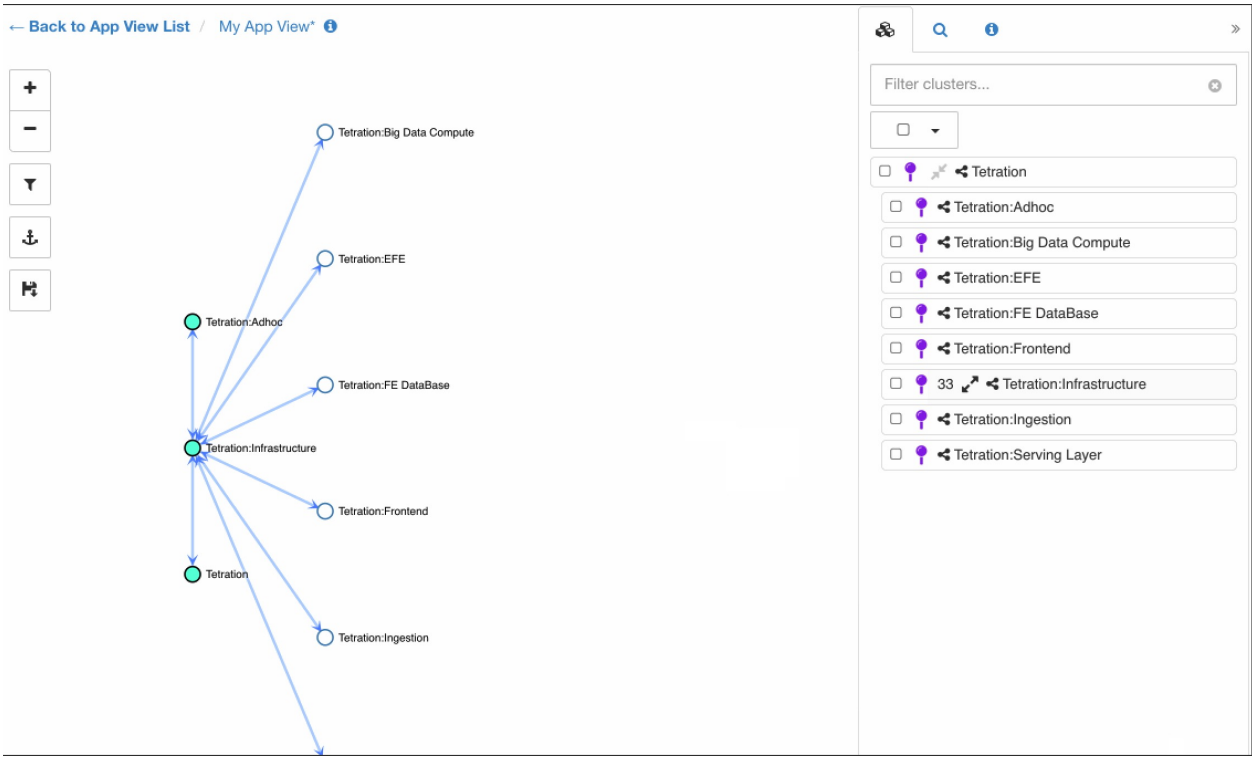


Fig. 6.18.1.5.1: App View collapsing Scopes

6.18.2 Zone View [static mode]

[static mode only] This view allows the user to view all of the clusters in the workspace under the provisioned subnet hierarchy. There could be many datacenters, zones and smaller subnets involved. Moreover, even if one small subnet is selected as part of ADM run, the selected workloads could be communicating with many other subnets in other zones or datacenters.

This page provides a high-level overview of network partitions involved in the form of a **sunburst** chart. The size of each section is proportional to the number of "observed" workloads contained under the corresponding subnet. You can zoom in into each of the subnets by clicking on the section, and zoom out by clicking on the circle at the center.

Furthermore, you can find any given cluster or workload by searching for them by name or IP address and clicking on the search results. The following figure illustrates the process.

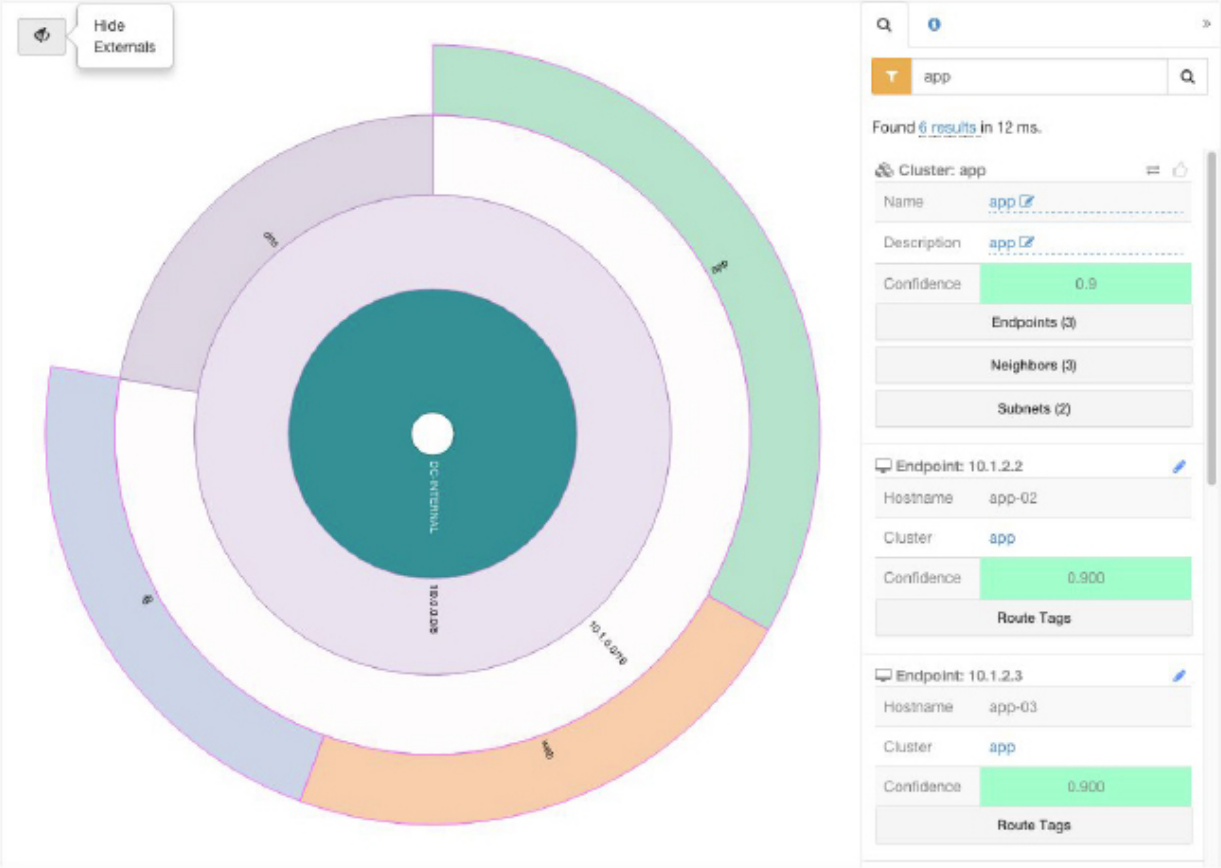


Fig. 6.18.2.1: Subnet Hierarchy View

6.18.3 History & Diff

The history view provides a timeline of the modifications applied to an Application workspace shared and edited by many users. The events highlighted in the history view include adding, removing and renaming workloads and clusters, moving workloads between clusters, creating and updating application views, uploading side information, submitting and aborting ADM runs and many more events alike. Also, the history view shows which user has made what modifications to the workspace.

You can navigate to the history view by clicking on the corresponding button on the ADM header (see picture below).

The history view is divided into three sections: "Application Activity Log", "Versions" and "Policy Versions". The first section contains events that apply to the whole application such as ADM runs and enforcement events. The latter two provide a list of the versions with summary information. The User can navigate to a more detailed view of the history of the version from there.

Every ADM run creates a new ADM Version (v*) of the workspace so that the run can be reverted by the user if the results are unexpected. The first ADM run generates version 1, and all modifications after that run, such as editing or approving clusters (but not a rerun), are also grouped under version 1. A subsequent ADM run generates a new version (unless the run failed).

Analyzing policies or enforcing the latest policies will create a new Policy Version (p*). These versions can not be edited, only deleted entirely.

You can compare the changes in clustering among versions, and policies that have been analyzed or enforced. See the *Diff View*. You can switch to any version by clicking on "Switch to Version" when viewing the list of version. In the example below, the workspace is switched to version 1.

Note: Running ADM algorithms after switching back to an older version of the workspace removes all of the later versions to maintain a linear history view. In the same example, it means that submitting an ADM run after switching to version 2 will delete version 3 if successful.

Clicking on any of the events in the history view provides more context information about the event on the side pane.

For example, clicking on an ADM run event reveals many useful information about the status, duration and configurations of that ADM run. Moreover, the side panel shows high level statistics about the changes to the existing clusters and workloads due to the run. More details about that is described in ADM *Diff View*.

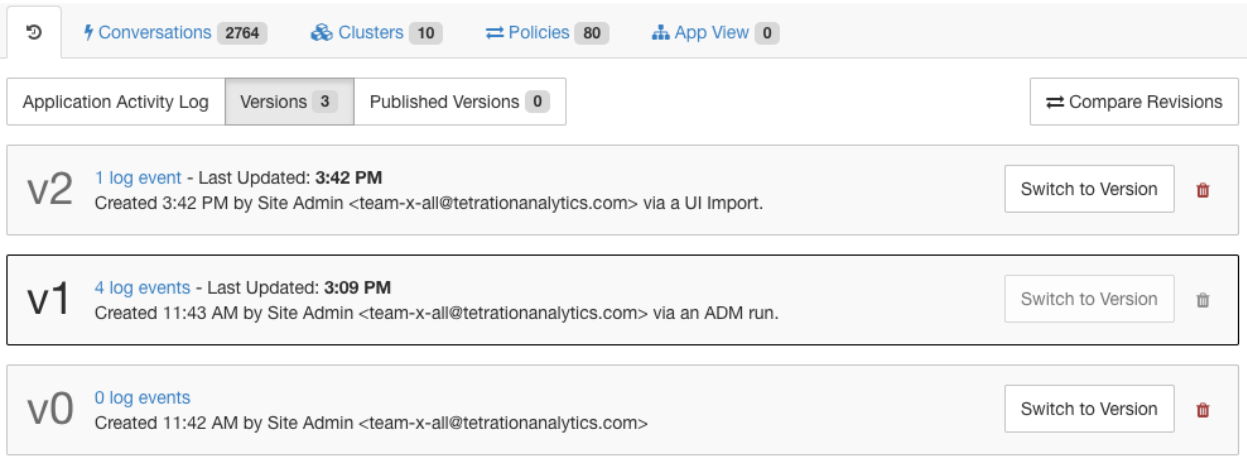


Fig. 6.18.3.1: List of ADM versions with summary information

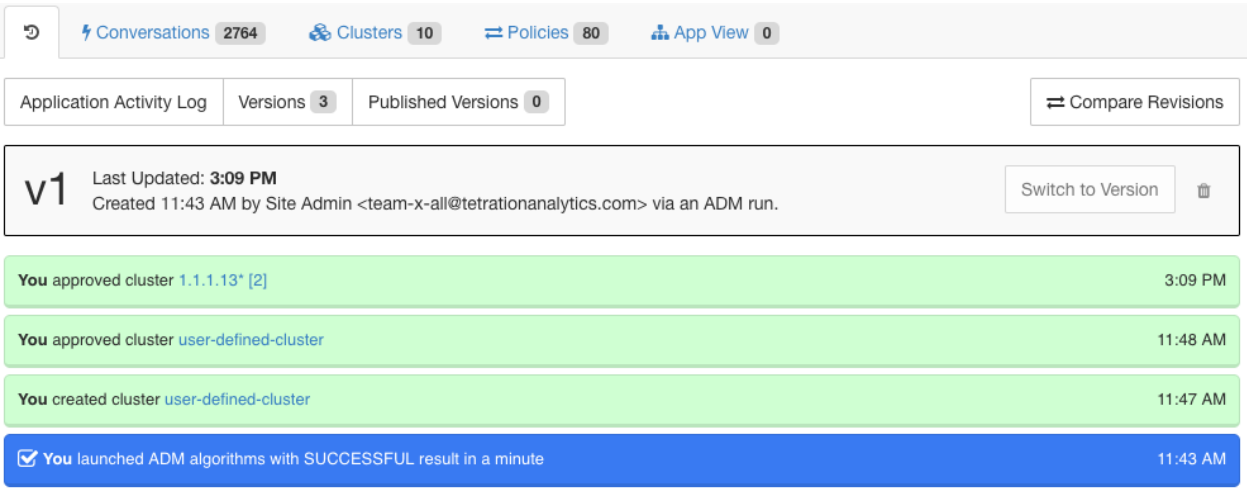


Fig. 6.18.3.2: Log of events applicable to version v1 of this application

You can click on these events to view detail information from past ADM Runs, including Exclusion Filters, External Dependencies and Advanced Configurations used:

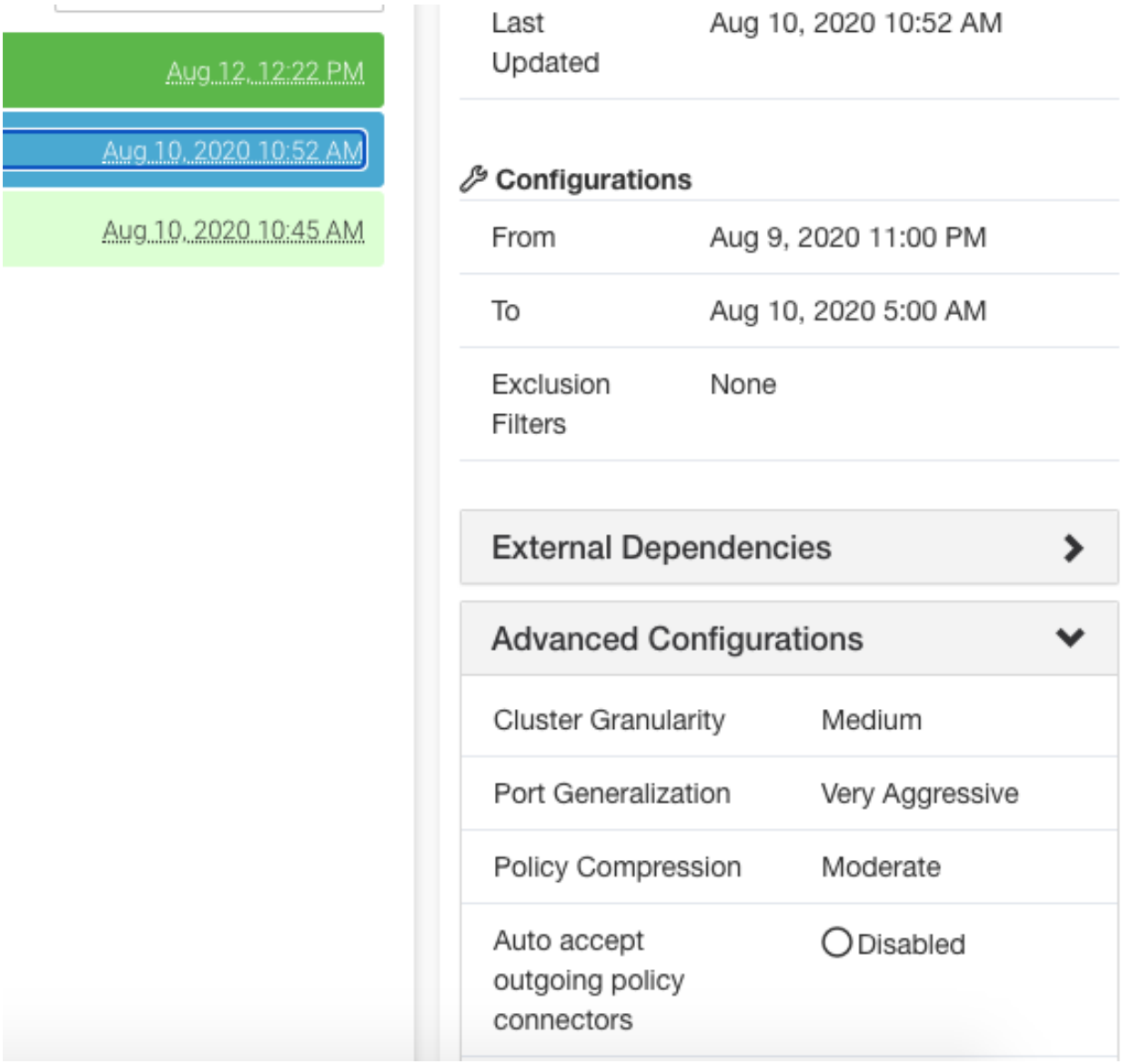


Fig. 6.18.3.3: Configurations used for paticular ADM Runs

6.18.3.1 Deleting Application Versions

Application versions generated from ADM runs (v* versions) can be deleted, unless it is the last remaining version. Policy versions (p* versions) can be deleted as long as the version is not being actively analyzed or enforced.

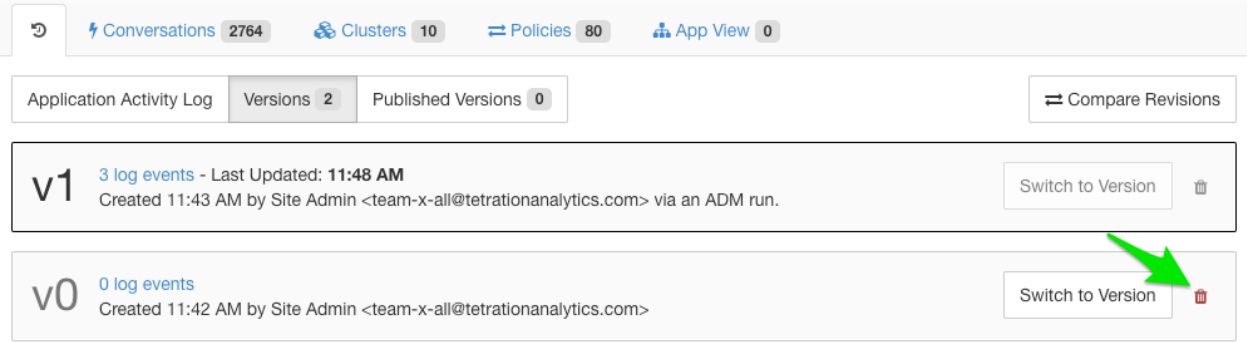


Fig. 6.18.3.1.1: Deleting Application Versions

6.18.3.2 Diff Views

ADM clusters diff view is designed to allow users to compare any two versions of the ADM workspace in terms of the effect of ADM reruns on existing clusters and workloads' memberships. In this release a policy diff view is also supported, see *Policy Diff*.

There are three ways to navigate to the clusters diff view:

1. Upon a successful ADM run, a message will appear indicating the success with a link that navigates to the diff view showing the effects of the run.

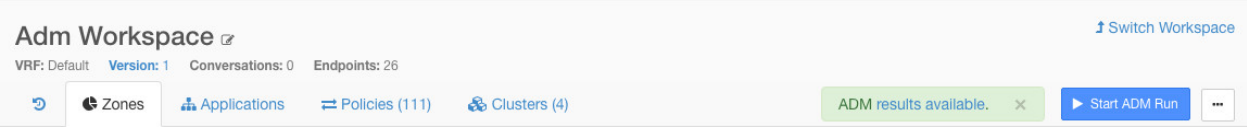


Fig. 6.18.3.2.1: Successful ADM Run

2. From history view by clicking on **Compare Revisions** button on the top-right corner of the page.
3. From the side panel, when it is showing context information for an ADM run by clicking on the button on the top right corner of the side panel. See figure below.

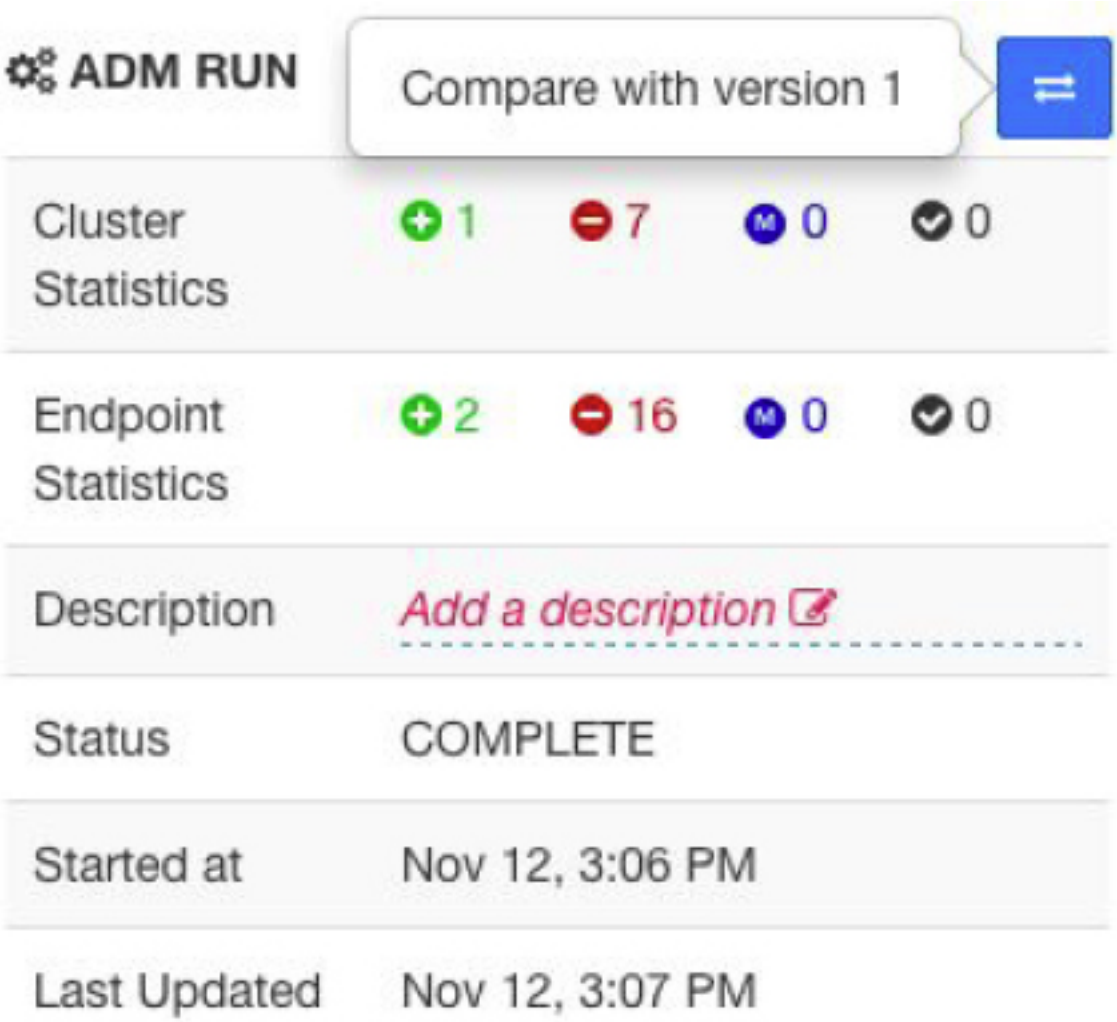


Fig. 6.18.3.2.2: Showing Context Information

At the top level, ADM diff view shows high level statistics about changes in clusters and workloads showing the number of added , deleted, modified and unchanged clusters and workloads.

The rest of the view is organized as a list of clusters in the order of added, deleted, modified and unchanged, each color coded to reflect the status as well as the number of workloads added to or removed from the cluster.

You may search for a particular cluster or workload by name or IP address. Clicking on any of the rows representing a cluster, expands that row to show how the contents of that cluster is changed.

NOTE: By default all the unchanged clusters are hidden, but they can be viewed by clicking on the button with the eye icon. Switching the ADM diff view to compare two other revisions is as simple as clicking on the revision numbers and selecting a different one from the dropdown menu.

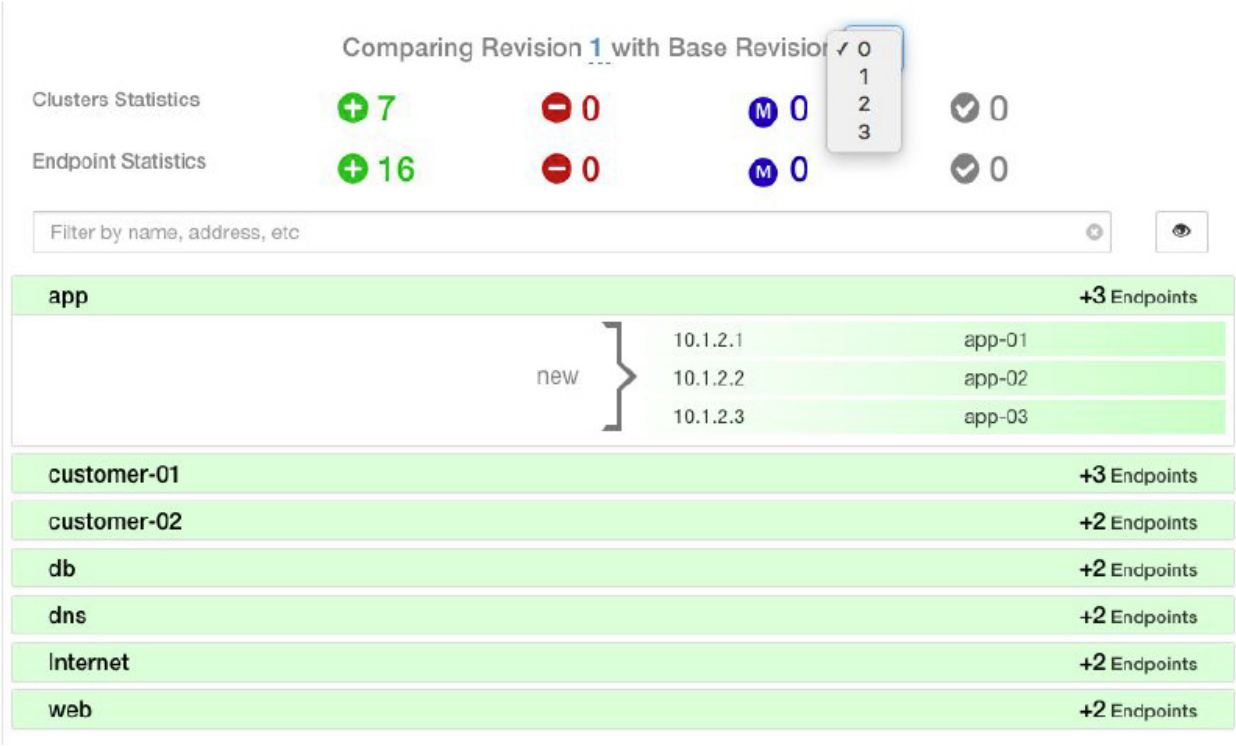


Fig. 6.18.3.2.3: ADM Clusters Diff View

6.18.3.3 Policy Diff

The policy diff view can be selected similar to cluster diff view. After choosing base and compare version, policy changes will be displayed in three categories: Absolute, Default and Catch All. Few feature of the diff table:

- Different services that belongs to the same policy are grouped together
- Filter policy changes by facet or by diff type
- Policy changes and services are paginated
- Download filtered policy changes as CSV

Table 6.18.3.3.1: Facet filter properties

Property	Description
Priority	e.g. 100
Action	e.g. ALLOW, DENY
Consumer	e.g. Consumer Cluster
Provider	e.g. Provider Cluster
Port	e.g. 80
Protocol	e.g. TCP

Table 6.18.3.3.2: CSV output columns

Column	Description
Rank	The category of the policy. e.g. ABSOLUTE, DEFAULT, CATCH_ALL
Diff	The diff type of the change. e.g. ADDED, REMOVED, UNCHANGED

Continued on next page

Table 6.18.3.3.2 – continued from previous page

Column	Description
Priority	e.g. 100
Action	e.g. ALLOW, DENY
Consumer Name	The name of the consumer cluster.
Consumer ID	The ID of the consumer cluster.
Provider Name	The name of the provider cluster.
Provider ID	The ID of the provider cluster.
Protocol	e.g. TCP
Port	e.g. 80

In the figure below, policy versions p1 and v1 are compared.

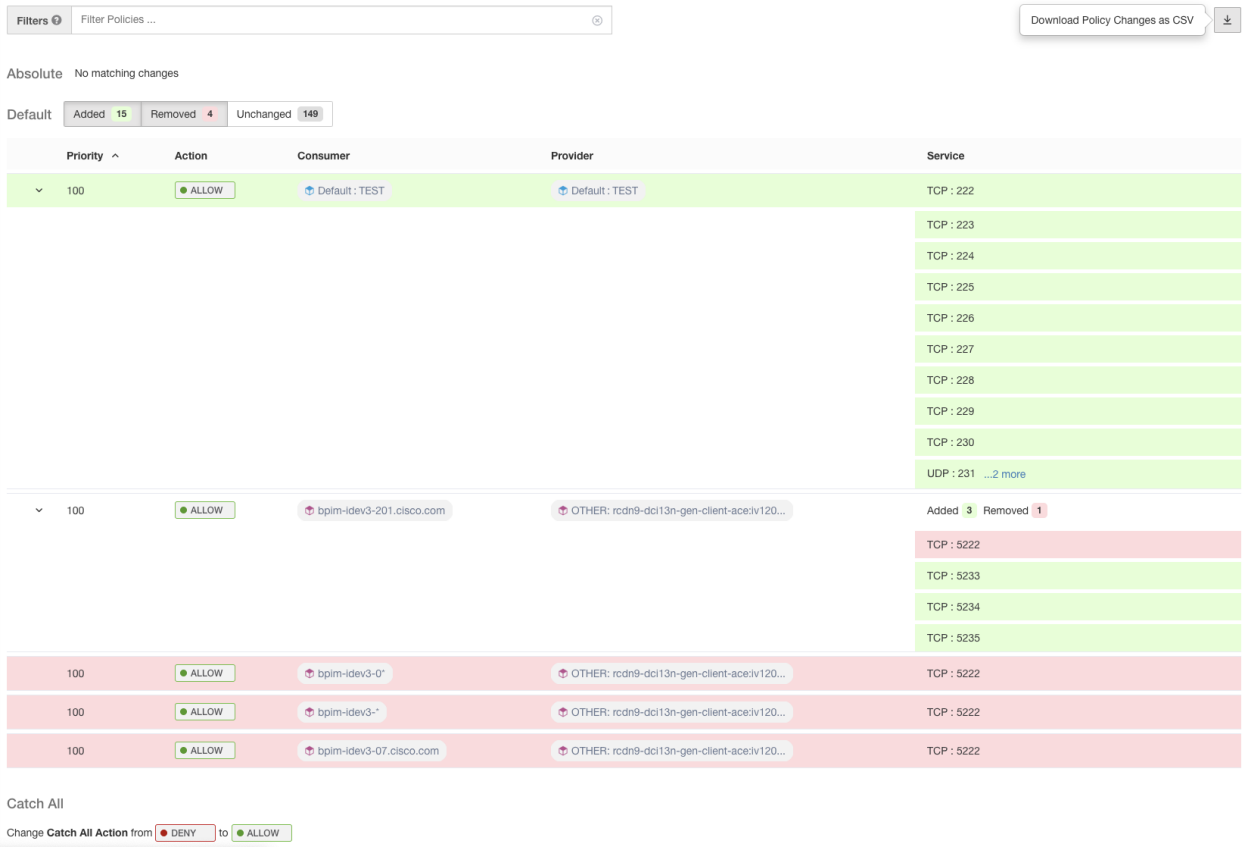


Fig. 6.18.3.3.1: Policy Diff View

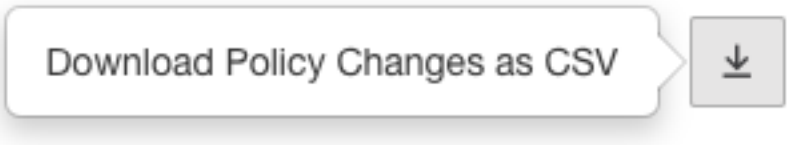


Fig. 6.18.3.3.2: Policy Diff View Download Button

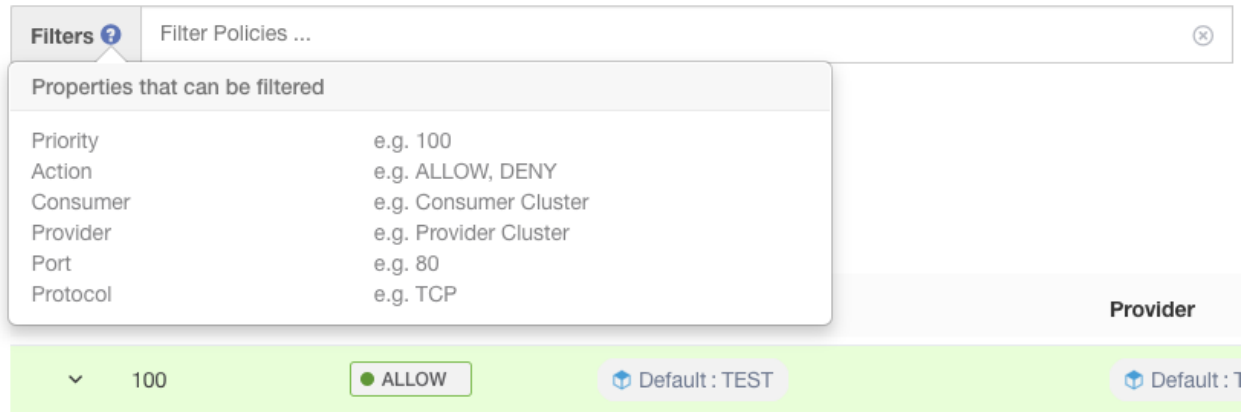


Fig. 6.18.3.3.3: Filtering Policy Diff View

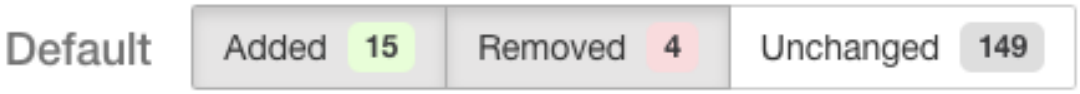


Fig. 6.18.3.3.4: Policy Diff View Diff Type Filter

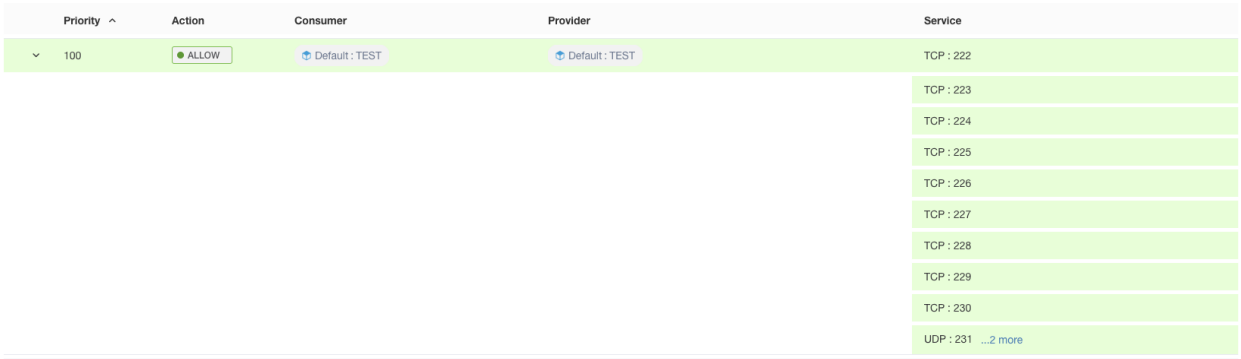


Fig. 6.18.3.3.5: Policy Diff View Grouping

Rank	Diff	Priority	Action	Consumer Name	Consumer ID	Provider Name	Provider ID	Protocol	Port
DEFAULT	REMOVED	100	ALLOW	bpimweb-idev3-0*	5efa608a14b8b728979bf885	OTHER: rtp1-dcm02n-oama-idev4:iv653	5efa608a14b8b728979bf863	TCP	6021
DEFAULT	REMOVED	100	ALLOW	bpimweb-idev3-0*	5efa608a14b8b728979bf885	OTHER: rtp1-dcm02n-oama-idev4:iv653	5efa608a14b8b728979bf863	TCP	6022
DEFAULT	REMOVED	100	ALLOW	bpim-idev3-201.cisco.com	5efa608a14b8b728979bf870	OTHER: rcdn9-dci13n-gen-client-ace:iv120	5efa608a14b8b728979bf866	TCP	5222
DEFAULT	REMOVED	100	ALLOW	bpim-idev3-203.cisco.com	5efa608a14b8b728979bf871	OTHER: rcdn9-dci13n-gen-client-ace:iv120	5efa608a14b8b728979bf866	TCP	5222

Fig. 6.18.3.3.6: Policy Diff View CSV Output

6.18.4 Import/Export

6.18.4.1 Export Application Workspace

All the relevant contents of clusters and policies in each application workspace can be downloaded as a single file in a number of popular structured document formats like JSON, XML and YAML. One can use such files for further in-house processing or ingestion by other policy enforcement or analysis tools.

Navigate to the **...** menu item on the application header and click on the **export** item. This will show the export dialog. You can choose whether the exported file should include only the cluster contents or cluster contents as well as the security policies among the clusters generated by ADM algorithms based on real network flows. Choose the desired format and click download to download the file into the local file system.

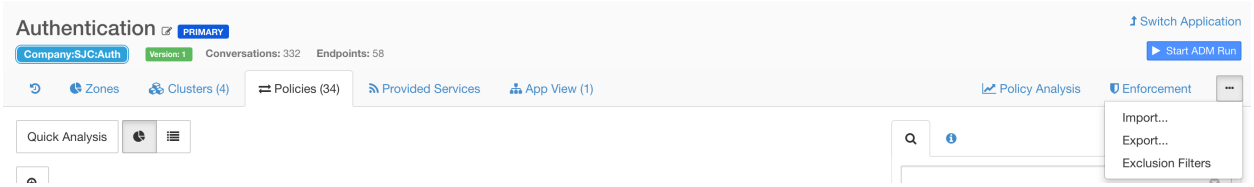


Fig. 6.18.4.1.1: Import/Export menu items

Fig. 6.18.4.1.2: Exporting Policies of an Application workspace

6.18.4.2 Export App View

In the case that application workspace is very large with thousands of workloads and hundreds of clusters, it might be desirable to export only the contents of a particular application view constructed by the users. Additionally, it may be desirable to export application policies at a coarser granularity than generated by ADM. You can use many features of the app view to construct a more limited and/or coarser view of policies by collapsing certain scopes. The exported file will have policy definitions that is close to the graph shown on the app view canvas.

To achieve this, navigate to the application view and click on the button on the left toolbar. This will reveal a dropdown menu including an export option. First, make sure the app view is saved by clicking on the **Save** menu item. Clicking the **Export** item shows a similar dialog as the one above.

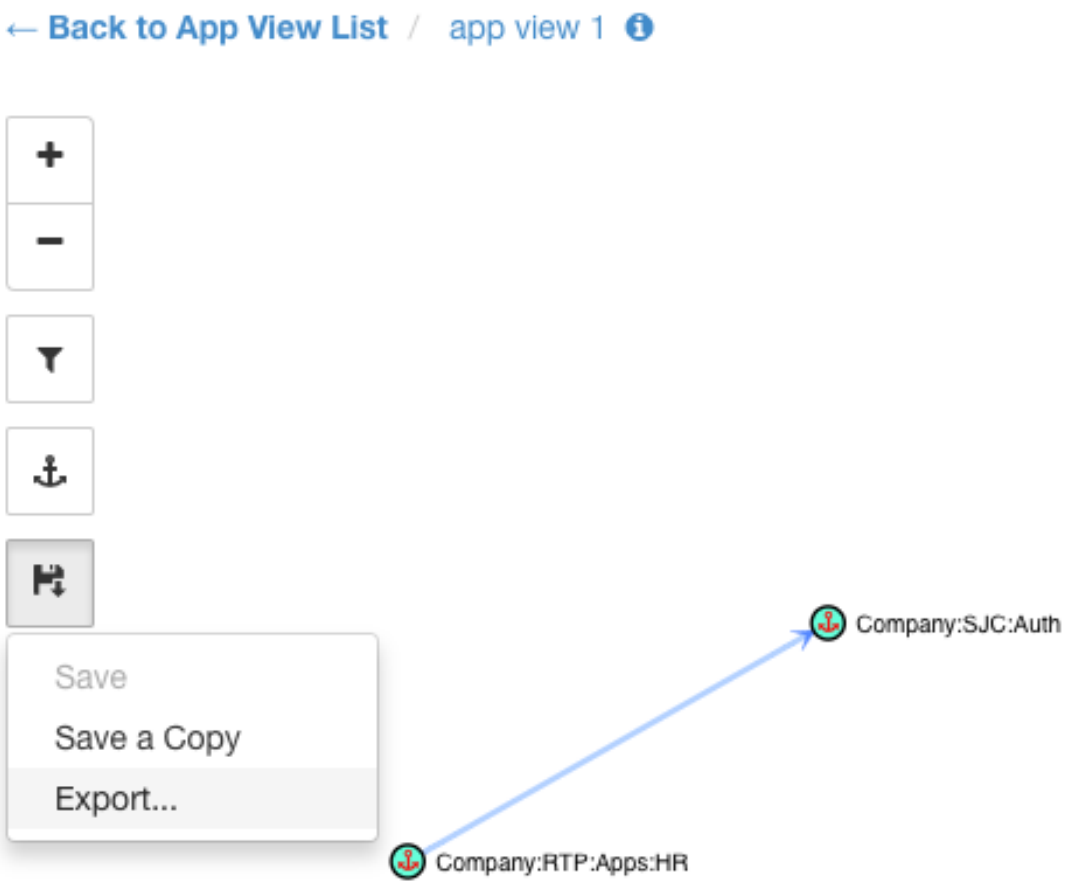


Fig. 6.18.4.2.1: Exporting a specific App View

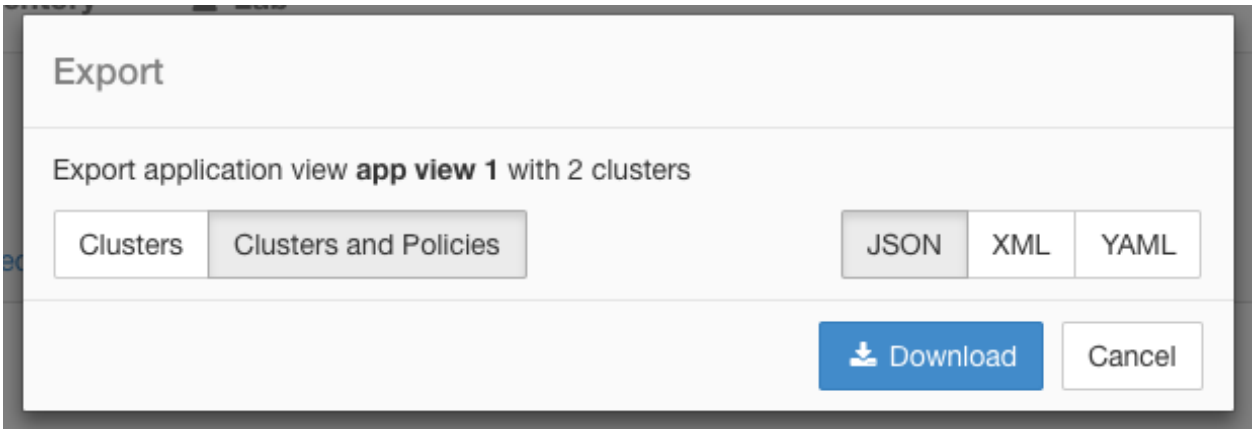


Fig. 6.18.4.2.2: Exporting Policies of an App View

Note: The app view does not show DENY policies and self-loops, i.e., policies with the same consumer and provider.

However, the exported file will include all the information related to DENY policies, self-loops and Catch-all action as well.

6.18.4.3 Import

You can import known cluster and policy definitions into an application by directly uploading a JSON file. Similar to ADM runs, uploading policies into an existing workspace creates a new version and places the cluster and policy definitions under the new version.

Click on the **Import** menu item from the **...** menu in application header. In the import dialog, you can select a JSON file with a valid format. A small sample JSON file demonstrating the schema for policies and clusters can be found by clicking on the **Sample** button.

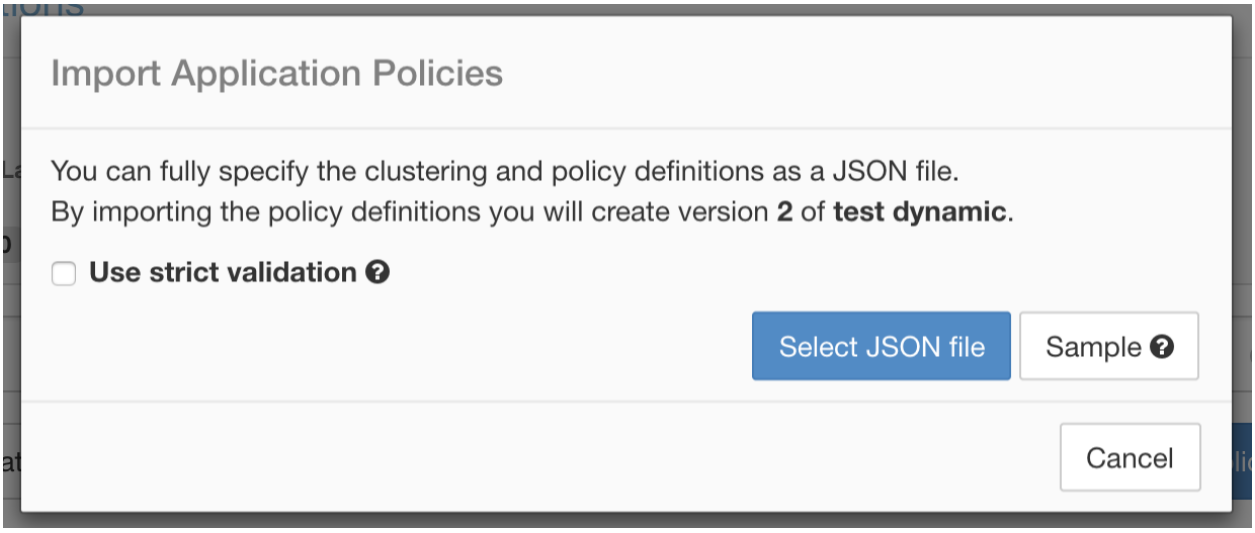


Fig. 6.18.4.3.1: Importing Clusters/Policies

Strict Validation if enabled, will return an error if the JSON contains unrecognized attributes. This is useful for locating typos or incorrectly identified optional fields.

Note: All imported policies are marked as approved by default unless explicitly marked as `approved: false`. You have the option to maintain such approved policies when running ADM algorithms to generate new set of policies. See *ADM Run Configuration* for more info.

Pro Tip: The schema of the JSON file retrieved by exporting an application workspace or app view is schema-compatible with the expected format for importing policies into an application. Therefore, you can clone policies from one application workspace to another using an export followed by an import. Note that many features may not work the same when exporting and then importing policies. For example, the conversations backing the policies are not included in the export and will not be present when importing the policies either.

6.18.4.4 Garbage Collection

A cleanup job, on a weekly basis, performs deletion of all workspace versions, except the most recent, which are not accessed for six months. This job also removes old policy experiments not accessed in the last 30 days.

6.19 Automated LB Config Support in ADM (F5 only)

ADM generates policies from configuration for load balancers connected to an external orchestrator. Generating policies from configuration minimizes ADM's reliance on flow data and improves the accuracy of clusters and policies generated by it.

It relies on clients to report flows to the load balancer for generating policies to permit this traffic.

Experimental Feature

This feature and its APIs are in **ALPHA** and are subject to changes and enhancements in future releases.

6.19.1 Terminology

VIP Virtual IP: IP to which the client sends traffic destined for a service.

SNIP SNAT IP: IP used by the load balancer for sending traffic to backend hosts.

BE Backend Endpoint: IP of the backend host.

HIP Health-check IP: Source IP used by the load balancer for sending health-check traffic to backend hosts.

Note: HIPs are the same as SNIPs in automap mode. However, HIPs and SNIPs can differ when a SNAT pool is configured.

6.19.2 Deployment

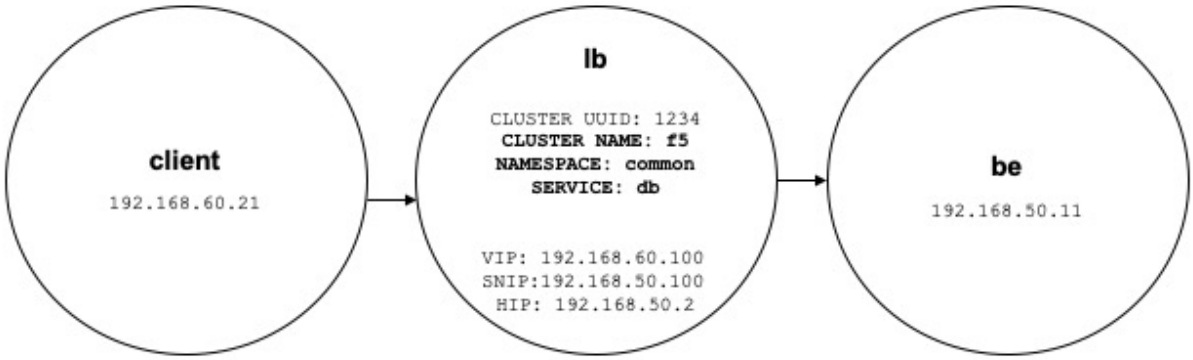


Fig. 6.19.2.1: Deployment

Consider the following deployment where load balancer VIPs, SNIPs, and HIPs are part of the *lb* scope, and BEs are part of the *be* scope. Scopes are created as follows

- client

  The client scope includes clients communicating with the load balancer. For the example above, the *client* scope query is as follows:

```
address eq 192.168.60.21 or address eq 192.168.60.22
```

- lb

  The F5 external orchestrator labels VIPs, SNIPs, HIPs, and BEs used by the load balancer. These labels can be used to construct scope queries, where *orchestrator_system/service_name* is used for selecting VIPs, *orchestrator_system/service_startpoint* SNIPs, and *orchestrator_system/service_healthcheck_startpoint* HIPs for the service. For the example above, a scope query that includes VIPs, SNIPs, and HIPs for service *db* is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
(user_orchestrator_system/service_name eq db or
 user_orchestrator_system/service_startpoint eq db or
 user_orchestrator_system/service_healthcheck_startpoint eq db)
```

Note: It is required that SNIPs and VIPs be part of the same scope.

- be

  user_orchestrator_system/service_endpoint selects BEs for a service. For the example above, a scope query that includes BEs for service *db* is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_endpoint eq db
```

6.19.3 Clusters

Each service generates up to four ADM clusters, of which only the service cluster is visible to the user. SNIP, HIP and BE clusters appear as related clusters for the service cluster. HIP and BE clusters are generated only when HIPs and BEs are present in the *lb* scope.

For the example above, ADM generates a SNIP cluster and HIP cluster in the *lb* scope that include SNIPs and HIPs for the service. Since BEs lie outside the *lb* scope, ADM does not generate a backend cluster but instead adds the *be* scope to the list of related clusters for *db*.

Clusters are generated as follows:

- Service

  The service cluster includes VIPs for a service. The query for the service cluster as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/namespace eq common and
user_orchestrator_system/service_name eq db
```

- SNIP

  SNIPs for a service are included in the SNIP cluster. The query for the SNIP cluster is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_startpoint eq db
```

- HIP

  HIPs for a service are included in the HIP cluster. The query for the HIP cluster is as follows:

```
user_orchestrator_system/cluster_id eq 1234 and
user_orchestrator_system/service_healthcheck_startpoint eq db
```

- Backend

  A backend cluster for the service is generated when one or more BEs are part of the *lb* scope. This doesn't apply to the example above, resulting in a backend cluster not being generated in the *lb* scope.

6.19.4 Policies

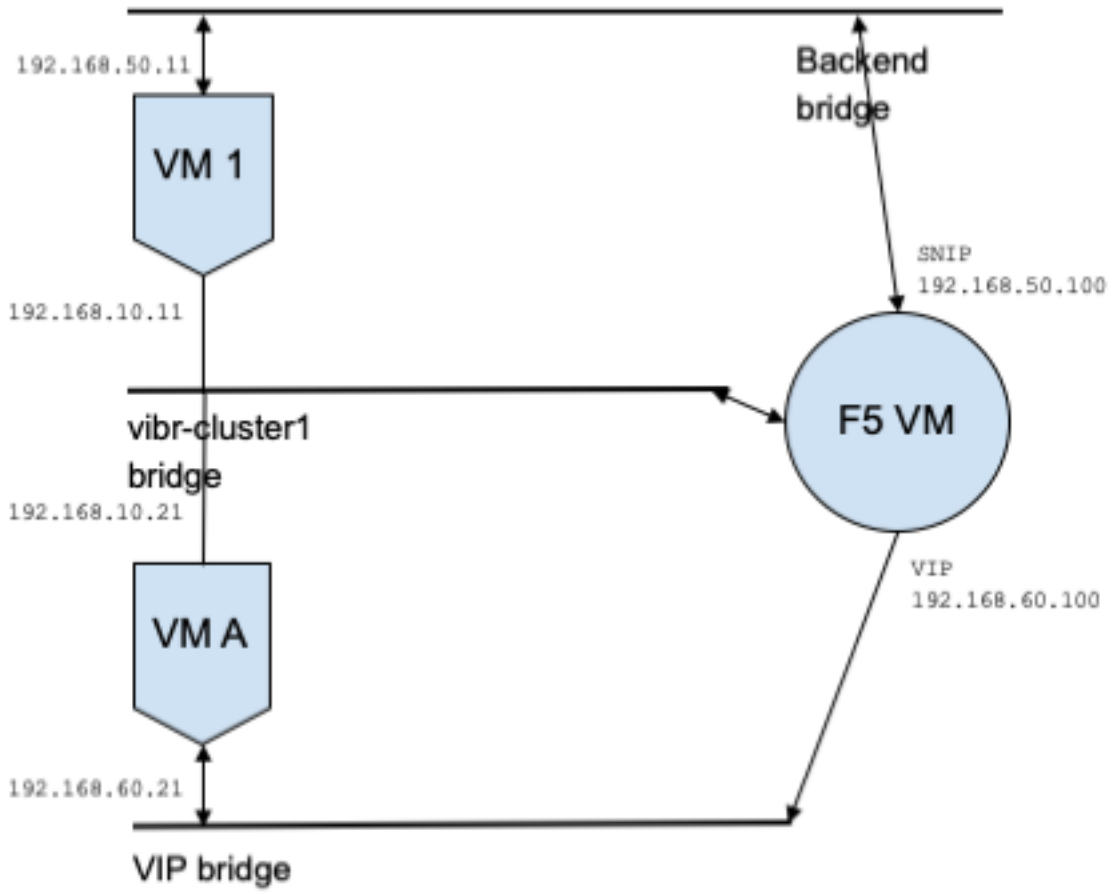


Fig. 6.19.4.1: Policy Generation

Assume we have a service *db* with VIP *192.168.60.100*, SNIP *192.168.50.100*, and a backend VM with IP *192.168.50.11* listening on port 10000. Traffic from client VM *192.168.60.21* to *db* results in the following policies:

- Policy from client to VIP

  The following policy permits from the client VM to service *db*:

```
{
  "src": "<uuid of client scope>",
  "dst": "<uuid of service cluster>",
  "l4_params": [
    {
      "port": [
        10000,
        10000
      ],
      "proto": 6,
    }
  ]
}
```

- Policy from SNIP to BE.

A policy permitting traffic from the SNIP to the BE is autogenerated from configuration, and shows up as a related policy for *db*.

```
{
  "src": "<uuid of SNIP cluster>",
  "dst": "<uuid of be scope>",
  "l4_params": [
    {
      "port": [
        10000,
        10000
      ],
      "proto": 6,
    }
  ]
}
```

A policy connector from the *lb* scope to the *be* scope pushes the following policy to it.

Consumer	Provider	Port	Protocol	Action
SNIP	be	10000	TCP	Allow

This generates firewall rules on BE host *192.168.50.11* allowing incoming traffic from LB SNIP *192.168.50.100* on port 10000.

- Policy from HIP to BE.

A policy permitting traffic from the HIP to the BE is autogenerated from configuration, and shows up as a related policy for *db*.

```
{
  "src": "<uuid of HIP cluster>",
  "dst": "<uuid of be scope>",
  "l4_params": [
    {
      "port": [
        0,
        0
      ],
      "proto": ICMP,
    }
```

(continues on next page)

(continued from previous page)

```
  ]
}
```

A policy connector from the *lb* scope to the *be* scope pushes the following policy to it.

Consumer	Provider	Port	Protocol	Action
HIP	be	0	ICMP	Allow

This generates firewall rules on BE host *192.168.50.11* allowing incoming ICMP traffic from LB HIP *192.168.50.2*.

6.19.5 Caveats

- When multiple services from the same load balancer instance have the same name, backend rules generated for any of these services will include backend pools for all of them, i.e. rules will be more permissive than needed.

CHAPTER

SEVEN

FORENSICS

The **Forensics** feature set enables monitoring and alerting for possible security incidents by capturing real-time forensic events and applying user-defined rules. Specifically, it enables:

- Defining of rules to specify forensic events of interest
- Defining trigger actions for matching forensic events
- Searching for specific forensic events
- Visualizing event-generating processes and their full lineages

Warning: When the **forensics** feature is enabled, the sensor may consume additional host resources depending on the sensor configuration. Please refer to section *Software Agent Config*.

7.1 Compatibility

The forensics signals are reported by the deep visibility agents on all platforms except AIX. Please refer to the Forensics signals section below for more information.

Forensics information is provided through Linux kernel APIs, Audit and syslog, Windows kernel APIs, Windows events, etc. In general, OS vendors guarantee compatibility within a major release. However, it is possible that APIs could differ slightly across platforms and minor releases, as OS vendors may backport features and fixes. As a result, some forensics event types might not be available on some platforms. Also, the agent does not attempt to recover or enable any OS services that are disabled when the agent starts.

For example, there are number of forensics signals that use Linux Audit Framework. If forensics is enabled, a deep visibility agent will insert Tetration audit rules into the system after the agent starts. The rule insertion requires the system to have augenrules utility installed and /etc/audit/rules.d directory. If any of these prerequisites are not satisifed, Tetration audit rules will not be inserted. As a result, Forensics signals including File Access and Raw Socket Creation will not be reported.

If an user has enabled forensics previously and disables it, the sensor will remove the audit rules inserted by Tetration. On Redhat 7.3 and CentOS 7.3, we observed an operating system bug that may impact the rule removal process. Here is how the sensor removes the audit rules: 1. Sensor removes the taau.rules in */etc/audit/rules.d/* 2. Sensor runs *$service auditd restart*. The OS will regenerate the rule set based on the *audit.rules* and **.rules* files in */etc/audit/rules.d/*. Then auditd will load the rules into the system

The operating system adds *-D* at the beginning of /etc/audit/rules.d/audit.rules file to clear all the rules before inserting the new rule set. However, on Redhat 7.3 and CentOS 7.3 machines the */etc/audit/rules.d/audit.rules* may not have *-D*. This is because the OS will create an empty */etc/audit/rules.d/audit.rules* file if this file does not exist and a

default rule file in the sub-directory of */usr/share/doc/audit-<version>/* does not exist either, e.g., */usr/share/doc/audit-2.8.4/rules/10-base-config.rules* is one possible default rule location. The exact OS behavior can be observed from the RPM update script by running *$rpm -qf -scripts /etc/audit/rules.d*

In Linux, some forensics signals rely on the observation of 64-bit system calls. 32-bit Linux system calls are not supported in the current release.

7.2 Forensics signals

The Forensics feature must be enabled for software sensors to capture and report forensic events. The feature can be enabled in Software Agent Config. Please refer to section *Software Agent Config* for more information.

When the Forensics feature is enabled, sensors will report the following forensic events.

Signal	Description
Privilege Escalation	Privilege escalations such as commands executed with sudo
User Logon	User logon events
User Logon Failed	User logon failed attempts
Shellcode	Suspicious shell executions resembling shellcode attempts
File Access	Accesses on very sensitive files such as password files
User Account	Adding or removing user accounts
Unseen Command	New commands that the sensor has not seen. Users can use command anomaly score to tune results based on scope. See *Unseen Command* for details.
Unseen Library	New library that sensors have not seen a process loaded before
Raw Socket Creation	Processes creating raw sockets (e.g., port knocking)
Binary Changed	Changes to hash values or modification times of known binaries
Library Changed	Changes to hash values or modification times of known libraries
Side Channel	Side channel attack attempts (Meltdown)
Follow User Logon	Descendant processes forked/executed after User logon events
Follow Process	Follow Process events report processes that match user forensic config rules based on process attributes such as binary path, command string, etc.
Network Anomaly	Anomalies in network traffic of the workload, see *PCR-based Network Anomaly detection* for more information

7.2.1 Privilege Escalation

When a process changes its privilege from low to high, it's considered a Privilege Escalation. In Linux, this means the user-id of a process has changed from non-zero to zero. There are legitimate cases such as changing the password for a normal user and other special-purpose binaries such as sudo. This event is currently not available in Windows. Privilege escalation in Windows is typically done through other mechanisms rather than changing the privilege of the process itself, i.e., integrity level. Privilege escalations on Windows are covered by other types of forensics events, such as unseen commands or binary changes below.

7.2.2 User Logon

User logon events including SSH, RDP, and other types of logons. Whenever available, sensors captures who, when, and how a user logs in. For example, for SSH in Linux, sensors report username, authentication type (password, public), and source IP.

7.2.3 User Logon Failed

Similar to User Logon events above, sensors report failed attempts to log in with similar information whenever available.

7.2.4 Shellcode

Shellcode events have different interpretations in Linux and Windows. In Linux, sensors identify processes running as an interactive shell without a login session or terminal. (There are no good reasons for an interactive shell running outside of a login session.) In this release, detection of shellcode events is limited in that it assumes the attack will utilize a shell already available in the system. If an attack uploads new binaries, sensors will flag these binaries as either unseen commands or binary changes, if they replace existing binaries. In Windows, every process linked with the PowerShell DLL will be labeled as shellcode. Users can create rules to filter out legitimate cases.

7.2.5 File Access

File Access events report accesses to very sensitive files, such as password files. In this release, the list of files to be monitored cannot be changed by users. In Linux, the sensor monitors write access to /etc/passwd. Sensor also monitors read and write accesses to /etc/shadow. Windows will not trigger this event in this release.

7.2.6 User Account

User Account events report the creation of local user accounts whenever the information is available.

7.2.7 Unseen Command

Unseen Command events report commands that the sensor has not seen before. An unseen command is defined as an unseen transition/edge from a parent to a child process. For example, assuming a web server (httpd) is executing a CGI script called abc.sh, when the sensor sees it for the first time, it will report abc.sh as an unseen command. Subsequent executions of abc.sh by the web server will not result in forensic events since the sensor has seen and reported it before. If a service or process never executes any binary, an unseen command event from that service/process indicates a possible compromise. Note that sensors are stateless across restarts, so a previously seen command will be reported again after a sensor restart.

Since 3.4, for SaaS clusters, each Unseen Command event is associated with a command anomaly score ranging from 0.0 to 1.0. The lower the score, the more anomalous the transition is. The command transitions, i.e. the tuples `(parent command line, command line)`, are cross-checked for anomalous transitions among those events having the same tuple below:

- The narrowest scopes that the sensor belongs to. E.g. the unseen command event is observed on workload `W` which belongs to the following scope lineages: `Root Scope -> A -> B -> C` and `Root Scope -> D -> E`. Then, the command is cross-checked among all workloads in scopes `C` and `E` (Note that `C` and `E` can be either overlapping or non-overlapping). The anomaly score of the event is the maximum of the anomaly scores of the event with respect to those 2 scopes.

- The execution path of the running process.
- The execution path of the parent process.
- The binary hash of the running process.

A score 1.0 means the same command transition having the same tuple `(narrowest scope, execution path, parent execution path, binary hash)` has been seen . A score 0.0 means such command transition with such execution path, parent execution path and binary hash of the running process has never been observed on any hosts within the same scopes. The anomaly score can be used to suppress similar unseen command alerts from firing within the same scope and reduce false positives. See *Tetration - Anomalous Unseen Command rule* for an example of how this score can be used.

Note that the anomaly score is only available for SaaS clusters in 3.4.

7.2.8 Unseen Library

Unseen Library events report libraries that the sensor has not seen a process loaded before. An unseen library is defined as an unseen pair of binary execution path and library path. For example, an application usually loads a relatively stable list of libraries. An attacker who has access to the machine may restart the application and LD_PRELOAD malicious libraries. When the sensor sees the newly loaded malicious libraries in this application binary execution path for the first time, it will report unseen library events. Subsequent load of the malicious libraries will not result in forensic events since the sensor has seen and reported it before. Legitimate cases include application loads new libraries after an upgrade or applications dynamically load new libraries. Note that sensors might report a previously seen library again after restart.

Note that this is an experimental feature and is subject to change in future releases.

7.2.9 Raw Socket Creation

Raw Socket Creation events are only supported on Linux in this release. Raw sockets are typically used to snoop or inject/spoof traffic. There are legitimate uses of raw sockets, such as in diagnosis tools like tcpdump, or when crafting special IP packets like ping or arp. Malicious uses include stealth scans to avoid logging by target/victim machines, malware port knocking, etc. Tetration sensors also create raw sockets for collecting flow-related information. (For consistency, sensors do not suppress events triggered by their own flow information collection.)

7.2.10 Binary Changed

Binary Changed events report changes to the file contents and attributes of binaries for running processes. Sensors record the file attributes of every running process. If a process runs a binary at the same path, but with different file attributes (ctime, mtime, size, or hash), the sensor will flag the process as a binary change. Legitimate cases include application upgrade.

7.2.11 Library Changed

Library Changed events report changes to the file contents and attributes of libraries for running processes. Sensors record the file attributes of loaded libraries. If a process loads a library at the same path, but with different file attributes (ctime, mtime, size, or hash), the sensor will flag the process with a library change. Legitimate cases include library upgrade.

Note that this is an experimental feature and is subject to change in future releases.

7.2.12 Side Channel

Side Channel events report running software that exploits side channel vulnerabilities. This release provides one side channel detection capabilities on selected Linux platforms: Meltdown. See the details below for supported machine configurations. These are advanced security features and therefore disabled by default. Users should expect to see increased CPU usage when side channel reporting is enabled. The CPU quota configured in the UI will still be honored. If the forensic collection sub-process of the sensor determines that its CPU usage is too high for too long, it will shut down and the parent sensor process will restart it with a small delay. Note that enabling this feature on old or unsupported kernels could lead to system instability. Testing in similar non-production environments is strongly recommended.

This feature can be turned on/off from the agent config page in the UI and they can be turned on/off in each agent config profiles.

Meltdown is a side channel attack that abuses the speculative execution and cache features in the CPU (https://meltdownattack.com/). It allows an attacker to read privileged-domain data from an unprivileged domain, e.g., reading kernel memory from a user space application without ring 0 privileges. Meltdown detection currently supports CentOS 7 and Ubuntu 16.04.

7.2.13 Follow User Logon

Follow User Logon events report descendant processes (up to 4 levels) that are executed after a User Logon event process (SSH, RDP, etc.). Processes reported under this Follow User Logon event are for auditing purposes and not necessary having any security events.

7.2.14 Follow Process

Follow Process events report processes that match user forensic config rules based on process attributes such as binary path, command string, etc. Processes reported under this Follow Process event are for auditing purposes and not necessary having any security events.

Example 1: Report processes run by cmd.exe or powershell.exe

> Event Type = Follow Process AND (Process Info - Exec Path contains cmd.exe OR Process Info - Exec Path contains powershell.exe)

Example 2: Report any processes which are created by winword.exe or excel.exe or powerpnt.exe.

> Event Type = Follow Process with_ancestor (Process Info - Exec Path contains winword.exe OR Process Info - Exec Path contains excel.exe OR Process Info - Exec Path contains powerpnt.exe)

Note: Follow Process events can be tracked by one of following process signals:

- Process Info - Exec Path
- Process Info - Command String
- Process Info - Username
- Follow Process - Parent Exec Path
- Follow Process - Parent Command String
- Follow Process - Parent Username

7.3 Forensic configuration

Forensics feature uses intent-based configuration. Intents specify how to apply forensic profiles to inventory filters. Forensic profile consists of multiple forensic rules. Note that profiles in an intent are applied in order from top to bottom.

7.3.1 Forensic rules

Note: The maximum number of rules per root scope is 100.

7.3.1.1 Adding a forensic rule

This section explains how to add new forensic rules.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Click **Create Rule**. The **Rules** pane appears.
4. Enter the appropriate values in the following fields.

Field	Description
Rule Name	Enter a name for the rule. Name cannot be empty.
Ownership scope	Enter an ownership scope for this rule.
Actions	Select actions when this rule is triggered. **Record** means matching security events will be persisted for further analysis. **Alert** action means to publish matching security events to Tetration Alert system.
Severity	Select severity level of this rule: **LOW**, **MEDIUM**, **HIGH**, **CRITICAL** or **REQUIRES IMMEDIATE ACTION**.
Clause	Enter a rule clause. A clause must contain security event signals from **either** a process forensic event **or** a workload event. A clause is invalid if it contains both process and workload signals.

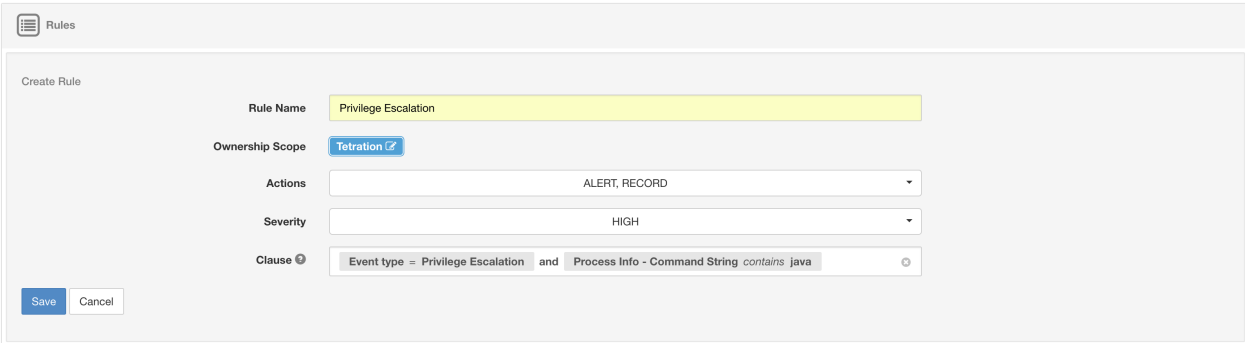


Fig. 7.3.1.1.1: Create rule

5. Click **Save**.

7.3.1.2 Editing a forensic rule

This section explains how to edit forensic rules.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Click **Create Rule**. The **Rules** pane appears.
4. Enter the appropriate values in the following fields.

Field	Description
Rule Name	Update a name for the rule. Name cannot be empty.
Ownership scope	Update an ownership scope for this rule.
Actions	Update actions when this rule is triggered. **Record** means matching security events will be persisted for further analysis. **Alert** action means to publish matching security events to Tetration Alert system.
Severity	Update severity level of this rule: **LOW**, **MEDIUM**, **HIGH**, **CRITICAL** or **REQUIRES IMMEDIATE ACTION**.
Clause	Update a rule clause. A clause must contain security event signals from **either** a process forensic event **or** a workload event. A clause is invalid if it contains both process and workload signals.

5. Click **Save**.

7.3.1.3 Basic forensic rule composition

A forensic rule must contain **exactly one** forensic event type (e.g. **Event Type == Unseen Command**). The following optional clauses should use attributes of that event (e.g. **Unseen Command - Parent Uptime**).

Below is one example using **Unseen Command** event type. Please look at our default rules and MITRE rules below for more examples.

EventType = Unseen Command and Unseen Command - Parent Uptime (microseconds) >= 60000000.

7.3.1.4 Default Tetration rules

Default Tetration rules are provided to help the users to construct rules that are meaningful in their environment. These rules are displayed in the forensic config page and they are not editable. The rules are available in all root scopes.

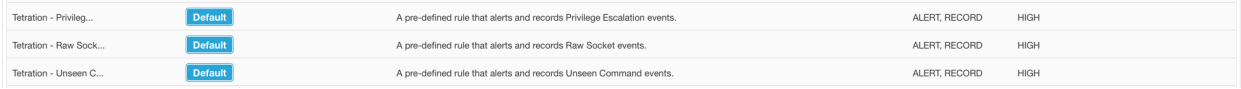


Fig. 7.3.1.4.1: Default rules

This release contains four Tetration forensic rules:

1) Name Tetration - Privilege Escalation

Clause EventType = Privlege Escalation and (**ProcessInfo - ExecPath** *doesn't contain* **sudo and ProcessInfo - ExecPath** *doesn't contain* **ping and Privilege Escalation Is** $\neq$ **Type - Suid Binary**)

Description This rule reports privilege escalation events that are not generated by setuid binaries. To reliably filter out the setuid binaries, it also filters out **sudo** and **ping** based on "ProcessInfo - ExecPath". Tetration users can also filter out other setuid binaries by defining their own rules.

2) Name Tetration - Unseen Command

Clause EventType = Unseen Command and Unseen Command - Parent Uptime (microseconds) >= 60000000 or ProcessInfo - ExecPath *contains* **/bash or ProcessInfo - ExecPath** *contains* **/sh or ProcessInfo - ExecPath** *contains* **/ksh or Parent - ExecPath** *contains* **httpd or Parent - ExecPath** *contains* **apache or Parent - ExecPath** *contains* **nginx or Parent - ExecPath** *contains* **haproxy**

Description This rule reports unseen command events that match one of the following criteria:

1. Process parent is alive for more than **60,000,000** microseconds.
2. Process ExecPath contains some type of shell, e.g., **/bash**, **/sh**, and **/ksh**.
3. Process parent ExecPath contains some type of server application, e.g., **httpd**, **apache**, **nginx**, and **haproxy**.

3) Name Tetration - Raw Socket

Clause EventType = Raw Socket Creation and (**Raw Socket - ExecPath** *doesn't contain* **ping and Raw Socket - ExecPath** *doesn't contain* **iptables and Raw Socket - ExecPath** *doesn't contain* **xtables-multi**)

Description This rule reports raw socket creation events that are not generated by **ping** and **iptables**. Tetration users can also filter out other binaries by defining their own rules.

4) Name Tetration - Network Anomaly with Unseen Command

Clause EventType = Network Anomaly and Network Anomaly - Unseen Command Count > 3 and Network Anomaly - Non-seasonal Deviation > 0

Description This rule reports network anomaly events that match the following criteria:

1. There are more than 3 Unseen Command events on the same workload within 15 minutes.
2. The *Non-seasonal PCR Deviation* is greater than `0` (which also means it is greater than or equal to `6.0` because `6.0` is the minimum reported deviation for all network anomaly events).

5) Name Tetration - Anomalous Unseen Command

Clause EventType = Unseen Command and Unseen Command - Anomaly - Score < 0.6

Description This rule reports unseen command events whose anomaly score is less than 0.6. This means only highly anomalous events whose commands do not look similar to previously observed commands are reported. The threshold 0.6 is decided based on Tetration's experiments on how similar commands are at different thresholds. See *Unseen Command* for a detailed explanation of the score.

6) Name Tetration - Unusual Parent of smss

Clause EventType = Follow Process and ProcessInfo - ExecPath *contains* **smss.exe and (Follow Process - ParentExecPath** *doesn't contain* **smss.exe and Follow Process - ParentExecPath** *doesn't contain* **System)**

Description This rule is specific for windows. This rule alerts if smss.exe has a parent that is different from another instance of smss.exe or the System process.

7) Name Tetration - Unusual Parent of wininit

Clause EventType = Follow Process and ProcessInfo - ExecPath *contains* **wininit.exe and Follow Process - ParentExecPath** *doesn't contain* **smss.exe**

Description This rule is specific for windows. This rule alerts if wininit.exe has a parent that is different from smss.exe.

8) Name Tetration - Unusual Parent of RuntimeBroker

Clause EventType = Follow Process and ProcessInfo - ExecPath *contains* **RuntimeBroker.exe and Follow Process - ParentExecPath** *doesn't contain* **svchost.exe**

Description This rule is specific for windows. This rule alerts if RuntimeBroker.exe has a parent that is different from svchost.exe.

9) Name Tetration - Unusual Parent of services

Clause EventType = Follow Process and ProcessInfo - ExecPath *contains* **services.exe and Follow Process - ParentExecPath** *doesn't contain* **wininit.exe**

Description This rule is specific for windows. This rule alerts if services.exe has a parent that is different from wininit.exe.

10) Name Tetration - Unusual Parent of lsaio

Clause EventType = Follow Process and ProcessInfo - ExecPath *contains* **lsaio.exe and Follow Process - ParentExecPath** *doesn't contain* **wininit.exe**

Description This rule is specific for windows. This rule alerts if lsaio.exe has a parent that is different from wininit.exe.

11) Name Tetration - Unusual Child of lsass

Clause (EventType = Follow Process and ProcessInfo - ExecPath *doesn't contain* **efsui.exe and ProcessInfo - ExecPath** *doesn't contain* **werfault.exe) with ancestor Process Info - ExecPath** *contains* **lsass.exe**

Description This rule is specific for windows. This rule alerts if lsass.exe has any descendants that are not efsui.exe or werfault.exe.

7.3.1.5 Default MITRE ATT&CK rules

Default MITRE ATT&CK rules are provided to alert techniques from the MITRE ATT&CK Framework (https://attack.mitre.org/). There are 24 rules pertaining to adverserial behaviour and most of them are mapped to a particular MITRE technique. The complete list of the rules is below.

1) Name Suspicious MS Office behavior

Clause (Event type = Follow Process and (Process Info - Exec Path *doesn't contain* **Windowssplwow64.exe) and (Process Info - Exec Path** *doesn't contain* **chrome.exe) and (Process Info - Exec Path** *doesn't contain* **msip.executionhost.exe) and (Process Info - Exec Path** *doesn't contain* **msip.executionhost32.exe) and (Process Info - Exec Path** *doesn't contain* **msosync.exe) and (Process Info - Exec Path** *doesn't contain* **ofccccaupdate.exe)) with ancestor (Process Info - Exec Path** *contains* **winword.exe or Process Info - Exec Path** *contains* **excel.exe or Process Info - Exec Path** *contains* **powerpnt.exe)**

Description This rule alerts and records if Microsoft Office processes (WINWORD.exe/EXCEL.exe/POWERPNT.exe) create any child processes. Based on our research we have allowed a few common child processes known to be created by these MS Office binaries, to reduce the amount of false positives.

2) Name T1015 - Accessibility features 1

Clause Event type = Follow Process (Process Info - Exec Path *contains* **cmd.exe or Process Info - Exec Path** *contains* **powershell.exe or Process Info - Exec Path** *contains* **cscript.exe or Process Info - Exec Path** *contains* **wscript.exe) and (Follow Process - Parent Exec Path** *contains* **winlogon.exe or Follow Process - Parent Exec Path** *contains* **atbroker.exe or Follow Process - Parent Exec Path** *contains* **utilman.exe)**

Description This rule alerts and records if any of the Accessibility features binaries (On-screen Keyboard, Magnifier, Sticky keys, etc) are abused and are tricked into opening cmd/powershell/cscript/wscript. The invocation of accessibility binaries is controlled by either winlogon, atbroker or utilman processes depending on from where they are

invoked (from the logon screen or after a user logs in). This rule captures suspicious child processes (cmd.exe, powershell.exe, cscript.exe, wscript.exe) of the accessibility processes (winlogon.exe, utilman.exe and atbroker.exe). Use this with **T1015 - Accessibility features 2** to also catch the additional child processes of these four suspicious child processes**

3) Name T1015 - Accessibility features 2

Clause Event type = Follow Process with ancestor ((Process Info - Exec Path *contains* **cmd.exe or Process Info - Exec Path** *contains* **powershell.exe or Process Info - Exec Path** *contains* **cscript.exe or Process Info - Exec Path** *contains* **wscript.exe) and (Follow Process - Parent Exec Path** *contains* **winlogon.exe or Follow Process - Parent Exec Path** *contains* **atbroker.exe or Follow Process - Parent Exec Path** *contains* **utilman.exe))**

Description This rule alerts and records if any of the Accessibility features binaries (On-screen Keyboard, Magnifier, Sticky keys, etc) are abused and are tricked into opening cmd.exe/powershell.exe/cscript.exe/wscript.exe. The invocation of accessibility binaries is controlled by either winlogon, atbroker or utilman processes depending on from where they are invoked (from the logon screen or after a user logs in). This rule captures child processes of the suspicious child processes of these processes (winlogon, utilman and atbroker). One should use this with **T1015 - Accessibility features 1** which alerts the suspicious child processes of accessibility binaries.

4) Name T1085 - rundll32

Clause (Event type = Follow Process and Process Info Exec Path *doesnt contain* **msiexec.exe and Process Info Exec Path** *doesnt contain* **WindowsSystem32SystemPropertiesRemote.exe with ancestor (Process Info - Exec Path** *contains* **rundll32.exe and Follow Process - Parent Exec Path** *doesnt contain* **msiexec.exe and not (Process Info -command string** *contains* **Windowssystem32shell32.dll or (Process Info -command string** *contains* **Windowssyswow64shell32.dll or (Process Info -command string** *contains* **WindowsSystem32migrationWinInetPlugin.dll))**

Description This rule alerts and records if rundll32.exe creates child processes. This binary can be called to execute arbitrary binary/dll or used by control.exe to install malicious control panel items. However, we have allowed if msiexec.exe is either the parent or the descedant of rundll32.exe. We have also permitted some of the common rundll32 commands that make use of well known dlls.

5) Name T1118 - InstallUtil

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **installutil.exe**

Description This rule alerts and records if InstallUtil.exe creates child processes.

6) Name T1121 - Regsvcs/Regasm

Clause Event type = Follow Process and (Process Info - Exec path *doesn't contain* **fondue.exe or Process Info - Exec path** *doesnt contain* **regasm.exe or Process Info - Exec path** *doesnt contain* **regsvr32.exe with ancestor (Process Info - Exec Path** *contains* **regasm.exe** or **Process Info - Exec Path** *contains* **regsvcs.exe)**

Description This rule alerts and records if regsvcs.exe or regasm.exe create child processes. However, we have permitted if fondue.exe/regasm.exe/regsvr32.exe is spawned by regasm.exe or regsvcs.exe to reduce the number of false positives.

7) Name T1127 - Trusted Developer Utilities - msbuild.exe

Clause (Event type = Unseen Command with ancestor Process Info - Exec Path *contains* **MSBuild.exe) and (Process Info - Exec Path** *doesn't contain* **Tracker.exe) and (Process Info - Exec Path** *doesn't contain* **csc.exe) and (Process Info - Exec Path** *doesn't contain* **Microsoft Visual Studio) and (Process Info - Exec Path** *doesn't contain* **al.exe) and (Process Info - Exec Path** *doesn't contain* **lc.exe) and (Process Info - Exec Path** *doesn't contain* **dotnet.exe) and (Process Info - Exec Path** *doesn't contain* **cvtres.exe) and (Process Info - Exec Path** *doesn't contain* **conhost.exe) and not (Event type = Unseen Command with ancestor (Process Info - Exec Path** *contains* **Tracker.exe or Process Info - Exec Path** *contains* **csc.exe or Process Info - Exec Path** *contains* **Microsoft Visual Studio or Process Info - Exec Path** *contains* **al.exe or Process Info - Exec Path** *contains* **lc.exe or Process Info - Exec Path** *contains* **dotnet.exe or Process Info - Exec Path** *contains* **cvtres.exe))**

Description This rule alerts and records if msbuild.exe creates child processes which do not belong to an allowlist of child processes it usually creates. This rule is currently Unseen Command based, as opposed to Follow Process, since Follow Process doesnt yet support allowing process subtrees. The current rule allows the following processes and their descedants: Tracker.exe, csc.exe, any process from "Microsoft Visual Studio" path, al.exe, lc.exe, dotnet.exe and cvtres.exe. The rule also allows conhost.exe. These processes can be seen during regular usage of MSBuild.exe (for e.g. compiling a project via Visual Studio). All the other descendants (not usual behavior) of MSBuild.exe are alerted.

8) Name T1127 - Trusted Developer Utilities - rcsi.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **rcsi.exe**

Description This rule alerts and records if rcsi.exe creates child processes.

9) Name T1127 - Trusted Developer Utilities - tracker.exe

Clause (Event type = Unseen Command with_ancestor Process Info - Exec Path *contains* **tracker.exe) and not (Event type = Unseen Command with_ancestor Process Info - Exec Path** *contains* **MSBuild.exe)**

Description This rule alerts and records if tracker.exe creates child processes and tracker itself is not a descendant of MSBuild.exe. Thus legitimate invocations of tracker via Visual Studio are approved, but other invocations are alerted. **Note:** One limitation with the Tracker.exe and the previous MSBuild.exe rules is that if an attacker uses MSBuild technique to create Tracker, and then make Tracker create a malicious child, it would not be alerted by either of the rules since Tracker having MSBuild as a ancestor is considered legitimate.

10) Name T1128 - Netsh Helper Dll

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **netsh.exe**

Description This rule alerts and records if netsh.exe creates child processes.

11) Name T1136 - Create Account

Clause Event type = User Account

Description This rule alerts and records if a new user is created.

12) Name T1138 - Application Shimming

Clause Event type = Follow Process Process Info - Exec Path *contains* **sdbinst.exe**

Description This rule alerts and records if sdbinst.exe is invoked.

13) Name T1180 - Screensaver

Clause Event type = Follow Process AND with ancestor Process Info - Exec Path *contains* **.scr**

Description This rule alerts and records if a process is created with ".scr" in the exec path.

14) Name T1191 - CMSTP

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **cmstp.exe**

Description This rule alerts and records if cmstp.exe creates child processes.

15) Name T1202 - Indirect Command Execution - forfiles.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **forfiles.exe**

Description This rule alerts and records if forfiles.exe creates child processes.

16) Name T1202 - Indirect Command Execution - pcalua.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **pcalua.exe**

Description This rule alerts and records if pcalua.exe creates child processes.

17) Name T1216 - Signed Script Proxy Execution - pubprn.vbs

Clause Event type = Follow Process with ancestor ((Process Info - Exec Path *contains* **cscript.exe or Process Info - Exec Path** *contains* **wscript.exe) and Process Info - Command String** *contains* **.vbs and Process Info - Command String** *contains* **script)**

Description This rule alerts and records if any vbs script is run using wscript.exe or cscript.exe, to create a new process, with a parameter "script". This technique could be used by an attacker to execute pubprn.vbs with a script parameter pointing to a malicious sct file which then gives code execution.

18) Name T1218 - Signed Binary Proxy Execution - msiexec.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **msiexec.exe**

Description This rule alerts and records if msiexec.exe creates child processes.

19) Name T1218 - Signed Binary Proxy Execution - odbcconf.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **odbcconf.exe**

Description This rule alerts and records if odbcconf.exe creates child processes.

20) Name T1218 - Signed Binary Proxy Execution - Register-CimProvider

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **Register-CimProvider.exe**

Description This rule alerts and records if Register-CimProvider.exe creates child processes.

21) Name T1220 - XSL Script Processing - msxsl.exe

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **msxsl.exe**

Description This rule alerts and records if msxsl.exe creates child processes.

22) Name T1220 - XSL Script Processing - wmic

Clause Event type = Follow Process and (Process Info - Exec Path *contains* **wmic.exe and Process Info - Command String** *contains* **.xsl)**

Description This rule alerts and records if an xsl script is used by wmic. This can be used to launch arbitrary binaries.

23) Name T1223 - Compiled HTML Files

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **hh.exe**

Description This rule alerts and records if hh.exe creates child processes.

24) Name T1003 - Credential Dumping - Lsass

Clause Event type = Follow Process and Process Info - Exec Path *contains* **procdump.exe and Process Info - Command String** *contains* **lsass**

Description This rule alerts and records if procdump.exe is used to dump the memory of lsass processes.

25) Name T1140 - Deobfuscate/Decode Files or Information

Clause Event type = Follow Process and Process Info - Exec Path *contains* **certutil.exe and (Process Info - Command String** *matches* **.*encode\s.* or Process Info - Command String** *matches* **.*decode\s.***

Description This rule alerts and records if certutil.exe is used to either encode or decode a file. This technique is often used by attackers to decode their encoded payload on the victim machine.

26) Name T1076 - Remote Desktop Protocol

Clause Event type = Follow Process and Process Info - Exec Path *contains* **tscon.exe**

Description This rule alerts and records if tscon.exe is executed. Attackers can use tscon.exe to hijacking existing RDP sessions.

27) Name T1197 - BITS Jobs - Powershell

Clause Event type = Follow Process and Process Info - Exec Path *contains* **powershell.exe and Process Info - Command String** *contains* **Start-BitsTransfer**

Description This rule alerts and records if the powershell.exe is used to run the cmdlet Start-BitsTransfer to copy/move files.

28) Name T1170 - MSHTA

Clause Event type = Follow Process with ancestor Process Info - Exec Path *contains* **mshta.exe**

Description This rule alerts and records if mshta.exe is used to run malicious HTA scripts that spawn child processes.

29) Name T1158 - Hidden Files and Directories

Clause Event type = Follow Process and (Process Info - Exec Path *contains* **attrib.exe and Process Info - Command String** *contains* **+h)**

Description This rule alerts and records if attrib.exe is used to set a file/directory as hidden.

30) Name T1114 - Email Collection

Clause Event type = Follow Process (Process Info - Command String *matches* **.*.(ost|pst)(\s|''|').* or Process Info - Command String** *matches* **.*.(ost|pst)$) Process Info - Exec Path** *doesn't contain* **outlook.exe**

Description This rule alerts and records if email files (.ost and .pst) are accessed from any other process other than outlook.exe.

31) Name T1070 - Indicator Removal on Host - Event Log

Clause Event type = Follow Process and Process Info - Exec Path *contains* **wevtutil.exe and Process Info - Command String** *matches* **.*\s(cl|clear-log)\s.***

Description This rule alerts and records if wevtutil.exe is used to clear event logs.

32) Name T1070 - Indicator Removal on Host - USN

Clause Event type = Follow Process and Process Info - Exec Path *contains* **fsutil.exe and Process Info - Command String** *matches* **.*\susn\s.* and Process Info - Command String** *matches* **.*\sdeletejournal.***

Description This rule alerts and records if fsutil.exe is used to delete USN journals.

33) Name T1053 - Scheduled Task

Clause Event type = Follow Process and Process Info - Exec Path *contains* **schtasks.exe and Process Info - Command String** *contains* **create**

Description This rule alerts and records if schtasks.exe is used to create new scheduled tasks.

34) Name T1003 - Credential Dumping - Vaultcmd

Clause Event type = Follow Process and Process Info - Exec Path *contains* **vaultcmd.exe and Process Info - Command String** *matches* **.*\/list.***

Description This rule alerts and records if vaultcmd.exe is used access Windows Credentials vault.

35) Name T1003 - Credential Dumping - Registry

Clause Event type = Follow Process and Process Info - Exec Path *contains* **reg.exe and ((Process Info - Command String** *contains* **save or Process Info - Command String** *contains* **export) and (Process Info - Command String** *contains* **hklm or Process Info - Command String** *contains* **hkey_local_machine) and (Process Info - Command String** *contains* **sam or Process Info - Command String** *contains* **security or Process Info - Command String** *contains* **system))**

Description This rule alerts and records if reg.exe is used dump certain registry hives.

36) Name T1201 - Password Policy Discovery 1

Clause Event type = Follow Process and Process Info - Exec Path *contains* **chage and Process Info - Command String** *contains* **-l**

Description This rule alerts and records if chage utility is used to list the password policy (password age policy) on a linux machine.

37) Name T1081 - Credentials in Files - Linux

Clause Event type = Follow Process and (Process Info - Exec Path *contains* **cat or Process Info - Exec Path** *contains* **grep) and (Process Info - Command String** *contains* **.bash_history or Process Info - Command String** *contains* **.password or Process Info - Command String** *contains* **.passwd)**

Description This rule alerts and records if attempts are made to search for passwords stored in files on a linux machine.

38) Name T1081 - Credentials in Files - Windows

Clause Event type = Follow Process and Process Info - Exec Path *contains* **findstr.exe and Process Info - Command String** *contains* **password**

Description This rule alerts and records if attempts are made to search for passwords stored in files on a windows machine.

39) Name T1089 - Disabling Security Tools

Clause Event type = Follow Process and ((Process Info - Exec Path *contains* **fltmc.exe and Process Info - Command String** *contains* **unload sysmon) or (Process Info - Exec Path** *contains* **sysmon.exe and Process Info - Command String** *contains* **/u))**

Description This rule alerts and records if attempts are made to unload sysmon driver using fltmc.exe or sysmon.exe

7.3.2 Forensic profiles

7.3.2.1 Add a profile

This section explains how to add new forensic profiles.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Click **Create Profile**. The **Profiles** pane appears.
4. Enter the appropriate values in the following fields.

Field	Description
Name	Enter a name for the profile. Name cannot be empty.
Ownership scope	Enter an ownership scope for this profile.
Rules	Add rules into this profile.

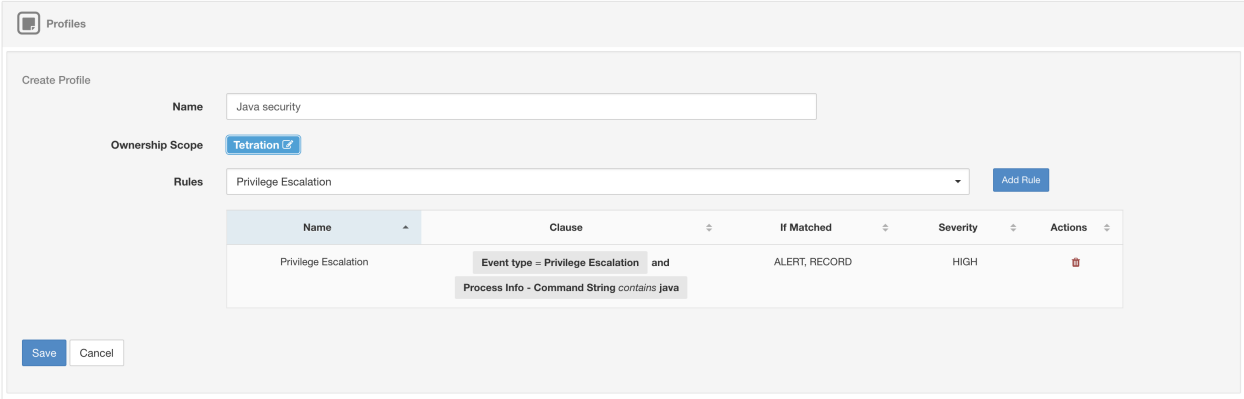


Fig. 7.3.2.1.1: Create profile

5. Click **Save**.

7.3.2.2 Edit a profile

This section explains how a user edit forensic profiles.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Find the profile you want to edit and click the **pencil** icon in the column on the right.
4. Enter the appropriate values in the following fields.

Field	Description
Name	Update a name for the profile. Name cannot be empty.
Ownership scope	Update an ownership scope for this profile.
Rules	Add/remove rules into this profile.

5. Click **Save**.

7.3.2.3 Clone a profile

This section explains how a user clone forensic profiles.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Find the profile you want to clone and click the **clone** icon in the column on the right.
4. Enter the name for the cloned profile.
5. Click **Save**.

7.3.2.4 Default profile - Tetration Profile

The Tetration profile contains eleven default forensic rules and can be added to intents. It is not editable by the user but it can be cloned. The cloned default forensic profile is editable.

Fig. 7.3.2.4.1: Default profiles

7.3.2.5 Default profile - MITRE ATT&CK Profile

The MITRE ATT&CK Profile contains 39 MITRE ATT&CK rules and can be added to intents. It is not editable by the user but it can be cloned. The cloned profile is editable. MITRE ATT&CK Profile includes the following rules:

1. Suspicious MS Office behavior
2. T1015 - Accessibility features 1
3. T1015 - Accessibility features 2
4. T1085 - rundll32
5. T1118 - InstallUtil
6. T1121 - Regsvcs/Regasm
7. T1127 - Trusted Developer Utilities - msbuild.exe
8. T1127 - Trusted Developer Utilities - rcsi.exe
9. T1127 - Trusted Developer Utilities - tracker.exe
10. T1128 - Netsh Helper Dll
11. T1136 - Create Account
12. T1138 - Application Shimming
13. T1180 - Screensaver
14. T1191 - CMSTP
15. T1202 - Indirect Command Execution - forfiles.exe
16. T1202 - Indirect Command Execution - pcalua.exe
17. T1216 - Signed Script Proxy Execution - pubprn.vbs
18. T1218 - Signed Binary Proxy Execution - msiexec.exe
19. T1218 - Signed Binary Proxy Execution - odbcconf.exe
20. T1218 - Signed Binary Proxy Execution - Register-CimProvider
21. T1220 - XSL Script Processing - msxsl.exe
22. T1220 - XSL Script Processing - wmic
23. T1223 - Compiled HTML Files
24. T1003 - Credential Dumping - Lsass
25. T1140 - Deobfuscate/Decode Files or Information
26. T1076 - Remote Desktop Protocol

27. T1197 - BITS Jobs - Powershell
28. T1170 - MSHTA
29. T1158 - Hidden Files and Directories
30. T1114 - Email Collection
31. T1070 - Indicator Removal on Host - Event Log
32. T1070 - Indicator Removal on Host - USN
33. T1053 - Scheduled Task
34. T1003 - Credential Dumping - Vaultcmd
35. T1003 - Credential Dumping - Registry
36. T1201 - Password Policy Discovery 1
37. T1081 - Credentials in Files - Linux
38. T1081 - Credentials in Files - Windows
39. T1089 - Disabling Security Tools

7.3.3 Forensic intents

7.3.3.1 Adding an intent

This section explains how to add new forensic intents.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Click **Create Intents**. The **Intents** pane appears.
4. Enter the appropriate values in the following fields.

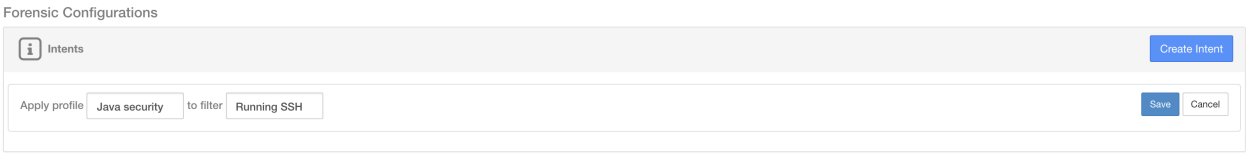


Fig. 7.3.3.1.1: Create intent

Field	Description
Profile	Select a forensic profile from the list.
Filter	Select an inventory filter from the list.

5. Click **Save**.

7.3.3.2 Editing an intent

This section explains how a user edit forensic intents.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click the **Settings menu** in the top-right corner.
2. Select **Forensic Config**. The **Forensic Configurations** appears.
3. Find the intent you want to edit and click the **pencil** icon in the column on the right.
4. Enter the appropriate values in the following fields.

Field	Description
Profile	Select a forensic profile from the list.
Filter	Select an inventory filter from the list.

5. Click **Save**.

7.3.4 Change Log

Site Admins and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each forensic rule, profile and intent by clicking on the icon as shown below.

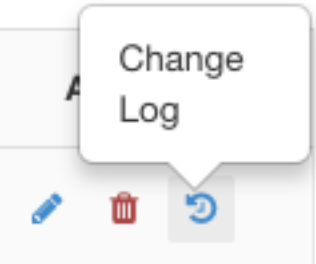


Fig. 7.3.4.1: Change log

These users can also view a list of deleted rules, profiles and intents by clicking on the **View Deleted Rules/Profiles/Intents** link below the corresponding table.

For more information on the **Change Log** see *Change Log*. Root scope owners are restricted to viewing change log entries for entities belonging to their scope.

7.4 Forensic visualization

7.4.1 Accessing forensic page

This section explains how to access forensic page.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. Click on **Security** link on the left panel.
2. Click on **Forensics** item. Forensic page appears.

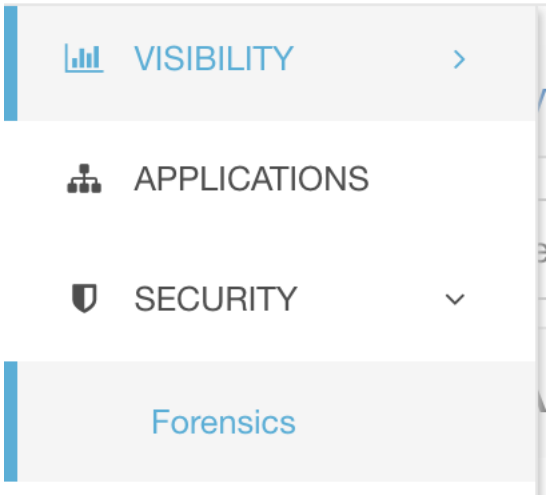


Fig. 7.4.1.1: Security forensic

7.4.2 Browsing forensic events

This section explains how to browse matching forensic events.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system and navigate to the forensic page.

1. Choose a specific range in the **Time Range Picker** at the top of the page.
2. Select **Severity** drop-down.
3. In **Filters**, enter filters for matching forensic events and click on "Filter Forensic Events".
4. Table of matching forensic events is updated, according to the selected time range, severity and filters.

Note: Forensic events are visible under the root scope level and will not visible upon switching to sub/child scopes.

7.4.3 Inspecting an forensic event

This section explains how to inspect forensic events.

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner (Root Scope)** in the system.

1. Click on the event to be inspected. **Process detail** pane appears.

Fig. 7.4.3.1: Forensic event table

2. On lineage tree, click on process to be inspected for details.

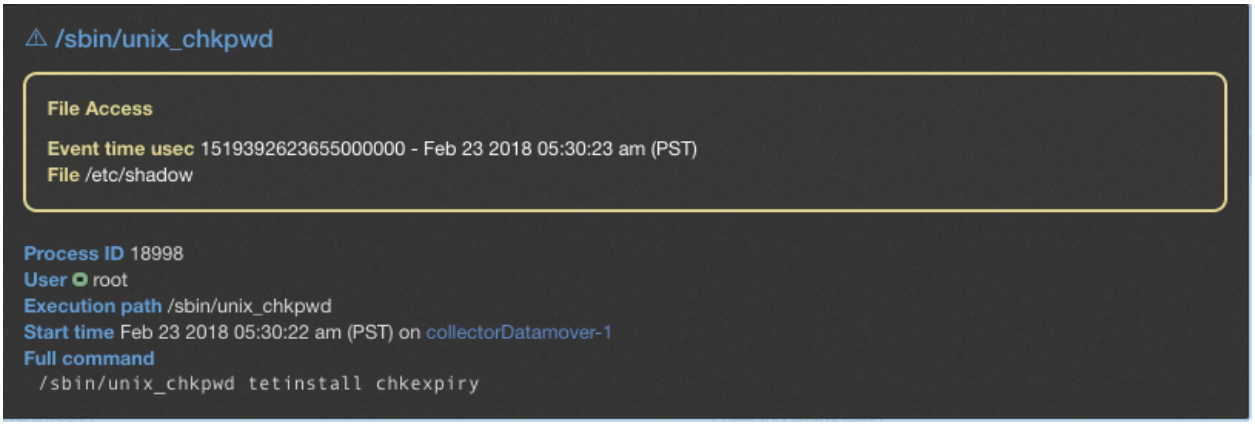


Fig. 7.4.3.2: Forensic process details

7.5 Fields Displayed in Forensic Events

Each Forensic Event has a number of fields which provide useful data. There are a few fields common to all the different types of forensic events, and there are a few fields unique to a particular forensic event.

Below is a list of the fields that are part of the UI. The first table describes fields common to all forensics event, followed by a table that describes process information that is displayed with each alert and then the tables with unique fields per forensic event. Note that some of the fields may be present in multiple tables, because of the way the data is stored and exported.

7.5.1 Common Fields

Field	Description
Bin attr ctime	Changed time in linux/ Create time in windows of the binary
Bin attr hash	Sha256 hash of the binary
Bin attr mtime	Modified time of the binary
Bin attr name	Name of the binary on the file system
Bin attr size	Size of the binary on the file system
Bin exec path	Full path of the binary
Cmdline	Full command line of the process that gets executed
Event time usec	Time (in microseconds) when this event is observed

7.5.2 Process Info

Field	Description
Process ID	Process ID of the process
Parent Process ID	Process ID of the parent of the process
User	User that executed the process
Execution path	Full path of the binary that corresponds to the process.
Start time	Time when the process was started
Full command	Full command line of the process that gets executed

7.5.3 Privilege Escalation

Field	Description
Parent cmdline	Full command line of the parent of the process
Parent exe	Full path of the parent of the process
Parent Uptime (microseconds)	Time since the parent of the process was executed
Parent Username	User that executed the parent of the process
Types bitmap suid binary	Indicates whether the binary has the suid bit set

7.5.4 User Logon

Field	Description
Auth type password	Indicates password authentication
Auth type pubkey	Indicates key based authentication
Type login ssh	Indicates that a user logged in via ssh
Type login win batch	Indicates windows batch login (Type 4, eg schtasks)
Type login win cached	Indicates logon via cached credentials (Type 11, CachedIntetractive)
Type login win interactive	Indicates interactive logon (Type 2, eg RDP)
Type login win network cleartext	Indicates logon via ssh (Type 8)
Type login win network	Indicates network login (Type 3, eg Psexec)
Type login win new cred	Indicates the usage of new credentials (Type 9, eg Runas command)
Type login win remote interactive	Indicates remote logon (Type 10, eg RDP)
Type login win service	Indicates that a service was started by SCM (Type 5)
Type login win unlock	Indicates that the workstation was unlocked (Type 7)
Src IP	The source IP from which the login event was generated
Src Port	The source port from which the login event was generated
Username	Username associated with the log in event

7.5.5 User Logon Failed

Field	Description
Auth type password	Indicates password authentication
Auth type pubkey	Indicates key based authentication
Type login ssh	Indicates that a user logged in via ssh
Type login win batch	Indicates windows batch login (Type 4, eg schtasks)
Type login win cached	Indicates logon via cached credentials (Type 11, CachedIntetractive)
Type login win interactive	Indicates interactive logon (Type 2, eg RDP)
Type login win network cleartext	Indicates logon via ssh (Type 8)
Type login win network	Indicates network login (Type 3, eg Psexec)
Type login win new cred	Indicates the usage of new credentials (Type 9, eg Runas command)
Type login win remote interactive	Indicates remote logon (Type 10, eg RDP)
Type login win service	Indicates that a service was started by SCM (Type 5)
Type login win unlock	Indicates that the workstation was unlocked (Type 7)
Src IP	The source IP from which the login event was generated
Src Port	The source port from which the login event was generated
Username	Username associated with the log in event

7.5.6 Shellcode

Field	Description
Signal sources bitmap cmd as sh no tty	Indicates that a shell process has no tty associated with it
Signal sources bitmap powershell	Indicates that the process has powershell dll loaded (System.Management.Automation)

7.5.7 File Access

Field	Description
File	Full path of the file that was accessed
Perm read perm	Indicates that the file had Read permission
Perm read write perm	Indicates that the file had Read and Write permissions
Perm write perm	Indicates that the file had Write permission

7.5.8 User Account

Field	Description
Username	Username of the user that was created
Ops acct add	Indicates that a new account was added

7.5.9 Unseen Command

Field	Description
Anomaly - Score	Score (0 to 1.0) indicating how frequently the command line was seen previously, lower score implies that the command is more anomalous
Anomaly - Similarity - High	True if the anomaly score is larger than 0.8 and is smaller than 1
Anomaly - Similarity - Medium	True if the anomaly score is larger than 0.6 and is smaller than or equal to 0.8
Anomaly - Similarity - Low	True if the anomaly score is larger than 0 and is smaller than or equal to 0.6
Anomaly - Similarity - Seen	True if the anomaly score is 1, i.e. the same command has been seen before
Anomaly - Similarity - Unique	True if the anomaly score is 0, i.e. the command has never been seen before
Parent cmdline	Full command line of the parent process
Parent exepath	Binary path of the parent process
Parent uptime	Time since the parent process was executed
Parent username	Username of the user that executed the parent process
Sensor uptime	Uptime of the sensor

7.5.10 Unseen Library

Field	Description
Lib Path	The full path of the library file that was previously not associated to the process

7.5.11 Raw Socket Creation

Field	Description
Exe Path	Full path of the process that created the raw socket

7.5.12 Library Changed

Field	Description
Library changed name	The full path of the Library that was changed

7.5.13 Side Channel

Field	Description
Signal sources bitmap meltdown	Indicates the use of Meltdown exploit

7.5.14 Follow User Logon

Field	Description
Username	Username that executed the process

7.5.15 Follow Process

Field	Description
Parent cmdline	Full command line of the parent process
Parent exepath	Binary path of the parent process
Parent uptime usec	Time since the parent process was executed
Parent username	Username of the user that executed the parent process
Time since last changed usec	Time elapsed between the process start time and its binary file change time (mtime)
Username	Username of the user that executed the process

7.5.16 Network Anomaly

Please see *Network Anomaly Detection page* for the list of attributes associated with Network Anomaly events.

7.6 Forensic Analysis - Searchable fields

The below tables describe searchable fields in the Forensics Analysis page search bar

7.6.1 Miscellaneous Fields

Field	Description
Forensic Rule Name	Events labeled by a particular forensic rule
Hostname	Events from a particular hostname
Sensor ID	Events from a particular Sensor
Severity	Events of a particular severity

7.7 Search Terms in Forensic Analysis

7.7.1 Common Fields

These fields are common to various event types. They have the prefix "Event name - Event", e.g., "Binary Changed - Binary Attribute - CTime (epoch nanoseconds)"

Field	Description
Binary Attribute - CTime (epoch nanoseconds)	Changed time in linux/ Create time in windows of the binary
Binary Attribute - Hash	Sha256 hash of the binary
Binary Attribute - MTime (epoch nanoseconds)	Modified time of the binary
Binary Attribute - Filename	Name of the binary on the file system
Binary Attribute - Size (bytes)	Size of the binary on the file system
Event Binary Path	Full path of the binary
Command Line	Full command line of the process that gets executed

7.7.2 Binary Changed

There are no other search terms other than the ones described in "Common Fields" table.

7.7.3 File Access

File Access search terms have the prefix "File Access - ", e.g., "File Access - Filename"

Field	Description
Filename	Full path of the file that was accessed
Is = Permission - Read	Indicates that the file had Read permission
Is = Permission - ReadWrite	Indicates that the file had Read and Write permissions
Is = Permission - Write	Indicates that the file had Write permission

7.7.4 Follow Process

Follow Process search terms have the prefix "Follow Process - ", e.g., "Follow Process - Parent Command Line"

Field	Description
Parent Command Line	Full command line of the parent process
Parent Exec Path	Binary path of the parent process
Parent Uptime (microseconds)	Time since the parent process was executed
Parent Username	Username of the user that executed the parent process
Process Start Time Since Last File Changed (microseconds)	Time elapsed between process start and the most recent (corresponding)file change
Username	Username associated with the process being followed

7.7.5 Follow User Logon

Follow User Logon search terms have the prefix "Follow User Logon - ", e.g., "Follow User Logon - Username"

Field	Description
Username	Username that is associated with a process

7.7.6 Ldap

Ldap search terms have the prefix "Ldap - ", e.g., "Ldap - Department"

Field	Description
Department	AMS Ldap user department associated with the proces username (if available)
Description	AMS Ldap user description associated with the proces username (if available)
Username	AMS Ldap username associated with the process (if available)

7.7.7 Library Changed

Library Changed search terms have the prefix "Library Changed - ", e.g., "Library Changed - Department"

Field	Description
Lib Filename	The full path of the Library that was changed

7.7.8 Privilege Escalation

Privilege Escalation search terms have the prefix "Privilege Escalation - ", e.g., "Privilege Escalation - Parent Command Line"

Field	Description
Parent Command Line	Full command line of the parent of the process
Parent Exec Path	Full path of the parent of the process
Parent Uptime (microseconds)	Time since the parent of the process was executed
Parent Username	User that executed the parent of the process
Type - Suid Binary	Indicates whether the binary has the suid bit set

7.7.9 Process Info

Process Info search terms have the prefix "Process Info - ", e.g., "Process Info - Binary Hash"

Field	Description
Binary Hash	Hash of the binary associated with the process
Command String Tokenized	Tokenized command line of the process.
Command String	Full command line of the process
Exec Path	Full path of the binary that corresponds to the process.

7.7.10 Raw Socket

Raw Socket search terms have the prefix "Raw Socket - ", e.g., "Raw Socket - Exec Path"

Field	Description
Exec Path	Full path of the process that created the raw socket

7.7.11 Shellcode

Shellcode search terms have the prefix "Shellcode - ", e.g., "Shellcode - Source - Not From Login"

Field	Description
Source - Not From Login	Indicates that a shell process has no tty associated with it
Source - Powershell	Indicates that the process has powershell dll loaded (System.Management.Automation)

7.7.12 Side Channel

Side Channel search terms have the prefix "Shellcode - ", e.g., "Shellcode - Source - Meltdown"

Field	Description
Source - Meltdown	Indicates the use of Meltdown exploit

7.7.13 Unseen Command

Unseen Command search terms have the prefix "Unseen Command - ", e.g., "Unseen Command - Anomaly - Similarity - High"

Field	Description
Anomaly - Score	Score (0 to 1.0) indicating how frequently the command line was seen previously, lower score implies that the command is more anomalous
Anomaly - Similarity - High	True if the anomaly score is larger than 0.8 and is smaller than 1
Anomaly - Similarity - Medium	True if the anomaly score is larger than 0.6 and is smaller than or equal to 0.8
Anomaly - Similarity - Low	True if the anomaly score is larger than 0 and is smaller than or equal to 0.6
Anomaly - Similarity - Seen	True if the anomaly score is 1, i.e. the same command has been seen before
Anomaly - Similarity - Unique	True if the anomaly score is 0, i.e. the command has never been seen before
Parent Cmdline	Full command line of the parent process
Parent Exepath	Binary path of the parent process
Parent Uptime	Time since the parent process was executed
Parent Username	Username of the user that executed the parent process
Sensor Uptime	Uptime of the sensor
Anomaly - Latest Similar Commands	5 latest previously observed command which are similar to the command of the event

7.7.14 Unseen Library

Unseen Library search terms have the prefix "Unseen Library - ", e.g., "Unseen Library - Lib Filename"

Field	Description
Lib Filename	The full path of the library file that was previously not associated to the process

7.7.15 User Account

User Account search terms have the prefix "User Account - ", e.g., "User Account - Account Name"

Field	Description
Account Name	Username of the user that was created
Operation - Add Account	Indicates that a new account was added

7.7.16 User Logon

User Logon search terms have the prefix "User Logon - ", e.g., "User Logon - Auth Type - Password"

Field	Description
Auth Type - Password	Indicates password authentication
Auth type - Pubkey	Indicates key based authentication
Login Type - Login Via SSH	Indicates that a user logged in via ssh
Login Type - Windows Login Batch	Indicates windows batch login (Type 4, eg schtasks)
Login Type - Windows Login Cached	Indicates logon via cached credentials (Type 11, CachedIntetractive)
Login Type - Windows Login Interactive	Indicates interactive logon (Type 2, eg RDP)
Login Type - Windows Network Cleartext	Indicates logon via ssh (Type 8)
Login Type - Windows Network	Indicates network login (Type 3, eg Psexec)
Login Type - Windows Login New Credential	Indicates the usage of new credentials (Type 9, eg Runas command)
Login Type - Windows Login Remote Interactive	Indicates remote logon (Type 10, eg RDP)
Login Type - Windows Login Service	Indicates that a service was started by SCM (Type 5)
Login Type - Windows Login Unlock	Indicates that the workstation was unlocked (Type 7)
Source IP	The source IP from which the login event was generated
Source Port	The source port from which the login event was generated
Username	Username associated with the log in event

7.7.17 User Logon Failed

User Logon Failed search terms have the prefix "User Logon Failed - ", e.g., "User Logon Failed - Auth Type - Password"

Field	Description
Auth Type - Password	Indicates password authentication
Auth type - Pubkey	Indicates key based authentication
Login Type - Login Via SSH	Indicates that a user logged in via ssh
Login Type - Windows Login Batch	Indicates windows batch login (Type 4, eg schtasks)
Login Type - Windows Login Cached	Indicates logon via cached credentials (Type 11, CachedIntetractive)
Login Type - Windows Login Interactive	Indicates interactive logon (Type 2, eg RDP)
Login Type - Windows Network Cleartext	Indicates logon via ssh (Type 8)
Login Type - Windows Network	Indicates network login (Type 3, eg Psexec)
Login Type - Windows Login New Credential	Indicates the usage of new credentials (Type 9, eg Runas command)
Login Type - Windows Login Remote Interactive	Indicates remote logon (Type 10, eg RDP)
Login Type - Windows Login Service	Indicates that a service was started by SCM (Type 5)
Login Type - Windows Login Unlock	Indicates that the workstation was unlocked (Type 7)
Source IP	The source IP from which the login event was generated
Source Port	The source port from which the login event was generated
Username	Username associated with the log in event

7.8 Forensics alerts

Forensic events can be found in the Tetration Alert System if their matching rules contain an **Alert** action.

7.8.1 Accessing forensic alerts

This section explains how to access forensic alerts.

Before You Begin

- You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.
- You must turn on alerts for **Forensics** alert source (please refer to Section lab-apps-turnon-alerts for more details).

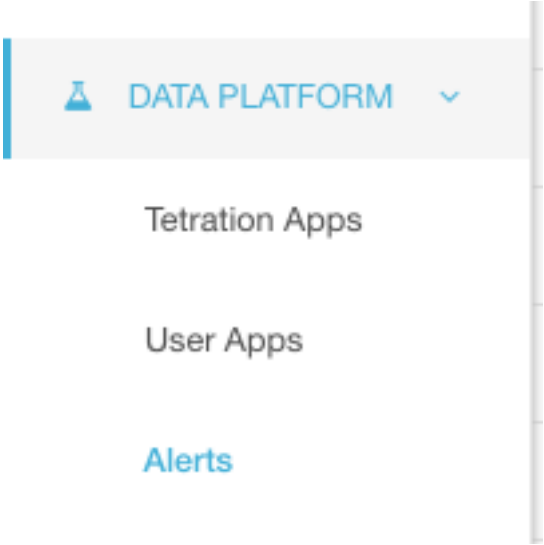


Fig. 7.8.1.1: Forensic alert

1. From the left toolbar, click on **Data Platform**.
2. Click on **Alerts**. Alert page appears.

7.8.2 Checking alert details

Before You Begin

You must login as **Site Admin**, **Customer Support** or **Scope Owner** in the system.

1. From the alert page, click on the alert to be checked.
2. Click on profile/rule to see the details of the matching forensic profile/rule. Note that if the matching profile/rule is updated after alerts are raised, there will be a warning indicator.

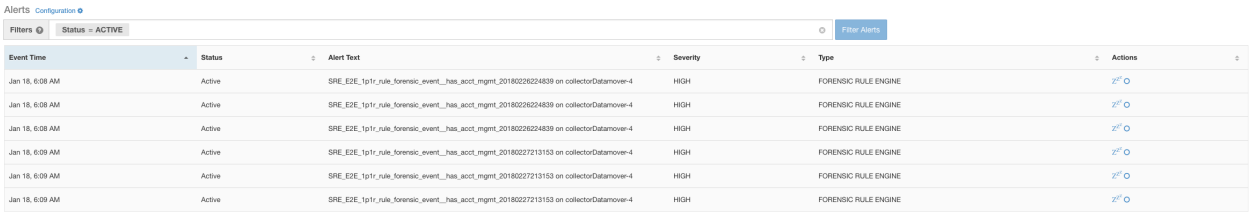


Fig. 7.8.2.1: Forensic alert page

In addition, you can snooze or include/exclude an alert. Please refer to Section *Current Alerts* for more details.

7.8.3 External integration

Forensics alerts can be sent to external monitoring tools such as syslog. The forensics alert is sent in JSON format. The JSON field defitions are defeined in the section "Fields Displayed in Forensic Events" above.

A sample JSON Kafka output is shown below:

```
{
  "severity": "HIGH",
  "tenant_id": 0,
  "alert_time": 1595573847156,
  "alert_text": "Tetration - Anomalous Unseen Command on collectorDatamover-1",
  "key_id":
↪"d89f926cddc7577553eb8954e492528433b2d08e:5efcfdf5497d4f474f1707c2:5efcfdf6497d4f474f1707d6:20196:0
↪NOT_SEEN",
  "alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type='TETRATION',␣
↪location_name='forensics', location_grain='MIN', root_scope_id=
↪'5efcfdf5497d4f474f1707c2'}/
↪db10d21631eebefc3b8d3aeaba5a0b1b45f4259e85b591763d7eaee9161ca076",
  "root_scope_id": "5efcfdf5497d4f474f1707c2",
  "type": "FORENSICS",
  "event_time": 1595573795135,
  "alert_details": "{\"Sensor Id\":\"d89f926cddc7577553eb8954e492528433b2d08e\",\
↪"Hostname\":\"collectorDatamover-1\",\"Process Id\":20196,\"scope_id\":\
↪"5efcfdf5497d4f474f1707c2\",\"forensic\":{\"Unseen Command\":\"true\",\"Unseen␣
↪Command - Sensor Uptime (microseconds)\":\"34441125356\",\"Unseen Command - Parent␣
↪Uptime (microseconds)\":\"35968418683\",\"Unseen Command - Parent Username\":\"root\
↪",\"Unseen Command - Parent Command Line\":\"svlogd -tt /local/logs/tetration/efe/ \
↪",\"Unseen Command - Parent Exec Path\":\"/sbin/svlogd\",\"Unseen Command - Anomaly␣
↪- Score\":\"0\",\"Unseen Command - Anomaly - Similarity - Unique\":\"true\",\
↪"Process Info - Command String\":\"gzip \",\"Process Info - Exec Path\":\"/bin/gzip\
↪"},\"profile\":{\"id\":\"5efcfdf6497d4f474f1707e4\",\"name\":\"Tetration Profile\",\
↪"created_at\":1593638390,\"updated_at\":1593638390,\"root_app_scope_id\":\
↪"5efcfdf5497d4f474f1707c2\"},\"rule\":{\"id\":\"5efcfdf6497d4f474f1707d6\",\"name\
↪":\"Tetration - Anomalous Unseen Command\",\"clause_chips\":\"[{\\\"type\\\":\\\
↪"filter\\\",\\\"facet\\\":{\\\"field\\\":\\\"event_type\\\",\\\"title\\\":\\\"Event␣
↪type\\\",\\\"type\\\":\\\"STRING\\\"},\\\"operator\\\":{\\\"label\\\":\\\"\\u003d\\\
↪",\\\"type\\\":\\\"eq\\\"},\\\"displayValue\\\":\\\"Unseen Command\\\",\\\"value\\\
↪":\\\"Unseen Command\\\"},{\\\"type\\\":\\\"filter\\\",\\\"facet\\\":{\\\"field\\\
↪":\\\"forensic_event__cmd_not_seen_data__cmdline_anomaly_info__score\\\",\\\
↪"title\\\":\\\"Unseen Command - Anomaly - Score\\\",\\\"type\\\":\\\"NUMBER\\\"},\\\
↪"operator\\\":{\\\"label\\\":\\\"\\u003c\\\",\\\"type\\\":\\\"lt\\\"},\\\
↪"displayValue\\\":\\\"0.6\\\",\\\"value\\\":\\\"0.6\\\"}]\",\"created_at\
↪":1593638390,\"updated_at\":1595539498,\"root_app_scope_id\":\
↪"5efcfdf5497d4f474f1707c2\"}}"
}
```


The value in `alert_details` is itself an escaped JSON string whose content for the above alert can be seen below:


```
{
  "Sensor Id": "d89f926cddc7577553eb8954e492528433b2d08e",
  "Hostname": "collectorDatamover-1",
  "Process Id": 20196,
  "scope_id": "5efcfdf5497d4f474f1707c2",
  "forensic": {
    "Unseen Command": "true",
    "Unseen Command - Sensor Uptime (microseconds)": "34441125356",
    "Unseen Command - Parent Uptime (microseconds)": "35968418683",
    "Unseen Command - Parent Username": "root",
    "Unseen Command - Parent Command Line": "svlogd -tt /local/logs/tetration/efe/ ",
    "Unseen Command - Parent Exec Path": "/sbin/svlogd",
    "Unseen Command - Anomaly - Score": "0",
    "Unseen Command - Anomaly - Similarity - Unique": "true",
    "Process Info - Command String": "gzip ",
```


(continues on next page)

(continued from previous page)


```
      "Process Info - Exec Path": "/bin/gzip"
    },
    "profile": {
      "id": "5efcfdf6497d4f474f1707e4",
      "name": "Tetration Profile",
      "created_at": 1593638390,
      "updated_at": 1593638390,
      "root_app_scope_id": "5efcfdf5497d4f474f1707c2"
    },
    "rule": {
      "id": "5efcfdf6497d4f474f1707d6",
      "name": "Tetration - Anomalous Unseen Command",
      "clause_chips": "[{\"type\":\"filter\",\"facet\":{\"field\":\"event_type\",\
↪"title\":\"Event type\",\"type\":\"STRING\"},\"operator\":{\"label\":\"=\",\"type\
↪":\"eq\"},\"displayValue\":\"Unseen Command\",\"value\":\"Unseen Command\"},{\"type\
↪":\"filter\",\"facet\":{\"field\":\"forensic_event__cmd_not_seen_data__cmdline_
↪anomaly_info__score\",\"title\":\"Unseen Command - Anomaly - Score\",\"type\":\
↪"NUMBER\"},\"operator\":{\"label\":\"<\",\"type\":\"lt\"},\"displayValue\":\"0.6\",\
↪"value\":\"0.6\"}]",
      "created_at": 1593638390,
      "updated_at": 1595539498,
      "root_app_scope_id": "5efcfdf5497d4f474f1707c2"
    }
}
```


The details of the forensic events are included in the field `forensic`. For the list of attributes of the forensic events, please see *Forensic event fields*. These attributes are also shown in the alert details in the UI.

7.9 Forensics score

7.9.1 Where to see forensic score

- **Security Dashboard**:

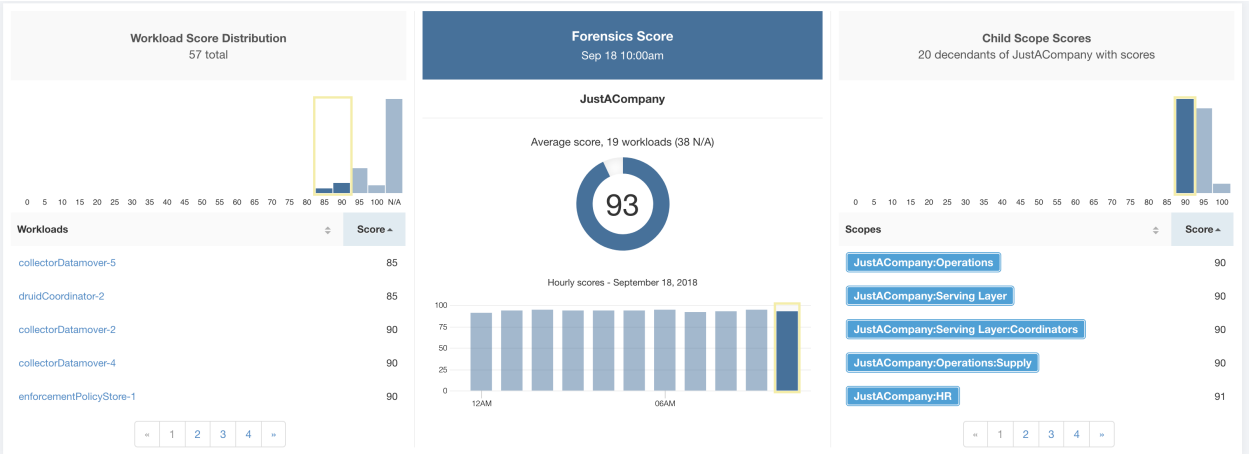


Fig. 7.9.1.1: Forensics Score section in Security Dashboard

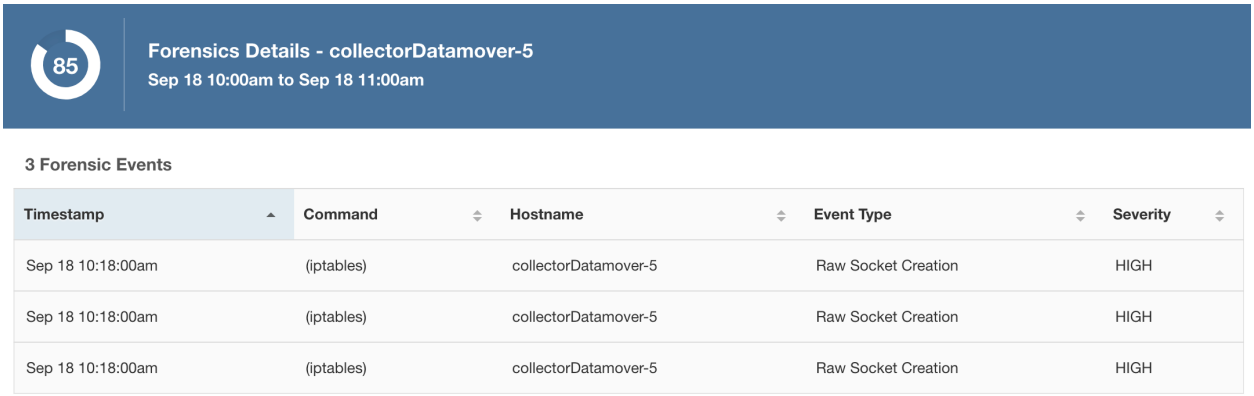


Fig. 7.9.1.2: Forensics Score Details section in Security Dashboard

7.9.2 How the forensic score is calculated

For each Workload we compute a Forensics Score. A Workload's Forensics Score is derived from the Forensic Events observed on that Workload based on the profiles enabled for this scope. A score of `100` means no Forensic Events were observed via configured rules in enabled profiles, and a score of `0` means there is a Forensic Event detected that requires immediate action. The Forensics Score for a Scope is the average Workload score within that Scope. Forensics Score for a given hour is a minimum of all scores within that hour.

- A Forensic Event with the severity `REQUIRES IMMEDIATE ACTION` will reduce the Score for the entire Scope to zero.
- A Forensic Event with the severity `CRITICAL` reduces a workload's score with the weight of `10`.
- A Forensic Event with the severity `HIGH` reduces a workload's score with the weight of `5`.
- A Forensic Event with the severity `MEDIUM` reduces a workload's score with the weight of `3`.
- A Forensic Event with the severity `LOW` doesn't contribute to the Forensics Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.

For example, a workload has 3 forensic events that match 2 rules with *CRITICAL* severity, 1 rule with *HIGH* severity , 1 rule with *LOW*, respectively. The forensic score for that workload is: `100 - 1*10 - 1*5 - 1*0 = 85`.

The Forensics Scores are `N/A` for workloads in which Forensics feature is not enabled.

7.9.3 How to improve forensic score

Tuning your Forensics Score can be done by adjusting the Forensic Rules enabled. Creating rules that are less noisy will give you a more accurate score. Acting upon and preventing legitimate Forensic Events (events that are evidence of an intrusion or other bad activity) is another good way to improve your Forensics Score.

7.9.4 Caveats

- Forensics Score details show **all** forensic events within that hour. That means Forensic Score details may show forensic events other than the ones used for computing forensic score.
- Forensics Score is currently available for `Deep Visibility` and `Enforcement` sensors.

7.10 PCR-based Network Anomaly detection

Network Anomaly feature detects abnormally large amounts of data flowing into or out of the workloads based on the concept of Producer Consumer Ratio (PCR). The PCR is defined as

```
      Egress app byte count - Ingress app byte count
PCR = ----------------------------------------------
      Egress app byte count + Ingress app byte count
```

The value of PCR is in the [-1.0, 1.0] range where

- PCR = 1.0 means the workload purely sends data out
- PCR = -1.0 means the workload purely receives data
- PCR = 0.0 means the workload has balanced amounts of data in and data out

Similar to other Forensics features, you can use the intent-based configuration to configure the Network Anomaly events you want to record and/or alert. Detected Network Anomaly events from workloads are exported every 5 minutes and are matched against configured rules 5 minutes later. As a result, new Network Anomaly events are only observed on the UI every 5 minutes with a delay of up to 10 minutes from the time of the event.

Note: In 3.2 and 3.1 versions of Tetration software, Network Anomaly detection was known as Data Leak detection.

7.10.1 Forensic rules for Network Anomaly events

Please refer to *Forensic configuration* on how to add forensic rules.

7.10.1.1 Rule attributes

This section explains the details of the attributes to define a Network Anomaly related rule. The simplest Network Anomaly rule is

Event Type = Network Anomaly

Below are other attributes in the Network Anomaly event to refine the rules for your data centers.

Attribute	Description
Host Name	The host name of the workload emitting this event.
Timestamp (epoch milliseconds)	The timestamp (in milliseconds) of the event.
PCR Deviation	The deviation of PCR from the mean at the event time as a multpile of historical standard deviation.
Non-seasonal Deviation	This is the PCR deviation after removing the seasonality pattern (e.g. by cron-jobs). The value of Non-seasonal Deviation is always larger than or equal to `6.0`.
PCR	The Producer Consumer Ratio.
EIR	The Egress Ingress Ratio, which is the ratio between the total Egress App Byte Count and the Ingress App Byte Count.
Egress App Byte Count	The egress application byte count, which is the total byte count of packet contents (excluding headers) flowing out of the workload.
Ingress App Byte Count	The ingress application byte count, which is the total byte count of packet contents (excluding headers) flowing into the workload.
Protocol	The protocol for which the PCR time series is calculated. Currently, the supported protocols are TCP, UDP, and Aggregate. Aggregate PCR is caluclated based on the total sum of TCP, UDP and ICMP byte counts.
User Logon Count	The number of user logon events on the workload within approximately the last 15 minutes. **Note**: this is the count of the User Logon events regardless of whether or not there are matched rules. In order to know the details of the User Logon events, you need to define rules to record the events for workloads of interests and view them in Forensics Analysis page.
User Logon Failed Count	The number of user logon failed events on the workload within approximately the last 15 minutes. **Note**: this is the count of the User Logon failed events regardless of whether or not there are matched rules. In order to know the details of the User Logon Failed events, you need to define rules to record the events for workloads of interests and view them in Forensics Analysis page.
Unseen Command Count	The number of unseen command events on the workload within approximately the last 15 minutes. **Note**: this is the count of the Unseen Command events regardless of whether or not there are matched rules. In order to know the details of the Useen Command events, you need to define rules to record the events for workloads of interests and view them in Forensics Analysis page.
Date Time (UTC) - Year	The year of the event time.
Date Time (UTC) - Month	The month of the event time (1, 2, . . .).
Date Time (UTC) - Day	The day of month of the event time (1, 2, . . .).
Date Time (UTC) - Hour	The hour of day of the event time (1, 2, . . . , 24).
Date Time (UTC) - Minute	The minute of hour of the event time (1, 2, . . . , 60).
Date Time (UTC) - Second	The second of minute of the event time (1, 2, . . . , 60).
Date Time (UTC) - Day of Week	The day of week of the event time (0 to 7, for Monday to Sunday).

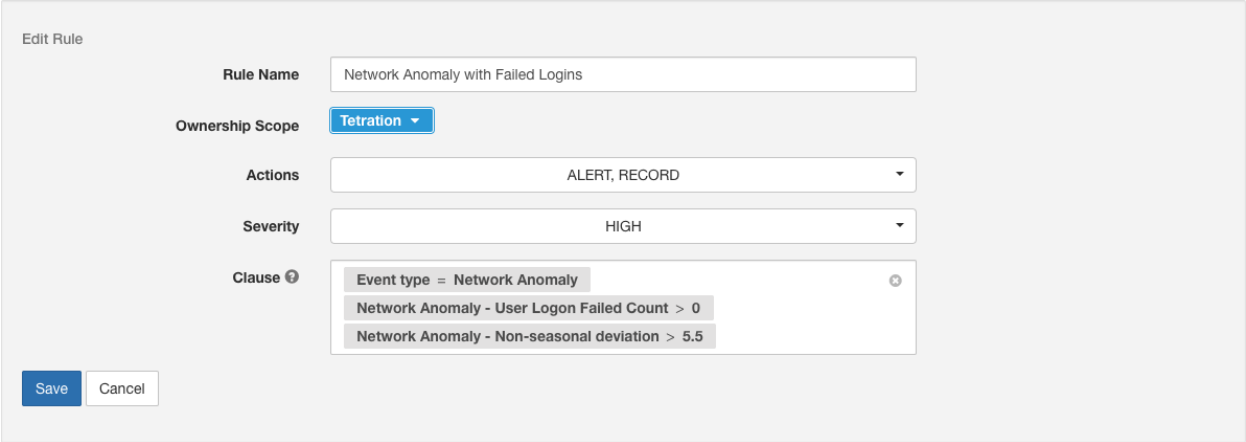


Fig. 7.10.1.1.1: Defining forensic rules for Network Anomaly events

Below are some sample rules:

Listing 7.10.1.1.1: Detects netwrok anomalies for UDP only.

```
Event Type = Network Anomaly AND Network Anomaly Is = Protocol - UDP
```

Listing 7.10.1.1.2: Detects very large deviation after removing seasonal pattern (if detected), with a threshold on the egress app byte count for a subset of workloads whose names contain *sensitiveDataServer*.

```
Event Type = Network Anomaly AND Network Anomaly - Non-seasonal Deviation > 10.0)
AND Network Anomaly - Egress App Byte Count > 1000000
AND Network Anomaly - Host Name CONTAINS sensitiveDataServer
```

Listing 7.10.1.1.3: Detects Network Anomaly events on workloads with unseen command events except the Network Anomaly events happen from 7.30AM UTC to 7.35AM UTC everyday.

```
Event Type = Network Anomaly AND Network Anomaly - Unseen Command Count > 0
AND ( Network Anomaly - Date Time (UTC) - Hour != 7
OR Network Anomaly - Date Time (UTC) - Minute < 30 OR Network Anomaly - Date Time␣
↪(UTC) - Minute > 35 )
```

7.10.1.2 Rule actions

Action	Description
RECORD	The matched events will contribute to the Network Anomaly Score and can be found via the Security Dashboard or the *Workload Profile Page / Network Anomaly Tab*.
ALERT	The matched events will show up in the *Alerts Page* and the chosen *Alert Publishers*.

The next section describes in more detail where to find detected Network Anomaly events in the UI.

7.10.2 Where to see Network Anomaly events

Note: Network Anomaly events are *not* currently shown in Forensics Analysis page. You can find Network Anomaly events in the following pages.

- **Security Dashboard**: Network Anomaly events that match rules with **RECORD** action can be found in the Network Anomaly score section in the Security Dashboard. If there are workloads with non-best (less than 100) scores, clicking on the workload name, you will be able to view the PCR time series and the Network Anomaly events on that workload. On the very right hand side of each row of the Network Anomaly event table, you can see action links that can help you search for flows and other forensic events around the time of the corresponding Network Anomaly event. See *Network Anomaly latency* for known delay in Network Anomaly score reporting.

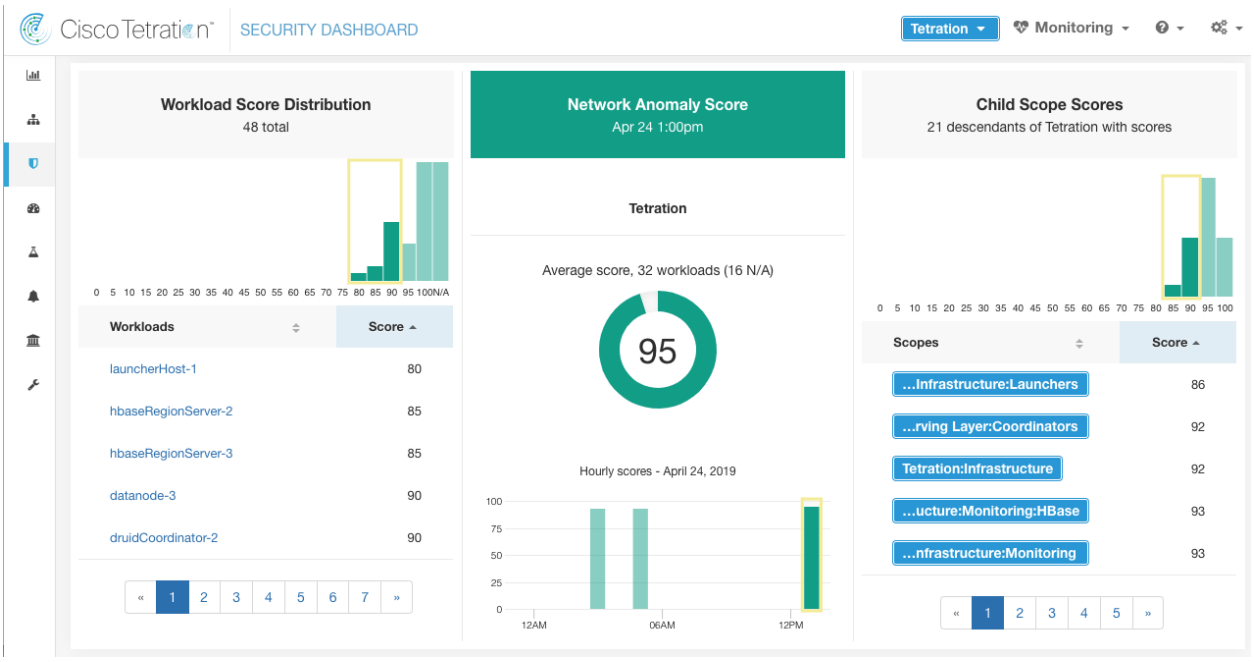


Fig. 7.10.2.1: Network Anomaly score in Security Dashboard

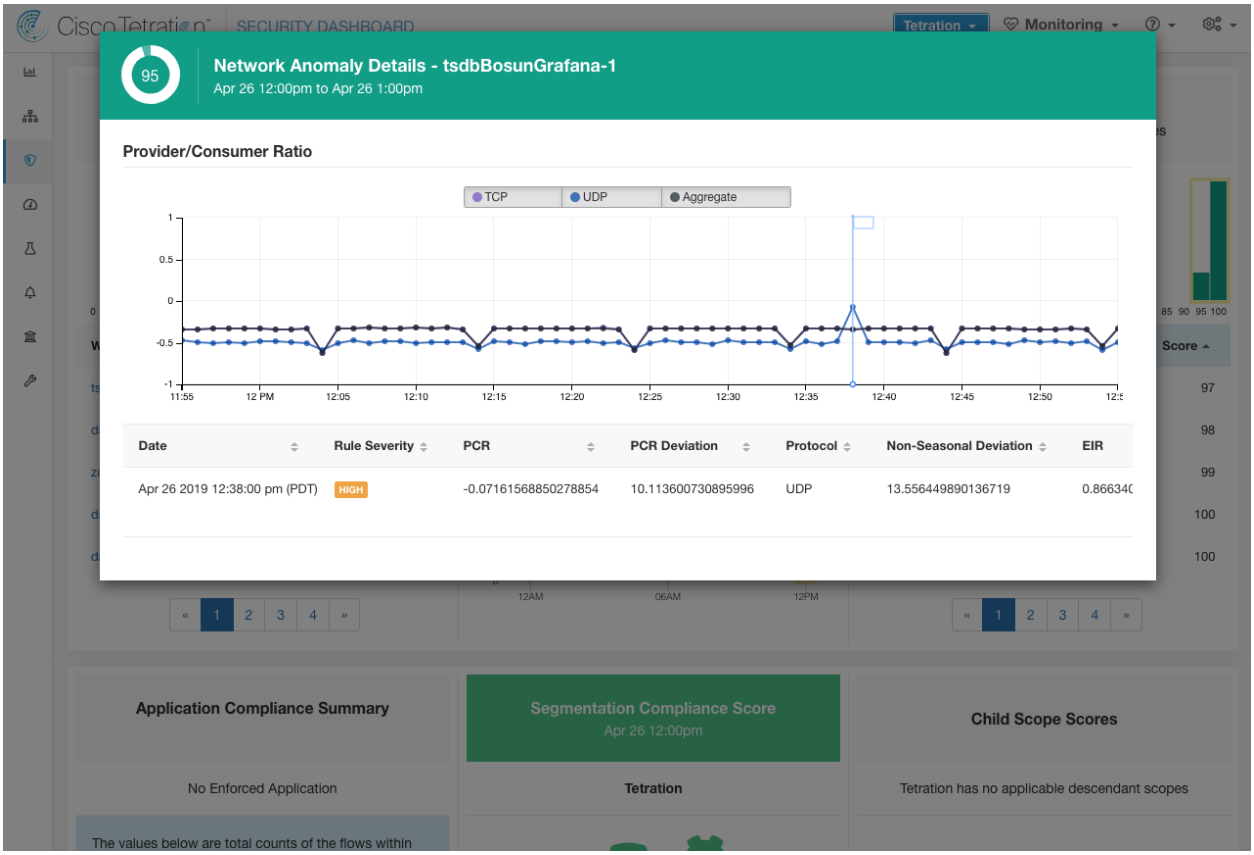


Fig. 7.10.2.2: Network Anomaly score in Security Dashboard drilled-down by workload

- *Workload Profile Page / Network Anomaly Tab*: on this page, you can see the PCR time series graph and the Network Anomaly events that match rules with **RECORD** action. What you can see on this page is very similar to what you find by clicking on the workload name in the security dashboard.

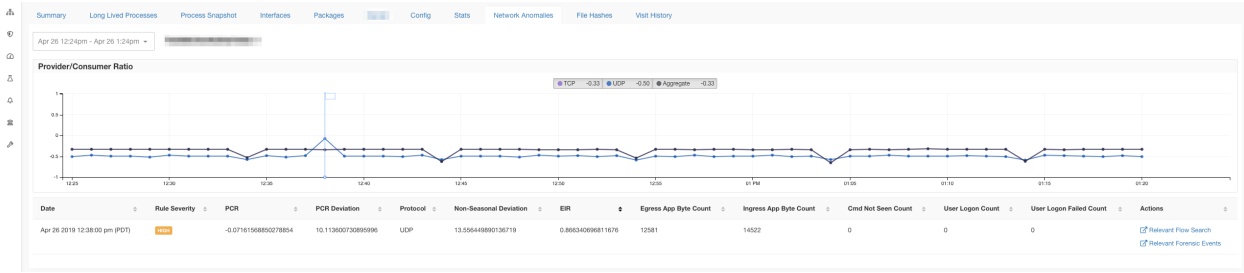


Fig. 7.10.2.3: Network Anomaly Tab in *Workload Profile Page*

- **Alerts**: If the Network Anomaly rule is configured with **ALERT** action, the matched events will be shown on the *Alerts Page* and are also available via Alert Publisher.

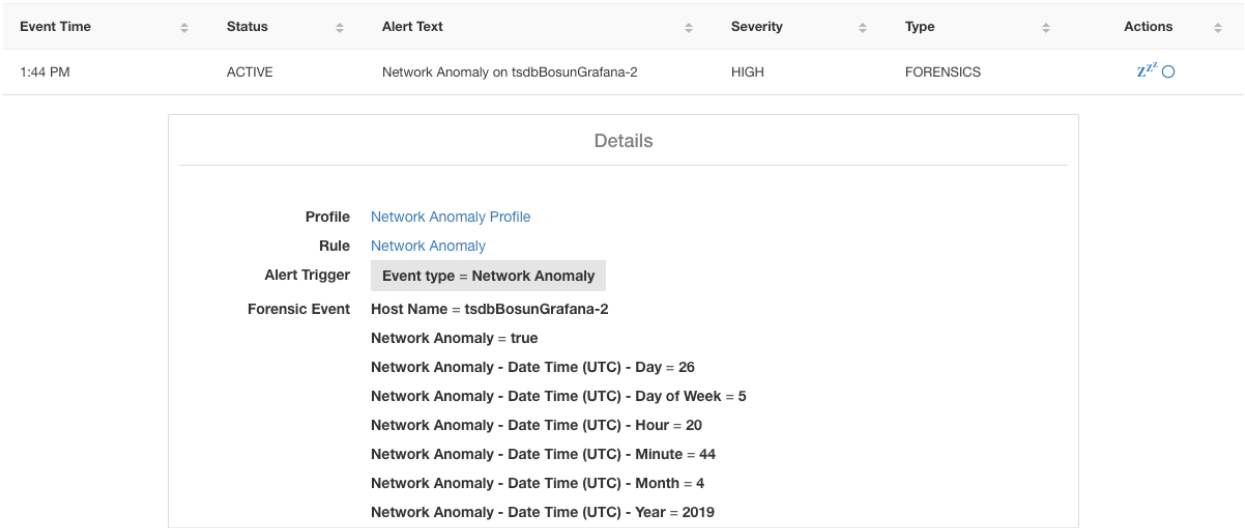


Fig. 7.10.2.4: Network Anomaly Alert

7.10.3 Rule severities and Network Anomaly scores

The Network Anomaly Score is computed similarly to the Forensics Score. For each Workload we compute a Network Anomaly Score. The Network Anomaly Score of a Workload is derived from the Network Anomaly Events observed on that Workload based on the profiles enabled for this scope. A score of 100 means no Network Anomaly Events were observed via configured rules in enabled profiles. A score of 0 means there is a Network Anomaly Event detected that requires immediate action.

- A Network Anomaly Event with the severity `REQUIRES IMMEDIATE ACTION` reduces the Score for the entire Scope to `0`.
- A Network Anomaly Event with the severity `CRITICAL` reduces a workload's score with the impact of `10`.
- A Network Anomaly Event with the severity `HIGH` reduces a workload's score with the impact of `5`.
- A Network Anomaly Event with the severity `MEDIUM` reduces a workload's score with the impact of `3`.
- A Network Anomaly Event with the severity `LOW` doesn't contribute to the Network Anomaly Score. This is recommended for new rules where the quality of the signal is still being tuned and is likely to be noisy.

For each workload, the total impact score is aggregated every 5 minutes to compute the score of that workload within those 5 minutes.

For workloads without Network Anomaly enabled sensor types, the Network Anomaly scores are `N/A`.

7.10.4 PCR data and Network Anomaly events retention

PCR data and Network Anomaly events are kept for 7 days.

7.10.5 Network Anomaly latency

- Network Anomaly scores reported in the security dashboard have 5-minute delays. For instance, the score of a workload for the hour 10:00am-10:59am is based on Network Anomaly events happen from 9:55am to 10:54am.

7.10.6 Caveats

- Old `Data Leak events` remain as `Data Leak events` instead of `Network Anomaly events`.
- Network Anomaly detection per protocol is a new feature in 3.3 and protocol is not set in old `Data Leak events`.

7.11 Process hash anomaly detection

As the name suggested, this feature detects process hash anomaly by assessing the consistency of process binary hashes across the system. The motivation of this feature is as follows. Imagine that you have a farm of Apache web servers that are cloned from the same setup configuration (e.g., those servers are deployed from the same automation scripts). Then you would expect that the hashes of httpd binaries on all servers are the same. If there is a mismatch, it is an anomaly and might worth a further investigation.

Formally, we define *process group* as the set of processes across workloads in the same rootscope that have the same combination of executable binary path, OS version, and package info (if applicable)[1]. In the example above, suppose that all Apache web servers are running httpd 2.4.43 on CentOS 7.7 and in the same rootscope, then the corresponding process group is the set of processes (across all servers) that have the same combination: binary path of `/usr/sbin/httpd` & OS version of `CentOS-7.7` & package version of `httpd-2.4.43`. It is expected that the hashes of all binaries in the same process group are the same, and an anomaly will appear if any mismatch is detected.

Besides detecting anomalous process hashes, this feature also detects process hashes that appear in a Flagged list *uploaded* by user. The motivation is that you may have a list of known malware hashes, and would like to know if a process associated with any of those hashes is run.

To reduce false alarms, we use the National Software Reference Library's Reference Data Set (RDS) provided by NIST (we also call it NIST RDS dataset) as a Benign list; a benign hash is considered "safe" (see *Threat Intelligence* on how to enable NIST RDS dataset). You can also *upload* your own hash Benign list.

In addition to the NIST RDS dataset, we also curate **Tetration Hash Verdict** service. When this service is enabled, if any known malware hash shows up, it will be detected as malicious hash. On the other hand, if the hash is known and legit, then it is also marked as benign in the anomaly analysis. Due to the extremely large dataset and fast updates that covers all known and legit process hashes that can be used to either approve or red flag processes running on a workload, Tetration Hash Verdict is only available via Tetration Cloud. Please refer to *Automatic Threat Intelligence Updates* to ensure Tetration Hash Verdict service is accessible from your appliance.

Output of this feature is a security score called **process hash score**. This score is calculated and output hourly. Like all other security scores, a higher process hash score is better. In particular, for a process hash:

- Hash score of `0` means that the hash is flagged or malicious
- Hash score of `100` means that the hash is either benign, or consistent across workloads (no mismatch)
- Hash score from `1` to `99` means that the hash is considered anomalous (i.e., there is some mismatch)

The process hash score of an workload is the minimum process hash score of all hashes observed in that workload, with `0` meaning there is a flagged or malicious process hash in the system, and `100` meaning there is no hash anomaly observed in the system.

[1] Package info is included since 3.4 release; in the previous releases, the process group is defined based on the combination of executable binary path and OS version only.

7.11.1 How to enable process hash feature

Process hash feature is enabled by default on deep visibility agents and enforcement agents; no forensic config is needed. If there are such agents in your system, you should begin to see scores within 2 hours after the system starts.

7.11.2 Where to see process hash score

- **Security Dashboard**:

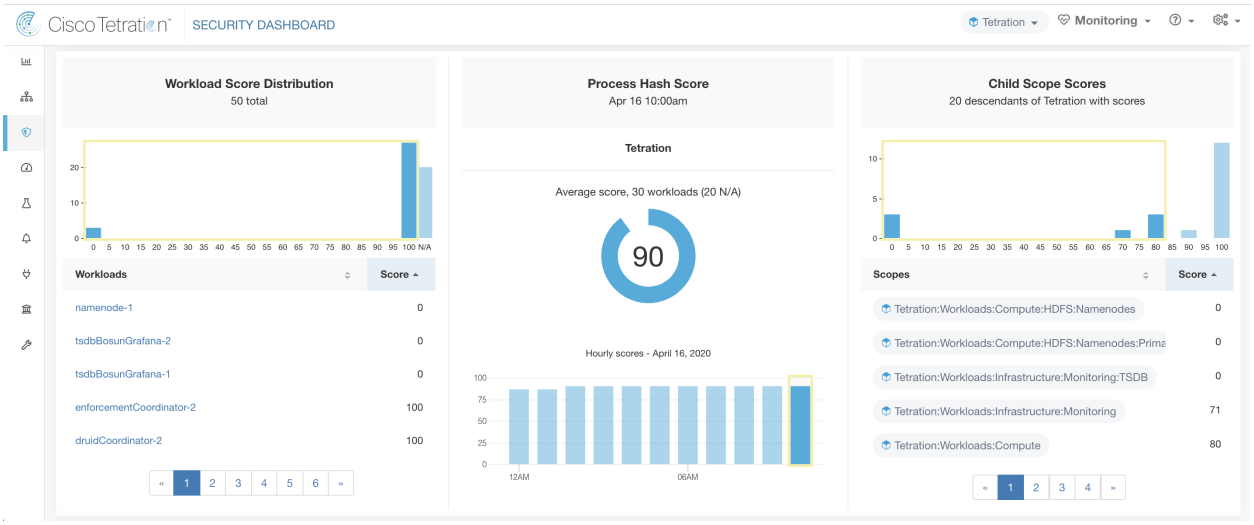


Fig. 7.11.2.1: Process Hash Score section in *Security Dashboard*

- **Workload Profile Page** / *File Hashes Tab*:

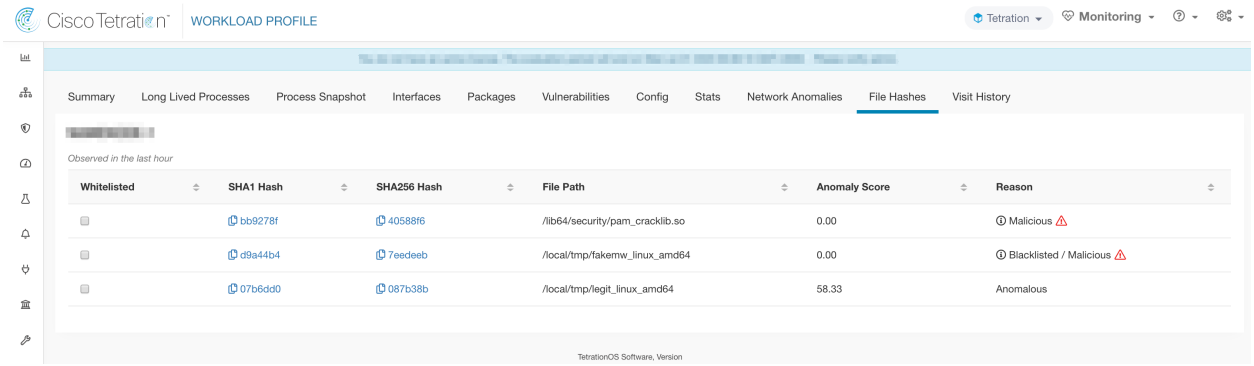


Fig. 7.11.2.2: File Hashes tab in *Workload Profile page*

7.11.3 How the process hash score is calculated

For each process hash we compute a score as follows:

1. If hash is flagged or malicious: `score = 0`
2. Else, if hash is benign: `score = 100`
3. Else, if hash is an anomaly: `score` is in the range of `[1, 99]`, the higher the better

4. Else: `score = 100`

The logic for calculating score in (3) is that we first calculate the minority score of the hash (which is one minus the population ratio of that hash in workload population under the same rootscope), then map it to range `[0.0, 1.0]` using information function `-log2(x)` if minority score of the hash is above `0.5`, then map the score again to range `[1.0, 99.0]`. Let us take the above example of Apache web server farm and consider the hash of `httpd`. Below are some scenarios:

- Suppose that `httpd` has two hash values (`h1` and `h2`) across 1000 servers in the farm: `h1` in 1 server, `h2` in the rest 999 servers. In this case:
  - `population_ratio(h1) = 0.001`, `population_ratio(h2) = 0.999`. Then:
  - `minority_score(h1) = 0.999`, `minority_score(h2) = 0.001`. Then:
  - `score(h1) = -log2(0.999) * 98 + 1 = 1.14`;
  - since `minority_score(h2) < 0.5`, `h2` is not considered an anomaly, hence `score(h2) = 100`.
- Suppose that `httpd` has two hash values (`h1` and `h2`) across 10 servers in the farm: `h1` in 1 server, `h2` in the rest 9 servers. In this case:
  - `population_ratio(h1) = 0.1`, `population_ratio(h2) = 0.9`. Then:
  - `minority_score(h1) = 0.9`, `minority_score(h2) = 0.1`. Then:
  - `score(h1) = -log2(0.9) * 98 + 1 = 15.90`;
  - since `minority_score(h2) < 0.5`, `h2` is not considered an anomaly, hence `score(h2) = 100`.
- Suppose that `httpd` has two hash values (`h1` and `h2`) across 2 servers in the farm: `h1` in one server, `h2` in the other. In this case:
  - `population_ratio(h1) = population_ratio(h2) = 0.5`. Then:
  - `minority_score(h1) = minority_score(h2) = 0.5`. Then:
  - `score(h1) = score(h2) = -log2(0.5) * 98 + 1 = 99.0`. This is the highest score for any hash that is considered an anomaly.
- Suppose that `httpd` has only one hash value (`h1`) across all servers. In this case, `minority_score(h1) = 0.0 < 0.5`; hence it is not considered an anomaly, and `score(h1) = 100`.

Finally, the process hash score of an workload is the minimum process hash score of all hashes observed in that workload.

Additional information about the `-log2(x)` information function can be found here.

7.11.4 How to improve process hash score

The process hash score of `0` on a workload means that a flagged or malicious process hash has shown up in that workload; preventing such a process to run again will improve the score. A positive process hash score less than `100` means that there is a process hash anomaly across your system; it may or may not be malicious but worth a further investigation. After a careful investigation, if the hash is concluded to be safe, adding it to your Benign list will also improve the score. User can mark anomalous hashes as 'benign' by clicking on the Benign checkbox in the File Hashes / Process Hash Details page or by *uploading a Benign list via OpenAPI*.

7.11.5 Threat info details

As mentioned earlier, if Tetration Hash Verdict service is enabled, any known malware hash when showing up would be flagged as malicious. In that case, additional threat information of the malicious hash (gathered via our threat

intelligence platform) will be provided. Currently the additional threat data include *threat name* and *severity*. Threat name is the name of the threat, while everity is a value from 1 to 5 to indicate how severe the threat is, where 1 means the least and 5 means the most severe.

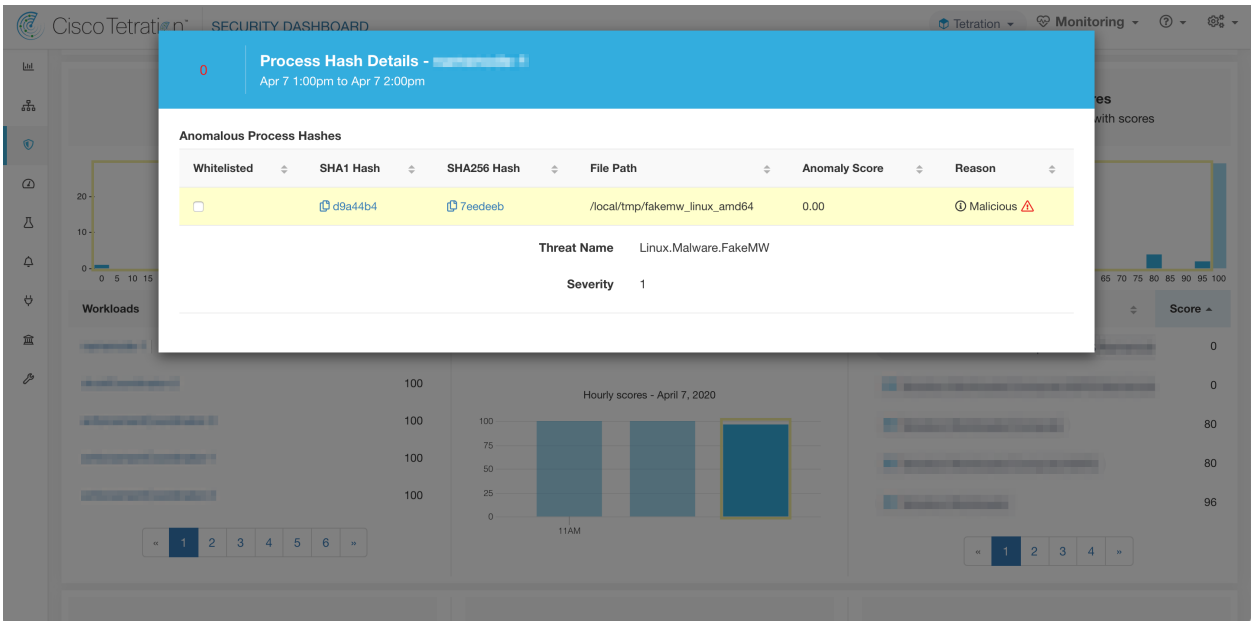


Fig. 7.11.5.1: User can click on the row of malicious hash to view its threat info details

7.11.6 Caveats

- Process hash analysis task is run once per hour, but it may take up to 2 hours for the expected scores/results to show in the security dashboard depending on the action. For examples:
  - If you upload your hash Flagged list and a process hash in that list shows up, it may take up to 1 hour for the score to be reflected in the security dashboard.
  - If you remove a hash from your Flagged list, it may take up to 2 hours for it to be completely cleared (and score to be reflected) in the security dashboard.
- Retention:
  - Detailed results from process hash analysis are kept for at least 7 days
- File Hashes tab in Workload Profile page only shows process hash details analyzed in the last hour
- Previous versions of deep visibility and enforcement agents, as well as AnyConnect endpoints only report SHA256 hash values. Thus, matching against SHA1 hash Flagged/Benign list is not supported for those agents.
- Process hash score is calculated with respect to a particular rootscope. If a workload belongs to multiple rootscopes, the process hash score of that workload is the minimum score across all rootscopes that it belongs to.
- To further reduce the false alarms in process hash anomaly analysis, we also mark all Tetration agent binaries as benign according to their file paths. This mechanism happens only when these hashes do not appear in any user-defined hash list, or are not flagged by Tetration Hash Verdict service.

CHAPTER

EIGHT

PERFORMANCE MONITORING

Warning: ACI Network Fabric Monitoring feature-set is disabled by default in Release 3.3.

The **Performance Monitoring** feature set enables monitoring and root-causing network and application performance issues by providing detailed time-series on fabric link metrics such as latency, drops and throughput. Furthermore, we provide the ability to map fabric topology and trace individual flows across the fabric. Furthermore, we can lookup all flows going over a specific switch port and egress queue. The above mentioned features are available via the **Fabric** page.

Other performance related metrics are available via the flow search page, where you can filter flows that are application or network limited or experiencing performance issues due to too many TCP retransmissions.

8.1 Feature Matrix

The network performance features are supported based on the sensors deployed in the network. This section provides details on the sensors required for a feature to work, and their limitations. Broadly,

- Fabric visibility and performance requires hardware sensors using Cisco Nexus 9300-FX or 9300-FX2 switches as leaf and Cisco Nexus 9300-FX line cards in spine running in Cisco ACI mode.
  - Physical topology discovery
  - Map flows to topology at a per-link + per-queue granularity
  - Hop by hop per-flow latency breakdown
  - Per-class per-flow buffer drop indicators
  - Per-link per-class aggregate counters, such as bandwidth, drops, latencies
  - Per-link aggregate counters, such as bandwidth, drops, latencies
- TCP performance requires *deep* software sensors on both ends of a flow.
  - Time series of flow classification into network and application-limited
  - Flow search for network limited and application-limited flows
  - Flow search for network events, such as MSS change, Congestion-Window reduction, and zero-window advertisement, retransmissions
  - Flow search for long-handshake flows

8.1.1 Hardware Sensor Supported Features

Switch configurations

- Cisco ACI single pod
- Cisco ACI multi pod
- Cisco ACI multi fabric
- Cisco NXOS mode (Standalone)

Cisco N9000 series Switches with Tetration sensors

- **EX LEAF - Cisco Nexus 9300-EX**
  - Cisco Nexus 93180YC-EX
  - Cisco Nexus 93108TC-EX
  - Cisco Nexus 93180LC-EX
- **FX LEAF - Cisco Nexus 9300-FX**
  - Cisco Nexus 93180YC-FX
  - Cisco Nexus 93108TC-FX
  - Cisco Nexus 9348GC-FX
- **FX2 LEAF - Cisco Nexus 9300-FX2**
  - Cisco Nexus C9336C-FX2
- **FX SPINE – Cisco Nexus 9500 series with FX line cards (Supported only in Cisco ACI mode)**
  - Cisco Nexus X9736C-FX line cards and appropriate fabric modules.

Currently, **Cisco ACI single pod** switch configuration with **FX SPINE** and **FX/FX2 LEAF** is required to support fabric topology and latency features. The Cisco ACI software release that supports these features is 3.1(x).

Both Tetration platforms, 39RU and 8RU, support the fabric features, although the scale limits are different for both, and are provided in the following table:

8.1.1.1 Scale Limits

Parameter	39RU	8RU
Total flow feature rate[1]	Up to 2 Million per second	Up to 500,000 per second
Number of switches (hardware sensors)[2]	100	100
Full visibility Flow rate per FX/FX2 LEAF[3]	See note[6]	See note[6]
Full visibility Flow rate per EX LEAF[4]	See note[6]	See note[6]
Full visibility Flow rate per FX SPINE	See note[6]	See note[6]
Theoretical Max Flow rate per switch for FX/FX2	32,000	32,000
Theoretical Max Flow rate per switch for EX	64,000	64,000
Theoretical Max Flow rate for FX SPINE switch	See note[5]	See note[5]

[1] All rates are in per second. The flow rate is measured in unidirectional flow events reported by each switch ASIC or software sensor in an export interval. A range is provided because the actual flow event (aka flow feature) rate depends on multiple factors like – **A)** The ratio of the mix of short lived flows and long lived flows in the network under observation. Short lived flows (eg DNS exchanges - which could be 1 unidirectional flow per packet) add more load on the system per flow event (aka flow feature), where as longer lived flows have a lower load per flow event (aka flow feature). – **B)** The number of security and performance features enabled – **C)** Other ingested events like process events and user streamed data.

8.1.1.2 Supported Features with Switch software

The fabric performance features currently work only with the following setup:

- Cisco ACI single pod configuration
- Cisco ACI software release from 3.1 up to 3.2.4
- Cisco ACI release 3.2.4 onwards and up to 4.1 have known compatibility issues with Tetration (Defects CSCvo84244 and CSCvp40617), and are not recommended.
- For other Cisco ACI releases, please check ACI release documentation to confirm compatibility for Tetration features.
- Cisco Nexus 9300-FX or 9300-FX2 LEAF and Cisco Nexus 9500 SPINE with X9736C-FX line cards
- Tetration enabled on SPINE and LEAF switches on the fabric

Capability	Cisco NXOS (Standalone)	ACI 2.3 or 3.0	ACI 3.1 with FX	ACI 3.2 with FX2
Flow visibility[10]	supported	supported	supported	supported
Total Fabric Packet drop indications	supported	supported	supported	supported
IP subnet based collection rules	supported	supported	supported	supported
Burst monitoring	not supported	not supported	supported	supported
Fabric Topology Discovery	not supported	not supported	supported[11]	supported[11]
Spine Sensor	not supported	not supported	supported[12]	supported[12]
Flows mapped on Switch Topology	not supported	not supported	supported[13]	supported[13]
VRF based collection rules	not supported	not supported	supported[14]	supported[14]
Total Fabric Latency per flow	not supported	not supported	supported[15]	supported[15]
Per flow per-link Latency	not supported	not supported	supported[16]	supported[16]
Flow packet drops per port per class	not supported	not supported	supported[17]	supported[17]
Per link aggregated latency	not supported	not supported	supported[18]	supported[18]
Per port per class aggregated drops	not supported	not supported	supported[19]	supported[19]

– **D)** Number of flow collisions (aka table misses) seen by the switch ASICs.

[2] Total flow feature rate into the Tetration platform supersedes all other scale numbers. If switches are sending flows at a high rate, this will result in fewer hardware sensors being supported by the platform.

[3] Although each switch flow table has space for 32K or 64K flows, flow collisions can occur at a lower rate, thus resulting in some flows getting ignored in analytics. Moreover, the Tetration software uses more compute to deal with flows that encounter table full, which reduces overall supported flow rate.

[6] **Refer to Cisco ACI and NXOS release notes**.

[4] EX and FX2 LEAF switches have two flow tables, while FX switches have one.

[5] Flow export rate from SPINE depends on number of Line-cards and switch software. Each SPINE line card has 4 flow tables and each flow table can export up to 32K flows.

[10] Hardware sensors export the telemetry using UDP. If a connection closing information for a flow is not received by Tetration, then that flow remains open in the pipeline reducing the number of new flows that can be ingested. In certain worst case scenario flow capacity can reduce by as

8.1.1.3 Collection rules on the Switch

The Cisco Nexus 9300-EX switches do not support per VRF based collection rules, but Cisco Nexus 9300-FX and 9300-FX2 switches have the hardware capability to support this, availability of this feature is also tied to the Cisco NXOS and Cisco ACI software releases. For hybrid switch environment, where both EX and FX/FX2 switches are connected to Tetration cluster, the collection rules must be configured carefully.

Collection Rules	EX LEAF	FX/FX2 LEAF[21]	FX SPINE[22]
Global IP/subnet rules	supported	supported	not supported
VRF based rules	not supported	supported	not supported[23]
Number of IPv4 rules	64 subnets	64 subnets	64 subnets[24]
Number of IPv6 rules	16 subnets	16 subnets	16 subnets

8.2 Network Fabric

Warning: ACI Network Fabric Monitoring feature-set is disabled by default in Release 3.3.

Fabric page provides a visual tool to view and explore the network fabric topology at any given time and gain insights in terms of network performance metrics such as bandwidth, latency and drops.

The following figure provides an overview of the fabric page. On top lies the corpus selector which looks and behaves similarly to the *Corpus Selector* on flow search. Corpus selector allows you to specify an time range within the range of available data. This time range is used to fetch/show relevant topology and time-series information throughout the page.

The main graph shows the network fabric topology as observed at the **end** of the time range specified by the corpus selector. This timestamp of the network topology as well some high level statistics are displayed on the sidebar. The content of the sidebar changes based on interactions with network topology graph.

much as 50%.

[11] Topology discovery is based on Link Layer Discovery Protocol (LLDP). Connected switches and servers that are not part of the fabric, but support LLDP are also shown in the topology.

[12] FX line cards on SPINE provide latency, not EX line cards.

[13] Currently, Ethernet and VLAN traffic can be mapped.

[14] See the following table for collection rule limitations

[15] Flows must enter and exit the fabric on FX or FX2 LEAF

[16] All switches on the flow path must support latency, FX/FX2 LEAF, SPINE with FX line card. Note also that on SPINE, the latency is computed only at the egress line card, thus for LEAF1 -> SPINE -> LEAF2 flow, LEAF1 -> SPINE latency would be more than SPINE -> LEAF2 latency.

[17] Packet drop indications provide the number of second intervals, where any packet of a flow was dropped due to output port buffer full.

[18] Per link latency calculation requires latency reporting switches (FX/FX2 LEAF and/or FX SPINE) on both ends of the link.

[19] Drop indications are provided on output ports from supported switches.

[21] Per VRF based rules filtering requires Cisco ACI 3.1 release. This functionality is not available in Cisco NXOS in the current time.

[22] Requires ACI 3.1 switch build. FX SPINE is support is not yet available in Cisco NXOS mode

[23] The collection rules that do not fit in the switch are applied in Tetration cluster.

[24] As IPv4 and IPv6 rules reuse the same resource; For each IPv6 rule, reduce IPv4 capacity by 4x.

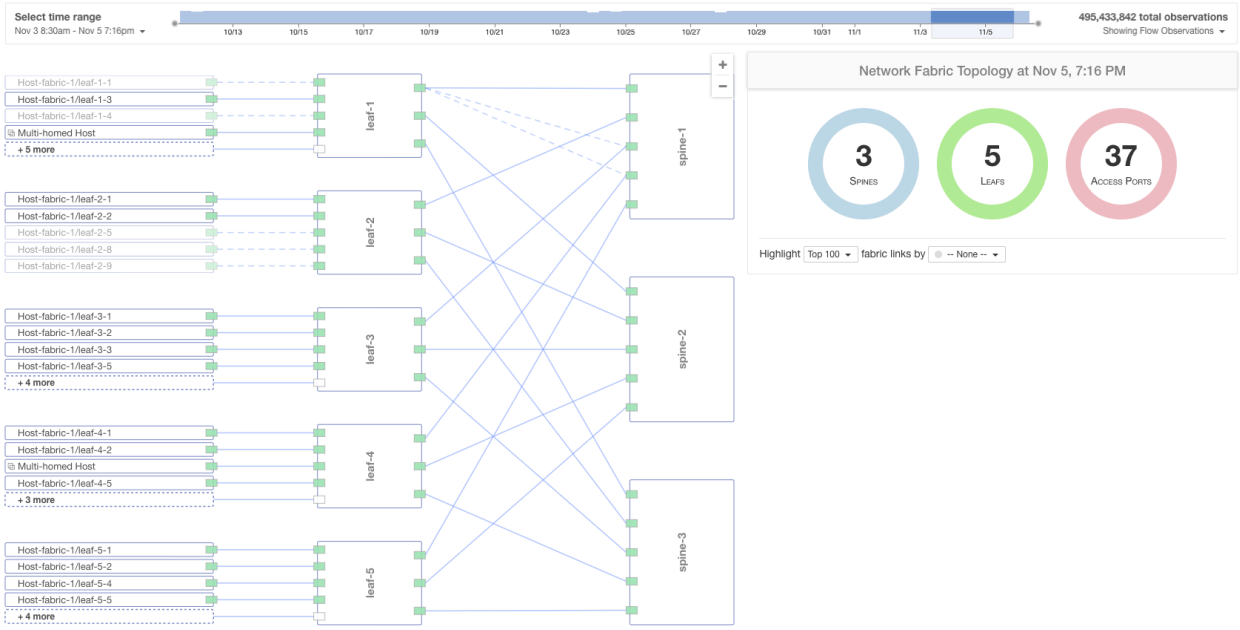


Fig. 8.2.1: Fabric page

8.2.1 Fabric Topology Graph

The fabric topology graph consists of three main tiers: The **spine** switches are displayed on the right; the **leaf** switches are displayed in the middle and hosts or other external devices are shown on the left of the graph. The total number of spine and leaf switches as well as the number of access ports on the leaf switches are displayed on the sidebar.

Note: Currently we only support standard spine and leaf network topologies. If there are additional non-standard links connecting leaf switches to each other, they will be ignored.

Depending on the zoom level and window size, not all switch ports or hosts may be visible. We show as many hosts as possible and group the rest under a single item indicating the number of endpoints not shown. Clicking on such group items expands them to reveal all the hidden endpoints. Alternatively, you can use the *Zoom In* button to gradually expand the graph and reveal more endpoints. Zooming out to the minimum zoom level again hides all the excess endpoints that do not quite fit in the window.

8.2.1.1 Multi-pod Fabrics

It is common for the network fabric span multiple pods across several geographic regions. Each pod consists of a number of leaf and/or spine switches connected to each other via an IP network (IPN) which is typically implemented by some WAN routers. We interpret the network devices connected to spine switches (that are not leaf switches) as IPN nodes. There is at most one IPN node per pod and all traffic across pods is routed via the IPN nodes. Such nodes are visually shown as a cloud icon (see figure below).

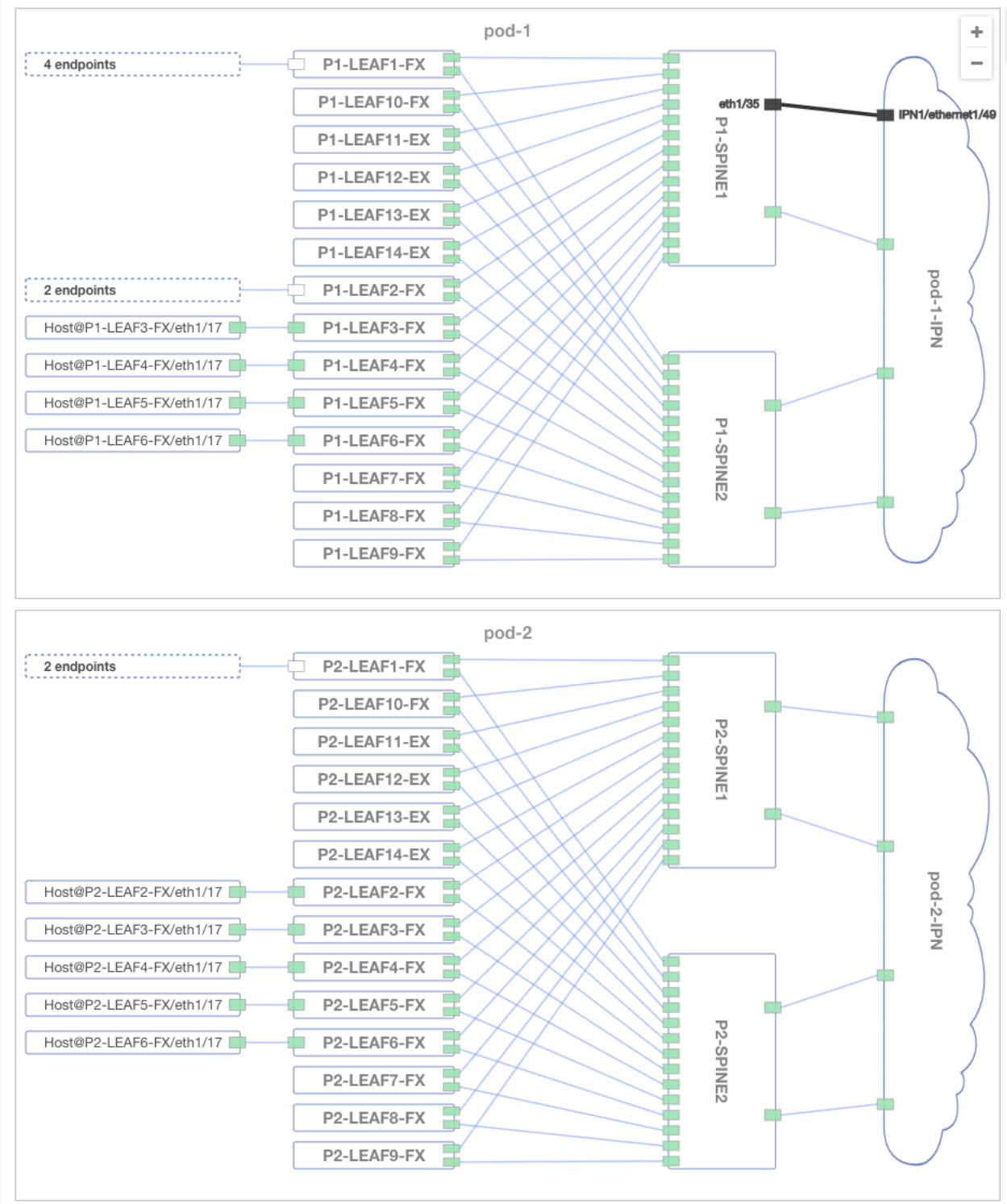


Fig. 8.2.1.1.1: Multi-pod fabric topology graph with IPN (IP Network) connectivity

The fabric and pod name (as reported by the switches) is shown at the top center of each pod. In the case of a single pod with an unknown name, the pod name and border will not be shown.

8.2.1.2 Fabrics with Remote Leaf Switches

Network fabrics spanning multiple locations may not require a full pod for certain smaller locations. In these scenarios a *remote leaf* switch is typically deployed in locations with smaller footprint. Remote leafs participate in one of the pod topologies and use a spine as the proxy for the rest of the fabric. The traffic between the remote leaf and the rest of the fabric always passes through the IPN nodes. Observe that the remote leafs are always shown as part of a generic "pod". This is so since in our current implementation we discover pod names through LLDP information and remote leafs do not have any LLDP neighbors to report on their pod name. Therefore, remote leafs from various pods will be shown under the generic pod with a single IPN node.

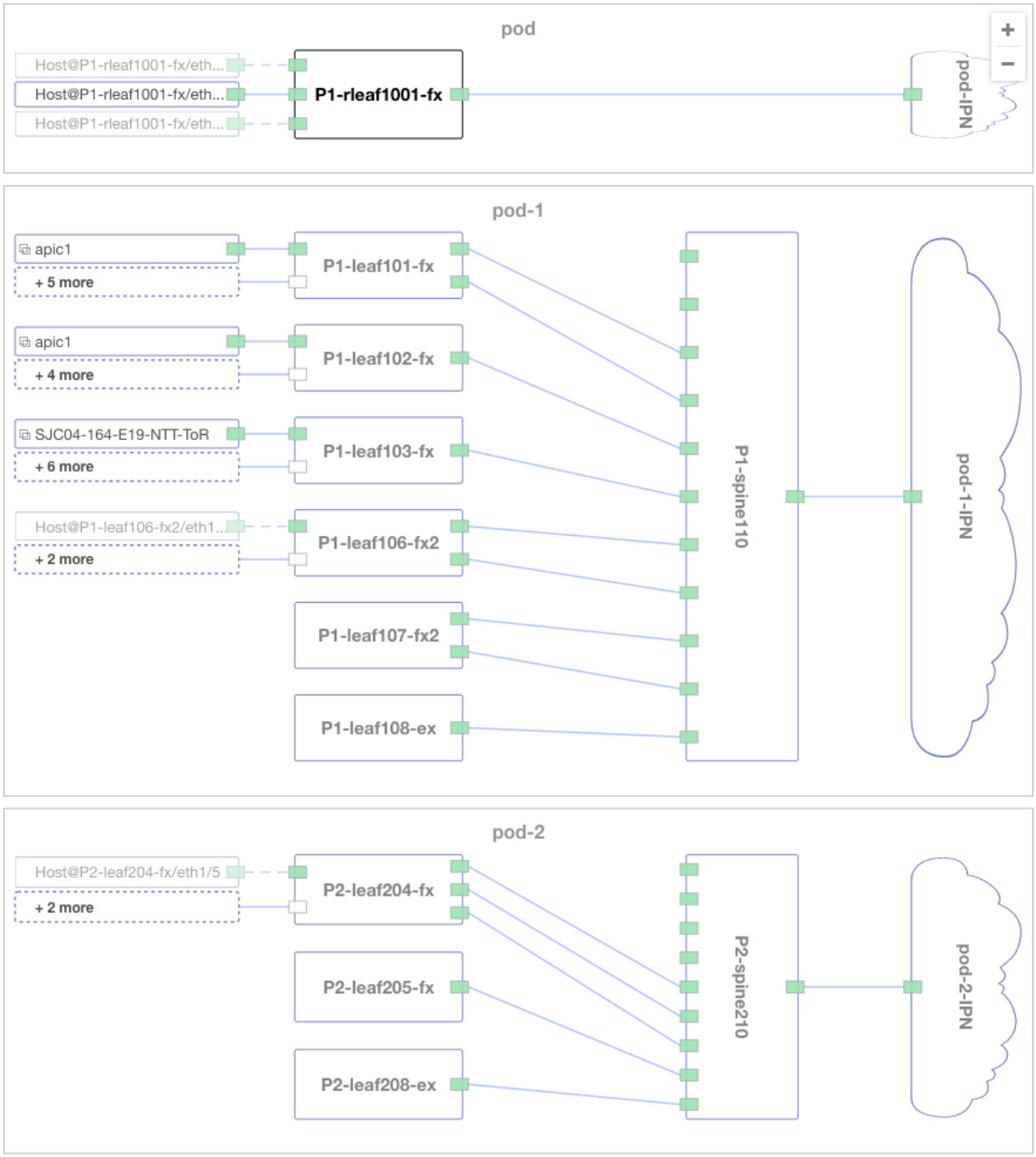


Fig. 8.2.1.2.1: Remote Leaf Switches

8.2.1.3 Multi-homed Hosts

Multi-homed hosts, that are hosts or external switches that are connected to leaf switches via more than one access port, are shown as separate items next to each corresponding access port. Each item is marked with a special Double Square icon. Hovering on each multi-homed host, highlights all items with the same name connecting to various access ports.

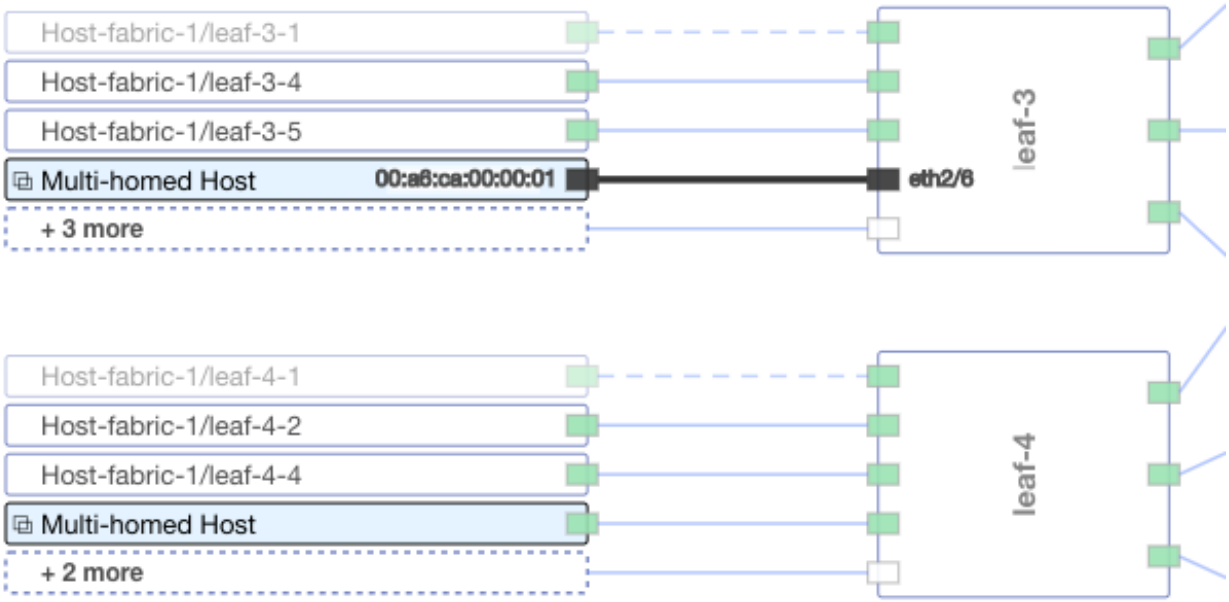


Fig. 8.2.1.3.1: Multi-homed Hosts

8.2.1.4 Inactive Links

In order to help keep track of changes in network topology over time, we show links and switches that are marked as *inactive* in addition to the current network topology. The inactive links are displayed as dashed lines and inactive switches and hosts are shown with lower opacity.

A link is marked inactive at a certain time, if it is not explicitly reported by LLDP and no flow observations are reported on that link on that time instance. A switch/host is marked as inactive if *all* of its connected links are marked as inactive. After an hour of inactivity, such links are removed from future instances of the network topology.

8.2.2 Fabric Link Information

Hovering on each port/link, reveals the port names (as reported by LLDP) on either side of the link. If the port name is not available, it is shown as *unknown*.

Clicking on any fabric link, reveals further information about that link on the sidebar. You can click anywhere on the chart background again to see the top level chart stats on the sidebar.

Note that each link shown on the topology graph represents two uni-directional connections. The **uplink** direction normally shown on the top represents the connection from a leaf to a spine, or from a host ot a leaf switch. We refer to the opposite direction as **downlink**.

There are up to three time-series chart shown for either direction of a fabric link:

- **Bandwidth**: The (layer 2) byte count reported on all flows passing through the link divided by the granularity interval. Note that the bandwidth metric represents the **average** bandwidth observed on a link, hence it may not show certain short term traffic bursts.
- **Average Latency**: The average per packet latency measured at the egress port of every switch. Note that flows with higher packet counts contribute a larger weight in average latency calculations, hence this metric represents the latency that each packet is experiencing on average.

- **Drop Indicators**: Every flow experiencing a packet drop in any one second interval counts as a single drop indicator. The chart shows the time-series of total drop indicators observed by all flows passing through a particular link in a given direction.
- **Burst Indicators**: Every flow experiencing a *burst event* in any one second interval counts as a single burst indicator. The chart shows the time-series of total burst indicators observed by all flows passing through a particular link in a given direction. For more information on the exact definition of the burst events please see *Burst detection*.
- **Burst+drop Indicators**: Every flow experiencing a packet drop in the same one second interval with burst event counts a single burst+drop indicator. The chart shows the time-series of total burst+drop indicators observed by all flows passing through a particular link in a given direction. See *Burst detection* for more info.

Fig. 8.2.2.1: Fabric Link Information

You can use the corpus selector to change the time range of any of the above time-series charts and all the relevant charts will get updated. The granularity of the time-series charts depends on the selected time range. Smaller time ranges result in finer granularity.

Note that changing the end time of the selected time range, loads a possibly different network topology. Therefore, a selected fabric link may not be valid at all times. Therefore, time-series data might not be available for all or part of

the selected time range.

Note: If all the latency or drop indicator values for a time-series chart is zero, or the metric is not available for the entirety of the selected time range, we do not show the chart for that metric.

Note: Latency information on fabric links connected to IPN links is not available. Currently fabric latency can only be measured for each flow from one switch to another, but we cannot break down the latency measurement to parts related to/from the IPN. The per flow fabric latency is still available under flow search, but we do not aggregate them for IPN links to avoid showing incorrect information.

Warning: Maximum measurable value for fabric latency is **6.8 milliseconds**. The fabric latency value of 6.8ms indicates that the actual average latency is greater than or equal that value.

8.2.2.1 Flow Search

Click on the *Find Flows* button on the top right of the fabric link info to navigate to Flow Search page and automatically filter flows passing through the selected link. If no flows are reported, make sure to check the selected time range and the selected scope on the page header.

8.2.2.2 Class

Before packets are sent out of an **egress** (outgoing) switch port, they are queued in one of the many egress queues and then served by priority. The class for each flow indicates the name/number of the egress queue through which the flow was served.

In order to view the time series information for flows of a particular class going through a fabric link, select one of the available classes from the drop-down on the top left corner of the fabric link info.

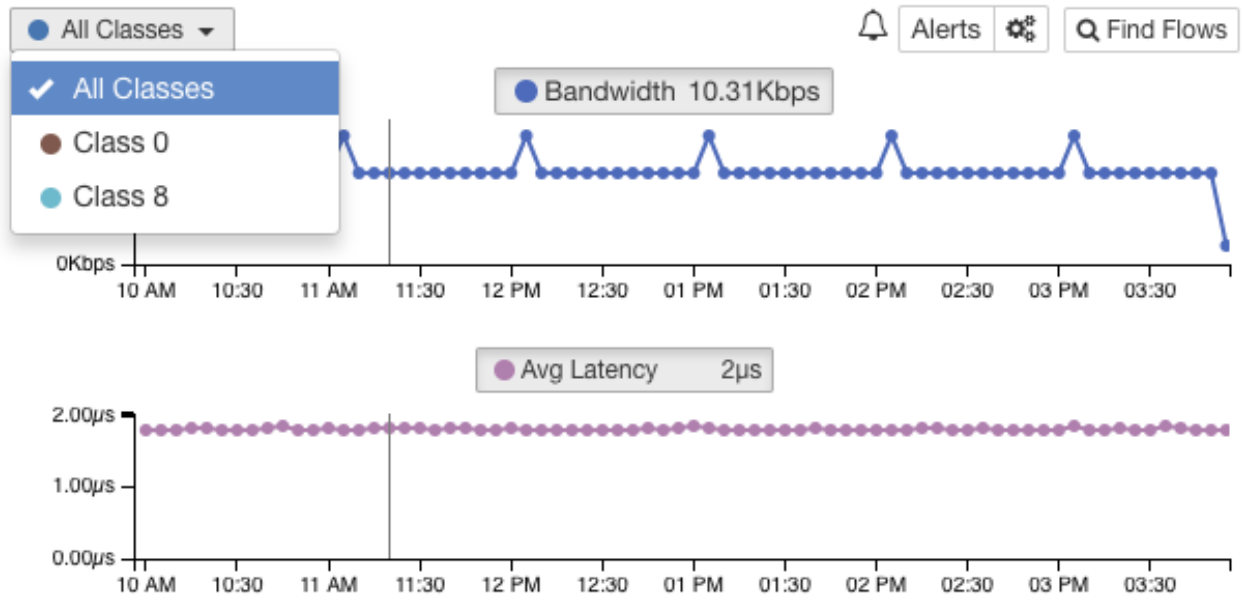


Fig. 8.2.2.2.1: Class selector

When *All Classes* is selected, we show an aggregate of the metric across all flows going though the link. Note that the available classes for a particular fabric link depends on the selected time range. For example, if no flow is with a class of 1 is observed during the selected time range, Class 1 will not be available for selection.

When a particular class is selected, Find Flows button navigates to flow search and filters flows by the selected class of the selected fabric link.

8.2.3 Highlighting Important Links

Performance management of large scale network deployments with dozens of switches and hundreds of links often requires the ability to bubble up anomalies or links through which flows are experiencing significant delays or packet drops. We provide a visual exploration tool on top of the fabric topology graph to quickly identify links with highest throughout, latency or drops indicators.

When selecting one the available metrics on the sidebar, we take the top fabric links based on that metric in the given time range and highlight them according to the metric value. For bandwidth we measure the top links based on the average byte counts on the selected time range, while for latency and drop indicators we consider the maximum value. This helps highlight links based on significant but transient flow latency or packet drops, but not traffic bursts which are more common.

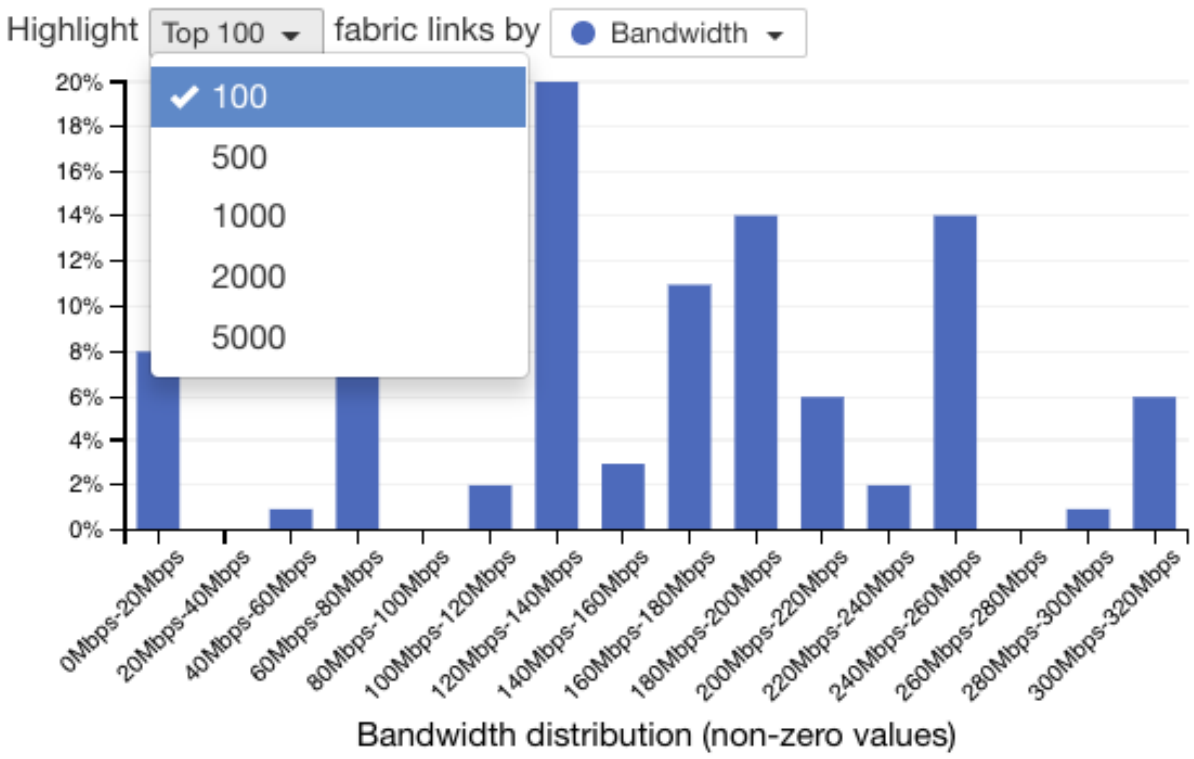


Fig. 8.2.3.1: Highlighting Important Links

In addition to highlighting the important links, we show a histogram chart showing which percentage of the links with non-zero metric value fall within each metric bucket. You may select a range of values over the histogram chart to limit the highlighted links to a more specific range of the metrics. You may de-select the buckets by clicking on the histogram chart background. The topology chart will automatically get updated and highlighted link thickness will get re-scaled. This is useful especially in cases where links are distributed according to a multi-modal distribution.

Note: Depending on the selected time range, some of the top links appearing in the histogram chart may not be valid in the current network topology graph. In these scenarios, it may be possible to have no link highlighted on the topology graph for a select a range of metric values in the histogram chart.

The following animation illustrates the mechanics of the above mentioned workflows.

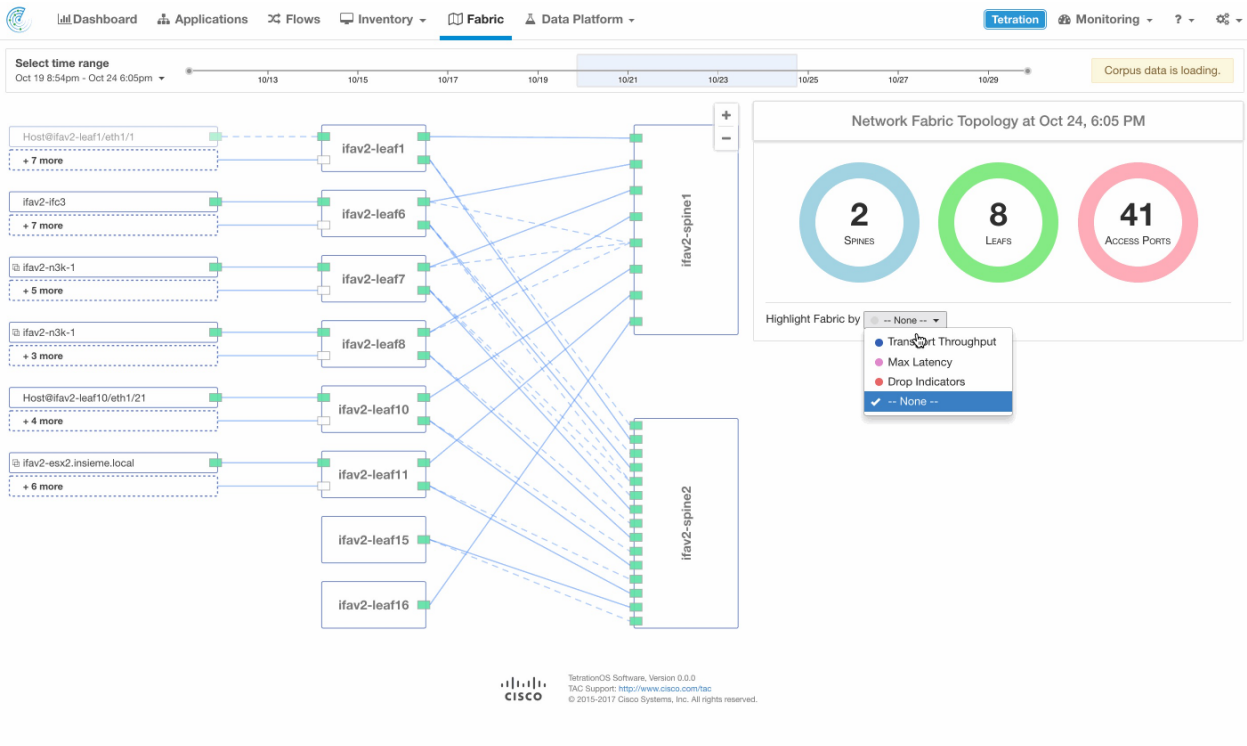


Fig. 8.2.3.2: Histogram chart

8.2.4 Flow Path Overlay

For all flows that are observed by Cisco switches (enabled with Tetration agent), we can map the path the flow has taken through the fabric and show the path overlayed on the fabric topology graph.

The flow details section in Flow Search page shows the path it has taken though the fabric in the forward (consumer to provider) and reverse (provider to consumer) direction under the *Fabric Paths* section. You may hover/click on each hop/link to see more details about the link class, latency and drop indicators and their corresponding time-series information if available. Clicking on the **Fwd** and **Rev** labels takes you to the fabric page where we show the forward or reverse flow path on top of the network topology graph.

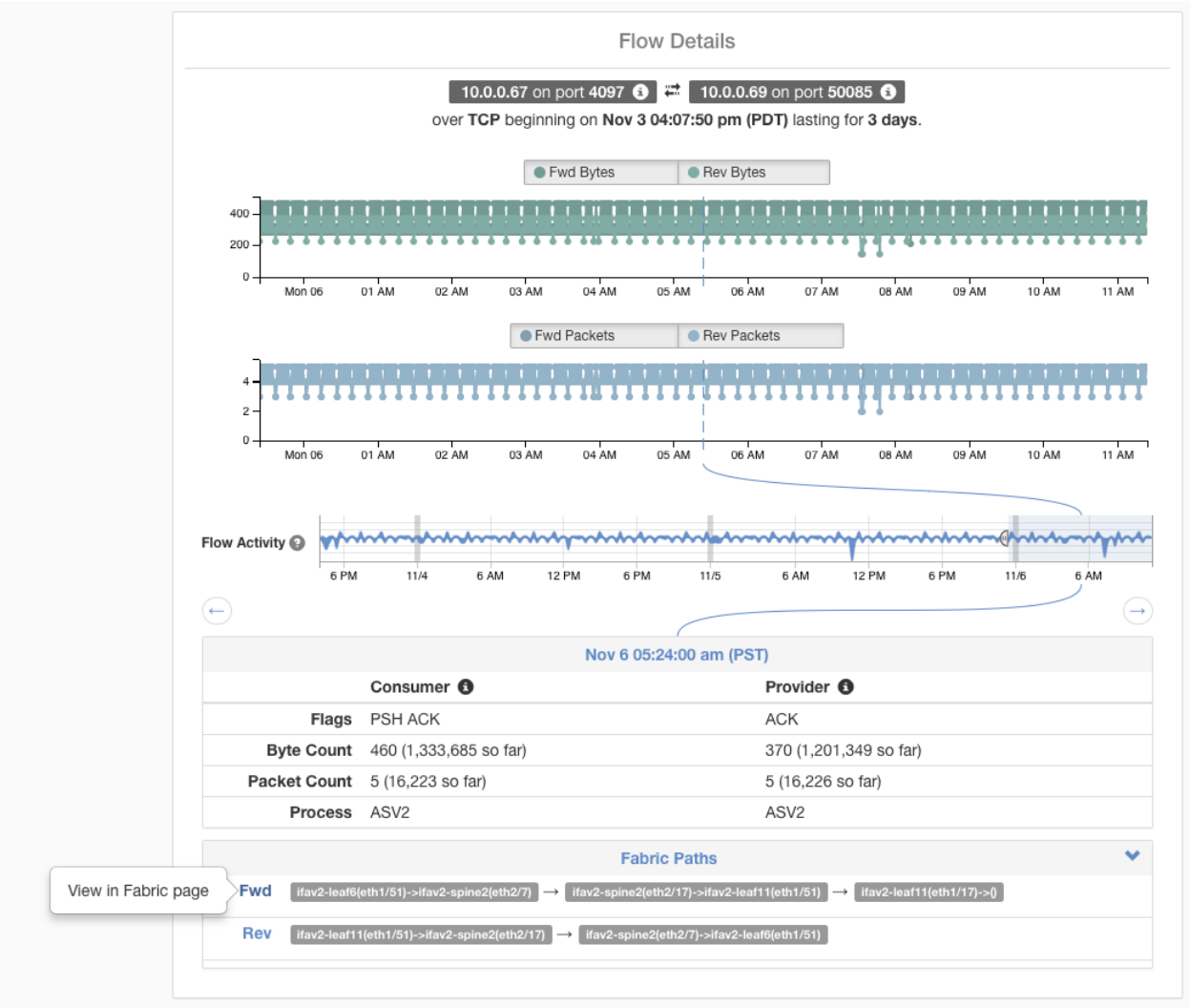


Fig. 8.2.4.1: Navigate from flow search

After navigating to the fabric page, we show a banner on top of the topology graph indicating that we are showing the flow path overlay with some basic information about the flow such as source and destination IP addresses, ports, protocol and timestamp of the flow observation. Note that when observing the *reverse* flow path, the source IP/port is the provider and the destination IP/port corresponds to the consumer.

By default we only show the switches and hosts that are participating in the flow path and the **Path Only** toggle is selected on the flow banner. You have the ability to view the path overlayed on the full network topology graph by selecting the **Show All** toggle. The overlayed flow path always starts at the source IP and terminates at the destination IP. If the IP address nodes overlap with other items of interest, you may drag them in a more desirable position (see the animation below).

Hovering on the *flow path* item on the flow banner reveals the details about the path including the hop by hop switch/host names, port and class information. See figure below:

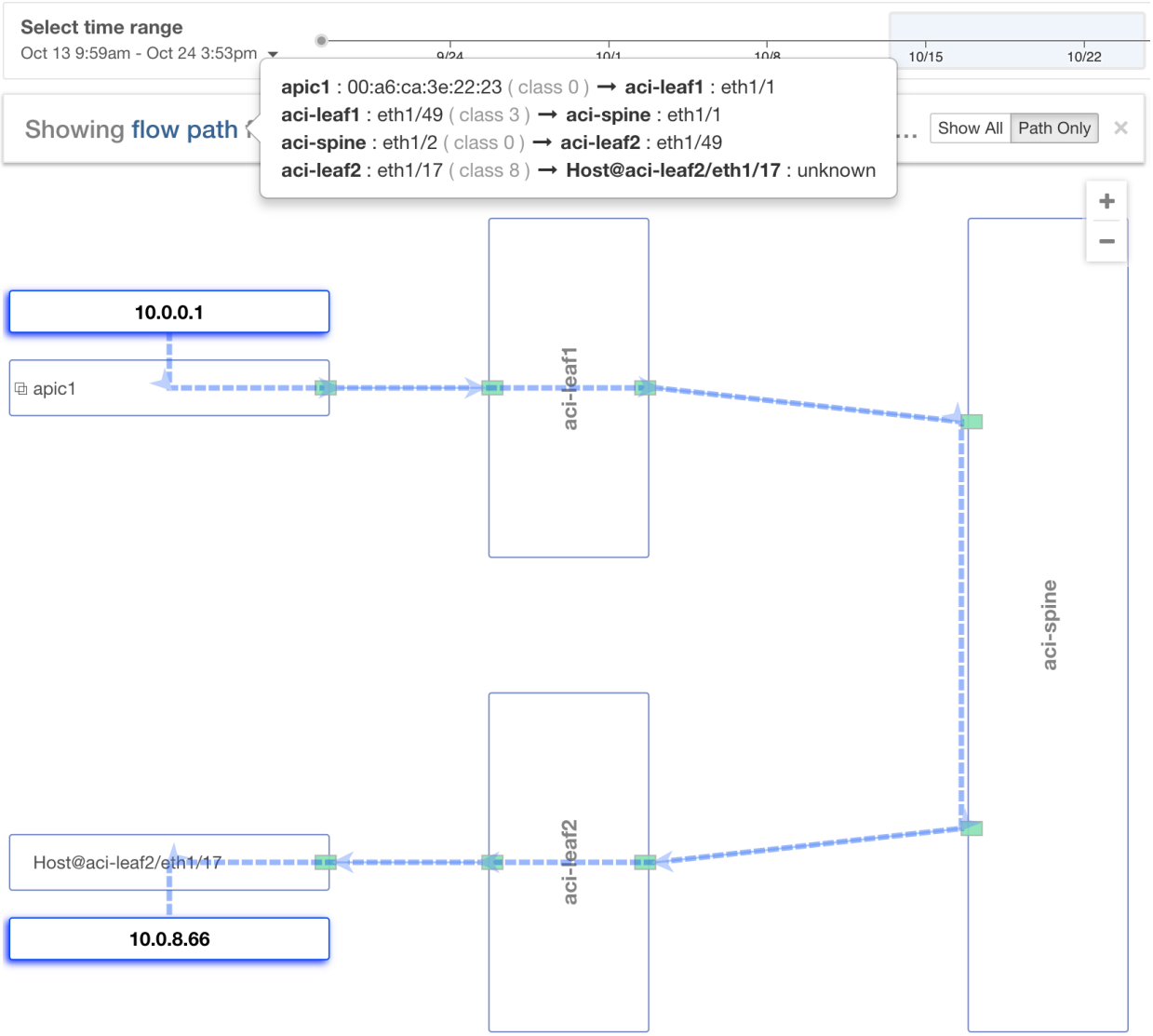


Fig. 8.2.4.2: Full flow path

In the ideal case where all the switch port information along the flow path is known, we show the path traversing the switch ports. However, in case of a partial/missing information, or when the flow starts/terminates at the switches, we may not observe all the ingress/egress switch ports. In these cases, the flow path is visualized to go through the center of the switch.

Another common case of partial information is when the observed flow timestamp does not match the topology timestamp. This can occur if the user changes the selected time range after navigating to the fabric page. The following figure shows that in these scenarios, we show the *partial path* for the flow and mark all the links that are not valid in the current topology with a **No match** warning sign.

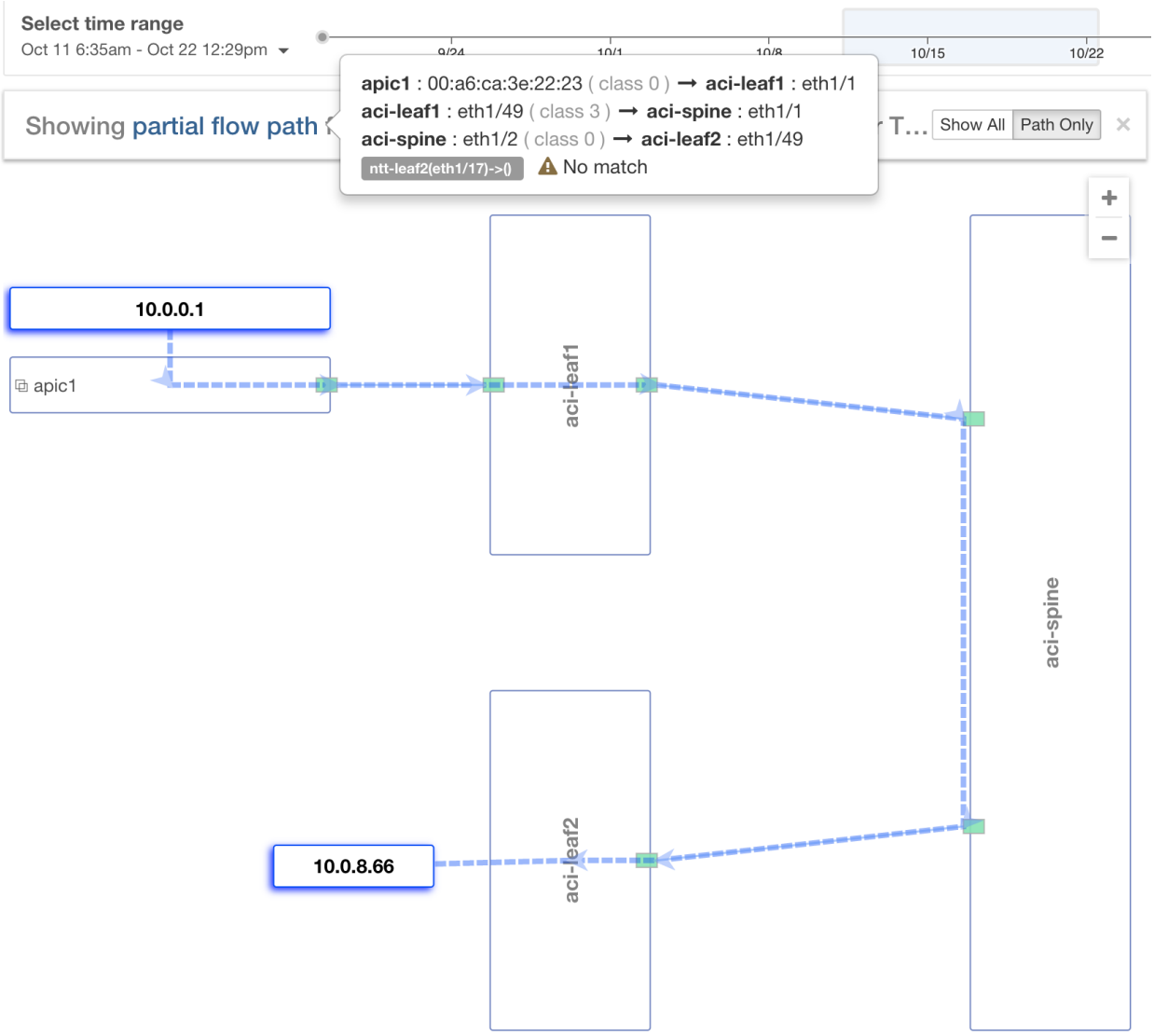


Fig. 8.2.4.3: Partial flow path

All other fabric page features such as link time-series information, link highlighting by metric, etc are available with flow path overlay. You may remove the flow path overlay by clicking on the x (dismiss) button on the right corner of the flow banner. See the following animation for more details on the mechanics of flow path overlay.

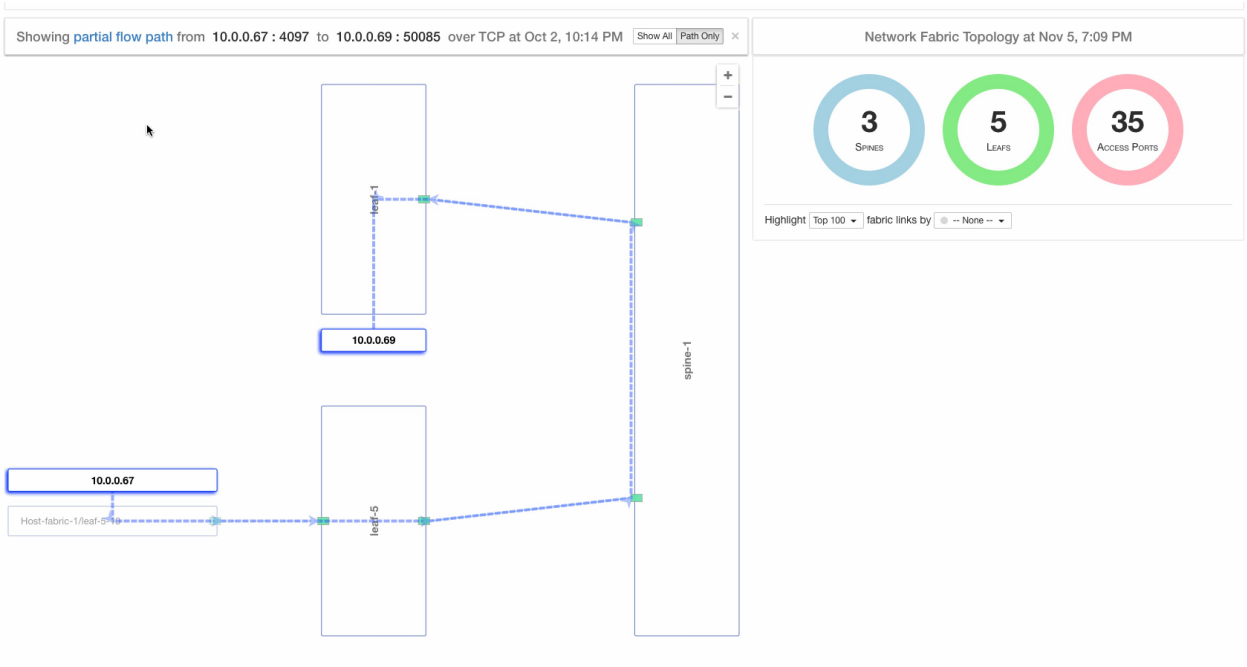


Fig. 8.2.4.4: Flow hightlight

8.2.5 Alerts

Fabric alerts provide a convenient way to receive notifications when certain links are experiencing abnormal load, drops or latency. You can configure multiple alerts for each fabric link based on one or a combination of the time-series metrics collected per link.

8.2.5.1 Configuring Fabric Alerts

First click on the fabric link of interest to reveal detailed link info on the sidebar. Then click on the Alert Configuration button to open the alert configuration modal.

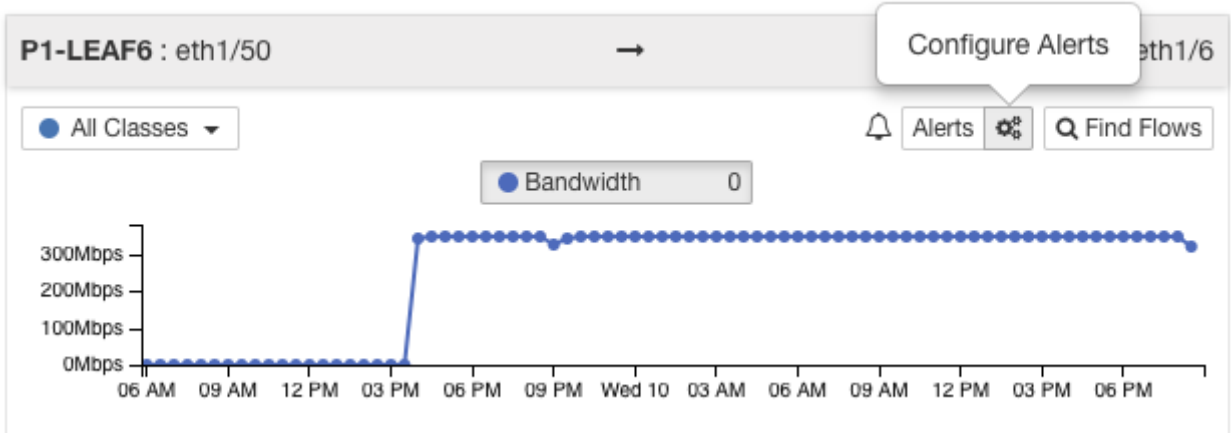


Fig. 8.2.5.1.1: Alert configuration button

The top portion of the alert configuration modal shows the list of configured alerts for the selected fabric link. Note that the configured alerts are scoped by Root Scope (Tetration VRF), so that different tenants have no visibility into each other's alerts.

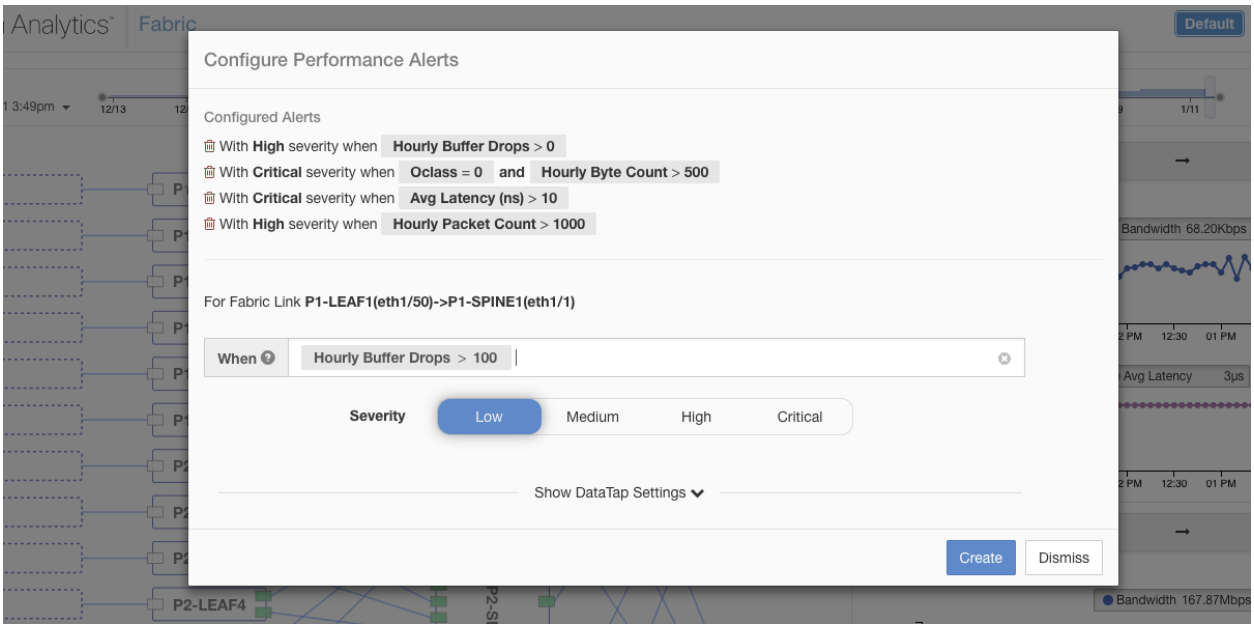


Fig. 8.2.5.1.2: Fabric alert configuration

To configure a new alert, you may enter the condition upon which the alert will be fired in the faceted input box. The alert condition could consist of one or more metrics crossing a certain threshold. When more than one facet are entered, the alert gets fired if and only if the condition represented by each of the facets is met. The following metrics are available for fabric alert configuration:

- **Hourly Packet Count**: The total number of packets traversed through the link, aggregated every hour across all tenants.
- **Hourly Byte Count**: The total number of bytes traversed through the link, aggregated every hour across all tenants.
- **Hourly Drop Indicators**: The total number of drop indicators recorded at link egress, aggregated every hour across all tenants.
- **Avg Latency (ns)**: Average per packet fabric link latency measured in nano seconds.

In addition to the above metrics, you may define a facet to select a particular **class** as part of the alert condition. In this case, any other metric threshold condition will be interpreted based on the aggregated metric for flows going through the selected class. For example, `Class = 3 & Hourly Packet Count > 100` will be triggered if the total hourly packet count for Class 3 flows going through the selected link exceeds 100.

The **severity** of the alert helps quickly focus on the most important alerts if the are too many active alerts. For more information on DataTap settings and advanced alert configurations, please refer to *Data Taps* and *Alerts*.

8.2.5.2 Highlighting Links by Alerts

You can highlight the top N fabric links based on the number of **active** alerts. The workflow is the same as highlighting links based on other metrics. The slight difference is that we show the active alert count next to the egress port for

each fabric link. For example, the number of active alerts for an uplink between leaf-1 and spine-2 is shown on the leaf-1 port.

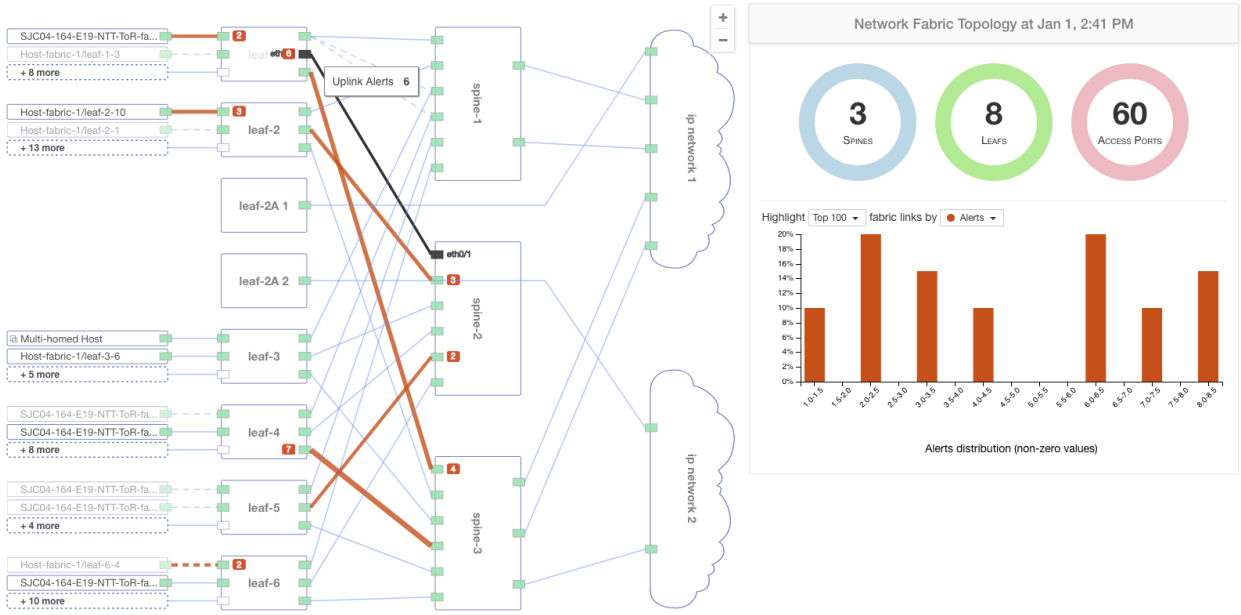


Fig. 8.2.5.2.1: Highlighting Links by Alerts

Note: The number of active alerts is calculated based on the selected time range. Therefore, links with alerts in the past that are currently dismissed may still be highlighted if the selected time range is in the past.

8.2.5.3 Viewing Fabric Alerts

The number of active alerts for any fabric link is shown on the sidebar next to the *Alerts* button. The active alert count is based on the selected time range and is shown whether the top N links are highlighted or not.

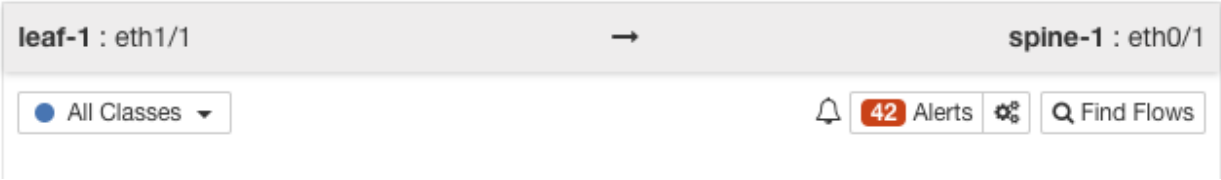


Fig. 8.2.5.3.1: Viewing Fabric Alerts

Clicking on the alert button navigates to the Tetration *Current Alerts* page and automatically populates the filters to focus on active alerts of the selected fabric link. You may further narrow down the alerts by severity and status.

The Clicking on any of the alert rows reveals more details about the alert such as the alert trigger, class and the observed metrics that caused the alert to trigger. Clicking on the fabric link hyper-link navigates back to the fabric page while highlighting the fabric link.

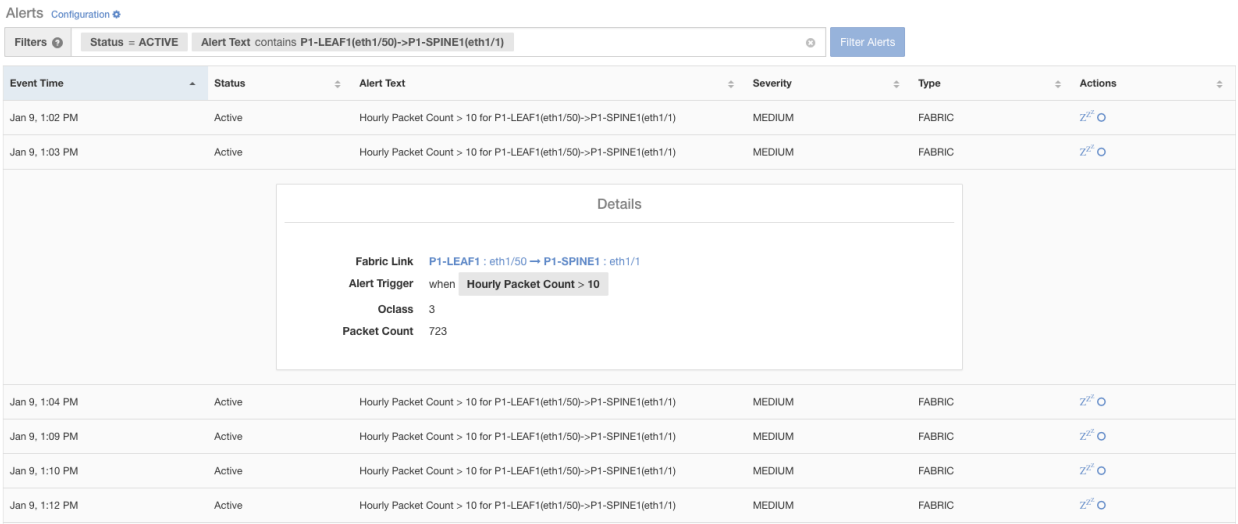


Fig. 8.2.5.3.2: Fabric alert detail

Fabric Alerts can be configured as:

1. Individual Alert
2. Summary Alert

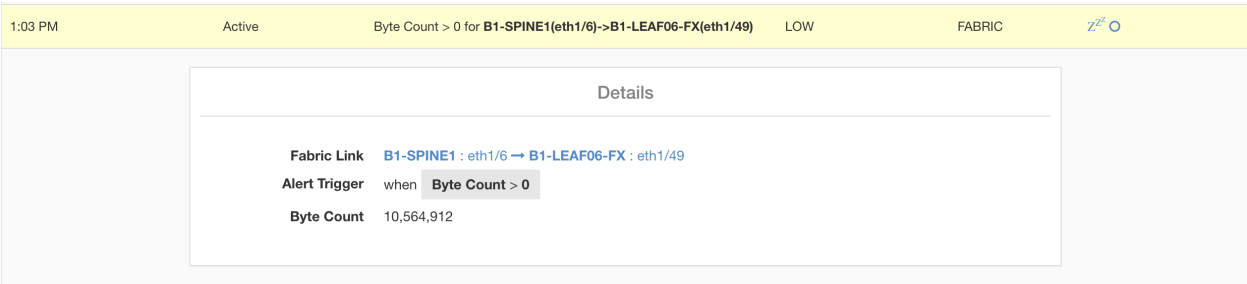


Fig. 8.2.5.3.3: Example of Individual Fabric Alert

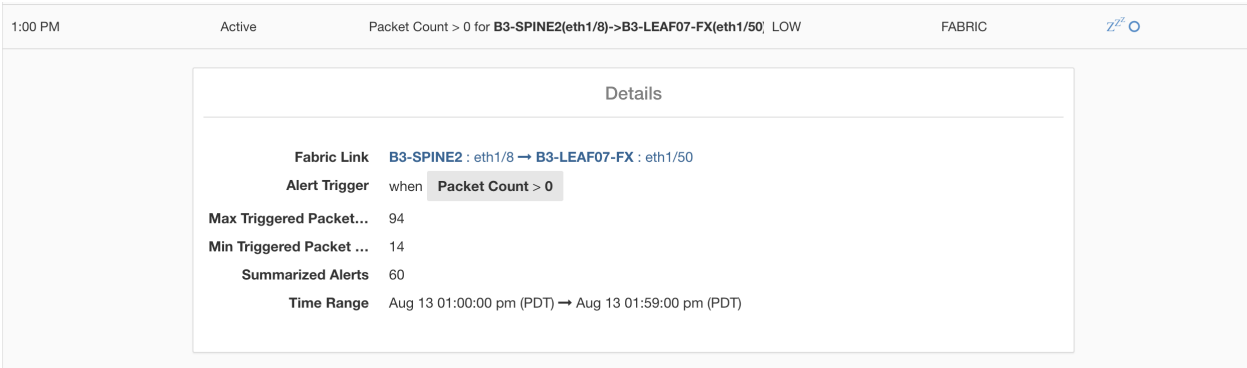


Fig. 8.2.5.3.4: Example of Summary Fabric Alert

8.2.5.4 Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* field is structured and contains the following subfields for fabric alerts:

Field	Alert Type	Format	Explanation
src_port	*all*	string	Source access port of Leaf/Spine switch
configured metric	*all*	int	Value of the metric configured
internal_trigger	*all*	json	Configuration query which triggered the alert
time_range	Summary	*[list, two timestamps]*	Unix timestamps of start and end time of summarized alerts
dst_name	*all*	string	Name of the ingress
dst_port	*all*	string	Destination access port of leaf/spine switch
summarized_alerts	Summary	integer	Count of alerts in summary
src_name	*all*	string	Name of the egress
link_id	*all*	string	egress -> ingress

Structure of *internal_trigger*: A list containing structured data. The structured data is a map containing the following fields:

Field	Format	Description
datasource	string	Source of Data (Eg: Fabric)
rules	json	Filter Query json with fields: field, type and value
label	string	Label for alert. Eg: Alert Trigger

Example of kafka output for Individual Fabric alert


```
{
  "severity": "LOW",
  "tenant_id": 676767,
  "alert_time": 1595733072450,
  "alert_text": "Packet Count > 0 for <link_id:B1-SPINE1(eth1/4)->B1-LEAF04-FX(eth1/
↪49)>",
  "key_id": "51469198-29d0-38fb-aa18-fed72682ac18",
  "root_scope_id": "5dcf0c65497d4f57bc71e367",
  "alert_conf_id": "5f1b399ca5875302ecd68504",
  "type": "FABRIC",
  "event_time": 1595725620000,
  "alert_details": "{\"src_port\":\"eth1/4\",\"internal_trigger\":{\"datasource\":\
↪"fabric\",\"rules\":{\"field\":\"pkt_count\",\"type\":\"gt\",\"value\":0},\"label\
↪":\"Alert Trigger\"},\"dst_name\":\"B1-LEAF04-FX\",\"dst_port\":\"eth1/49\",\"src_
↪name\":\"B1-SPINE1\",\"pkt_count\":57694,\"link_id\":\"B1-SPINE1(eth1/4)->B1-LEAF04-
↪FX(eth1/49)\"}"
}
```


Example of kafka output for Summary Fabric alert

```
{
  "severity": "LOW",
  "tenant_id": 676767,
  "alert_time": 1597189325689,
  "alert_text": "Byte Count > 0 for <link_id:B1-SPINE1(eth1/6)->B1-LEAF06-FX(eth1/49)>
↪",
  "key_id": "948779ca-726b-3e48-81de-b09d0f544c25",
  "root_scope_id": "5dcf0c65497d4f57bc71e367",
  "alert_conf_id": "5f20782aa7a5ea08f0c28c33",
  "type": "FABRIC",
  "event_time": 1597183200000,
  "alert_details": "{\"src_port\":\"eth1/6\",\"max_triggered_byte_count\":12672738,\
↪"min_triggered_byte_count\":10538358,\"internal_trigger\":{\"datasource\":\"fabric\
↪",\"rules\":{\"field\":\"byte_count\",\"type\":\"gt\",\"value\":0},\"label\":\
↪"Alert Trigger\"},\"time_range\":[1597183200000,1597186740000],\"dst_name\":\"B1-
↪LEAF06-FX\",\"dst_port\":\"eth1/49\",\"summarized_alerts\":60,\"src_name\":\"B1-
↪SPINE1\",\"link_id\":\"B1-SPINE1(eth1/6)->B1-LEAF06-FX(eth1/49)\"}"
}
```


Example of un-stringified alert details for Individual Fabric alert


```
{
    "dst_name": "B1-LEAF04-FX",
    "dst_port": "eth1/49",
    "internal_trigger": {
        "datasource": "fabric",
        "label": "Alert Trigger",
        "rules": {
            "field": "pkt_count",
            "type": "gt",
            "value": 0
        }
    },
    "link_id": "B1-SPINE1(eth1/4)->B1-LEAF04-FX(eth1/49)",
    "pkt_count": 57694,
    "src_name": "B1-SPINE1",
    "src_port": "eth1/4"
}
```


Example of un-stringified alert details for Summary Fabric alert


```
{
    "dst_name": "B1-LEAF06-FX",
    "dst_port": "eth1/49",
    "internal_trigger": {
        "datasource": "fabric",
        "label": "Alert Trigger",
        "rules": {
            "field": "byte_count",
            "type": "gt",
            "value": 0
        }
    },
```


(continues on next page)

(continued from previous page)


```
    "link_id": "B1-SPINE1(eth1/6)->B1-LEAF06-FX(eth1/49)",
    "max_triggered_byte_count": 12672738,
    "min_triggered_byte_count": 10538358,
    "src_name": "B1-SPINE1",
    "src_port": "eth1/6",
    "summarized_alerts": 60,
    "time_range": [
        1597183200000,
        1597186740000
    ]
}
```


8.3 Burst detection

Since the 2.3 release, Tetration flow monitoring supports monitoring a flow's burstiness across time for **Deep** and **FX/FX2 Hardware sensors**. See figure below for an intuitive definition of a burst:

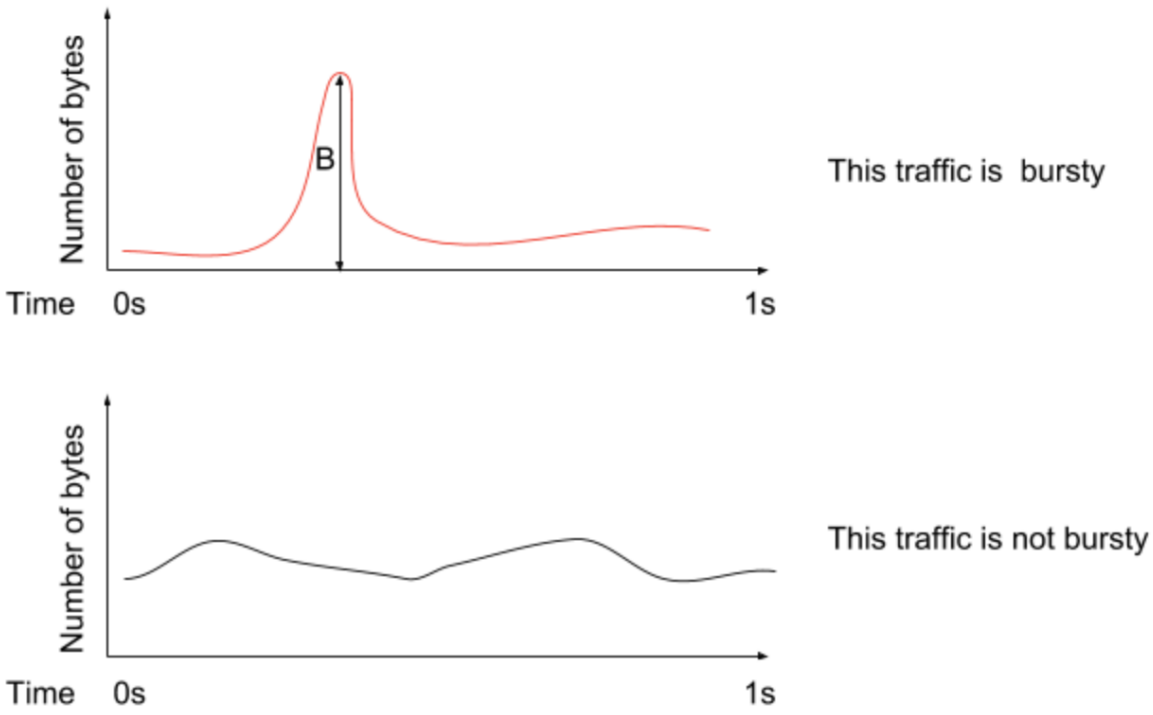


Fig. 8.3.1: Definition

More precisely, divide the time period of 1s into 1000 time slices of 1ms each. Let B be the maximum number of bytes sent within a 1ms time slice, and let T be the total number of bytes sent within the 1s time period. We monitor only those bursts that meet the following thresholds:

- B >= 20kB AND

- B/T >= 0.2, i.e., at least 20% of traffic within the second is sent within the 1ms interval

If there are multiple bursts that satisfy the above criteria within the 1s time period, we pick the largest one.

For **FX/FX2 Hardware sensors**, the time slice for measuring bursts is 64usec. For **EX Hardware sensors**, the time slice is 8ms. If there are multiple sensors along a flow's path, the largest burst (in kB) that satisfies the aforementioned thresholds will be recorded.

8.3.1 Burst aggregation

Over one minute flow observation, we publish the following statistics per-flow direction (both forward and reverse). They are available on flow search.

- Burst indicators: The number of bursts satisfying the thresholds.
- Burst + drop indicators: The number of instances where there was a packet drop for the flow that coincided with its maximum burst. This feature is only available on FX/FX2 hardware sensors.
- Max burst size: The size of the maximum burst in the observation (in kilobytes).

8.4 TCP Performance Debugging using Tetration on itself

Contents

8.4.1 Overview

This document is a case-study of how we debugged performance issues in a real-world large-scale complex application, and how we tuned the system to gain a 10% performance improvement. The goals of this document are to:

- Highlight the new performance related features in Tetration monitoring appliance.
- Show how we can use the new features to understand an application's network-oriented performance.

8.4.2 Problem

Internally, the Tetration appliance consists of multiple services implementing different functionality. With limited resources, it becomes important to understand where the application bottlenecks are, and how one can optimise the system to improve overall performance.

On first glance, optimising a complex system can seem daunting, but it's important to:

1. Understand the overall system architecture to focus attention to important components in the critical path.
2. Identify application-specific key performance indicators to measure and quantify the benefits of any change to the system.

... before attempting to do any performance analysis.

At a high level, the Tetration architecture consists of the following services in the critical path:

Sensors → Collectors → Real time data processing pipelines → Summaries

8.4.3 Goals

We care about two main aspects:

1. The total ingestion rate measured in flow events/second. Increasing the maximum ingest rate allows us to capacity plan and scale better.
2. The average processing time for the data processing pipelines. This allows us to keep up with the scale, and also support more features.

Both aspects are related to each other. We will focus on reducing the average processing time for the data processing pipelines, which will enable us to scale better.

8.4.4 How did we investigate?

1. Search for network and application bottlenecks after scaling up the load on the cluster. We only focus on resources such as the physical network and the application's network stack where we have visibility with Tetration sensors.
2. Dig deeper into the cause for network and application bottlenecks.
3. Fix the cause on a live cluster and repeat the analysis.

8.4.4.1 Searching for network / application bottlenecks

Flow search exposes the following facets with which one can search for flows experiencing common network / application issues:

Table 8.4.4.1.1: TCP Performance Facet details

Value	Explanation
Consumer App limited	The consumer application is not draining data fast enough from the TCP socket buffer, and advertised a "Zero Window" to the Provider (and NOT network limited).
Provider App limited	The provider application is not draining data fast enough from the TCP socket buffer, and advertised a "Zero Window" to the Consumer (and NOT network limited).
App limited	Either Consumer App limited OR Provider App limited.
Network limited	Flows experiencing lower throughput due to congestion window reductions and an increase in packet retransmissions (and NOT App limited).
Both	Flows either App limited or network limited.

Using the "TCP Performance" facet, we searched for flows that are limited either in the network or in the application. We found the following:

1. There were consumer and provider limited flows to the Hadoop distributed file system's (HDFS) data nodes (port 50010 in the "Top Port" view)

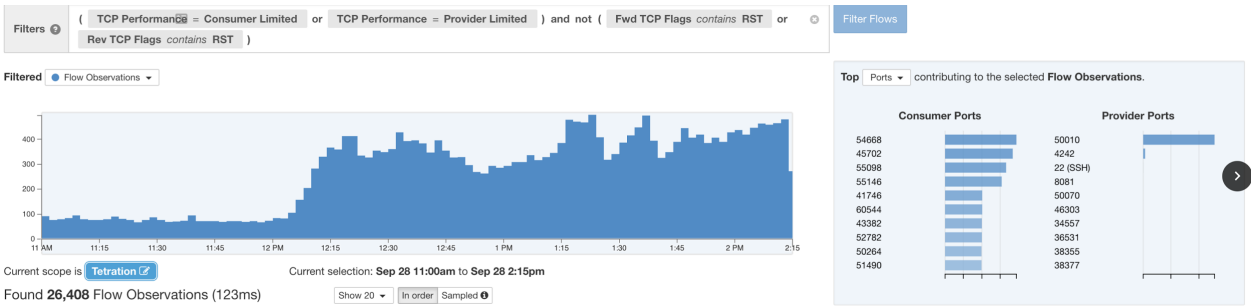


Fig. 8.4.4.1.1: Consumer and provider limited flows

2. There was an increase in the number of provider-limited, i.e., HDFS-limited flows after starting the scale test at around 12:00PM.

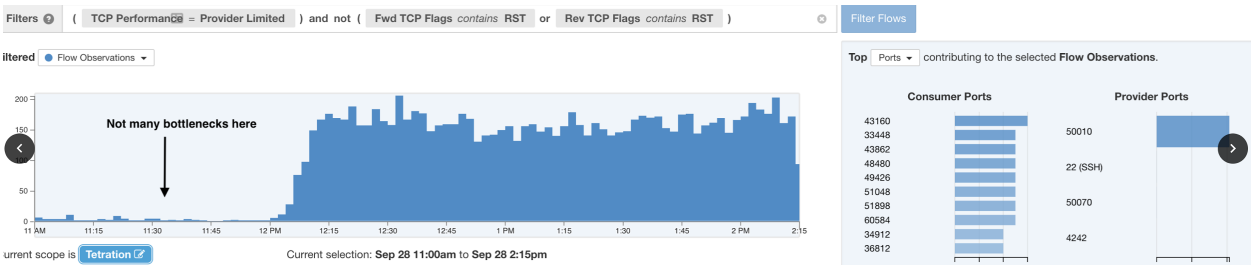


Fig. 8.4.4.1.2: Increase in the number of provider-limited

3. Using the "Top Hostname" view, we see that the consumers are mostly hosts with "collectorDatamover" in their hostnames. This shows that the data node providers are probably unable to keep up with the data pushed by the collectors.

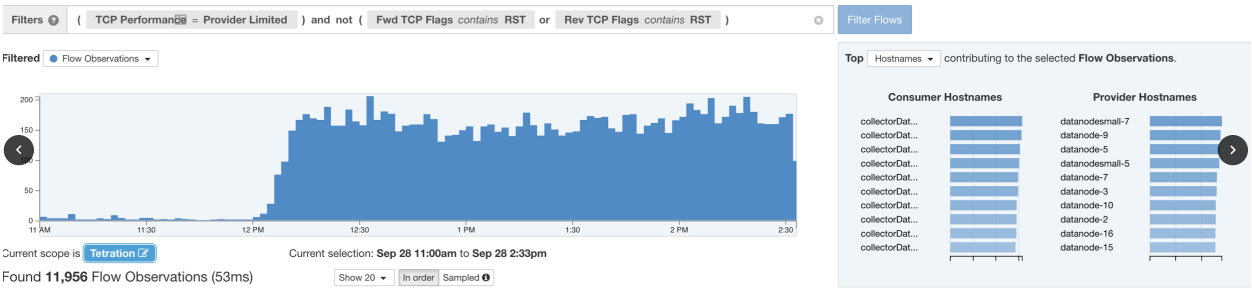
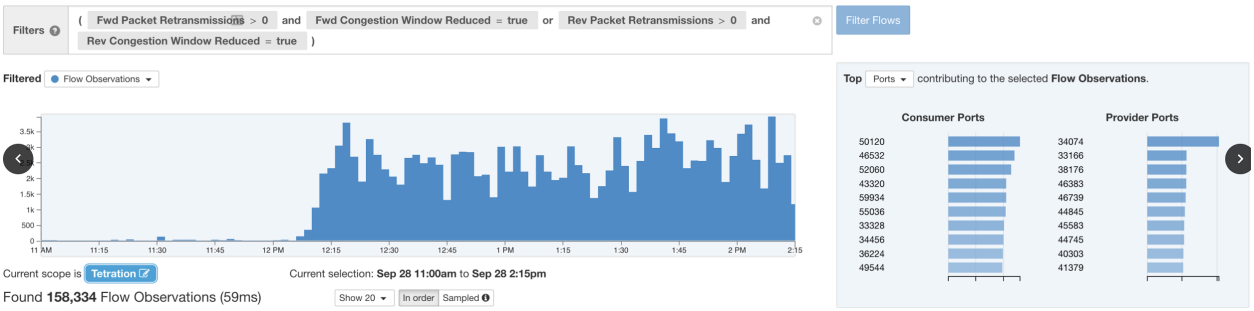
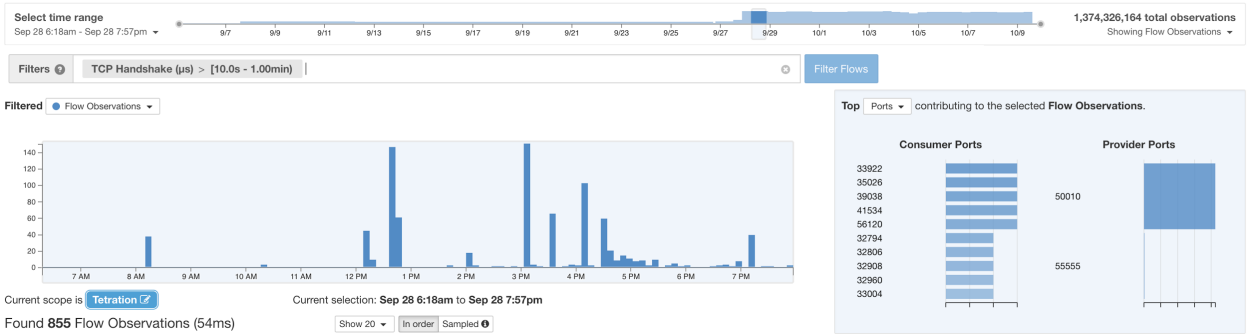


Fig. 8.4.4.1.3: Consumers are mostly hosts with "collectorDatamover" in their hostnames

4. We also found several network limited flows after the stress-test began:



5. Some TCP connections to data nodes (port 50010) took a long time (multiple seconds) to establish:



8.4.4.2 Reducing application bottlenecks

After observations 1—3, our immediate reaction to was to tune the TCP send and receive buffers, especially on the datanode machines. Once we increased the receive buffers, we saw a drop in the number of application bottlenecked flows as one would expect:

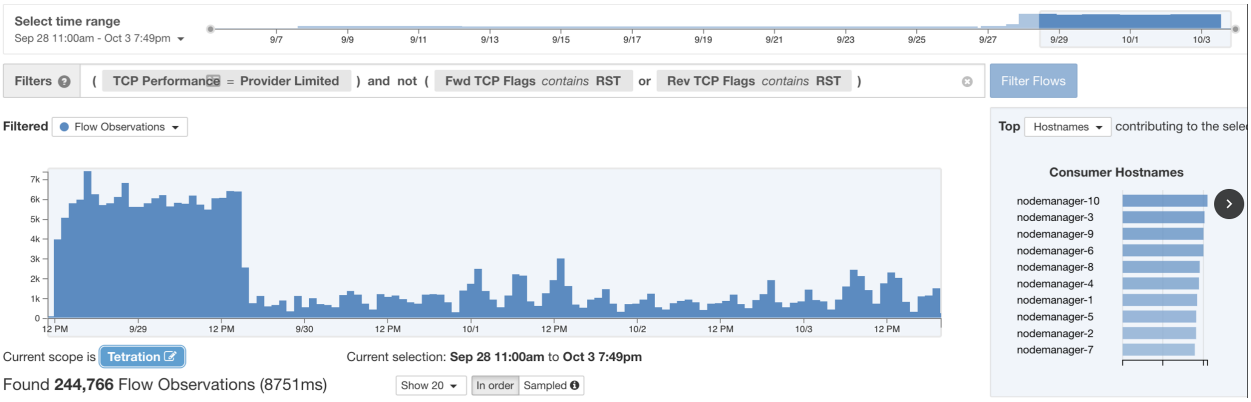


Fig. 8.4.4.2.1: Reducing application bottlenecks

Note that there still are some flows that are bottlenecked on some applications, but what we noticed was that the bottlenecks at the datanodes disappeared, and they were spread across other minor applications.

Before digging any deeper, we checked if our key performance indicator (average processing time) had any improvement and noticed that it did not have any statistically significant improvement. After some thought, we realised that while we reduced data copy stalls between "collectorDatamovers" and data nodes, this improvement did not benefit the more expensive data processing at the pipelines. The key takeaway is that the data copy itself is not the key determining factor in overall processing time.

8.4.4.3 Network bottlenecks

Continuing the search to components downstream from Sensors → Collectors, we noticed that flows in the data processing pipelines were never application-limited. In fact, pipeline flows were mostly network limited. Network drops reduce TCP throughput, which in turn slow down applications. Drops, especially at early stages of a TCP flow can severely impact its throughput. With per-flow visibility from multiple vantage points, we confirmed that the flows with high establish latency were due to SYN drops. In fact, many short data transfer flows durations were all clustered around 3s. A quick search showed us that the SYN retransmit timeout is set to 3s by default.

Our data pipeline workload is soft-realtime, and the tail performance of the slowest network flows have an impact on the overall performance. Although it's possible to architect applications such that they are not impacted by straggling network flows, it's beyond the scope of this discussion.

8.4.4.4 Reducing packet drops

We looked at network drop counters at various points in our network stack (VMs, hypervisor, physical NIC, and network switches) and found that a majority of packet drops happened on the virtual NIC between the hypervisor and the VM. At scale, although we were not saturating our network interfaces, we noticed we were running close to CPU capacity. Our hypothesis was that contention for CPU resources at the hypervisor layer due to time sharing with VMs causes packet drops. We quickly confirmed this by checking the counters in /proc/net/softnet_stat on the hypervisors, which indicated packet drops on the hypervisor virtual NICs.

There are many ways to reduce packet drops, but we tested a quick fix by increasing the MTU size from 1500B to 9000B (jumbo frames) at the VM, hypervisor, NIC, and switches. Increasing MTU reduces the number of packets per second, which decreases CPU usage. We immediately saw a reduction in the number of TCP retransmissions as well as the number of flows with duration clustered around 3s (due to SYN retransmissions):

Reduction in packet retransmissions:

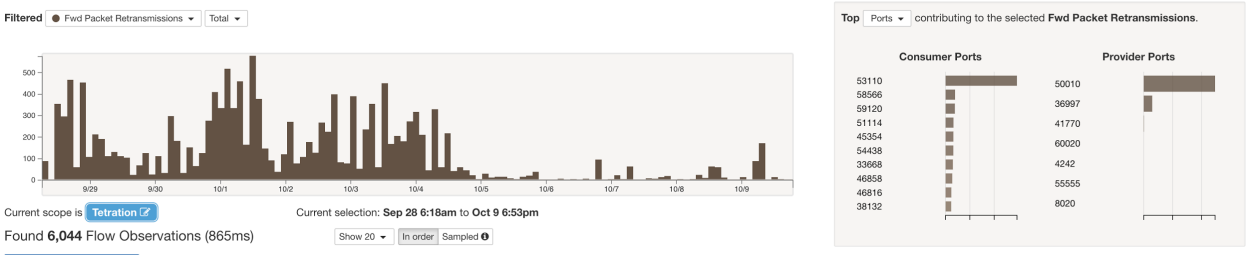


Fig. 8.4.4.4.1: Reduction in packet retransmissions

Reduction in SYN retransmissions:

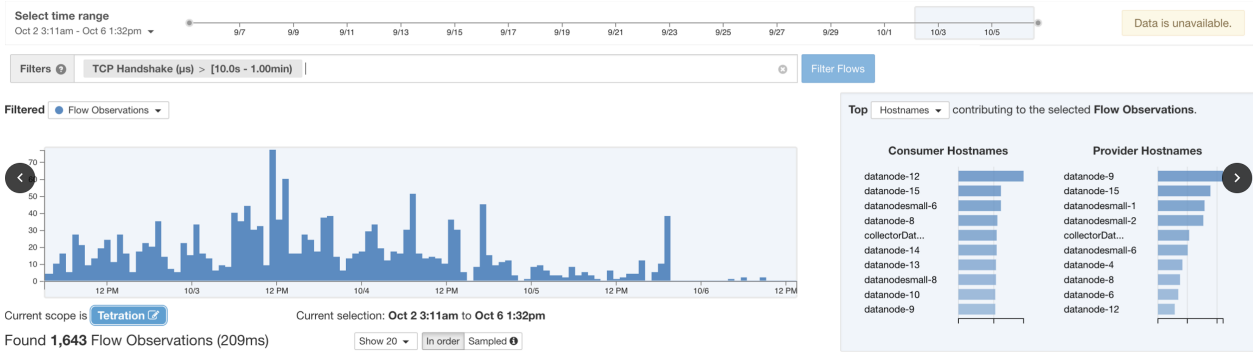


Fig. 8.4.4.4.2: Reduction in SYN retransmissions

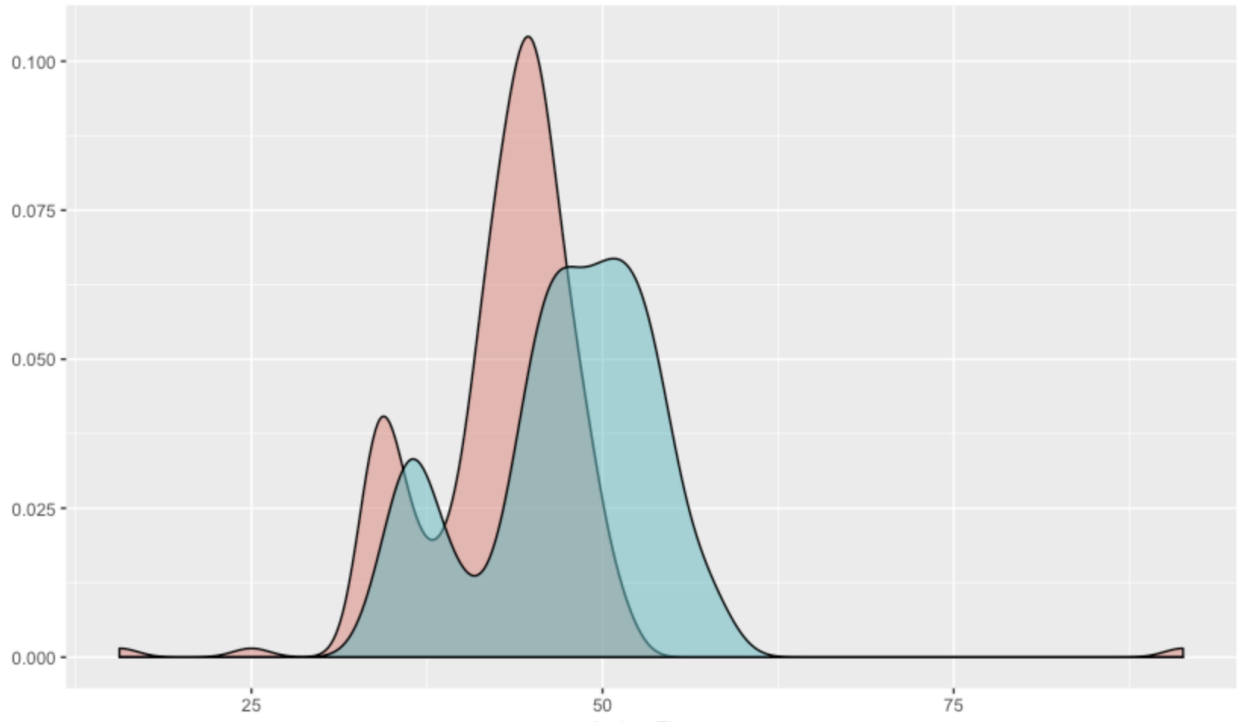
8.4.4.5 Did this change help?

While we did not see an immediate change, we monitored the data processing times over a few hours and visually noticed a small difference. Our processing times vary due to many factors such as input sizes, the type of input, random events (e.g., garbage collection), and other periodic hourly jobs. When we looked at the distribution of the data processing pipeline runtimes before and after the change, we noticed 10% reduction in the average runtime for many pipelines, and a comparable reduction in the standard deviation of their runtimes as well. Below are the distributions.

X-axis is the processing time. Y-axis is the density.

Legend:

- blue: before the MTU change
- red: after the MTU change



Subsequently, in another experiment we tuned the virtual NIC buffer sizes (leaving the MTU at 1500B) between the hypervisor and VMs and saw a similar performance improvement. Unlike the MTU change that significantly reduced packet drops, this change completely eliminated packet drops at all software network layers.

8.4.5 Takeaways

There are always bottlenecks in the system, and bottlenecks in the critical path are most important. A balanced system design would mean we hit bottlenecks at all resources (CPU, memory, IO, network) when pushed to its limit ensuring optimal resource utilisation. Therefore, it's important to:

1. Monitor each aspect of the system and its performance.
2. Reason about observed bottleneck indicators within the context of the overall system architecture to focus attention on the critical components.
3. Understand that even if the network utilisation is not 100%, it could still be a bottleneck if the workload is bursty.
4. Understand natural variations in the workload to ensure a fair comparison before and after a tweak.

CHAPTER

NINE

DATA PLATFORM

Warning: Neighborhood and Lookout Annotation have been relocated. All other platform pages are deprecated. These other platform features will no longer be available in the next major release, 3.6.

Warning: Data Platform feature provides users and site admins access to Tetration data. With this high privilege access, improper usage may impact Tetration functionality. **Users discretion is advised.**

The Data Platform allows users to use apps built by Tetration or to build their own apps using a combination of Tetration data and their own data.

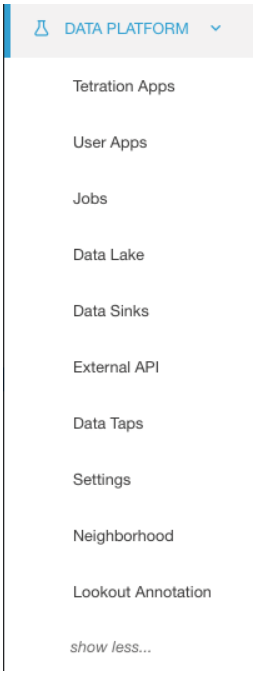


Fig. 9.1: Data Platform

All users now view some Data Platform features.

- All users have access to enable Tetration Apps, and to view Data Taps and Data Sinks.
- Users with a Developer role can additionally create User Apps, access debug logs, schedule Jobs, and can access the Data Lake (upload, write, and delete data).

- Root scope owners are now Data Platform admins, and are allowed to create/delete Data Taps, Data Sinks and Visualization Data Sources.

For more information about user roles see /settings/roles

Navigating the Data Platform

The Data Platform menu is on the left and expands when you click on it.

9.1 Important Changes

9.1.1 Release 3.5 Notes

1. Neighborhood and Lookout Annotation have been relocated. All other Data Platform pages are deprecated. These other Data Platform features will no longer be available in the next major release, 3.6.
2. Data Platform admin is now called Data Tap Admin.
3. Zeus data feed has been discontinued. Users do not have to take any action. However, if there are any filter and policy related to zeus tag, rules are no longer effective. Users can chose to clean up the filter and any related policy if needed but this will not break anything.

9.1.2 Release 3.4 Notes

1. Neighborhood Geo feature has been updated and now has improved map visibility and time range selection. See *Neighborhood*
2. Lookout Annotation now has a concept of a 'compromised machine'. Lookout Annotation will also fully update annotations to latest threat lists (including clearing labels for ips no longer in the lists). See *Lookout Annotation*

9.1.3 Release 3.3 Notes

1. **Changes involving User Apps feature**
   - Spark is upgraded to version 2.3. This may require some changes to Notebooks in User Apps. See *Spark 2+ changes for User Apps*
   - Improvements to error messages when using ExternalAPI. See *External API Connections Debugging*
   - Additional option for writing/reading JSON blob when using `IO.write`/`IO.read`. See PythonSampleNB and ScalaSampleNB notebooks. Remember to *Reset Sample User Apps* from *Data Platform - Settings* to get latest versions of the notebooks.
   - New example notebooks. See *ScalaSampleNB* for overview of sample notebooks.
2. **Changes to Neighorhood App. See *Neighborhood***
   - New Geo Feature
3. **Compliance Alert changes. See *Compliance***
   - Can be configured on Applications with Live Analysis profiles (in addition to Enforced applications)
4. **Changes to Lookout Annotation. See *Lookout Annotation***
   - Lookout Annotations may become disabled on upgrade to 3.3. Please double-check that sources are enabled if you wish to have those annotations and/or alerts.

9.1.4 Older Changes

9.1.4.1 Release 3.1 Notes

1. **Data Lake Tetration Machine and Inventory data are deprecated.**
   - Machine and Inventory data will no longer be available in User Apps nor as a VDS for dashboards, starting from 3.2.
2. **Changes to Neighborhood App. See *Neighborhood***
   - New feature: query filters. Neighborhood graph can now be filtered based on provider port and protocol.
   - Enabled by default on all root scopes.
3. **Changes to Compliance App. See *Compliance***
   - Compliance App is now implicit, and is not shown in the App Store.
   - Compliance alerts are only on Enforced Applications.
   - New alert configuration option to include constituent flows.
4. **Alerts App**
   - Implicit configuration via new "Alerts > Configuration" page; Alerts App is no longer shown in the App Store.
5. **Data Sinks and MDT**
   - Data Sinks and MDT created in releases 2.3 and prior, used Kafka port 9093 to send or read data from outside the cluster. Starting 3.1 release the Kafka port has been changed to 443. Users have two options for this:
     - Change the port from 9093 to 443 in the kafkaBrokerIps.txt file
     - Delete Data Sink and recreate a new one

9.1.4.2 Release 2.3 Notes

1. **Managed Data Taps**
   - Tetration now provides **Managed Data Taps (MDT)** (These connect to a Kafka cluster hosted within the Tetration cluster)
   - A Managed Data Tap called `Alerts` is automatically created for each root scope. This `Alerts` MDT can be used when configuring alerts within Tetration Apps.
   - See *Managed Data Taps* for more information about MDTs and how to read messages sent to an MDT.
2. **Summary Alerts**
   - Some alerts can now be configured to send at a less frequent interval.
   - This feature is currently available for Compliance, Fabric, and Neighborhood alerts.
   - See *Summary Alerts* for more information.
3. **New Tetration App: Lookout Annotation App**

- This app provides automatic annotation of inventory matching ip lists, along with alerts when flows are found with these inventory ip items. See *Lookout Annotation*

4. **Scoping of Shared Data is changed from tenant to root scope.**
   - If there are more than one root scopes per tenant, the data will be moved to one of the root scopes within the tenant.
   - In case of one root scope per tenant, this change is transparent. No action is required from user.
   - In case of multiple root scopes per tenant, please re-upload the data if its not in the root scope that you expect.
5. **Data Sinks**
   - Data Sinks created prior to 2.3 release will not work upon upgrade to 2.3 release.
   - Data Tap Admin can delete the pre 2.3 release Data Sink, and create new Data Sinks.
   - Data not processed by Data Sink Dumper could be lost during upgrade.
6. **Other Notes**
   - Compliance alerts now incorporate the trigger condition in their Key Id. This means that:
     - Alerts for rejected counts can be snoozed independently of alerts of escaped, for example
     - Old snooze status will not have an effect on the new alerts generated. With the new summary alert feature, users should consider whether snoozing or summary alerts are preferred for the alert in question.

9.1.4.3 Release 2.2 Notes

1. **Compliance Tetration Apps**
   - Compliance alerts are now configurable. To generate compliance alerts, users **must** configure alerts on each compliance instances
   - Enabling compliance apps **will not** generate any alerts
   - Any compliance alerts that were snoozed or allowed will need to be re-snoozed or re-allowed when upgrading to 2.2.x
   - Compliance alerts will no longer go to the "compliance" topic in a Data Tap. Please see Data Taps for more information.
2. **Data Taps**
   - Data Taps now encapsulates a topic. Topic is a mandatory input while creating a new Data Tap.
   - Data Taps configured in releases prior to 2.2.x will be automatically migrated to use a `defaultDatatap` topic.
   - `Data Tap Admin` can update the topic to the topic of their choice. Once Data Tap configuration values are successfully validated and updated, subsequent messages will be sent to the Data Tap with the newly configured topic.
3. **User role**
   - `Lab` user role (configured as service provider) is no longer present
   - Developer role is the newly introduced role for users to use all Data Platform features. The role must contain `DEVELOPER` capability

- Data Platform admin role is newly introduced for users to perform admin-level Data Platform features. The role must contain `OWNER` capability of a root scope

4. **Neighborhood Tetration App**
   - Neighborhood alerts will need to be re-snoozed or re-allowed when upgrading to 2.2.x
   - Neighborhood alerts will no longer go to the "neighborhood" topic in a Data Tap. Please see Data Taps for more information.
5. **Demo Collector (NetFlow)**
   - NetFlow Collector now sends all fields coming in the packet instead of just flow data..
6. **DataSink Dumper App**
   - DataSink Dumper App only expects **JSON** format. The `NetFlow Collector` type has been deprecated.

9.2 Tetration Apps

Tetration applications are specialized applications written by Tetration. Some of these applications run at a micro-batch interval of one minute, while others run at 1 hr intervals. To stop these application from running, users can simply deactivate them. Users are not allowed to modify the content of these applications.

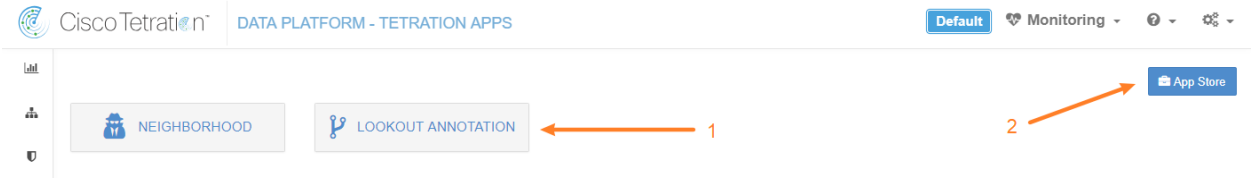


Fig. 9.2.1: Default Tetration Applications

1. Neighborhood and Lookout Annotation are enabled by default for each root scope. These can disabled from the the AppStore.
2. Tetration App Store. New apps can be enabled, or existing apps can be disabled.

9.2.1 Adding an App

To add an app:

1. Click on App Store
2. Then select the App you would like to enable by selecting + Create an Instance or Enable. On enabling an app or adding an instance, a user must accept the EULA.

9.2.2 Removing an app

Apps enabled on a root scope can be disabled from the AppStore. Instance based apps can be disabled by removing all instances of that app.

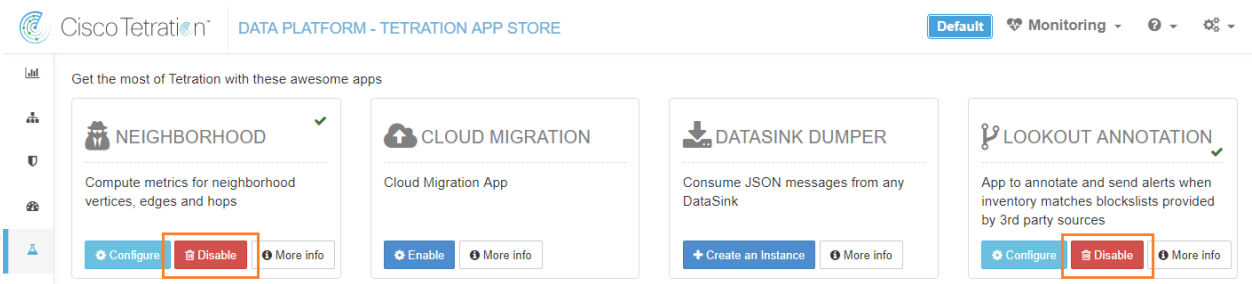


Fig. 9.2.2.1: Default enabled apps such as Neighborhood and Lookout Annotation can be disabled from the Tetration AppStore page.

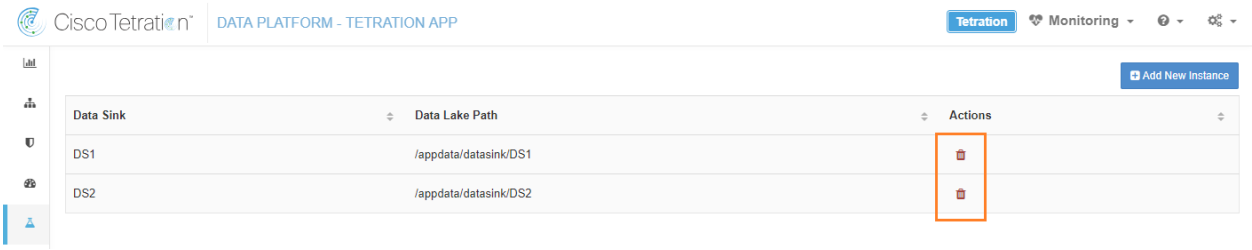


Fig. 9.2.2.2: Delete all instances of Data Sink Dumper app to remove the app.

9.2.3 Available Tetration Apps

9.2.3.1 Compliance

Compliance generates alerts on Policy Violations, specifically Escaped flows, as well as Rejected flows. Individual flow violations are rolled up to same alert type, Consumer Scope, Provider Scope, Provider Port and Protocol tuple when generating alerts.

Compliance alerts are generated every few minutes. To control the alert volume, consider snoozing individual alerts, or using summary alert options when configuring alerts.

Note: Only users with roles of *Enforce* or above will be able to configure compliance alerts. And only users with role of *Owner* (or *Site Admin*) will be able to configure alert publishers and notifiers.

Types of Compliance Alerts

Following section lists the types of Compliance alerts available in Tetration. Some of these alerts can be configured on either Live Analysis or Enforced Applications; one of these can only be configured on enforced applications.

- Escaped Flows: Triggered when any escaped flows within scope of application.
- Rejected Flows: Triggered when the number of rejected flows matches the condition. ie. such as when the number of rejected flows is over some specified threshold.

- Catch-all Flows: Triggered when any flows used the 'catch-all' enforcement policy rule. This type of alert is only available for Enforced applications. More details about this alert below (*Catch-All Alerts*)

There are 4 options for frequency/style of alerts.

- Individual → Enable: Default option. Sent out as soon as processed.
- Individual → Enable with Flows: Include five-tuple of every flow matching alert.
  - Up to 100 5-tuples will be included per alert, so many more alerts may be generated than the default option.
  - This option will not work for 'catch-all' alerts; those alerts will not include "constituent flows" regardless of this selection.
- Summary → Hourly: Receive a once-an-hour summary alert aggregating individual alerts.
  - Recommended to 'Disable' individual alerts with this option.
- Summary → Daily: Similar to hourly summary alert except over a full day.
  - Daily summary is sent after end of UTC day

Note: The *Enable with Flows* option for compliance alerts causes additional load on the pipeline that processes compliance alerts, and the alerts pipeline which handles snoozing and ignoring alerts. There is a UI warning about performance impact. If there are a large number of flows with compliance violations, this could cause a large number of compliance alerts which could slow down all alerts generally. Customer support dashboard *Hawkeye [Charts]* → Tetration Apps → Batch Processing Time → *compliance* and *alerts* can used to monitor the performance (Hawkeye Charts are available for on-prem Tetration; for TaaS, talk to Tetration Operations).

Example Alert

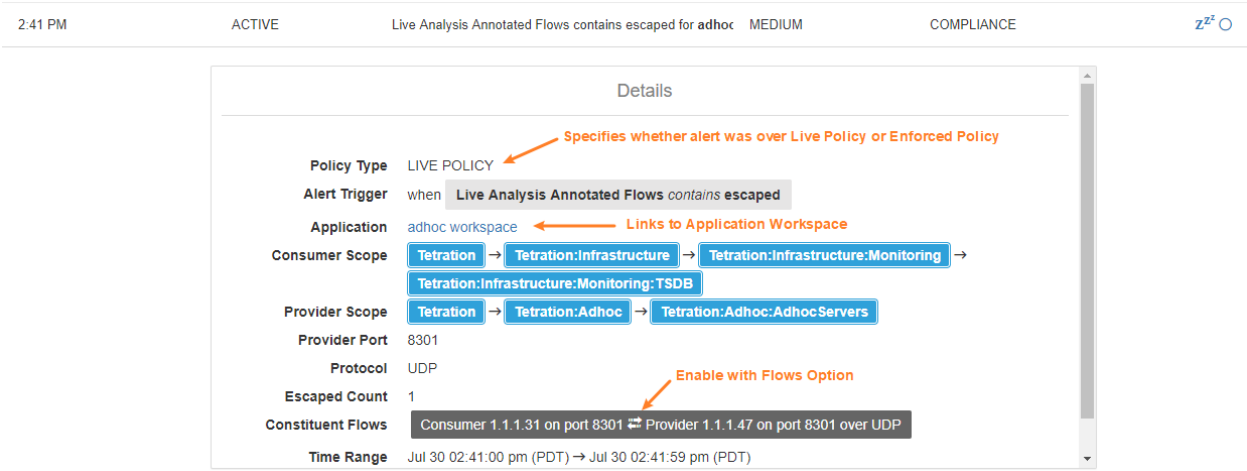


Fig. 9.2.3.1.1: Example of a Compliance Alert. Selecting an Application will link to the relevant Application UI page

Note: To avoid graph points being partially hidden on the y-axis, the time range selected on the Application page will have some buffer of time before and after the alert time range.

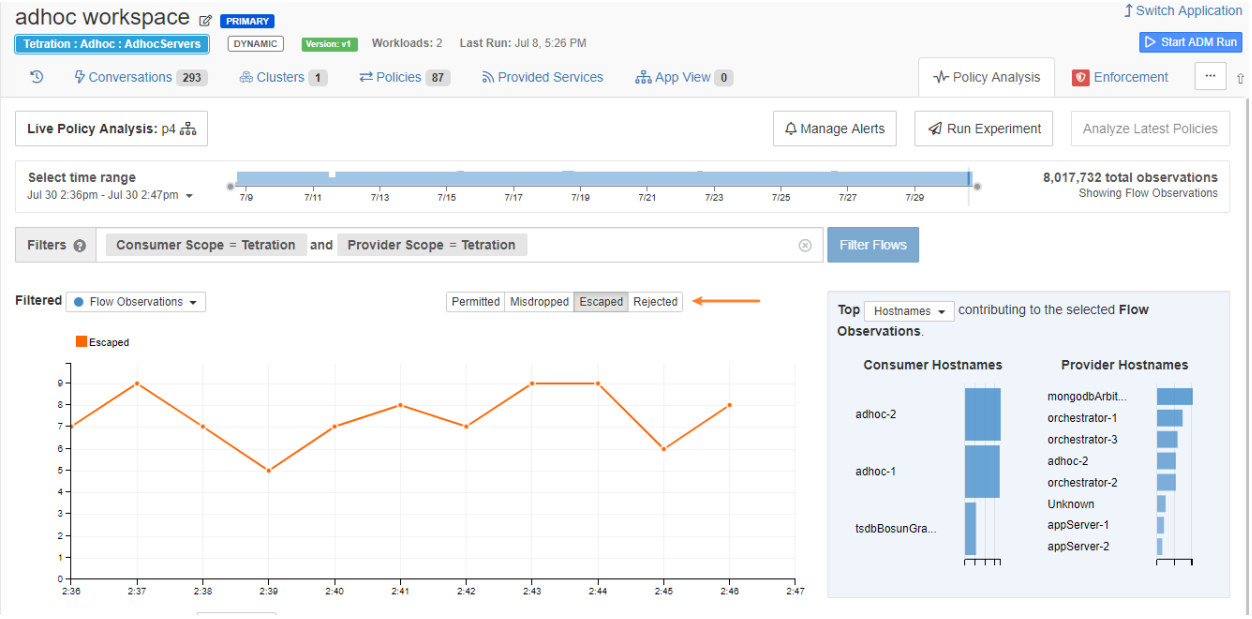


Fig. 9.2.3.1.2: Compliance Alert links to Application Workspace page. Note that the alert was for 2:41 pm, and the time range shown in the graph adds 5 min before and 5 min after that point.

Catch-All Alerts

Catch-all alerts are slightly different than the other compliance alerts in that they will tell you the specific application workspace where the catch-all rule was triggered. This means that the alert should only be configured at the top-scope of interest.

For example, suppose there is a scope Adhoc with two sub-scopes Adhoc-Server and Adhoc-Kafka. Suppose each of these three scopes has a primary application with enforcement enabled; they are named "Adhoc Application", "Adhoc Servers Application", and "Adhoc Kakfa Application" respectively. If the catch-all alert is configured on "Adhoc Application" it will generate alerts indicating if the catch-all rule was triggered for "Adhoc Application", "Adhoc Servers Application", or "Adhoc Kafka Applcation".

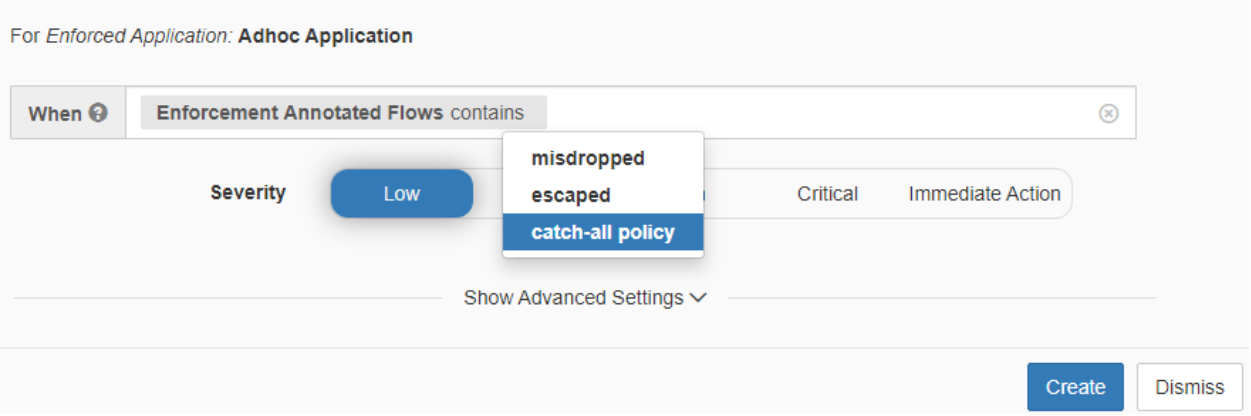


Fig. 9.2.3.1.3: Configuration of a catch-all compliance alert.

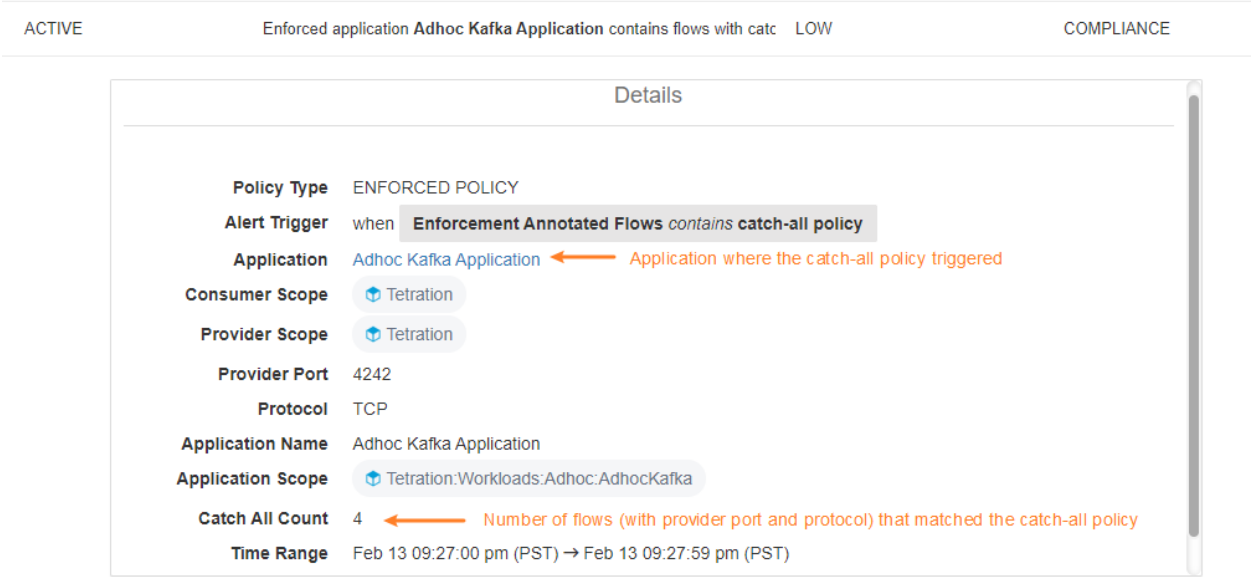


Fig. 9.2.3.1.4: Example of a generated alert from the above configuration.

Notes about the alert details:

- *Application* will link to the enforced applcation
  - Select *permitted* and *rejected* to see all flows within that application around the time of the alert
  - Filter the flows by adding filters based on details in the alert; these are not added to the selection by default when linked from the alert.
    - Provider Port
    - Protocol
    - Deciding Enforcement Policy Rank = Catch All
- Ignore *Consumer Scope* and *Provider Scope*: these will be the root scope of the flows triggering the alert, and are not meaningful.

Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* and *alert_details_json* fields are structured and will contain the following subfields for compliance alerts.

Field	Alert Type	Type	Explanation
internal_trigger	*all*	structured	Configuration query which triggered the alert
policy_type	*all*	string	Policy Type: ENFORCED_POLICY, LIVE_POLICY
application_scope_id	Catch-All	string	Scope of the application
application_id	Catch-All	string	Application id of the workspace where catch-all rule was hit
application_name	Catch-All	string	Application name for workspace where catch-all rule was hit
consumer_scope_ids	*all*	list[string]	List of consumer scope ids. This list contains scope ids down the scope hierarchy tree. Ignore this field for catch-all alerts (see prior note above).
consumer_scope_names	Escaped and Rejected	list[string]	List of scope names corresponding to the consumer_scope_ids
provider_scope_ids	*all*	list[string]	List of provider scope ids. This list contains scope ids down the scope hierarchy tree. Ignore this field for catch-all alerts (see prior note above).
provider_scope_names	Escaped and Rejected	list[string]	List of scope names corresponding to the provider_scope_ids
protocol	*all*	string	Protocol for all flows triggering this specific alert
provider_port	*all*	int	Provider port for all flows triggering this specific alert
escaped_count	Escaped	long	Count of escaped flows in time_range
rejected_count	Rejected	long	Count of rejected flows within the time_range
catch_all_count	Catch-All	long	Count of flows matching catch all policy in the specified application
time_range	*all*	list[long]	Contains two timestamps.
constituent_flows	*any*	list[struct]	Only included if configured with "Enabled with Flows"
summarized_alerts	*any*	int	Only included for summary alerts. The number of individual alerts which are being summarized over.

Structure of *constituent_flows*: A list containing structured data. The structured data is a map containing the following fields:

Field	Format	Description
consumer_address	string	Consumer address of flow
consumer_port	long	Consumer port of flow
provider_address	string	Provider ip address of flow
provider_port	long	Provider port of flow
protocol	string	Protocol of flow

Example of Summarized Escaped Alert

Note in the following example the *"summarized_alerts"* field, which is only present when summary alerts enabled.

```
{
      "severity": "LOW",
      "tenant_id": 0,
```

(continues on next page)

(continued from previous page)


```
      "alert_time": 1596704731226,
      "alert_text": "Enforcement Annotated Flows contains escaped for <application_
↪id:5f04b0b9755f024d4e36a279>",
      "alert_details_json": {
              "consumer_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "consumer_scope_names": ["Default"],
              "provider_scope_names": ["Default"],
              "provider_port": 443,
              "application_id": "5f04b0b9755f024d4e36a279",
              "escaped_count": 91,
              "provider_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "policy_type": "ENFORCED_POLICY",
              "protocol": "TCP",
              "internal_trigger": {
                      "datasource": "compliance",
                      "rules": {
                              "field": "policy_violations",
                              "type": "contains",
                              "value": "escaped"
                      },
                      "label": "Alert Trigger"
              },
              "time_range": [1596700740000, 1596704339999],
              "policy_category": ["ESCAPED"],
              "summarized_alerts": 38
      },
      "key_id": "1ef4a974-be89-31de-abe9-dc71cb0170ad",
      "alert_text_with_names": "Enforcement Annotated Flows contains escaped for␣
↪Enforced Application j1",
      "root_scope_id": "5efcfdf5497d4f474f1707c2",
      "type": "COMPLIANCE",
      "alert_details": "{\"consumer_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\
↪"consumer_scope_names\":[\"Default\"],\"provider_scope_names\":[\"Default\"],\
↪"provider_port\":443,\"application_id\":\"5f04b0b9755f024d4e36a279\",\"escaped_
↪count\":91,\"provider_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"policy_type\":\
↪"ENFORCED_POLICY\",\"protocol\":\"TCP\",\"internal_trigger\":{\"datasource\":\
↪"compliance\",\"rules\":{\"field\":\"policy_violations\",\"type\":\"contains\",\
↪"value\":\"escaped\"},\"label\":\"Alert Trigger\"},\"time_range\":[1596700740000,
↪1596704339999],\"policy_category\":[\"ESCAPED\"],\"summarized_alerts\":38}",
      "alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type=
↪'TETRATION_PARQUET', location_name='compliance', location_grain='HOURLY', root_
↪scope_id='5efcfdf5497d4f474f1707c2'}/
↪e2d056049f96e66424714c0d2d4c15daa564698d97b9d5a756f4022bc75dc160",
      "alert_conf_id": "5f15cca71a0c231ebd66ca3b",
      "event_time": 1596700740000
}
```


Example of Escaped Alert with Flows Enabled

Note the following example contains *'constituent_flows'*; this field will only show up if alert is configured with *"Enabled with Flows"*.


```
{
      "severity": "LOW",
      "tenant_id": 0,
```


(continues on next page)

(continued from previous page)


```
      "alert_time": 1596705758010,
      "alert_text": "Enforcement Annotated Flows contains escaped for <application_
↪id:5f04b0b9755f024d4e36a279>",
      "alert_details_json": {
              "consumer_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "consumer_scope_names": ["Default"],
              "provider_scope_names": ["Default"],
              "provider_port": 5660,
              "application_id": "5f04b0b9755f024d4e36a279",
              "constituent_flows": [{
                      "consumer_port": 17131,
                      "protocol": "TCP",
                      "consumer_address": "172.26.231.193",
                      "provider_address": "172.31.163.140",
                      "provider_port": 5660
              }],
              "escaped_count": 1,
              "provider_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "policy_type": "ENFORCED_POLICY",
              "protocol": "TCP",
              "internal_trigger": {
                      "datasource": "compliance",
                      "rules": {
                              "field": "policy_violations",
                              "type": "contains",
                              "value": "escaped"
                      },
                      "label": "Alert Trigger"
              },
              "time_range": [1596705480000, 1596705539999],
              "policy_category": ["ESCAPED"]
      },
      "key_id": "8f0cfcb5-f8c1-3130-a069-3721b7d50159",
      "alert_text_with_names": "Enforcement Annotated Flows contains escaped for␣
↪Enforced Application j1",
      "root_scope_id": "5efcfdf5497d4f474f1707c2",
      "type": "COMPLIANCE",
      "alert_details": "{\"consumer_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\
↪"consumer_scope_names\":[\"Default\"],\"provider_scope_names\":[\"Default\"],\
↪"provider_port\":5660,\"application_id\":\"5f04b0b9755f024d4e36a279\",\"constituent_
↪flows\":[{\"consumer_port\":17131,\"protocol\":\"TCP\",\"consumer_address\":\"172.
↪26.231.193\",\"provider_address\":\"172.31.163.140\",\"provider_port\":5660}],\
↪"escaped_count\":1,\"provider_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"policy_
↪type\":\"ENFORCED_POLICY\",\"protocol\":\"TCP\",\"internal_trigger\":{\"datasource\
↪":\"compliance\",\"rules\":{\"field\":\"policy_violations\",\"type\":\"contains\",\
↪"value\":\"escaped\"},\"label\":\"Alert Trigger\"},\"time_range\":[1596705480000,
↪1596705539999],\"policy_category\":[\"ESCAPED\"]}",
      "alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type=
↪'TETRATION', location_name='compliance', location_grain='MIN', root_scope_id=
↪'5efcfdf5497d4f474f1707c2'}/
↪69a5b11a25fc8a804f9ac9ddf8e069bfea82d68b0f1d6ff0f38e075c614cc9c0",
      "alert_conf_id": "5f15cca71a0c231ebd66ca3b",
      "event_time": 1596705480000
}
```

Example of Catch-All Alert

Following is an example of hourly summary alert for catch-all alert.


```
{
      "severity": "LOW",
      "tenant_id": 0,
      "alert_time": 1597093502738,
      "alert_text": "Enforced application <application_id:5f04b0b9755f024d4e36a279>␣
↪contains flows with catch-all policy",
      "alert_details_json": {
              "provider_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "policy_type": "ENFORCED_POLICY",
              "consumer_scope_ids": ["5efcfdf5497d4f474f1707c2"],
              "protocol": "ICMP",
              "application_name": "j1",
              "internal_trigger": {
                      "datasource": "compliance",
                      "rules": {
                              "field": "policy_violations",
                              "type": "contains",
                              "value": "catch-all policy"
                      },
                      "label": "Alert Trigger"
              },
              "time_range": [1597090020000, 1597092959999],
              "application_scope_id": "5efcfdf5497d4f474f1707c2",
              "provider_port": 0,
              "summarized_alerts": 2,
              "application_id": "5f04b0b9755f024d4e36a279",
              "catch_all_count": 3
      },
      "key_id": "6e4aa157-0eb4-39f7-935a-a9a90eedda93",
      "alert_text_with_names": "Enforced application j1 contains flows with catch-all␣
↪policy",
      "root_scope_id": "5efcfdf5497d4f474f1707c2",
      "type": "COMPLIANCE",
      "alert_details": "{\"provider_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\
↪"policy_type\":\"ENFORCED_POLICY\",\"consumer_scope_ids\":[\
↪"5efcfdf5497d4f474f1707c2\"],\"protocol\":\"ICMP\",\"application_name\":\"j1\",\
↪"internal_trigger\":{\"datasource\":\"compliance\",\"rules\":{\"field\":\"policy_
↪violations\",\"type\":\"contains\",\"value\":\"catch-all policy\"},\"label\":\
↪"Alert Trigger\"},\"time_range\":[1597090020000,1597092959999],\"application_scope_
↪id\":\"5efcfdf5497d4f474f1707c2\",\"provider_port\":0,\"summarized_alerts\":2,\
↪"application_id\":\"5f04b0b9755f024d4e36a279\",\"catch_all_count\":3}",
      "alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type=
↪'TETRATION_PARQUET', location_name='compliance', location_grain='HOURLY', root_
↪scope_id='5efcfdf5497d4f474f1707c2'}/
↪dde1d2c9bb4d957da458dc21fc42849bdf0c02474928c1b0ec7b905c08cbfa35",
      "alert_conf_id": "5f0f8c941a0c234ed0ba0b86",
      "event_time": 1597090020000
}
```

Compliance Walkthrough

The following sections show a step-by-step walkthrough from configuring Compliance alerts, through viewing the generated alerts and managing the alert volume.

Prerequisites

Compliance will only work on a Primary Workspaces with Live Analysis or Enforcement enabled. See *Navigating to Applications* for information about Application Workspaces and how to create them.

For the purposes of Compliance, the following images indicate the relevant points of an Application workspace.

1. Primary Workspace: Only primary workspaces can be used for compliance
2. Enforcement Analysis page. See *Enforcement* for information about these pages.
3. Button to publish latest policies: "Enforce Policies"
4. Currently published/enforced policy: If this shows disabled or no policy, then Compliance will not generate any alerts.

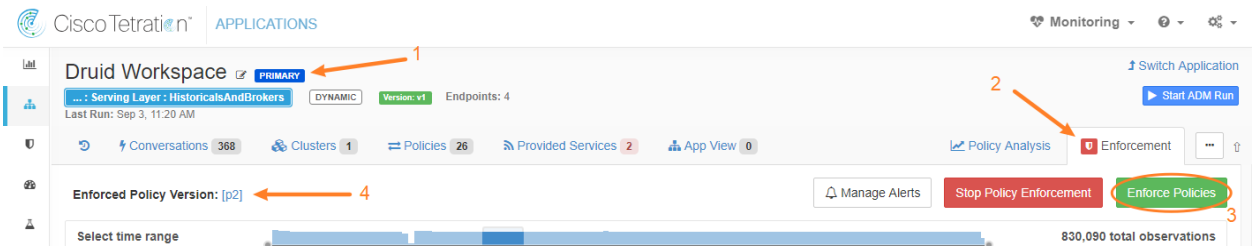


Fig. 9.2.3.1.5: Enforcement page of an Application

For Live Analysis Alerts, the relevant points are similar, except that the Policy Anlysis tab should be used.

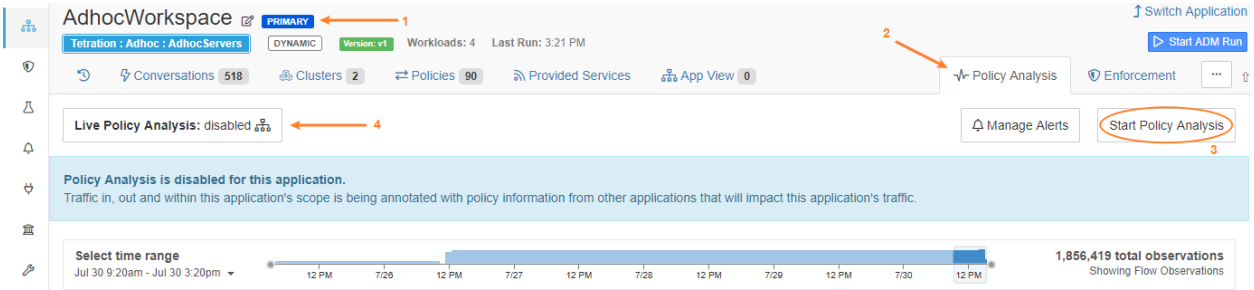


Fig. 9.2.3.1.6: Live Analysis page of an Application. Live Analysis has not been enabled yet in this example, and no alerts would be generated.

Note: If policy type is "Enforced" for a compliance instance, and Enforcement becomes disabled, Compliance will stop generating alerts for the enforced policy. See Applications → Policy Enforcement *Enforcement* for information on enabling and disabling Enforcement. Similarly, if Live Policy Analysis is 'disabled', then no alerts on the live analysis policy will be generated.

Enabling Compliance App

As of Tetration 3.0, Compliance App is enabled implicitly when configuring alerts, and will no longer show in UI.

Configuring Compliance Alerts

Compliance Alerts can be configured from an Application Workspace page from either the Enforcement tab or Policy Analysis tab, or from the Alert Configuration Page, via the alert configuration modal (see *Alert Configuration Modal* for general information about the modal). From this modal alert configurations for the compliance instance can be added.

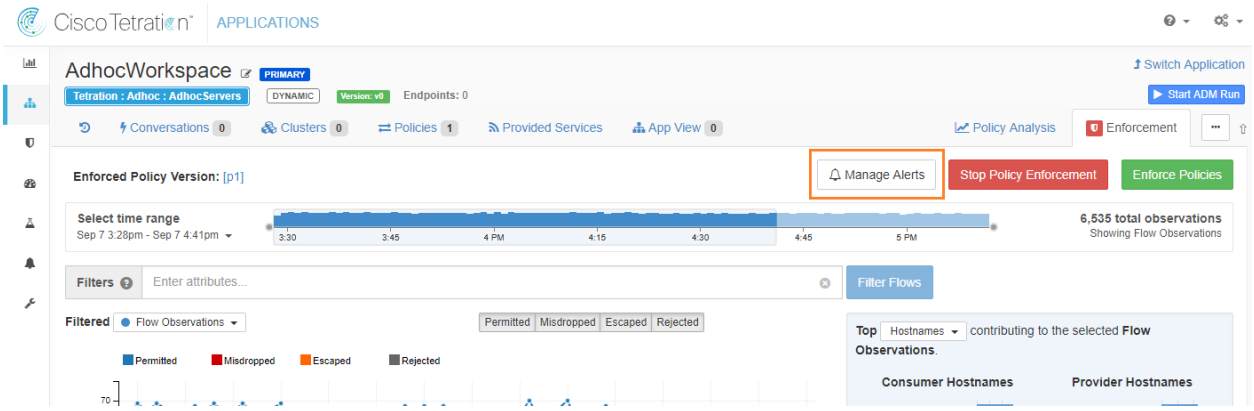


Fig. 9.2.3.1.7: Access to manage alerts modal from enforced workspace.

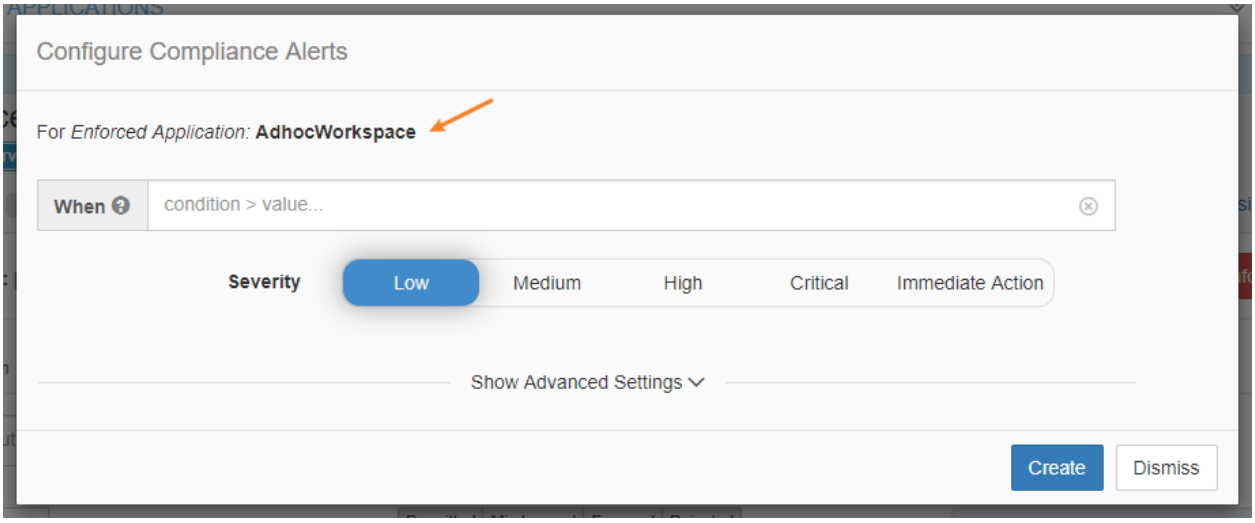


Fig. 9.2.3.1.8: Alert configuration modal for Compliance from Enforcement tab. Note that the modal will say "For Enforced Application:"

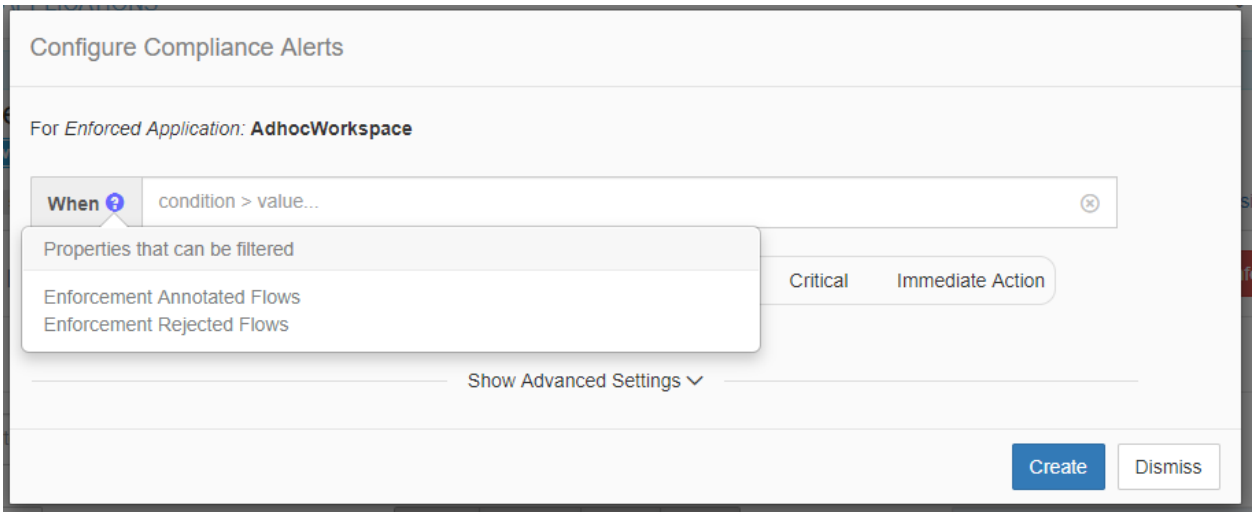


Fig. 9.2.3.1.9: Hovering on icon will show the possible alert trigger options. For compliance, alerts can be generated when flows are escaped, or when the number of rejected flows exceeds some threshold.

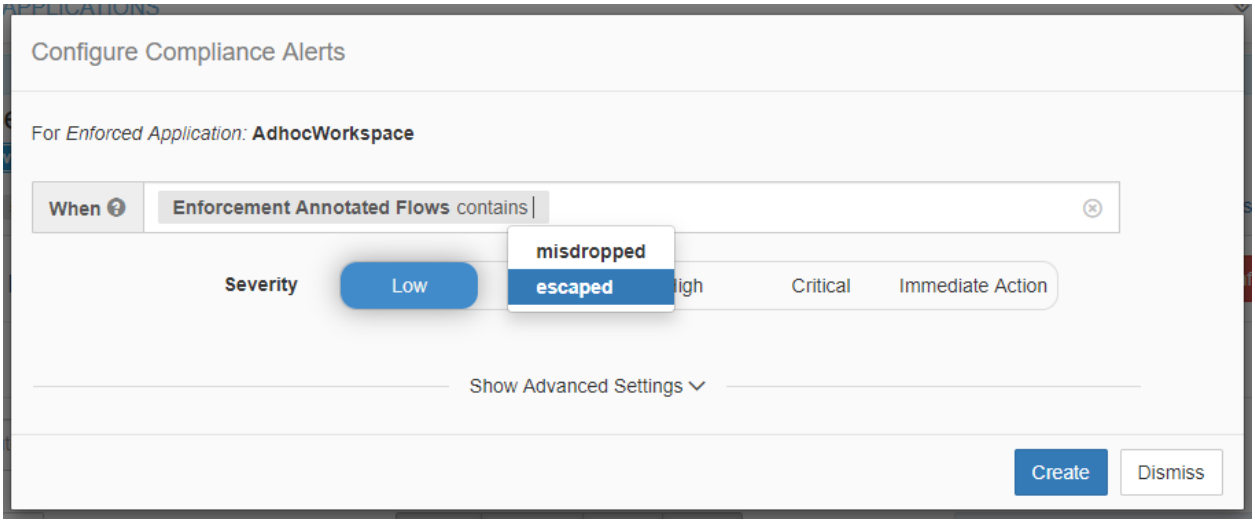


Fig. 9.2.3.1.10: Typing in the query box will also show valid options for configuring alerts.

Note: Multiple query conditions (using "and" or "or") is not currently allowed for compliance alerts. When trying to create alerts like this, you will receive an error message.

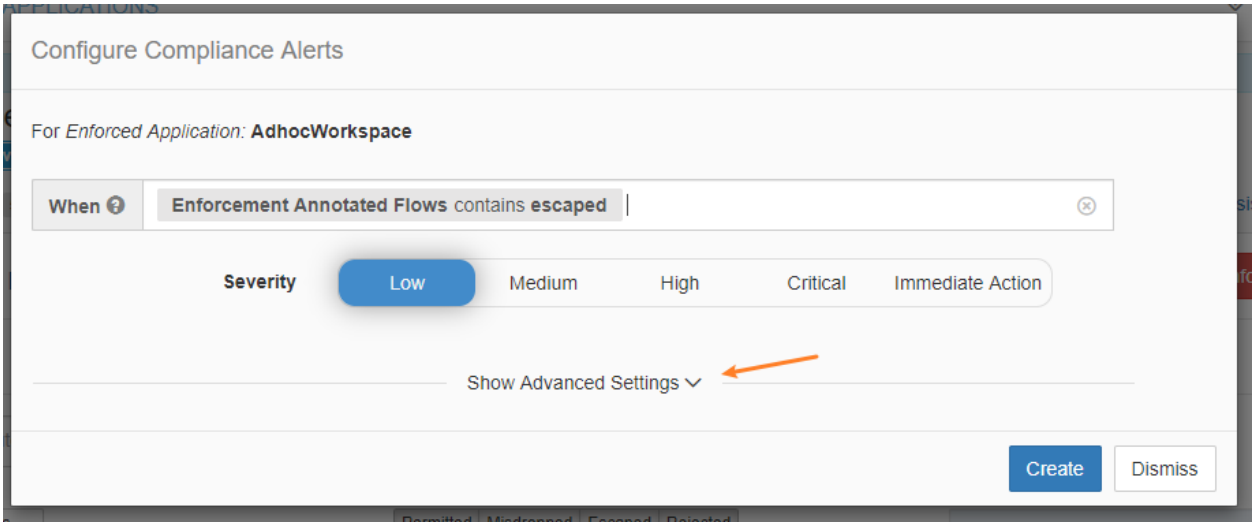


Fig. 9.2.3.1.11: To choose Summary Alert options, click "Show Advanced Settings" to expand the modal.

The expanded modal shows 2 addtional options:

1. Whether to send individual (minute) level alerts. Options are "Enable", "Enable with Flow Details" (for including constituent flows in the alert details), or "Disable" (for choosing summary alerts).
2. Summary alert options: None, Hourly, or Daily

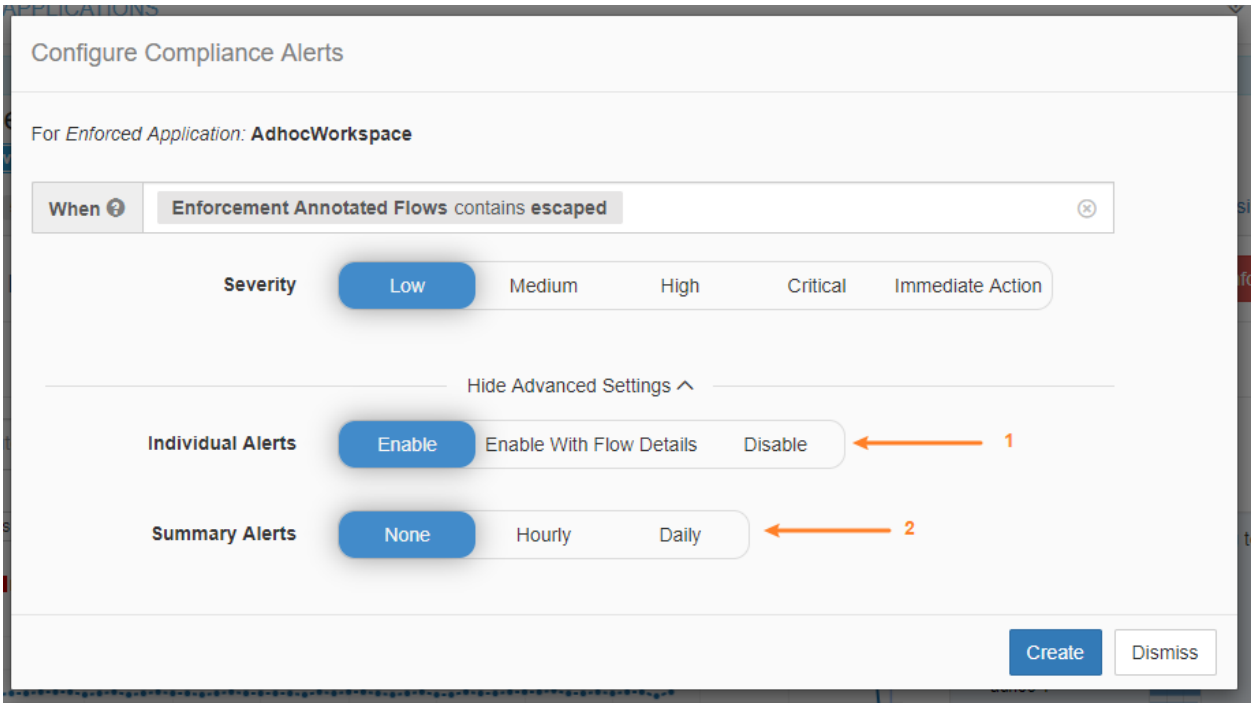


Fig. 9.2.3.1.12: Expanded modal options

Note: Individual alerts can be enabled while also selecting a summary alert. All alerts will be sent at the same severity

level, however.

Note: Chooosing to disable individual alerts, while forgetting to choose a summary alert, will result in an error message.

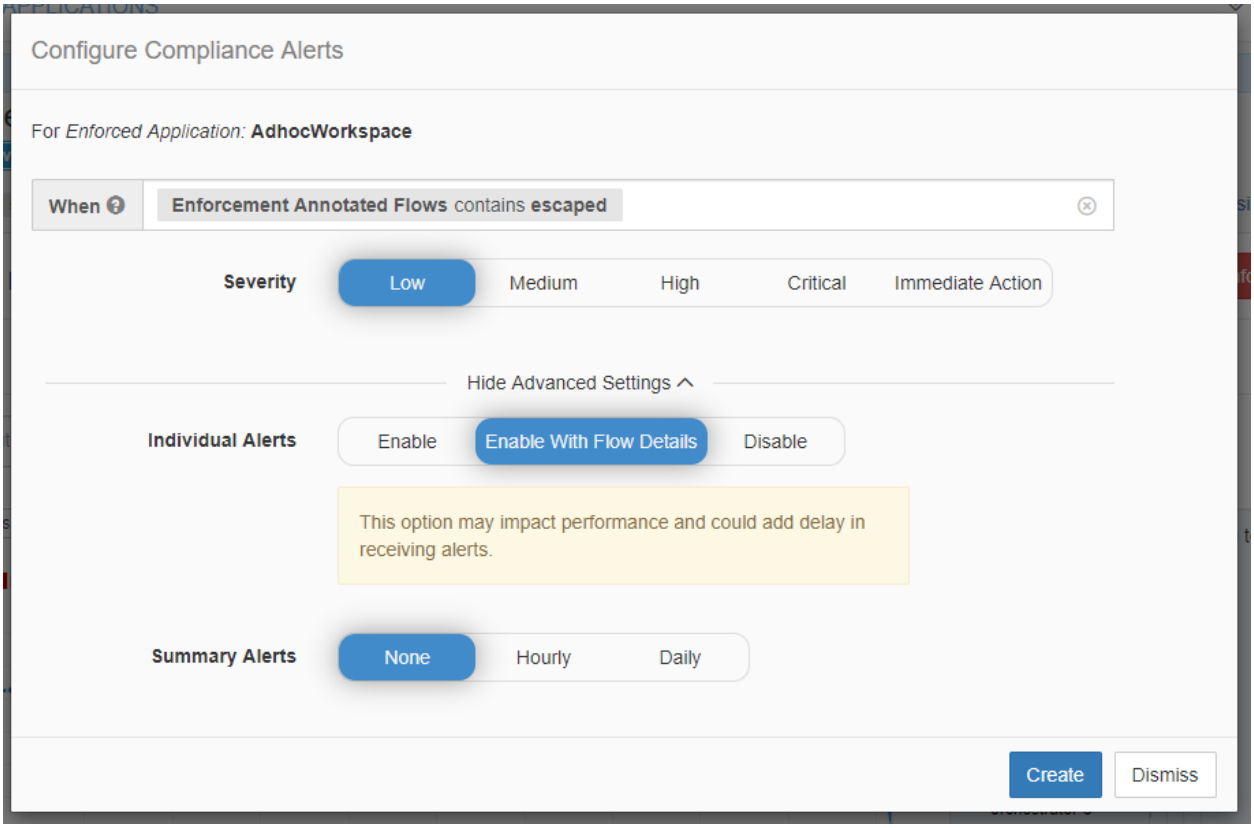


Fig. 9.2.3.1.13: Choosing "Enable with Flow Details". This will generate alerts with 'constituent flows' embedded in the alert details. Warning: this could substantially increase the time to process alerts (if there are many policy violations).

Note: "Enable with Flow Details": up to 100 constituent flows will be included; if there are more flows, the alert will be broken into multiple alerts. Summary Alerts will not include 'consituent flows' in the alert details.

Note: clicking "Create" will create an alert configuration **and** close the modal.

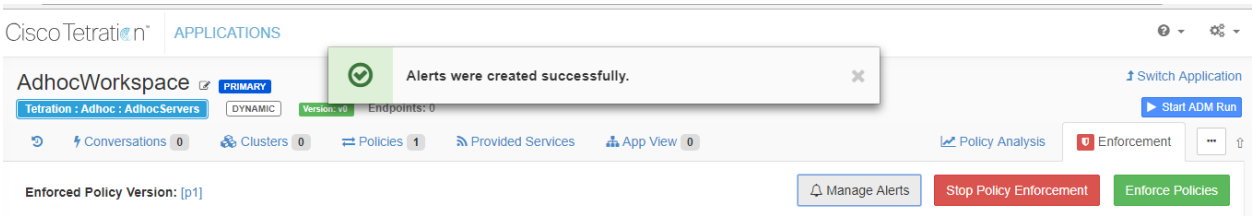


Fig. 9.2.3.1.14: Message displayed after successful alert creation.

Clicking the "🔔 Manage Alerts" will re-open the modal so that another alert can be created.

1. List of existing configured alerts. These can be quickly deleted by clicking the trashcan icon. Note: This will show both Live Analysis and Enforcement Alerts.
2. Here we select the 'Manage Alerts' button from the 'Policy Analysis' tab. So the modal shows we are configuring an alert on the Live Analysis policy.
3. Configuring another alert. Mulitple alerts with the same alert trigger condition (for the same policy type and workspace) can not be created. In this example, an alert on rejected count is created as an hourly summary alert with Low severity.
4. "Create" will create this additional alert, and close the modal.

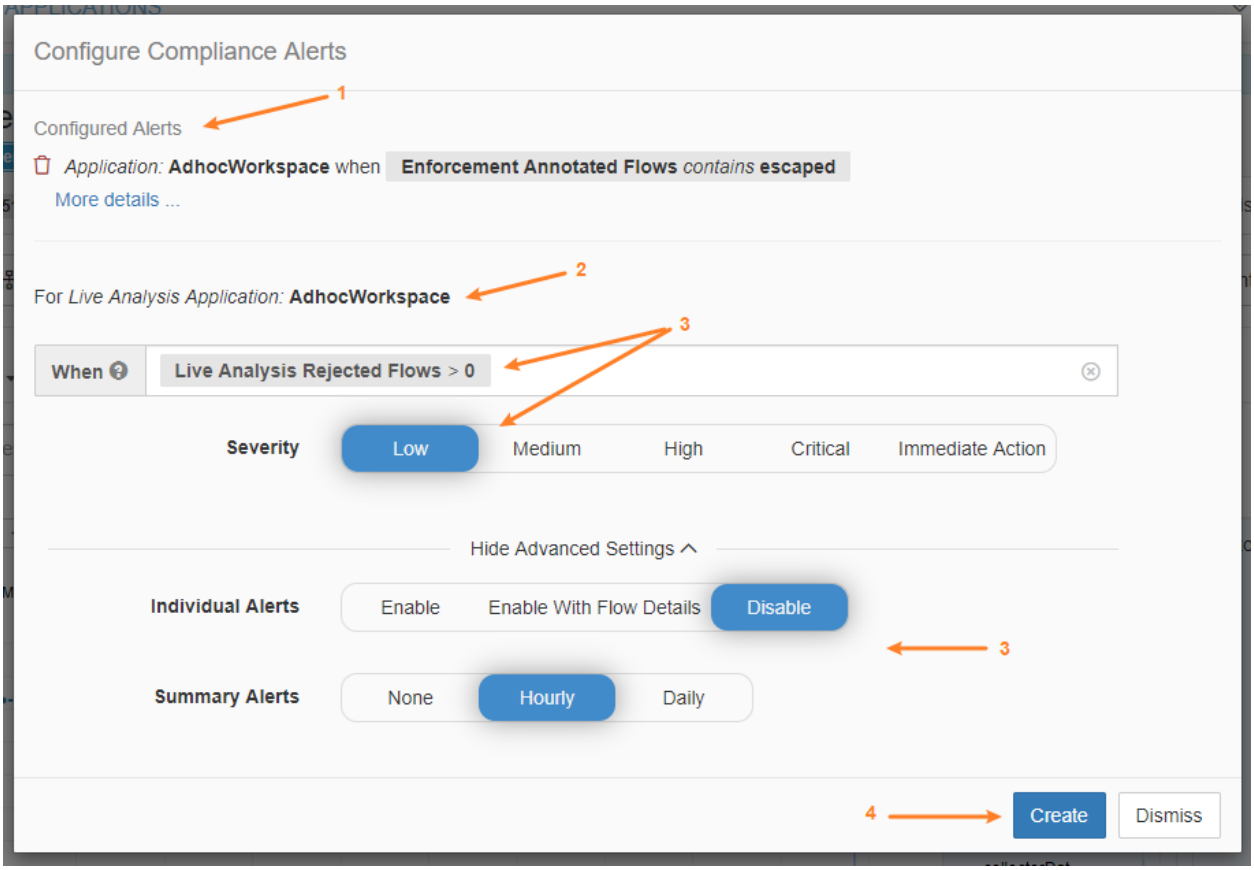


Fig. 9.2.3.1.15: If there are existing alerts these will be displayed at the top. Clicking the trashcan icon will delete that alert. The trigger condition will specify if the alert is on Live Analysis policy or Enforced policy.

Viewing Alert Configurations

All alert configurations created using the *Alert Configuration Model* can be viewed from the Alert Configuration Page. From the left navigation menu go to `Alerts` then `Configuration`. The upper half of this page depicts alert sources and connections to publishers and notifiers. The lower half of this page lists configured alerts.

1. Clicking an alert source will filter the list of all configured alerts just to alert configurations matching that type.
2. Selecting 🔔 will open the alert configuration modal to configure additional alerts. (*Alert Configuration Modal*) Note: When configuring from this page, the policy type must be selected first.

3. Selecting alert publishers (Only available to *Owner* and *Site Admin* users.) See next section
4. List of configured alerts. Filter has been populated by clicking 'Compliance' above.
5. Configured compliance alerts. Clicking will expand with details.

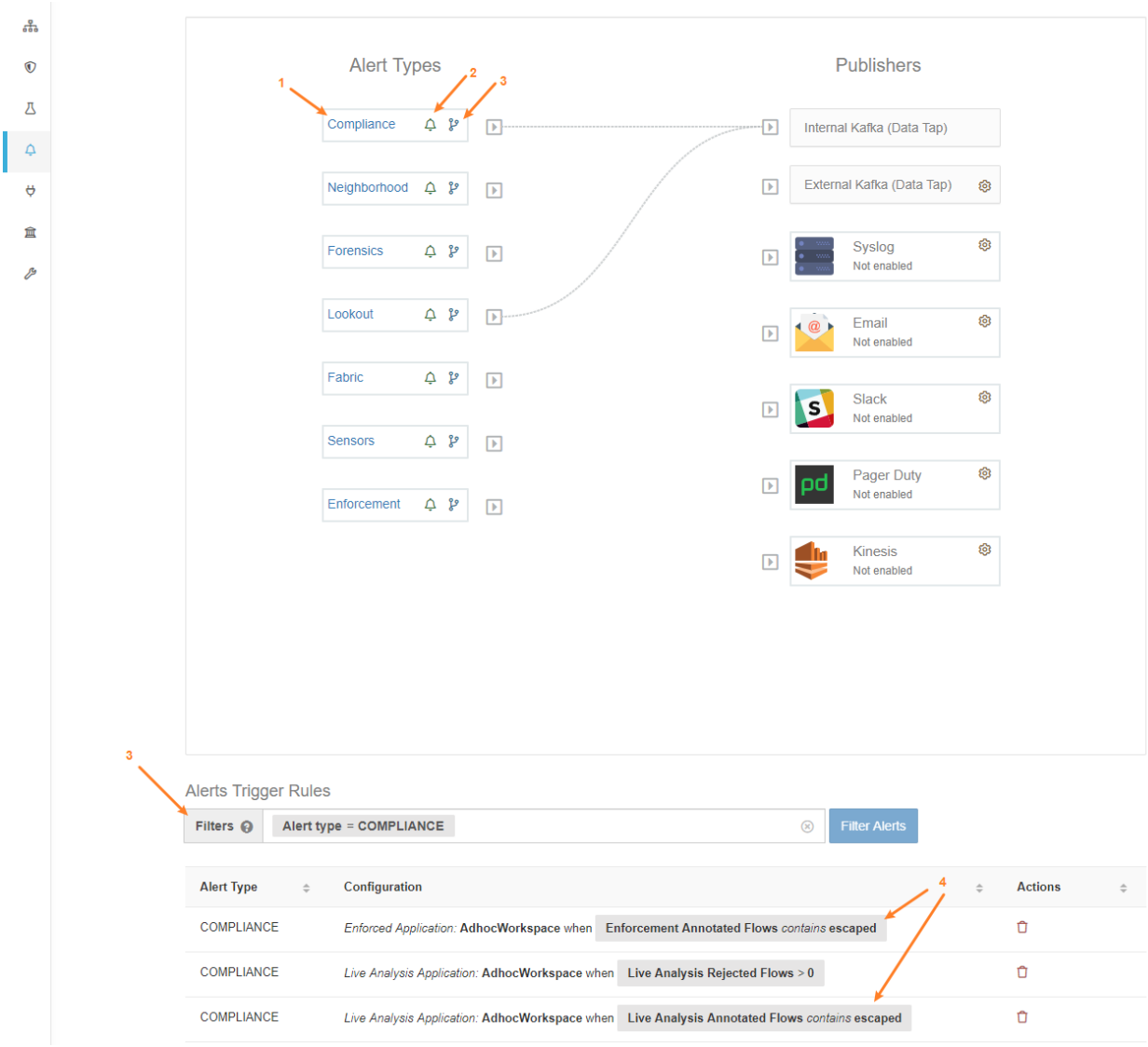


Fig. 9.2.3.1.16: Alert Configuration page with Compliance alerts selected.

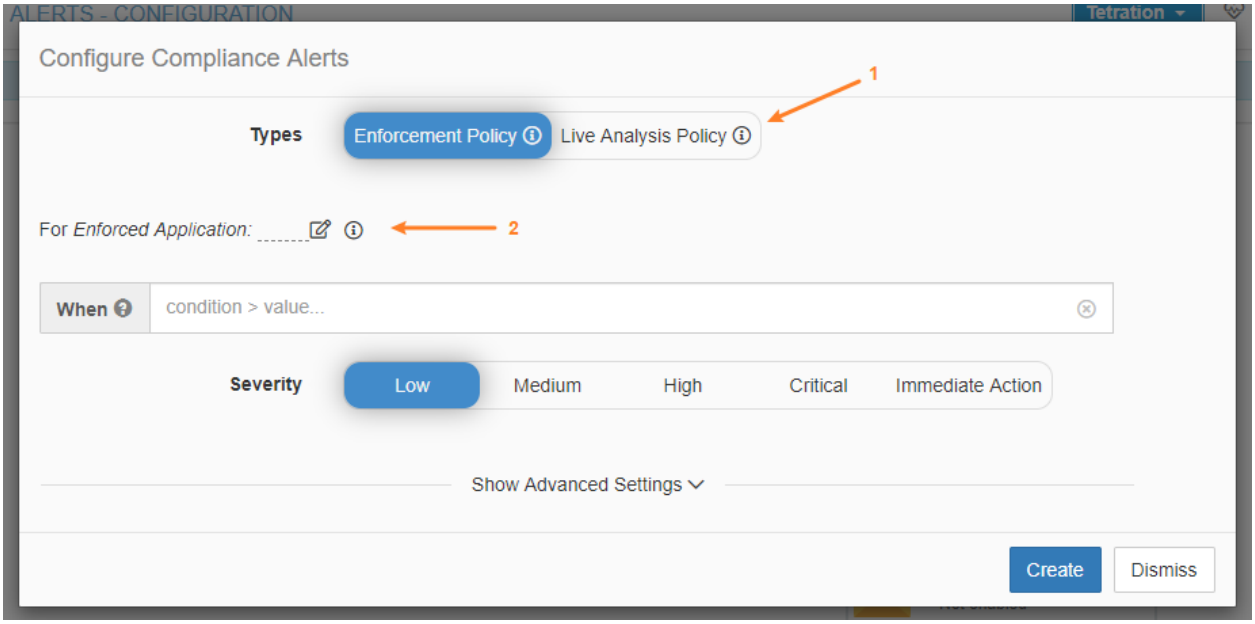


Fig. 9.2.3.1.17: Modal on the Alert Configuration page requires selecting the policy type. (1) Select the policy type first before selecting the application (2).

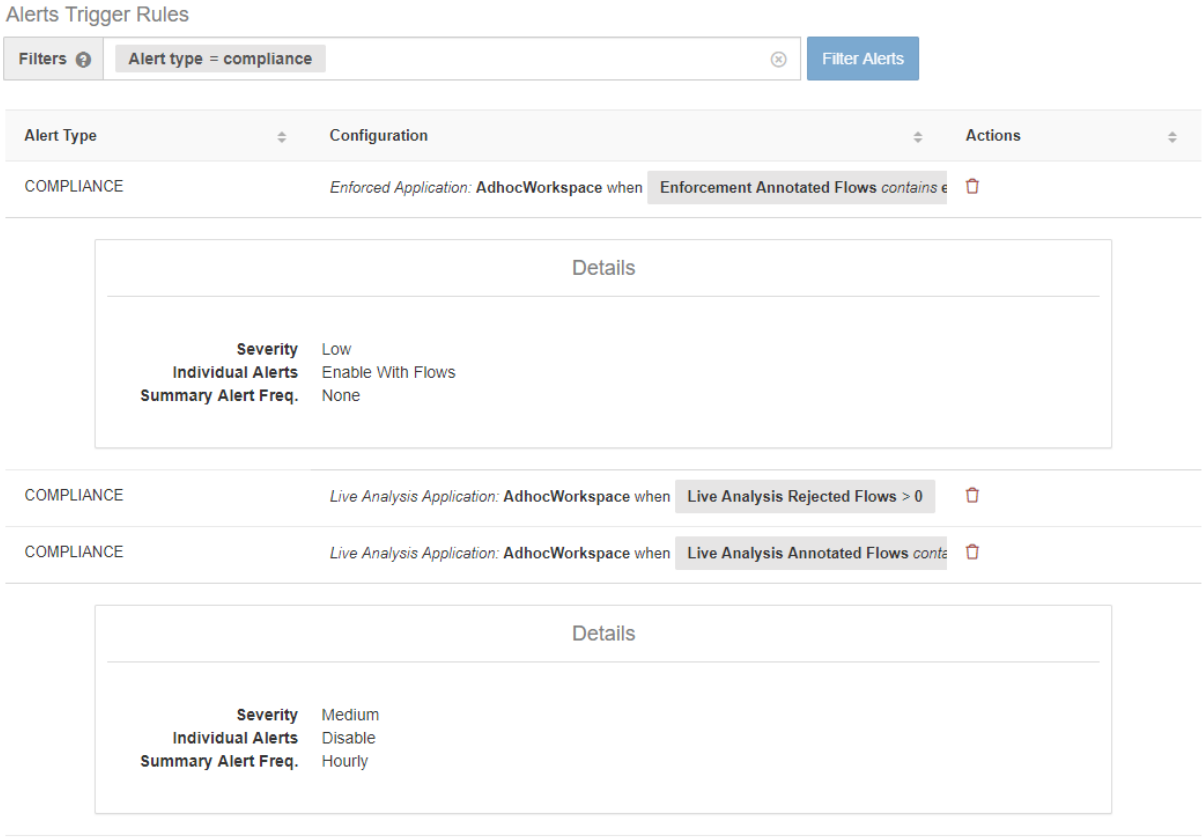


Fig. 9.2.3.1.18: Clicking each alert configuration will expand with details.

Note: Expanding the configured alerts on the alert configuration page is currently the only way to see whether individual (minute) level alerts or summary alerts will be generated.

Integration with the Alert App

Alert publishers must be explicitly specified by a user with role of *Owner* or *Site Admin*. Go to *Alerts* > *Configuration* to view or configure publishers and notifiers.

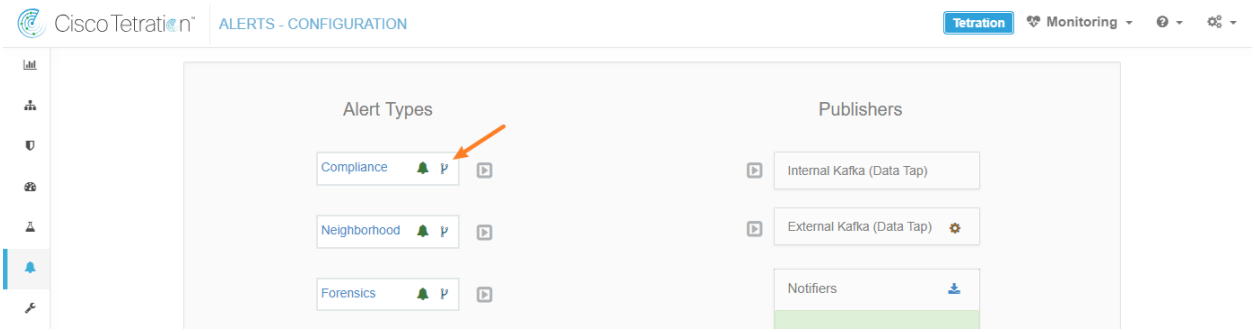


Fig. 9.2.3.1.19: Root scope owners will be able to configure publishers.

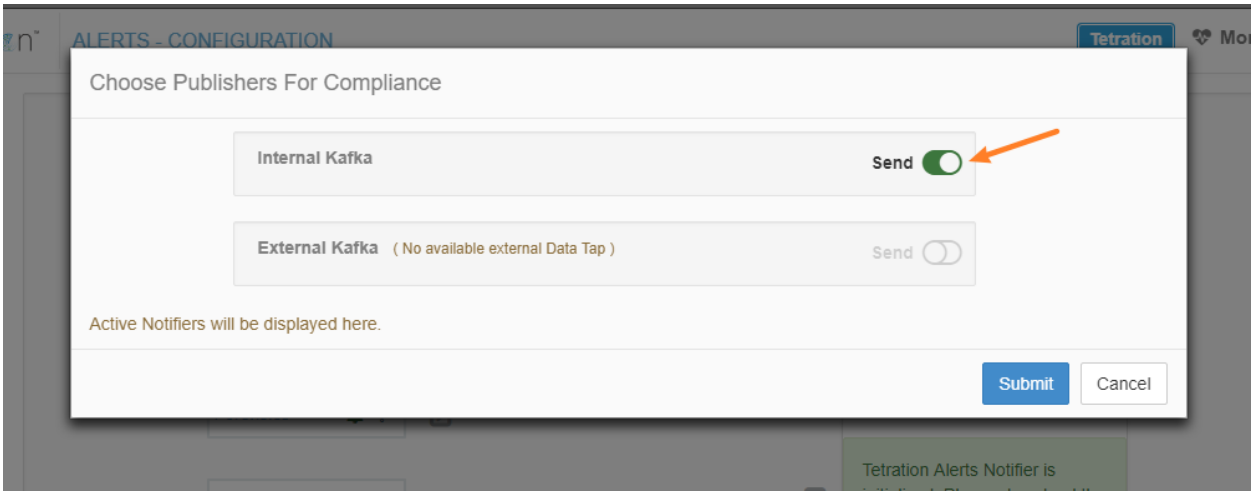


Fig. 9.2.3.1.20: Select publishers and notifiers (if applicable). Here, owner has toggled compliance alerts to be sent to MDT.

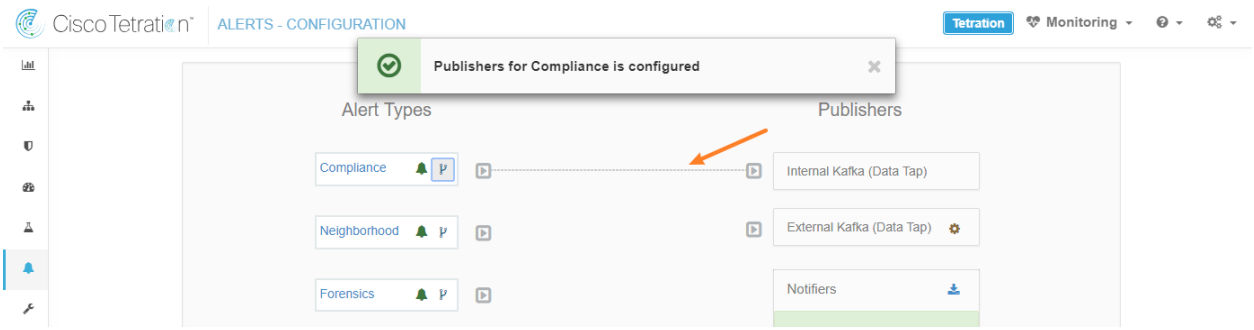


Fig. 9.2.3.1.21: Once a publisher or notifier is enabled, a line will depict the connection from left side (alert sources) to right side (publishers and notifiers).

Viewing Alerts

For general information about viewing alerts in the UI, see *Current Alerts*. The alerts page is accessed from *Alerts* > *Current Alerts*

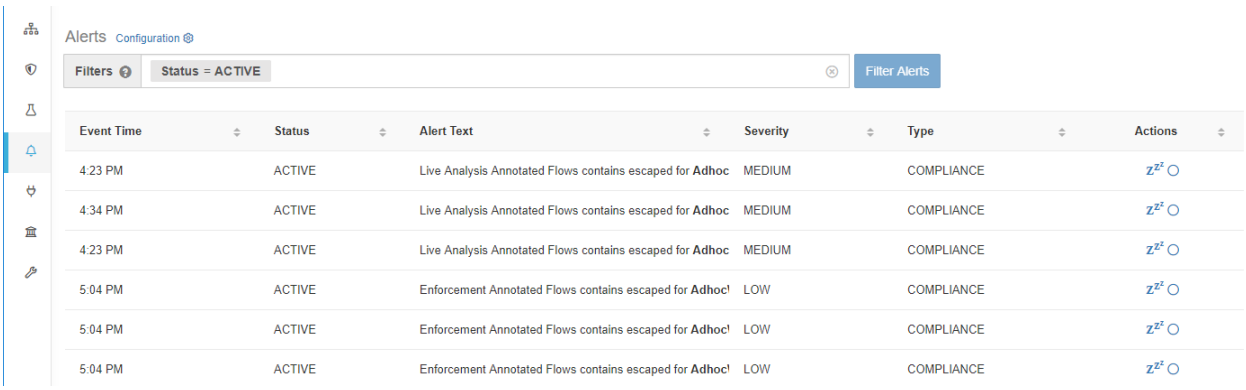


Fig. 9.2.3.1.22: List of compliance alerts that were generated and sent to a Data Tap.

Clicking any alert will expand the alert to show the details. Below are examples of alerts previously configured (shown in *Viewing Alert Configurations*).

In the figures below:

1. The policy type
2. The application workspace where compliance is enabled. This will link to the Policy Analysis or Enforcement page with filtered flows. See *Types of Compliance Alerts* for more info.
3. The count of flows where this alert was triggered.
4. If this is a summary alert, the alert will show a count of the number of individual (minute) alerts that were aggregated into this alert.
5. The time range of the alert. For summary alerts, this is the maximum encapsulating time range of the individual alerts. (ie. minimum time of all alerts that were summarized to maximum time of all alerts that were summarized).

Note: As with all generated alerts, "Event Time" is the beginning of the time range of the alert. This means that for summary alerts it is the beginning of the event range, rather than the end of the event range. It may help to think of

Event Time as first occurence.

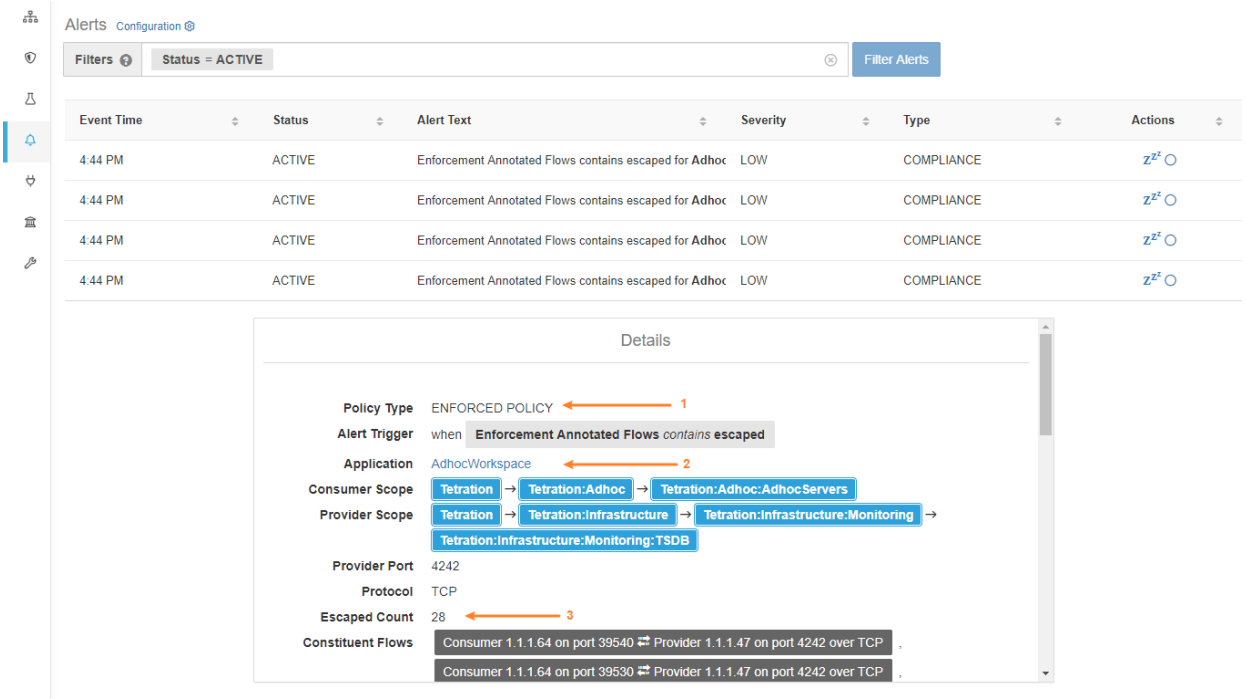


Fig. 9.2.3.1.23: Example of individual alert with flow details on escaped flows for enforcement policy. Note: the escaped count is the number of escaped flows. The 'Constituent Flows' will show the detailed list of flows; this will only be added if the alert was configured with flow details.

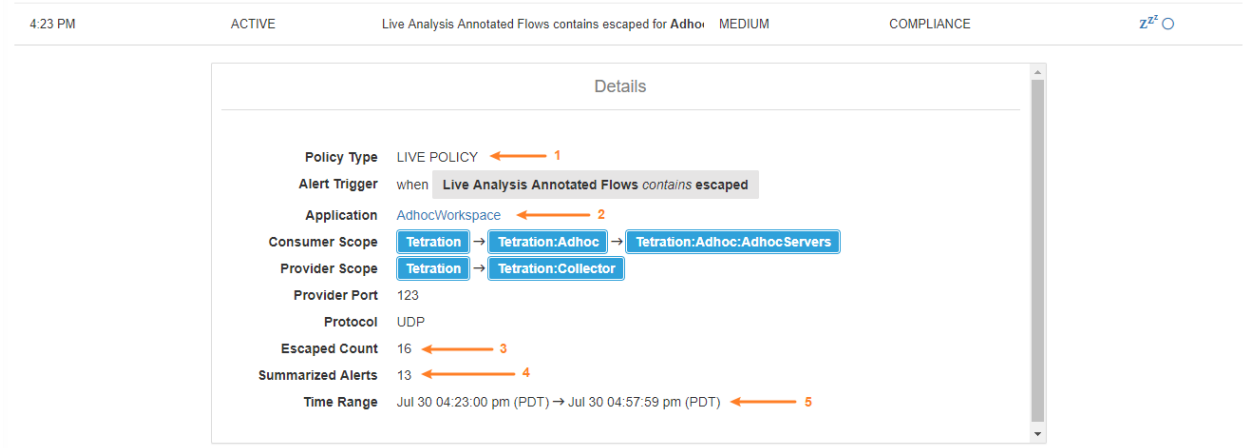
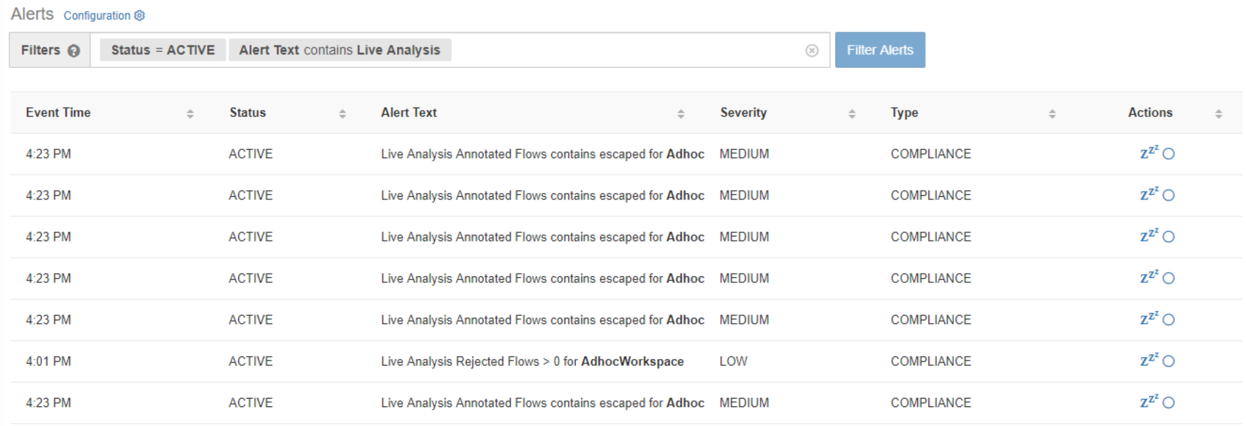


Fig. 9.2.3.1.24: Example of hourly alert on escaped flows for live analysis policy. Note: the escaped count is the total number of escaped flows seen across all the individual (minute) alerts, while the 'Summarized Alerts' count refers to the number of minute level alerts that were summarized.



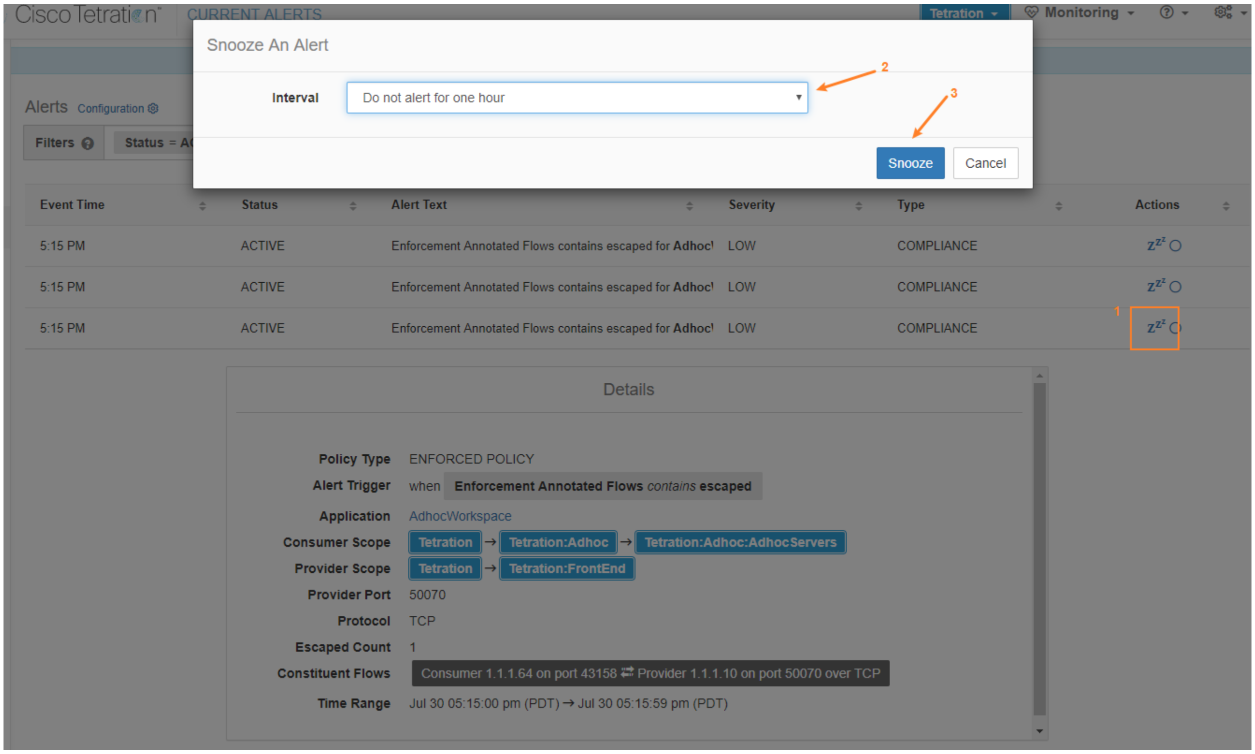
Status = **ACTIVE** **Alert Text** contains **Live Analysis**

Event Time	Status	Alert Text	Severity	Type	Actions
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○
4:01 PM	ACTIVE	Live Analysis Rejected Flows > 0 for **AdhocWorkspace**	LOW	COMPLIANCE	zzz ○
4:23 PM	ACTIVE	Live Analysis Annotated Flows contains escaped for **Adhoc**	MEDIUM	COMPLIANCE	zzz ○

Fig. 9.2.3.1.25: It is also possible to filter the list of alerts by `Alert text contains`. This may be helpful in finding more specific alerts.

Snoozing Alerts

See *Snoozing Alerts* for a general discussion about snoozing and ignoring specific alerts.



Snooze An Alert

Interval Do not alert for one hour

Event Time	Status	Alert Text	Severity	Type	Actions
5:15 PM	ACTIVE	Enforcement Annotated Flows contains escaped for Adhoc	LOW	COMPLIANCE	zzz ○
5:15 PM	ACTIVE	Enforcement Annotated Flows contains escaped for Adhoc	LOW	COMPLIANCE	zzz ○
5:15 PM	ACTIVE	Enforcement Annotated Flows contains escaped for Adhoc	LOW	COMPLIANCE	zzz ○

Details

Policy Type ENFORCED POLICY
Alert Trigger when **Enforcement Annotated Flows** *contains* **escaped**
Application AdhocWorkspace
Consumer Scope Tetration → Tetration:Adhoc → Tetration:Adhoc:AdhocServers
Provider Scope Tetration → Tetration:FrontEnd
Provider Port 50070
Protocol TCP
Escaped Count 1
Constituent Flows Consumer 1.1.1.64 on port 43158 ⇄ Provider 1.1.1.10 on port 50070 over TCP
Time Range Jul 30 05:15:00 pm (PDT) → Jul 30 05:15:59 pm (PDT)

Fig. 9.2.3.1.26: To snooze an alert, click on the "Zzz" icon for the alert (1). Then choose the snooze interval (2), and 'Snooze' (3). All alerts for the interval selected will be suppressed.

After snoozing an alert, all alerts with the same Key Id will not be shown by default (default Alerts page filtering is to show `Status = ACTIVE`)

Example of snoozed alert, and Key Id fields.

1. Filtering for only snoozed alerts
2. These fields form the Key Id for compliance alerts.
3. Click "zzZ" icon to Un-snooze a previously snoozed alert

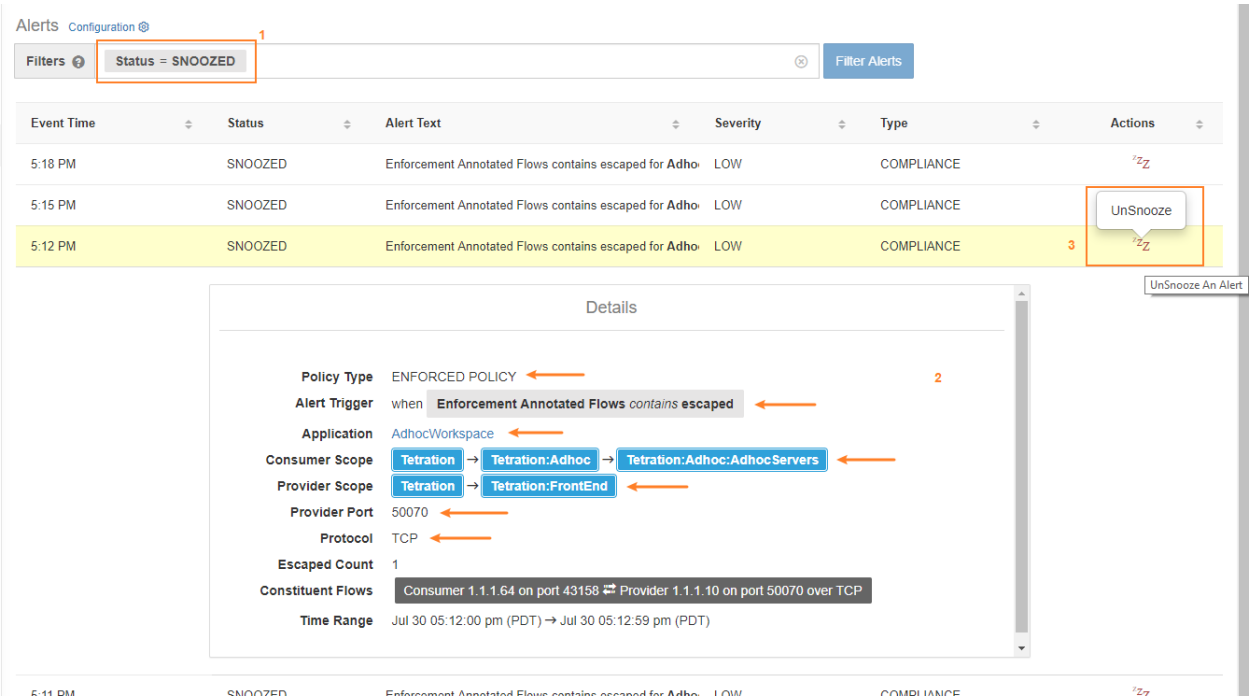


Fig. 9.2.3.1.27: To see the snoozed alert(s) change the filter to `Status = SNOOZED`

To Remove Compliance

Delete all alert configurations on a workspace, and/or delete the workspace, to remove compliance.

9.2.3.2 Neighborhood

The neighborhood app allows a user to explore aggregated flow data by Geo location, or in terms of Neighborhoods around a node such as edges and paths between nodes. The neighborhood app also allows several types of alerts to be set up, such as geo related alerts, and node, edge, and hop based alerts.

Note:

***Prerequisites*:**

1. Create a scope hierarchy or run ADM and enable live analysis to actually see a neighborhood graph. Neighborhood must have subscopes, or filters** or clusters** annotated on the flows in order to display a graph. ** Filters and clusters must be part of a primary workspace with live analysis or enforcement enabled, and must be part of the scope of that workspace.
2. Neighborhood geo must have geo data pack upload via threat intelligence.
3. For neighborhood geo, user's WebBrowser must also have access to the Mapbox API's for map rendering.

Accessing

Neighborhood can be directly accessed under `Visibility` in left menu. Or from `Data Platform` → `Tetration Apps`. Enabling/disabling Neighborhood can only be done on `Tetration Apps` page.

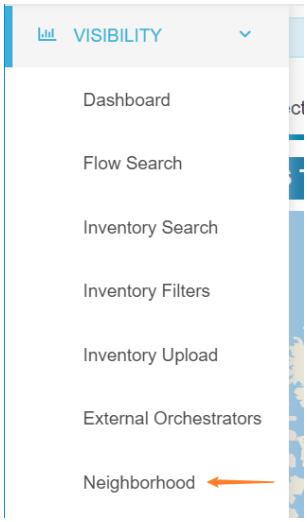


Fig. 9.2.3.2.1: Neighborhood feature can be accessed under 'Visibility' menu.

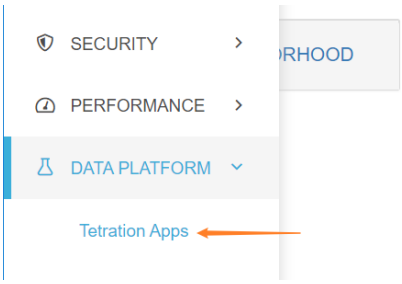


Fig. 9.2.3.2.2: Neighborhood feature can be accesssed from under Data Platform: Tetration Apps. Neighborhood can also be enabled/disabled from Tetration Apps.

How to enable/disable

Neighborhood is automatically enabled on all root scopes. It will be listed under Tetration Apps (if enabled).

Fig. 9.2.3.2.3: Neighborhood listed in Tetration Apps

If disabled, it can be manually enabled from App Store.

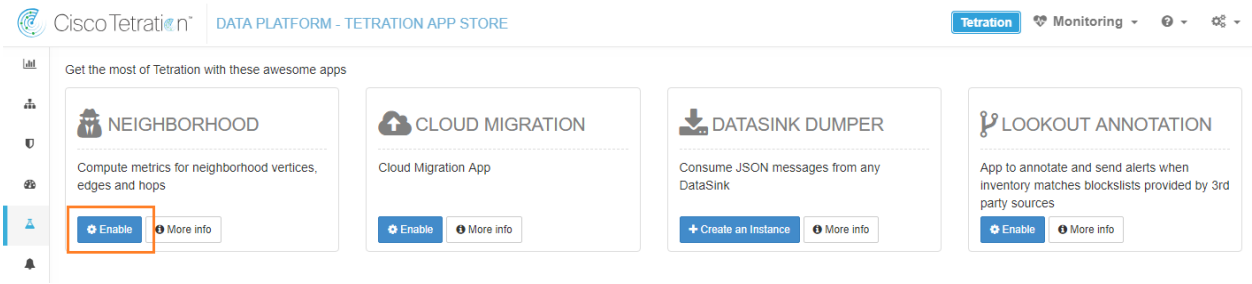


Fig. 9.2.3.2.4: Select Neighborhood in App Store

From the App Store you can disable Neighborhood if not needed.

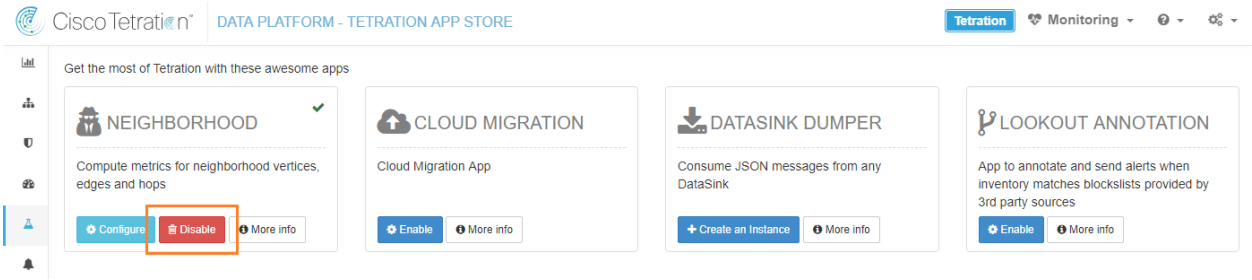


Fig. 9.2.3.2.5: Disable Neighborhood in App Store

Terminology

Node

- Nodes can be *Scopes* or *Inventory Filters/Clusters* that are part of a primary application workspace with Live Analysis or Enforcement enabled. Additionally, filters must have an ownership scope corresponding to a workspace where Live Analysis or Enforcement is enabled.

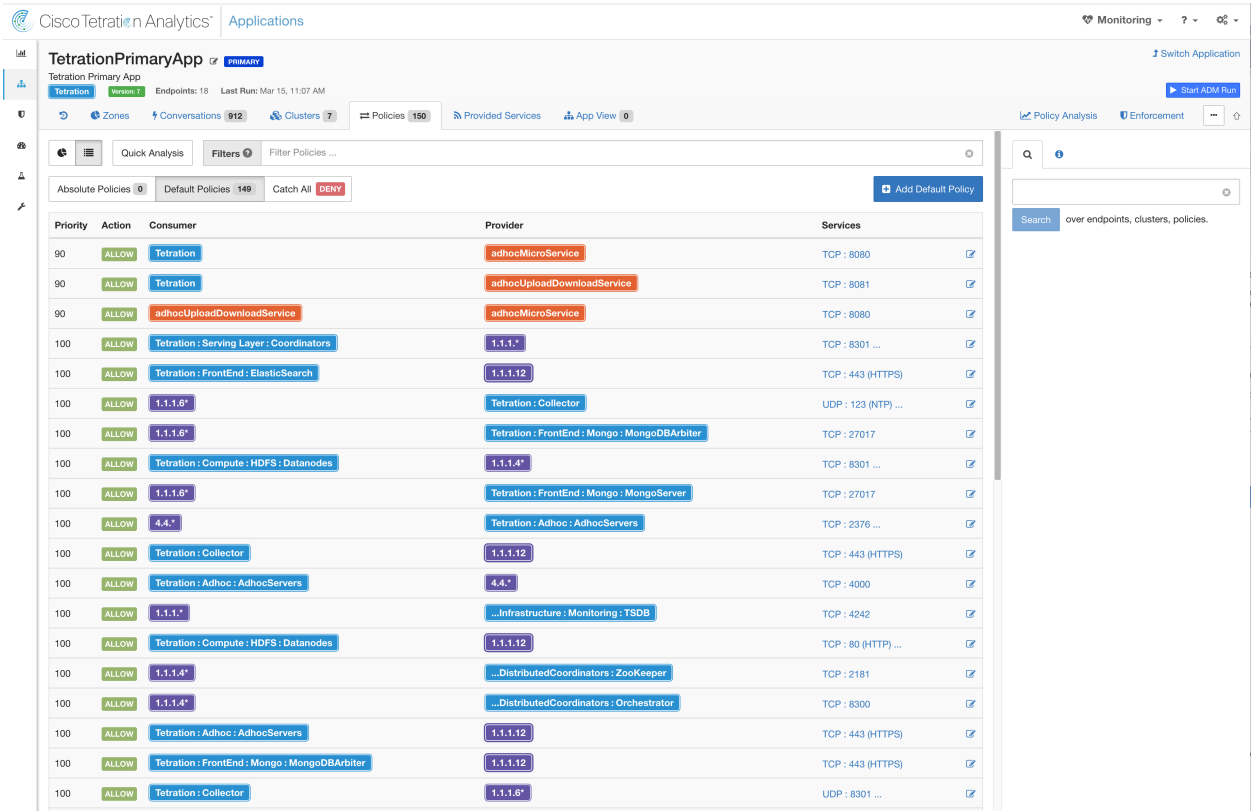


Fig. 9.2.3.2.6: Application Live Analysis

Limits

- For inventory filters and clusters, each individual scope has a size limit: 500.
- The priority is given to latest inventory filters, then latest clusters. Latest by update time.

Exploring Neighborhood Data

Clicking the "Neighborhood" App will change the view to the Neighborhood UI where neighborhood data can be explored.

Neighborhood has five different views:

1. Geo Inbound
2. Geo Outbound
3. Inbound Neighbors
4. Outbound Neighbors
5. Paths

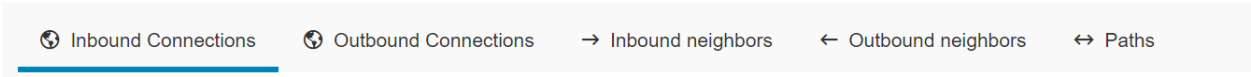


Fig. 9.2.3.2.7: Neighborhood exploration options

Exploring Geo Data

Neighborhood geo exposes two directions of geo data:

1. Inbound. Aggregate view of flows from a Geo location (such as Country) to a Scope (or Filter/Cluster)
2. Outbound. Aggregate view of flows from a Scope (or Filter/Cluster) to a Geo location (such as Country)

Note that Geo view is based on the Source/Consumer Scope (Outbound) or Destination/Provider Scope (Inbound), and not on the direction of data. Within a selected geo view 'Bytes Sent' or 'Bytes Received' can chosen.

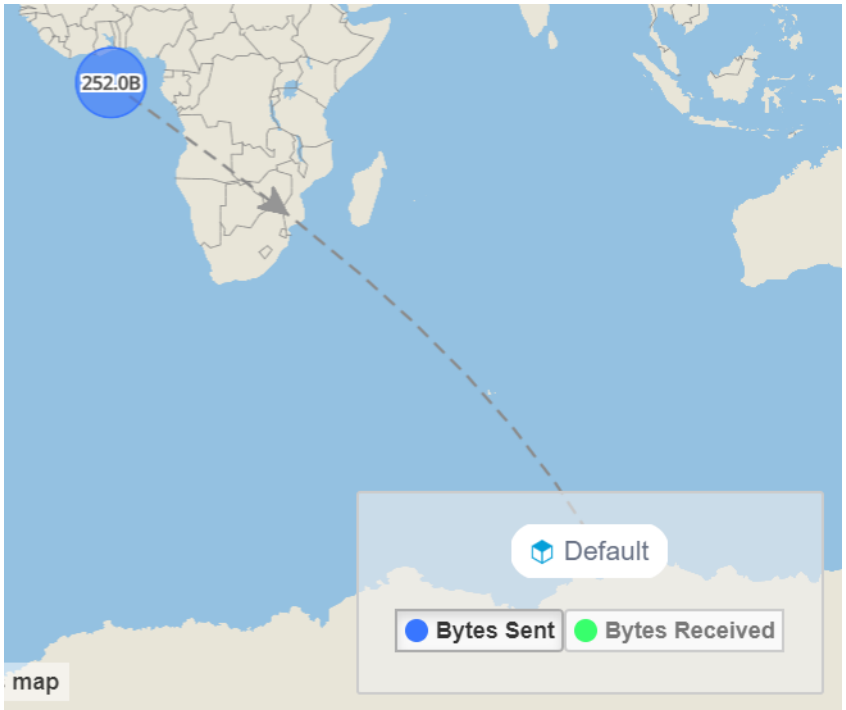


Fig. 9.2.3.2.8: Inbound: Geo Location → Node

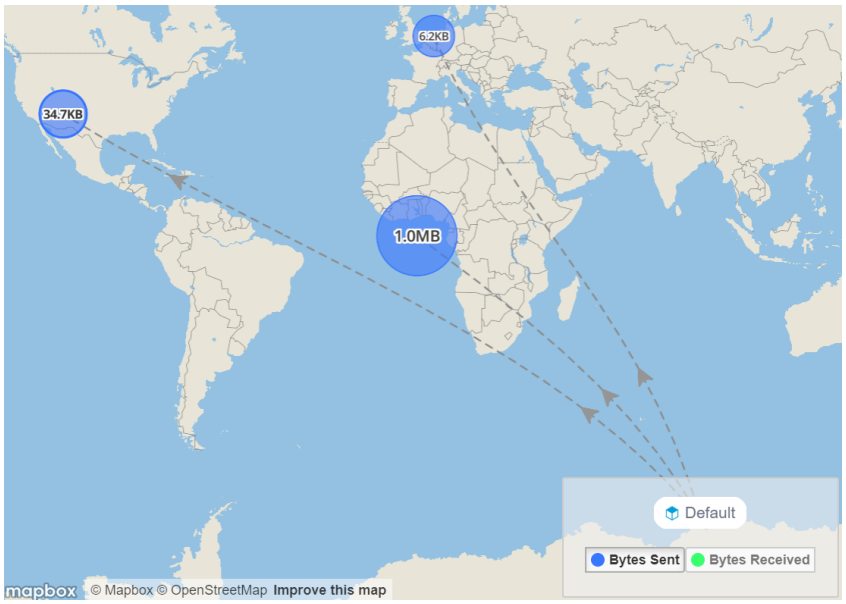


Fig. 9.2.3.2.9: Outbound: Node → Geo Location

Supported Filters

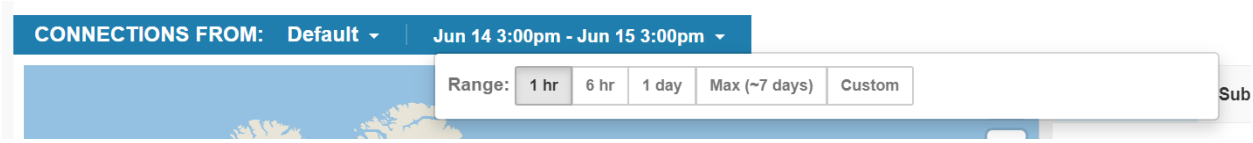


Fig. 9.2.3.2.10: Options for selecting the time range to aggregate data over. Note: limited to last 7 days.

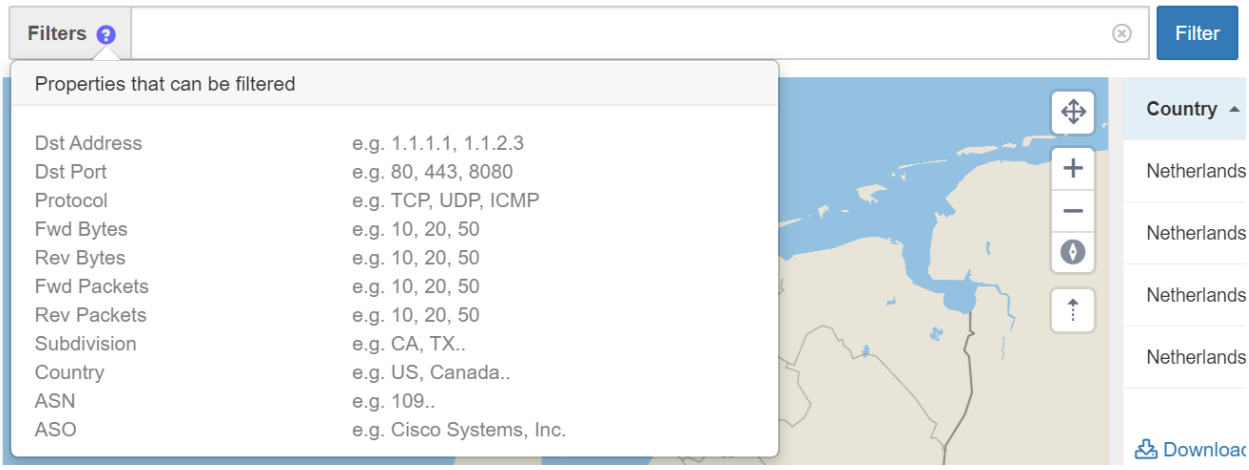


Fig. 9.2.3.2.11: Options for filtering neighborhood data

The filter input also supports "," and "-" for Port, Consumer Address and Provider Address, by translating "-" into range queries. The following is an example of a valid filter:

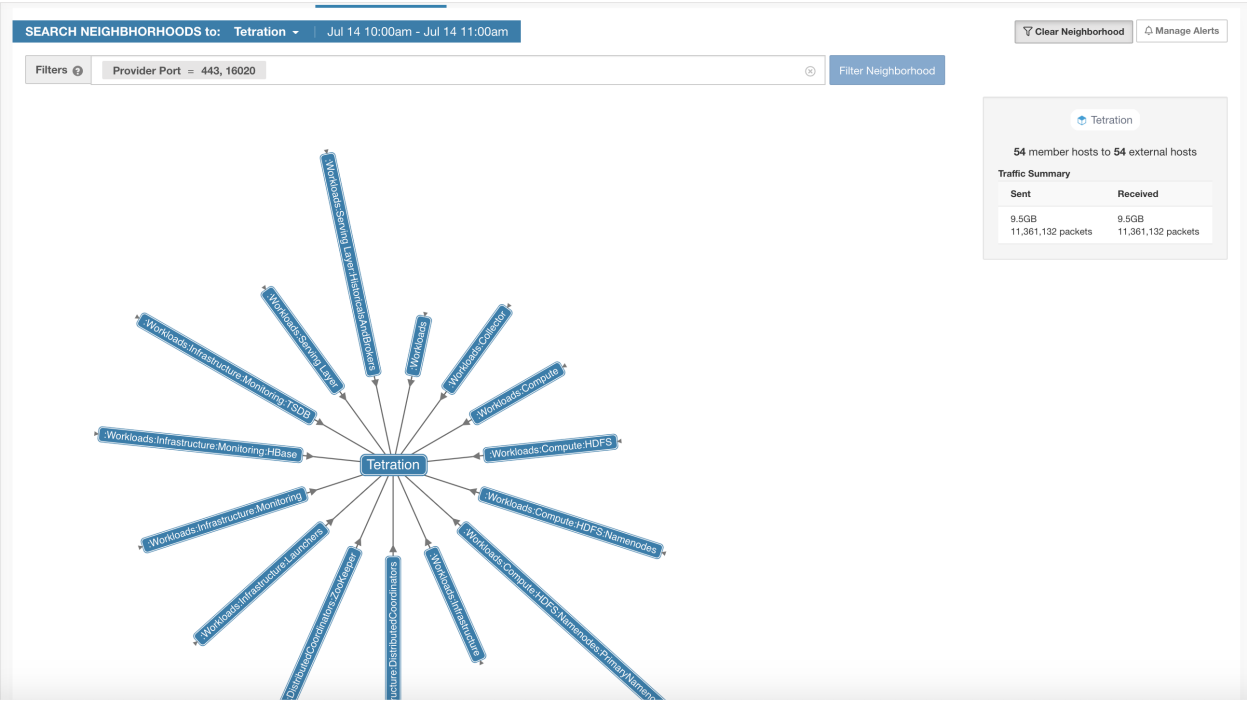


Fig. 9.2.3.2.12: Example: Filter input supports “,” for Ports

Navigation

Main points for navigating Geo page:

0. Navigate to Inventory Profile for detail inventory information including more historical Geo data.
1. Node Selection. Note: only scope with Geo data available will be displayed in dropdown list.
2. Time Range Selection.
3. Toggle Filter Selection On, or Clear
4. Data with unknown geo location will be displayed as coming from or going to ‘Null Island’
5. Arrows indicating if source of flows is from the scope/node (shown here) or from the world.
6. Multiple geo locations may be grouped on map. A hand pointer icon will indicate if the cluster is clickable to zoom in and disambiguate.
7. Clicking a row in the table will set the country and subdivision as a filter, zoom in map, and display multiple addresses.
8. Takes map to Fullscreen Mode.
9. Map zoom in centering around region below the mouse.
10. Map zoom out.
11. Drag the button in place to change the map’s bearing for a more 3D look.
12. Toggle off lines and arrows along with their hovered popup to emphasize data clusters.
13. Horizontally **resize** the map to emphasize either map or table display.

Other points to note:

- Bottom right of map will display the selected Node/Scope. Bytes Sent or Bytes Received can be selected.
- Bottom of table provides a download link for the json data.

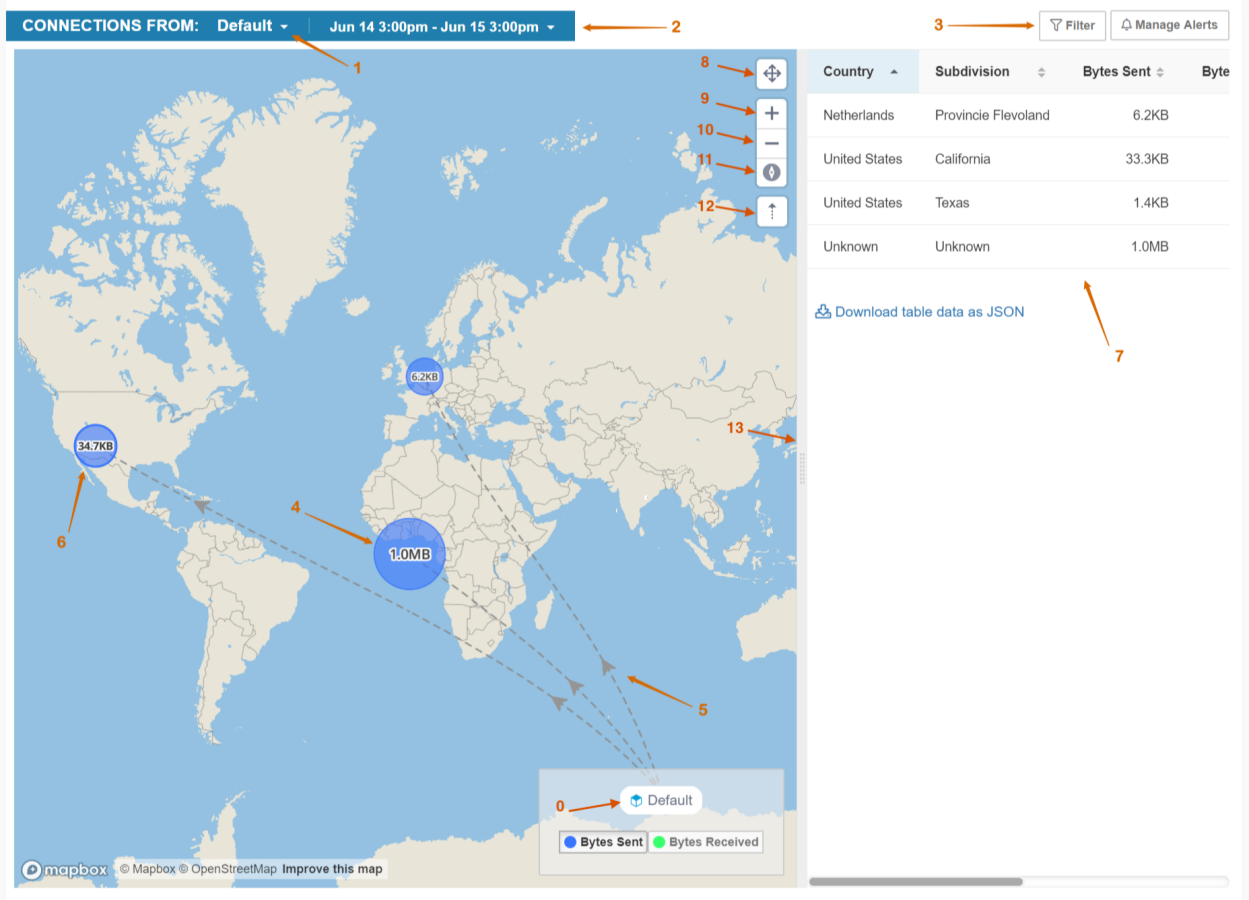


Fig. 9.2.3.2.13: Highlighted navigation points for Geo

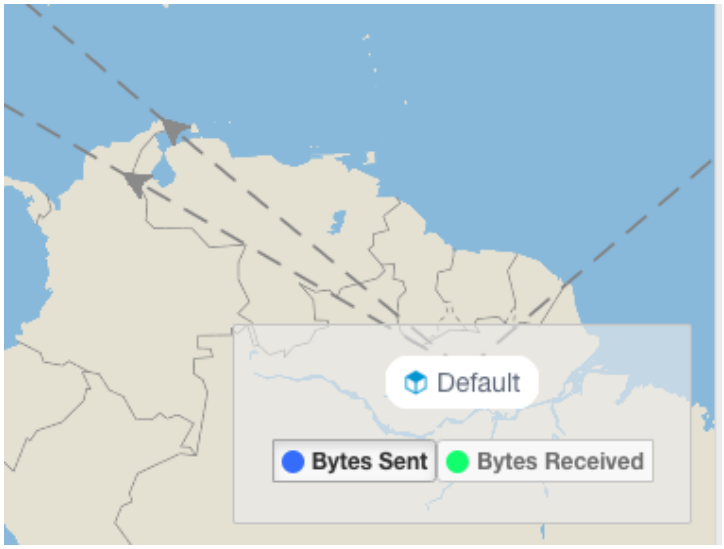


Fig. 9.2.3.2.14: Example #0. Clicking on the Node/Scope will lead to *Inventory Profile*.

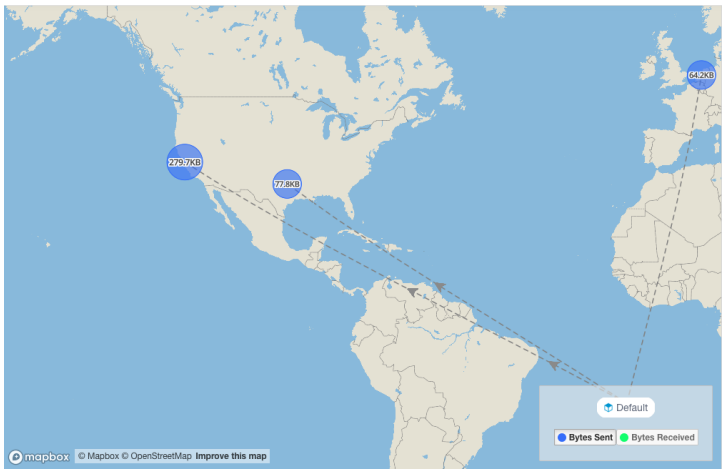


Fig. 9.2.3.2.15: Example #6. Clicking on a group of clustered points on the map (identified by hand icon) will zoom in to disambiguate multiple clustered points.

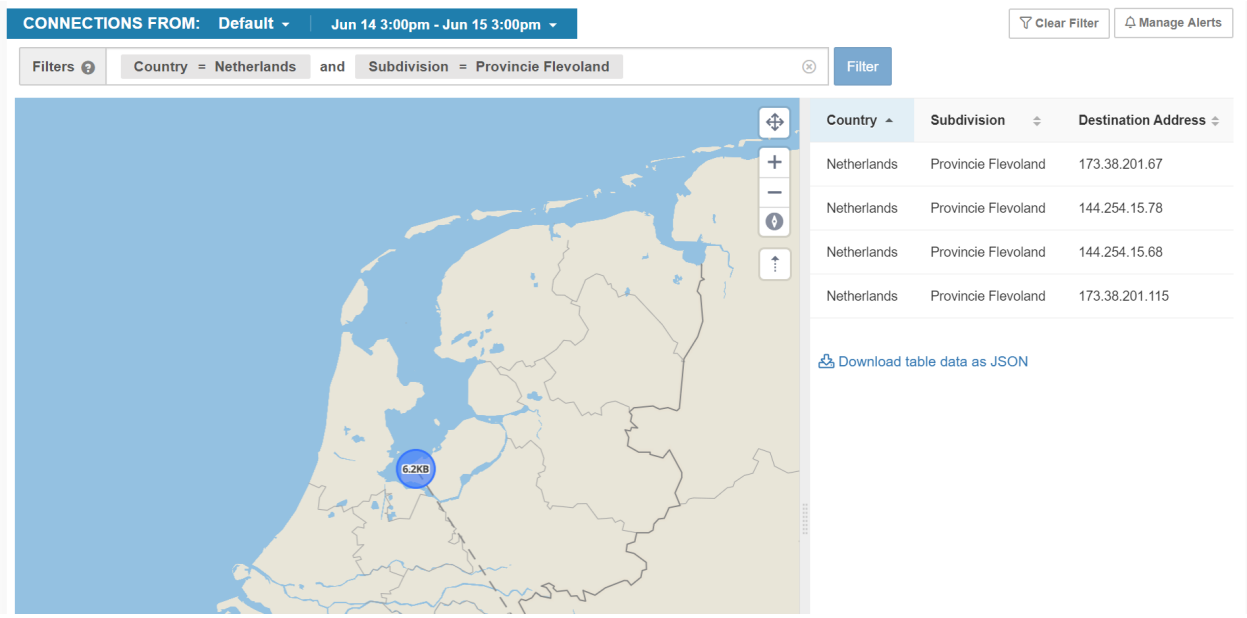
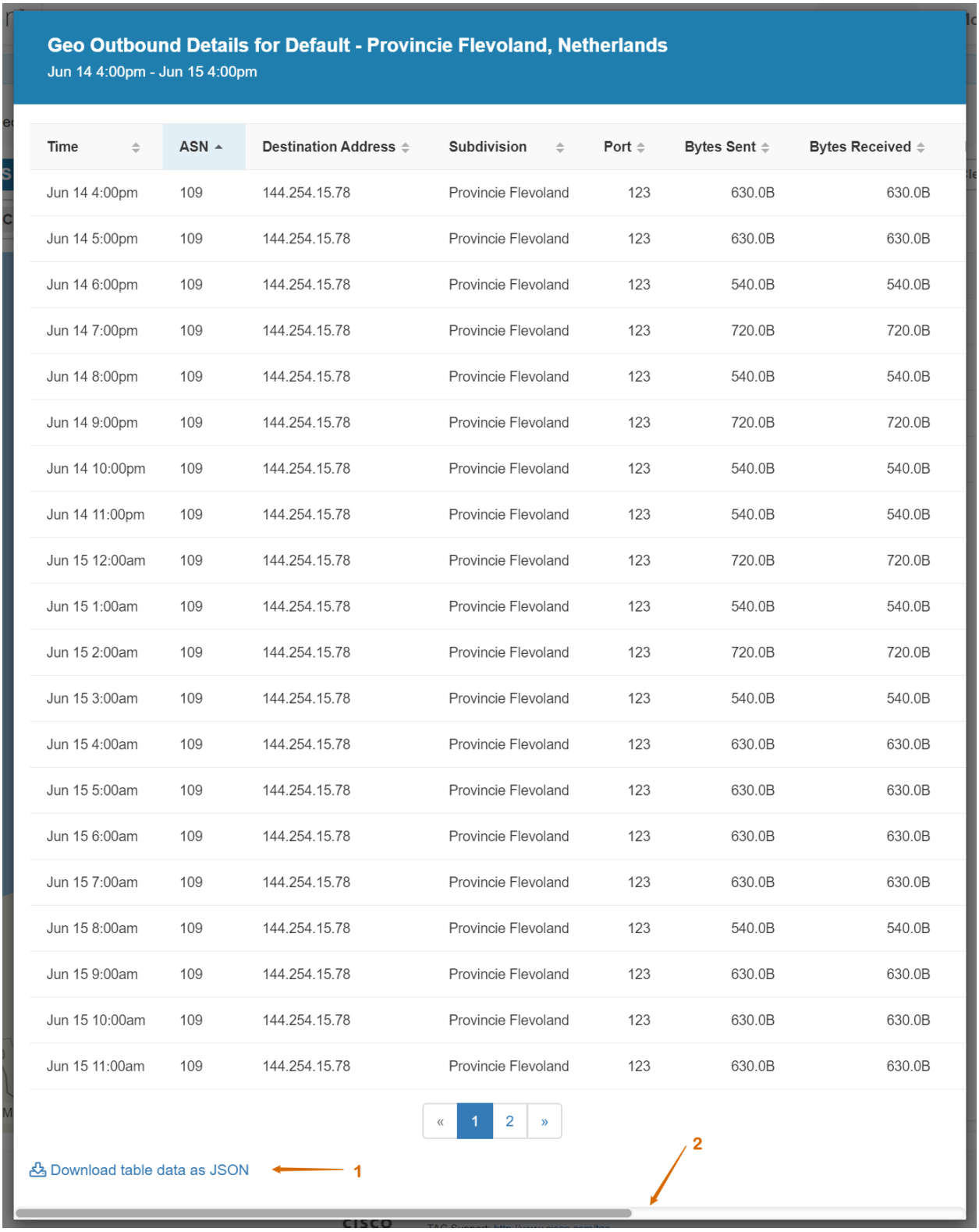


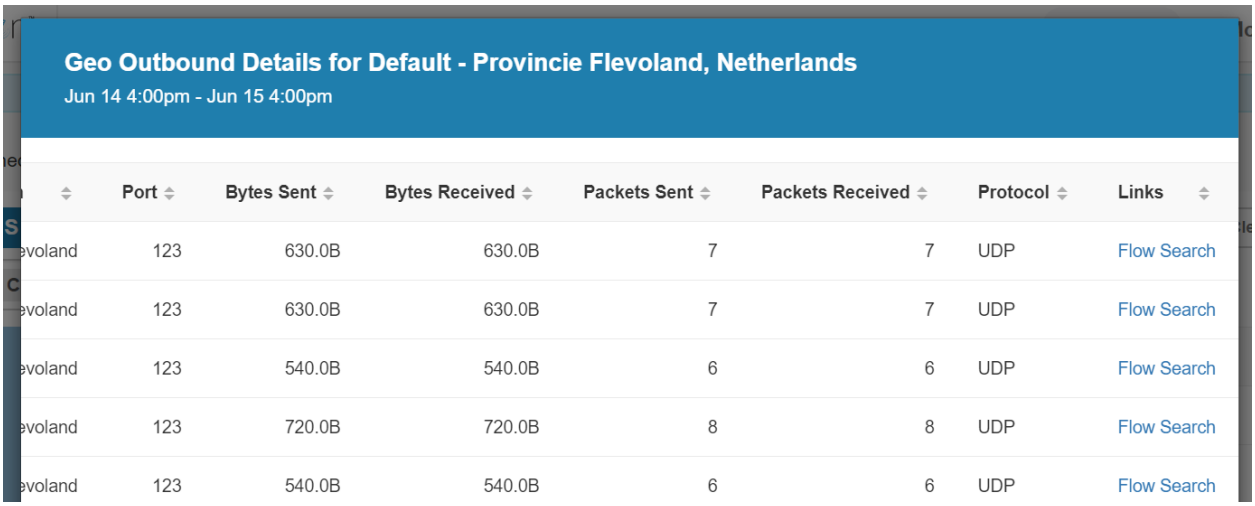
Fig. 9.2.3.2.16: Example #7. Clicking a row in the table will set those properties in the filter and zoom in. Table will then show multiple addresses.

After selecting specfic source and destination, a detailed view will popup.



Time	ASN	Destination Address	Subdivision	Port	Bytes Sent	Bytes Received
Jun 14 4:00pm	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 14 5:00pm	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 14 6:00pm	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 14 7:00pm	109	144.254.15.78	Provincie Flevoland	123	720.0B	720.0B
Jun 14 8:00pm	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 14 9:00pm	109	144.254.15.78	Provincie Flevoland	123	720.0B	720.0B
Jun 14 10:00pm	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 14 11:00pm	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 15 12:00am	109	144.254.15.78	Provincie Flevoland	123	720.0B	720.0B
Jun 15 1:00am	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 15 2:00am	109	144.254.15.78	Provincie Flevoland	123	720.0B	720.0B
Jun 15 3:00am	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 15 4:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 5:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 6:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 7:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 8:00am	109	144.254.15.78	Provincie Flevoland	123	540.0B	540.0B
Jun 15 9:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 10:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B
Jun 15 11:00am	109	144.254.15.78	Provincie Flevoland	123	630.0B	630.0B

Fig. 9.2.3.2.17: Clicking on row from prior address list view will pop up the details view. 1. This data can be downloaded. 2. Scroll right to get to additional columns, such as flow search link



	Port	Bytes Sent	Bytes Received	Packets Sent	Packets Received	Protocol	Links
evoland	123	630.0B	630.0B	7	7	UDP	Flow Search
evoland	123	630.0B	630.0B	7	7	UDP	Flow Search
evoland	123	540.0B	540.0B	6	6	UDP	Flow Search
evoland	123	720.0B	720.0B	8	8	UDP	Flow Search
evoland	123	540.0B	540.0B	6	6	UDP	Flow Search

Fig. 9.2.3.2.18: After scrolling right in detail view. Flow search link from detail view.

Exploring Neighborhoods

Exploration of aggregated node (Scope, Filter, Cluster) data has 3 versions:

1. Inbound: Aggregated flows with the selected node as a destination
2. Outbound: Aggregated flows with the selected node as the source
3. Paths: Aggregate view of flows where one source node and one destination node are constrained. Note that these are aggregated node-to-node edges, but are otherwise unrelated.

Inbound and Outbound Exploration

Choose to enter a node of interest. And select either Inbound or Outbound

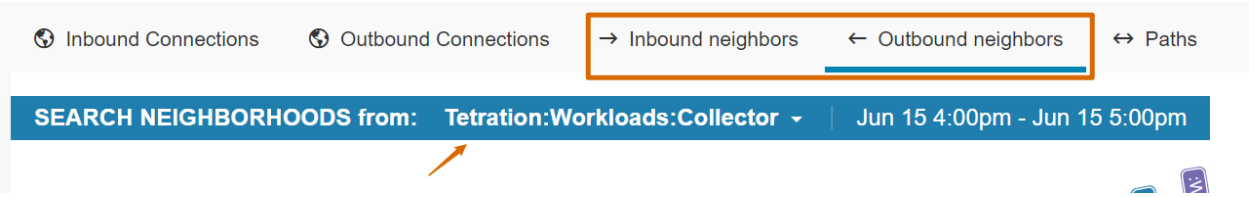


Fig. 9.2.3.2.19: Exploring Neighborhood Data

A radial tree will be shown with the selected node in the center, and adjacent nodes up to two hops away radiating inward. Below the radial tree will be a list of paths toward the selected node.

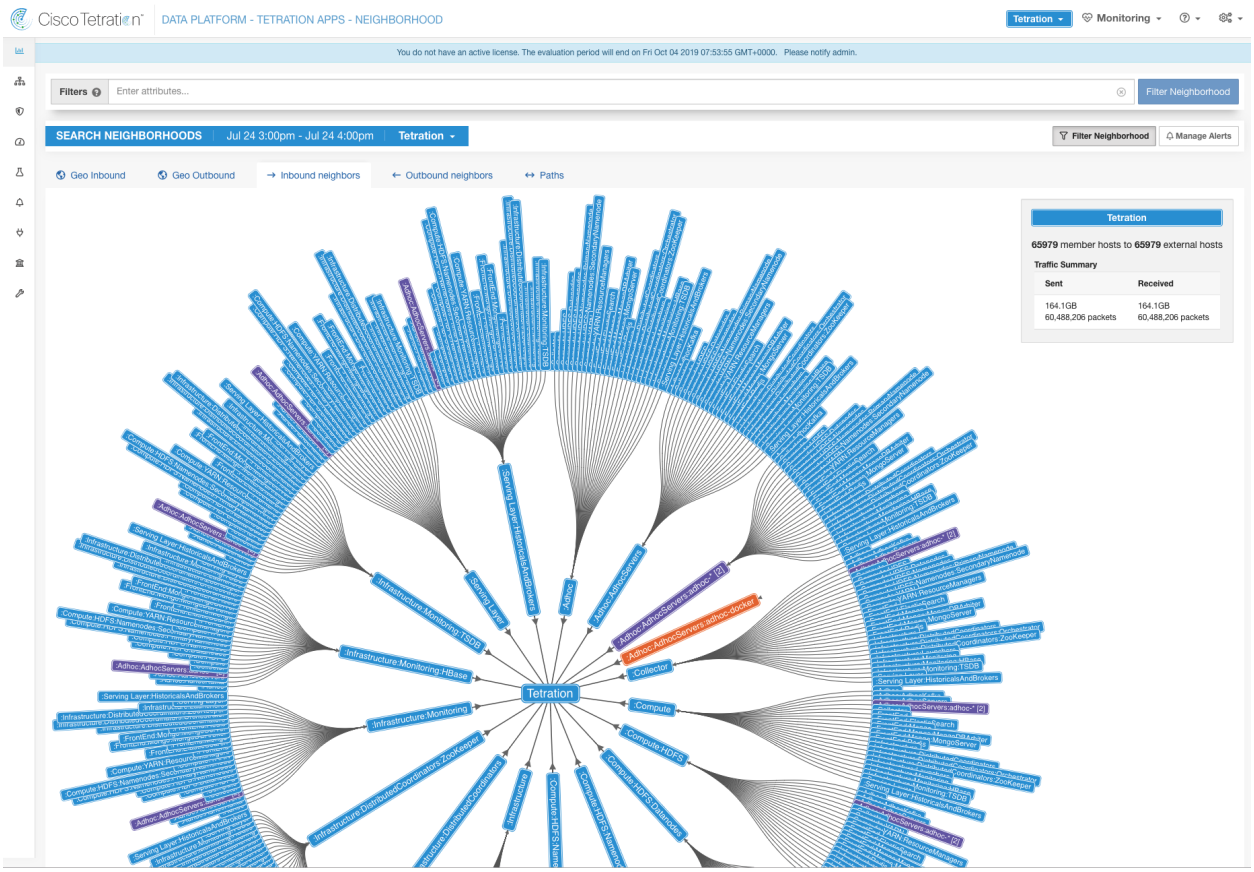


Fig. 9.2.3.2.20: Node

Path Exploration

Selecting “Paths” instead of inbound/outbound will allow specifying both a source and a destination.

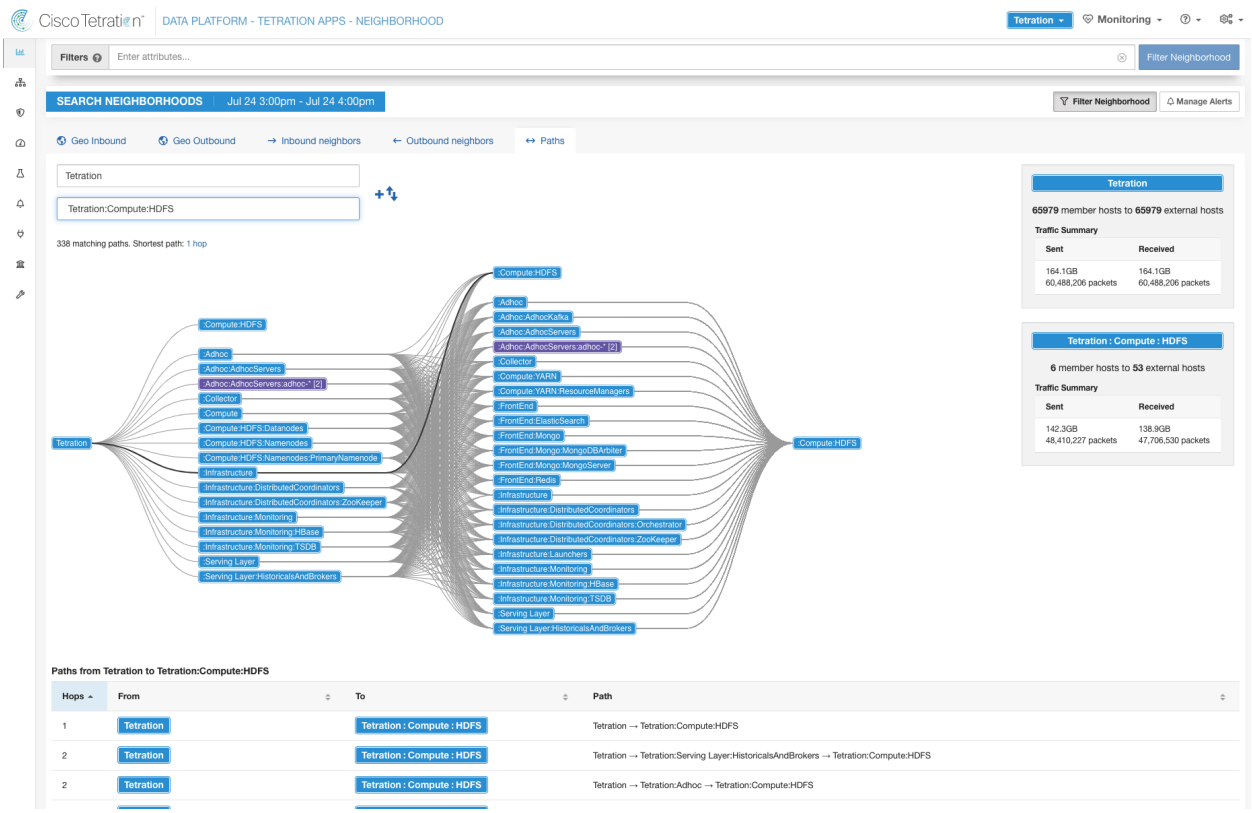


Fig. 9.2.3.2.21: Path

Filter Options

The neighborhood graph can be filtered by specifying additional filter options. Currently supported filters are *Provider Port* and *Protocol*.

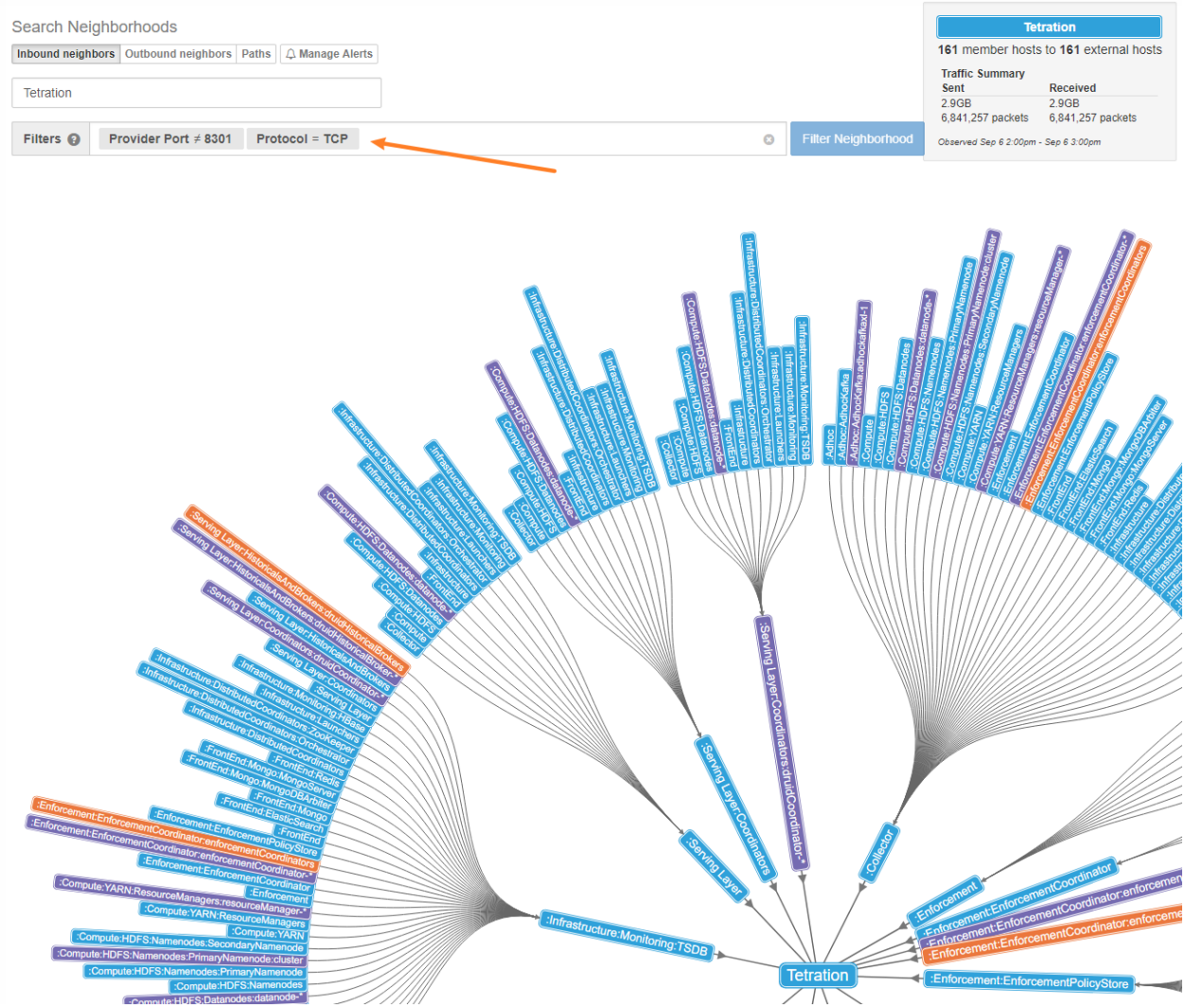


Fig. 9.2.3.2.22: Filtering nodes

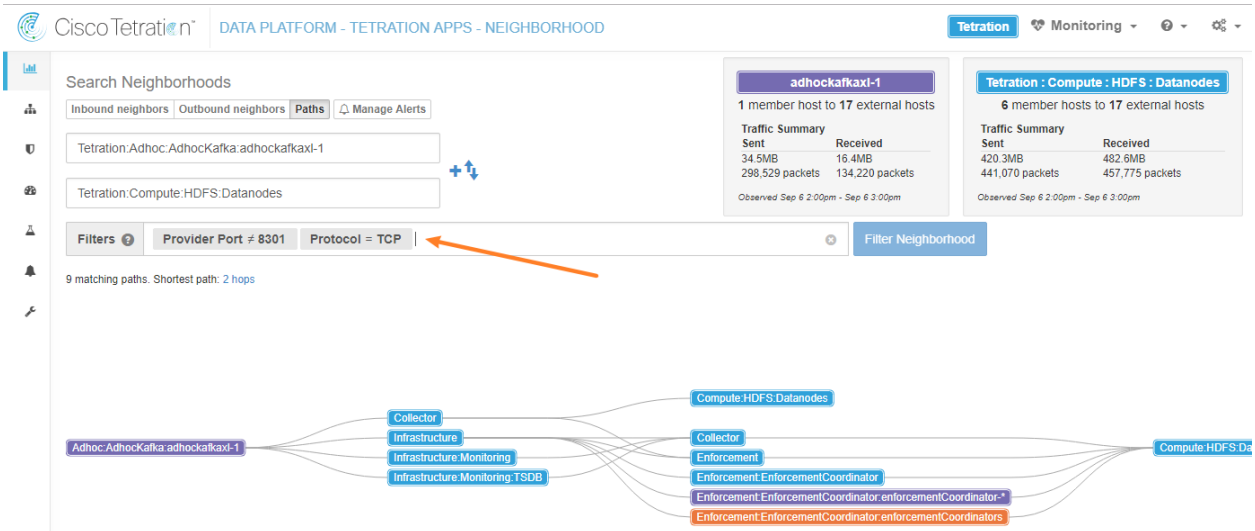


Fig. 9.2.3.2.23: Filtering paths

Clicking any path listed below the graph, will expand with details about the path and provide links to flow search.

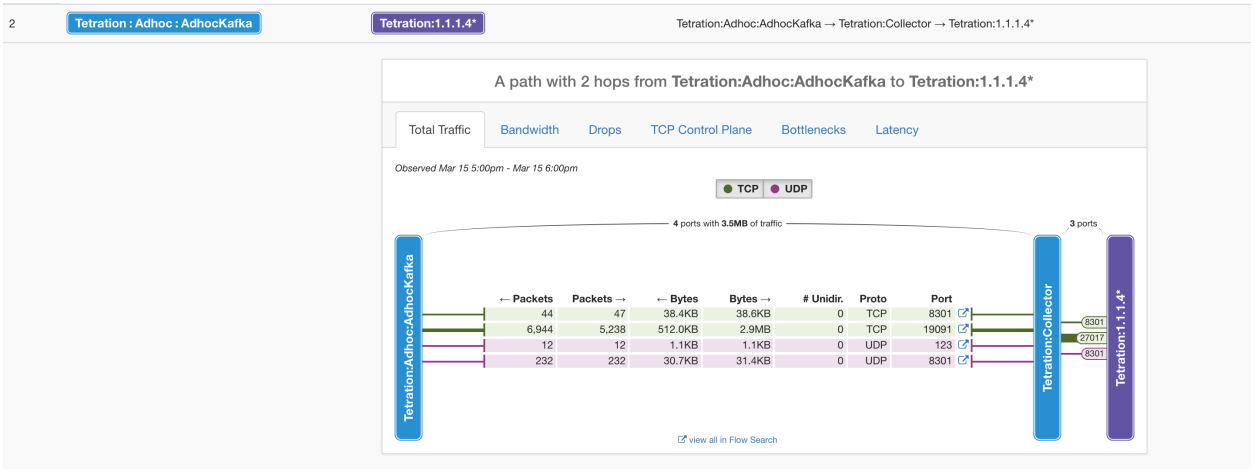


Fig. 9.2.3.2.24: Path details

Path Details

Path details contains tabs showcasing different groups of metrics: *Total Traffic*, *Bandwith*, *Drops*, *TCP Control Plane*, *Bottlenecks*, *Latency*. Note: some metrics may not be available depending on flow data collection method.

Total Traffic

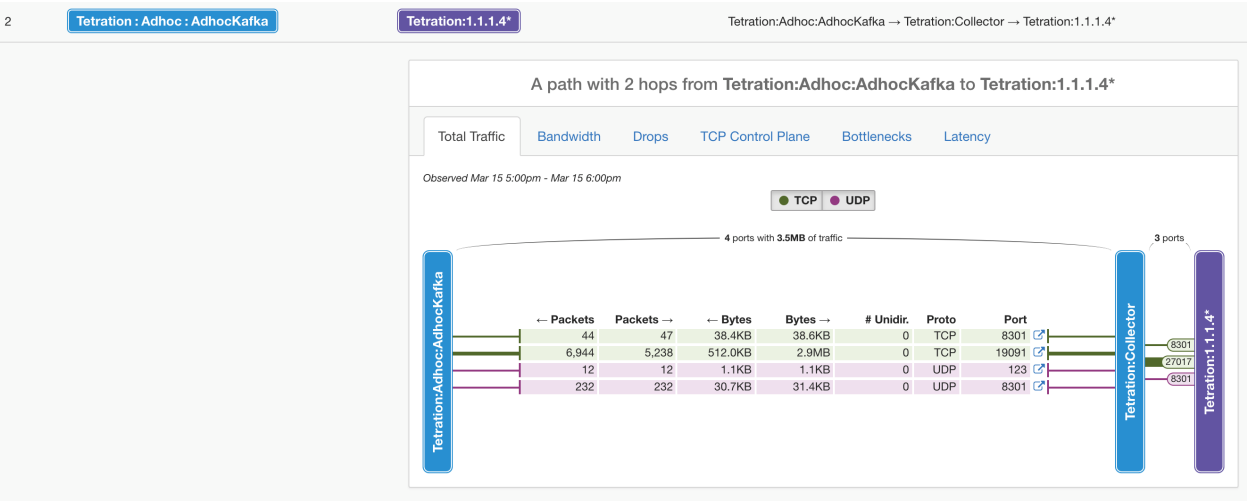


Fig. 9.2.3.2.25: Total Traffic

Bandwidth

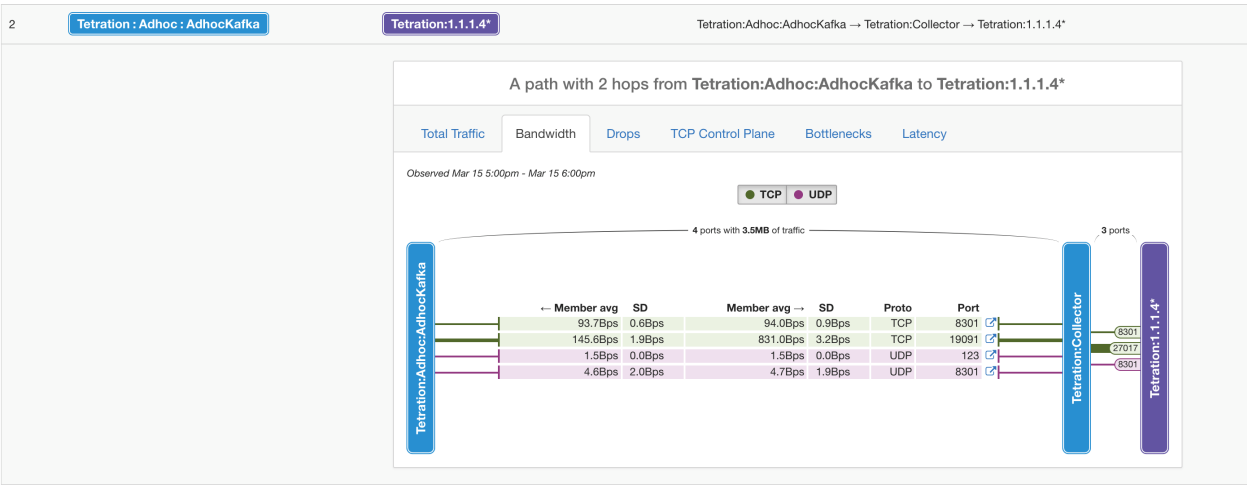


Fig. 9.2.3.2.26: Bandwidth

Drops

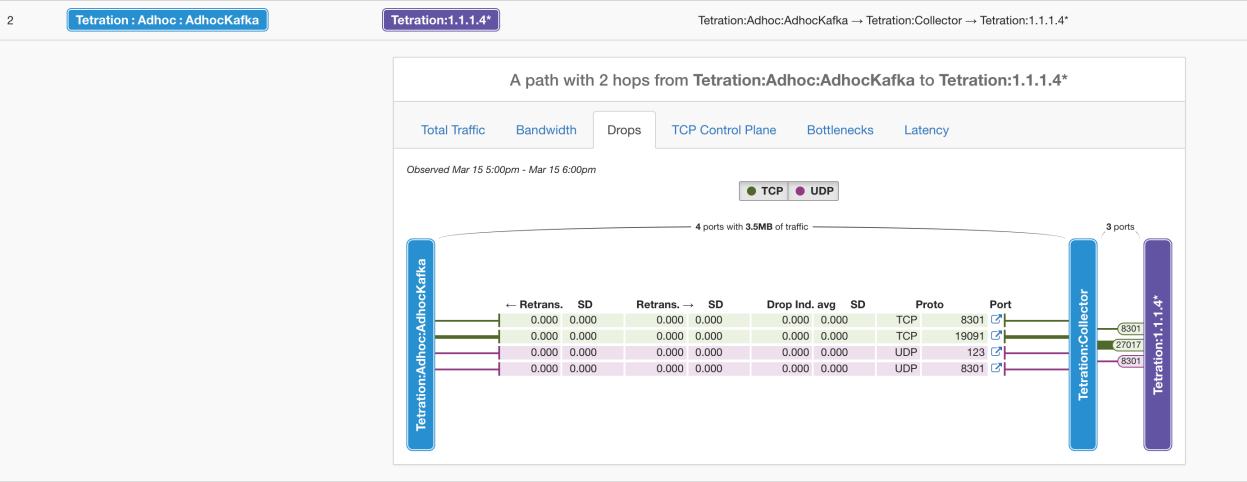


Fig. 9.2.3.2.27: Drops

TCP Control Plane

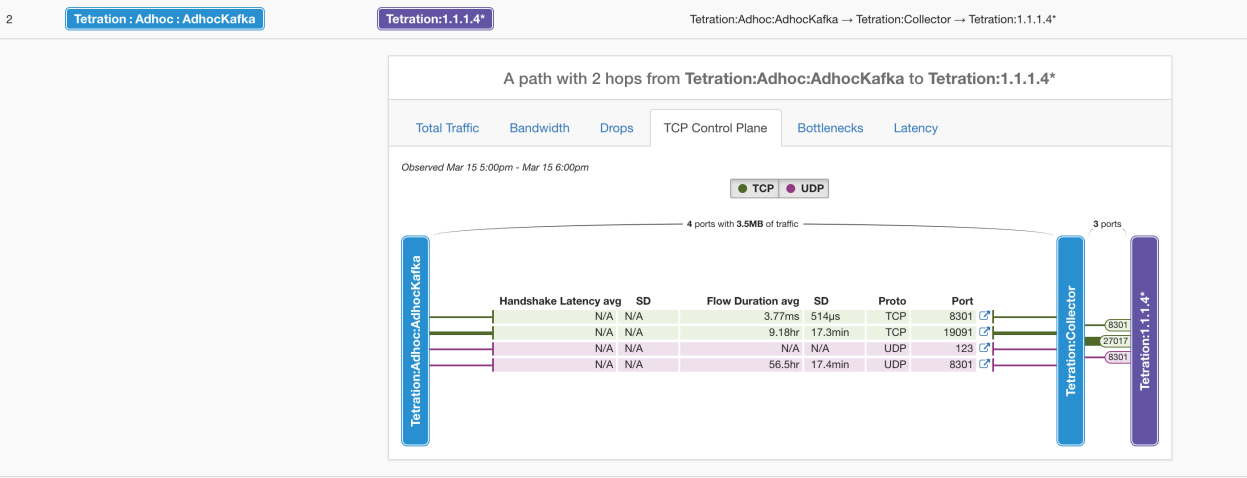


Fig. 9.2.3.2.28: TCP Control Plane

Bottlenecks

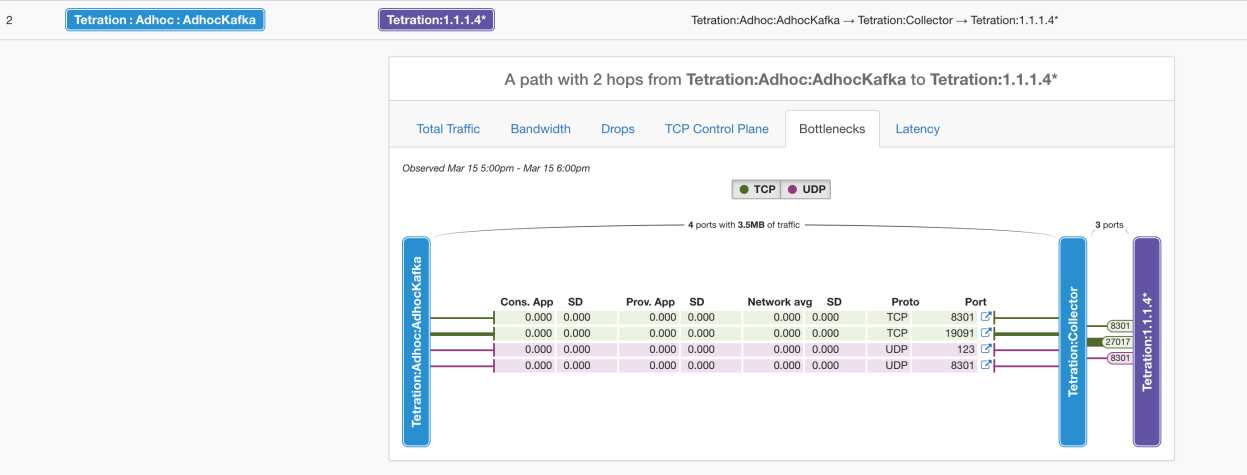


Fig. 9.2.3.2.29: Bottlenecks

Latency

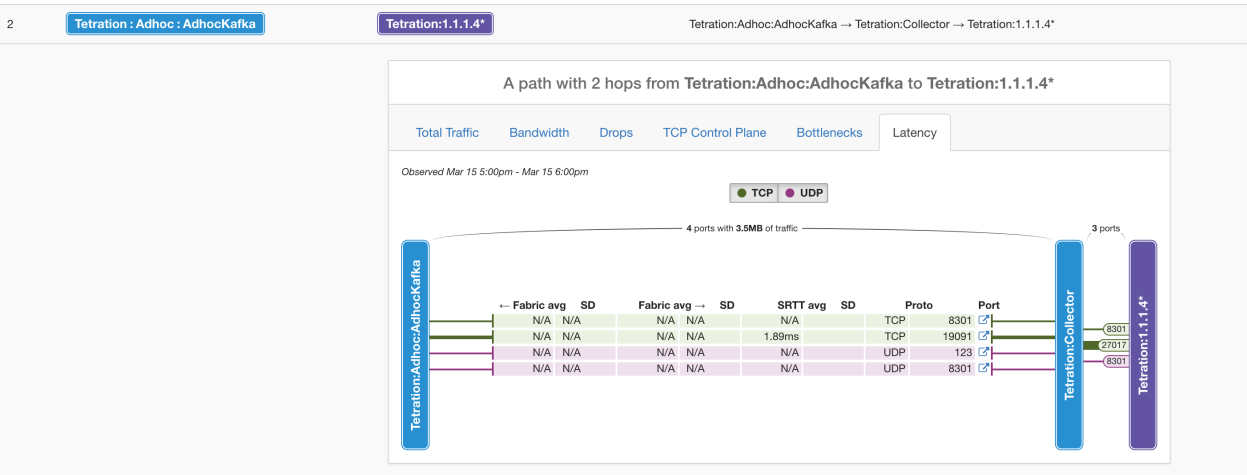


Fig. 9.2.3.2.30: Latency

Neighborhood Alerts

How to set up alerts

- To configure alerts click the 'Manage Alerts' button. This opens the *Alert Configuration Modal*. Different types of alerts are available for Nodes, Edges, and Paths.
- To see what alert trigger configurations are available for each, the user user could selected the type (such as Node), then click ? to see the options.
- After forming the alert trigger, the user could expand the alert configuration to change the alert frequency. The default frequency is 'hourly', but can be changed to 'daily'.

Supported Alerts

Type	Condition	Note
Geo	Direction	** Only to be used in conjunction with ASO and Country
Geo	ASO	Check ASO condition (= or $\neq$) according to direction (above)
Geo	Country	Check Country condition (= or $\neq$) according to direction (above)
Path	Any Hops	Check path not through specified node
Path	Path	Compare path size with specified value
Edge	Avg SRTT	Compare Avg SRTT with specified value
Edge	Max SRTT	Compare Max SRTT with specified value
Edge	Unidirectional Flows	Check unidirectional flow or not
Node	Membership Count	Compare Membership count with specified value
Node	Adjacency Count	Compare Adjacency count with specified value

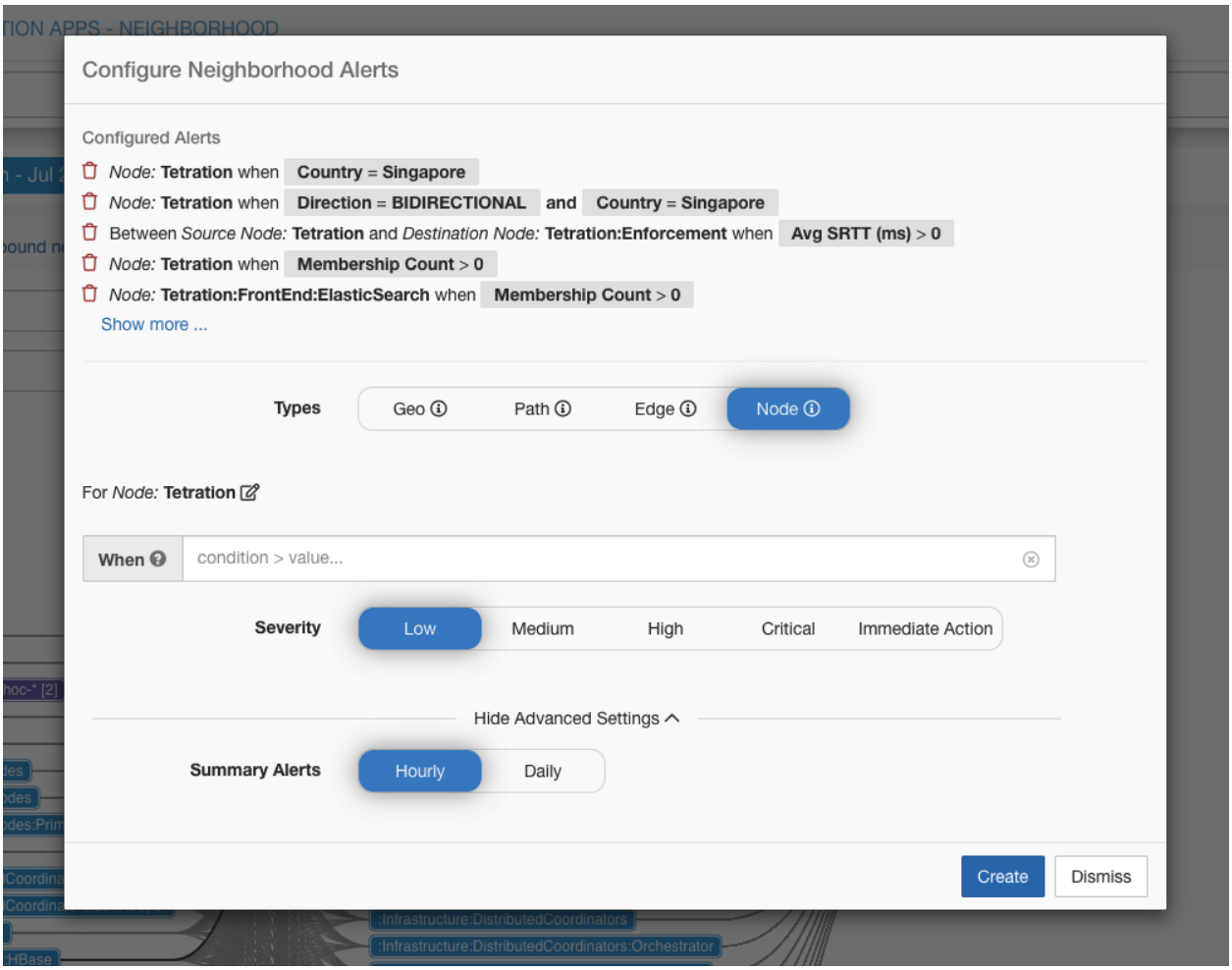


Fig. 9.2.3.2.31: Manage alerts

Warning: Configured alerts on a subscope or filter will not be automatically deleted if the subscope or filter is deleted. New clusters with equivalent queries will remain relevant, but if a cluster or filter is no longer used in

the latest live analysis policies, then no alerts will be generated that use those clusters and filters, and the outdated alert configurations will remain. Users should periodically review their configured alerts to make sure they remain relevant.

How to view alerts

- **A valid Data Tap must be selected for the Neighborhood alerts. Alerts will only be visible in the UI if they were successfull**
  - Alert publishers and notifiers can be chosen from Alerts → Configuration (Root Scope Owners or Site Admins).
- **After configuring alerts and setting up data tap, alerts can be viewed in the UI under Alerts → Current Alerts.**
  - User can use **Type = NEIGHBORHOOD** in the filter selection box. See *Current Alerts* for more filtering options.
  - Alert details can be seen by clicking an alert.

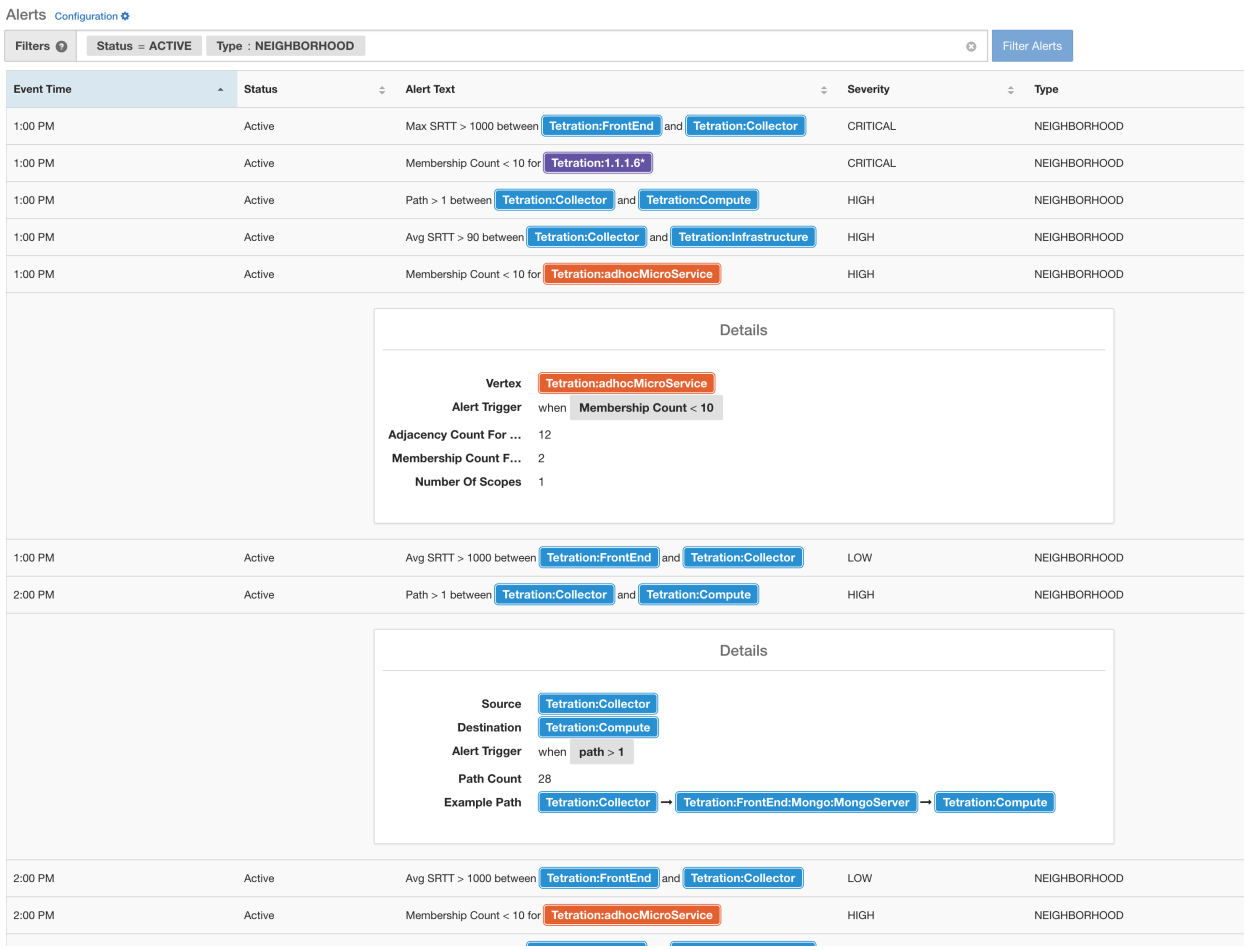


Fig. 9.2.3.2.32: Neighborhood alerts

Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* field is structured and contains the following subfields for neighborhood alerts

Note: Subject (interval name of node) is the neighborhood node that triggered the alert.

Field	Alert Type	Format	Explanation
neighborhood_subjects_id	*all*	string	neighbrohood node id
neighborhood_subjects_name	*all*	string	neighbrohood node name
internal_trigger	*all*	structured	query describing alert trigger (details in next table)
country	*geo*	string	country name
subdivision	*geo*	string	subdivision name
aso	*geo*	string	aso name
flow	*geo*	string	flow details triggered alert (src and dst ip)
vertex_neighborhood_subjects_id	*all*	string	same as neighborhood_subjects_id
adjacency_count_for_example_vertex	*node*	integer	adjacency count of given node
membership_count_for_example_vertex	*node*	integer	membership count of given node
src_neighborhood_subjects_id	*edge, path*	string	src neighborhood subject id (scope, cluster, or filter)
src_neighborhood_subjects_name	*edge, path*	string	src neighborhood subject name (scope, cluster, or filter)
dst_neighborhood_subjects_id	*edge, path*	string	dst neighborhood subject id (scope, cluster, or filter)
dst_neighborhood_subjects_name	*edge, path*	string	dst neighborhood subject name (scope, cluster, or filter)
number_of_edges	*edge*	integer	number of edges triggered alerts
max_srtt	*edge*	integer	max value of srtt across flows with triggered condition
avg_srtt	*edge*	integer	avg value of flow srtt in triggered alerts
unidirectional_flow_count	*edge*	string	number of flows (plural)
example_path_neighborhood_subjects_id	*path*	array[string]	list of ids consisting of scopes, clusters, or filters comprising one example path matching the trigger condition
example_path_neighborhood_subjects_name	*path*	array[string]	list of subjects consisting of scopes, clusters, or filters comprising one example path matching the trigger condition
number_of_unique_paths	*path*	integer	number of unique paths matching the trigger condition

The *internal_trigger* fields are structured and contain the following subfields for alert trigger

Field	Format	Explanation
datasource	string	alert type
rules	string	collection of query evaluation rules
filters	string	list of combination query rules
type	string	query rule type (e.g. eq, lt, gt. . .)
value	string	user input values in alert configuration
label	string	"Alert Trigger"

Example of alert_details for Geo (ASO) alert

```
{
  "neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "country":"United States",
  "subdivision":"Texas",
  "internal_trigger":{
    "datasource":"geo",
    "rules":{
       "filters":[
          {
             "field":"direction",
             "type":"eq",
             "value":"BIDIRECTIONAL"
          },
          {
             "field":"aso",
             "type":"eq",
             "value":"CISCOSYSTEMS"
          }
       ],
       "type":"and"
    },
    "label":"Alert Trigger"
 },
 "neighborhood_subjects_name":"Default",
 "vertex_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
 "flow":"72.163.32.44 -> Default"
}
```

Example of alert_details for Geo (Country) alert

```
{
  "neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "country":"Netherlands",
  "subdivision":"Provincie Flevoland",
  "internal_trigger":{
    "datasource":"geo",
    "rules":{
       "field":"country",
       "type":"eq",
       "value":"Netherlands"
    },
    "label":"Alert Trigger"
```

(continues on next page)

(continued from previous page)

```
  },
  "neighborhood_subjects_name":"Default",
  "vertex_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "flow":"173.38.201.67 -> Default"
}
```

Example of alert_details for Node (Adjacency Count) alert

```
{
  "adjacency_count_for_example_vertex":7,
  "neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b6",
  "internal_trigger":{
    "datasource":"vertex",
    "rules":{
       "field":"adjacency_count",
       "type":"gt",
       "value":-1
    },
    "label":"Alert Trigger"
 },
 "neighborhood_subjects_name":"Default:cluster",
 "vertex_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2:c_
↪5f04b0efc5445388852786b6"
}
```

Example of alert_details for Node (Membership Count) alert

```
{
  "neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "internal_trigger":{
    "datasource":"vertex",
    "rules":{
       "field":"membership_count",
       "type":"gt",
       "value":0
    },
    "label":"Alert Trigger"
  },
  "neighborhood_subjects_name":"Default",
  "membership_count_for_example_vertex":156,
  "vertex_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2"
}
```

Example of alert_details for Edge (srtt avg) alert

```
{
  "internal_trigger":{
    "datasource":"edge",
    "rules":{
       "field":"srtt_usec_avg",
```

(continues on next page)

(continued from previous page)

```
        "type":"gt",
        "value":-1
    },
    "label":"Alert Trigger"
  },
  "src_neighborhood_subjects_id":"5efcfe0f497d4f49adebc74e",
  "dst_neighborhood_subjects_name":"Tetration:AdhocKafka",
  "dst_neighborhood_subjects_id":"5efcfe0f497d4f49adebc6ee",
  "number_of_edges":2,
  "max_srtt":0,
  "avg_srtt":0,
  "src_neighborhood_subjects_name":"Tetration:Collector"
}
```

Example of alert_details for Edge (max srtt) alert

```
{
  "internal_trigger":{
    "datasource":"edge",
    "rules":{
        "field":"srtt_usec_max",
        "type":"gt",
        "value":-1
    },
    "label":"Alert Trigger"
  },
  "src_neighborhood_subjects_id":"5efcfe0f497d4f49adebc74e",
  "dst_neighborhood_subjects_name":"Tetration:AdhocKafka",
  "dst_neighborhood_subjects_id":"5efcfe0f497d4f49adebc6ee",
  "number_of_edges":2,
  "max_srtt":0,
  "avg_srtt":0,
  "src_neighborhood_subjects_name":"Tetration:Collector"
}
```

Example of alert_details for Edge (unidirection flow) alert

```
{
  "unidirectional_flow_count":1,
  "internal_trigger":{
    "datasource":"edge",
    "rules":{
        "field":"num_unidirectional_flows",
        "type":"gt",
        "value":0
    },
    "label":"Alert Trigger"
  },
  "src_neighborhood_subjects_id":"5efcfe0f497d4f49adebc74e",
  "dst_neighborhood_subjects_name":"Tetration:AdhocKafka",
  "dst_neighborhood_subjects_id":"5efcfe0f497d4f49adebc6ee",
  "number_of_edges":1,
```

(continues on next page)

(continued from previous page)

```
  "src_neighborhood_subjects_name":"Tetration:Collector"
}
```

Example of alert_details for Path (hop size between two specified Node) alert

```
{
  "number_of_unique_paths":2,
  "example_path_neighborhood_subjects_id":[
    "5efcfdf5497d4f474f1707c2",
    "5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b6",
    "5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b7"
  ],
  "internal_trigger":{
    "datasource":"hop",
    "rules":{
       "field":"hops",
       "type":"gt",
       "value":0
    },
    "label":"Alert Trigger"
  },
  "src_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "dst_neighborhood_subjects_name":"Default:collectorDatamover-*",
  "dst_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b7
↪",
  "src_neighborhood_subjects_name":"Default",
  "example_path_neighborhood_subjects_name":[
    "Default",
    "Default:cluster",
    "Default:collectorDatamover-*"
  ]
}
```

Example of alert_details for Path (any hops Not through specified Node) alert

```
{
  "number_of_unique_paths":2,
  "example_path_neighborhood_subjects_id":[
    "5efcfdf5497d4f474f1707c2",
    "5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b6",
    "5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b7"
  ],
  "internal_trigger":{
    "datasource":"hop",
    "rules":{
       "filter":{
          "field":"path_by_neighborhood_subjects_id",
          "type":"contains",
          "value":"5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b5"
       },
       "type":"not"
    },
```

(continues on next page)

(continued from previous page)

```
    "label":"Alert Trigger"
  },
  "src_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2",
  "dst_neighborhood_subjects_name":"Default:collectorDatamover-*",
  "dst_neighborhood_subjects_id":"5efcfdf5497d4f474f1707c2:c_5f04b0efc5445388852786b7
↪",
  "src_neighborhood_subjects_name":"Default",
  "example_path_neighborhood_subjects_name":[
    "Default",
    "Default:cluster",
    "Default:collectorDatamover-*"
  ]
}
```

9.2.3.3 Data Sink Dumper

The DataSink Dumper app can be used to consume data from a DataSink. This app will read records from the DataSink topic in Kafka, convert the records to parquet format and make available in the Data Lake. Users can then access this data and use it in their user apps.

Enabling from app store

You can create an instance by going to Tetration App Store and creating an instance as shown below

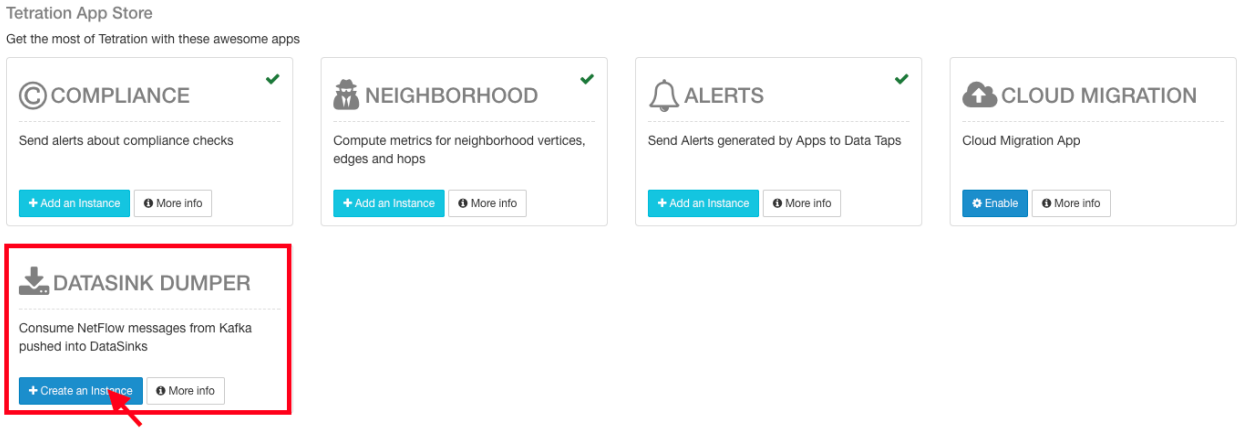


Fig. 9.2.3.3.1: Click on Add New Instance to add a new instance

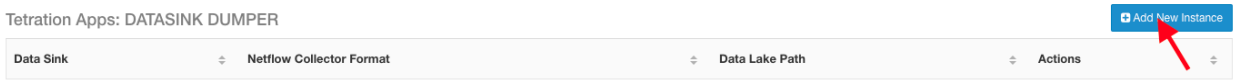


Fig. 9.2.3.3.2: Select the datasink from drop down menu. Datasink Dumper app expects the data in **JSON** format

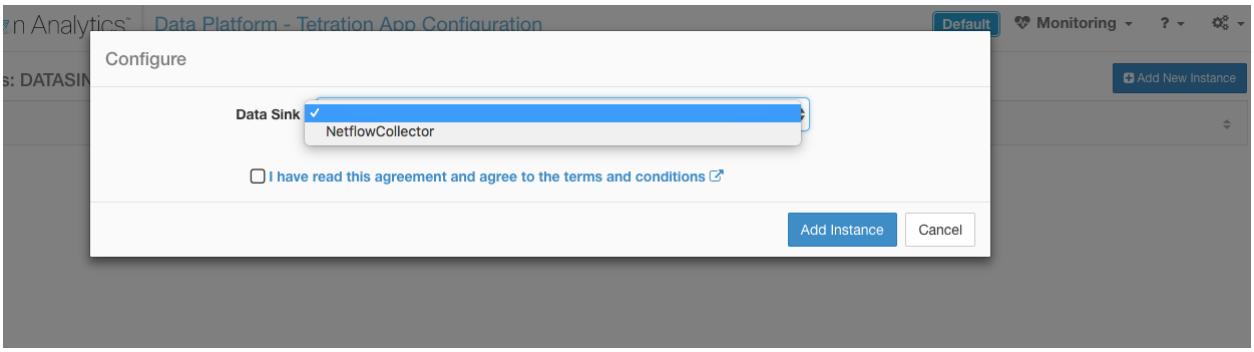


Fig. 9.2.3.3.3: Select the datasink from drop down menu. Datasink Dumper app expects the data in **JSON** format

Deleting an Instance

Fig. 9.2.3.3.4: Deleting an Instance

Viewing the list of data files created in the Data Lake

To view the Data in Data Lake, open a user app click on Data Lake → App → datasink → **Datasink Name** → **Hourly Batch**

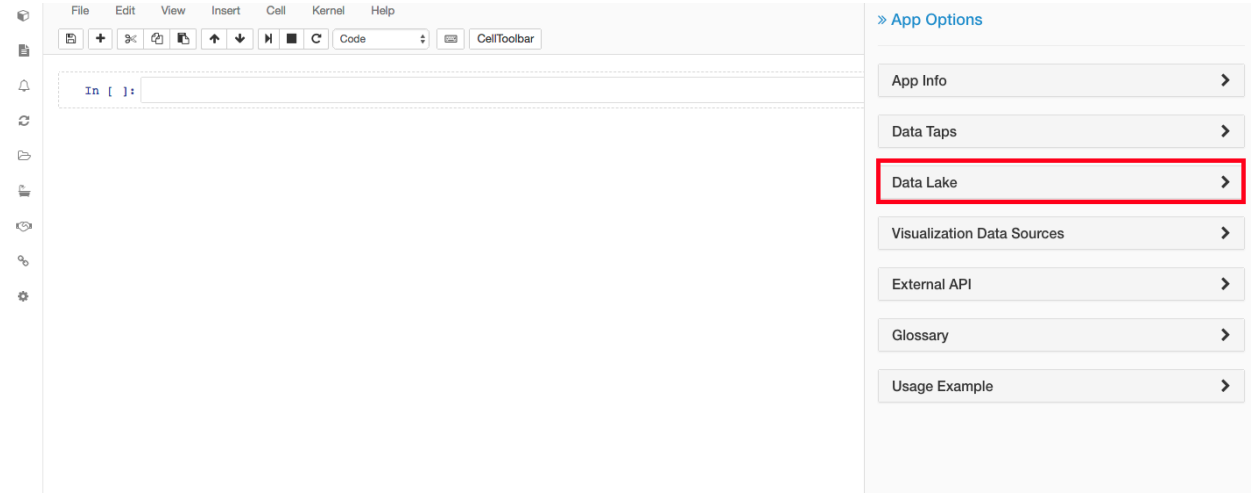


Fig. 9.2.3.3.5: Datasink

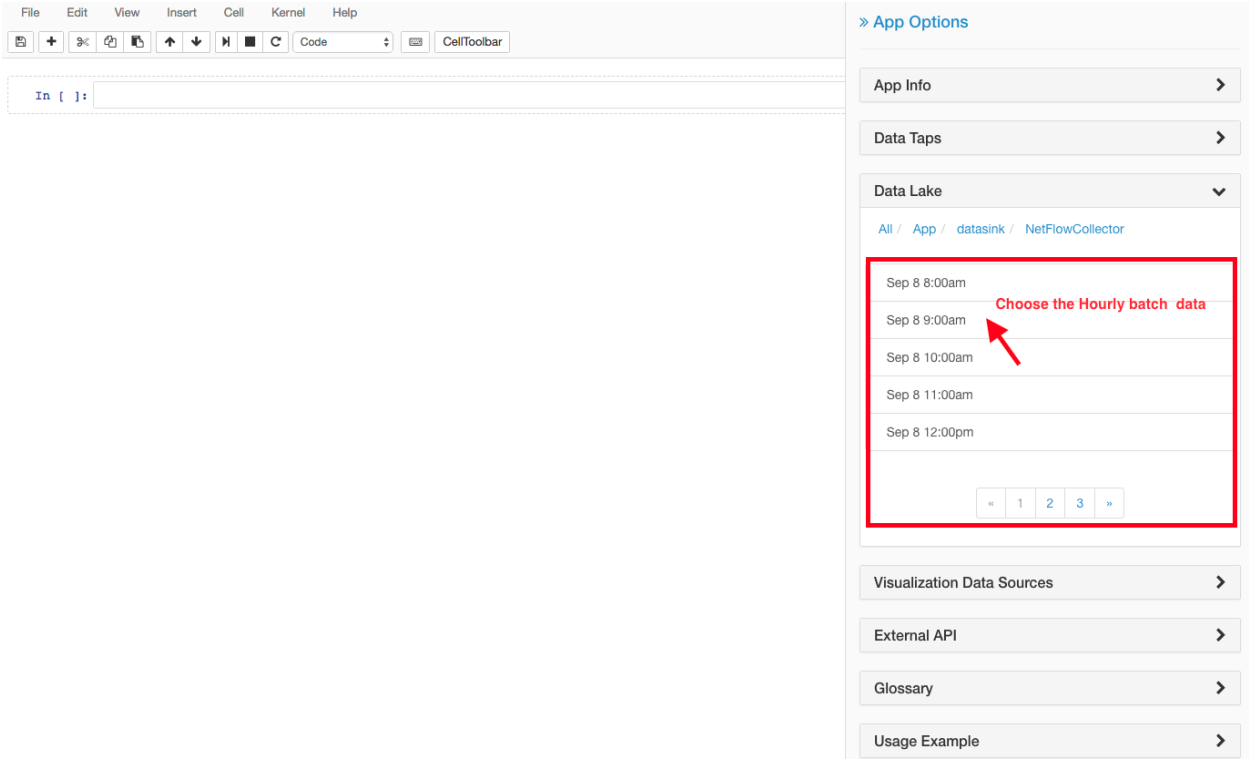


Fig. 9.2.3.3.6: Datasink

Reading the data from a UserApp

Click on the Usage Example section and choose the Read API

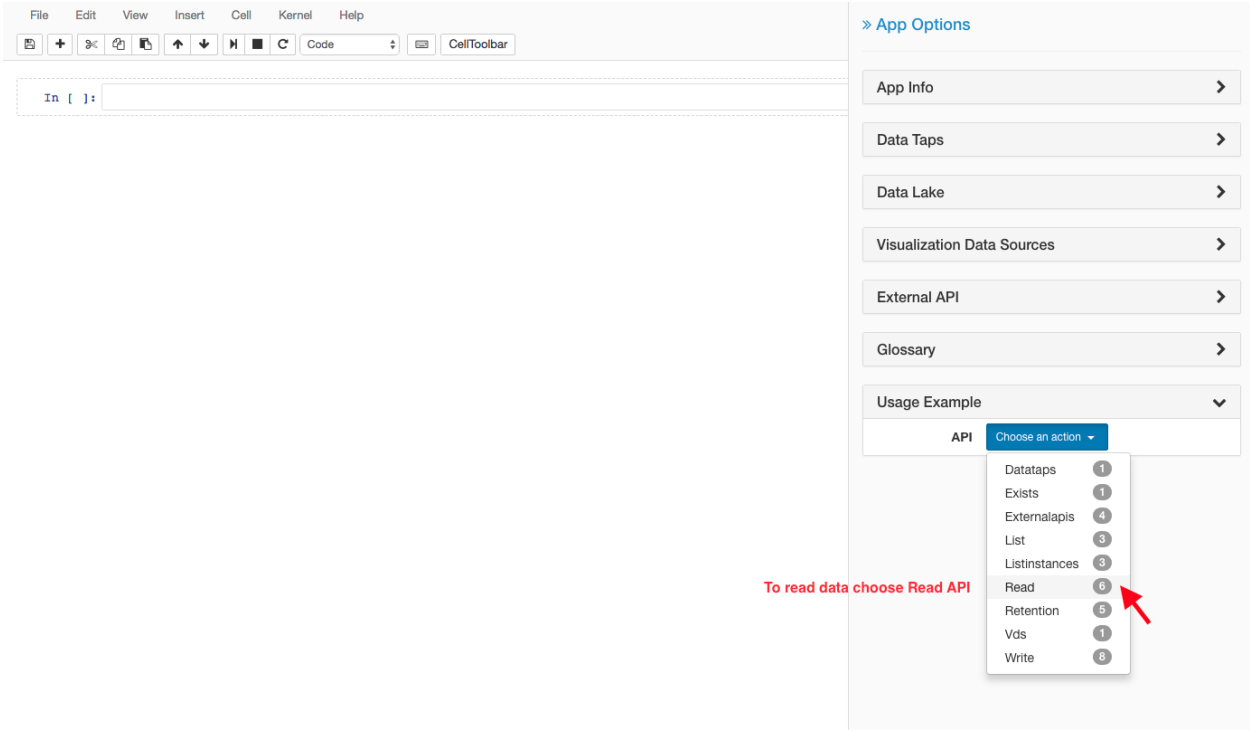


Fig. 9.2.3.3.7: Example-1

Go to the relevant API call

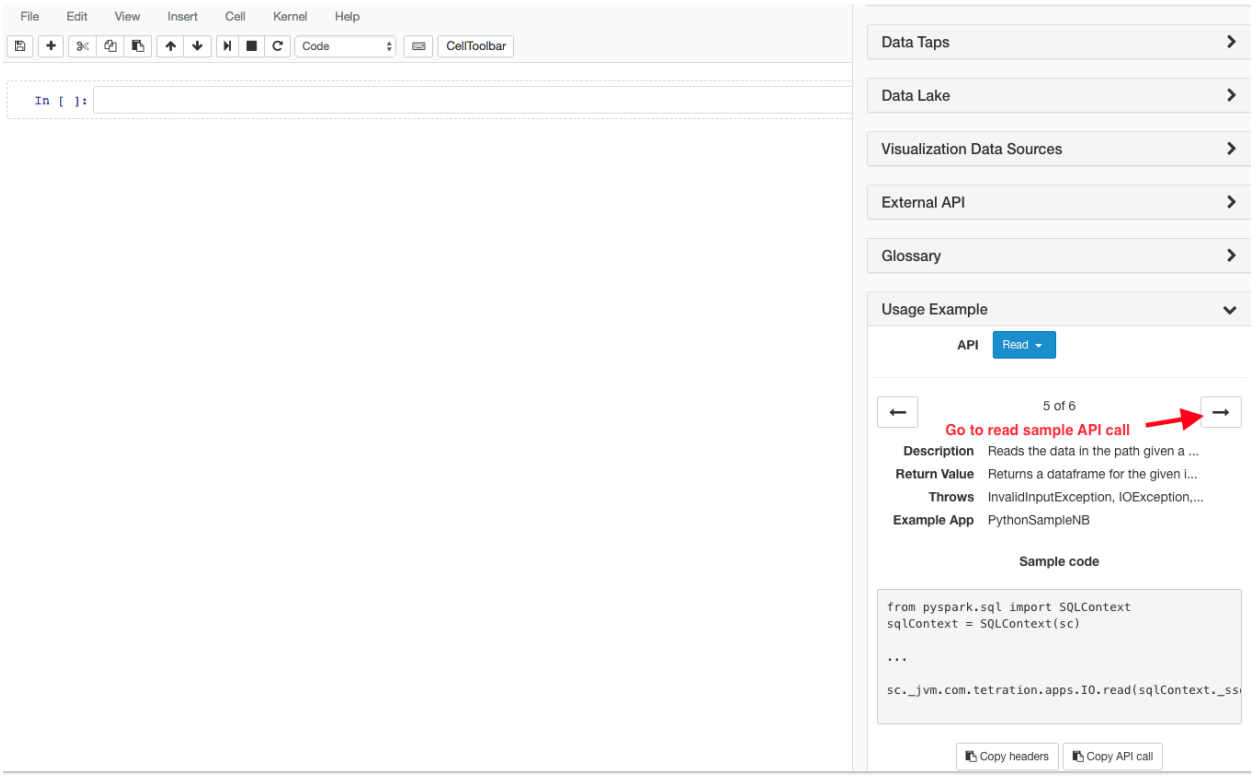


Fig. 9.2.3.3.8: Example-2

Copy Header and API call from Usage Example and paste in the App.

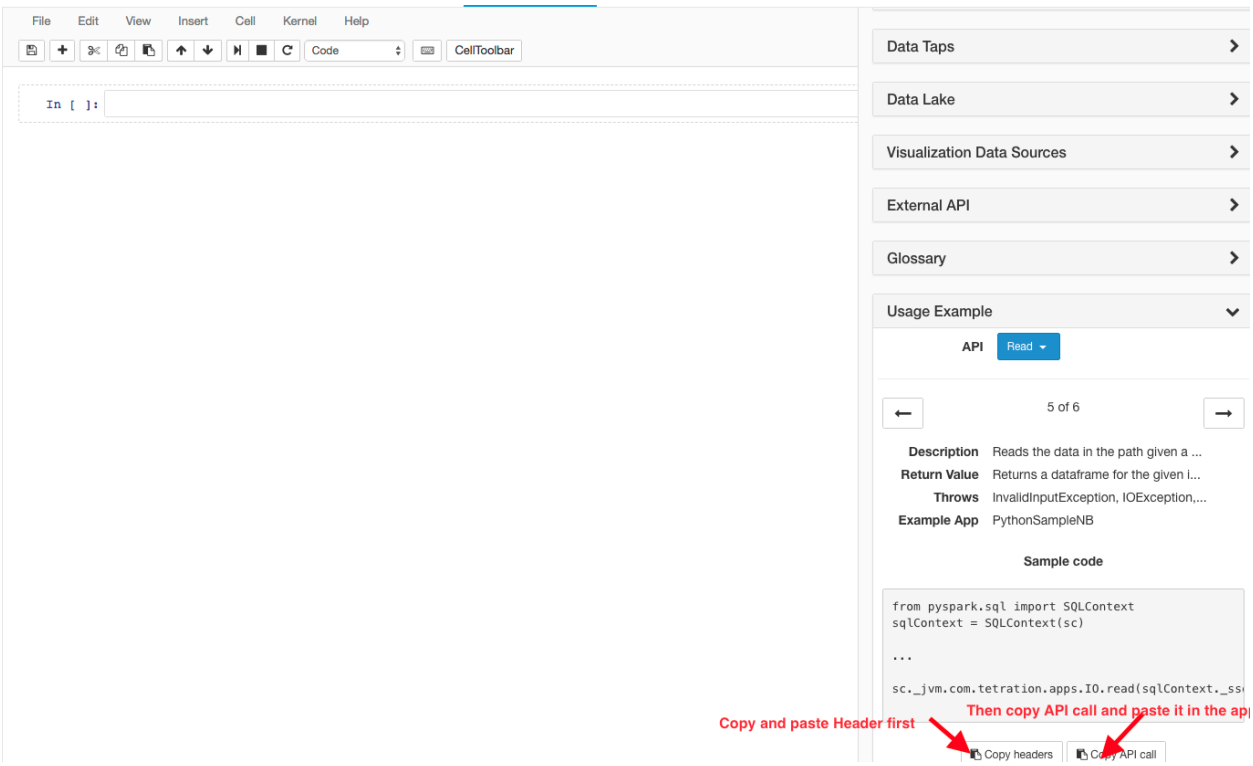
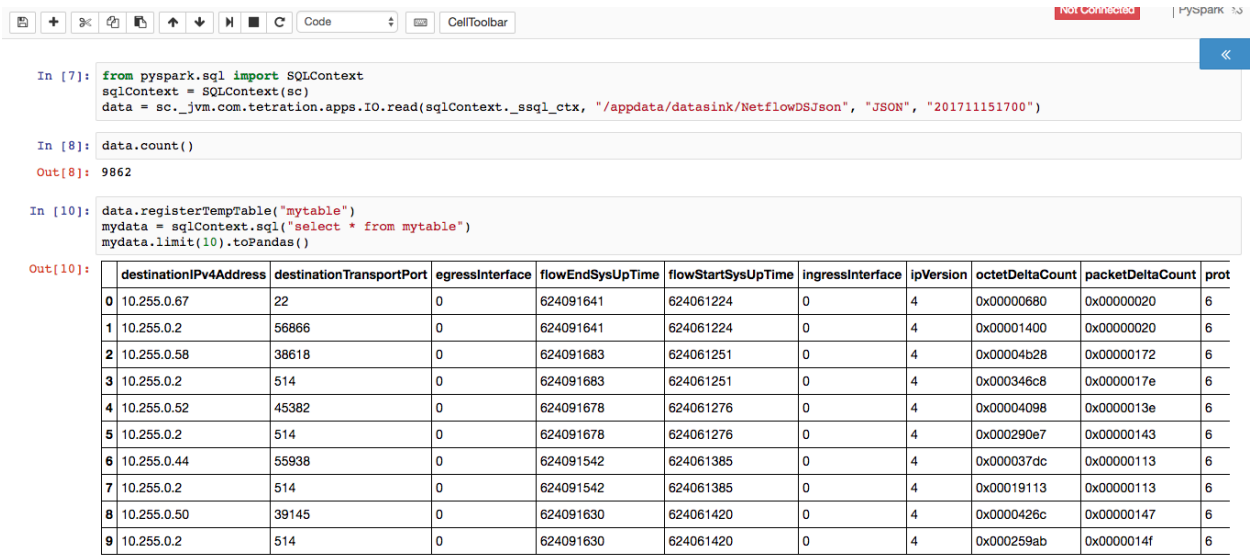


Fig. 9.2.3.3.9: Example-3

You can read the data as shown in the following User App example



	destinationIPv4Address	destinationTransportPort	egressInterface	flowEndSysUpTime	flowStartSysUpTime	ingressInterface	ipVersion	octetDeltaCount	packetDeltaCount	prot
0	10.255.0.67	22	0	624091641	624061224	0	4	0x00000680	0x00000020	6
1	10.255.0.2	56866	0	624091641	624061224	0	4	0x00001400	0x00000020	6
2	10.255.0.58	38618	0	624091683	624061251	0	4	0x00004b28	0x00000172	6
3	10.255.0.2	514	0	624091683	624061251	0	4	0x000346c8	0x0000017e	6
4	10.255.0.52	45382	0	624091678	624061276	0	4	0x00004098	0x0000013e	6
5	10.255.0.2	514	0	624091678	624061276	0	4	0x000290e7	0x00000143	6
6	10.255.0.44	55938	0	624091542	624061385	0	4	0x000037dc	0x00000113	6
7	10.255.0.2	514	0	624091542	624061385	0	4	0x00019113	0x00000113	6
8	10.255.0.50	39145	0	624091630	624061420	0	4	0x0000426c	0x00000147	6
9	10.255.0.2	514	0	624091630	624061420	0	4	0x000259ab	0x0000014f	6

Fig. 9.2.3.3.10: Datasink dumper user app example

9.2.3.4 Lookout Annotation

The Lookout Annotation App provides several aspects:

1. Creating inventory tags matching ip threat lists. These lists contain ip addresses and subnets published by external resources regarding things like C&C servers.
2. Alerting on flows which have been tagged as matching the inventory specified. This could be the aforementioned threat tags or user uploaded *lookout_* annotations.

There is 1 active threat source: Bogon. By default this will be disabled. The Zeus source is removed.

Note: As of 3.3, the Zeus and Bogon tags are no longer part of the User Annotation space. Upon upgrade to 3.3 these annotations may become disabled. Double-check that all desired annotations are enabled.

Note: As of 3.5, Zeus tags are no longer available. Any inventory filter and policy related to zeus tag will be ineffective, suggesting to remove related filters and policies so there are no stale data but this will not break anything.

Otherwise, users do not need to take any other action.

Warning: Although these tags are no longer shown in the User Annotation space, the number of subnets will still affect the global subnet limits.

Making use of Lookout Annotation

To give some context to how the Lookout Annotation tags can be used, consider the following simplified example where Workloads and Endpoints are communicating with each other and with external services.

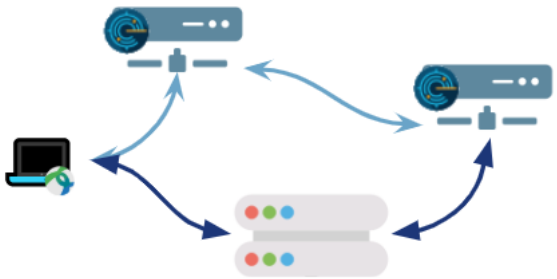


Fig. 9.2.3.4.1: Workloads and Endpoints communicating with each other and external services

With each updated threat feed, Lookout Annotation can create tags for known bad ips, and/or identify flows where communication occurred with a known threat.

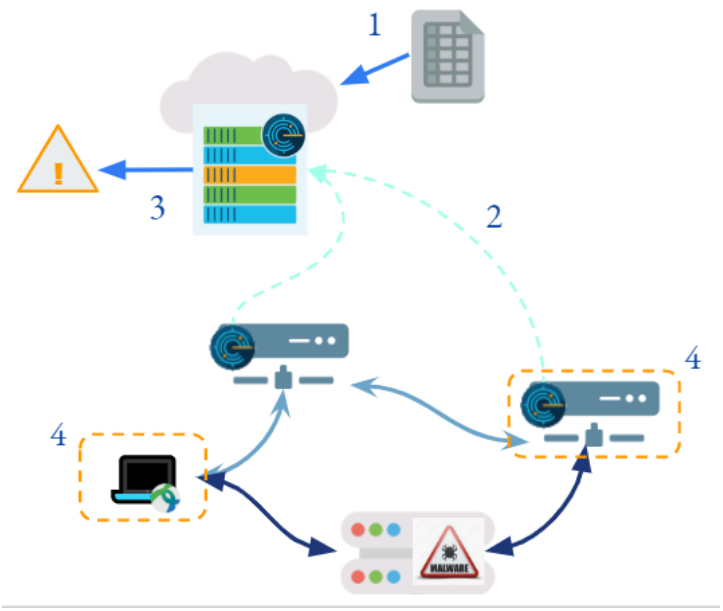


Fig. 9.2.3.4.2: Tetration receives an updated threat feed (1), which allows it to identify an external service as a threat. When flows are collected (2), connections to this known threat can be identified and alerts can be created and sent (3)

For flows seen connecting to a known threat, we'll consider those as direct connections.

While alerting on either of such communications is one option. Another option is to create policies directly blocking communication of these sorts.

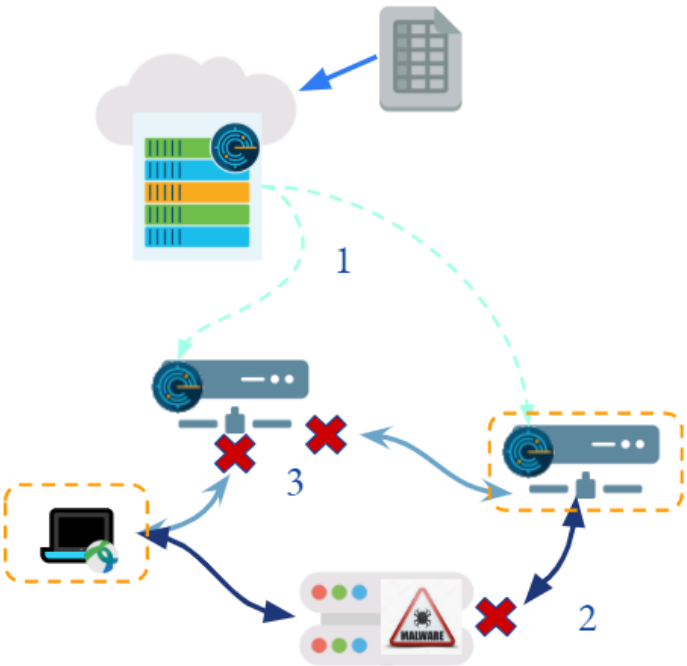


Fig. 9.2.3.4.3: By creating creating policies involving threat tags, communication can be blocked. (1) Pushing updated policy. (2) Directly blocking communication to a known threat ip.

A potential non-threat use case for Lookout Annotation is to upload *lookout_*-prefixed user annotations. Direct connection alerts on these tags can be created. This option for user annotations could be used for example when decommisioning workloads, and wishing to be alerted if there is still communication to or from these workloads.

Navigating the Lookout Annotation App

Enabling the Lookout Annotation App

The Lookout Annotation App is automatically enabled on new Root Scopes, up to a limit of 256 Root Scopes, but the annotations will be disabled by default. The app can also be enabled manually through the App Store.

To enable the app manually click on Enable button on **Data Platform** → **Tetration Apps**

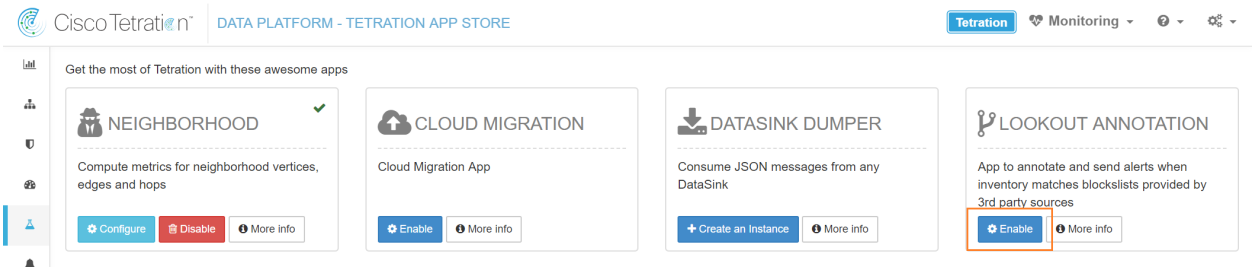


Fig. 9.2.3.4.4: Enabling Lookout Annotation from 'App Store'

Click on **Continue** to enable the app

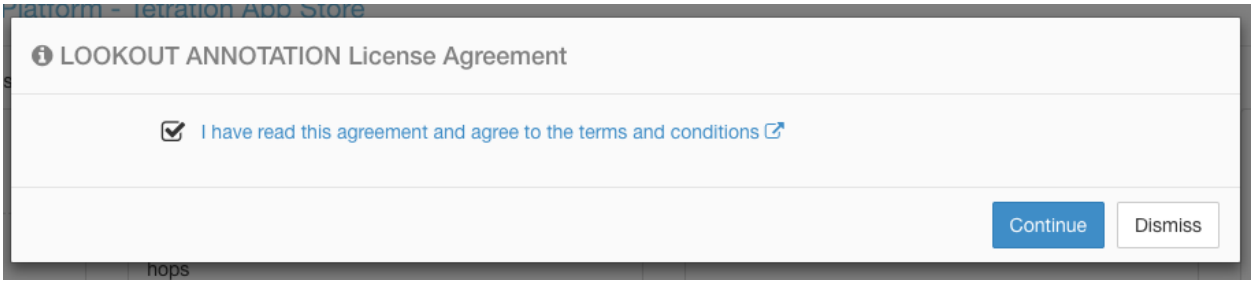


Fig. 9.2.3.4.5: Accepting EULA to enable the app

Accessing Lookout Annotation App from Tetration Apps page.

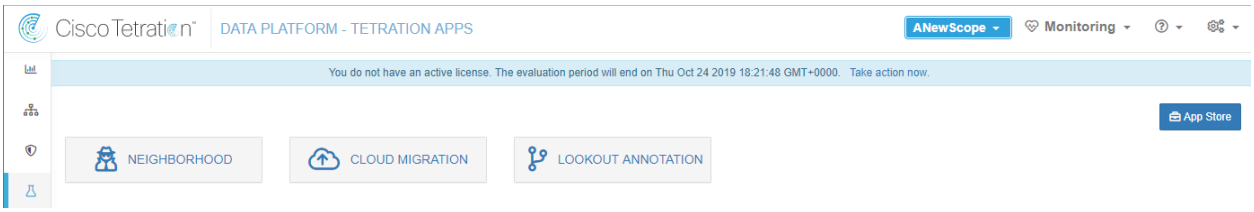


Fig. 9.2.3.4.6: Accessing Lookout Annotation App from Tetration Apps page

This page can also be accessed from **Security** → **Lookout**

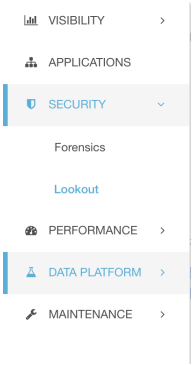


Fig. 9.2.3.4.7: Lookout Annotation under Security menu

Lookout Annotation App Page

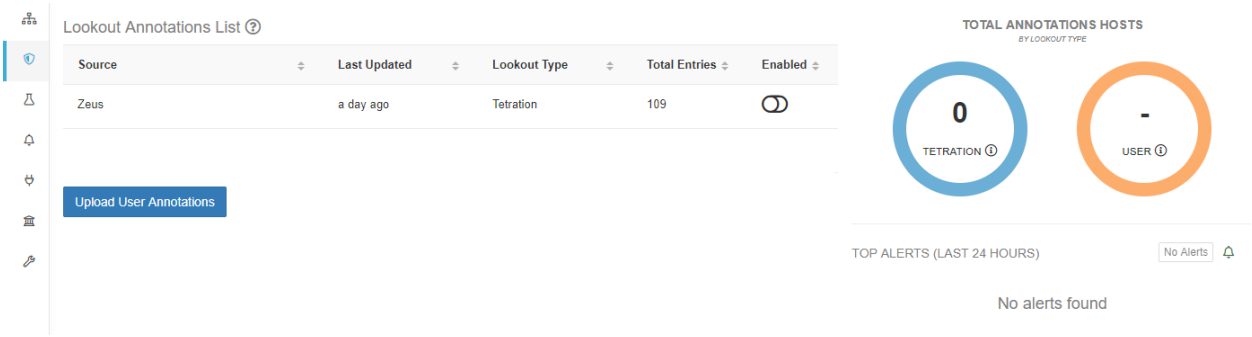


Fig. 9.2.3.4.8: Default Lookout Annotation App page after creating a new root scope (and initial annotations are updated). Note that the default sources are disabled by default.

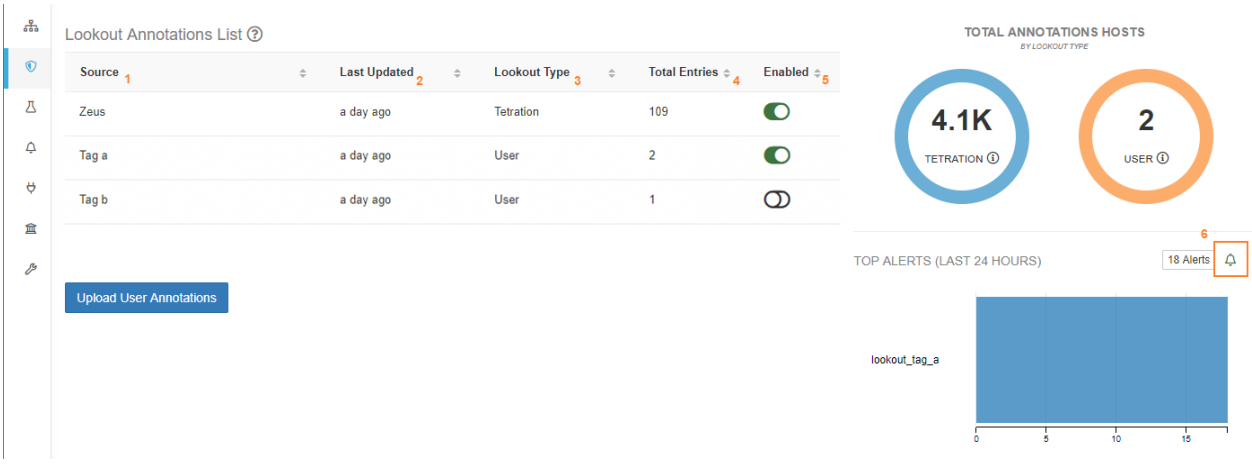


Fig. 9.2.3.4.9: Lookout Annotation App Page after enabling default sources and adding user `lookout_` tags

1. Data source for the tags
2. Time the annotation app last saw an update for this source. The annotation app runs every 24 hours.

3. `Tetration` indicates that this tag was updated from the UAS by Tetration. `User` indicates tags uploaded by the user with prefix of `lookout_`
4. How many IP/Subnets exists with this source
5. This button can be used to enable/disable tagging for each source
6. Open the alert configuration model to add alerts

The alerts tag shows the total number of lookout alerts generated. The chart shows the top n lookout alerts.

Lookout Annotation Alerts

Alerts can be configured using the *Alert Configuration Model*. See *Alert Configuration Modal* for general information about the model

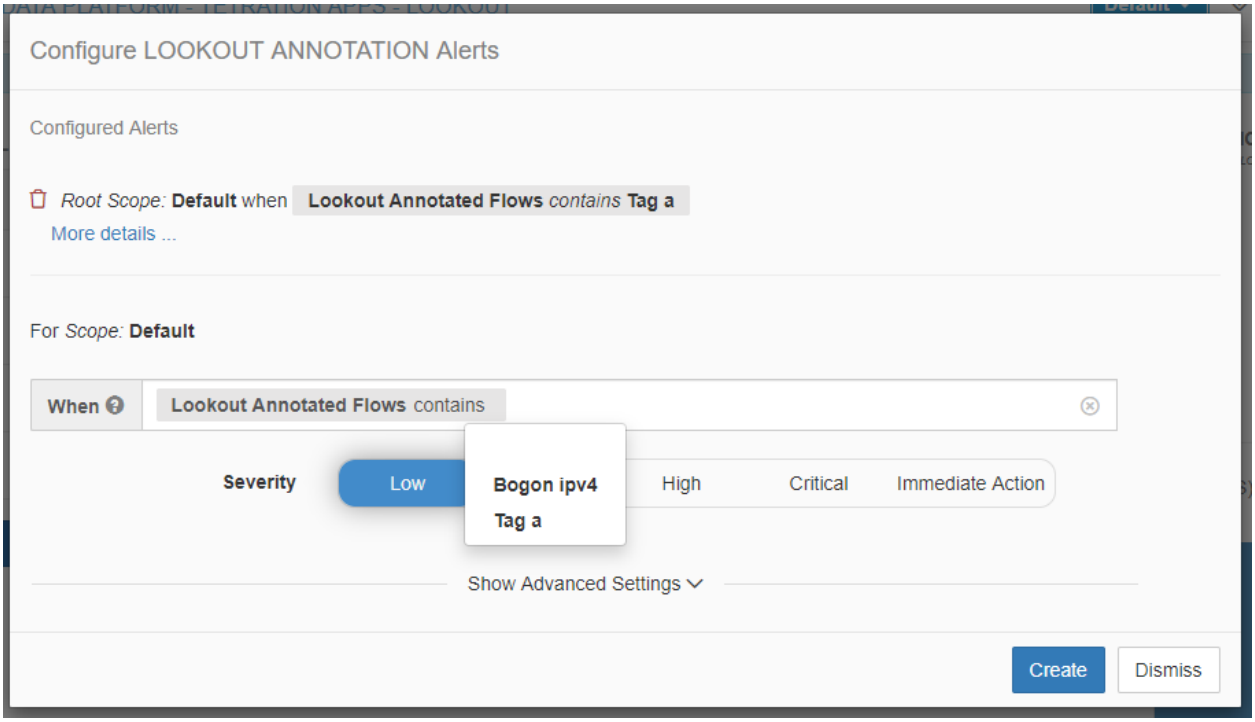


Fig. 9.2.3.4.10: Configuring alert

> **Warning:** If a tag is disabled, and there was an alert configured on that tag, the configured alert will not be deleted, but will also not be able to generate any alerts. Please review configured alerts periodically to ensure they remain relevant.

As of 3.4, configured alerts will no longer show in the configuration modal, and are only shown on the alert configuration page.

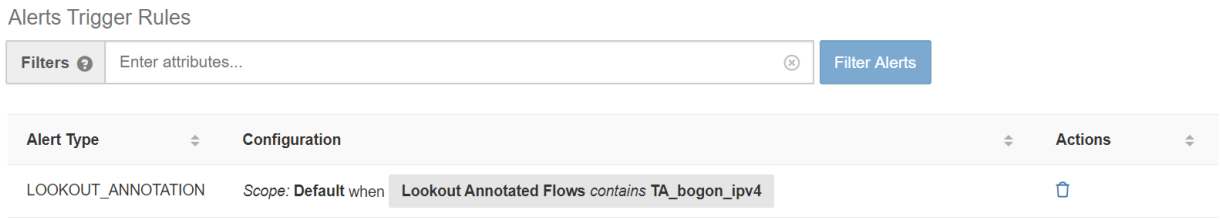


Fig. 9.2.3.4.11: Viewing configured alerts on the Alert Configuration page.

When flows are found with the matching tags, alerts will be sent to a Data Tap, and can be viewed in the UI under *Alerts → Current Alerts*. See *Current Alerts* for more information about the Alerts page.

Lookout Annotation UI Alert Details

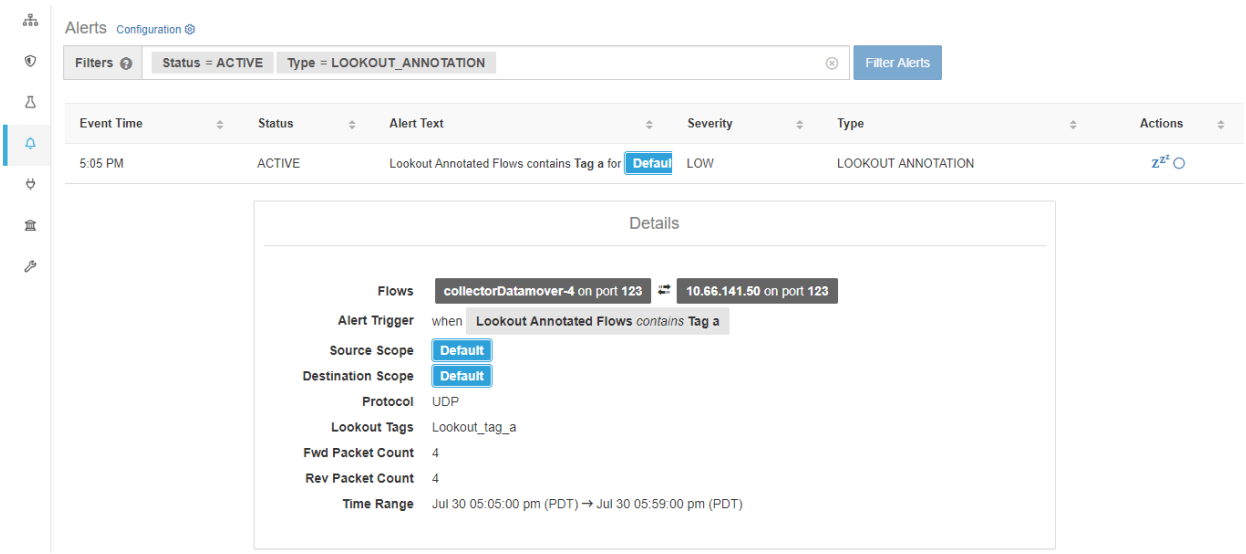


Fig. 9.2.3.4.12: Alert details

The above image shows the alert details for Lookout Annotation. On clicking on the flows, you can reach the flow search page for this particular flow.

Lookout Alerts Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* field is structured and will contain the following subfields for lookout alerts.

Field	Type	Explanation
lookout_tags	list	Tags could be threat tags, or user uploaded tags which are from source scope or destination scope
scope_id	string	Configured scope under which to search for flows matching condition
src_scope_id	list	List of all scope ids associated with the src_address
src_scope_names	list	List of all scope names associated with the src_address
src_address	string	Consumer address
src_hostname	string	Consumer hostname
src_port	int	Consumer port
dst_scope_id	list	List of all scope ids associated with the dst_address
dst_scope_names	list	List of all scope names associated with the dst_address
dst_address	string	Provider address
dst_hostname	string	Provider hostname
dst_port	int	Provider port
protocol	string	Flow transmitted rules
fwd_packet_count	long	Total counts of forward packets across all flows being aggregated
rev_packet_count	long	Total counts of reverse packets across all flows being aggregated
internal_trigger	string	Configuration query which triggered the alert
time_range	list	First and last batch timestamps seen from the aggregated flow data

Example of Alert Details

After alert_details is parsed as json (unstringified), then it would look like following

```
{
  "alertDetails": {
    "dst_scope_id": [
      "5efcfdf5497d4f474f1707c2"
    ],
    "dst_scope_names": [
      "Default"
    ],
    "dst_hostname": "",
    "src_scope_id": [
      "5efcfdf5497d4f474f1707c2"
    ],
    "lookout_tags": [
      "TA_zeus"
    ],
    "dst_address": "224.0.0.252",
    "fwd_packet_count": 2,
    "src_scope_names": [
      "Default"
    ],
    "src_port": 52986,
    "protocol": "UDP",
    "internal_trigger": {
      "datasource": "lookout_annotation",
      "rules": {
        "field": "lookout_tags",
        "type": "contains",
        "value": "TA_zeus"
      },
```

(continues on next page)

(continued from previous page)


```
      "label": "Alert Trigger"
    },
    "scope_id": "5efcfdf5497d4f474f1707c2",
    "time_range": [
      1595023680000,
      1595023740001
    ],
    "src_address": "172.26.230.139",
    "dst_port": 5355,
    "rev_packet_count": 0,
    "src_hostname": ""
  }
}
```


Uploading and Removing user tags for Lookout Annotation

User can upload/remove tags with `lookout_` prefix. See ../../inventory/upload for details.

Note: If two tags are uploaded with same name(example 'ABC' and 'abc') in inventory upload, the lookout annotation pipeline would not be able to push lookout annotation metrics for user tags for that particular root scope.

User can clear the `lookout_` annotation tags they added. See ../../inventory/upload for details.

Note: As of 3.3, user will no longer be able to edit `TA_zeus` and `TA_bogon_ipv4` tags.

Using annotations by Lookout Annotation App

Flow Search and Profile Pages

Lookout Annotation tags from threat sources show up as * *TA* tags, such as * *TA Bogon Ipv4*

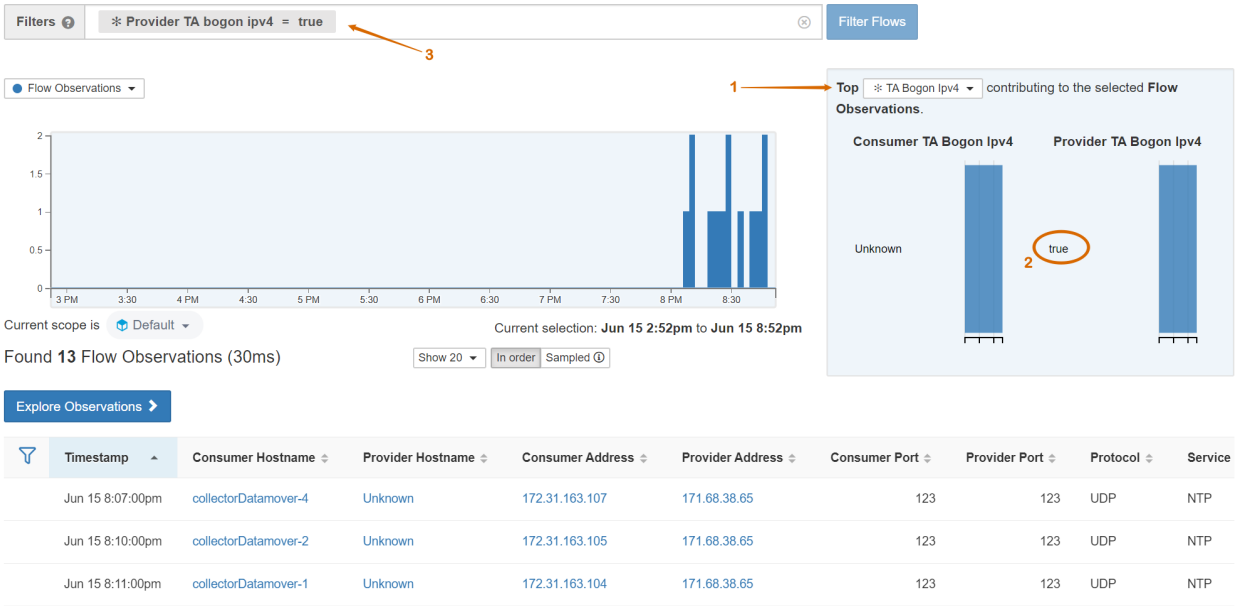


Fig. 9.2.3.4.13: From 'Top' dropdown select the TA tag (1). Those annotated as such, will have *true* or other known tag value (2); clicking this can add this selection to the filter drill-down. Flows filtered to those matching direct threat connections (3).

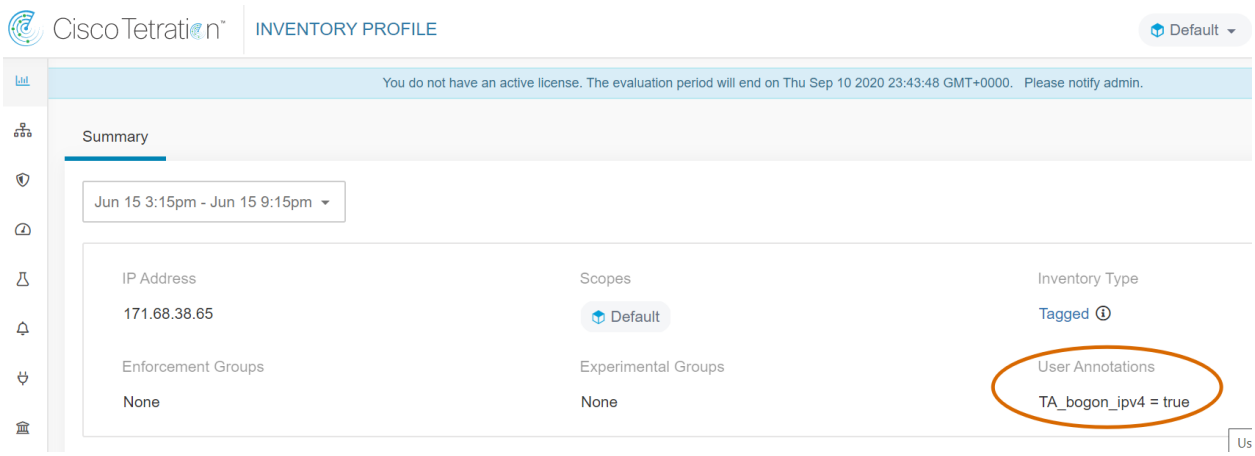


Fig. 9.2.3.4.14: Inventory Profile page for an ip that was annotated as a threat by Lookout Annotation

Filters and Policy

Note: Zeus tags are removed in 3.5. Any inventory filter and policy related to the zeus tag will be ineffective. Users can chose to clean up filters and any related policy if needed but this will not break anything.

To use the tags created by Lookout Annotation in a policy, an inventory filter based on the tag must be created first.

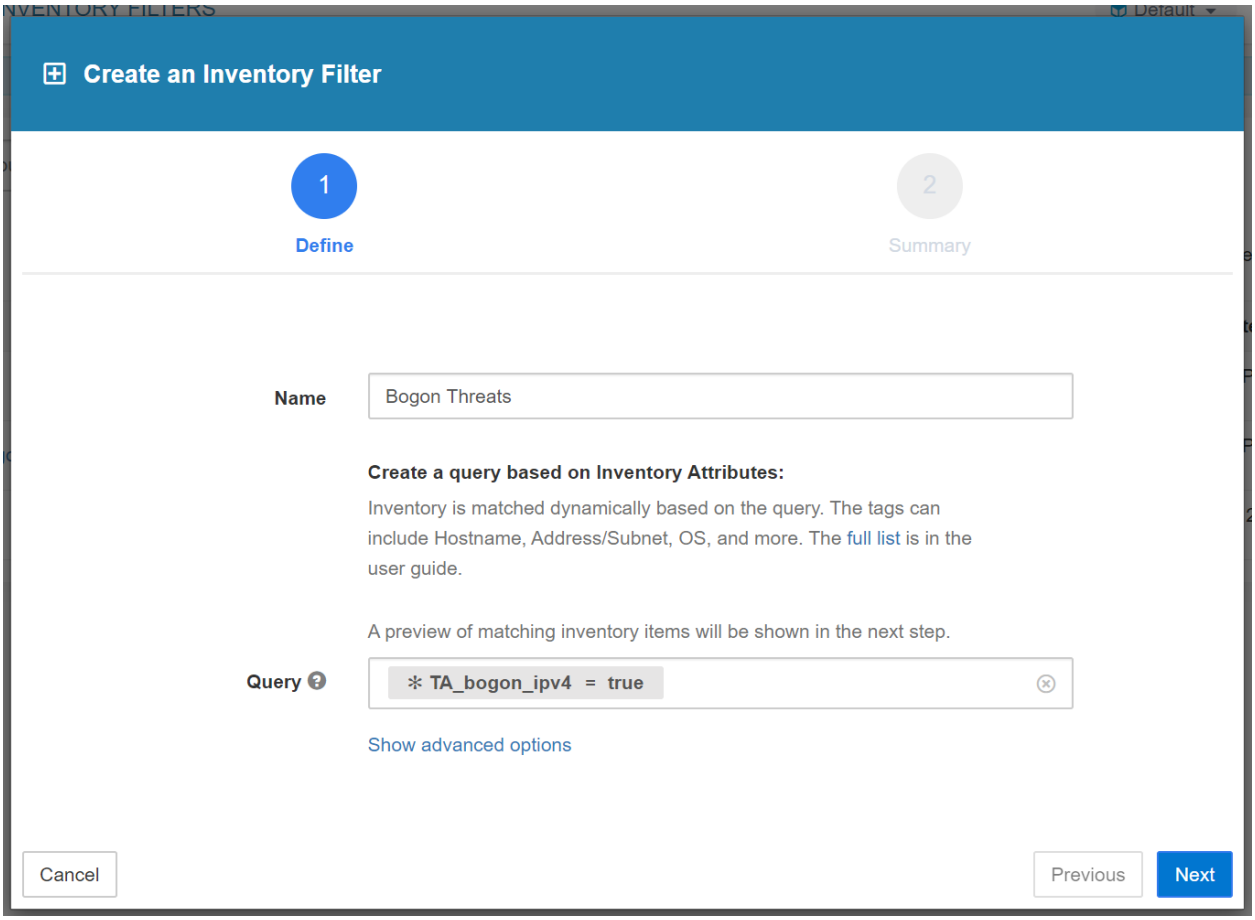


Fig. 9.2.3.4.15: Creating an inventory filter for a threat tag.

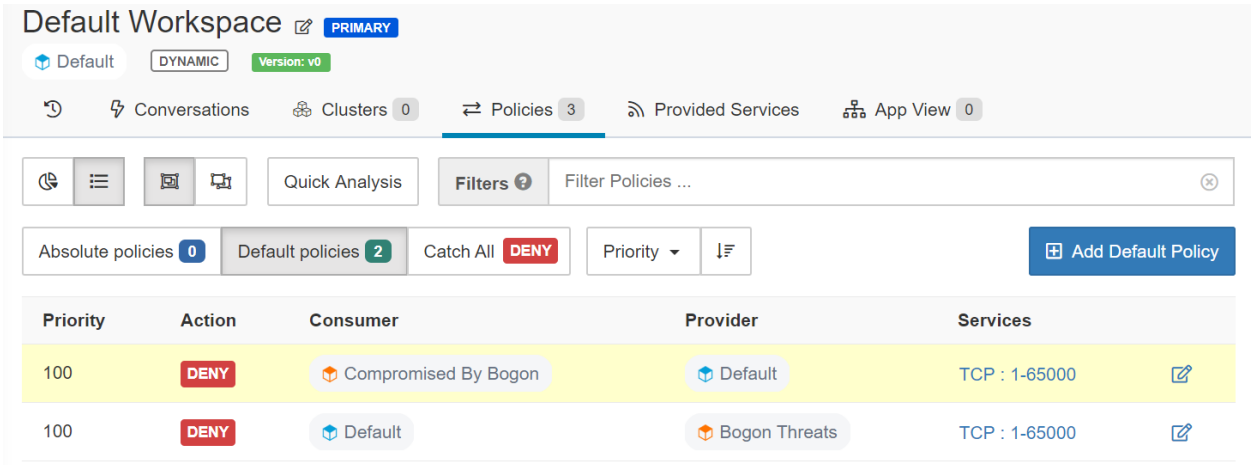


Fig. 9.2.3.4.16: Application workspace setting Deny policies with direct connection tag and compromised tag. Warning: creating a direct connection policy could result in a large number of ips or subnets to be blocked being pushed to the workload.

Clearing Annotations by Lookout Annotation

Lookout Annotation threats are automatically updated with new threat data: new ips will be added and old ips will be removed. Enable 'Tetration Cloud Connection' to make sure you have the latest up-to-date data; see *Threat Intelligence*

To completely remove all Lookout Annotation tags: disable Lookout Annotation from the Tetration App Store.

9.3 User Apps

User applications help bridge the gap between the collected data set and users who want to do more with them. In other words, users can write their own custom applications to apply analytics/machine learning algorithm or tasks to achieve their goal, i.e. **Data Visualization**, **Data Exploration**, **Implement Data Dependent Workflows**.

Any changes made to an example application may be reverted. We recommend users to clone an existing example application before making changes if they want to prevent accidental resets. To manually trigger a reset of the sample applications (and update to latest samples apps after an upgrade), see *Settings*.

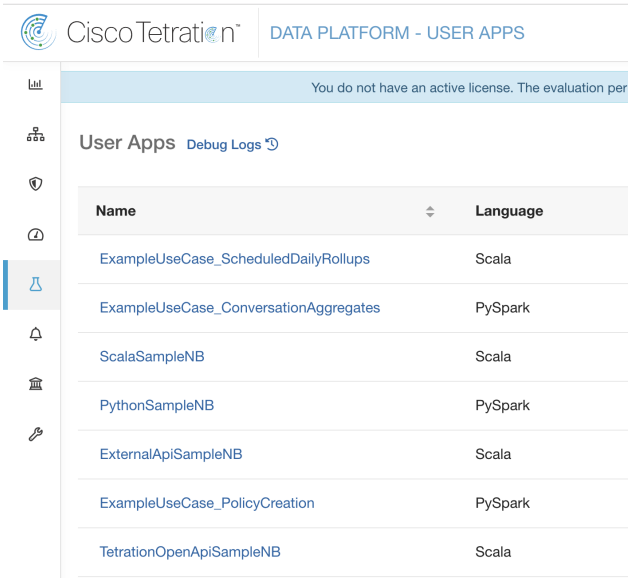


Fig. 9.3.1: Sample applications provided by Tetration

When editing a user app, the user can expand the app sidebar on the right to get help about the supported APIs to perform specific tasks. See *Using App Sidebar*.

9.3.1 Spark 2+ changes for User Apps

> **Warning:** Spark was upraded to spark version 2.3 in Tetration 3.3. This may require some changes to existing User Apps.

Potential changes needed in User Apps after upgrading to Tetration 3.3

1. Get the sqlContext from the SparkSession/SparkContext

- Scala

```
val sqlContext = spark.sqlContext
```

- Python

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

2. Other potential differences between Spark 1.6 and Spark 2.3
   - Change `.map` to `.rdd.map` in Python code. Example:

```
# The following
sqlContext.sql(query).map(tuple).collect()
# Should be changed to
sqlContext.sql(query).rdd.map(tuple).collect()``
```

9.3.2 Adding New Apps

Click on the 

button to create a new application, both application name and language are required inputs.

Note: Application names should be unique

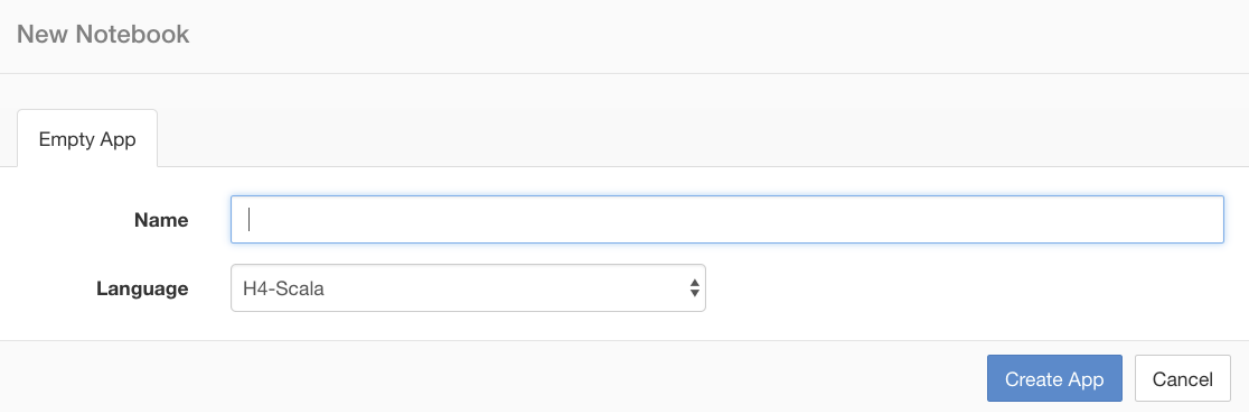


Fig. 9.3.2.1: New App

9.3.3 Cloning Existing Apps

Click on the clone  button to clone an existing application, the new application's name will be appended by the word 'Copy' by default.

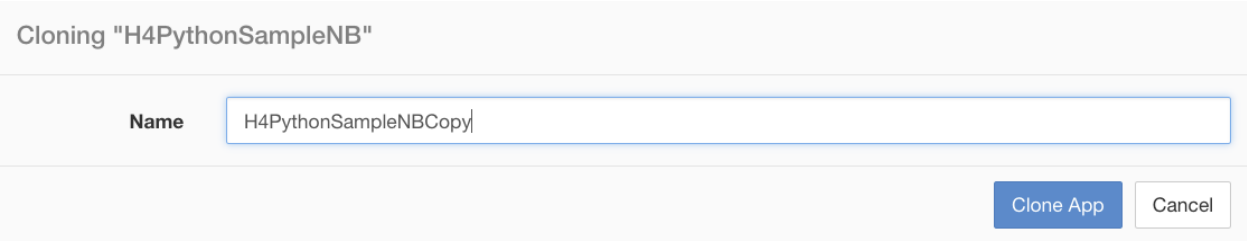


Fig. 9.3.3.1: Cloning Existing Apps

9.3.4 Import/Export User Apps

Apps can be exported by clicking on the export button in the user apps page.

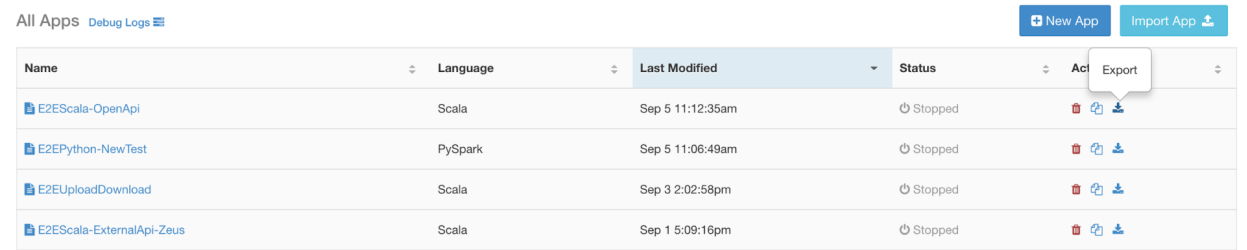


Fig. 9.3.4.1: Export App Icon

Importing Apps

1. Click on the Import App button.
2. Select the file to be uploaded. Only .ipynb files are allowed, and only those created by Tetration will work. Click on Open.
3. In the Upload screen, select Name and Language and click Upload.

Note: Name and Language are auto detected. User's can override Name and Language.

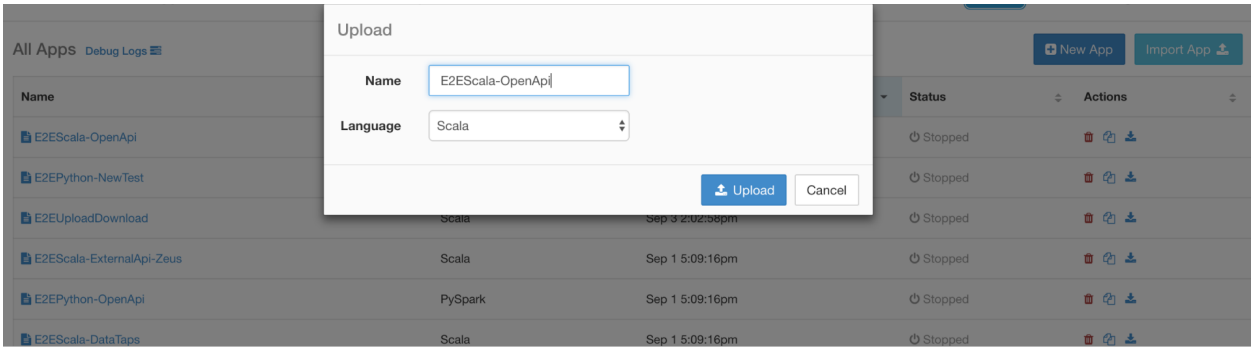


Fig. 9.3.4.2: Import App model

9.3.5 Using Apps

Open an existing application or create a new one to start using an application interactively. By default, user can expand the application sidebar on the right to get help about supported APIs to perform a specific task.

Note: We recommend users to utilize the sidebar to write their applications.

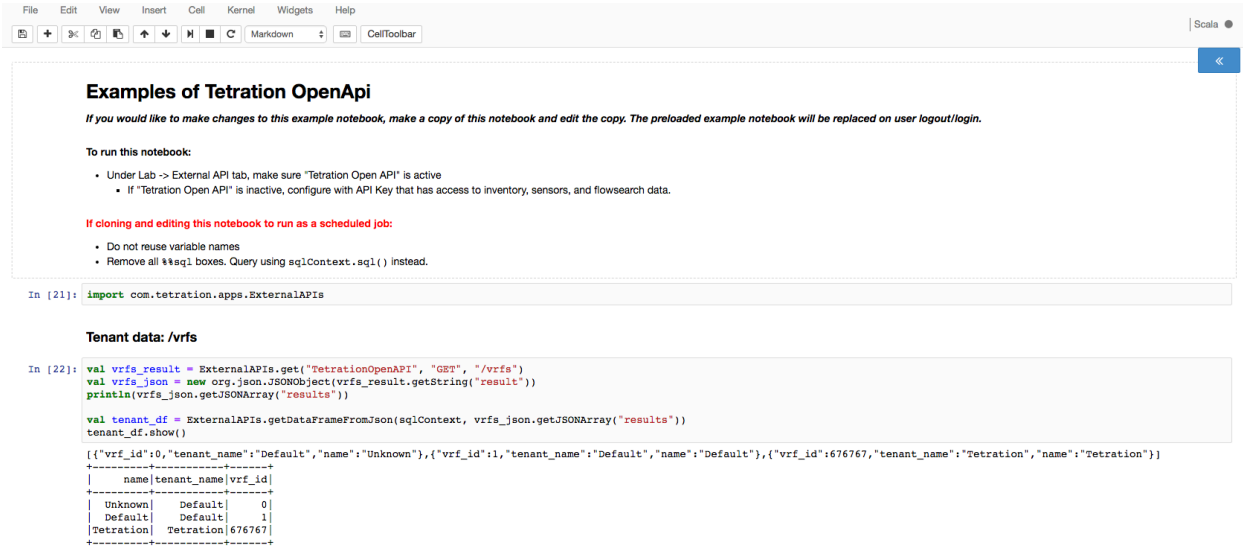


Fig. 9.3.5.1: App content

9.3.6 Using App Sidebar

Users can click on the « button on the top right corner, expand the **Usage Example** section and choose the appropriate API call helpers to be used in their applications.

The following shows how to use **Usage Examples** to get the desired API read call:

```
data = sc._jvm.com.tetration.apps.IO.read(sqlContext._ssql_ctx, "InputPath", "PARQUET
↪", "LAST24HOURS")
```

The following are the steps to get to the sample code above:

1. Choose the desired usage example from the API drop-down list.

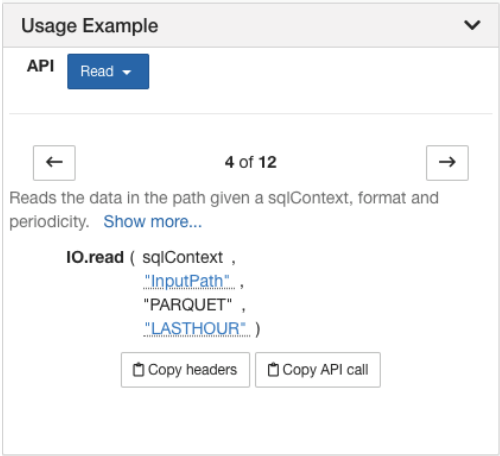


Fig. 9.3.6.1: Choose the desired usage example

2. Choose the desired data input path by clicking on "InputPath" and select a path.

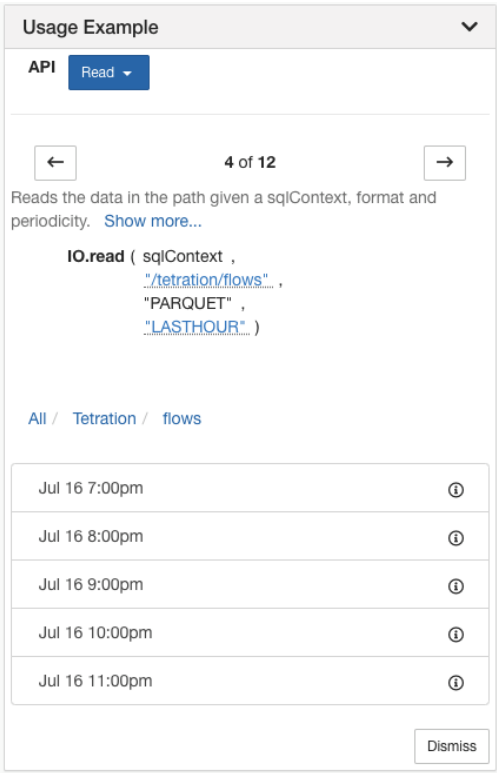


Fig. 9.3.6.2: Choose the desired data path

3. Choose the desired data period by clicking on "LASTHOUR" and choose a period.

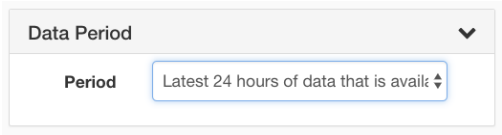


Fig. 9.3.6.3: Choose the desired data period

4. Click on the Dismiss button.
5. Click on the Copy API call and paste the content into the application cell.

9.3.7 Data Lake Data

See *Data Lake* and *Data Lake - Tetration Data* for information about Tetration provided Data Lake sources.

9.3.8 Debug Logs

This page reflects the container logs which help users to debug applications that are being run. External links typically point to **yarn application master** and **spark driver consoles** and may be available in the page depending on your role. Container logs list output in ascending order.

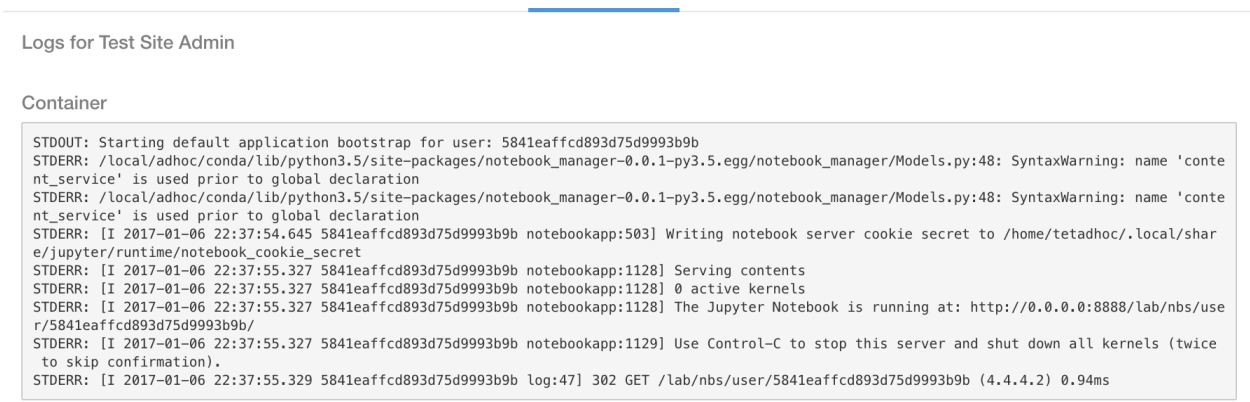


Fig. 9.3.8.1: Debug Logs

9.3.9 Software Version Supported

Package	Version
bokeh	0.12.13
kafka	> 0.10.1.x
matplotlib	2.1.1
numpy	1.11.3 (Scheduled jobs: 1.13.0)
pandas	0.21.1 (Scheduled jobs: 0.20.2)
patsy	0.4.1
scala	2.11.*
scikit-learn	0.19.1
scipy	1.0.0

Continued on next page

Table 9.3.9.1 – continued from previous page

Package	Version
seaborn	0.7.1
statsmodels	0.6.1
sympy	1.0
python	3.5

9.3.10 Additional Details about User App APIs

9.3.10.1 Sending Messages and Alerts from User Apps

Data Taps can be used to send messages and alerts from a user app. A 'message' is any generic json formatted string. An 'alert' is required to have a *severity* field. Messages will not show up in the UI, whereas alerts will show up in the UI (if under the 60/min limit).

Both messages and alerts use the same *DataTaps.sendMessage(message, datatap)* interface.

```
In [102]: import com.tetration.apps.DataTaps;
          val message = "{\"message_id\":\"1048576\",\"message_text\":\"some message\"}";
          val res = DataTaps.sendMessage(message, "Messages")
          println("Messages sent: " + res)

          Messages sent: 1
```

Fig. 9.3.10.1.1: Sending a message through a data tap.

Alerts require some fields to be set:

1. *alert_time* (if 0, it will be overridden with the time the alert was sent). Alert time will not be shown in the UI; instead *event_time* is shown in UI (see example below)
2. *severity*: Severity must match one of *LOW*, *MEDIUM*, *HIGH*, *CRITICAL*.

```
In [119]: val minimalAlert = """{
          "severity":"MEDIUM",
          "alert_time":0,
          "alert_text":"A user app alert"
          }""".filter(_ >= ' ')
          val res = DataTaps.sendMessage(minimalAlert, "Alerts")
          println("Alerts sent: " + res)

          Alerts sent: 1
```

Fig. 9.3.10.1.2: Sending an alert through a data tap.

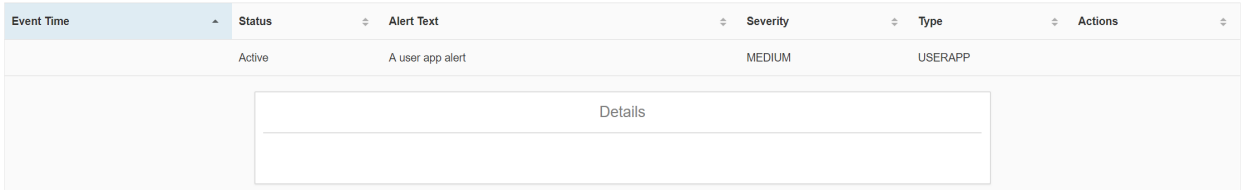


Fig. 9.3.10.1.3: Above minimal alert as seen in the UI. (As noted above, *alert_time* is not shown)

```
In [121]: val alert_with_details = """{
"severity":"HIGH",
"alert_text":"A user app alert with details",
"alert_time":0,
"event_time":1537602720000,
"alert_details":"{\"other_notes\":\"Some additional info\"}"
}""".filter(_ >= ' ');
val res = DataTaps.sendMessage(alert_with_details, "Alerts")
println("Alerts sent: " + res)

Alerts sent: 1
```

Fig. 9.3.10.1.4: An alert with more details: *event_time* and *alert_details*

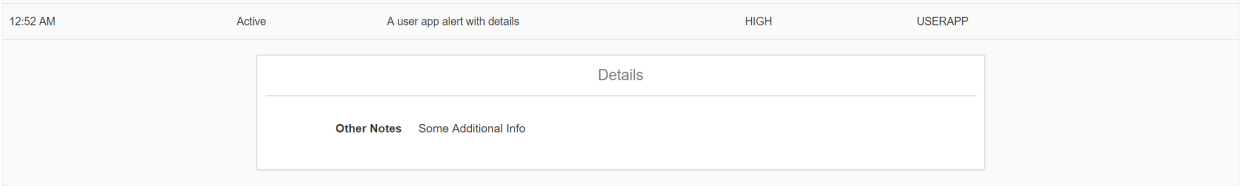


Fig. 9.3.10.1.5: Event time is shown in the UI. Alert details are shown after clicking the alert to expand.

Note: A maximum of 60 User App alerts will be sent per minute. For UI visibility of the sent alerts, a maximum of 60 per minute per root scope will apply across all alerts (User Apps and Tetration alert combined, with preference based on severity).

Warning: No link from User Apps to Internal and External Data Taps is shown in the Alert Configuration page. If a DataTap is available, it can be used by the UserApp.

9.3.10.2 External API Connections Debugging

External API fetches additional data by creating API key and then configuring Tetration OpenAPI or Zeus endpoint. Here are some examples of error messages which are displayed on the user apps for troubleshooting endpoint connectivity issues.

Example symptoms include:

- *Inactive Tetration OpenAPI*
- *Invalid OpenAPI Key*
- *Unreachable Server*
- *Invalid Endpoint*
- *Query Params Mistyped or Missing*
- *Invalid Payload*

Inactive Tetration OpenAPI

TetrationOpenAPI is inactive and should get configured.

Tenant data: /vrfs

```
val vrfs_result = ExternalAPIs.get("TetrationOpenAPI", "GET", "/vrfs")
val vrfs_json = new org.json.JSONObject(vrfs_result.getString("result"))
val vrfs_json_array = vrfs_json.getJSONArray("results")
println(vrfs_json_array)

val tenant_df = ExternalAPIs.getDataFrameFromJson(sqlContext, vrfs_json_array)
tenant_df.show()
```

```
Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 400, error message: External API URL could not be formed correctly,
because TetrationOpenAPI is not configured
StackTrace:   at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:42)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
```

Fig. 9.3.10.2.1: TetrationOpenAPI needs to get configured.

Invalid OpenAPI Key

TetrationOpenAPI is configured, however the attached API key is not valid.

Tenant data: /vrfs

```
val vrfs_result = ExternalAPIs.get("TetrationOpenAPI2", "GET", "/vrfs")
val vrfs_json = new org.json.JSONObject(vrfs_result.getString("result"))
val vrfs_json_array = vrfs_json.getJSONArray("results")
println(vrfs_json_array)

val tenant_df = ExternalAPIs.getDataFrameFromJson(sqlContext, vrfs_json_array)
tenant_df.show()
```

```
Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 401, error message: OpenAPIClient Request Failed, OpenApi returned: Unauthorized
StackTrace:   at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:42)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
```

Fig. 9.3.10.2.2: API key needs to get configured.

Unreachable Server

Here are two possible situations when server is unreachable.

Zeus Connection Timeout

External service is unreachable, should check whether Outbound HTTP is enabled, see *Company*.

Zeus external api

Fetching zeus IP blocklist data and comparing to recent flow data.

```
/* Fetch external ip blocklist data */
val blocklist_result = ExternalAPIs.get("Zeus", "IP Blocklist")
val ip_blocklist = blocklist_result.getString("result").split(",")
println("Number of ips in blocklist: " +  ip_blocklist.length)
println("Ex. First 10 ips in list: " + ip_blocklist.take(10).deep.mkString(", "))
```

```
Name: com.tetration.apps.exceptions.PublisherProxyException
Message: Failed to send request to proxy service, status code is 500 - Internal Server
Error, error message: Proxy service may be unable to reach external service. Timed out
trying to reach external service. Retry or double-check proxy settings
StackTrace:   at com.tetration.apps.PublisherProxyClient.errorMsgParity(PublisherProxy
Client.java:166)
  at com.tetration.apps.PublisherProxyClient.getAPI(PublisherProxyClient.java:45)
  at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:30)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:17)
```

Fig. 9.3.10.2.3: connection timeout.

Publisher Proxy Down

External API sends requests through publisher proxy which is down. Users should retry sending requests and check publisher proxy status.

Tenant data: /vrfs

```
val vrfs_result = ExternalAPIs.get("TetrationOpenAPI", "GET", "/vrfs")
val vrfs_json = new org.json.JSONObject(vrfs_result.getString("result"))
val vrfs_json_array = vrfs_json.getJSONArray("results")
println(vrfs_json_array)

val tenant_df = ExternalAPIs.getDataFrameFromJson(sqlContext, vrfs_json_array)
tenant_df.show()
```

```
Name: com.tetration.apps.exceptions.PublisherProxyException
Message: Failed to send request to proxy service, status code is 503 - Service Unavailable, error message:
Adhoc proxy service may be unreachable; Please retry
StackTrace:   at com.tetration.apps.PublisherProxyClient.errorMsgParity(PublisherProxyClient.java:163)
  at com.tetration.apps.PublisherProxyClient.getAPI(PublisherProxyClient.java:45)
  at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:30)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
```

Fig. 9.3.10.2.4: publisher proxy down.

Invalid Endpoint

1. External API should be **TetrationOpenAPI** instead of **TetrationOpen**.

Tenant data: /vrfs

```
val vrfs_result = ExternalAPIs.get("TetrationOpen", "GET", "/vrfs")
val vrfs_json = new org.json.JSONObject(vrfs_result.getString("result"))
val vrfs_json_array = vrfs_json.getJSONArray("results")
println(vrfs_json_array)

val tenant_df = ExternalAPIs.getDataFrameFromJson(sqlContext, vrfs_json_array)
tenant_df.show()
```

```
lastException = null

Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 400, error message: External API URL could not be formed correctly,
because ExternalAPI: TetrationOpen is incorrect
StackTrace:   at com.tetration.apps.PublisherProxyClient.errorMsgParity(PublisherProxyClien
t.java:159)
  at com.tetration.apps.PublisherProxyClient.getAPI(PublisherProxyClient.java:45)
  at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:30)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
```

Fig. 9.3.10.2.5: invalid external api.

2. External API extension field should be **GET** instead of **post**.

Tenant data: /vrfs

```
val vrfs_result = ExternalAPIs.get("TetrationOpenAPI", "post", "/vrfs")
val vrfs_json = new org.json.JSONObject(vrfs_result.getString("result"))
val vrfs_json_array = vrfs_json.getJSONArray("results")
println(vrfs_json_array)

val tenant_df = ExternalAPIs.getDataFrameFromJson(sqlContext, vrfs_json_array)
tenant_df.show()
```

```
Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 400, error message: External API URL could not be formed correctly,
because ExternalAPIExtension: post is invalid
StackTrace:   at com.tetration.apps.PublisherProxyClient.errorMsgParity(PublisherProxyClien
t.java:159)
  at com.tetration.apps.PublisherProxyClient.getAPI(PublisherProxyClient.java:45)
  at com.tetration.apps.ExternalAPIsLib.get(ExternalAPIsLib.java:30)
  at com.tetration.apps.ExternalAPIs.get(ExternalAPIs.java:23)
```

Fig. 9.3.10.2.6: invalid external api extension field.

Query Params Mistyped or Missing

Query Params should be **flowsearch** instead of **flowsearchtype**.

```
val query_payload = new org.json.JSONObject("""{
    "T0": "2017-03-09T09:00:00-0700",
    "T1": "2017-03-09T19:00:00-0700",
    "scopeName": "Tetration",
    "limit": 10,
    "filter": {}
}""")
//Comment out the following two lines to use the above hard coded date/times rather than current time
query_payload.put("T0", hour_ago.format(DateTimeFormatter.ISO_OFFSET_DATE_TIME))
query_payload.put("T1", current_time.format(DateTimeFormatter.ISO_OFFSET_DATE_TIME))
println(query_payload)

val flowsearch_response = ExternalAPIs.post("TetrationOpenAPI", "POST", "/flowsearchtype", query_payloa
//println(flowsearch_response)
if (flowsearch_response.has("result")) {
    val flowsearch_result = new org.json.JSONObject(flowsearch_response.getString("result"))
    val flowsearch_df = get_sql_dataframe_from_flowsearch(flowsearch_response)
    println("Number of result rows: " + flowsearch_df.count())
    flowsearch_df.show(1)
} else {
    println("Unable to fetch data from Tetration OpenApi.  Is it configured?")
}
```

```
{"filter":{},"scopeName":"Tetration","limit":10,"T0":"2019-07-23T23:16:59.067Z","T1":"2019-07-24T00:1
6:59.067Z"}
```

```
Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 404, error message: OpenAPIClient Request Failed, OpenApi returned: Not Found
StackTrace:   at com.tetration.apps.ExternalAPIsLib.post(ExternalAPIsLib.java:71)
  at com.tetration.apps.ExternalAPIs.post(ExternalAPIs.java:41)
```

Fig. 9.3.10.2.7: mistyped query params.

Invalid Payload

Query payload key should be **scopeName** instead of **scopeNmae**.

```
val query_payload = new org.json.JSONObject("""{
    "T0": "2017-03-09T09:00:00-0700",
    "T1": "2017-03-09T19:00:00-0700",
    "scopeNmae": "Tetration",
    "limit": 10,
    "filter": {}
}""")
//Comment out the following two lines to use the above hard coded date/times rather than current time
query_payload.put("T0", hour_ago.format(DateTimeFormatter.ISO_OFFSET_DATE_TIME))
query_payload.put("T1", current_time.format(DateTimeFormatter.ISO_OFFSET_DATE_TIME))
println(query_payload)

val flowsearch_response = ExternalAPIs.post("TetrationOpenAPI", "POST", "/flowsearch", query_payload.toSt
//println(flowsearch_response)
if (flowsearch_response.has("result")) {
    val flowsearch_result = new org.json.JSONObject(flowsearch_response.getString("result"))
    val flowsearch_df = get_sql_dataframe_from_flowsearch(flowsearch_response)
    println("Number of result rows: " + flowsearch_df.count())
    flowsearch_df.show(1)
} else {
    println("Unable to fetch data from Tetration OpenApi.  Is it configured?")
}
```

```
{"filter":{},"limit":10,"T0":"2019-07-23T23:16:59.067Z","T1":"2019-07-24T00:16:59.067Z","scopeNmae":"Te
tration"}

lastException = null
```

```
Name: com.tetration.apps.exceptions.OpenAPIRequestException
Message: status code is 400, error message: OpenAPIClient Request Failed, OpenApi returned: Bad Request
StackTrace:   at com.tetration.apps.ExternalAPIsLib.post(ExternalAPIsLib.java:71)
  at com.tetration.apps.ExternalAPIs.post(ExternalAPIs.java:41)
```

Fig. 9.3.10.2.8: invalid payload.

9.3.11 Sample Notebooks

9.3.11.1 ScalaSampleNB

This sample scala notebook contains examples of the following,

- Loading and querying tetration flow data
- Loading and saving to a file
- Sending alerts

9.3.11.2 PythonSampleNB

This sample pyspark notebook contains examples of the following,

- Exploring tetration flow data
- Plotting graphs
- Reading and writing JSON data

9.3.11.3 ExternalApiSampleNB

This sample scala notebook shows an example of using an external API.

9.3.11.4 ExampleUseCase_PolicyCreation

This sample pyspark notebook shows an example of creating policies based on the scope-to-scope communication (flows) seen in the past hour and creates new ADM workspace containing these policies.

9.3.11.5 TetrationOpenApiSampleNB

This sample scala notebook shows examples of using multiple external APIs.

9.3.11.6 ExampleUseCase_ScheduledDailyRollups

This sample notebook can be used to perform daily/weekly rollups on the hourly data instances in the data lake. User can achieve this by cloning and scheduling the notebook to run as an hourly job. See *Jobs* to understand how to schedule a job. User can also provide data filters to perform rollups only on specific types of data flows. When a clone of this notebook is scheduled as an hourly job, it produces daily aggregate only once a day, but it will check every hour to see if the previous day's complete data is available and if it is not yet aggregated. Similarly it performs a weekly rollup once a week, but checks every hour to see if the previous week's complete data is available and if it is not yet aggregated. User can edit the notebook to specify the data input/output location and formats to get the aggregated data in a specific format and in a specific location in the *Data Lake*.

9.3.11.7 ExampleUseCase_ConversationAggregates

This sample notebook contains examples for fetching, aggregating and computing various stats on the aggregated data. Specifically, user can provide a specific date range (and an optional data filter) for which data aggregation can be performed. This is similar to the daily aggregation done in the "ExampleUseCase_ScheduledDailyRollups" notebook. If user has a notebook similar to the "ExampleUseCase_ScheduledDailyRollups" scheduled to run as an hourly job, then user can skip the first section in this notebook and can provide the daily aggregation output location provided in "ExampleUseCase_ScheduledDailyRollups" as the aggregated data input location in this notebook.

User can compute specific stats on the aggregated data by executing the later sections in the notebook that show computation of some stats as examples. Specifically, the notebook shows an example where the total data transfer into, out of and within a scope are computed for a chosen scope id. This example can extended to estimate the data transfer cost across cloud platforms, as done in the (deprecated) Cloud Migration App, by multiplying the computed total data transfer by the data transfer pricing. User can plot graphs corresponding to the computed stats by modifying or using the plotting examples shown in the notebook as is.

This notebook also shows an example of how to generate the output as a single file.

9.3.11.8 PolicyEffectiveness

Tetration micro-segmentation reduces risk exposure of the applications, by learning services and generating allow-list policies in ADM. This example App gathers the policies using OpenAPI, and measures policy effectiveness as the reduction in risk exposure for each application. Users of the App can customize risk measures for application, provider, consumer or service port.

9.4 Jobs

Once users finish their application, they can go to the jobs page to schedule their applications at a recurring schedule, hourly, daily, weekly, monthly or yearly. The jobs schedules can be isolated using scope configuration.

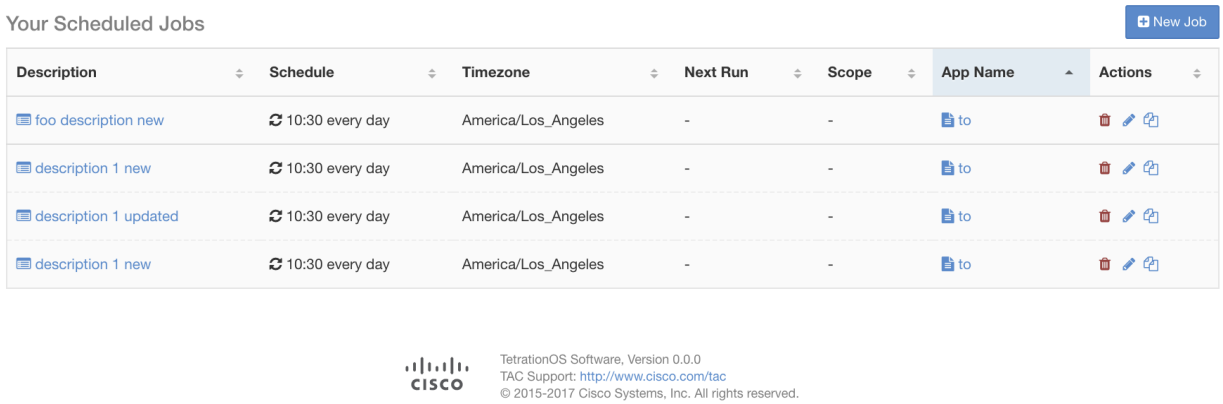


Fig. 9.4.1: User's Recurring Jobs List

9.4.1 Data Dependent Scheduling

Using `IO.exists()`, users can figure out if a previous instance of a job which produces data has completed. Thereby, users can chain job instances based on availability of a data set. This feature can work with users' generated data sets.

9.4.2 Adding New Jobs

Click on the 

button to schedule a new job.

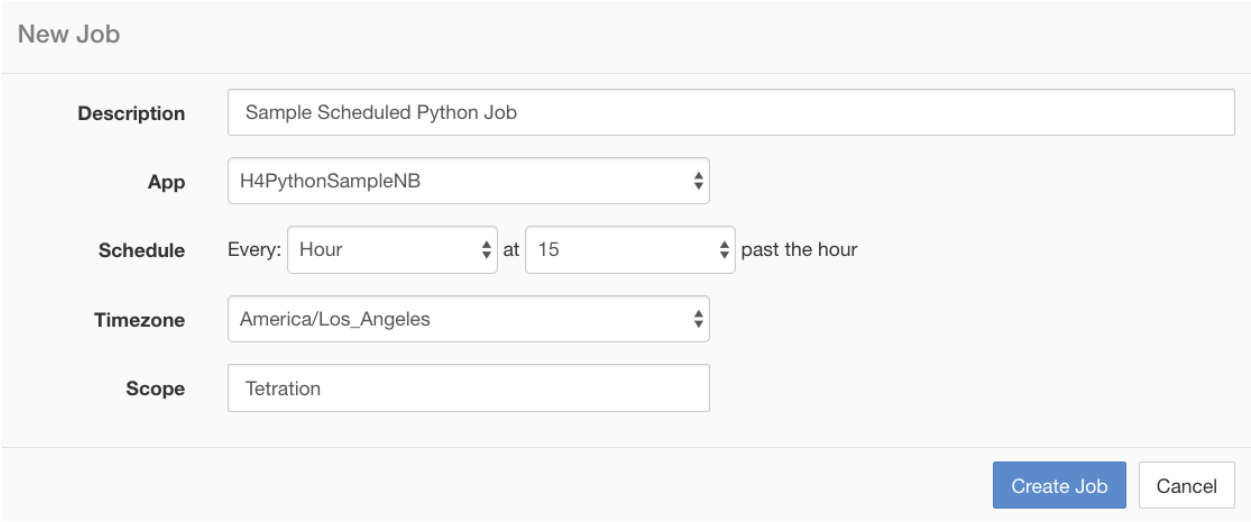
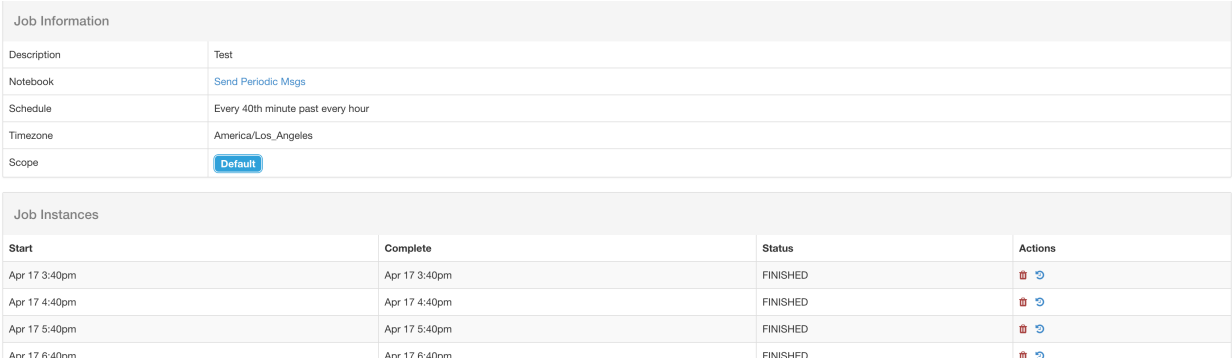


Fig. 9.4.2.1: Schedule New Job

9.4.3 Job Details

Each user can view job runs, its instances and their corresponding logs to debug any issues.



Job Information

Description	Test
Notebook	Send Periodic Msgs
Schedule	Every 40th minute past every hour
Timezone	America/Los_Angeles
Scope	Default

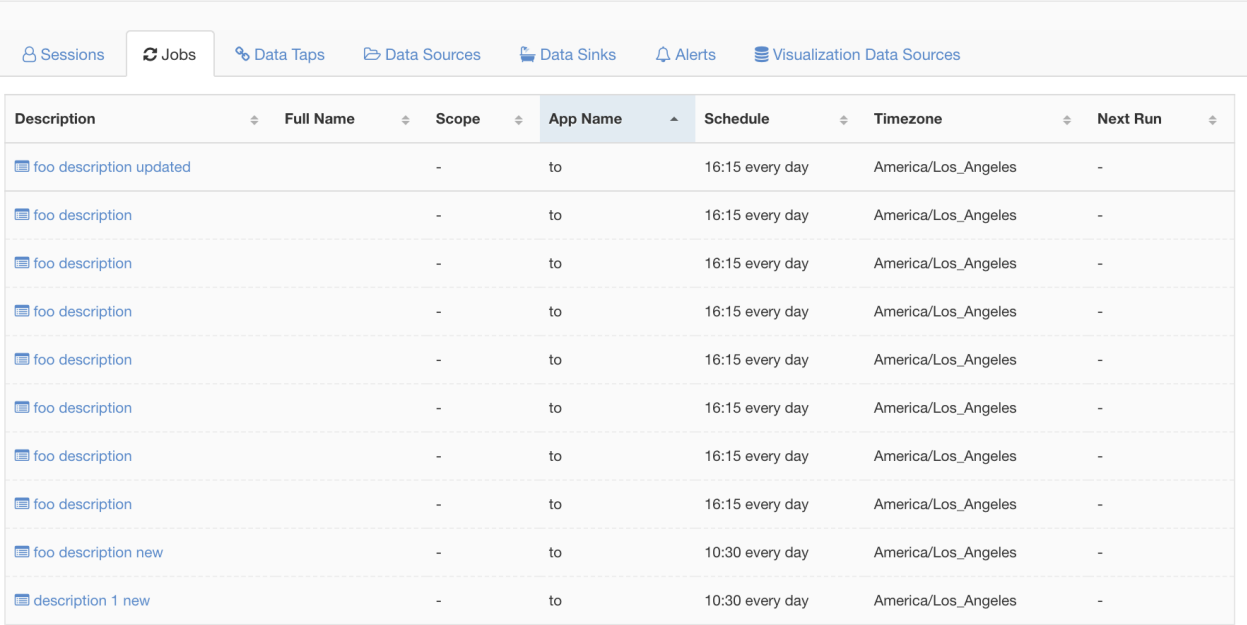
Job Instances

Start	Complete	Status	Actions
Apr 17 3:40pm	Apr 17 3:40pm	FINISHED	
Apr 17 4:40pm	Apr 17 4:40pm	FINISHED	
Apr 17 5:40pm	Apr 17 5:40pm	FINISHED	
Apr 17 6:40pm	Apr 17 6:40pm	FINISHED	

Fig. 9.4.3.1: Job Instance

9.4.3.1 Data Tap Admin Section

Data Tap Admins can navigate to **Maintenance > Data Tap Admin > Jobs** to view all scheduled jobs within their **Root Scope**, and view each jobs instances and corresponding instance logs.



Sessions | Jobs | Data Taps | Data Sources | Data Sinks | Alerts | Visualization Data Sources

Description	Full Name	Scope	App Name	Schedule	Timezone	Next Run
foo description updated		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description		-	to	16:15 every day	America/Los_Angeles	-
foo description new		-	to	10:30 every day	America/Los_Angeles	-
description 1 new		-	to	10:30 every day	America/Los_Angeles	-

Fig. 9.4.3.1.1: DP Admin Job list

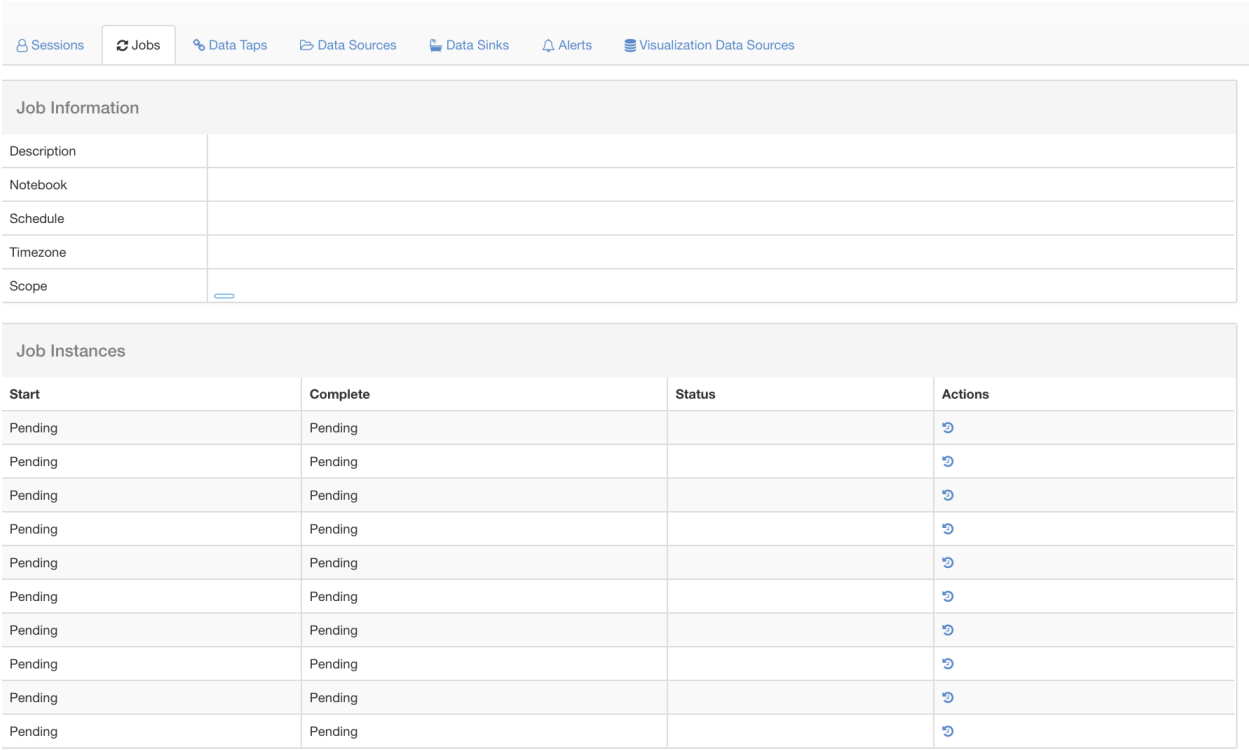


Fig. 9.4.3.1.2: DP Admin Job Instance

9.5 Data Lake

Users with Developer capability and above have access to data available in the Data Lake. (See *Roles* for information about roles and capabilities).

Users can use this page to explore the available directories, find a directory location of the desired data, or upload additional data sets into Tetration system. Users can delete the data sets that they own and upload in the **shared** directories.

Whenever a user writes a data sets, a default of 3 instances is imposed. User can override this settings through `IO.write()` and `IO.imposeRetention()`

Note: Browsing of data lake hides the actual files as the files maybe in binary formats, e.g. PARQUET, or maybe very large even though they maybe JSON or CSV. **You can read the data sets using the IO.read() APIs**

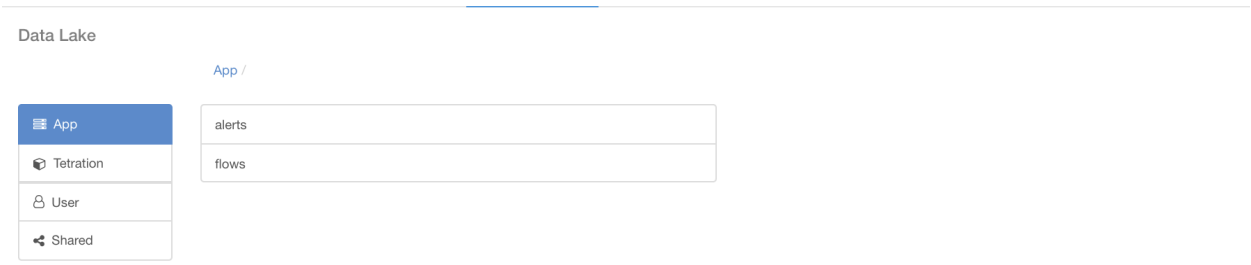


Fig. 9.5.1: Data Lake > Apps

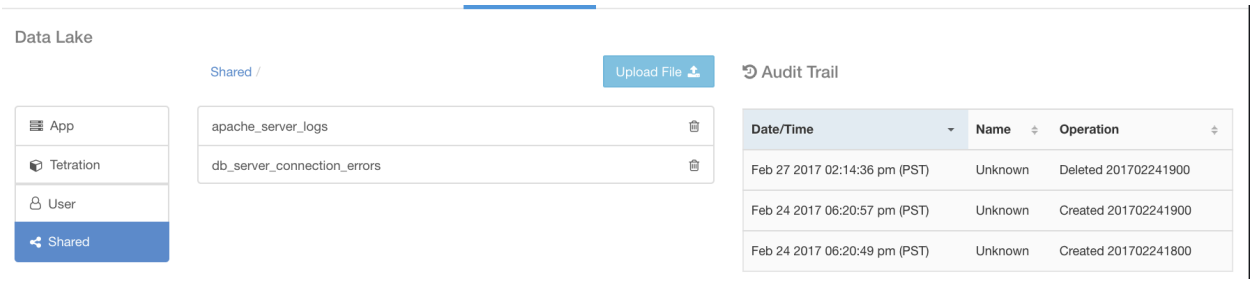


Fig. 9.5.2: Data Lake > Shared

9.5.1 Data Sources

9.5.1.1 Data Lake - Tetration Data

Tetration currently provides 2 data sources directly available via the `IO.read` library. Additional Tetration data is available via *External API*.

Some notes about columns are shown below. Full schemas can be obtained using *.printSchema()* on a dataframe from a Scala notebook.

Flows and Shallow Flows

Flows

- Scala `IO.read(sqlContext, "/tetration/flows/")`
- Python `sc._jvm.com.tetration.apps.IO.read(sqlContext._ssql_ctx, "/tetration/flows/")`

Shallow Flows

- Scala `IO.read(sqlContext, "/tetration/shallowflows/")`
- Python `sc._jvm.com.tetration.apps.IO.read(sqlContext._ssql_ctx, "/tetration/shallowflows")`

Shallow flows deprecation notice

Please note that shallow flows datasource is deprecated.

Notes about some columns or prefixes/suffixes. Although schemas are the same, shallow flows will have more empty values.

Column or prefix/suffix	Notes
timestamp	string (example format: "2018-03-28T16:00:00Z"). Minute interval this data corresponds to.
**_port*	integer
proto	string. Protocol, ex "UDP"
start_timestamp	long. Milliseconds, the start of the flow
fwd_*_count	long. Number of times this characteristic was seen. Ex. *fwd_ack_count* is number of ACKs seen.
rev_*_count	long. Number of times this characteristic was seen.
vrf_id	long
fwd_policy_permitted	string ("" or "PERMITTED"). Enforcement result. See *Enforcement*
fwd_policy_escaped	string ("" or "ESCAPED"). Enforcement result
fwd_policy_rejected	string ("" or "REJECTED"). Enforcement result
*rev_policy_**	string. Enforcement result. Similar to above *fwd_policy_**
**_scope_id*	array[string]. All applicable scopes.
**_scope_name*	array[string]
**_enforcement_epg_id*	array[string] All applicable Enforcement Groups. See *Enforcement*
**_experimental_epg_id*	array[string] All applicable Live Analysis Groups. See *Live Analysis*
**_user_tags*	array[string]
**_is_internal*	boolean

Note: See *Segmentation* for information about ADM and policy groups.

Machine

Warning: Deprecated. To be removed in 3.2. Flows data will contain equivalent data.

- Scala `IO.read(sqlContext, "/tetration/machine/")`
- Python `sc._jvm.com.tetration.apps.IO.read(sqlContext._ssql_ctx, "/tetration/machine/")`

Column	Notes
timestamp	string (example format: "2018-03-28T16:00:00Z")
host_uuid	string. Unique id that can be used across hostname changes, etc.
hostname	string. Currently empty unless hostname has changed; choose a longer time range to find last change.
os	string. Currently empty unless os has changed; choose a longer time range to find last change.
os_version	string. Currently empty unless version has changed; choose a longer time range to find last change.
rx_packet_count	long
rx_byte_count	long
rx_flow_count	long
tx_packet_count	long
tx_byte_count	long

Continued on next page

Table 9.5.1.1.2 – continued from previous page

Column	Notes
tx_flow_count	long

9.5.2 Capability

Directories	Read	Write	Upload	Delete	Browse
App	✓ (By **Root Scope**)	No	No	No	✓
Tetration	✓ (By **Scope**)	No	No	No	✓
Shared	✓ (By **Root Scope**)	No	✓	✓	✓
Users (owned)	✓	✓	No	✓	✓
Users (others)	No	No	No	No	No

Data Platform Users Capability

9.5.3 Upload Shared Data

Click on 

to upload one or more data files. These files can be accessed within by the user's application.

Note: User can upload a maximum of 10 GByte file **per upload** and the time to upload may vary depending on the user's network upload bandwidth. Uploading a 10 GByte file on a 10 Mbps uplink will take about **2 hours and 30 minute**.

9.5.4 Download User Data

Note: User can download a maximum of 10Gbyte file **per data set** and the time to download may vary depending on the user's network download bandwidth and the network settings inside the Tetration cluster. Downloading a 10GByte file on a 50Mbps downlink will take around **1 hour**. Downloading a 5GByte file on a similar downlink takes around **20 minutes**.

9.5.5 Data Tap Admin Section

Data Tap Admins can navigate to **Maintenance** > **Data Tap Admin** > **Data Lake** to view the directory structures, delete and download data sets for any users (within the same root scope). In addition to deleting **shared** data lake directories, Data Tap Admin users can also delete **app** data lake directories.

Data sets that the Data Platform admins cannot delete will be managed by Tetration data management component.

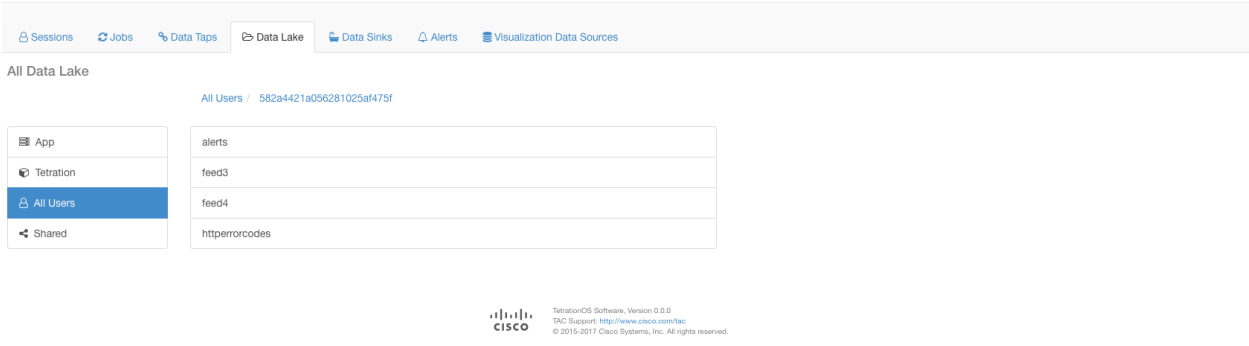


Fig. 9.5.5.1: All Users Data Lake

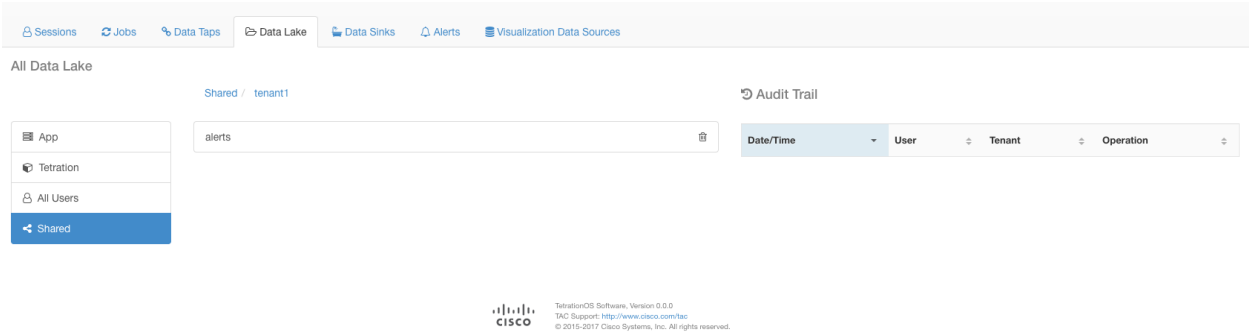


Fig. 9.5.5.2: All Shared Data Lake

9.5.6 Capability

Directories	Read	Write	Upload	Delete	Browse
App	No	No	No	✓	✓
Tetration	No	No	No	No	✓
Shared	No	No	No	✓	✓
Users (owned)	No	No	No	✓	✓
Users (others)	No	No	No	✓	✓

Data Tap Admins Capability

9.6 External API

External API provides a way to fetch additional data via api's for use inside User Apps. Current external api sources supported

- Tetration OpenApi: Fetch data from other Tetration clusters, and join within a User App to perform cross Tetration cluster analysis. OpenApi can also be used to access datasources in the same cluster not available via the Data Lake.

- Zeus: Fetches blocked domain and ip lists.

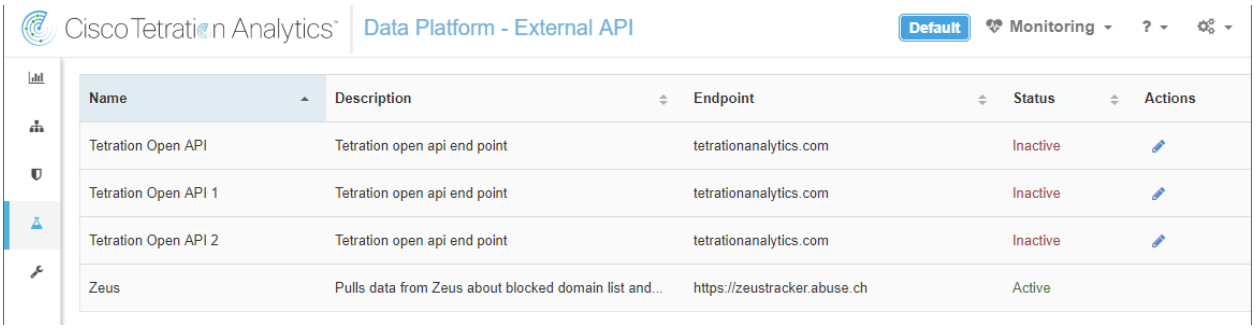


Fig. 9.6.1: Supported External API endpoints

9.6.1 Configuring Tetration OpenAPI

External API for Tetration OpenAPI can be used to access another Tetration cluster (or the same Tetration cluster) and fetch the same data available via Open API.

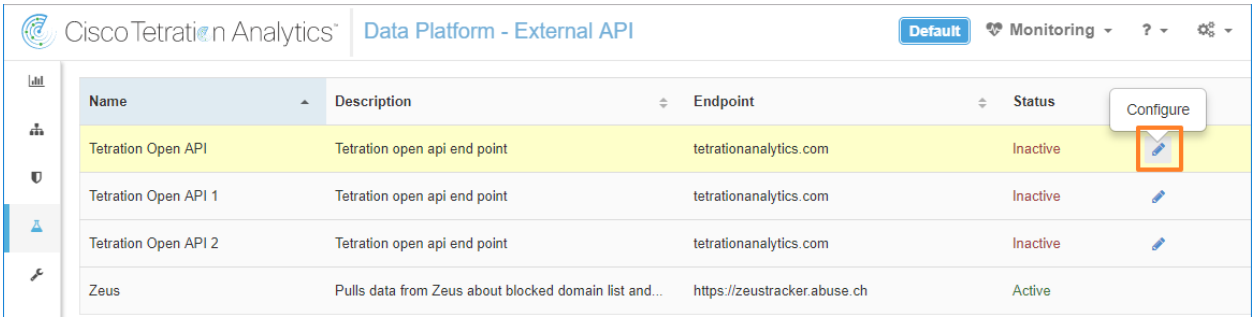


Fig. 9.6.1.1: lick the edit icon under Actions to edit an endpoint and enable.

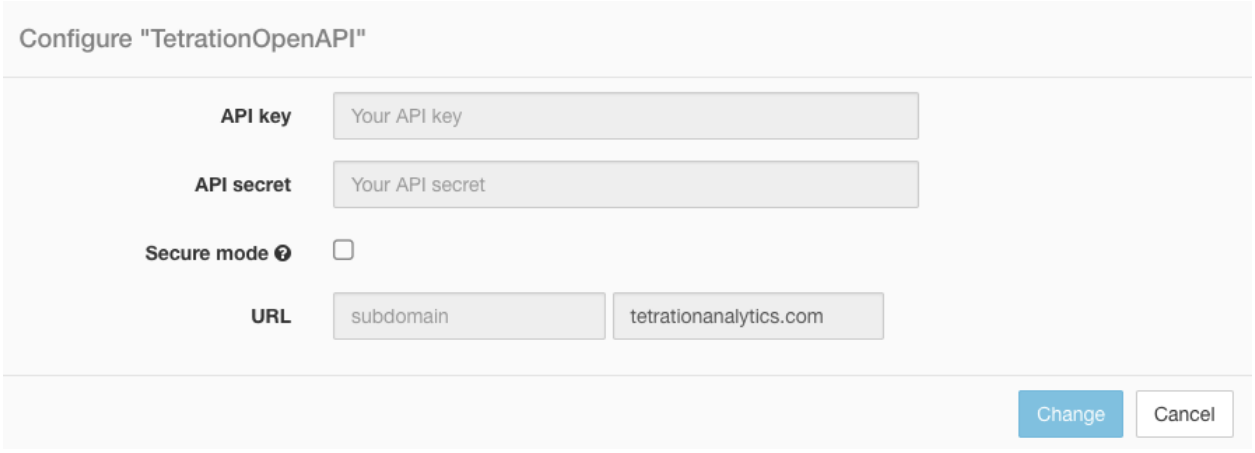


Fig. 9.6.1.2: Editing Tetration OpenAPI endpoint. Click "Change", then edit the values, "Validate", and configure. Listing will change status to "Active" when enabled.

9.6.1.1 Using a configured Tetration OpenApi endpoint

Please see the *TetrationOpenApiSampleNB* User App (*Sample applications provided by Tetration*) and use the sidebar helper tool (*Using App Sidebar*) for help with using the configured Tetration OpenAPI within a User App. Also see *OpenAPI* for information on how to construct correct payloads for querying Tetration OpenAPI.

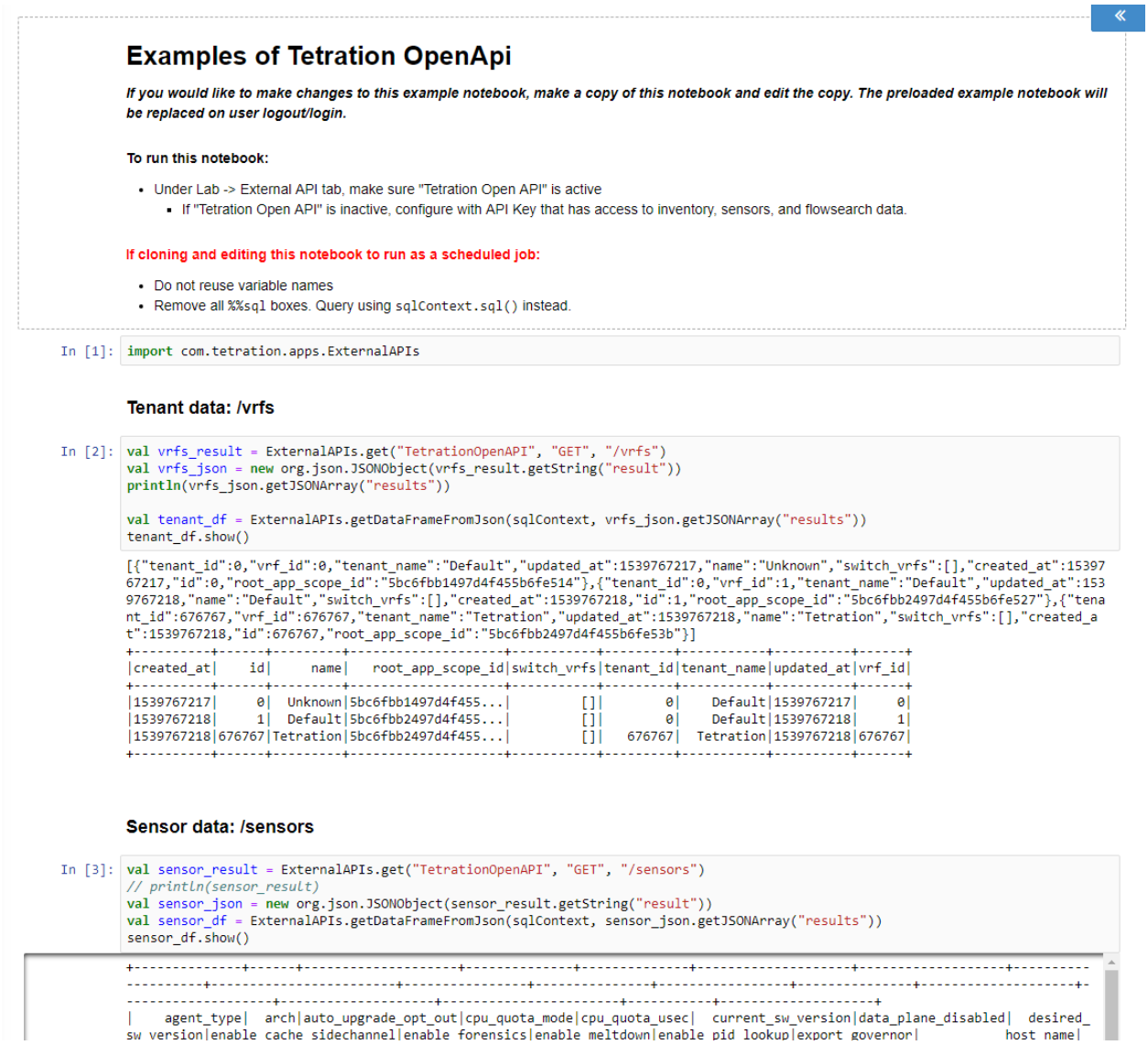


Fig. 9.6.1.1.1: Preview of *TetrationOpenApiSampleNB* User App

9.6.2 Using Other External Api Endpoints

Please see the *ExternalAPISampleNB* User App (*Sample applications provided by Tetration*) for an example of comparing Zeus blocked ips to flows within the tetration cluster.

Example of External data sources

If you would like to make changes to this example notebook, make a copy of this notebook and edit the copy. The preloaded example notebook will be replaced on user logout/login.

If cloning and editing this notebook to run as a scheduled job:

- Do not reuse variable names
- Remove all `%%sql` boxes. Query using `sqlContext.sql()` instead.

In [13]:
```
import com.tetration.apps.ExternalAPIs
```

Zeus external api

Fetching zeus IP blocklist data and comparing to recent flow data.

In [14]:
```
/* Fetch external ip blocklist data */
val blocklist_result = ExternalAPIs.get("Zeus", "IP Blocklist")
val ip_blocklist = blocklist_result.getString("result").split(",")
println("Number of ips in blocklist: " +  ip_blocklist.length)
println("Ex. First 10 ips in list: " + ip_blocklist.take(10).deep.mkString(", "))
```

```
Status of the API call is 200 with status text being OK
Number of ips in blocklist: 127
Ex. First 10 ips in list: 101.200.81.187, 103.19.89.118, 103.230.84.239, 103.26.128.84, 103.4.52.150, 103.7.59.135, 104.238.15
8.106, 108.174.157.123, 109.127.8.242, 109.229.210.250
```

In [15]:
```
/* Load flow data */
import com.tetration.apps.IO
val flow_data = IO.read(sqlContext, "/tetration/flows/", "PARQUET", "LASTHOUR")
flow_data.registerTempTable("flows")
println("Number of source ips in flow data: " + sqlContext.sql("select src_address from flows group by src_address").count())
println("Number of destination ips in flow data: " + sqlContext.sql("select dst_address from flows group by dst_address").count()
val flow_addresses = sqlContext.sql("select src_address, dst_address from flows group by src_address, dst_address")
```

```
Reading data for path /tetration/flows/ with format PARQUET with periodicity LASTHOUR
Number of source ips in flow data: 276
Number of destination ips in flow data: 62
```

In [16]:
```
flow_addresses.show()
```

```
+--------------------+--------------------+
|         src_address|         dst_address|
+--------------------+--------------------+
|       172.28.126.75|      172.31.166.108|
|       172.26.230.33|       172.28.126.76|
|fe80:0000:0000:00...|ff02:0000:0000:00...|
|       172.28.126.75|      172.31.166.109|
|       172.28.126.10|        172.28.126.1|
|      172.28.126.193|        172.28.126.1|
|      172.28.126.187|        172.28.126.1|
|      172.28.126.231|        172.28.126.1|
|      172.28.126.225|        172.28.126.1|
|      172.28.126.219|        172.28.126.1|
|      172.28.126.101|        172.28.126.1|
|      172.31.166.112|        171.68.38.65|
|      172.31.166.112|        171.68.38.66|
|      172.31.166.106|        171.68.38.66|
|      172.28.126.254|        172.28.126.1|
|       172.28.126.75|          224.0.0.22|
|      172.28.126.130|        172.28.126.1|
|      172.28.126.248|        172.28.126.1|
|      172.28.126.124|        172.28.126.1|
|       172.26.230.31|      172.31.166.102|
+--------------------+--------------------+
only showing top 20 rows
```

In [17]:
```
val src_in_blocklist = flow_addresses.col("src_address").isin(ip_blocklist:_*)
val dst_in_blocklist = flow_addresses.col("dst_address").isin(ip_blocklist:_*)
val src_blocklist_flows = flow_addresses.withColumn("src_in_blocklist", src_in_blocklist)
val blocklist_flows = src_blocklist_flows.withColumn("dst_in_blocklist", dst_in_blocklist)
blocklist_flows.registerTempTable("blocklist_flows")
blocklist_flows.show()
```

```
+--------------------+--------------------+----------------+----------------+
|         src_address|         dst_address|src_in_blocklist|dst_in_blocklist|
+--------------------+--------------------+----------------+----------------+
|       172.28.126.75|      172.31.166.108|           false|           false|
|       172.26.230.33|       172.28.126.76|           false|           false|
|fe80:0000:0000:00...|ff02:0000:0000:00...|           false|           false|
|       172.28.126.75|      172.31.166.109|           false|           false|
```

Fig. 9.6.2.1: Preview of *ExternalAPISampleNB* User App

9.7 Data Taps

Note: Tetration Currently supports writing to Kafka Brokers 0.9.x, 0.10.x, 1.0.x and 1.1.x for Datataps

To push any alerts out from Tetration cluster, user needs to use a configured data taps. Data Tap Admin users are the only ones who can configure and activate new/existing data taps. Users can only view data taps that belong to their **Tenant**.

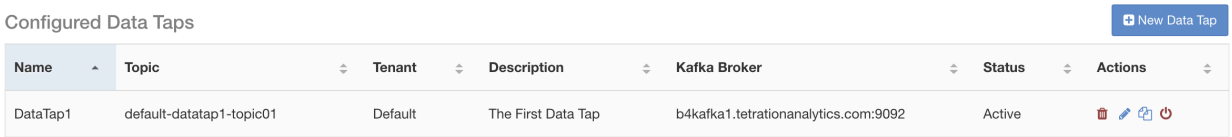


Fig. 9.7.1: Available Data Taps

9.7.1 Recommended Kafka Config

While configuring Kafka cluster, Tetration recommends to use the ports from 9092, 9093 or 9094 since, these are the ports Tetration opens for outgoing traffic for Kafka.

The following are the recommended settings for Kafka Brokers:

```
broker.id=<incremental number based on the size of the cluster>
auto.create.topics.enable=true
delete.topic.enable=true
listeners=PLAINTEXT://:9092
port=9092
default.replication.factor=2
host.name=<your_host_name>
advertised.host.name=<your_adversited_hostname>
num.network.threads=12
num.io.threads=12
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
log.dirs=<directory where logs can be written, ensure that there is sufficient space to hold the kaf
num.partitions=72
num.recovery.threads.per.data.dir=1
log.retention.hours=24
log.segment.bytes=1073741824
log.retention.check.interval.ms=300000
log.cleaner.enable=false
zookeeper.connect=<address of zookeeper ensemble>
zookeeper.connection.timeout.ms=18000
```

9.7.2 Data Tap Admin Section

Data Tap Admins can navigate to **Maintenance > Data Tap Admin > Data Taps** page to view and configure all available data taps. The data taps are configured per **Tenant**.

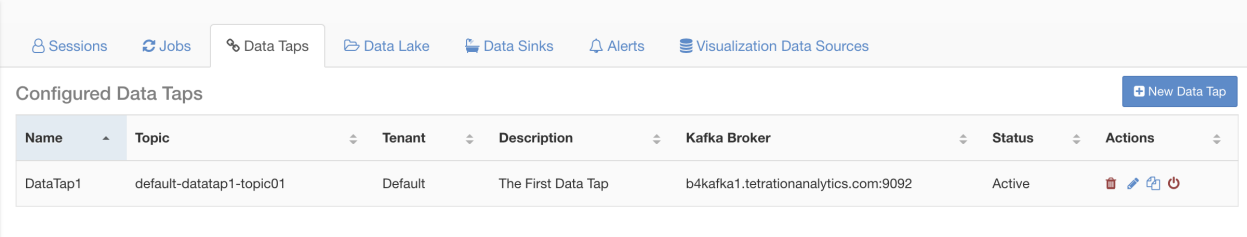


Fig. 9.7.2.1: All Available Data Taps

9.7.3 Adding New Data Tap

Data Tap Admins can click on the 

to add new data tap

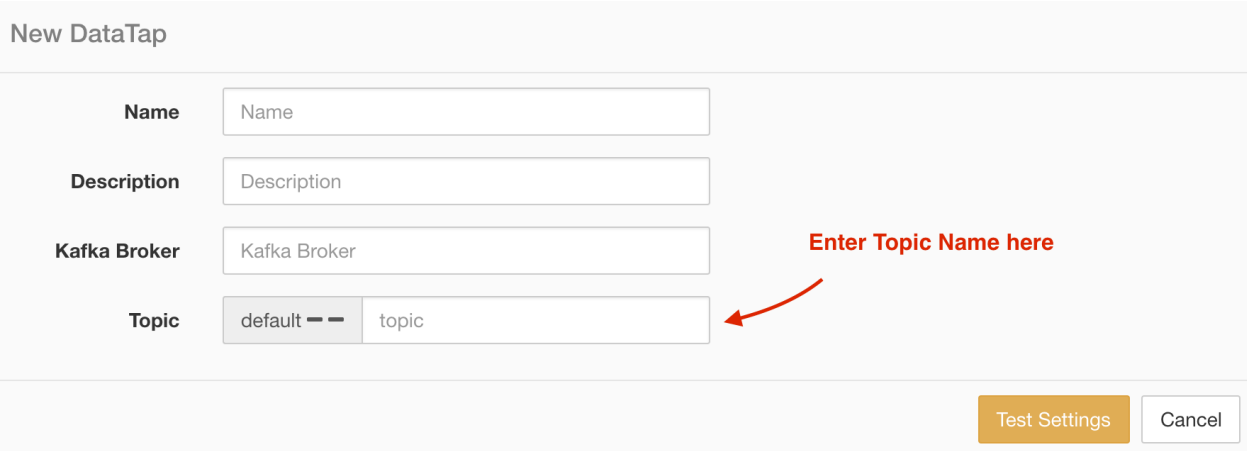


Fig. 9.7.3.1: Adding New Data Tap

Note: Changing any Data Tap values will require settings to be validated.

9.7.4 Deactivating a Data Tap

To temporarily prevent messages from leaving Tetration a Data Tap Admin can deactivate a data tap. Any messages to that data tap will not be sent. The data tap can reactivated at any time.

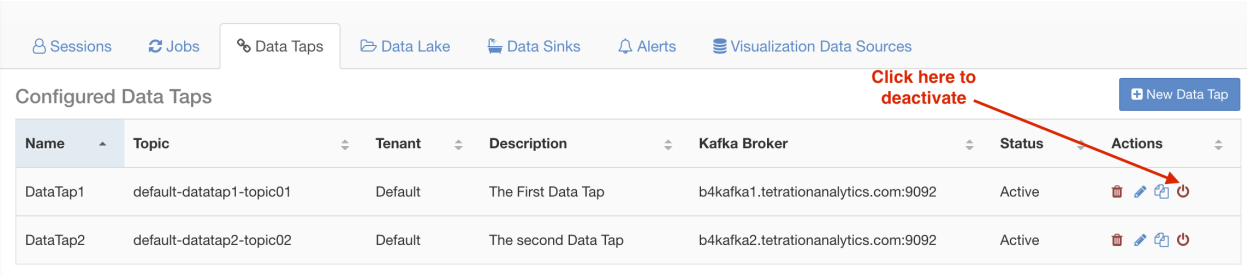


Fig. 9.7.4.1: Deactivating a Data Tap

9.7.5 Deleting a Data Tap

Deleting a datatap will delete any Tetration Apps instances that depend on that app. For example, if a user has specified that Compliance alerts should be sent to DataTap A (in the alerts tetration app), and an admin deletes DataTap A, then the Alerts app will no longer list DataTap A as an alert output.

9.8 Managed Data Taps

Managed Data Taps (MDT) are Data Taps hosted within the Tetration cluster. It is completely secure in terms of authentication, encryption and authorization. To send and receive messages from MDTs, clients needs to be authenticated, and data sent over the wire is encrypted, and only authorized users can read/write messages from/to Tetration MDT. Tetration provides Client certificates to be downloaded from the UI. Tetration uses Apache Kafka 1.1.0 as the messages broker, and, recommends clients to use secure clients compatible with the same version.

MDTs are automatically created upon the creation of root scope. Every root scope has an Alerts MDT created. To pull any alerts out from the Tetration cluster, user needs to use the Alerts MDT. Data Tap Admin users are the only ones who can download the certificates. Users can only view MDT that belong to their **root scope**.

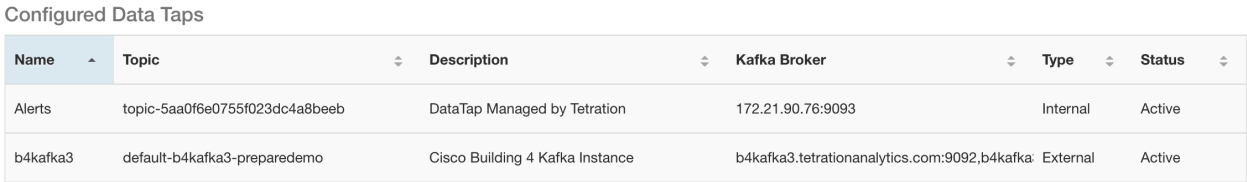


Fig. 9.8.1: List of configured Data Taps

All Tetration App alerts are sent to MDT by default, but can be changed to other Data Taps. There are two choices for downloading the certs:

1. JKS (Jave Keystore format). JKS format works well with Java Client
2. Certs. Regular certs are easier to use with Go Clients.

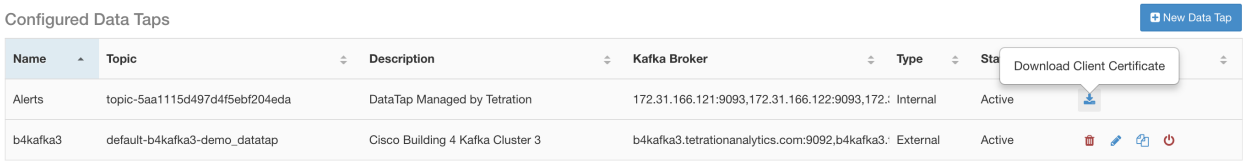


Fig. 9.8.2: Download

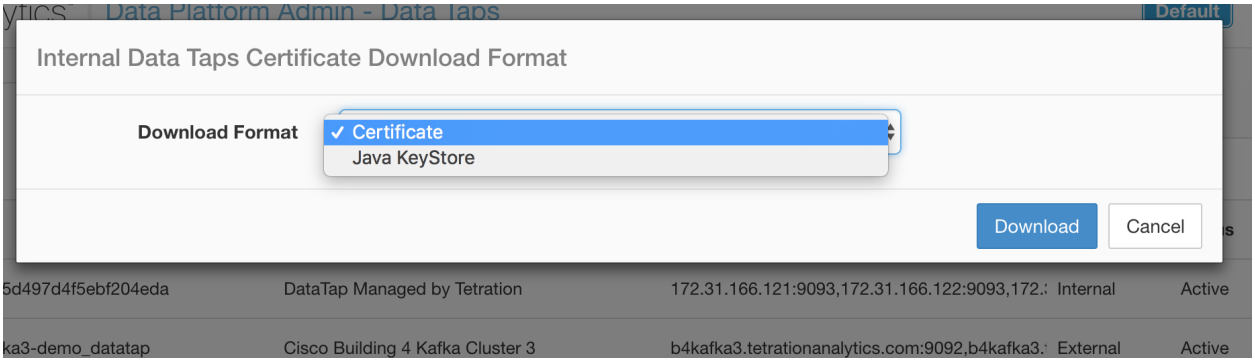


Fig. 9.8.3: Cert types

9.8.1 Java Keystore

Upon downloading the Alerts.jks.tar.gz, user you should see the following files that contain information to connect to Tetration MDT to receive messages:

1. kafkaBrokerIps.txt - This file contains the IP address string, that kafka client should use to connect to Tetration MDT.
2. topic.txt - This file contains the topic this client can read the messages from. Topics are of the format topic-<root_scope_id>. This root_scope_id can be used later while setting up other properties in Java Client
3. keystore.jks - Keystore the Kafka Client should use in the connection settings shown below.
4. truststore.jks - Truststore the Kafka Client shoud use in the connection settings shown below.
5. passphrase.txt - This file contains the password to be used for #3 and #4.

Following the Kafka settings should be used while setting up Consumer.properties (Java client) that uses the keystore and truststore:

```
security.protocol=SSL
ssl.truststore.location=<location_of_truststore_downloaded>
ssl.truststore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.keystore.location=<location_of_truststore_downloaded>
ssl.keystore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.key.password=<passphrase_mentioned_in_passphrase.txt>
```

Following set of Properties should be used while setting up the Kafka Consumer in Java code:

```
Properties props = new Properties();
props.put("bootstrap.servers", brokerList);
```

(continues on next page)

(continued from previous page)

```
props.put("group.id", ConsumerGroup-<root_scope_id>);  // root_scope_id is same as␣
↪mentioned above
props.put("key.deserializer",   "org.apache.kafka.common.serialization.
↪StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.
↪StringDeserializer");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("security.protocol", "SSL");
props.put("ssl.truststore.location", "<filepath_to_truststore.jks>");
props.put("ssl.truststore.password", passphrase);
props.put("ssl.keystore.location", <filepath_to_keystore.jks>);
props.put("ssl.keystore.password", passphrase);
props.put("ssl.key.password", passphrase);
props.put("zookeeper.session.timeout.ms", "500");
props.put("zookeeper.sync.time.ms", "250");
props.put("auto.offset.reset", "earliest");
```

9.8.2 Certificate

If end user wants to use Certificates, they can use Go clients using Sarama Kafka library to connect to Tetration MDT. Upon downloading Alerts.cert.tar.gz, user should see the following files:

1. kafkaBrokerIps.txt - This file contains the IP address string that Kafka Client should use to connect to Tetration MDT
2. topic - This file contains the topic this client can read the messages from. Topics are of the format topic-<root_scope_id>. This root_scope_id can be used later while setting up other properties in Java Client.
3. KafkaConsumerCA.cert - This file contain the KafkaConsumer certificate.
4. KafkaConsumerPrivateKey.key - This file contains the Private Key for the Kafka Consumer.
5. KafkaCA.cert - This file should be used in the root CA certs listing in the Go client.

See the following example of Go Client to connect to Tetration MDT. (Attach the Sample Go Code)

Sample Go Client to consume alerts from MDT

9.9 Data Sink

Data Sinks allow Tetration to ingest data from outside. For this we use Kafka Brokers (Apache Kafka version 1.1.0). A Data Tap Admin can create a Datasink endpoint, where data can be written. Anyone with the correct certificates for the Data Sink topic can push data into the Tetration ecosystem.

All users can see a list of Data Sinks configured by their Data Tap Admin.

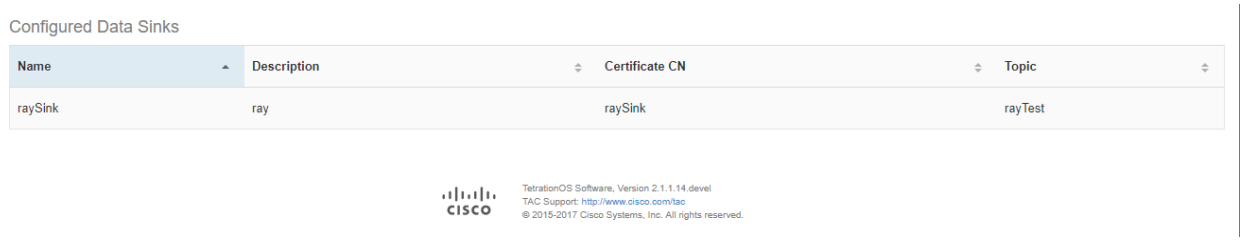


Fig. 9.9.1: Available Data Sinks

9.9.1 Data Tap Admin Section

Data Platform admins can navigate to **Maintenance > Data Tap Admin > Data Sinks** page to view and configure all available data sinks. The data sinks are configured per **Root Scope**.

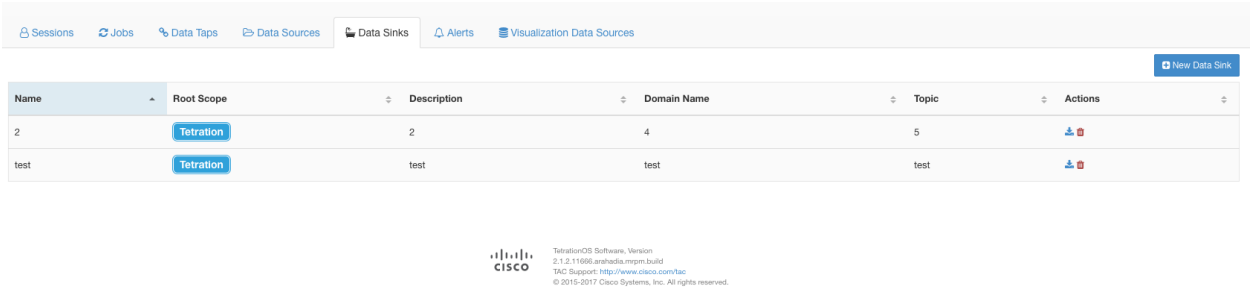


Fig. 9.9.1.1: All Available Data Sinks

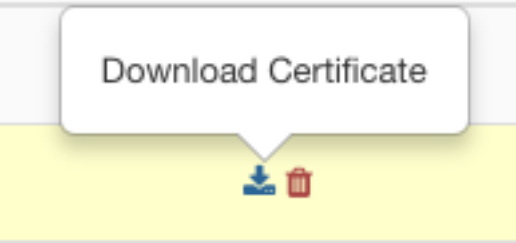


Fig. 9.9.1.2: Downloading Data Sinks Client Generated Certificate

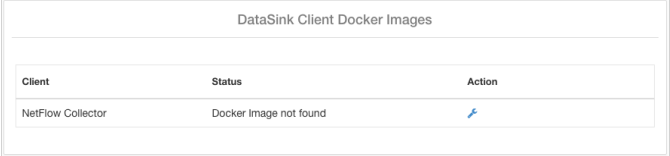


Fig. 9.9.1.3: Preparing Netflow Collector Docker Image for selected Data Sink

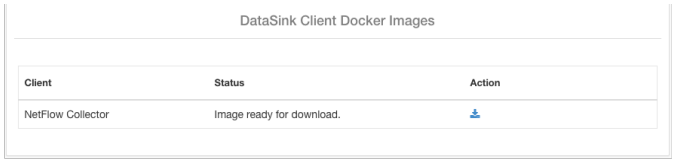


Fig. 9.9.1.4: Downloading Netflow Collector Docker Image for selected Data Sinks

9.9.2 Adding New Data Sink

To create a new data sink, click on 

. A Data Tap Admin has to provide following details to configure a datasink: 1. Name - Name of Datasink 2. Description - A sample description for the datasink 3. Root Scope - Select the scope of the datasink. There is a drop down menu. 4. Certificate CN (Common Name) - Common Name that should be used in the certificates. For example - Netflow11. It serves as an identifier for the name in the Certificate, and must be unique across tenants. 5. Topic - Name of the topic that Kafka should write to for this datasink.

Note: Topic name must be unique across all tenants. Max length is 255 alphanumerics characters including . (dot), _ (underscore), and – (hyphen)

Warning: Creating a data sink with previously deleted `topic` must be avoided for at least **48 hours** since the time of `topic` deletion

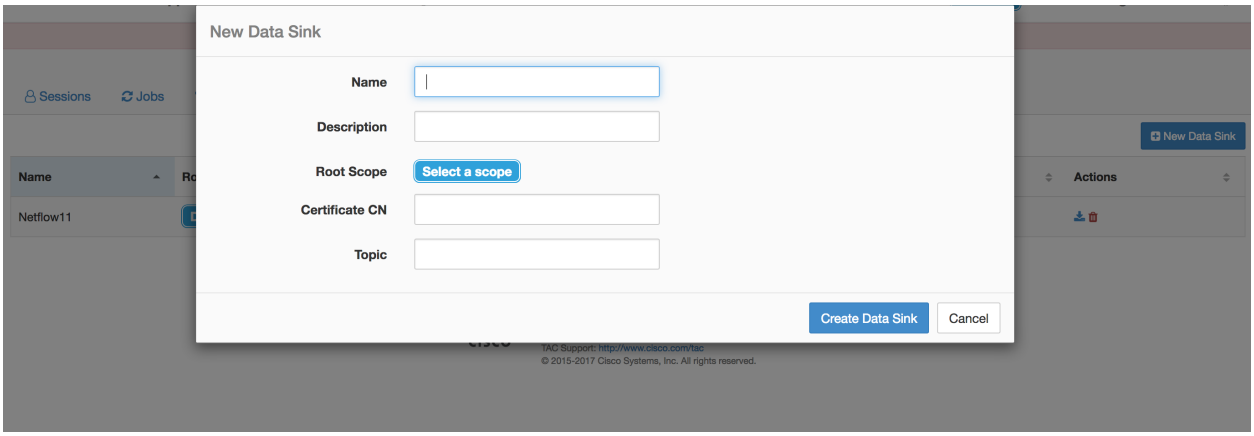


Fig. 9.9.2.1: Add new Data Sink

After a datasink is configured, it shows up as follows.

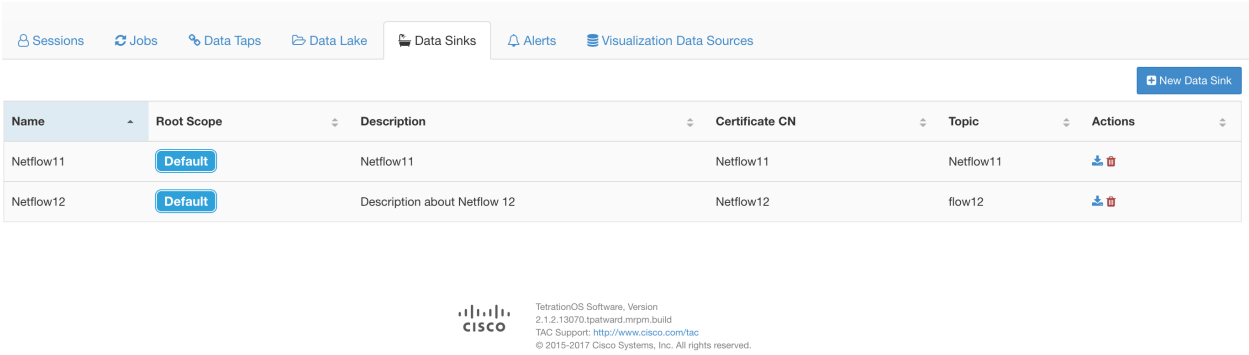


Fig. 9.9.2.2: After configuration

9.9.3 Deleting a Data Sink

A Datasink can be deleted by clicking on the trash icon. Tetration removes all traces of the datasink, and a user can no longer write to a Datasink for which it had previously downloaded the certificates and other related files. Backend services typically take from 30 seconds to 1 minute to stop the access completely.

9.9.4 How to use a Data Sink

Datasink can be current used in two ways to push data from outside into Tetration. More details follow in subsequent section. 1. Using a Kafka Client 2. Using a Producer App that can be downloaded from Data Tap Admin Section.

9.9.4.1 Using a Kafka Client

Note: NetFlow Collector is an **Alpha** release in this build.

1. User needs to download a tar.gz file associated with the datasink. This tar.gz file contains 4 things that are needed to connect to Tetration Kafka cluster.

   1. Keystore.jks - A Java keystore that contains the key pair for the client to use.
   2. Truststore.jks - A Java keystore that contains all the certificates that client will trust.
   3. Passphrase.txt - This file contains the passphrase that will be used to decrypt the keystore.jks and truststore.jks
   4. kafkaBroker_ip_address.txt: This file contains the IP addresses of the Kafka Brokers which clients will connect to.

   2. Untar the file into a directory from where a user needs to publish data to a Datasink.
   3. Apache Kafka comes with a standard Kafka Client for Producer and Consumer which can be used with the following config. Following is the config for Kafka Client (Producer) to send data to Tetration. The untarred file will contain the config that Kafka Producer needs to be setup correctly.
   4. Use the following command to start a Kafka Client (Producer). This is the console client through which you can send messages to the topic created in datasink.

```
$> bin/kafka-console-producer --producer.config <location_of_producer.properties> --
↪topic <topic_created_in_datasink> --broker-list <list_of_brokers_mentioned_in_
↪KafkaBroker_file>
```

Producer config (producer.properties file contents)

```
security.protocol=SSL
ssl.truststore.location=<location_of_truststore_downloaded>
ssl.truststore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.keystore.location=<location_of_truststore_downloaded>
ssl.keystore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.key.password=<passphrase_mentioned_in_passphrase.txt>
```

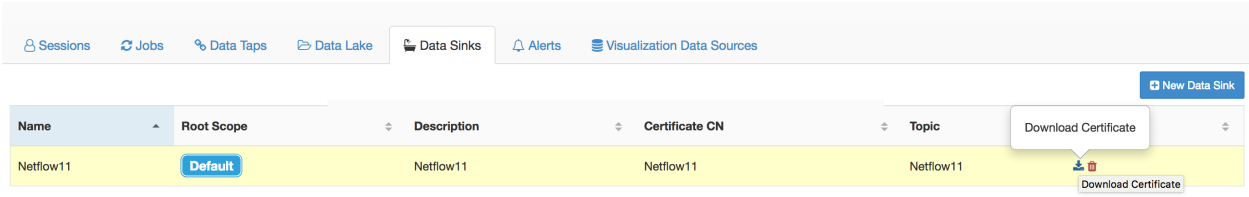


Fig. 9.9.4.1.1: Data Sink Download Certificate

9.9.4.2 Producer Apps

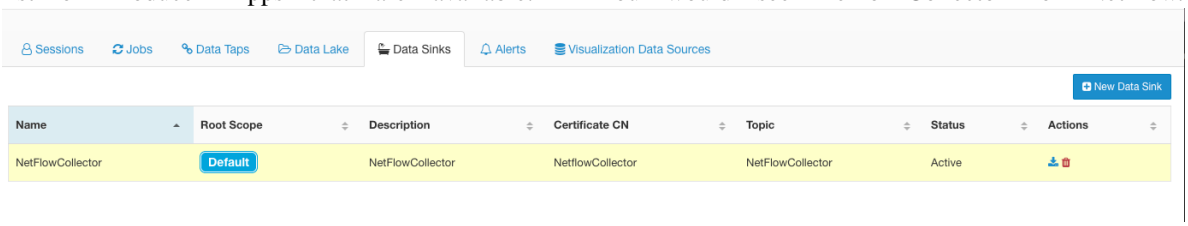
Tetration provides downloadable apps that feed specific data to Tetration via Data Sinks.

Demo Collector (NetFlow)

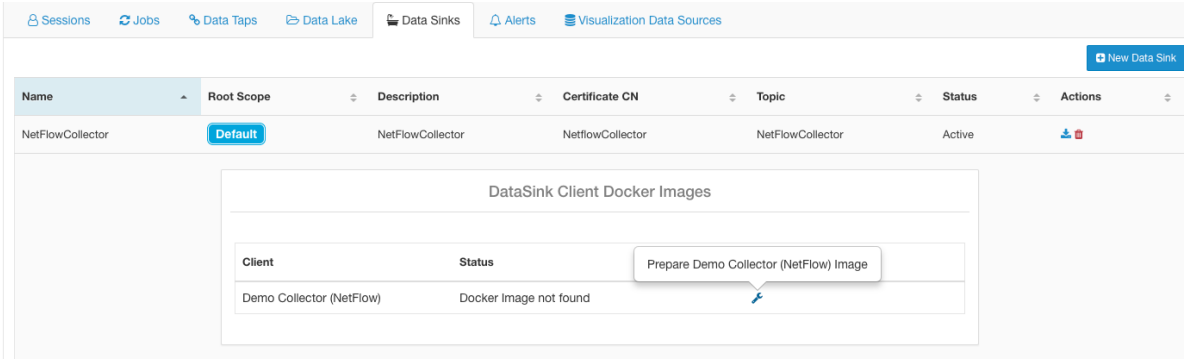
Netflow collector can be used to run as a docker application. Netflow collector listens on standard NetFlow and IPFix ports to collect Netflow/IPFix records and sends them to the associated datasink on the Tetration cluster.

To Download the NetFlow Collector for a particular Data Sink

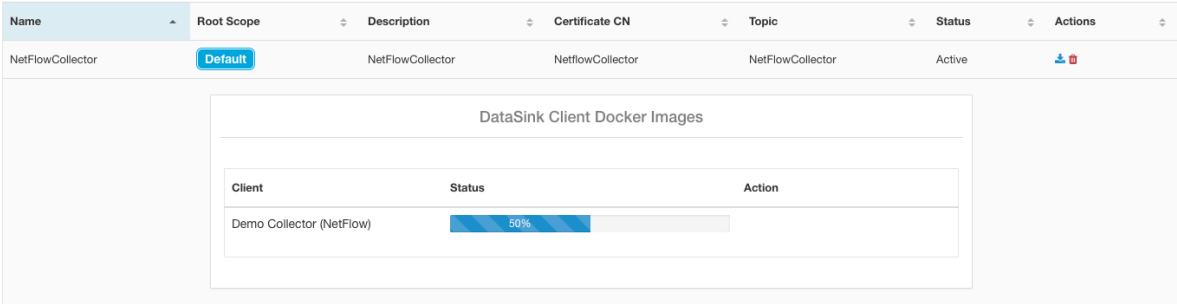
1. Click on a datasink to see a list of Producer Apps that are available. You would see Demo Collector for NetFlow.



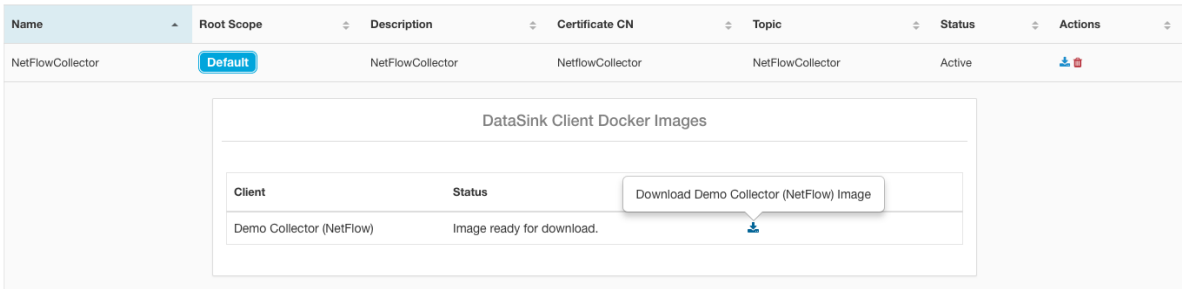
2. Click on the Action button to start preparing docker image.



3. The preparation can take up to 2 minutes. While the image is being prepared, a status bar will be displayed.



4. Once the image is ready you can download the image. A prepared client will remain available for 7 days. If a client is no longer available, a new client can be prepared and downloaded.



5. Untar the tar.gz file and follow the instructions in README file to run the client. You need the NetFlow(4729) and IPFix(4739/6343) ports available on the machine where you run this collector.

```
linux-host:~$ls -ltr NetFlowCollector_tet_netflow_docker.tar.gz
-rw-r--r-- 1 tetter tetter 82375762 Sep  7 14:57 NetFlowCollector_tet_netflow_docker.tar.gz
linux-host:~$tar xvzf NetFlowCollector_tet_netflow_docker.tar.gz
netflow_docker/README
netflow_docker/run.sh
netflow_docker/NetFlowCollector_tet_netflow_docker.tar.gz
linux-host:~$cd netflow_docker/
linux-host:~/netflow_docker$cat README
The following application runs as a netflow collector inside a docker container.
This container will talk to the TetrationCluster, and securely send the netflow
data it collects to associated DataSink

Pre-requisits to tun this application is to have Docker installed

To run the application in bash shell, please run the run.sh script using the following command
    sudo sh run.sh
linux-host:~/netflow_docker$sudo sh run.sh
```

6. Once your application is running, any NetFlow/IPFix records received by the client would be pushed to the datasink on the tetration cluster.

9.10 Visualization Data Sources

We can create custom data sources for visualization on dashboards which are called VDS or Visualization Data Sources. VDS can be created and edited by either of Tetration Apps or Data Tap Admin (DP Admin).

VDS has the following variants, described in the table below.

VDS Type	Read Access	Write Access	Create/Edit Access
Tetration	All Users	Only Tetration Internal Comp./Apps	Only Tetration Internal Comp./Apps
Shared (at Root Scope)	Users under shared root scope	Users under shared root scope via User Apps	Data Tap Admin
Private	Users who created the VDS	User who created the VDS	Data Tap Admin

Note: Edit access includes changing behavior of columns in a data source as to whether the column can be group-by enabled or likes. Read section below for more details.

9.10.1 Creating a Visualization Data Source (VDS)

DP Admin can navigate to Maintenance > Data Tap Admin > Visualization Data Source to create a VDS as shown below.

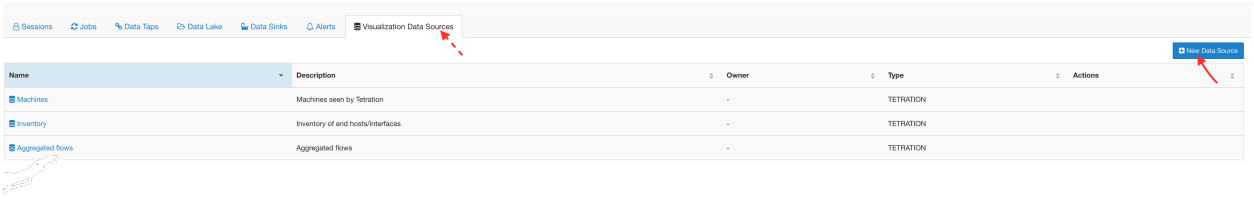


Fig. 9.10.1.1: Navigate to Data Source (VDS)

VDS can also be created by Tetration Apps and backend components which can be available for users to consume on dashboard.

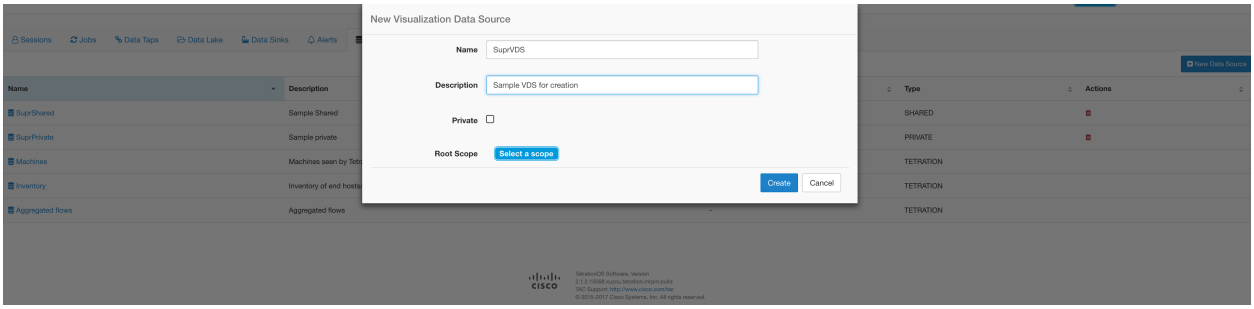


Fig. 9.10.1.2: Creation modal

There are 2 types of VDS that can be created:

1. Shared VDS - shared at root scope, accessible by anyone whose scope matches the root scope or its children
2. Private VDS - shared for a user, is not accessible by anyone else. The intent of Private VDS is to allow Data Platform users to first experiment with

Note: Tetration will override `scope_id` field inside VDS for both Shared and Private VDS in order to provide the right access permissions on VDS.

9.10.2 Editing a Visualization Data Source

Column permissions for a Visualization data source can be edited by Root Scope Owners.

VDS can be edited on the following aspects:

- Whether the column is **enabled** for users. This is useful if there are experimental features in shared VDS which the admin can prevent access from dashboard users.
- Whether a column can be enabled for **Groupby** queries. Groupby queries are compute and memory intensive if the cardinality is high for the given dimension. Therefore, VDS owner can decide to disable groupby on dimensions which may cause high latency queries during charting. This decision has to be done alongside the dashboard creator/user to determine slowness.
- Whether the column is available for **filter** use within the queries.

All of the above edits are available only for dimension columns, which holds string values instead unlike the metric columns which holds numerical values.

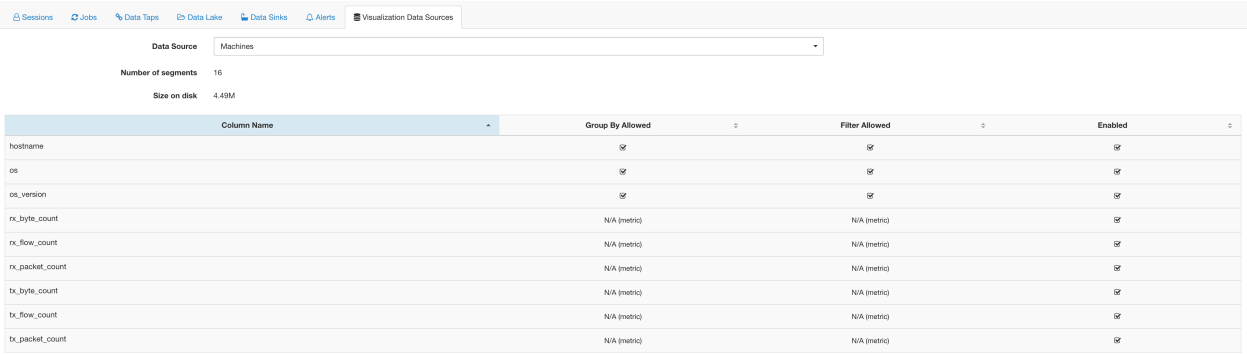


Fig. 9.10.2.1: VDS Columns

Note: Any changes to the metadata of a VDS or creation of a new VDS will reflect on UI with a lag of atmost 10 min. Alternatively, the resque job **H4Resque::RefreshVisualizationDatasources** can be run to update immediately. (only Site Admin or Customer Support users can do this)

9.10.3 Creating a Dashboard

Refer to *Dashboard* for dashboard creation

9.11 Writing data to a VDS

Users can write to shared and private VDS using User Apps. The write call is exposed in the side toolbar helper. Writing VDS is supported in both python and scala.

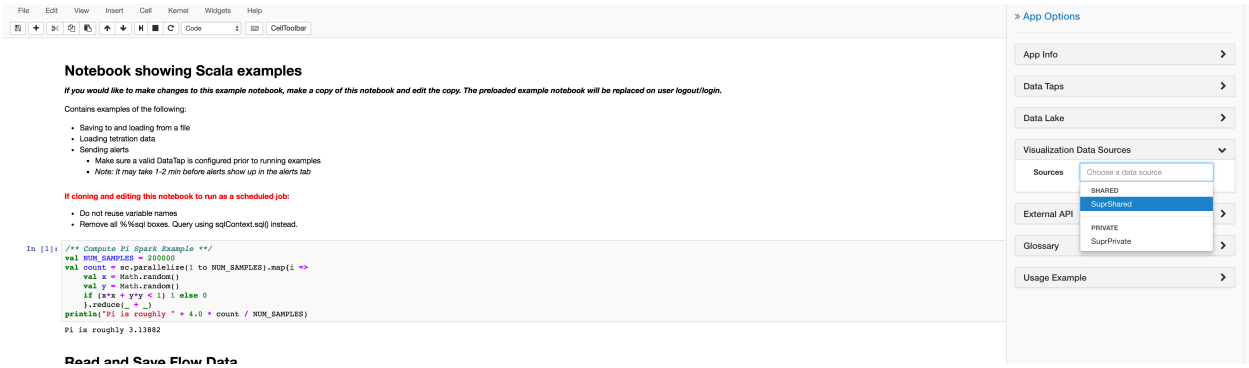
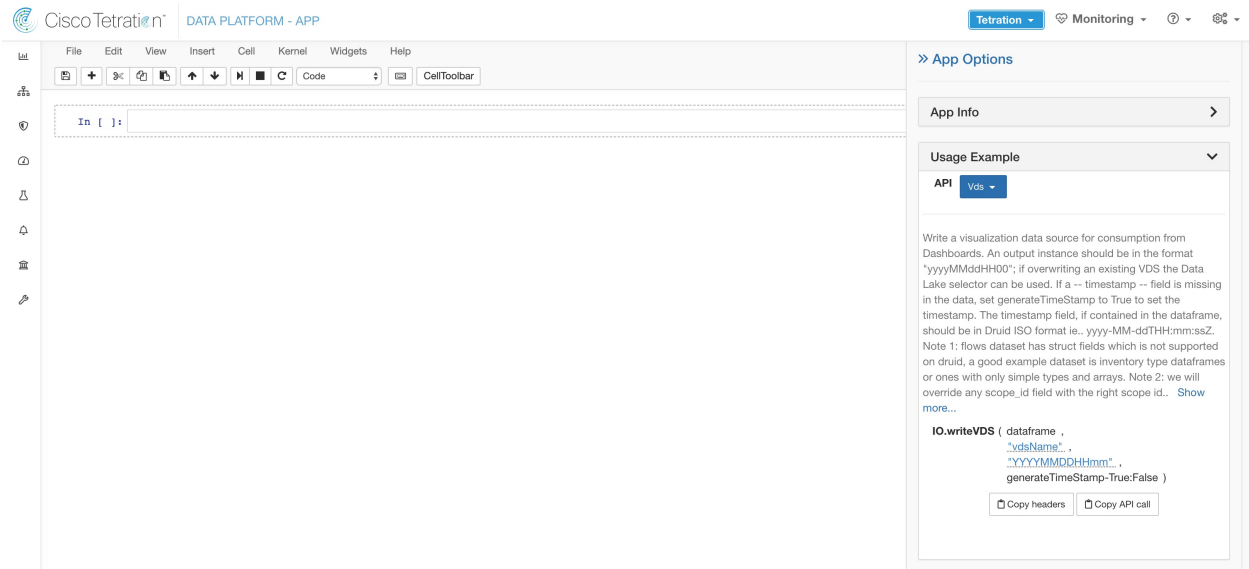


Fig. 9.11.1: Sidebar notebook

The shared or private VDS can be picked in VDS listing and the subsequent API call or headers can be copied.



Sidebar help

9.12 Settings

This page allows you to perform factory reset on user data sources and sample user apps

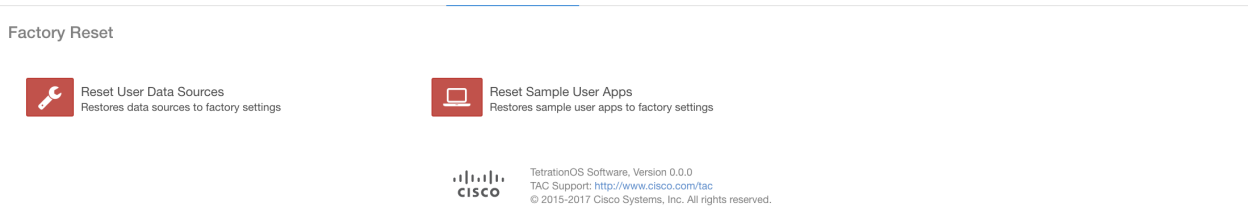


Fig. 9.12.1: Factory Reset buttons

Reset User Data Sources will clear all the data that a user generates under Data Lake - User tab.

Example notebooks are not automatically reset on user logout/login. Settings > "Reset Sample User Apps" is the only way to reset the notebooks now.

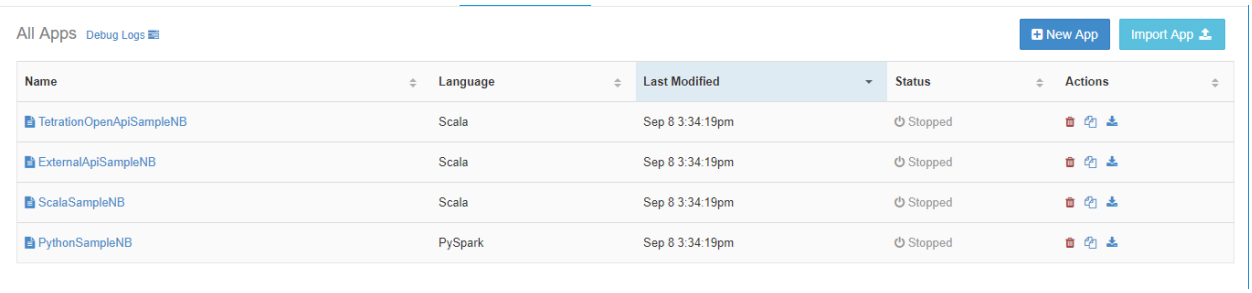


Fig. 9.12.2: Default User Apps

9.13 Users Sessions

A user session is associated with active applications. Data Tap Admins can view all active sessions by navigating to **Maintenance > Data Tap Admin > Sessions**. A session can be terminated by a Data Tap Admin which in turn terminates all the associated active applications.

Note: A total of five **concurrent** user sessions and 15 running applications **across sessions** are allowed.

Sessions | Jobs | Data Taps | Data Sources | Data Sinks | Alerts | Visualization Data Sources

Terminate	Email	First Name	Last Name	Last Data Platform Activity
	tester@testdomain.com	Test	User	a year ago
	tester-2fa@testdomain.com	Test2FA	User2FA	a year ago
	siteadmin@testdomain.com	Test	Site Admin	a year ago
	labuser@testdomain.com	Test	Lab User	a year ago
	scopeowner@testdomain.com	Test	Scope Owner	a year ago

Fig. 9.13.1: All Users Application Sessions

9.14 User Access Matrix

Following is the user access matrix for Data Platform features. Users with **Developer** or higher capability have access to all features. **Developer or higher** capabilities includes **Developer**, **Enforce** and **Owner**.

Table 9.14.1: Matrix for Data Lake data

	Tetration Apps	User Apps	Alerts	Jobs
All Users	Add/Modify	•	View/Configure	•
Developer (or higher)	Add/Modify	View/Edit	View/Configure	View/Edit

	Data Sink	External API	Data Taps	Settings
All Users	View	•	View	•
Developer (or higher)	View	View/Edit	View	Reset options

	App data	Tetration	User	Shared
All Users	•	View	•	•
Developer (or higher)	View	View	View/Delete/Download	View/Upload/Delete

CHAPTER

TEN

ALERTS

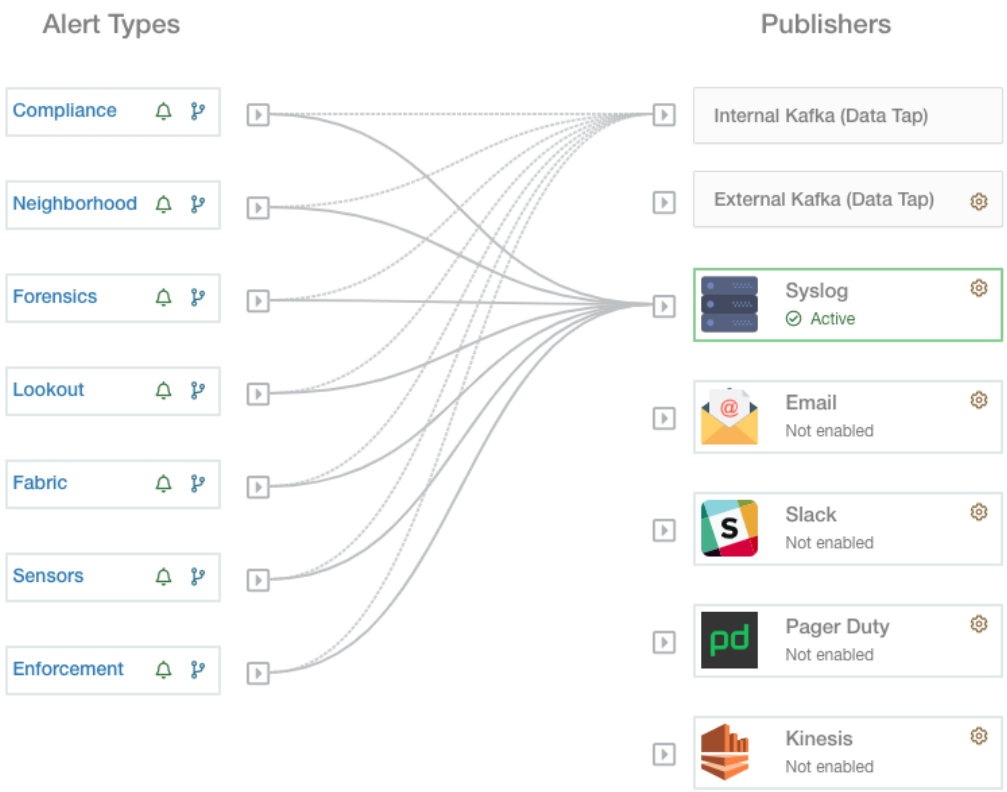


Fig. 10.1: Alerts Configuration allows you to configure alerts and select publishers to send alerts.

Alerts within Tetration consist of many integrated components. These can roughly be divided as:

Visibility:

- Alerts Page: Located at Alerts > Current Alerts. This page consist of a preview of alerts that were sent to a Data Tap

Alert Sources and Configuration:

- Tetration components and App Store apps: These may be referred to as *alert generators* or *alert datasources*. Alert generators determine whether an alert should be **created**. For example, Lookout Annotation

and Neighborhood are both Tetration App Store apps which are alert generators. Some alert generators are not listed in the app store, such as Enforcement and Compliance.

- Alert Configuration: Determined by the app/component, but many use a common interface (referred to as *Alert Configuration Modal*) that has features such as configuration of the Data Tap and summary alert options
- Alerts Configuration Page: Located at Alerts > Configuration. This page provides both alert configurations configured using the common modal, and alert publisher and notifier settings.

Sending Alerts:

- Alerts App: An implicit Tetration App that sends generated alerts to a configured Data Tap. The Alerts App handles features such as snoozing and mute, in essence determining which alerts should be **sent**
- Alerts Publisher: Limits how many alerts are visible in the UI, and pushes alerts to Kafka (MDT or DataTap) for external consumption.
- Edge Appliance: Pushes alerts to other systems such as Slack, PagerDuty, Email, etc.

10.1 Configuring Alerts

Alerts Configuration allows you to configure alerts trigger rules and select publishers to send alerts. Alert types shown in this page vary from different user roles. Alert publishers can be either Kafka (Data Tap) or Notifiers.

Note: Tetration 3.0 removed Alerts App and Compliance App from the Tetration App Store. You will be able to configure alerts including the compliance alerts in this page without creating an Alert App instance or Compliance App instance.

10.1.1 Create Alerts

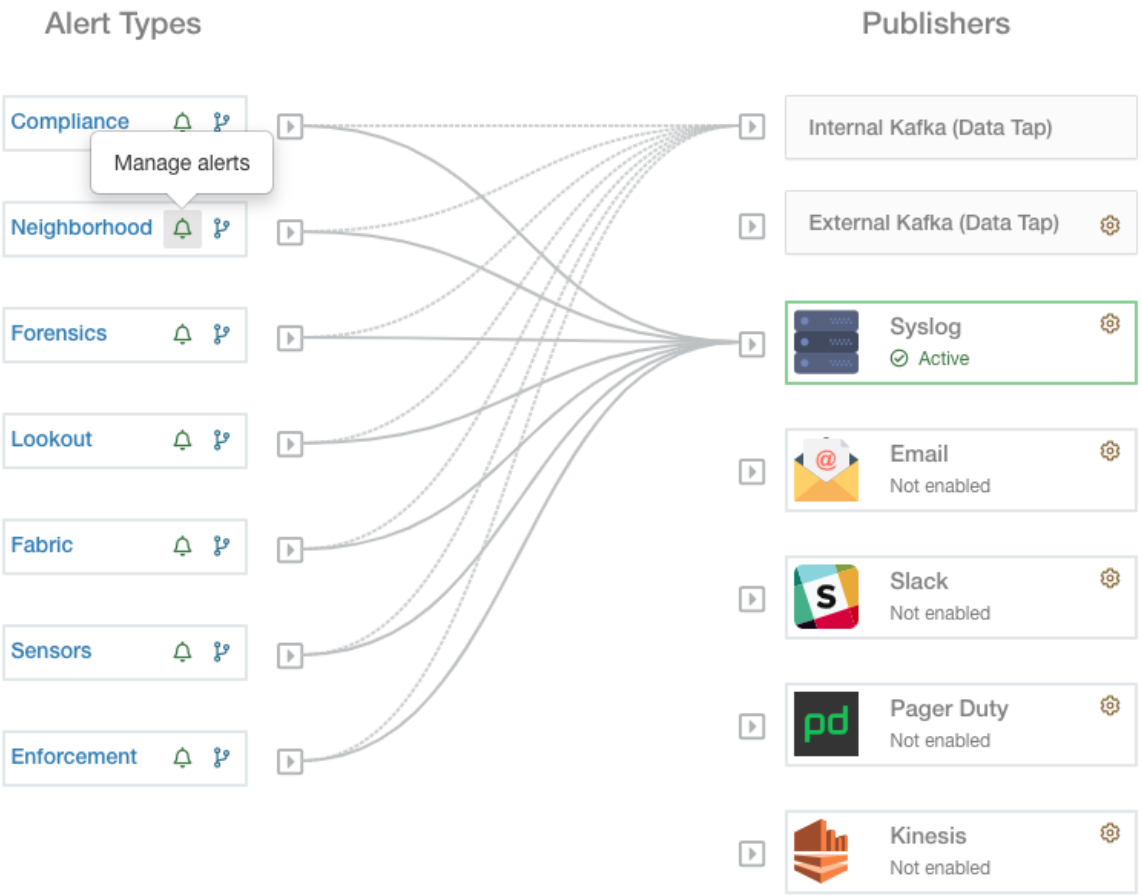


Fig. 10.1.1.1: Click the green bell icon to start creating an alert (trigger rule).

Several components use a common *Alert Configuration Modal* for configuring alerts. At the moment this includes the following (please see user guide for each for more details about configuring their specific alerts):

- *Compliance*
- *Neighborhood*
- *Network Fabric*
- *Lookout Annotation*
- *Enforcement*
- *Sensors*

Note: For Compliance alert type, only users with at least Enforced capability on the currently selected scope will be able to create an alert trigger rule.

Note: For Enforcement and Sensors alert type, alert trigger rule will be enforced on the currently selected root scope.

The following types do not have a configuration modal.

- *Forensics*: Configure using forensic rules
- *Connectors*
- Federation
- ./admiral

10.1.2 Alert Configuration Modal

The *Alert configuration modal* consists of 6 sections:

1. The type of alert. *Note:* This is only shown when the configuration of the alert varies by *subject* (Currently only shown for Neighborhood alerts)
2. The *subject* of the alert: ie. *"what we are going to alert over"* This is dependent on the app, and may be pre-populated when the alert modal is contextual
3. The condition on which an alert will be triggered: ie. *"when will we generate an alert"*. A list of available conditions can be found by hovering over the ? *Note:* this list will show those conditions available specifically for the type of alert alert currently being configured
4. Alert severity selection. If there are many alerts generated, alerts with higher severity will be visible in the UI preferentially over alerts with lower severity.
5. Additional configuration options consisting of Summary Alert options. Click "Show Advanced Settings" to expand.
6. Close Modal: "Create" if adding a new alert and all configuration options specified. Or "Dismiss" if not adding a new alert

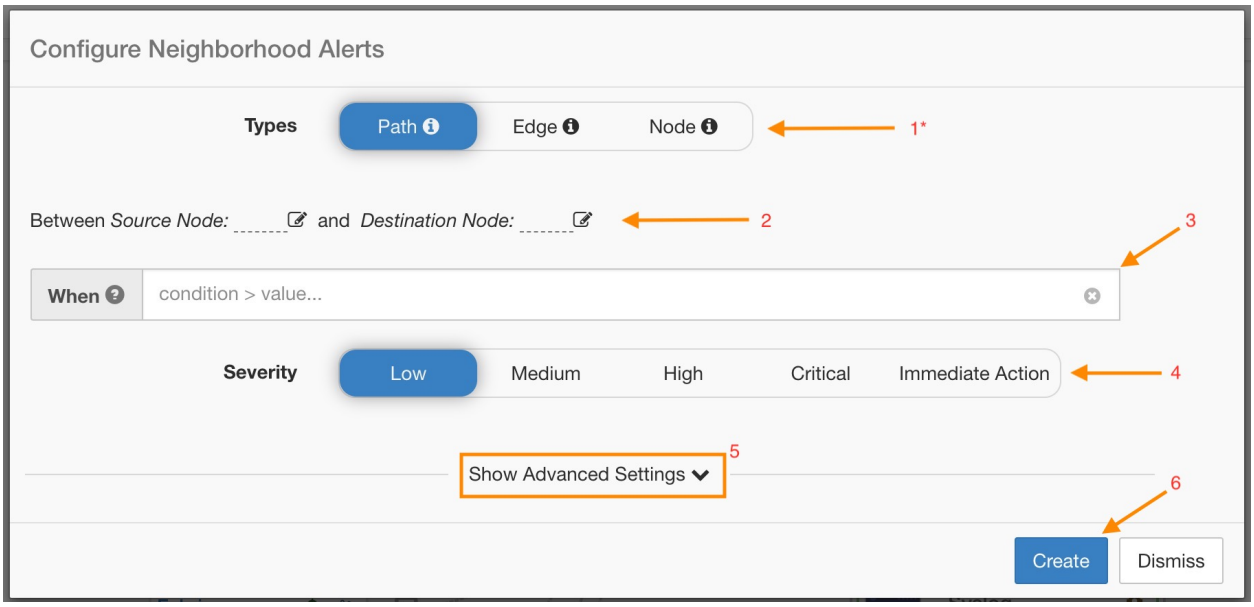


Fig. 10.1.2.1: Alert configuration modal

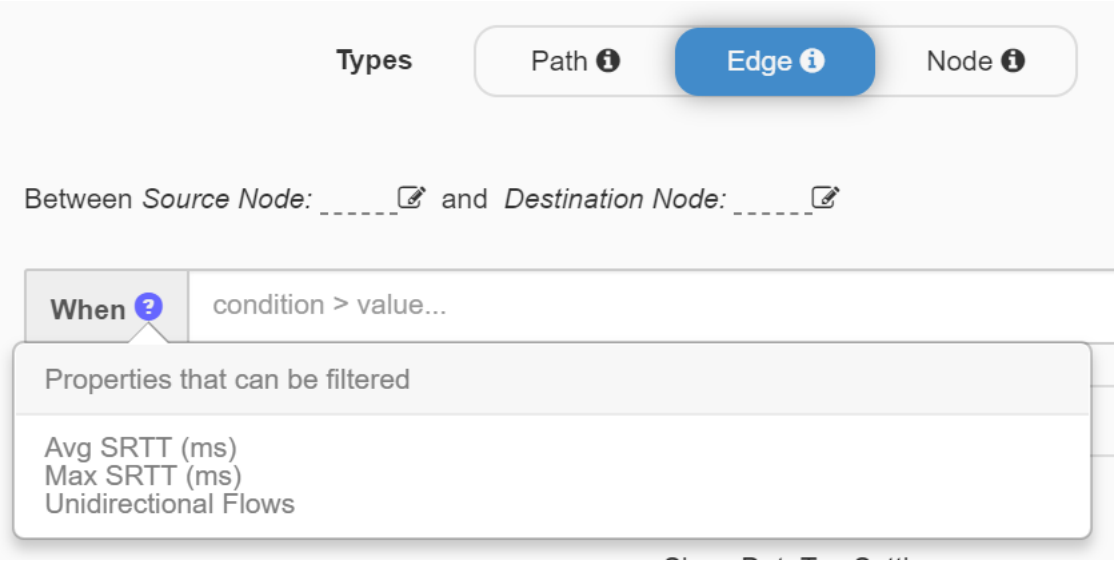


Fig. 10.1.2.2: Hovering over ? will display a list of available properties for creating an alert trigger. This list is context dependent on the type of alert selected.

Additional configuration options, shown when clicking "Show Advanced Settings":

1. Clicking "Hide Advanced Settings" will collapse the expansion.
2. Summary Alert options (if available). Availability is dependent on the app generating the alert. See *Summary Alerts* for more info.

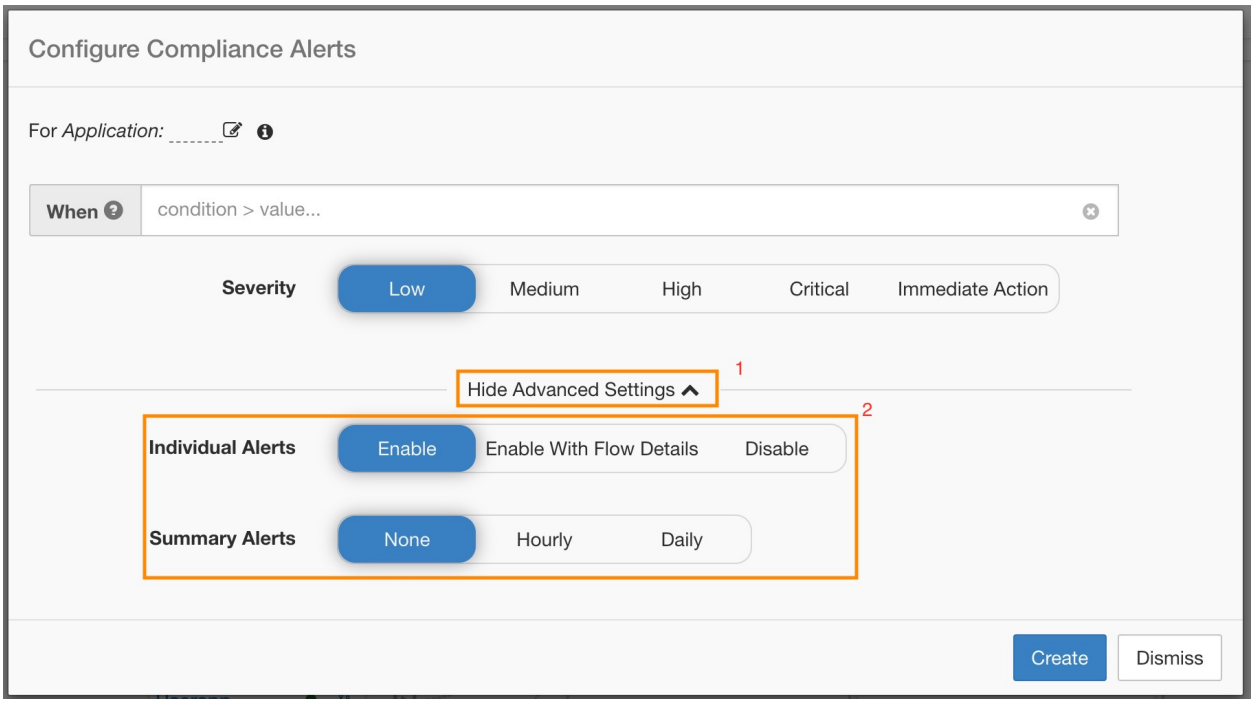


Fig. 10.1.2.3: Alert configuration modal advanced options

10.1.2.1 Summary Alerts

Summary Alerts are allowed for some apps, and configuration options are dependent on the app.

- "Individual Alerts" generally refers to alerts which are generated over non-aggregated (or minimally aggregated) information, and are likely to have a time range of 1 minute. Note that this does not necessarily mean the alerts are actually generated and sent at a minute interval; the individual alerts will still be generated at the *App Frequency* interval.
- "Summary Alerts" refers to alerts generated over metrics produced over an hour, or to the summarization of less frequent alerts.

App	App Frequency[1]	Individual Alerts	Hourly Alerts	Daily Alerts
Compliance	Minute	Yes: at app frequency	Summary of Individual	Summary of Individual
Neighborhood	Hourly	—	Yes	Summary of Hourly
Fabric	Hourly	Yes: minute[2]	Summary of Individual	Summary of Individual
Lookout Annotation	Hourly	—	Yes	—
Enforcement	Minute	Yes: at app frequency	Summary of Individual	Summary of Individual
Sensors	Minute	Yes: at app frequency	Summary of Individual	Summary of Individual

Note: **Event Time** shown in the UI of summary alerts represents the first occurrence of the same type alert over the past hour or a specified interval window

10.1.2.2 Note on Summarization versus Snoozing

Summarization applies to the entire set of alerts generated according the alert configuration, while snoozing applies to a specific alert. This distinction is minor when the alert configuration is very specific, but is notable when the alert configuration is broad.

- For example, Compliance configuration is quite broad: an application workspace, and on which type of violation an alert should be generated. Thus, summarization would apply to all alerts triggered by a 'escaped' condition, while snoozing would apply to a very specfic consumer scope, provider scope, provider port, protocol, and the escaped condition.
- On the opposite end, a Neighborhood alert configured to alert on a path between source scope and destination scope with a hop count less than some amount, will generate a very specific alert.

Other distinctions

- Snoozing will only result in an alert being sent when a new alert is generated after the snooze interval has passed. There is no indication of how many suppressed alerts might have occurred during the snooze interval.

[1] App Frequency is approximately how often the app runs and generates alerts. For example, Compliance has a flexible run frequency, and may actually compute alerts over a couple minutes together

[2] Fabric alerts are produced hourly when the app runs (note that the App Frequency is *hourly*), so in practice Fabric alerts will be produced and sent in batches after each hour of data is processed, even though the indiviudal alert option is a *minute* of data. This means that if the data would produce two alerts per minute, all 120 alerts are actually generated and sent at the end of the hour, and are likely to result in a summary alert showing in the UI.

- A summary alert will be generated at the specified frequency, so long as any alerts were generated within that interval. Summary alerts provide a count of the number of alerts triggered within the window, along with aggregated or range metrics.

10.1.3 Tetration Alerts Notifier (TAN)

Note: With release 3.3.1.x, TAN is moving to **Tetration Edge Appliance**.

Alert Notifiers provide capabilities to send alerts through various tools such as Amazon Kinesis, Email, Syslog and Slack in the currently selected scope. As scope owner or site admin, each notifier can be configured with required credentials and other information specific to the notifier application.

10.1.4 Configure Notifiers

To configure notifieres, first we need to enable connector. Alert related connectors can only be configured once Tetration Edge Appliance is deployed. See *Tetration Virtual Appliances for Connectors* for details on how to deploy Tetration Edge appliance.

After the Tetration Edge appliance is set up, you can configure each notifier with its specific required input. Note that once Tetration Edge appliance is set up, you will be able to see dashed lines connecting Alert Types to Internal Kafka(Data Tap). This is due the fact that notifier is build upon the Internal Kafka(Data Tap).

Please refer to Connectors for Alert Notifications for details on how to configure each alert notifier

10.1.5 Choose publishers for alerts

Scope owners and site admins can choose publishers to send alerts. Publishers includes Kafka (Data Tap) and notifiers.

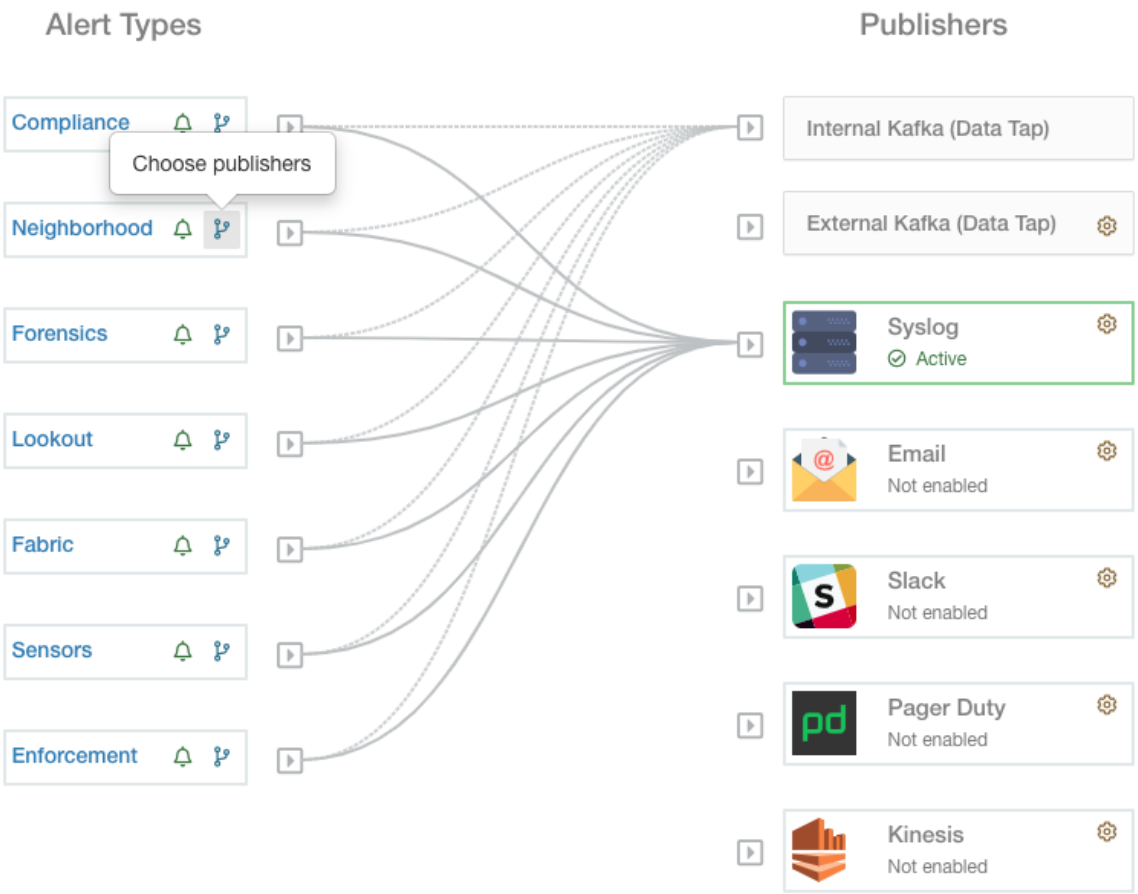


Fig. 10.1.5.1: Click the button shown in the figure to open a modal to select publishers for the alert type.

Note: Only Site Admins and scope owners are able to choose publishers to send alerts.

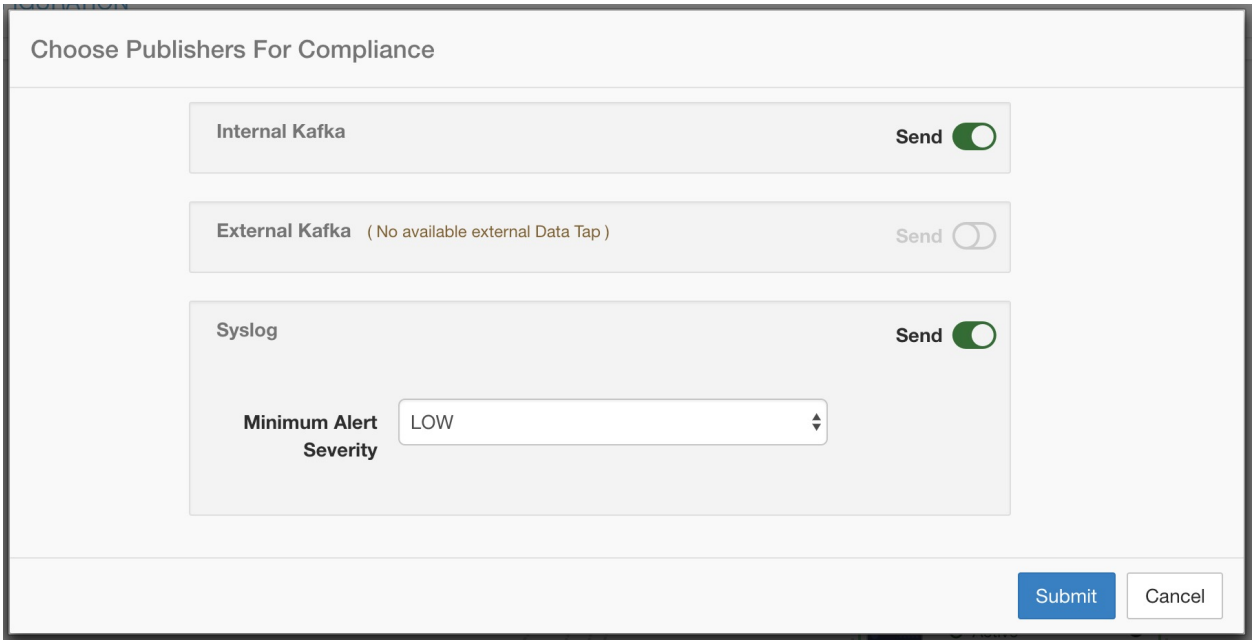


Fig. 10.1.5.2: All avaliable publishers will be displayed in this modal including the Internal Kafka/External Kafka and active notifiers. You can toggle the send button to choose the publishers for the alert type. Note that Minimum Alert Severity refers to the severity level where one certain alert must reach this level to be sent through publishers.

Note: Choosing external datataps can have an impact on the maximum number of alerts that can be processed; maximum number of alerts that can be processed could be reduced to up to 14000 alerts per minute batch.

10.1.6 External syslog tunneling moving to TAN

Note: Starting 3.1.1.x release, the syslog tunneling feature will move to TAN. To configure syslog for getting platform level syslog events, user would need to configure TAN on Tetration Edge appliance on `Default` Rootscope. Once Tetration Edge appliance is configued on `Default` Rootscope, syslog server can be setup as shown below. To enable platform alerts, enable syslog notifications for `Platform`. This can be done by enabling `Platform`-> `Syslog` connection.

Please refer to *Syslog Connector* for details of how to configure syslog.

10.1.7 Connection chart

The connection chart displays the connections between alert types and publishers. Once you choose a publisher for an alert type, a line will be established between that alert type and the publisher. Note that the line pointing to the Internal Kafka(Data Tap) will always be dashed line since it represent an internal mechanism of how alerts notification build upon.

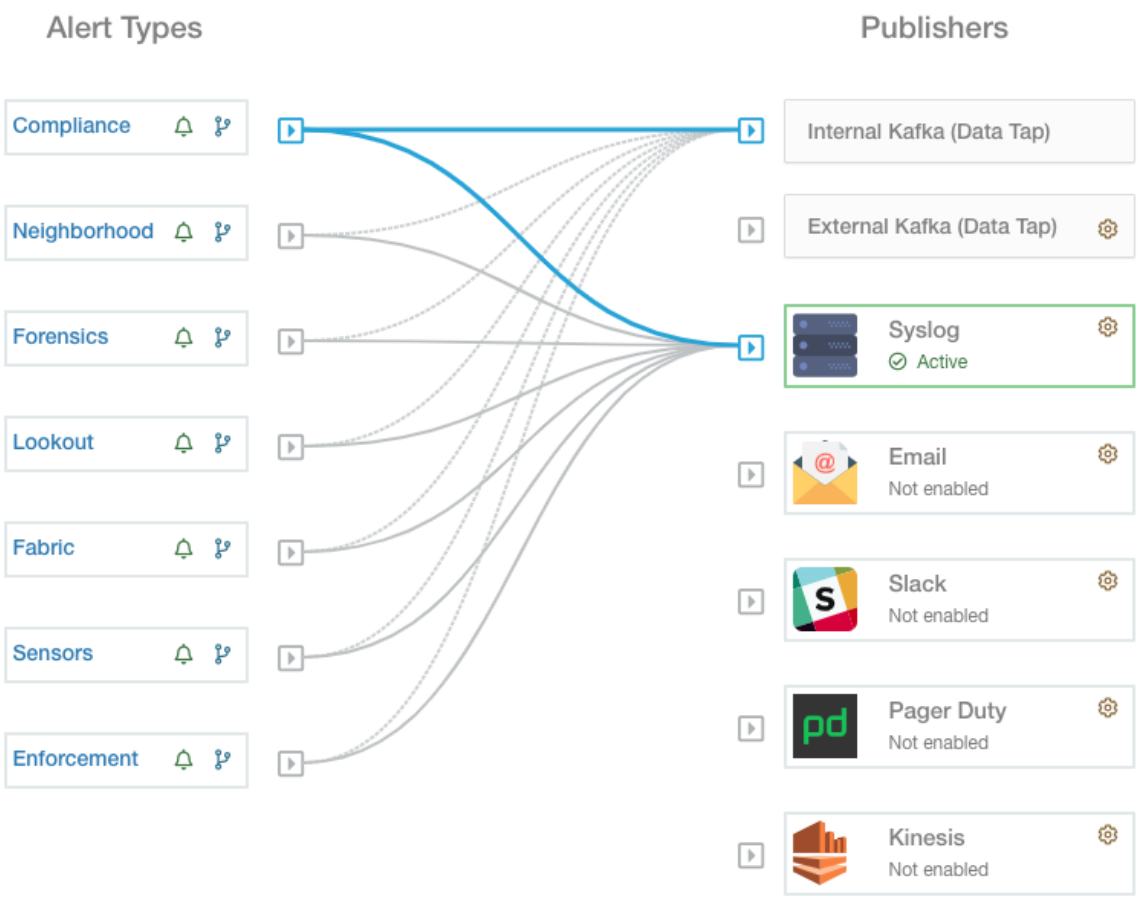


Fig. 10.1.7.1: As shown in this figure, once Syslog is chosen as a publisher for Neighborhood alerts, a line is established between them. Note that, hovering on the circled area in the figure will highlight the connections that are only associated with Neighborhood alerts.

Note: User App generated alerts are not shown in the Alert Configuration page. User Apps will be able to send messages and alerts to any configured Data Tap.

10.1.8 Viewing Alerts Trigger Rules

A list of all alerts trigger rules configured will show in the table below the Connection chart.

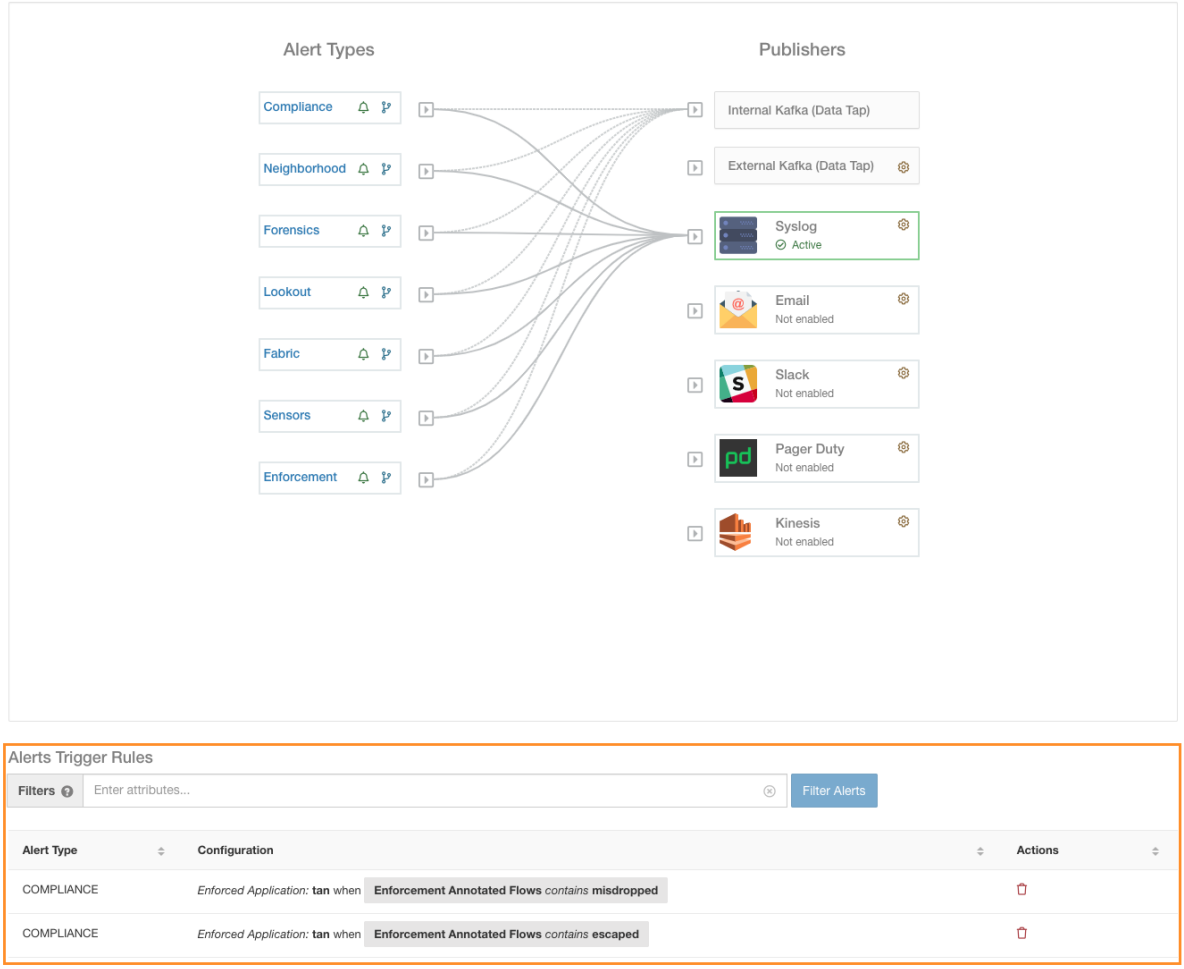


Fig. 10.1.8.1: Alerts Trigger Rules Table can be used to filter alerts trigger rules by alert type, alert frequency and alert trigger condition. **Note**: Alert trigger condition is an exact match condition.

10.1.8.1 Alerts Trigger Rules Details

Each row in the Alerts Trigger Rules Table can be clicked to expand with configuration details

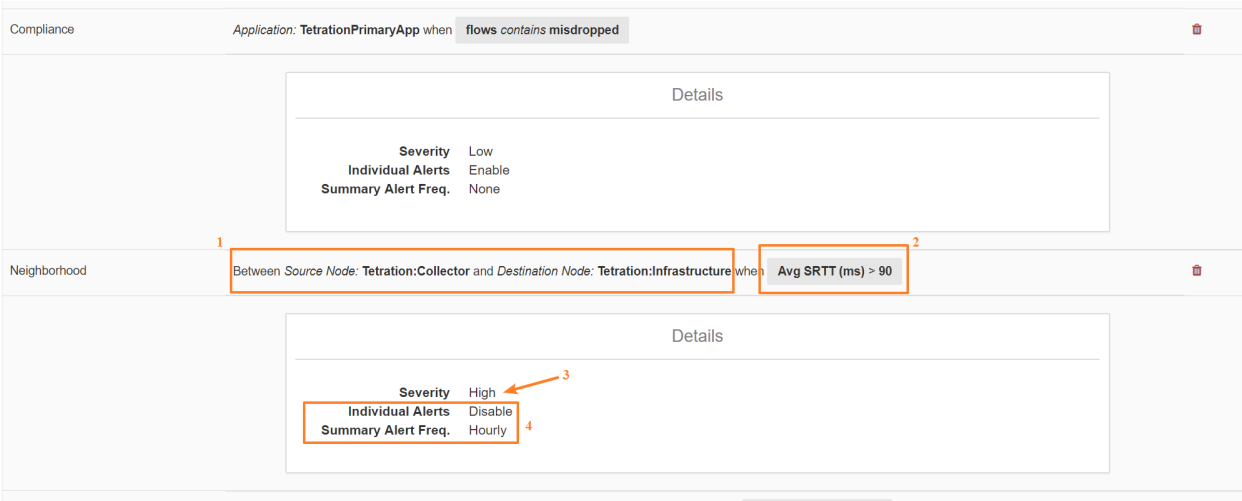


Fig. 10.1.8.1.1: Expanded alert configuration

1. Subject: *what an alert will be about*
2. Trigger: *when an alert will be generated*
3. Severity assigned to the alert (may affect which alerts are visible in the UI if there are many alerts generated at the same time)
4. Alert Frequency: Whether individual and/or summary alerts will be generated.

10.2 Current Alerts

The Alerts page is structured as shown below. Alerts can be filtered by type, status (active or snoozed), and severity (critical, high, medium, or low). By default, the listed alerts are filtered to `active` alerts (snoozed and muted alerts are not shown by default).

> **Warning:** Only alerts that contain severity value of LOW, MEDIUM, or HIGH will be shown in the **Alerts** page. All alerts irrespective to the severity values will always be sent to the configured kafka broker.

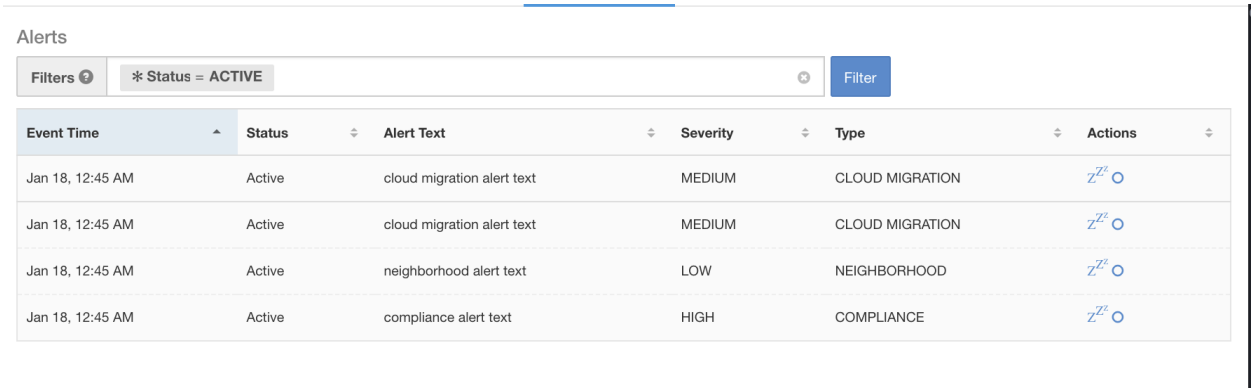


Fig. 10.2.1: Current alerts listing

Expanding for Alert Details

If more detail about a specific alert is desired, simply click on the alert to see further information.

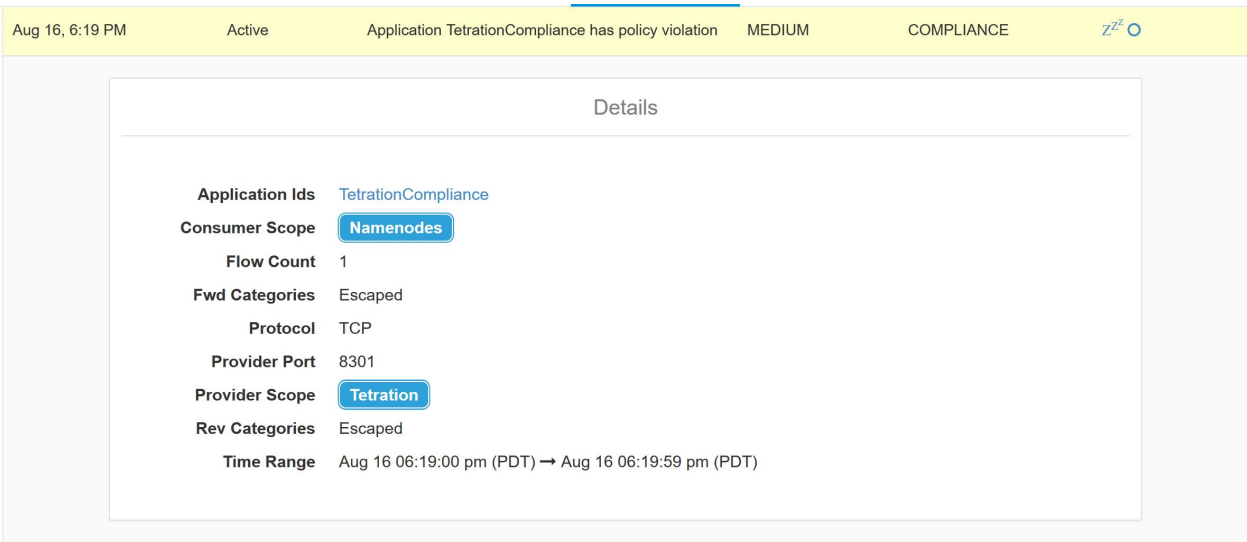


Fig. 10.2.2: Alert details

Note: If the volume of alerts per minute is high, a "Summary" alert will be displayed in the UI instead of individual alerts. Summary alerts can not be snoozed or hidden (muted).

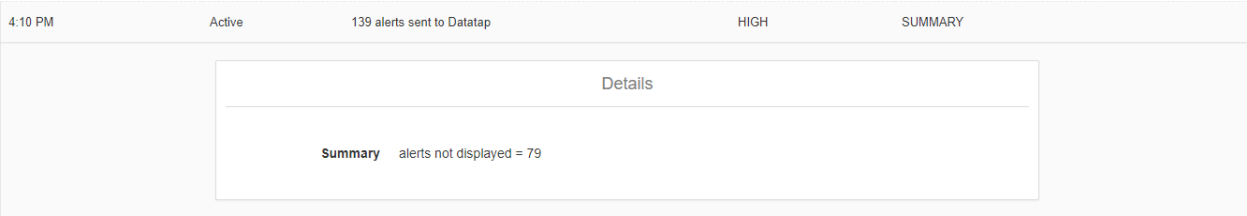


Fig. 10.2.3: Summary alert

Summary alert indicating how many alerts were sent to the Data Tap but suppressed from the UI

A note about viewing alerts in the UI

- Only 60 alerts/minute/root scope will be visible in the UI. A higher volume of alerts will result in the above mentioned summary alert in the UI
- Preference will be given to Critical alerts, then those with High severity, followed by Medium severity, and finally Low severity
- There is a maximum number of alerts visible in the UI at any point in time; older alerts will be dropped as new alerts come in.

See *Limits*

10.2.1 Snoozing Alerts

Note: User App created alerts can not be snoozed or ignored (muted) at this time.

The Alerts App allows for alerts of the same *'type'* to be snoozed (suppressed) for a chosen amount of time. Note that "type of alert" is defined differently depending on the application that Alerts has currently been configured for. As an example for Alerts on Compliance, "type of alert" is defined as the four tuple: consumer scope, provider scope, protocol, and provider port.

To see these fields for an alert, simply click on the alert in question to get the alert details.

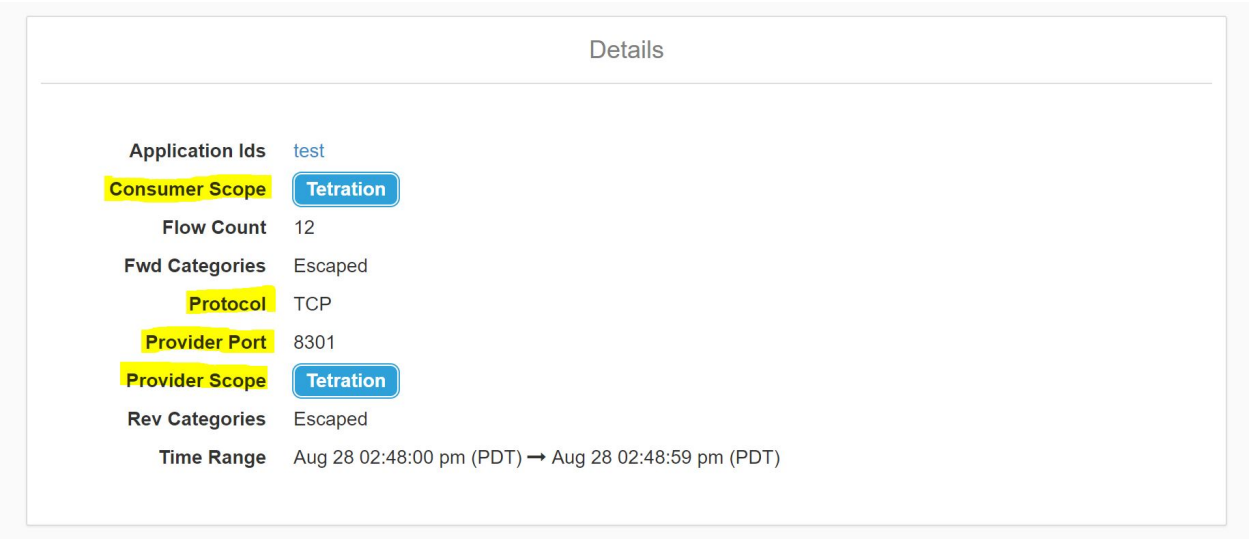


Fig. 10.2.1.1: Alert details

Snoozing an alert

To snooze an alert, click the snooze button under Actions for the particular alert type to be snoozed and specify the duration.

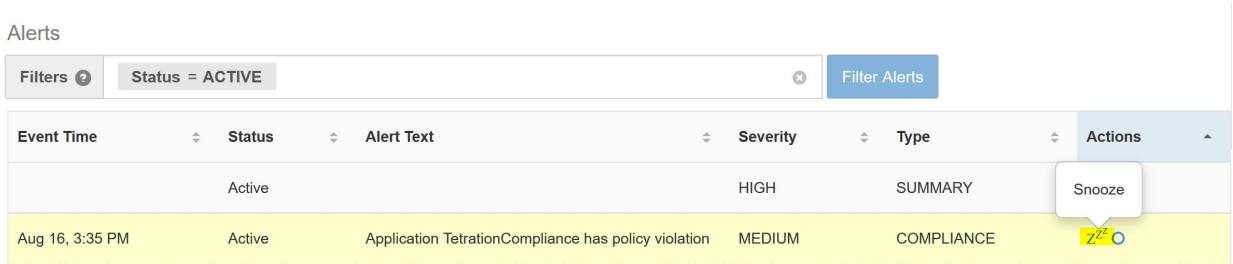


Fig. 10.2.1.2: Snoozing an alert

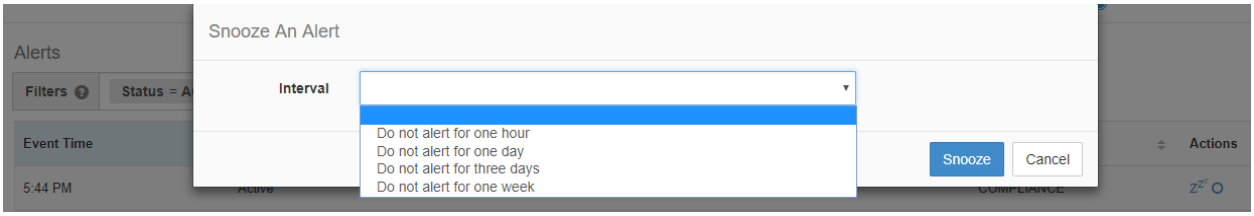


Fig. 10.2.1.3: Snoozing interval

As seen, alerts can be snoozed for four different intervals: one hour, one day, three days, or one week.

Muting an alert is essentially a snooze-forever action, and that button can also be found under Actions: ○

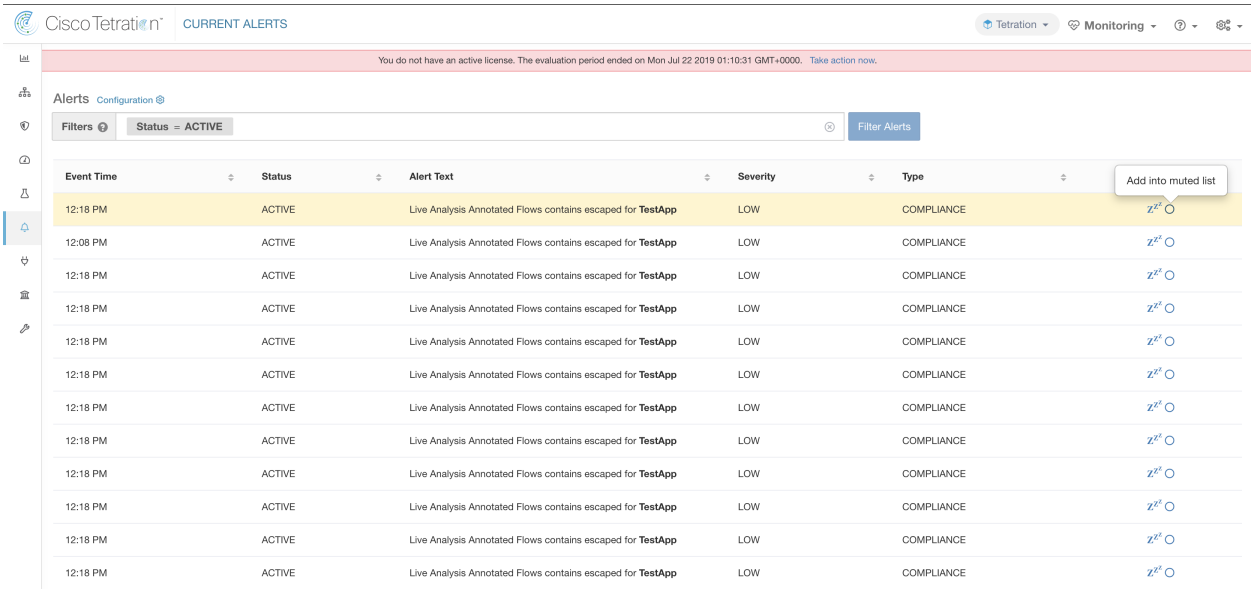


Fig. 10.2.1.4: Muting alerts by adding to muted list

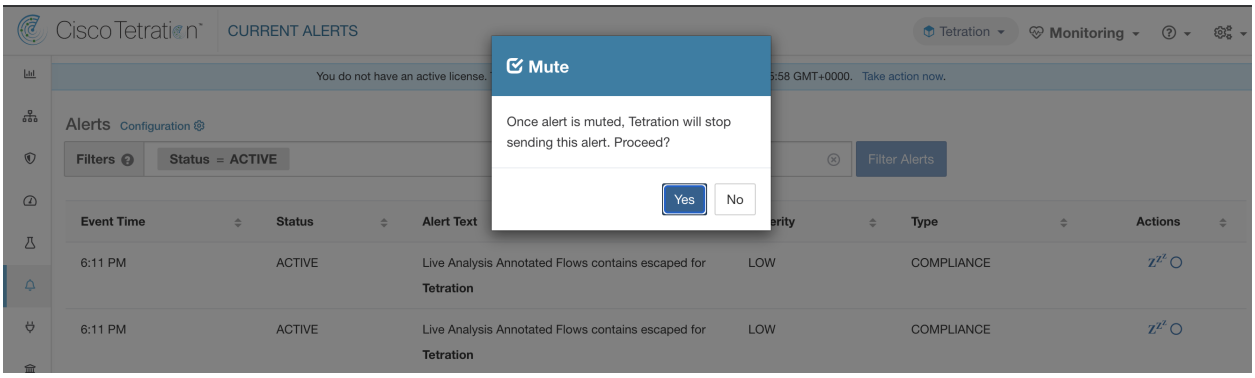


Fig. 10.2.1.5: Confirmation of muting an alert.

When an alert is 'muted', the user will not be sent this type of alert until the alert is removed from the muted list.

Removing snooze or muted state

To un-snooze a type of alert that was previously snoozed, first filter by snoozed alerts so only those alerts that have been snoozed are visible.

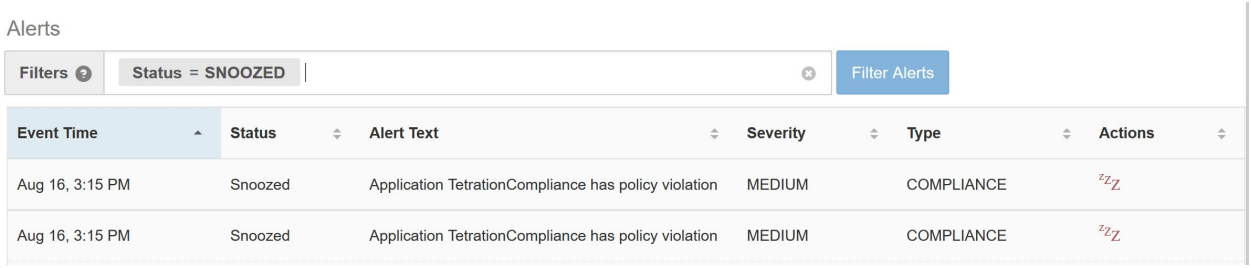


Fig. 10.2.1.6: Snoozed alerts filter

Then simply click the un-snooze button zzz for the desired alert under Actions as follows, and confirm the action.

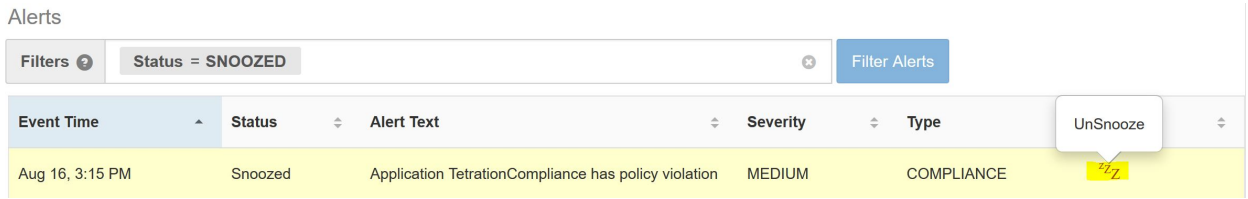


Fig. 10.2.1.7: Unsnooze alerts

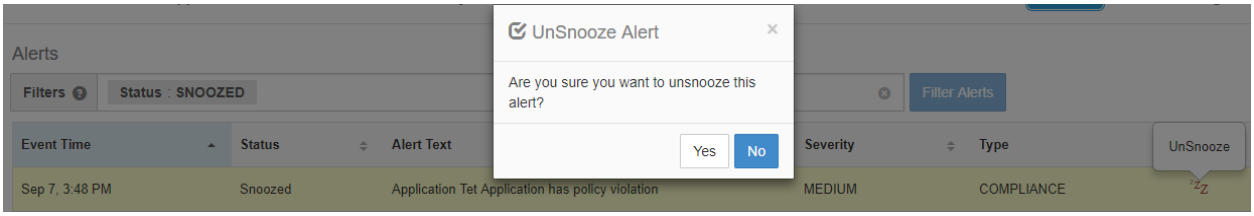


Fig. 10.2.1.8: Unsnooze alerts confirmation

This process is identical for removing an alert from the muted list, except filter by muted alerts.

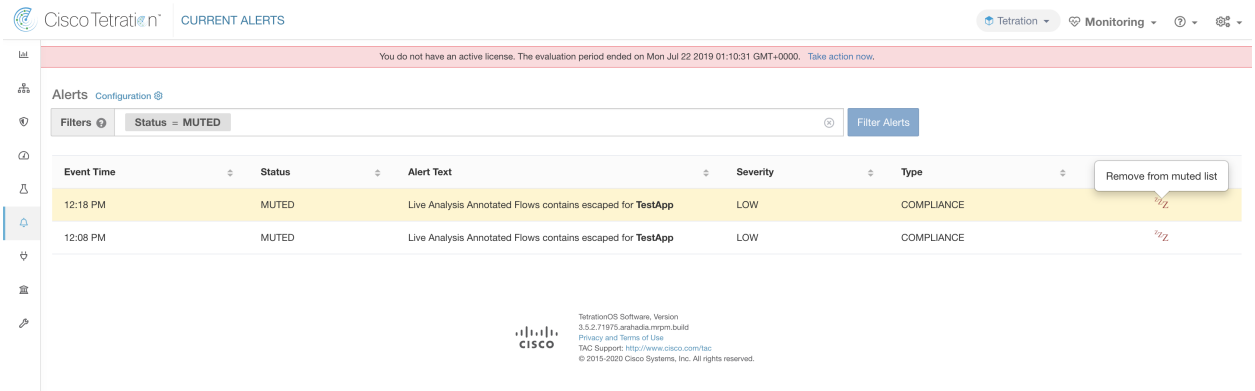


Fig. 10.2.1.9: Selecting “Muted” alerts and removing from muted list using zzz

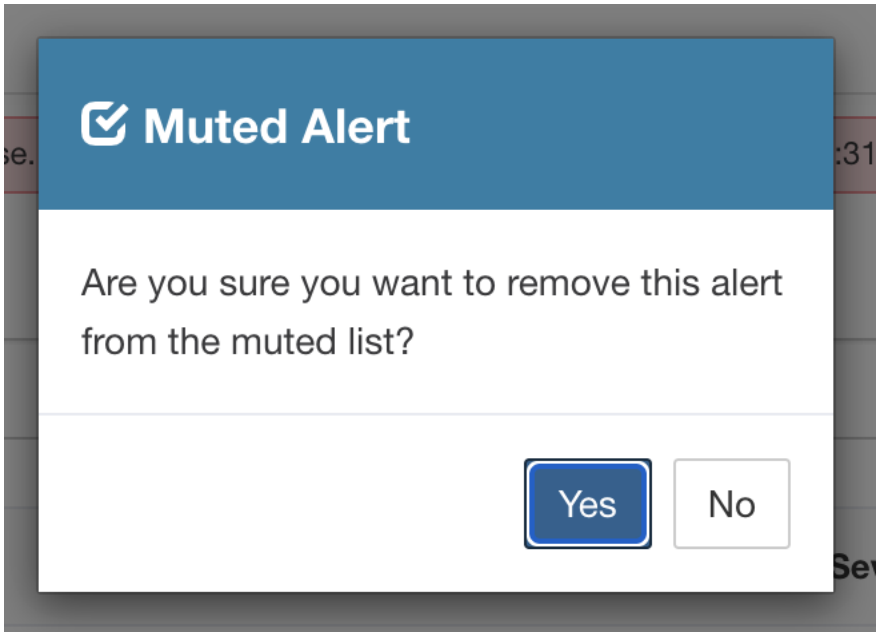


Fig. 10.2.1.10: Unmute alerts confirmation

Admiral Alerts

Admiral is an integrated alerting system which replaces Bosun from previous releases. More details can be found below at *Admiral Alerts*

10.3 Alert Details

10.3.1 Common Alert Structure

All alerts follow an overall common structure, but each type of alert will vary in its alert details.

The common structure is as follows. This structure corresponds to the json message structure available through Kafka DataTaps.

Field	Format	About
root_scope_id	string	Scope Id corresponding to top scope in scope hierarchy.
key_id	string	id field used for determining 'similar' alerts. Identical key_id's can be snoozed.
type	string	Type of the alert. Fixed set of string values: COMPLIANCE, NEIGHBORHOOD, USERAPP, FORENSICS, ENFORCEMENT, FABRIC, LOOKOUT_ANNOTATION, SENSOR, PLATFORM, FEDERATION, CONNECTOR
event_time	long	timestamp of when the event triggered (or if event spanned a range, then the beginning of the range). This timestamp is in epoch milliseconds (UTC).
alert_time	long	timestamp of when the alert was first attempted to be sent. This will be after the timerange of the event. This timestamp is in epoch milliseconds (UTC)
alert_text	string	Title of the alert
alert_text_with_names	string	Same content as alert_text but with any id fields replaced by corresponding name. This field may not exist for all alerts
severity	string	Fixed set of string values: LOW, MEDIUM, HIGH, CRITICAL, IMMEDIATE_ACTION. This is the severity of the alert. For some types of alerts these values are configurable.
alert_notes	string	Usually not set. May exist in some special cases for passing additional information through Kafka DataTaps
alert_conf_id	string	id of the alert configuration that triggered this alert. May not exist for all alerts
alert_details	string	Structured data. String-i-fied json. See feature details for specific alert type, since the exact structure of this field varies based on the type of alert.
alert_details_json	json	Same content of alert_details, but not string-i-fied. Only present for compliance alerts, and only available through Kafka.
tenant_id	string	May contain vrf corresponding to root_scope_id. Or may contain 0 as default value. Or may not be present at all.
alert_id	string	Internal generated temporary id. Best ignored.

The fields within *alert_details* vary based on the type of alert. See each feature section for explanation and list of fields:

- Compliance: *Alert Details*
- Neighborhood: *Alert Details*
- Lookout Annotation: *Lookout Alerts Details*
- Forensics: *Example* and *Forensic event fields*
- Sensor: *Sensor Alert Details*
- Enforcement: *Enforcement Alert Details*
- Connector: *Alert Details*

Additional alert types for on-prem clusters

- Fabric: fabric-alert-details
- Federation: federation-alert-details
- Platform: *Alert Details*

10.3.2 General Alert Format by Notifier

Variation across notifier types. The following contains examples of how alerts display across various notifier types.

10.3.2.1 Kafka (DataTaps)

Kafka (DataTap) messages are in JSON format. Example below; see above alert_details for some additional examples.


```
{
"severity": "LOW",
"tenant_id": 0,
"alert_time": 1595207103337,
"alert_text": "Lookout Annotated Flows contains TA_zeus for <scope_
↪id:5efcfdf5497d4f474f1707c2>",
"key_id": "0a4a4208-f721-398c-b61c-c07af3be9413",
"alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type='TETRATION_
↪PARQUET', location_name='lookout_annotation', location_grain='HOURLY', root_scope_
↪id='5efcfdf5497d4f474f1707c2'}/
↪bd33f37af32a5ce71e888f95ccfe845305e61a12a7829ca5f2d72bf96237d403",
"alert_text_with_names": "Lookout Annotated Flows contains TA_zeus for Scope Default",
"root_scope_id": "5efcfdf5497d4f474f1707c2",
"alert_conf_id": "5f10c7141a0c236b78148da1",
"type": "LOOKOUT_ANNOTATION",
"event_time": 1595204760000,
"alert_details": "{\"dst_scope_id\":[\"5efcfdf5497d4f474f1707c2\"],\"dst_scope_names\
↪":[\"Default\"],\"dst_hostname\":\"\",\"src_scope_id\":[\"5efcfdf5497d4f474f1707c2\
↪"],\"lookout_tags\":[\"TA_compromised_zeus\",\"TA_zeus\"],\"dst_address\":\"172.26.
↪231.255\",\"fwd_packet_count\":3,\"src_scope_names\":[\"Default\"],\"src_port\":137,
↪\"protocol\":\"UDP\",\"internal_trigger\":{\"datasource\":\"lookout_annotation\",\
↪"rules\":{\"field\":\"lookout_tags\",\"type\":\"contains\",\"value\":\"TA_zeus\"},\
↪"label\":\"Alert Trigger\"},\"scope_id\":\"5efcfdf5497d4f474f1707c2\",\"time_range\
↪":[1595204760000,1595204820001],\"src_address\":\"172.26.230.124\",\"dst_port\":137,
↪\"rev_packet_count\":0,\"src_hostname\":\"\"}"
}
```


10.3.2.2 Email

Information about configuring Email alerts: *Email Connector*

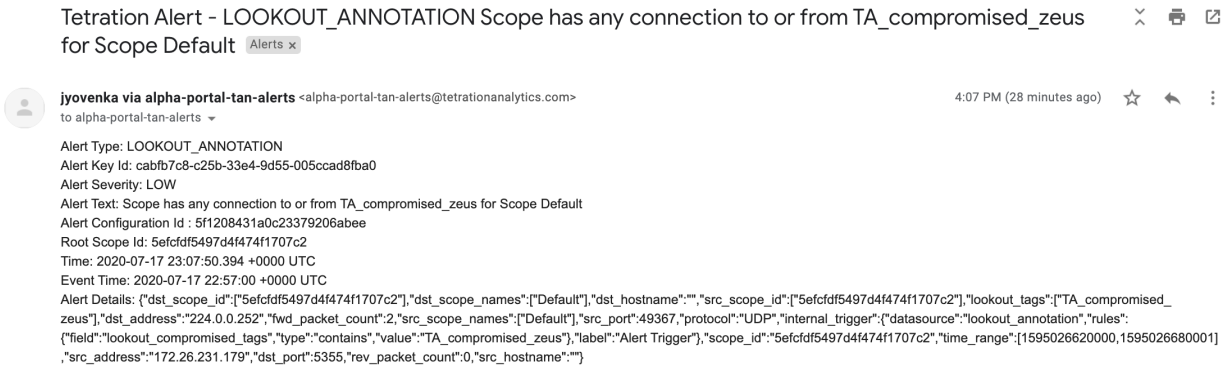


Fig. 10.3.2.2.1: Example of a Tetration alert when configured to send to email.

10.3.2.3 PagerDuty

Information about configuring PagerDuty alerts: *PagerDuty Connector*

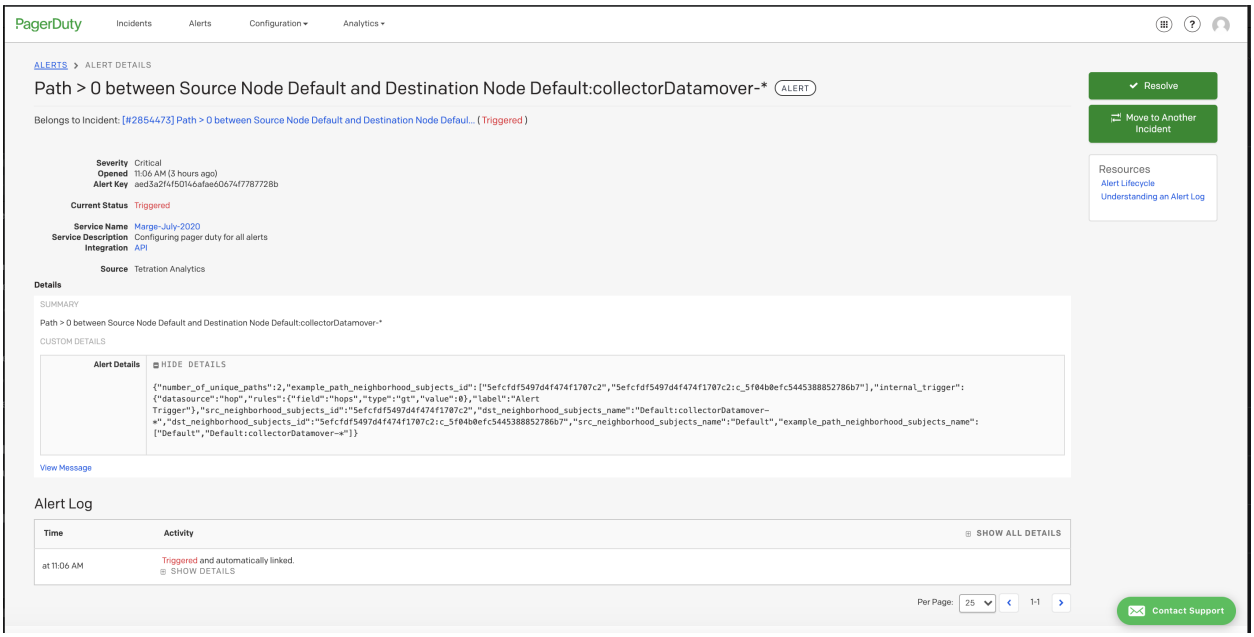


Fig. 10.3.2.3.1: Example of a Tetration alert in PagerDuty.

Alerts sent to PagerDuty will be considered a re-trigger of the same alert based on the key_id.

Severity is mapped to PagerDuty severity as follows:

Tetration Severity	PagerDuty Severity
IMMEDIATE_ACTION	critical
CRITICAL	critical
HIGH	error
MEDIUM	warning
LOW	info

10.3.2.4 Syslog

Information about configuring Syslog alerts, and adjusting severity mapping: *Syslog Connector*

```
Aug  2 18:45:21 tan-5f035bae1a0c231d5880d7f8-tac-demo-data-ingest Tetration Alert[26841]: [DEBUG] {"keyId":"3ee0d8b7-bc81-3427-9e84-6b9f8fedb98c","eventTime":"159639
3720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"L
OW","tenantId":"0","type":"COMPLIANCE","alertDetails":"{\"consumer_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"consumer_scope_names\":[\"Default\"],\"provider_scope
_names\":[\"Default\"],\"provider_port\":53,\"application_id\":\"5f04b0b9755f024d4e36a279\",\"constituent_flows\":[{\"consumer_port\":37367,\"protocol\":\"UDP\",\"co
nsumer_address\":\"172.31.163.139\",\"provider_address\":\"171.70.168.183\",\"provider_port\":53},{\"consumer_port\":39652,\"protocol\":\"UDP\",\"consumer_address\":
\"172.31.163.137\",\"provider_address\":\"171.70.168.183\",\"provider_port\":53},{\"consumer_port\":63811,\"protocol\":\"UDP\",\"consumer_address\":\"172.31.163.136\
",\"provider_address\":\"171.70.168.183\",\"provider_port\":53},{\"consumer_port\":57418,\"protocol\":\"UDP\",\"consumer_address\":\"172.31.163.138\",\"provider_addr
ess\":\"173.36.131.10\",\"provider_port\":53},{\"consumer_port\":12599,\"protocol\":\"UDP\",\"consumer_address\":\"172.31.163.141\",\"provider_address\":\"173.36.131
.10\",\"provider_port\":53},{\"consumer_port\":7385,\"protocol\":\"UDP\",\"consumer_address\":\"172.31.163.140\",\"provider_address\":\"173.36.131.10\",\"provider_po
rt\":53}],\"escaped_count\":6,\"provider_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"policy_type\":\"ENFORCED_POLICY\",\"protocol\":\"UDP\",\"internal_trigger\":{\"
datasource\":\"compliance\",\"rules\":{\"field\":\"policy_violations\",\"type\":\"contains\",\"value\":\"escaped\"},\"label\":\"Alert Trigger\"},\"time_range\":[1596
393720000,1596393779999],\"policy_category\":[\"ESCAPED\"]}","rootScopeId":"5efcfdf5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"
Enforcement Annotated Flows contains escaped for Enforced Application j1"}
Aug  2 18:45:21 tan-5f035bae1a0c231d5880d7f8-tac-demo-data-ingest Tetration Alert[26841]: [DEBUG] {"keyId":"8f0cfcb5-f8c1-3130-a069-3721b7d50159","eventTime":"159639
3720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"L
OW","tenantId":"0","type":"COMPLIANCE","alertDetails":"{\"consumer_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"consumer_scope_names\":[\"Default\"],\"provider_scope
_names\":[\"Default\"],\"provider_port\":5660,\"application_id\":\"5f04b0b9755f024d4e36a279\",\"constituent_flows\":[{\"consumer_port\":17131,\"protocol\":\"TCP\",\"
consumer_address\":\"172.26.231.193\",\"provider_address\":\"172.31.163.140\",\"provider_port\":5660}],\"escaped_count\":1,\"provider_scope_ids\":[\"5efcfdf5497d4f47
4f1707c2\"],\"policy_type\":\"ENFORCED_POLICY\",\"protocol\":\"TCP\",\"internal_trigger\":{\"datasource\":\"compliance\",\"rules\":{\"field\":\"policy_violations\",\"
type\":\"contains\",\"value\":\"escaped\"},\"label\":\"Alert Trigger\"},\"time_range\":[1596393720000,1596393779999],\"policy_category\":[\"ESCAPED\"]}","rootScopeI
d":"5efcfdf5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"Enforcement Annotated Flows contains escaped for Enforced Application j1
"}
Aug  2 18:45:21 tan-5f035bae1a0c231d5880d7f8-tac-demo-data-ingest Tetration Alert[26841]: [DEBUG] {"keyId":"1ef4a974-be89-31de-abe9-dc71cb0170ad","eventTime":"159639
3720000","alertTime":"1596393968822","alertText":"Enforcement Annotated Flows contains escaped for \u003capplication_id:5f04b0b9755f024d4e36a279\u003e","severity":"L
OW","tenantId":"0","type":"COMPLIANCE","alertDetails":"{\"consumer_scope_ids\":[\"5efcfdf5497d4f474f1707c2\"],\"consumer_scope_names\":[\"Default\"],\"provider_scope
_names\":[\"Default\"],\"provider_port\":443,\"application_id\":\"5f04b0b9755f024d4e36a279\",\"constituent_flows\":[{\"consumer_port\":17792,\"protocol\":\"TCP\",\"c
onsumer_address\":\"172.26.231.193\",\"provider_address\":\"172.31.163.133\",\"provider_port\":443}],\"escaped_count\":1,\"provider_scope_ids\":[\"5efcfdf5497d4f474f
1707c2\"],\"policy_type\":\"ENFORCED_POLICY\",\"protocol\":\"TCP\",\"internal_trigger\":{\"datasource\":\"compliance\",\"rules\":{\"field\":\"policy_violations\",\"t
ype\":\"contains\",\"value\":\"escaped\"},\"label\":\"Alert Trigger\"},\"time_range\":[1596393720000,1596393779999],\"policy_category\":[\"ESCAPED\"]}","rootScopeId"
:"5efcfdf5497d4f474f1707c2","alertConfId":"5f15cca71a0c231ebd66ca3b","alertTextWithNames":"Enforcement Annotated Flows contains escaped for Enforced Application j1"}
```


Fig. 10.3.2.4.1: Example of several Tetration alerts sent to syslog.

10.3.2.5 Slack

Information about configuring Slack alerts: *Slack Connector*

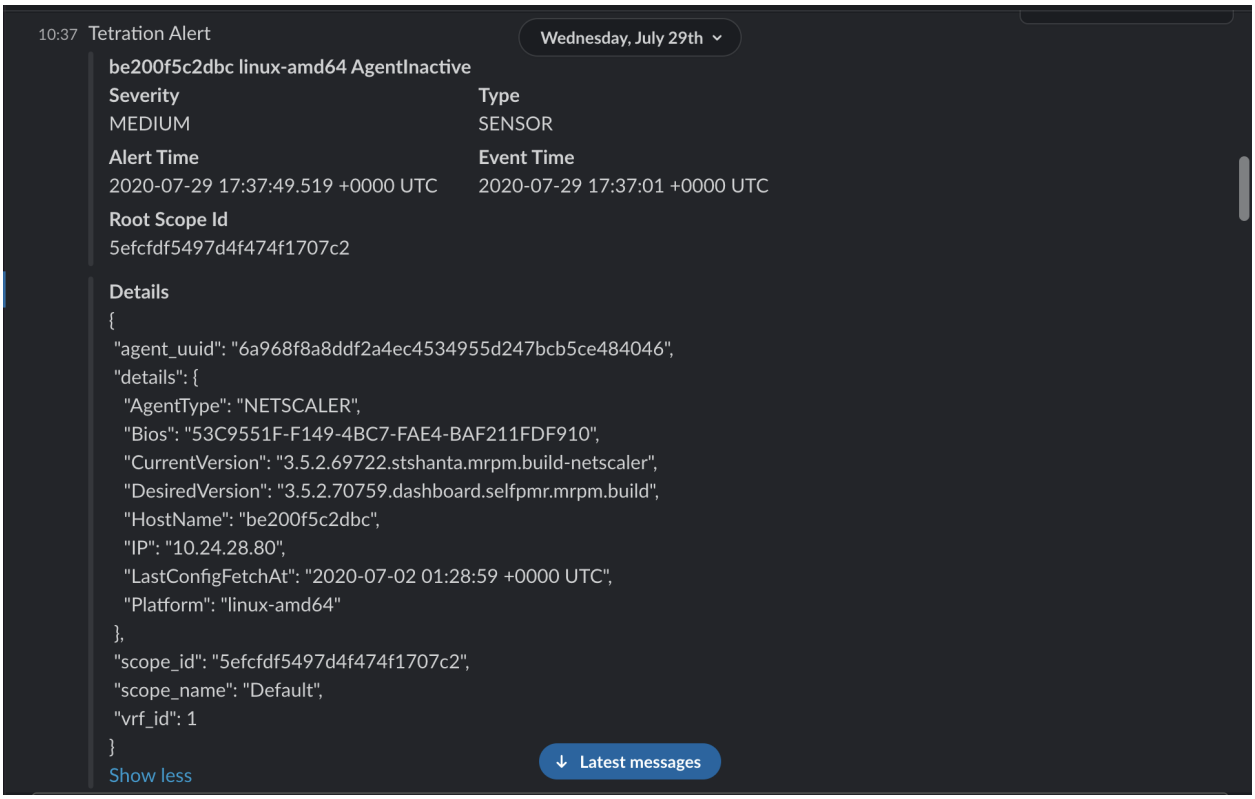


Fig. 10.3.2.5.1: Example of a Tetration alert sent to slack channel.

10.3.2.6 Kinesis

Information about configuring Kinesis alerts: *Kinesis Connector*

Kinesis alerts are similar to Kafka alerts, as these are both message queues.

CHAPTER

ELEVEN

MAINTENANCE

The Maintenance item in the top-level menu can be expanded to see all pages available based on your role.

11.1 Service Status

The **Service Status** page displays the health of all services that are used in Cisco Tetration cluster along with their dependencies. The graph view shows the health of the service, each node in the graph shows the health of the service and an edge represents dependency on other services. Unhealthy services are marked either red when the service is unavailable and orange when the service is degraded but available. A green node will indicate that the service is healthy. For more debug information on these nodes, use tree view which has the **Expand All** button to show all child nodes in the dependency tree. "Down" indicates that the service is not functional and "Unhealthy" indicates that the service is not fully functional.

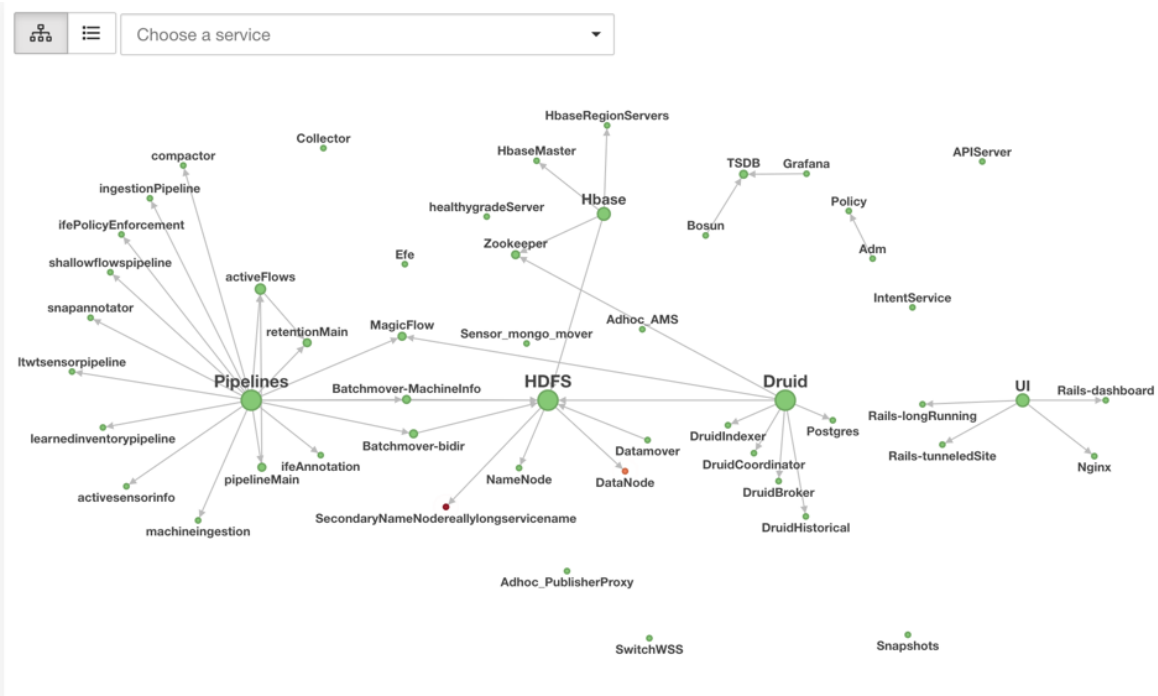
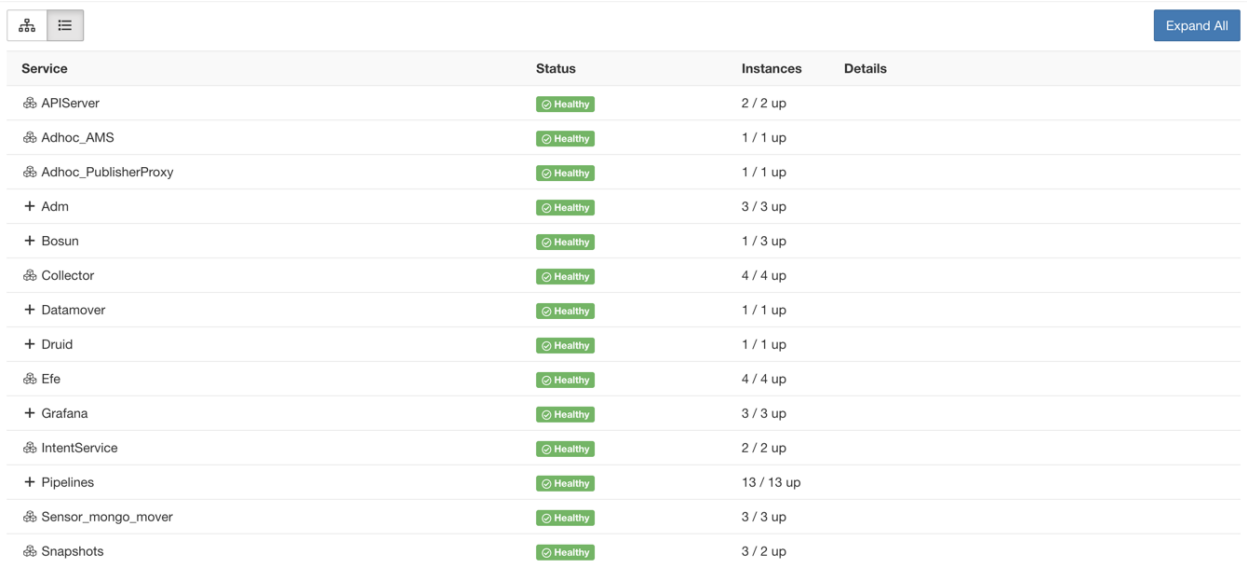


Fig. 11.1.1: Service Status page



Expand All

Service	Status	Instances	Details
APIServer	Healthy	2 / 2 up	
Adhoc_AMS	Healthy	1 / 1 up	
Adhoc_PublisherProxy	Healthy	1 / 1 up	
Adm	Healthy	3 / 3 up	
Bosun	Healthy	1 / 3 up	
Collector	Healthy	4 / 4 up	
Datamover	Healthy	1 / 1 up	
Druid	Healthy	1 / 1 up	
Efe	Healthy	4 / 4 up	
Grafana	Healthy	3 / 3 up	
IntentService	Healthy	2 / 2 up	
Pipelines	Healthy	13 / 13 up	
Sensor_mongo_mover	Healthy	3 / 3 up	
Snapshots	Healthy	3 / 2 up	

Fig. 11.1.2: Service Status page

11.2 Admiral Alerts

Admiral is an integrated alerting system which replaces Bosun (Monitoring -> Sentinel under Customer Support privilege) from previous releases. It processes alerts off the service health reported by *Service Status*. Thus, users have a unified way of determining service/cluster health. Service Status shows the current (point in time) health of a service. The service is considered down when it reports as red on service status, otherwise it is considered as up. Uptime is the time when the service is reported as up. Admiral evaluates service health reported by service status over a period of time and raises an alert if the service uptime percentage falls below a certain thresold. This evaluation over a duration of time ensures that we reduce false positives and alert only on true service outages.

As services are different in their alerting needs, this percentage and time interval are fixed differently for different services.

Customers can use admiral notifications to be notified of these events. They are also visible on the Current Alerts UI under type PLATFORM.

Note: Only a chosen subset of services have an admiral alert associated with them. If a service is not in the above subset, no admiral alert will be raised when it goes down. This subset of services with admiral alerts and their alerting threshold percentages and time intervals are fixed i.e. not user configurable.

The following sections describe admiral alerts and notifications in more detail.

11.2.1 Lifecycle of Admiral Alert

Admiral checks for the uptime of services on service status. It raises an alert when this uptime becomes lower than the pre-configured threshold for alerting.

As an example, Rpminstall is a service which is used to install rpms during deploy, upgrades, patches etc. It is configured to generate an admiral alert if its uptime is less than 80% over one hour. If Rpminstall service goes

down for a duration longer than the threshold specified above, an admiral alert for Rpminstall is generated with status ACTIVE.

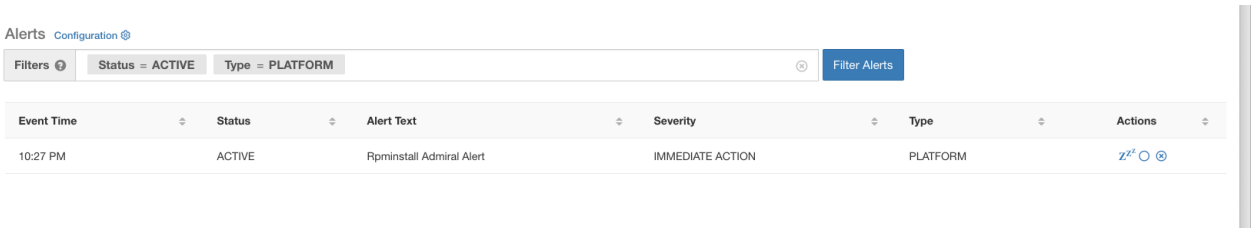


Fig. 11.2.1.1: Active Admiral Alert

When the service recovers, its uptime percentage starts increasing. When the uptime goes higher than its threshold , the alert auto closes and its status moves to CLOSED. In the Rpminstall example described above, Rpminstall Admiral Alert will auto close when its uptime goes over 80% in one hour.

Note: The close of alert will ALWAYS lag the service becoming normal. This is because admiral looks at service health over a duration of time. In the above example, since Rpminstall alert threshold is set to 80% of an hour of uptime , it will need to be up for at least 48 minutes (80% of one hour) before the alert will close.

There is NO action required from the user to close the alert. This ensures that all ACTIVE admiral alerts indicate a current underlying issue that needs attention.

Note: No dedicated notification is generated when alerts close.

Once an alert moves to CLOSED, it will no longer show under ACTIVE alerts. Closed alerts can still be seen on the UI using the filter Status=CLOSED as shown below:

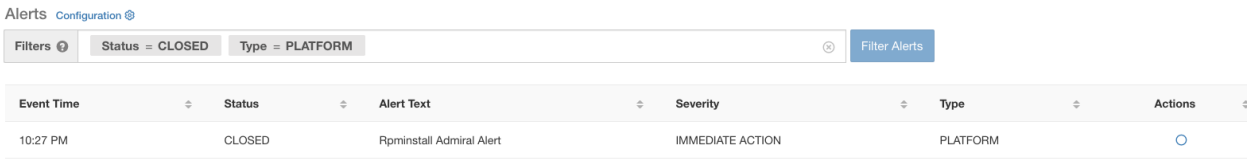


Fig. 11.2.1.2: Admiral Alert Auto Closes When Service Recovers

There are two kinds of admiral alerts:

1. Individual Admiral Alert
2. Admiral Summary Alert

11.2.2 Individual Admiral Alert

The alerts described above i.e. the ones raised for individual services come in this category. Their alert Text always contains <Service Name> Admiral Alert. This makes it easy to filter individual alerts by service or by the "Admiral Alert" suffix.

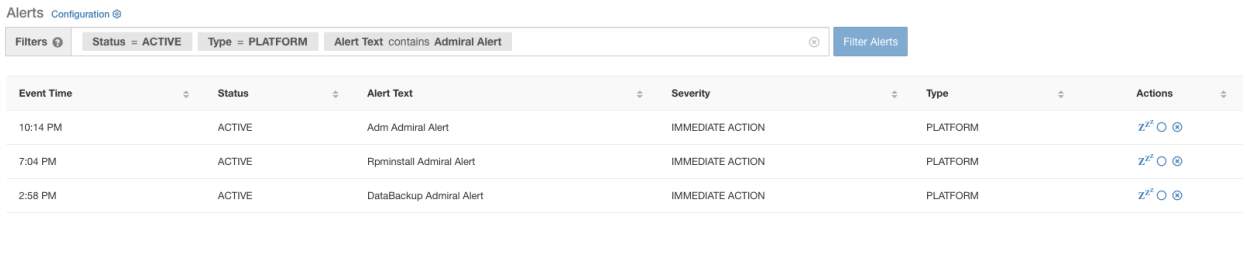


Fig. 11.2.2.1: Alert Text Filter For Individual Admiral Alerts

More attributes of this service are described in _admiral_indiv_details-label

11.2.3 Summary Admiral Alert

Admiral generates daily Summary Alerts at midnight UTC. They contain a list of currently active alerts and all alerts closed within the last one day. This allows the user to see the overall cluster health reported by admiral in one place. This is also useful to see closed alerts which do not generate a dedicated notification otherwise. If the cluster is healthy and no alerts were closed within the last one day, no summary notifications are generated for that day. This is done to reduce unnecessary notifications and noise.

The Alerts Text in this case is always "Admiral Summary". This makes it easy to filter summary alerts as shown below.

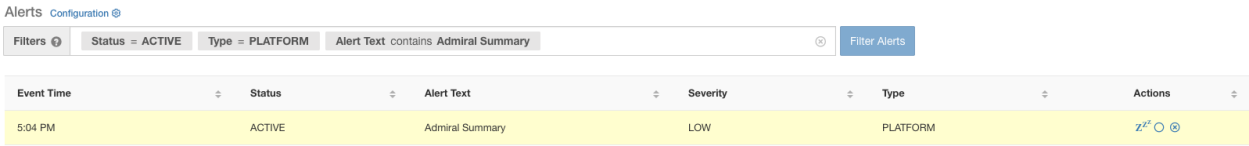


Fig. 11.2.3.1: Admiral Summary Text Filter

More attributes of this service are described in _admiral_summary_dets-label

11.2.4 Alert Details

Individual Alerts

On clicking the alert for an individual admiral alert, it expands to show fields useful for debugging and analyzing the alert.

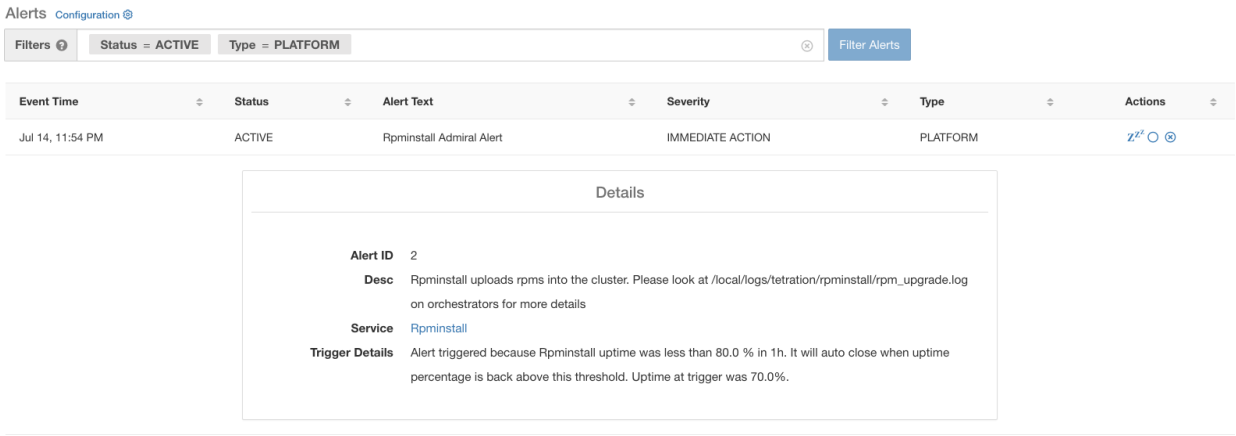


Fig. 11.2.4.1: Alert Details

The various fields are:

Field	About
Alert ID	Each alert has a unique id called its Alert ID. This helps uniquify a particular incidence of a service going down. As mentioned earlier, when the underlying uptime of the service being reported by the alert becomes normal, the alert auto closes. If the same service goes down again subsequently, a new alert with a different Alert ID is generated. Thus the alert id helps uniquify each incident of the alert being raised.
Desc	The description field contains additional information about the service issue causing the alert.
Service	This contains a link taking the user to the service status page where the current status of the service can be seen. User can also get more details on why the service is being marked down in the service status page.
Trigger Details	This contains the details on the trigger thresholds for the service. User can understand when to expect the alert to close after it's underlying service is restored by looking at these thresholds. For eg: Rpminstall threshold is mentioned as 80% uptime over one hour. Thus rpminstall service needs to be up for at least 48 minutes (80% of one hour) before the alert will auto close. This also shows the uptime value seen for the service when the alert was fired.

A sample JSON Kafka output is shown below:

```
{
"severity": "IMMEDIATE_ACTION",
"tenant_id": 0,
"alert_time": 1595630519423,
"alert_text": "Rpminstall Admiral Alert",
"key_id": "ADMIRAL_ALERT_5",
"alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type='TETRATION',␣
↪location_name='platform', location_grain='MIN', root_scope_id=
↪'5efcfdf5497d4f474f1707c2'}/
↪66eb975f5f987fe9eaefa81cee757c8b6dac5facc26554182d8112a98b35c4ab",
"root_scope_id": "5efcfdf5497d4f474f1707c2",
"type": "PLATFORM",
```

(continues on next page)

(continued from previous page)

```
"event_time": 1595630511858,
"alert_details": "{\"Alert ID\":5,\"Service\":\"Rpminstall\",\"Desc\":\"Rpminstall␣
↪uploads rpms into the cluster. Please look at /local/logs/tetration/rpminstall/rpm_
↪upgrade.log on orchestrators for more details\",\"Trigger Details\":\"Alert␣
↪triggered because Rpminstall uptime was less than 80.0 % in 1h. It will auto close␣
↪when uptime percentage is back above this threshold. Uptime at trigger was 65.0%. \
↪"}"
}
```

All individual alerts follow the above format. The services (from service status) which are covered by admiral monitoring are listed below:

Service	Trigger Condition	Severity
Adm	Service Uptime falls below 90% in last one hour.	IMMEDIATE ACTION
DataBackup	Service Uptime falls below 90% in last 6 hours.	IMMEDIATE ACTION
DiskUsageCritical	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION
RebootRequired	Service Uptime falls below 90% in last one hour.	IMMEDIATE ACTION
Rpminstall	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION
SecondaryNN_checkpoint_status	Service Uptime falls below 90% in last one hour.	IMMEDIATE ACTION

For 8RU/39 RU physical clusters, the following services are monitored additionally:

Service	Trigger Condition	Severity
DIMMFailure	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION
DiskFailure	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION
FanSpeed	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION
ClusterSwitches	Service Uptime falls below 80% in last one hour.	IMMEDIATE ACTION

Note: Admiral relies on processing metrics generated by Service Status to generate alerts. If metric retrieval is not possible for a prolonged duration (For Eg: If service status is down), then an alert (TSDBOracleConnectivity) is raised notifying that service based alert processing is off on the cluster.

Summary Alerts

Summary alerts are informational in nature and are always set to LOW priority. On clicking an admiral summary alert, it expands to show various fields containing summary information on admiral alerts.

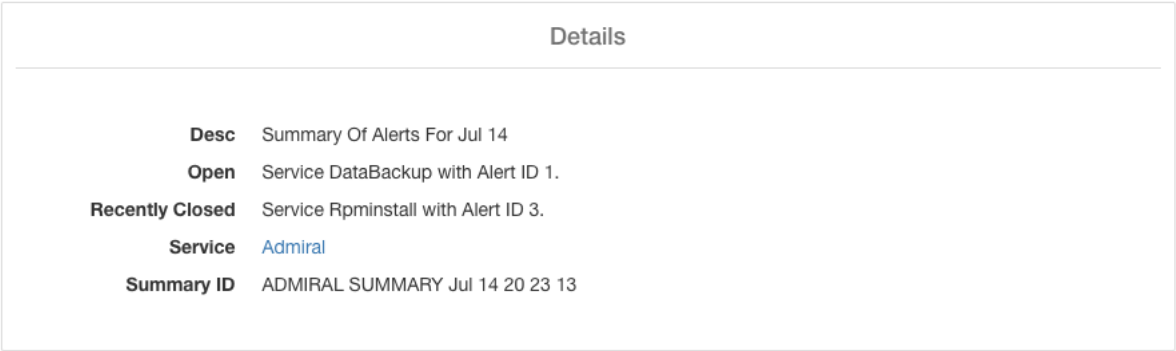


Fig. 11.2.4.2: Details for Admiral Summary Alert

Field	About
Desc	The description field contains the day for the daily summary.
Open	The open alerts indicate which alerts were active when the summary was generated.
Recently Closed	This contains alerts which closed within the last 24 hours i.e. during the day for which summary was generated. Each alert's ID is also included. Since the alerts auto close, a given service could have gone down and created an alert, then become normal and alert auto closed. It could have done this multiple times in a day in which case recently closed will list each incident along with its unique alert id. However, this is not expected to happen often given that each service has to be up for a threshold time before its alert is closed. User can filter with Status = CLOSED to get more information on each incident.
Service	Service Status link for Admiral which is the service processing and generating the daily summary.
Summary ID	ID of the summary alert.

A sample JSON Kafka output is shown below:


```
{
  "severity": "LOW",
  "tenant_id": 0,
  "alert_time": 1595721914808,
  "alert_text": "Admiral Summary",
  "key_id": "ADMIRAL_SUMMARY_Jul-26-20-00-04",
  "alert_id": "/Alerts/5efcfdf5497d4f474f1707c2/DataSource{location_type='TETRATION',␣
↪location_name='platform', location_grain='MIN', root_scope_id=
↪'5efcfdf5497d4f474f1707c2'}/
↪e95da4521012a4789048f72a791fb58ab233bbff63e6cbc421525d4272d469aa",
  "root_scope_id": "5efcfdf5497d4f474f1707c2",
  "type": "PLATFORM",
  "event_time": 1595721856303,
  "alert_details": "{\"Desc\":\"Summary of alerts for Jul-26\",\"Recently Closed\":\
↪"None\",\"Open\":\" Service Rpminstall with Alert ID 5.\",\"Service\":\"Admiral\",\
↪"Summary ID\":\"ADMIRAL_SUMMARY_Jul-26-20-00-04\"}"
}
```


An example summary alert containing a service raising multiple alerts in a day is shown below:

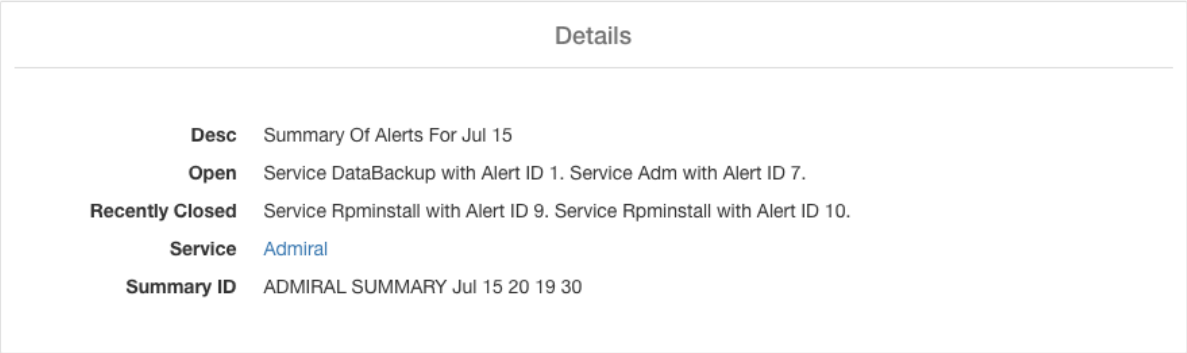


Fig. 11.2.4.3: Multiple Alerts

11.2.5 User Actions

Since admiral alerts generate an individual notification only once per alert, including/excluding or snoozing specific alerts are not needed. Alerts auto close when the service becomes normal for threshold uptime as described above. There is a force close option available to forcibly close an alert. This should normally be used only to remove summary alerts from UI as individual alerts auto close.

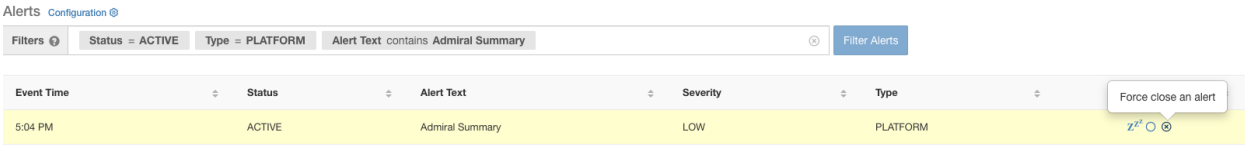


Fig. 11.2.5.1: Force Close Alert

> **Warning:** Individual Alerts should not be force closed. Doing so while the underlying service is still down or its uptime is below its expected threshold will lead to another alert getting raised for the same service on the next admiral processing iteration.

11.2.6 Admiral Notifications

Admiral Alerts are of Type PLATFORM. As such, these alerts can be configured to be sent to various publishers by appropriate connections for Platform Alerts via the configuration page ./configuration. For convenience, the connection is turned on between Platform Alerts and Internal Kafka by default which allows admiral alerts to be seen on the Current Alerts UI without any manual configuration.

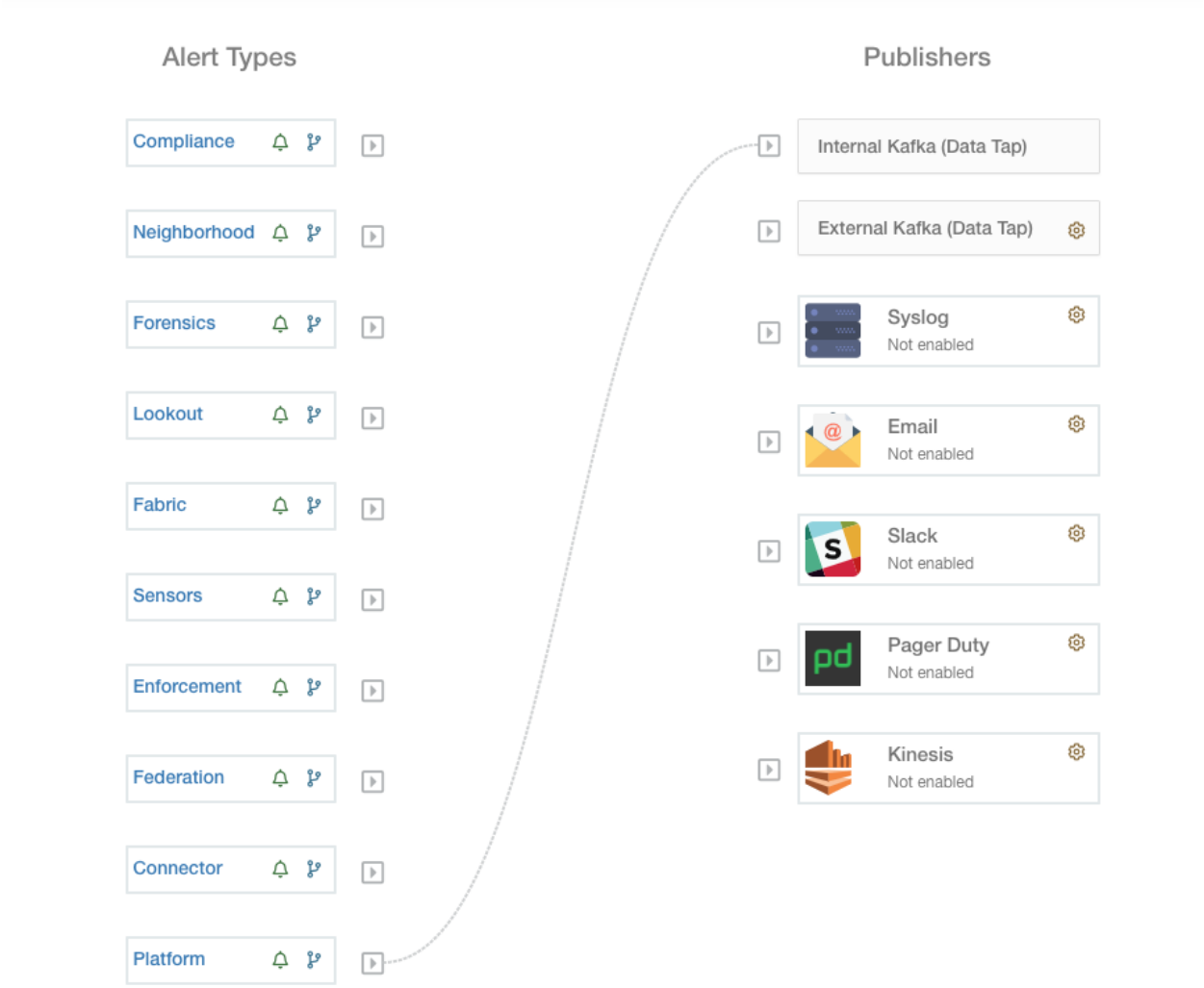
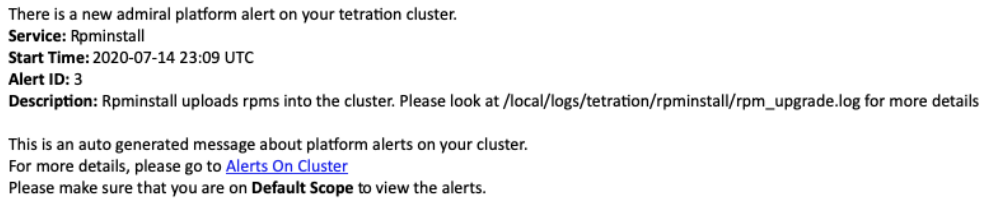


Fig. 11.2.6.1: Platform Alerts Configuration

Admiral Alerts are also sent to the email address configured under Company -> Cluster Configuration -> Admiral Alert Email.


There is a new admiral platform alert on your tetration cluster.
Service: Rpminstall
Start Time: 2020-07-14 23:09 UTC
Alert ID: 3
Description: Rpminstall uploads rpms into the cluster. Please look at /local/logs/tetration/rpminstall/rpm_upgrade.log for more details

This is an auto generated message about platform alerts on your cluster.
For more details, please go to Alerts On Cluster
Please make sure that you are on **Default Scope** to view the alerts.

Fig. 11.2.6.2: Sample Admiral Email

Thus, users can receive admiral notifications even if they don't have the TAN edge appliance setup. This is similar to Bosun behavior in previous releases.

cluster_state	Enabled till 2020-10-11 19:15:49 UTC
Cluster UUID	8194c5ef-65df-8aa1-5963-d10514761b6f
Admiral Alert Email	admiral@test.com

Fig. 11.2.6.3: Admiral Email

These email notifications are generated based on the same triggers as Current Alerts UI. Thus, they are sent on alert creation and a daily summary email at midnight UTC. The daily summary email lists all active alerts and those closed within the last 24 hours.

Daily summary of admiral platform alerts:

State:Active

Service: DataBackup
Start Time: 2020-07-14 21:58 UTC
Alert ID: 1
Description: The last successful checkpoint was over 48 hours ago.

State:Closed

Service: Rpminstall
Start Time: 2020-07-14 22:41 UTC
Alert ID: 2
Description: Rpminstall uploads rpms into the cluster. Please look at /local/logs/tetration/rpminstall/rpm_upgrade.log for more details

This is an auto generated message about platform alerts on your cluster.
For more details, please go to Alerts On Cluster
Please make sure that you are on **Default Scope** to view the alerts.

Fig. 11.2.6.4: Sample Summary Admiral Email

If there are no active alerts and no alerts closed within the last 24 hours, the summary emails are skipped to reduce email noise.

11.3 Cluster Status

The **Cluster Status** page can be accessed by Site Admin users but the actions can be carried out by **Customer Support** users only. It shows the status of all the physical servers in Cisco Tetration rack. Each row in UI represents a physical node with details such as it's hardware and firmware configuration and CIMC IP address (if assigned). The detail view of the node can be viewed by clicking on the row. In this page, we can also change CIMC password of the nodes and enable/disable external access to them. Orchestrator state is also displayed on the cluster status page to provide context for customer support.

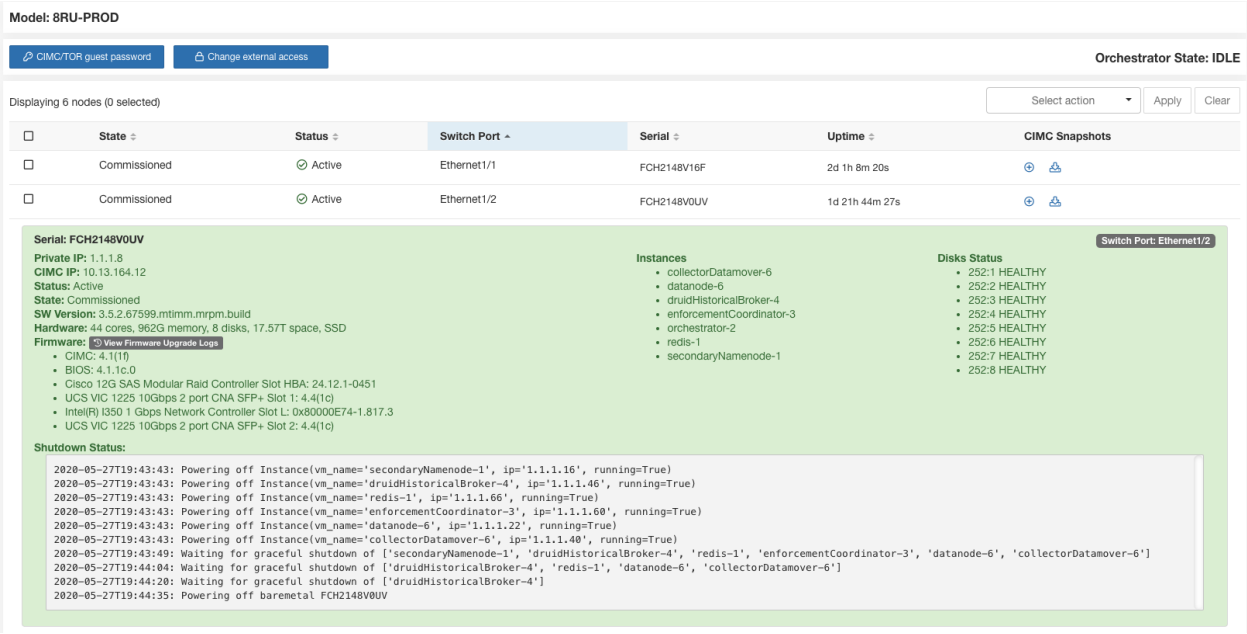


Fig. 11.3.1: Cluster Status

Actions that affect all nodes

Changing CIMC password and enabling/disabling external CIMC access can be done using the "CIMC/TOR guest password" and "Change external access" buttons and these actions affect all nodes in the cluster.

External CIMC access details

Clicking on the "Change external access" button will open a pop-up that provides status of external CIMC access and allows external access to CIMC to be enabled, renewed or disabled.

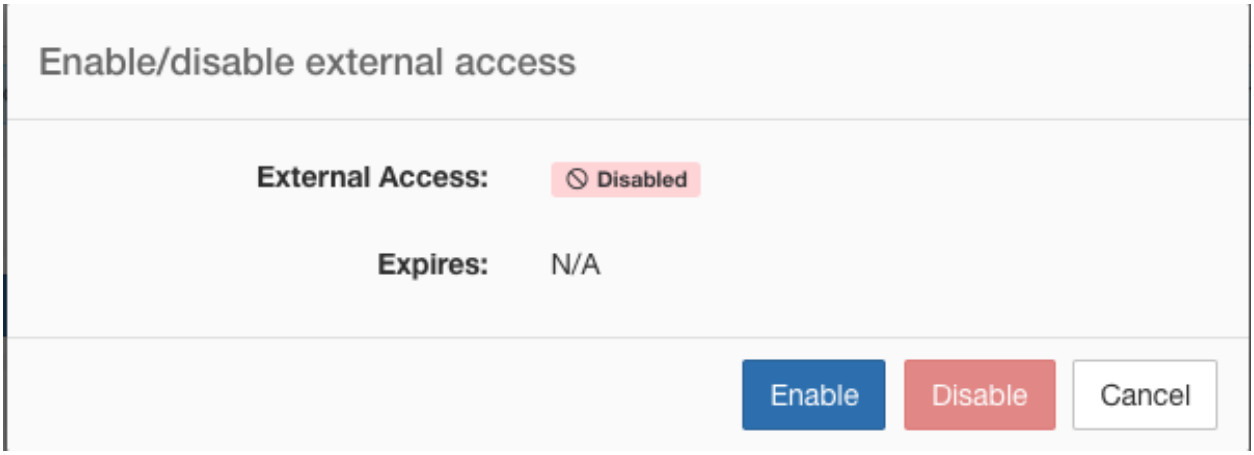


Fig. 11.3.2: External CIMC Access Disabled

Clicking on the "Enable" button will configure the cluster in the background to to enable external CIMC access, it can take up to 60 seconds for those tasks to complete and external CIMC access to be fully enabled. When external CIMC access is enabled the pop-up will show when access is set to automatically expire and the "Enable" button changes to "Renew" to reflect that you can renew external CIMC access. Renewing external CIMC access moves the expire time to be two hours from the current time.

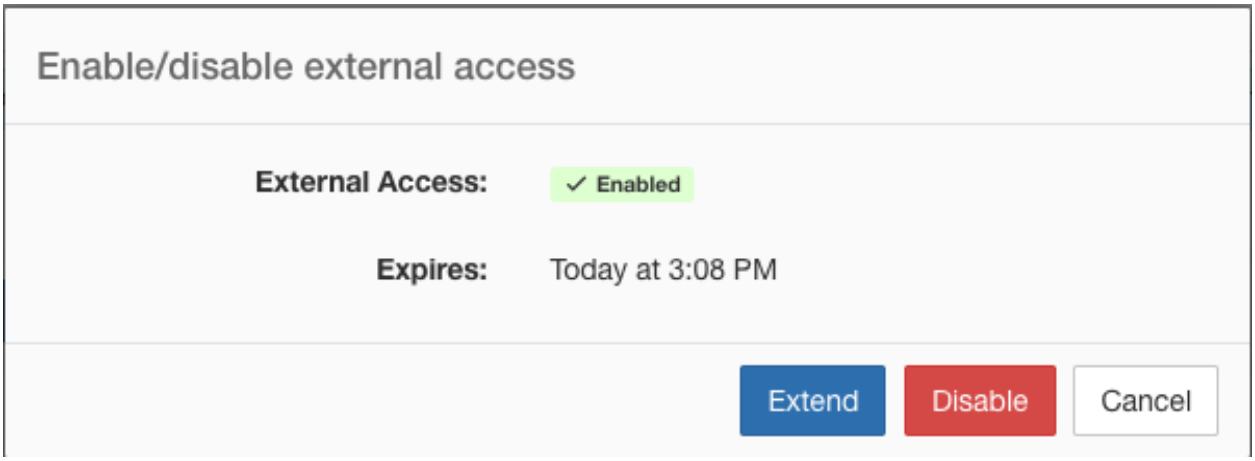


Fig. 11.3.3: External CIMC Access Enabled

If external CIMC access is enabled, the CIMC IP address in the node details (viewable by clicking on a row for a node) will become a clickable link that allows you directly access the CIMC WebUI - you may need to reload the cluster status page for the links to become visible.

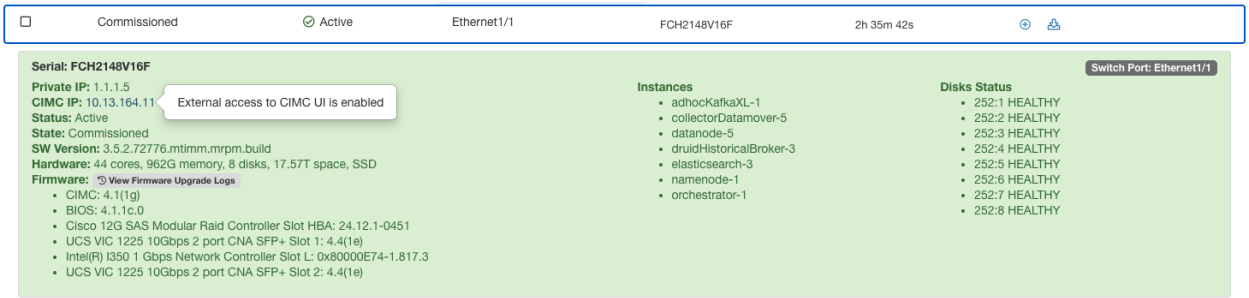


Fig. 11.3.4: External CIMC Access Node Details

The CIMC WebUI usually has a self signed certificate, accessing the CIMC WebUI will likely result in an error in the browser indicating that the certificate is not valid. If you are using Google Chrome this may require you to type "thisisunsafe" without quotes when the invalid certificate error is shown in Google Chrome to bypass the certificate check and access the CIMC WebUI.

Within the CIMC WebUI, KVM access is only functional if the CIMC version is 4.1(1g) or later. Once external CIMC access is enabled, it will be automatically disabled in 2 hours time unless access is renewed or disabled.

Disabling external CIMC access will configure the cluster in the background to disable external CIMC access, it can take up to 60 seconds for those tasks to complete and external CIMC access to be fully disabled.

Physical Node Details

Field	Description
Status	The **Status** field indicates the power status of the node. Possible values are: - **Active**-The node is powered on. - **Inactive**-The node is not powered on/connected.
State	The **State** field indicates the cluster membership state for the node. Possible values are: - **New**-The node is not part of the cluster yet. - **Initialized**-The node is part of the cluster. However, Cisco Tetration Analytics software is not fully installed on the node yet. - **Commissioned**-The node is up and running with Cisco Tetration software. The SW version field is also indicated and it turns red if an individual node does not have the same version as that of the whole cluster. - **Decommissioned**-The node has been removed from the cluster (for RMA purposes). The node should be replaced with new hardware. A node can be decommissioned via decommission action, refer actions below.
Switch Port	It refers to the switch port of the two switches on which the physical node is connected.
Uptime	It indicates the time for which the node has been running without a restart or shutdown.
CIMC Snapshots	Can be used to initiate a CIMC Tech Support collection and download a CIMC Tech Support.

Actions

Action	Description
Commission	Select this action to integrate new nodes into the cluster. Only nodes with state **New** are selectable for this action.
Decommission	Select this action to remove nodes that are part of the cluster currently. Only the nodes with state **Commissioned** or **Initialized** are selectable for this action.
Reimage	Select this action to reinstall the tetration software within the box. This could erase all contents of the box and is especially useful to upgrade the bare metal operating system from an older version to a new one. This step is required once a bare metal is decomissioned.
Firmware upgrade	Firmware information is available for the nodes where CIMC IP is reachable. This action is helpful to upgrade firmware on the nodes with older versions.
Power off	Select this action to power down the nodes. Please note that Nodes with status **Inactive** and **Shutdown in progress** cannot be powered down.

11.3.1 Firmware upgrade details

The Cisco Tetration physical appliance bundles a Unified Computing System (UCS) Cisco Integrated Management Controller (CIMC) Host Upgrade Utility (HUU) ISO. The firmware upgrade option on the Cluster Status page can be used to update a physical bare metal to the version of UCS firmware included in the HUU ISO that has been bundled in the Tetration RPMs.

A bare metal host can have the firmware update started on it when the status is *Active* or *Inactive* as long as the bare metal state is not *Initialized* or *SKU Mismatch*. Only one bare metal can have its UCS firmware updated at a time. In order to start the firmware update, the Tetration orchestrator state must be *Idle*. When the UCS firmware update is initiated, some of the UI functionality specific to the Cluster Status page may be temporarily impacted if the consul leader, active orchestrator or active firmware manager (fwmgr) need to be switched to other hosts - these switch overs should occur automatically. During the firmware update, the firmware details for the bare metal being updated will not be displayed and after the update it may take up to 15 minutes for the firmware details to display again in the Cluster Status page. Prior to starting the firmware update please check the Service Status page to verify all services are healthy.

When you initiate a firmware update on a bare metal, fwmgr will verify the update can continue, gracefully power down the bare metal if needed, then login to the CIMC on the bare metal and start the HUU based firmware update. That HUU based firmware update process involves booting the bare metal into the HUU iso, doing the update, rebooting CIMC to activate the new firmware then booting the bare metal back into the HUU iso to verify the update was completed. The overall update process can take 2+ hours for a G1 bare metal or 1+ hours for a G2 bare metal. Once the firmware update process is initiated, the Service Status page may indicate some services are unhealthy since a bare metal and all the virtual machines running on that bare metal are no longer active in the cluster. Once the firmware update completes, it can take an additional 30 minutes for the bare metal to become active in the cluster again and additional time may be needed for all services to become healthy again. If services do not recover within 2 hours after a firmware update please contact Cisco Technical Support for assistance.

You can click on a bare metal node in the Cluster Status page to expand details about the bare metal. Once a firmware update is initiated, you can click the *View Firmware Upgrade Logs* button to view the status of the firmware update. This log will contain the overall status of the firmware update at the very top and will be one of the following:

- *Firmware update has been triggered*: The firmware update was requested but has not started yet. During this status fwmgr will be checking to make sure the services required for the firmware update are functional and that CIMC can reach those services.

- *Firmware update is running*: The firmware update has been started. When a firmware update reaches this state, CIMC and HUU are in control of the update and the Tetration cluster will report the status it gets from CIMC about the update.
- *Firmware update has timed out*: This indicates that some process from the firmware update has exceeded the time we expect it to complete in. The overall firmware update process has a 240 minute time limit once it enters the *Firmware update is running* phase. During the firmware update CIMC may become unreachable when it reboots into the new version, this unreachable state has a timeout of 40 minutes before the firmware update is declared as timed out. Once the firmware update has started, the monitoring of that update will timeout after 120 minutes.
- *Firmware update has failed with an error*: This indicates that an error occurred and the firmware update has failed. CIMC usually does not give an indication of success or failure so this state usually indicates an error occurred prior to the firmware update actually running.
- *Firmware update has finished*: The firmware update finished without running into any errors or time outs. CIMC usually does not give an indication of success or failure, it is best to verify that the UCS firmware versions are updated once those details become available in the Cluster Status page - it can take up to 15 minutes for those details to become available.

Below the overall status in the *View Firmware Upgrade Logs* pop-up is an *Update progress* section that will contain timestamped log messages indicating the progress of the firmware update. Once the *Rebooting Host In Progress* status is displayed in these log messages, CIMC is in control of the update and the cluster is monitoring that update - most log messages after this come directly from CIMC and are only added to the list of log messages if the status of the update changes.

Below the *Update progress* section of the *View Firmware Upgrade Logs* pop-up a *Component update status* section will be shown once CIMC starts providing individual component update statuses. This section can give a quick overview of the status of the update of the various UCS components on the bare metal.

11.4 Data Backup And Restore (DBR)

Data backup and restore copies certain data from Tetration cluster to an off-site storage. In the event of a disaster, data can be restored from this off-site storage to any cluster of same form-factor.

1. Data backup and restore is supported only for *physical clusters* (both 8RU and 39RU) and is **NOT** supported on virtual appliances.
2. Data can be backed up to any external object store compatible with S3V4 API. While any object store can be used, Tetration does require sufficient bandwidth and storage to back up data.
3. At least 200TB of storage is recommended for backup. Lack of space will cause backup failures.
4. Data can only be restored to a cluster of compatible form-factor, e.g. data from a 8RU cluster can be restored only to another 8RU.

11.4.1 Backup

Backup is triggered once a day at the scheduled time, based on user configuration. A successful backup is called a *checkpoint*. Checkpoint is a point in time snapshot of the cluster's primary data-stores (HDFS, Druid, Mongo, Consul and Vault). Note that not all data is backed up. Only what is necessary for restoring flow database, ADM and enforcement is backed up. A successful checkpoint can be used to restore the data onto another cluster or the same cluster.

Data in Mongo, Consul and Vault is always fully backed up for every checkpoint. HDFS and Druid contribute to the bulk of the data backed up and hence only the incremental changes are backed up. Optionally, full backup can be triggered on a schedule or on-demand for all data sources. A full backup copies every object in a checkpoint even if it is already copied and the object has not changed. This can add significant load on the cluster, on the network between the cluster and the object store and the object store itself. It is recommended not to enable full backup on a schedule and use on-demand workflow when needed. A full backup might be necessary if there are any corruption in the objects or object store has any unrecoverable hardware failures. Additionally if the bucket provided for backup changes, an automatic full backup will be forced.

11.4.2 Pre-Requisites

1. DBR is a licensed feature. Please obtain the right license for DBR by following the instructions in the licensing page of the cluster.
2. Access Key and Secret Key for the object store. DBR does not work with pre-authenticated link for object store.
3. Configure any policing to throttle the bandwidth used by the Tetration appliance to object store.
4. Configure the FQDNs and make sure sensor hosts can resolve the FQDNs.

Note that once DBR is enabled, only current and future software agent versions would be available for installation and upgrades. The software agent versions that are older than the current cluster version will be hidden due to incompatability.

11.4.2.1 Sensor/Kafka FQDNs Requirements

Sensors use an IP address to get control information from Tetration appliance. To enable DBR and allow for seamless fail-over after a disaster, sensors need to switch to using FQDN. Upgrading Tetration cluster is not sufficient for this switch. Sensors support using FQDN starting release 3.3 and above. So to enable sensor fail-over and make them DBR ready, ensure sensor is upgraded to release 3.3.

If not configured, the default FQDNs are:

IP Type	Default FQDN
Sensor VIP	wss{{cluster_ui_fqdn}}
Kafka 1	kafka-1-{{cluster_ui_fqdn}}
Kafka 2	kafka-2-{{cluster_ui_fqdn}}
Kafka 3	kafka-3-{{cluster_ui_fqdn}}

The FQDNs can be changed in Company -> Cluster Configuration page.

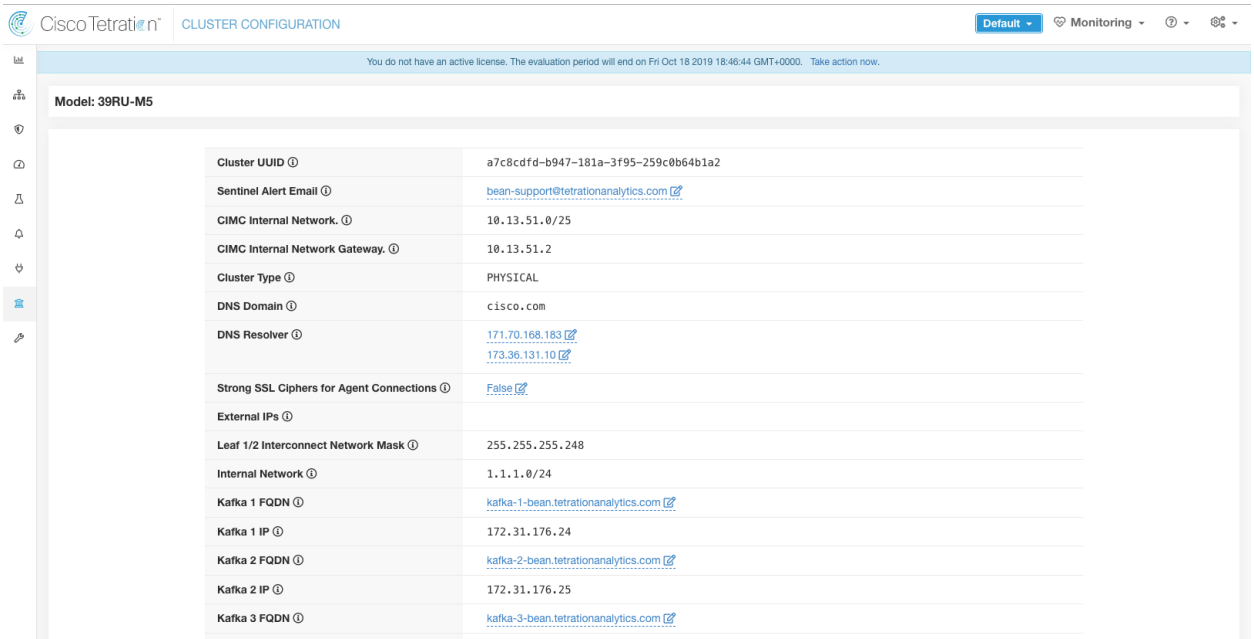


Fig. 11.4.2.1.1: FQDNs/IP for DBR in Cluster Configuration Page

Update the DNS record for these FQDN with the IPs provided in the same page. Here is the mapping of IP and FQDN.

Field name	Corresponding IP Field	Description
Sensor VIP FQDN	Sensor VIP	Update the FQDN to connect to cluster control plane
Kafka 1 FQDN	Kafka 1 IP	Adhoc Kafka node 1 IP
Kafka 2 FQDN	Kafka 2 IP	Adhoc Kafka node 2 IP
Kafka 3 FQDN	Kafka 3 IP	Adhoc Kafka node 3 IP

NOTE: FQDN for sensors VIP and kafka hosts can only be changed before DBR is configured. Once DBR is configured, FQDN cannot be changed.

11.4.3 Object Store Requirements

The object store should provide a S3V4 complaint interface.

Bucket

Create a new and dedicated bucket for Tetration in the object store. Only Tetration cluster should have write access to this bucket. Tetration cluster will write objects and manage retention on the bucket. Provision at least 200TB of storage for the bucket and obtain an access and secret key for the bucket. Tetration would NOT work with pre-authenticated links.

If using Cohesity as object store, disable multi-part uploads while scheduling.

HTTPS

Tetration data backup supports only https interface with the object store. This is to ensure that data in transit to the object store is encrypted and secure. If the storage SSL/TSL certificate is signed by trusted third party CA, the cluster will use that to authenticate the object store. In case the object store uses self-signed certificate, the public key or the CA can be uploaded by selecting the *Use Server CA Certificate* option.

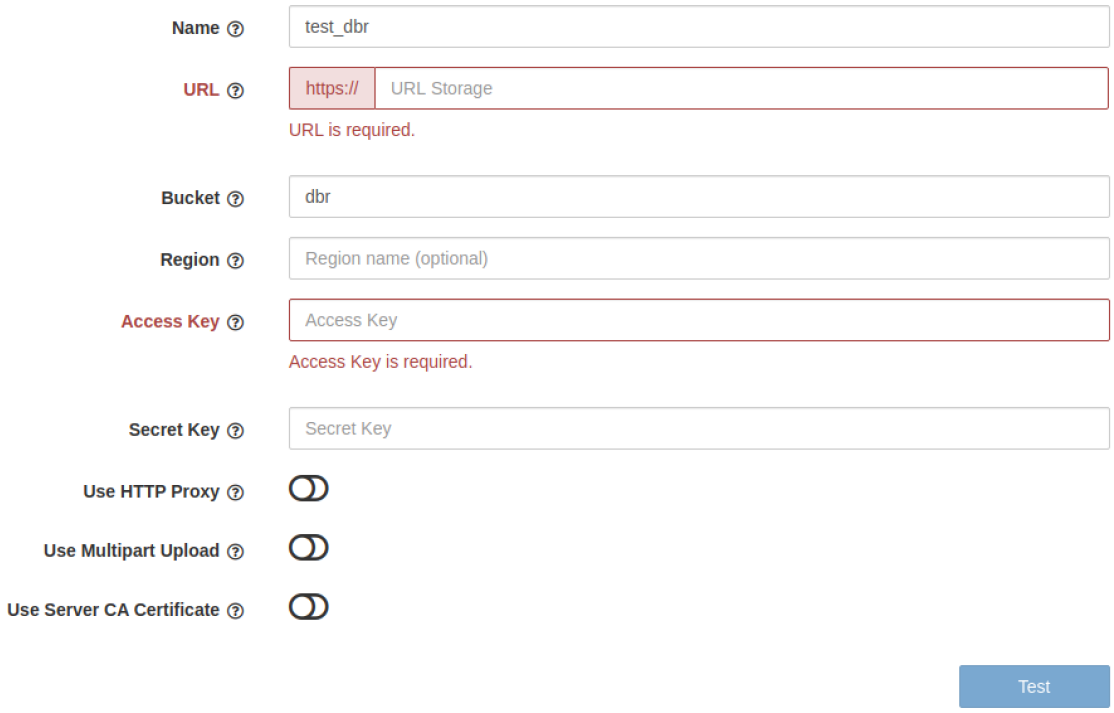


Fig. 11.4.3.1: Server CA Certificate option to provide self signed certificates.

Server Side Encryption

It is also strongly recommended to turn ON server-side encryption for the bucket provided to Tetration. Tetration cluster will use HTTPS to transfer data to object store. However the object store should encrypt the objects to ensure the data at rest is secure.

11.4.4 Configuration

Step 1 - Planning

Backup provides a planner to test the access to the object store, determine the storage requirement and the backup duration needed for each day. This can be used to experiment before actually configuring schedule.

To use DBR calculators, navigate to *Maintenance -> Data Backup*. If DBR is not configured, this will navigate to the Data Backup landing page.

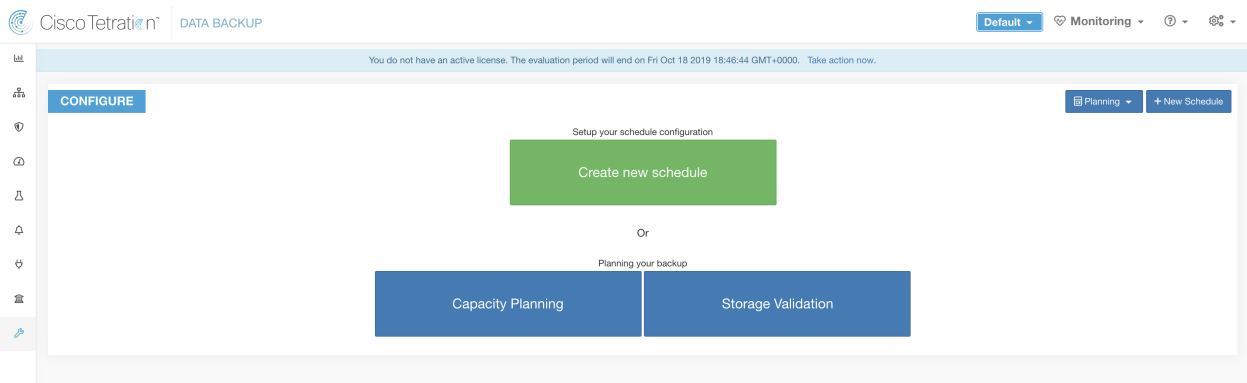


Fig. 11.4.4.1: Backup Landing Page

Note: If there is no Data Backup option under Maintanence, ensure you have the license to enable DBR

To ensure the storage is compatible with Tetration, use the "Storage Planning" option. Click on Storage Planning, to enter the storage configuration. The validation will test:

- Access/authenticate the object store and bucket.
- Upload to and download from the configured bucket.
- Bandwidth checks.

This can take around 5 minutes to complete.

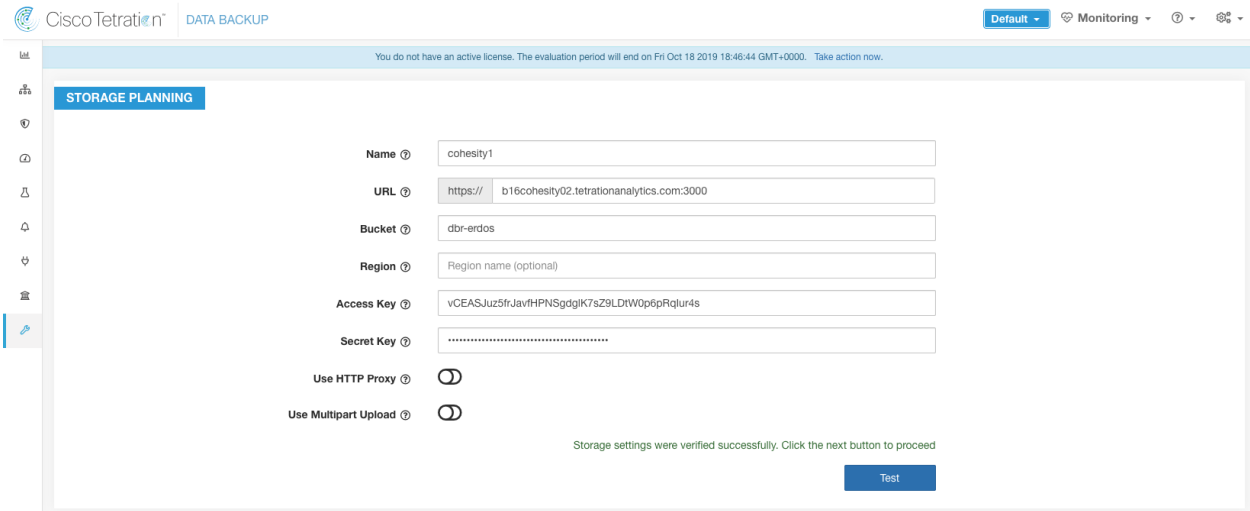


Fig. 11.4.4.2: Backup Storage Planning Page

When the test completes there will be a status message. If the test fails, ensure:

1. URL is correct.
2. Access/secret key is correct.
3. Bucket exists.
4. Configure proxy if storage needs to be accessed directly.
5. If using Cohesity, disable Multi-part upload.

Capacity Planner can be used to plan storage size and backup window estimates

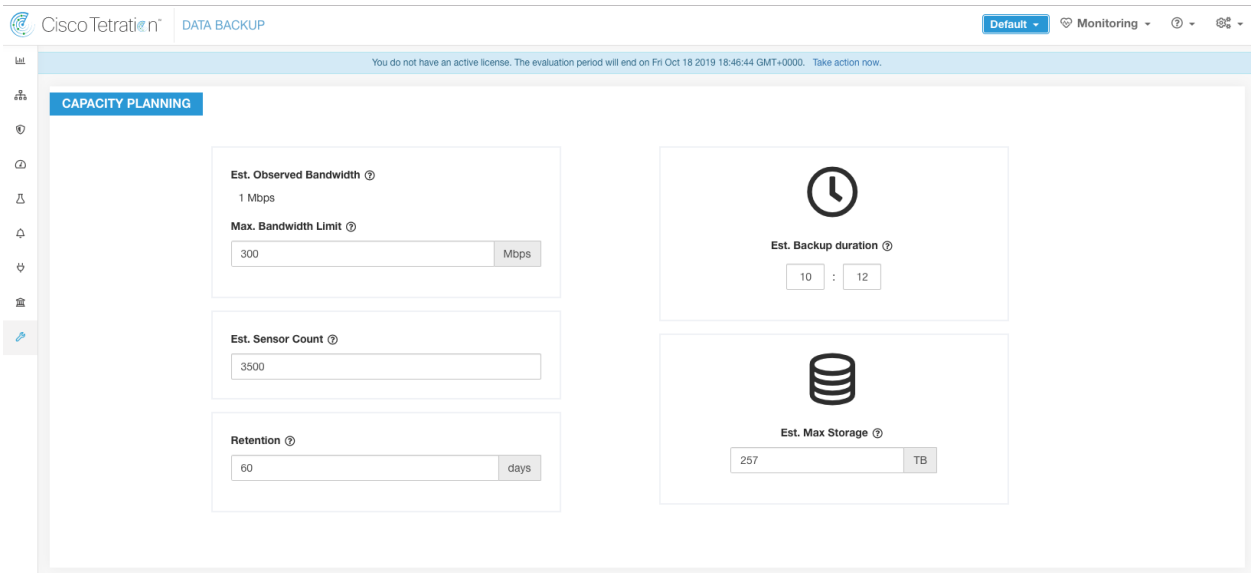


Fig. 11.4.4.3: Backup Capacity Planning

- Max Bandwidth Limit: Maximum bandwidth allowed to use while backing up data. This bandwidth must at most be the policer configuration that will throttle data to the object store.
- Est. Sensor Count: This defaults to existing registered sensors, but can changed based on forecasts.
- Retention: Expected days of retention in the object store.
- Est. Backup Duration: Time required to backup one day's data. This is an estimate based on typical sensor load, est. sensor count and maximum bandwidth configured above.
- Est. Max Storage: This is the estimate of maximum storage required by Tetration to support specified retention and est. sensor count.

Step 2 - Configure

Tetration will copy data to object store only in the configured time-window. Backup Configuration Wizard goes through the storage/window configuration steps, similar to the Planner.

To configure backup, click on the "Create new schedule" in the data backup landing page. While configuring backup fir the first time, the pre-checks will run to ensure the FQDNs are resolvable and resolves to the right IP. Once that's validated, an update is pushed to all sensors currently registered with the cluster to switch to using FQDNs. Without FQDN, the sensors cannot fail-over to another cluster after a disaster event. To support this sensors must be upgraded to the latest version supported by the cluster and all the sensors should be able to resolve the sensor VIP FQDN. As of release 3.3 only deep visibility and enforcement sensors support DBR and will switch to using FQDN. Rest of the sensors will continue to use IP.

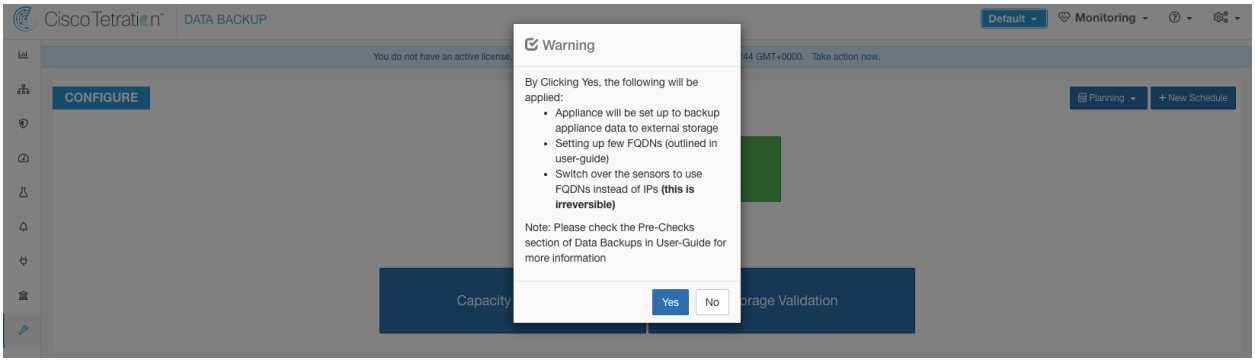


Fig. 11.4.4.4: Backup Warning - Ensure FQDNs are set.

Click Yes on the warning box to proceed with running pre-reqs. If there are any failures in pre-reqs checks, the status will show as failed with a detailed log:

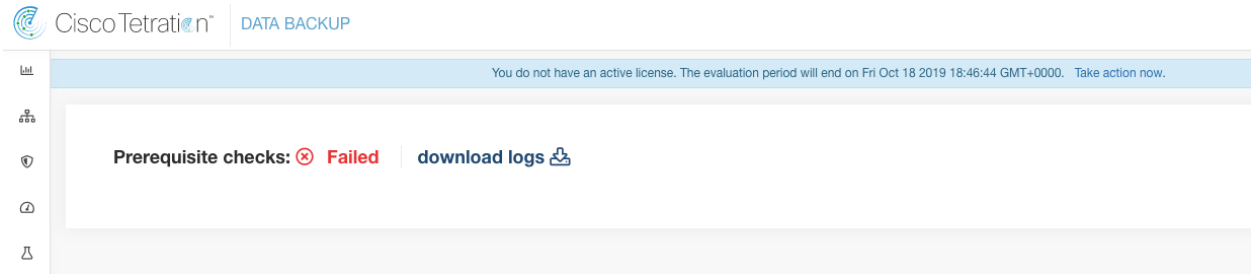


Fig. 11.4.4.5: Failed Pre-Requirements

When all the pre-requirement checks pass, proceed to entering the storage information:

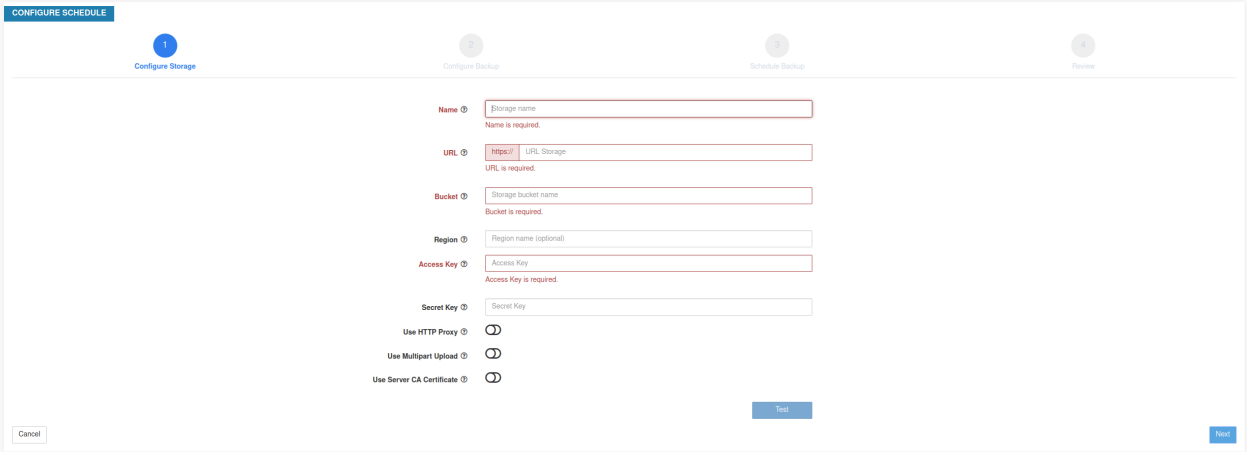


Fig. 11.4.4.6: Storage Configuration

When the storage is validated, click next to the planning capacity:

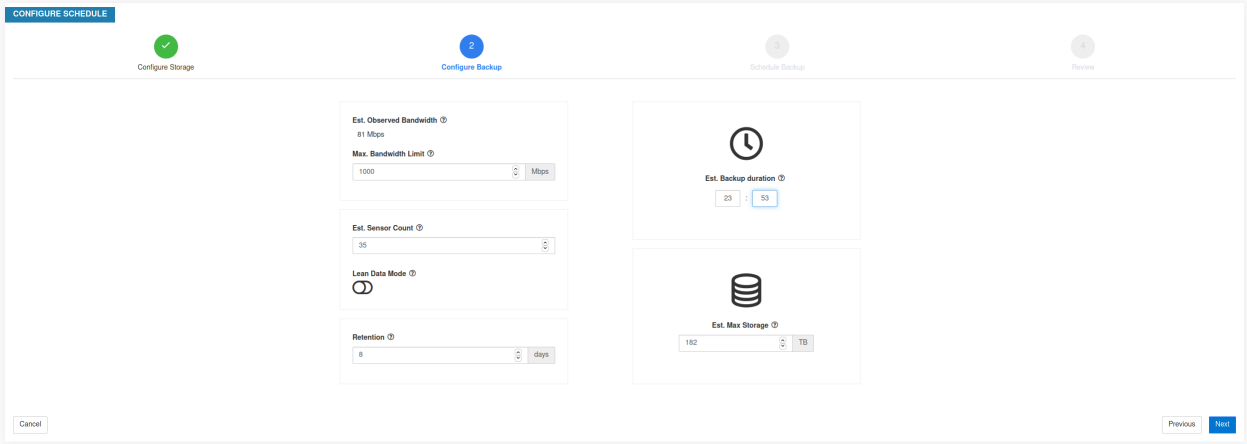


Fig. 11.4.4.7: Capacity Planning

These two steps are exactly same as in the Planning phase. Flow data is not backed up, if *lean data mode* is selected. This may be useful if the backup storage is limited. Click next to navigate to configure schedule.

- Set starting backup point from today: (default selected) - this option will ignore all files created before midnight UTC on the day of configuration. In a cluster that's been running for a while, there could be a lot of data to backup on the first day and might overwhelm the cluster, network and the object store. All configuration will be still be backed up irrespective of this option.
- Timezone - defaults to browser timezone.
- Allowed Start backup window - Time in hour/minute when backup will start (in 24 hour format).
- Enable recurring full backup (default unselected) - Selecting this will give an option to select a schedule for full backups. Recommended to not use full backup as a schedule.
- Continuous backup - When this option is selected, a backup is taken as frequently as possible.

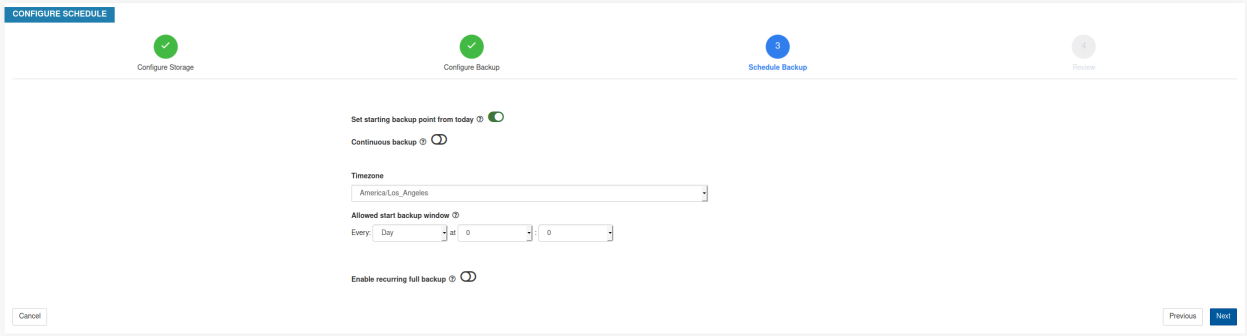


Fig. 11.4.4.8: Backup Scheduling

The final step is to review and initiate the backup job.

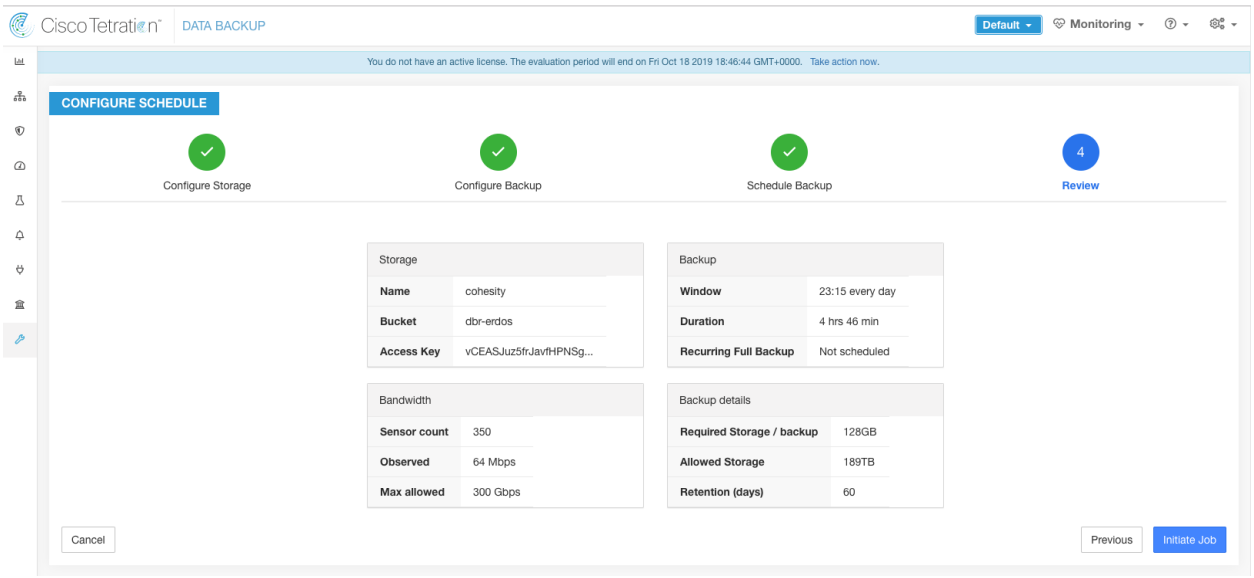


Fig. 11.4.4.9: Backup Configuration Review

11.4.5 Backup Status

After configuration, backup will be triggered everyday at the scheduled time. Status of the backups can be seen in the Data Backup dashboard (Maintenance -> Data backup).

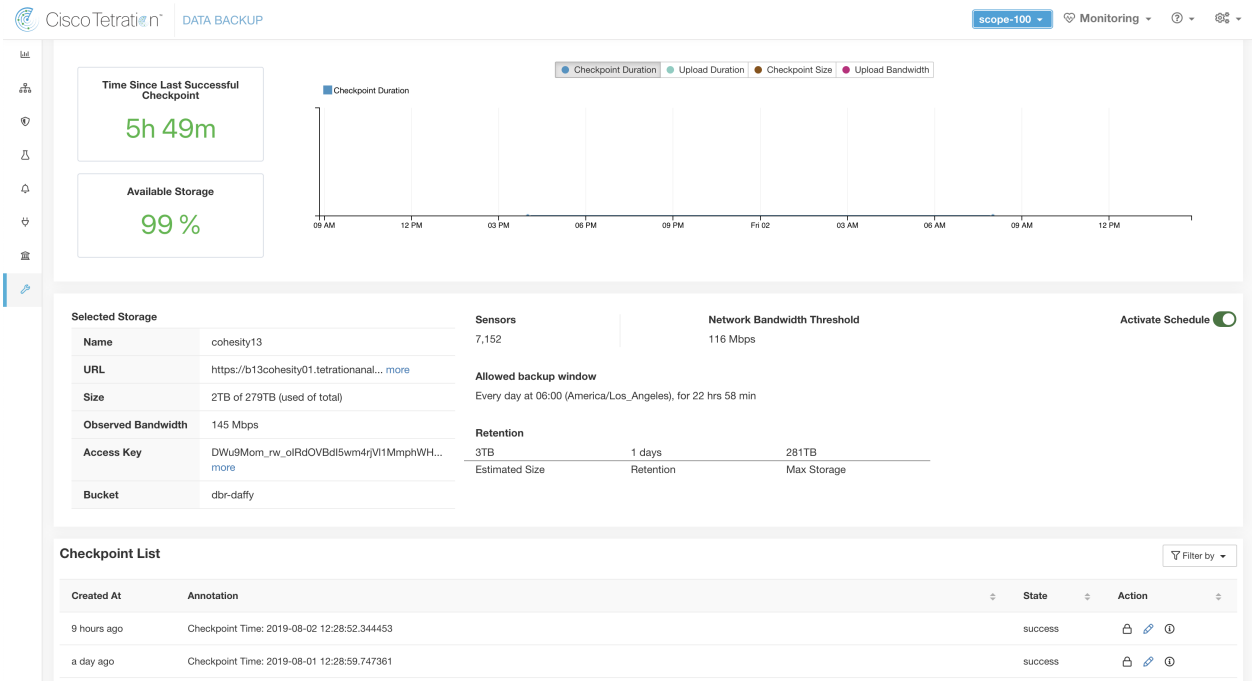


Fig. 11.4.5.1: Backup/Checkpoint Status

Time since last successful checkpoint should be less than 24 hours + the time it takes to checkpoint. If the checkpoint + backup takes around 6 hours, then the time since last successful checkpoint should be less than 30 hours.

There are few other graphs in the dashboard about the checkpoints and backup.

The table shows all the checkpoints. Checkpoint labels can be edited and the label will be available while choosing a checkpoint during restores. Label can be edited by clicking on the edit option under *Action* for a checkpoint.

A checkpoint goes through multiple phases and these are the possible states:

- created/pending : Checkpoint is just created and waiting to be copied.
- running : Data is getting actively backed up to external storage.
- success : Checkpoint is complete and is successful, can be used for restores.
- failed : Checkpoint is complete and is failed, cannot be used for restores.
- deleting/deleted : An aged-out checkpoint is going through deletion.

To change the schedule or the bucket, click on "New/Edit Schedule". This will guide you through the same wizard used to setup backup.

11.4.5.1 Deactivating Schedule

Backups can be deactivated by disabling the "Activate Schedule" button. It is recommended to deactivate the backup schedule before making changes to the schedule. Please deactivate a schedule only when no checkpoint is in progress. Running a test, or disabling the schedule while a checkpoint is in progress may cause the checkpoint in progress to fail.

11.4.6 Object Store Retention

Tetration cluster manages the life-cycle of objects in the bucket. User should not delete or add objects to the bucket. Doing so might lead to inconsistencies and corrupt successful checkpoints. In the configuration wizard the max storage to use is specified. Tetration will ensure its usage of bucket will stay within this limit. There is a storage retention service that ages out objects and deletes them from the bucket. As soon as storage usage reaches a threshold, computed based on the configured max storage and incoming data rate, the retention will try to delete *un-preserved* checkpoints to reduce the usage to T1. The retention will also keep a minimum of 2 successful checkpoints at any time and all the preserved checkpoints (whichever is more). If retention cannot delete any checkpoints to make space, checkpoints will start failing.

11.4.7 Preserving checkpoints

As new checkpoints get created, old ones will age-out and deleted. However, checkpoints can be preserved, preventing it from being deleted by retention. A preserved checkpoint will not be deleted. If there are multiple preserved checkpoints, at some point there wouldn't storage for new objects and aged-out checkpoints cannot be deleted because they were preserved. As a best practice, use preserved on a need basis and update the Label for the checkpoint with the reason and validity as a reference. To preserve a checkpoint, click on the lock icon on the right.

6 days ago	Checkpoint Time: 2019-07-28 05:22:14.801987	success	
7 days ago	Checkpoint Time: 2019-07-27 05:22:57.127565	success	
8 days ago	Checkpoint Time: 2019-07-26 05:13:28.891415	success	
9 days ago	Checkpoint Time: 2019-07-25 05:15:07.606218	success	
10 days ago	Checkpoint Time: 2019-07-24 05:23:32.655903	success	

Fig. 11.4.7.1: Preserving Checkpoints

11.4.8 Restores

A cluster has to be in the DBR standby mode to be restored using backed up data. Currently, a cluster can be set to standby mode only during deploy.

Following combinations are allowed:

Primary Cluster SKU	Standby Cluster SKU
8RU-PROD	8RU-PROD, 8RU-M5
8RU-M5	8RU-PROD, 8RU-M5
39RU-GEN1	39RU-GEN1, 39RU-M5
39RU-M5	39RU-GEN1, 39RU-M5
OCI	OCI

11.4.8.1 Standby Mode deployment

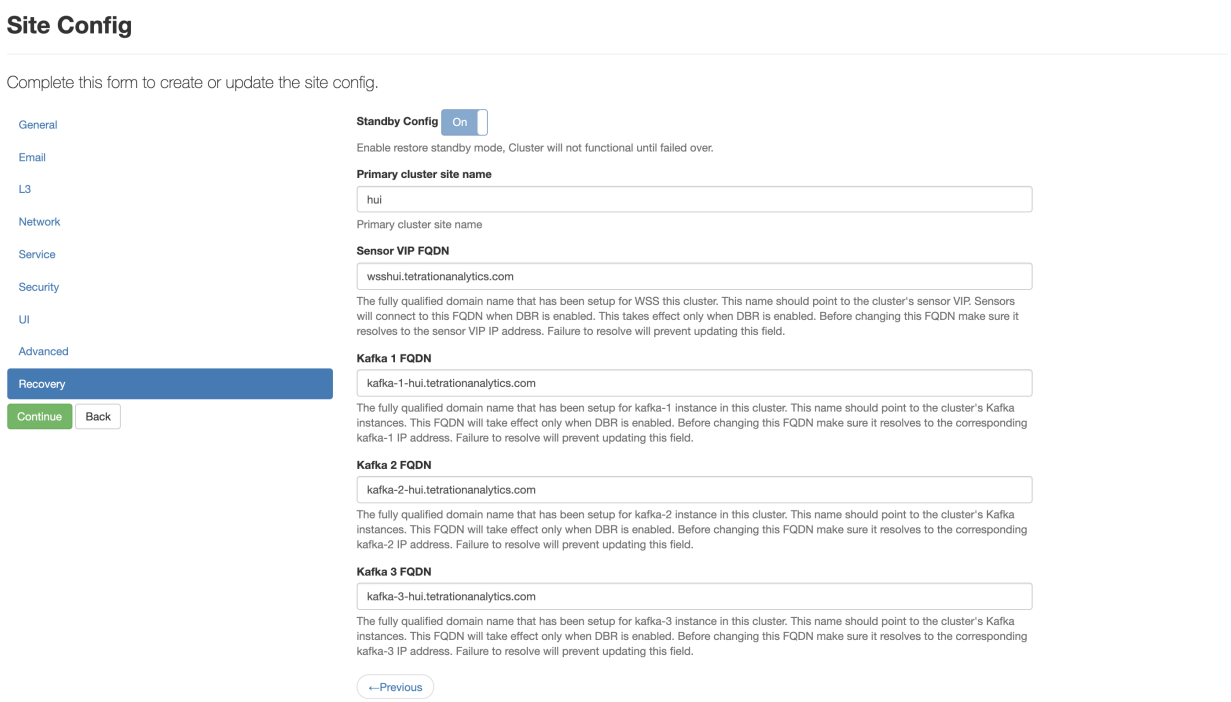
Contact Cisco to initiate data restore.

A cluster can be deployed in the Standby mode by configuring the recovery options in site information. While configuring site information during deployment, configure the restore details under the Recovery tab.

To deploy the cluster in standby mode, configure the following under the Recovery tab.

1. Set the *Standby Config* to *On*.
2. Configure Primary cluster name and FQDNs.

Rest of the deployment is exactly same as regular deployment.



Primary cluster name and FQDNs can be reconfigured after deployment to make the standby cluster track another cluster. This can be reconfigured at a later time before fail-over is triggered from the Cluster Configuration page.

A cluster in DBR standby mode will show the *Standby Mode* Banner.

Fig. 11.4.8.1.1: Standby Banner

Click on the *Data Restore* tab in the side menu to go to the DBR Restore page.

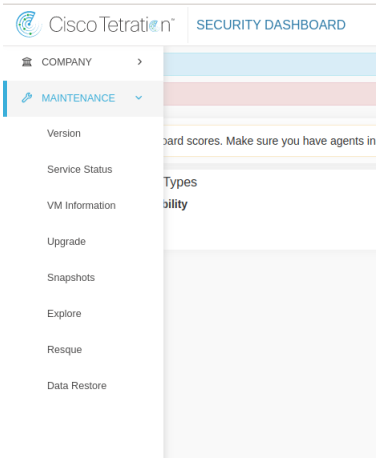


Fig. 11.4.8.1.2: Data Restore menu

11.4.8.2 Data Prefetch

Before the cluster can be restored, it must prefetch data. The data is prefetched from the same storage bucket that is used for backing up data. Credentials must be provided for the backup service to download from the storage. If a storage is never set up for prefetch, the data restore tab will take the user to the setup wizard directly.

Standby cluster interacts only with the S3 storage. When the backup on Primary cluster is updated to use a different storage/bucket, the storage on standby cluster must be updated.

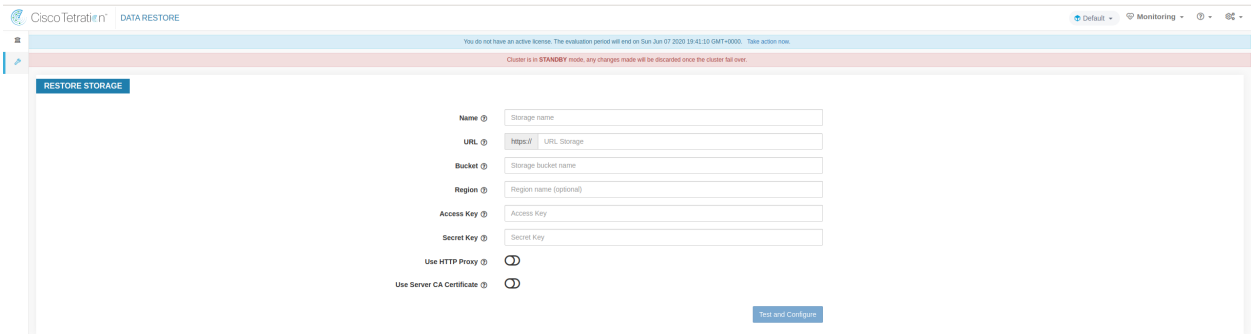


Fig. 11.4.8.2.1: Storage setup wizard

Once the information is tested, storage is auto configured for prefetch. The DBR restore tab should now show the prefetch status.

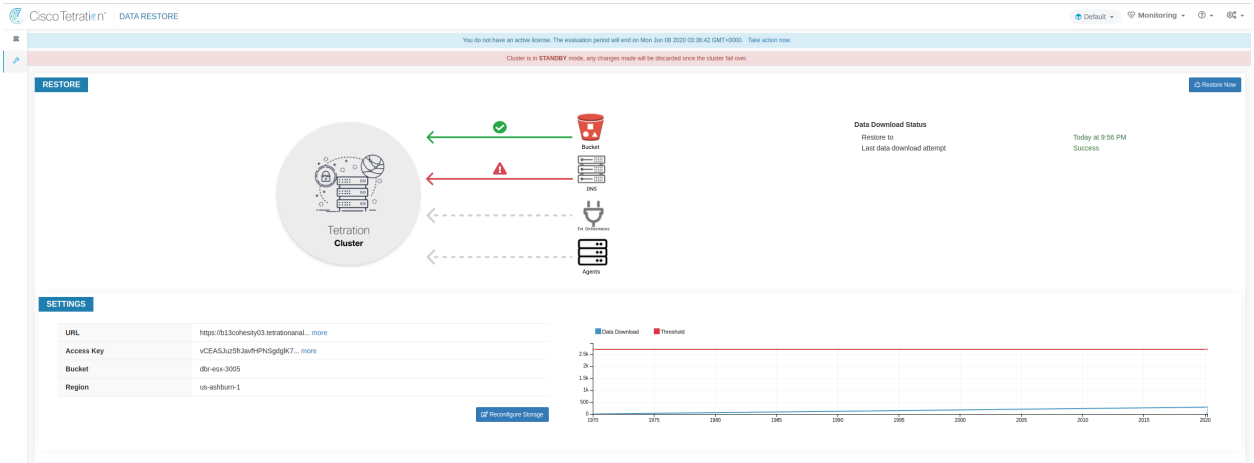


Fig. 11.4.8.2.2: DBR Prefetch Status

The status page provides the user with a variety of data.

1. The top left part has a graphic indicating readiness of various components for starting a restore. To check the data, please hover over the components. The associated data then shows up in the top right part.

   Bucket: It shows the prefetch status. If the latest data is more than 45 minutes old, it shows up in red.

   DNS: It shows the Kafka, and WSS FQDN resolutions with respect to standby cluster IPs. During restore, if the FQDNs are not updated to standby cluster IPs, the sensor will not be able to connect. Once the FQDNs start resolving to the standby cluster, this would turn green.

   Ext. Orchestrators: This shows the connectivity to external orchestrators from the standby cluster.

   Agents: This shows the number of agents that have successfully switched over to the standby cluster. This is only relevant after a restore has been triggered.

2. The top right part shows the information relevant to the chosen graphic in the top left part. In the top right corner, clicking on the *Restore Now* will initiate the restore process.

3. The bottom left part shows the prefetch storage settings in use.

4. The bottom right part shows a graph of prefetch delays.

A data prefetch updates several necessary components to ensure a fast restore. If a data prefetch fails, it will show the reason on the status page.

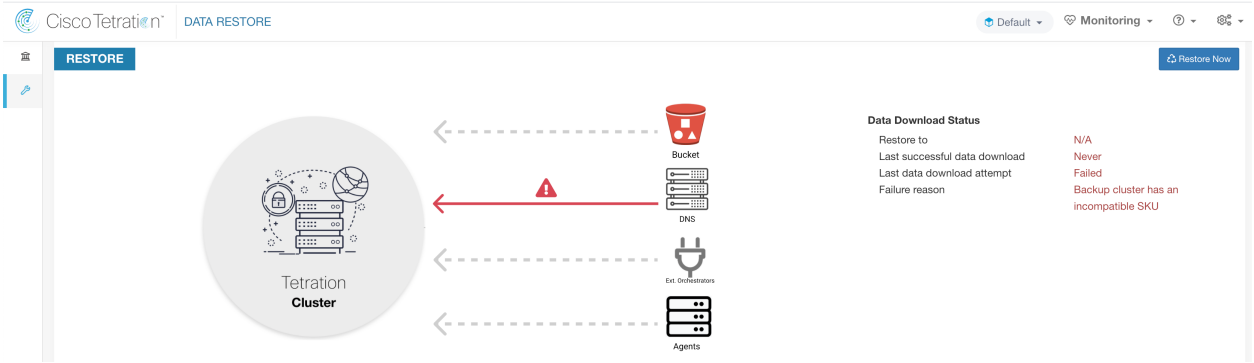


Fig. 11.4.8.2.3: DBR Prefetch Error Case

Here are some common errors that can cause prefetch failures.

S3 Access Error: In this case the data from the storage could not be successfully downloaded. This may happen due to invalid credentials, a change in the storage policies or temporary network issues.

Incompatible Cluster Versions: Restore can only be done to a cluster running the same Tetration version as the backup cluster. This can likely happen during upgrades, when only one of the clusters is deployed. Or, during deploy when a different version is used for deploying. Upgrading the clusters to a common version would resolve this issue.

Incompatible SKU Versions: Please take note of the allowed SKUs for standby clusters, given the primary cluster. Only specific SKUs are allowed for restore of the primary cluster SKU.

11.4.8.3 Cluster Restore

A cluster restore can be triggered by clicking on the *Restore Now* button in the top right corner of the restore status page. Before a restore action can be triggered, an acknowledgement is asked.

Cluster data is restored in 2 phases.

Mandatory Phase: The data needed to restart services is restored first. This data is already prefetched. The time taken by mandatory phase depends on the number of sensors installed, amount of data backed up, etc. During the mandatory phase, the UI is not accessible. **Working TA guest keys are required for any support during mandatory the phase, should such a need arise.**

Lazy Phase: Cluster data (like flow DB in druid) is restored in the background and will not block cluster deployment. The cluster UI is accessible, and will have a banner while restore is in progress. During this phase, the cluster is operational and data pipelines are functioning normally.

11.4.9 Upgrades (with DBR)

When **DBR** is enabled on the cluster, it is recommended to deactivate the schedule before starting the upgrade (See *Deactivating Schedule*). This will ensure that a successful backup exists before upgrade is started, and that no new backup is being uploaded. A schedule should only be deactivated when a checkpoint is not in progress, to avoid failed checkpoint.

11.5 VM Information

The **VM Information** page displays all virtual machines that are part of the Cisco Tetration cluster. It displays their deployment status during cluster bring up or upgrade(if any) and also public IPs. Note that all VMs in the cluster are not part of a public network therefore they may not have a public IP.

Filters | Enter attributes... | Filter

Displaying 106 of 106 virtual machines

Private IP	Instance Name	Public IP	Deploy Status	Uptime	Logs
1.1.1.99	adhoc-1		Deployed	1d 14h 17m 58s	View Logs
1.1.1.100	adhoc-2		Deployed	1d 14h 18m 52s	View Logs
1.1.1.144	adhocKafkaXL-3	172.31.239.154	Deployed	1d 14h 18m 2s	View Logs
1.1.1.143	adhocKafkaXL-2	172.31.239.153	Deployed	1d 14h 18m	View Logs
1.1.1.142	adhocKafkaXL-1	172.31.239.152	Deployed	1d 14h 18m 6s	View Logs
1.1.1.41	appServer-2	172.31.239.135	Deployed	1d 14h 19m 15s	View Logs
1.1.1.40	appServer-1	172.31.239.134	Deployed	1d 14h 19m 17s	View Logs
1.1.1.96	collectorDatamover-16	172.31.239.151	Deployed	1d 14h 18m 16s	View Logs
1.1.1.94	collectorDatamover-14	172.31.239.149	Deployed	1d 14h 18m 18s	View Logs
1.1.1.95	collectorDatamover-15	172.31.239.150	Deployed	1d 14h 18m 10s	View Logs
1.1.1.92	collectorDatamover-12	172.31.239.147	Deployed	1d 14h 18m 27s	View Logs
1.1.1.93	collectorDatamover-13	172.31.239.148	Deployed	1d 14h 18m 21s	View Logs
1.1.1.90	collectorDatamover-10	172.31.239.145	Deployed	1d 14h 18m 25s	View Logs
1.1.1.91	collectorDatamover-11	172.31.239.146	Deployed	1d 14h 18m 20s	View Logs
1.1.1.89	collectorDatamover-9	172.31.239.144	Deployed	1d 14h 18m 50s	View Logs
1.1.1.88	collectorDatamover-8	172.31.239.143	Deployed	1d 14h 18m 52s	View Logs
1.1.1.81	collectorDatamover-1	172.31.239.136	Deployed	1d 14h 19m 2s	View Logs
1.1.1.83	collectorDatamover-3	172.31.239.138	Deployed	1d 14h 19m	View Logs
1.1.1.82	collectorDatamover-2	172.31.239.137	Deployed	1d 14h 19m 6s	View Logs
1.1.1.85	collectorDatamover-5	172.31.239.140	Deployed	1d 14h 19m 43s	View Logs

Fig. 11.5.1: VM Information

11.6 Upgrading Cluster

There are two types of upgrade. This section describes the "full" upgrade process. During this upgrade all VMs in the cluster except for Orchestrator-VMs are shut down, new VMs are deployed, and the services are re-provisioned. All the data within the cluster are persisted during this upgrade. Except a downtime of around 2 hours during this upgrade.

11.6.1 Initiating Upgrade

Upgrade is initiated from the cluster page (main UI). Click Maintenance and on Upgrade as seen in

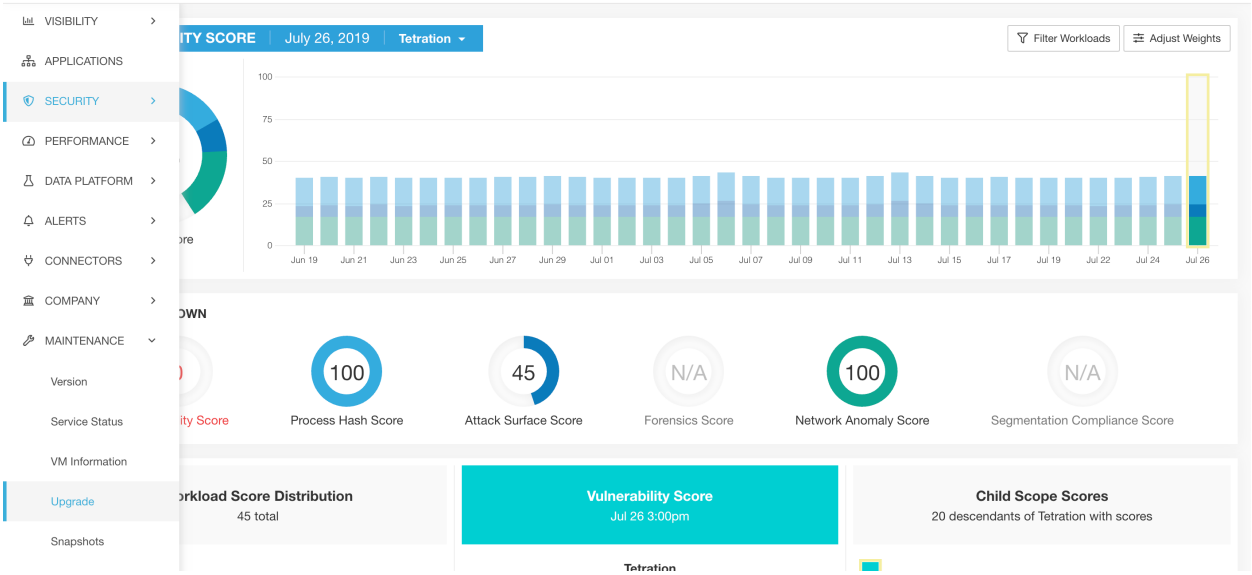


Fig. 11.6.1.1: Initiating Upgrade

In the upgrade page, you have option to either upgrade/patchupgrade/shutdown/reboot the cluster.

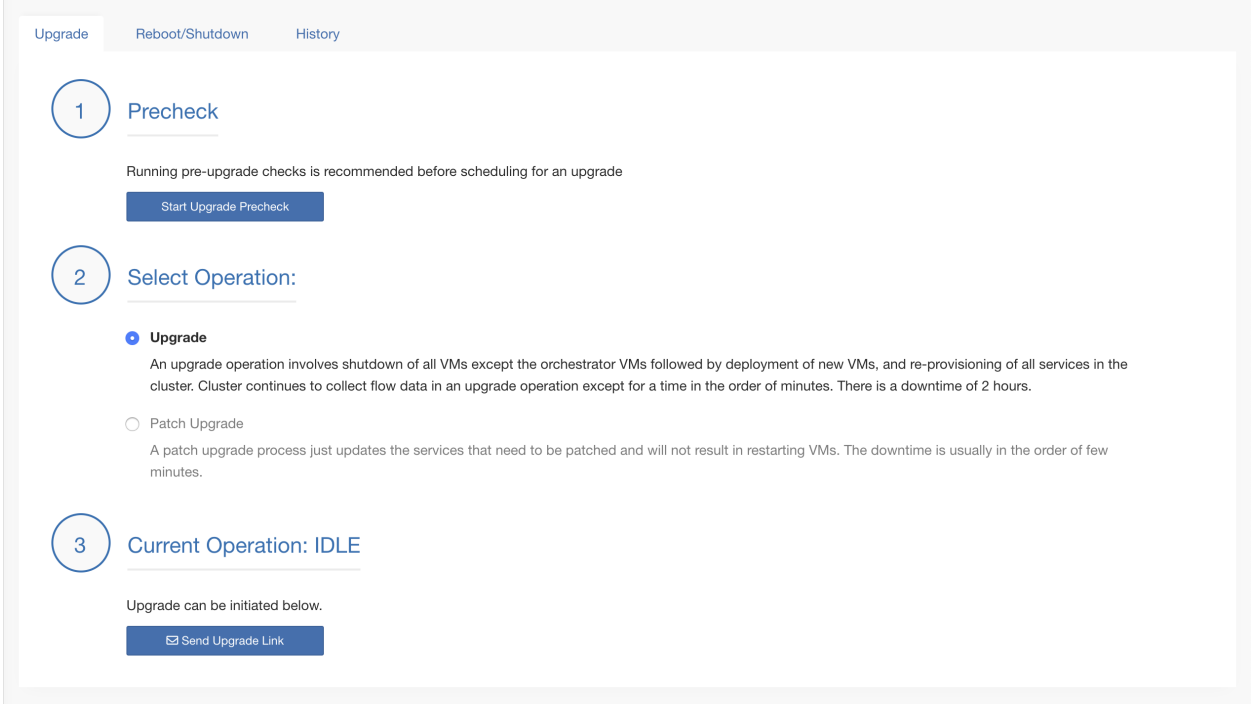


Fig. 11.6.1.2: Upgrade Tab

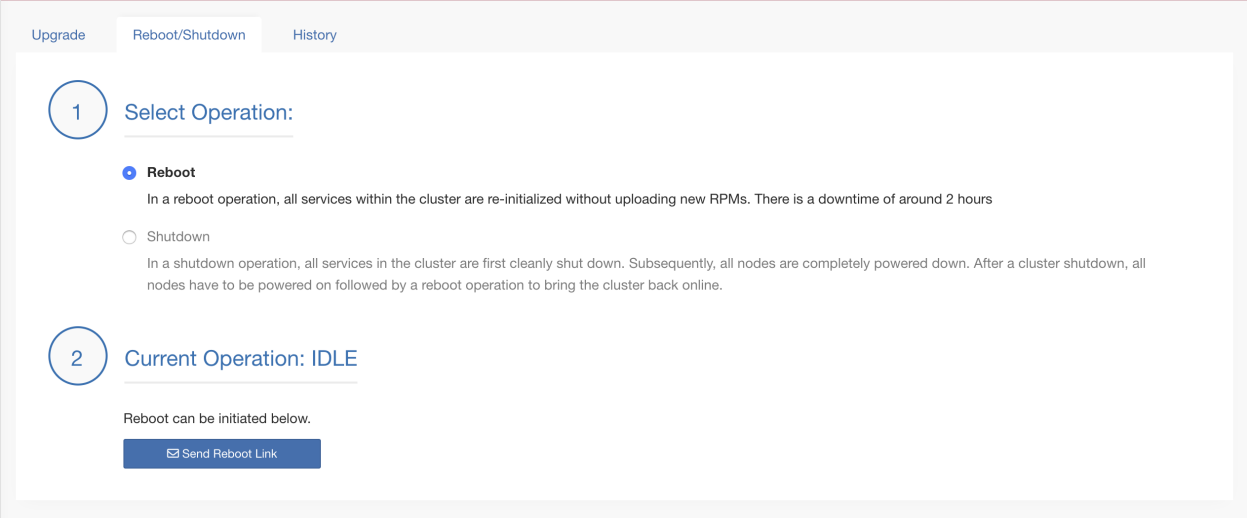


Fig. 11.6.1.3: Reboot/Shutdown Tab

To initiate a full upgrade, click on the Send Upgrade Link. Full Upgrade will shut down all the VMs other than the orchestrator VMs and upgrade all of them and re-deploy them. This results in 2+ hours of cluster downtime. Patch upgrade will minimize the downtime, but just updating the services that need to be patched and will not result in VM restarts. The downtime is usually in the order of few minutes. To initiate Patch Upgrade click on Send Patch Upgrade Link. Use Send Reboot Link to initiate cluster reboot after a power down. Clicking on either of these links will generate an email with a link in it and will send it to the user who initiated the upgrade.

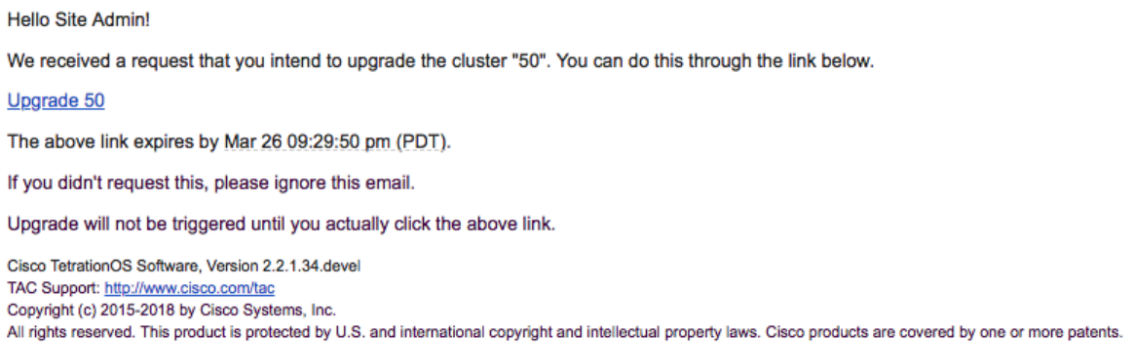


Fig. 11.6.1.4: Initiate a full upgrade

Before sending the email, orchestrator runs a number of verification checks to make sure the cluster is upgradable. The checks include:

1. Checks to see there are no decommissioned nodes
2. Checks each bare metal to make sure there are no hardware failures. This covers:
   (a) Drive failure
   (b) Drive predicted Failure
   (c) Drive missing
   (d) StorCLI failures

(e) MCE log failures

3. Checks to ensure we have all the BMs in commissioned state. Nothing less than 36 servers for 39RU and 6 for 8RU.

If there are any of these failures, an upgrade link will not be sent and you will see 500 error with information like HW failure or missing host and check orchestrator logs for more info. In this scenario, use explore to tail -100 on /local/logs/tetration/orchestrator/orchestrator.log in the host orchestrator.service.consul. This will provide detailed information about which one of the 3 checks caused the failure. This usually requires fixing the hardware and recommissioning the node. Once that is done we can restart upgrade by clicking on "Send Upgrade Link".

11.6.2 RPM Upload

Click on the link in the email will connect to the setup UI in the cluster. Setup UI is a operations UI that will be used for deploy/upgrade of the cluster. The initial page will show the list of RPMs that are currently installed in the cluster. This is also the upload page to upload all the RPMs

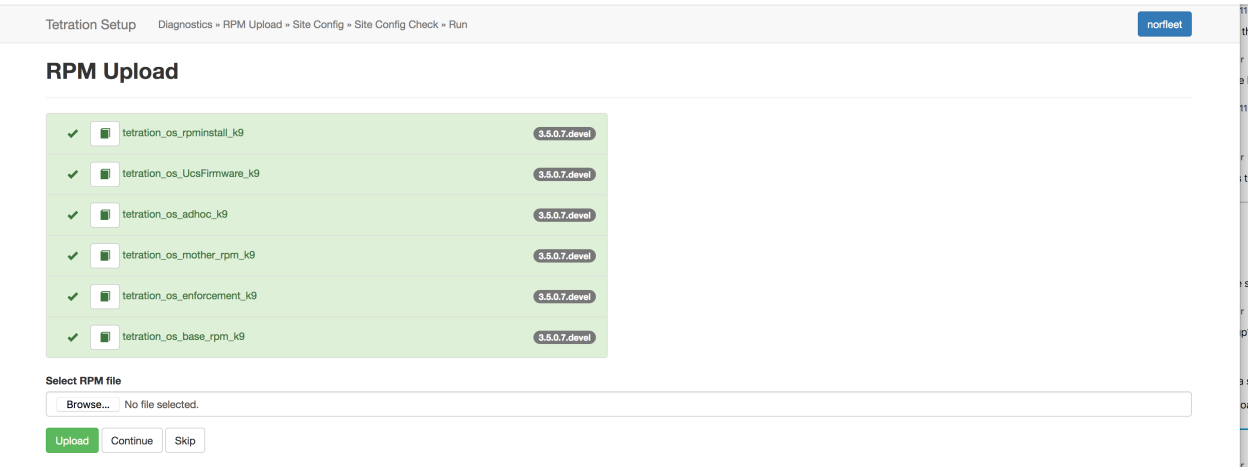


Fig. 11.6.2.1: RPM Upload

Upload the RPMs in the order that is shown on setup UI. The order is

1. tetration_os_rpminstall_k9
2. tetration_os_UcsFirmware_k9
3. tetration_os_adhoc_k9
4. tetration_os_mother_rpm_k9
5. tetration_os_enforcement_k9
6. tetration_os_base_rpm_k9

Note: For Tetration-V clusters deployed on vSphere, please be sure to also upgrade the tetration_os_ova_k9 RPM and do not upload the tetration_os_base_rpm_k9.

Uploading any other order will result in upload failure. Until all the RPMs are uploaded in the correct order Continue button will be disabled.

Logs for each upload can be seen by clicking on the Log symbol on the left of every RPM. Also uploads that failed will be marked RED in color.

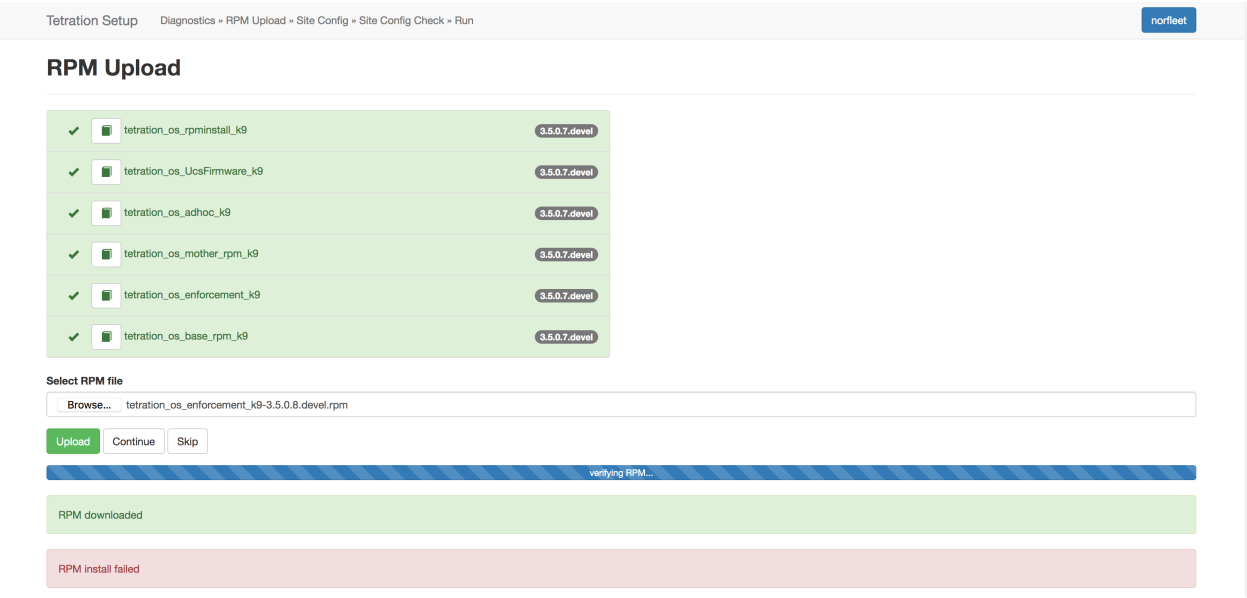


Fig. 11.6.2.2: RPM Upload log

11.6.3 Site Info

The next step is to update the site info. Not all site info fields are update-able. Only the following fields can be updated:

1. SSH public Key
2. Sentinel Alert Email (for Bosun)
3. CIMC Internal Network
4. CIMC Internal Network Gateway
5. External Network - NOTE - do not change the existing external network, you can add additional networks by appending to the existing ones. Changing or Removing existing network will make the cluster unusable.
6. DNS Resolvers
7. DNS Domain
8. NTP Servers
9. SMTP Server
10. SMTP Port
11. SMTP Username (Optional)
12. SMTP Password (Optional)
13. Syslog Server (Optional)
14. Syslog Port (Optional)
15. Syslog Severity (Optional)

Note: The syslog server severity ranges from critical to informational. Severity needs to be set to warning or higher (informational) for bosun alerts.

Note: From 3.1 version, **External syslog via setup UI is not supported**. Users will have to configure TAN Appliance to export data to syslog. Refer to *External syslog tunneling moving to TAN* for more details.

Note: Tetration supports secure SMTP communication with mail servers that support SSL/TLS communication via the STARTTLS command. The standard port for servers that support secure traffic is usually 587/TCP, but many servers also accept secure communication on the standard 25/TCP port.

Tetration does not support the SMTPS protocol for communicating with external mail servers.

Rest of the fields are NOT updatable. If there are no changes, click on Continue to trigger the Pre-Upgrade Checks, else update the fields and then click on Continue.

11.6.4 Pre Upgrade Checks

Before we start upgrades we do few checks on the cluster and ensure things are in order before we start upgrading:

1. RPM version checks - checks to ensure all the RPMs are uploaded and the version is correct. It doesn't check if the order was correct, just checks if it was uploaded. Note Order checks are done as a part of upload itself.
2. Site Linter - Does Site Info Linting
3. Switch Config - Configures the Leafs/Spine switches
4. Site Checker - Does DNS, NTP and SMTP server checks. Sends an email at the end with a token, the email is sent to the primary site admin account. If any of the services - DNS, NTP or SMTP is not usable, this step will fail.
5. Token Validation - Enter the token sent in the email and hit Continue.

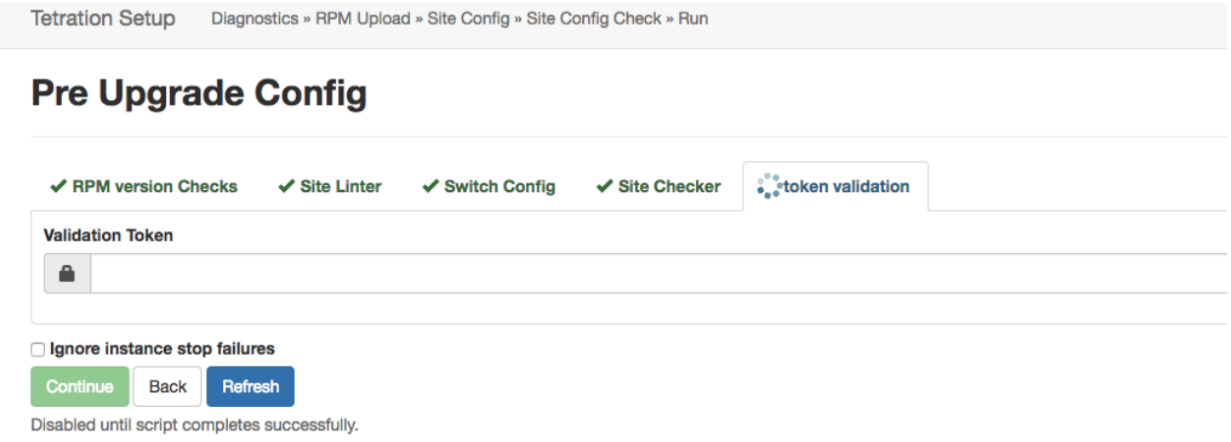


Fig. 11.6.4.1: Pre Upgrade Checks

11.6.5 Upgrading the Cluster

Once the pre-upgrade step finishes, after entering the token received in the "verify token email", you can hit "Continue" to start the upgrade. There is an additional option called "Ignore Stop Failures". Do not check this option. This is a recovery option when upgrade fails when certain services wouldn't shut down. Using this option will blindly shut the VMs down which can create failures when the services come back up. Use this option under Engineering's supervision.

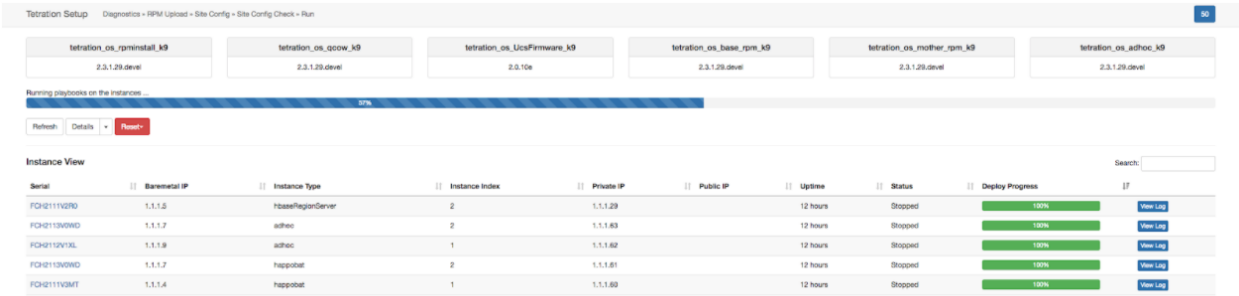


Fig. 11.6.5.1: Upgrading the Cluster

On clicking on "Continue" - Upgrade will start.

1. On the top right clicking on the cluster name will show the site info used.
2. Below that will have all tetration_os RPMs and their versions.
3. The global upgrade bar will show the upgrade progress. It will be blue in color while things are in progress, green when done and red when it fails. Right above the progress bar will show the current status of upgrade.
4. Then there are 3 buttons:
   (a) Refresh - will refresh the page
   (b) Details - Clicking on Details will show all the steps that have completed during this upgrade. Clicking on the arrow next to it will show all the logs that can be opened. More on this later.
   (c) Reset - This will have an option to Reset Orchestrator State. This Option will cancel the upgrade and take you back to the start. Do NOT use this unless the upgrade had failed and also give few minutes after upgrade had failed to let all the process reach completion before restarting upgraded.
   (d) Resume - When the upgrade fails, depending the stage it failed, Resume option will show up. Clicking on Resume will re-start upgrade from the previous stable part.
5. Then there are the instance view. Every individual VMs deploy status is tracked. The columns include:
   (a) Serial - Baremetal Serial that hosts this VM
   (b) Baremetal IP - the Internal IP assigned to this Baremetal
   (c) Instance Type - the type of VM
   (d) Instance Index - Index of the VM - there are multiple VMs of the same type for high-availability.
   (e) Private IP - the Internal IP assigned to this VM
   (f) Pubic IP - the routable IP assigned to this VM - not all VMs have this.
   (g) Uptime - Uptime of the VM
   (h) Status - Can be Stopped, Deployed, Failed, Not Started or In Progress.

(i) Deploy Progress - Deploy Percentage

(j) View Log - button to view the deploy status of the VM

11.6.6 Logs

There are two type of logs:

1. VM deployment logs - these logs can be seen by clicking on "View Log" button.
2. Orchestration Logs. These can be seen by clicking on the arrow next to the details button. It will show up:

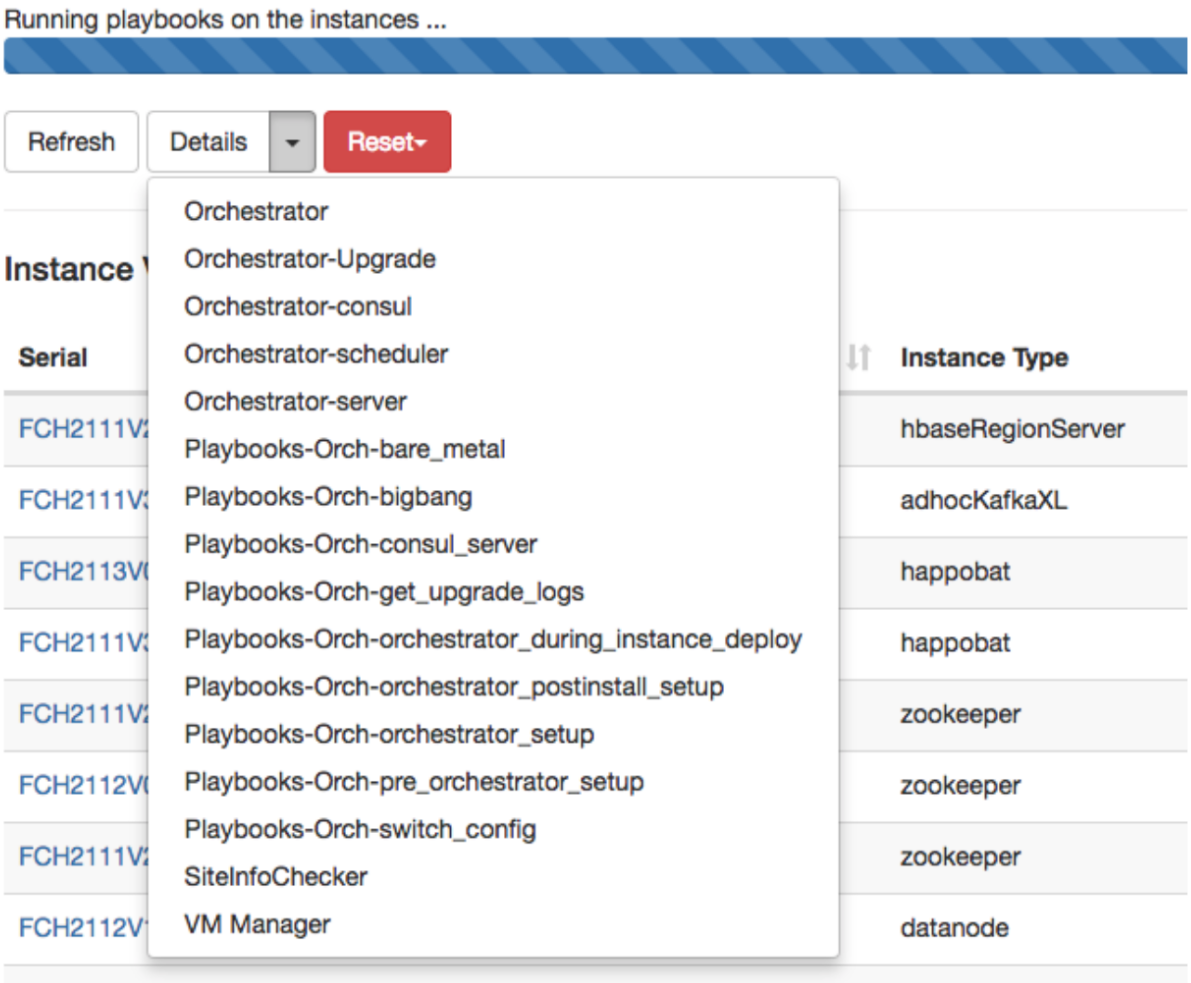


Fig. 11.6.6.1: Logs

Each of the links will point to the logs.

1. Orchestrator - Orchestrator log - this is the first place to track progress. Any failures will point to another log to look at.
2. Orchestrator-Upgrade - NOP for 2.3
3. Orchestrator-consul - consul logs that runs on primary orchestrator

4. Orchestrator-Scheduler - VM scheduler logs - which VM got placed on which baremetal and the scheduling log.
5. Orchestrator-server - HTTP server logs from orchestrator
6. Playbooks-* - all the playbook logs that run on orchestrator.

11.6.7 Running Pre-Upgrade Checks any time

Occasionally, after scheduling an upgrade and while initiating an upgrade, there might be a hardware failure or cluster is not ready to be upgraded. This might require to be fixed before proceeding with upgrades. Instead of waiting until an upgrade window, Pre-Upgrade checks can be initiated any time. These checks can be run any number of times and any time except when an upgrade/patch/reboot is initiated. To run Pre-Upgrade Checks any time, go to the Upgrade Page.

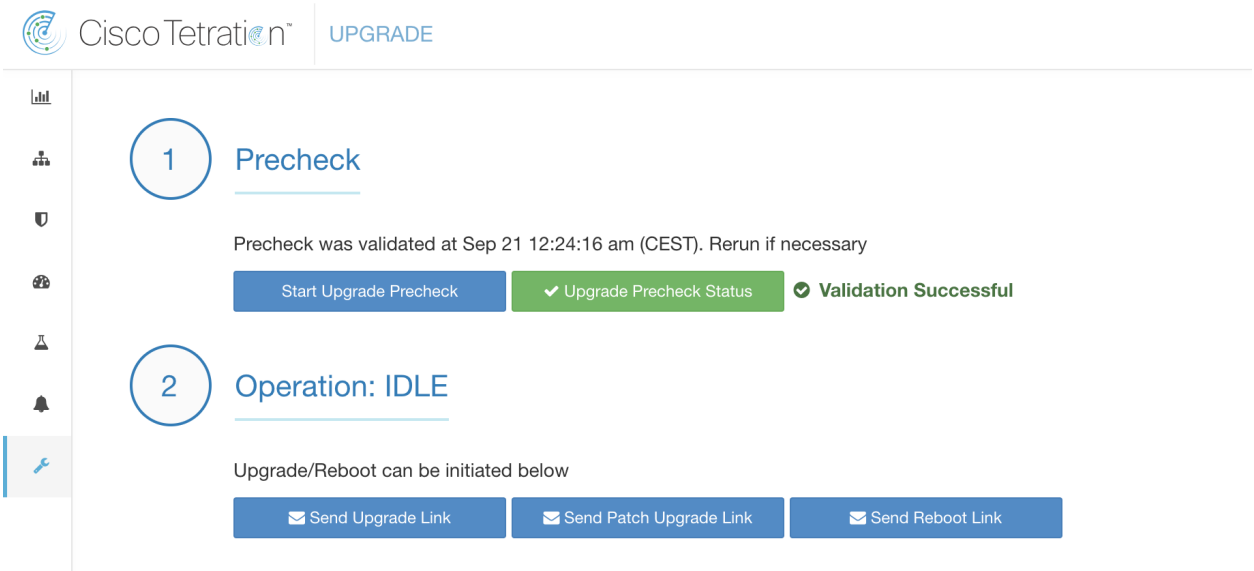


Fig. 11.6.7.1: Running Pre-Upgrade Checks any time steps

Click on the Start Upgrade Precheck. This will initiate the pre-upgrade checks and will transition to running state:

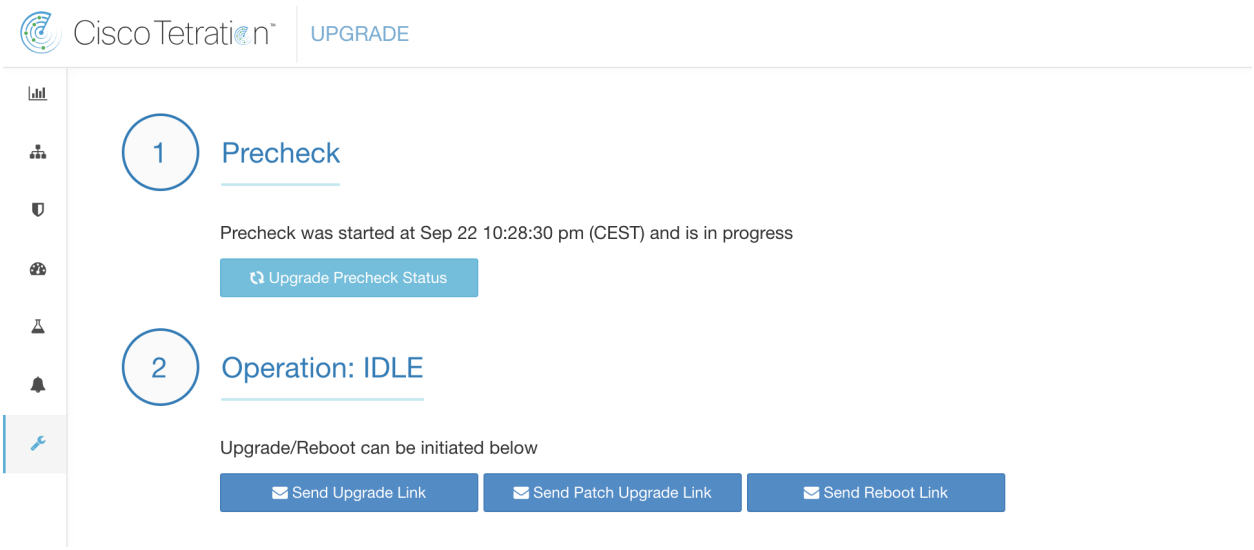


Fig. 11.6.7.2: Running Pre-Upgrade Checks any time steps

During this time orchestrator runs all the pre-upgrade checks. Once all the checks pass, an email will be sent to the user who initiated the check with an email token. Enter the token to complete the pre-upgrade checks.

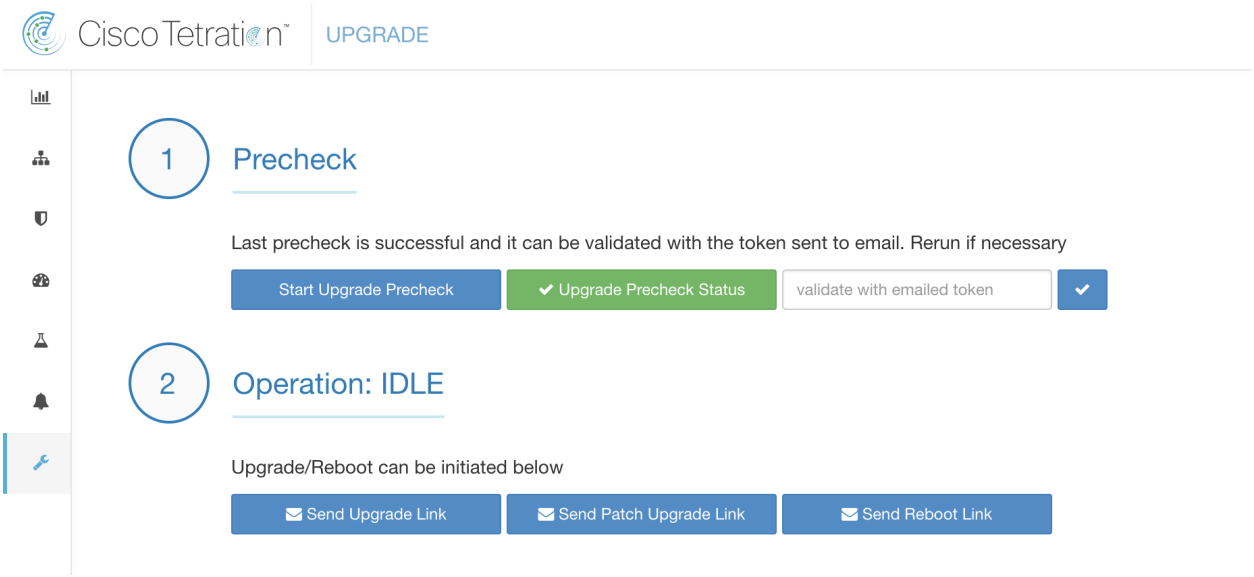


Fig. 11.6.7.3: Running Pre-Upgrade Checks any time steps

If there are any failures during pre-upgrade checks it will transition to failed state and will show which task failed. Any time the status can be checked and will show up in a new dialog box.

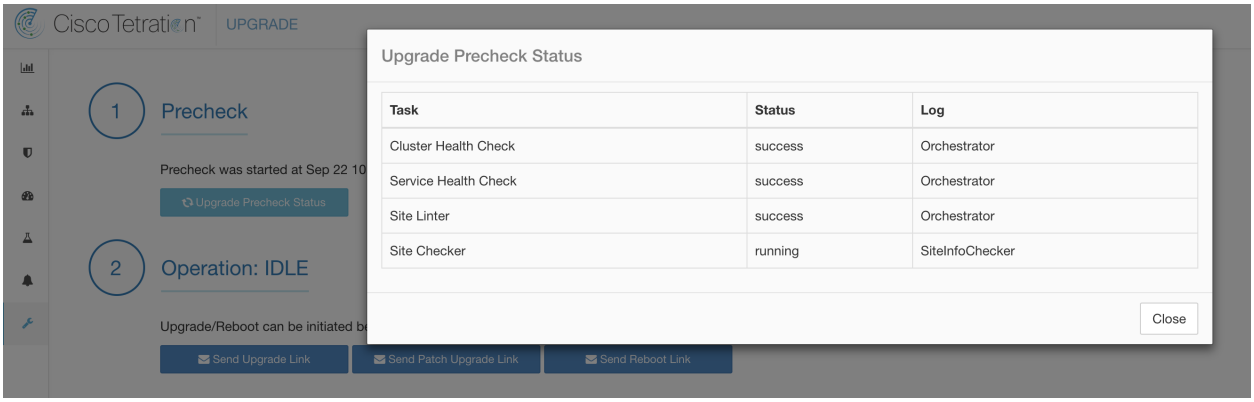


Fig. 11.6.7.4: Running Pre-Upgrade Checks any time steps

11.6.8 Data Backup and Restore (DBR)

If **DBR** is enabled on the cluster, please also see *Upgrades (with DBR)*.

11.7 Snapshots

11.7.1 Accessing the Snapshot Creation User Interface

Users with **Customer Support role** can access the snapshot tool by selecting Maintenance -> Snapshots from the top-level menu.

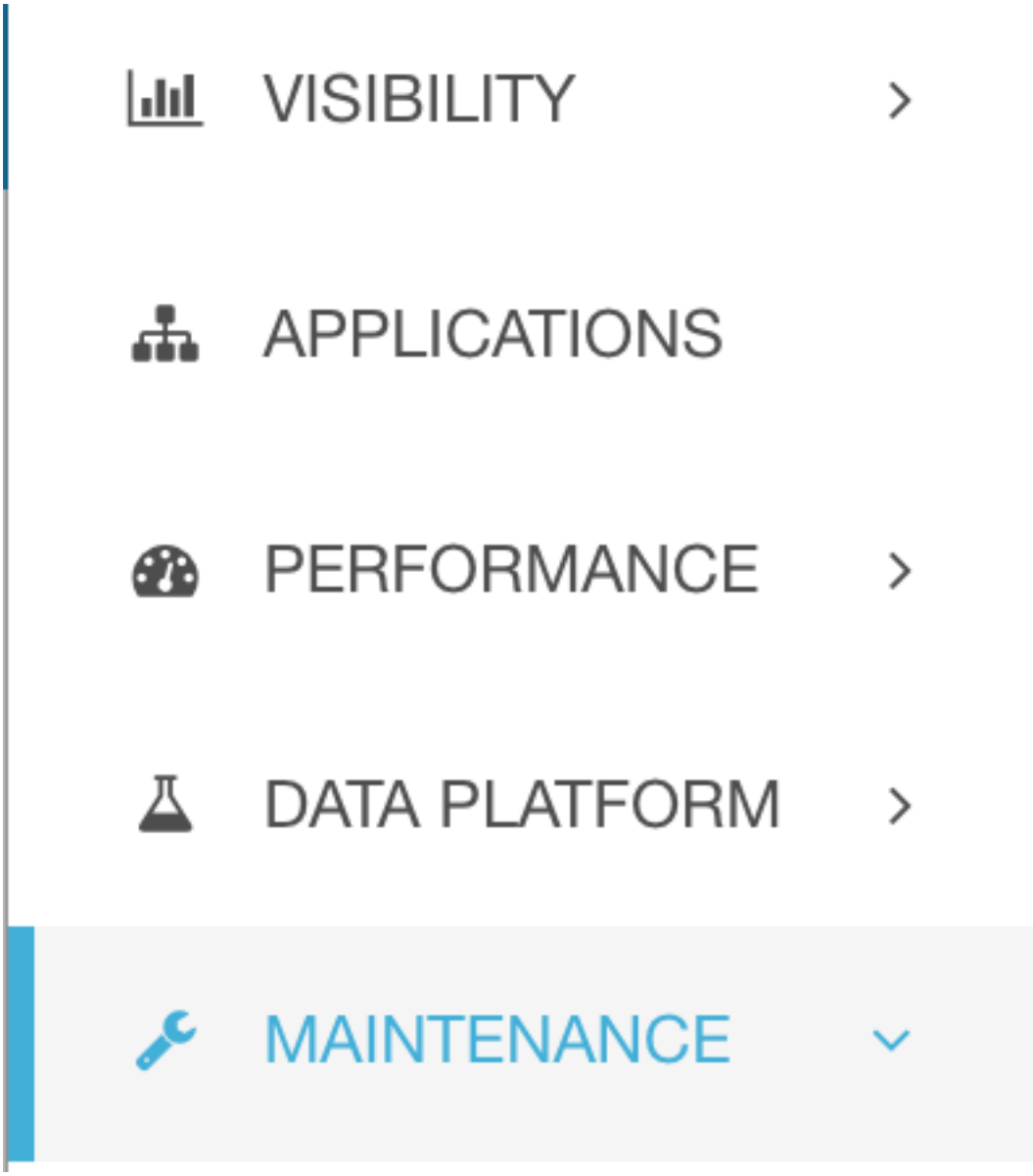


Fig. 11.7.1.1: Snapshot tool

Snapshot file list user interface:

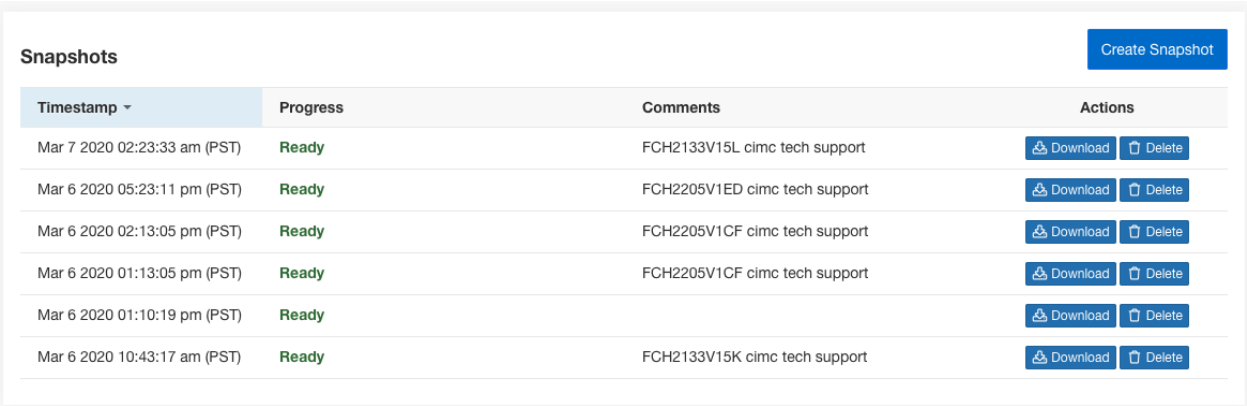


Fig. 11.7.1.2: Snapshot file list

The Snapshot tool can be used to create a Classic Snapshot or a Cisco Integrated Management Controller (CIMC) technical support bundles. Clicking on the Create Snapshot button on the Snapshot file list page loads a page to choose a Classic Snapshot or a CIMC Snapshot (technical support bundle). The option to choose a CIMC Snapshot is disabled on Tetration Software Only (ESXi) and Tetration as a Service (TaaS).

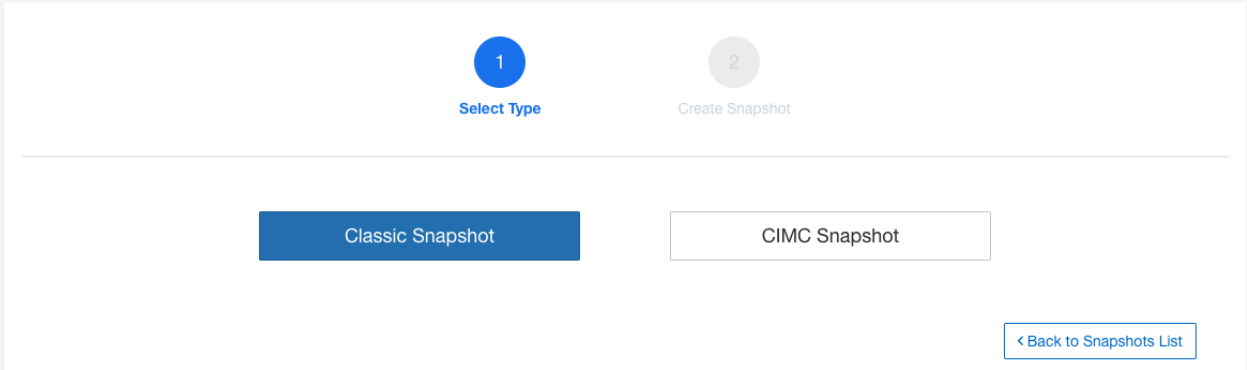


Fig. 11.7.1.3: Snapshot or CIMC Tech Support options

Clicking on the Classic Snapshot button loads the Snapshot tool runner user interface:

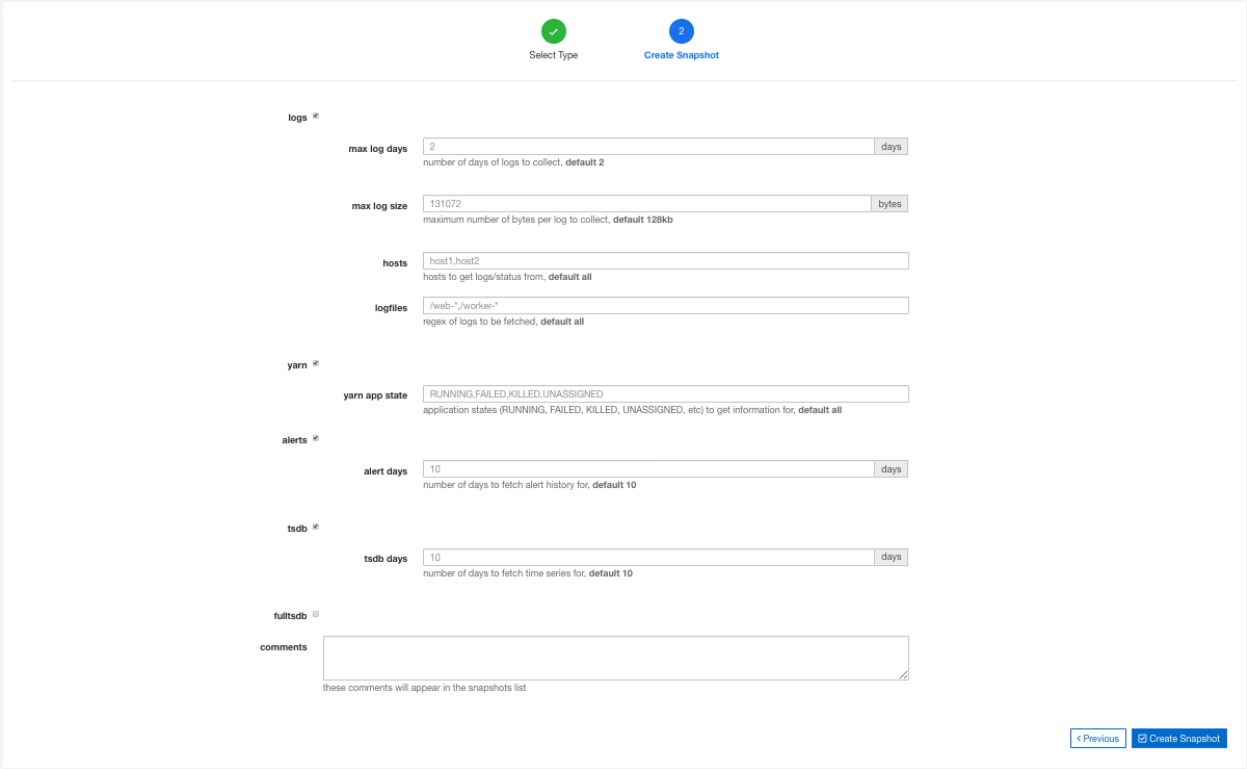


Fig. 11.7.1.4: Snapshot tool runner

Clicking on the CIMC Snapshot button loads the CIMC Technical Support tool runner user interface:

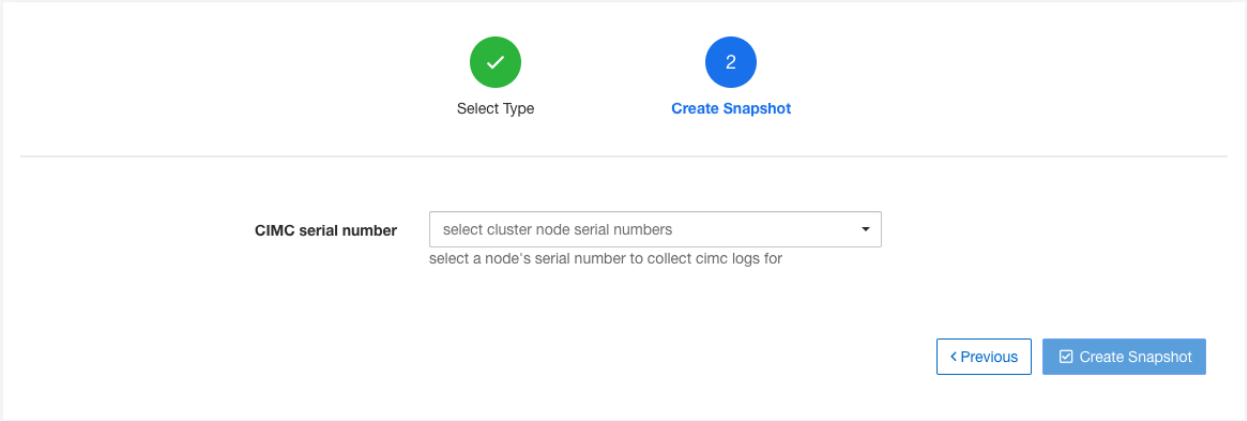


Fig. 11.7.1.5: CIMC Technical Support runner

11.7.2 Creating a Snapshot

Selecting Create Snapshot with the default options, the Snapshot tool collects:

- Logs
- State of Hadoop/YARN application and logs

- Alert history
- Numerous TSDB statistics

It is possible to override the defaults and specify certain options.

- logs options
  - max log days - number of days of logs to collect, default 2.
  - max log size - maximum number of bytes per log to collect, default 128kb.
  - hosts - hosts to get logs/status from, default all.
  - logfiles - regex of logs to be fetched, default all.
- yarn options
  - yarn app state - application states (RUNNING, FAILED, KILLED, UNASSIGNED, etc) to get information for, default all.
- alerts options
  - alert days - the number of days worth of alert data to collect.
- tsdb options
  - tsdb days - the number of days worth of tsdb data to collect, increasing this can create very large Snapshots.
- fulltsdb options
  - fulltsdb - a JSON object that can be used to specify startTime, endTime fullDumpPath, localDumpFile and nameFilterIncludeRegex to limit which metrics are collected.
- comments - can be added to describe why or who is collecting the snapshot.

After selecting Create Snapshot, a progress bar for the snapshot is displayed at the top of the Snapshot file list page. When the snapshot completes, it can be downloaded using the Download button on the Snapshots file list page. Only one snapshot can be collected at a time.

11.7.3 Creating a CIMC Technical Support Bundle

On the CIMC Snapshot (technical support bundle) page, select the serial number of the node the CIMC Technical Support Bundle should be created for and click the Create Snapshot button. A progres bar for the CIMC Technical Support Bundle collection will appear in the Snapshot file list page and the comments section will reflect that the CIMC Technical Support Bundle collection has been triggered. Once the CIMC Technical Support Bundle collection is complete, the file can be downloaded from the Snapshot file list page.

11.7.4 Using a Snapshot

Untarring a snapshot creates a ./clustername_snapshot directory that contains the logs for each machine. The logs are saved as text files that contain the data from several directories from the machines. The Snapshot also saves all the Hadoop/TSDB data that was captured in JSON format.

```
~/Downloads/tet-snapshot $ ls -lhrGg
total 93840
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 zookeeper-3
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 zookeeper-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 zookeeper-1
drwxr-xr-x@ 1691 staff    56K Mar 30 15:23 yarn
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 tsdbBosunGrafana-3
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 tsdbBosunGrafana-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 tsdbBosunGrafana-1
-rw-r--r--@    1 staff    45M Mar 30 15:22 tsdb.json
-rw-r--r--@    1 staff   4.8K Mar 30 15:19 tet_snapshot_manifest.json
-rw-r--r--@    1 staff    34K Mar 30 15:24 snapshot_report.log
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 secondaryNamenode-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 resourceManager-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 resourceManager-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 redis-3
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 redis-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 redis-1
drwxr-xr-x@   41 staff   1.4K Mar 30 15:21 orchestrator-3
drwxr-xr-x@   41 staff   1.4K Mar 30 15:21 orchestrator-2
drwxr-xr-x@   41 staff   1.4K Mar 30 15:21 orchestrator-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-9
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-8
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-7
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-6
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-5
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-4
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-3
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-10
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 nodemanager-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:24 namenode-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 mongodbArbiter-1
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 mongodb-2
drwxr-xr-x@   42 staff   1.4K Mar 30 15:23 mongodb-1
```

Fig. 11.7.4.1: Using a Snapshot

When opening the packaged index.html in a browser, there are tabs for:

- Terse list of alert state changes.

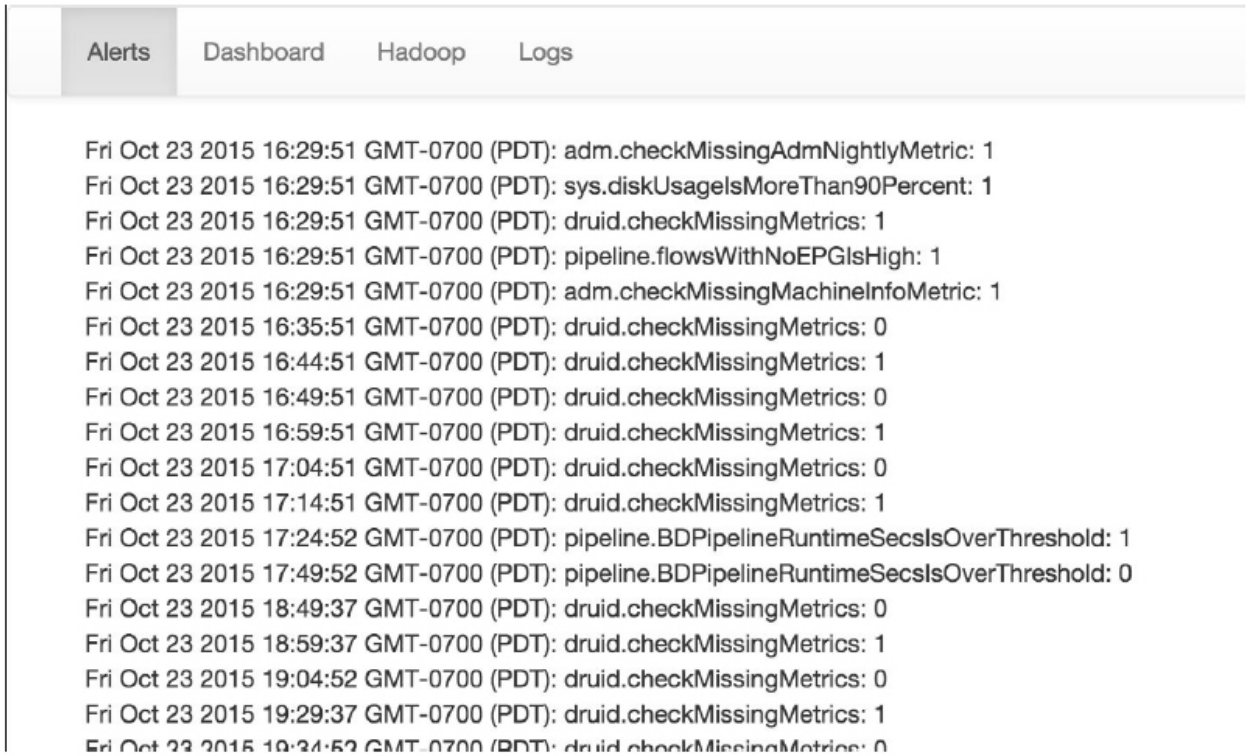


Fig. 11.7.4.2: Terse list of alert state changes

- Reproduction of grafana dashboards.

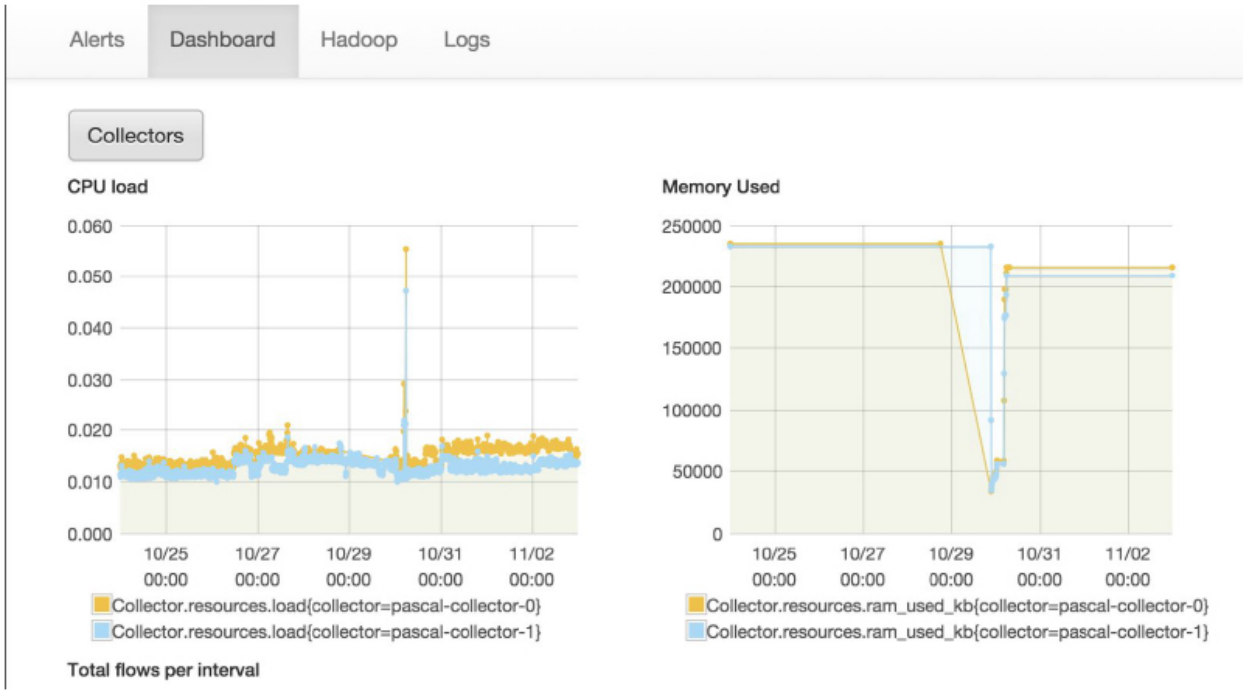


Fig. 11.7.4.3: Reproduction of grafana dashboards

- Reproduction of the Hadoop Resource Manager front end that contains jobs and their state. Selecting a job displays the logs for the job.

Alerts Dashboard Hadoop Logs

RUNNING FAILED All jobs

state	id	name	applicationType	elapsedTime
RUNNING	application_1442528378995_192995	com.tetration.pipeline.PipelineMain	SPARK	948440504
RUNNING	application_1442528378995_107366	com.tetration.pipeline.ActiveFlow	SPARK	2419532064
RUNNING	application_1442528378995_107368	com.tetration.pipeline.UberBidirCopier	SPARK	2419507170
RUNNING	application_1442528378995_107367	com.tetration.retention.RetentionMain	SPARK	2419512413
RUNNING	application_1442528378995_107[illegible]	com.tetration.pipeline.UberMachineInfoCopier	SPARK	2420352532
RUNNING	application_1442528378995_256357	attacks-index-generator-Optional.of([2015-11-02T23:21:00.000Z/2015-11-02T23:22:00.000Z])	MAPREDUCE	10483
RUNNING	application_1442528378995_256356	aggregated_flows-index-generator-Optional.of([2015-11-02T23:21:00.000Z/2015-11-02T23:22:00.000Z])	MAPREDUCE	10178
RUNNING	application_1442528378995_256355	hosts-index-generator-Optional.of([2015-11-02T23:22:00.000Z/2015-11-02T23:23:00.000Z])	MAPREDUCE	10513
RUNNING	application_1442528378995_256348	aggregated_flows-index-generator-Optional.of([2015-11-02T23:19:00.000Z/2015-11-02T23:20:00.000Z])	MAPREDUCE	115046
RUNNING	application_1442528378995_256354	sensor_stats-index-generator-Optional.of([2015-11-02T23:22:00.000Z/2015-11-02T23:23:00.000Z])	MAPREDUCE	10721
RUNNING	application_1442528378995_256351	aggregated_flows-index-generator-Optional.of([2015-11-02T23:20:00.000Z/2015-11-02T23:21:00.000Z])	MAPREDUCE	60209
RUNNING	application_1442528378995_256344	aggregated_flows-index-generator-Optional.of([2015-11-02T23:18:00.000Z/2015-11-02T23:19:00.000Z])	MAPREDUCE	164729
FINISHED	application_1442528378995_253998	attacks-index-generator-Optional.of([2015-11-02T13:32:00.000Z/2015-11-02T13:33:00.000Z])	MAPREDUCE	47868
FINISHED	application_1442528378995_253997	sensor_stats-index-generator-Optional.of([2015-11-02T13:33:00.000Z/2015-11-02T13:34:00.000Z])	MAPREDUCE	24514

Fig. 11.7.4.4: Reproduction of the Hadoop Resource Manager

- List of all logs collected.

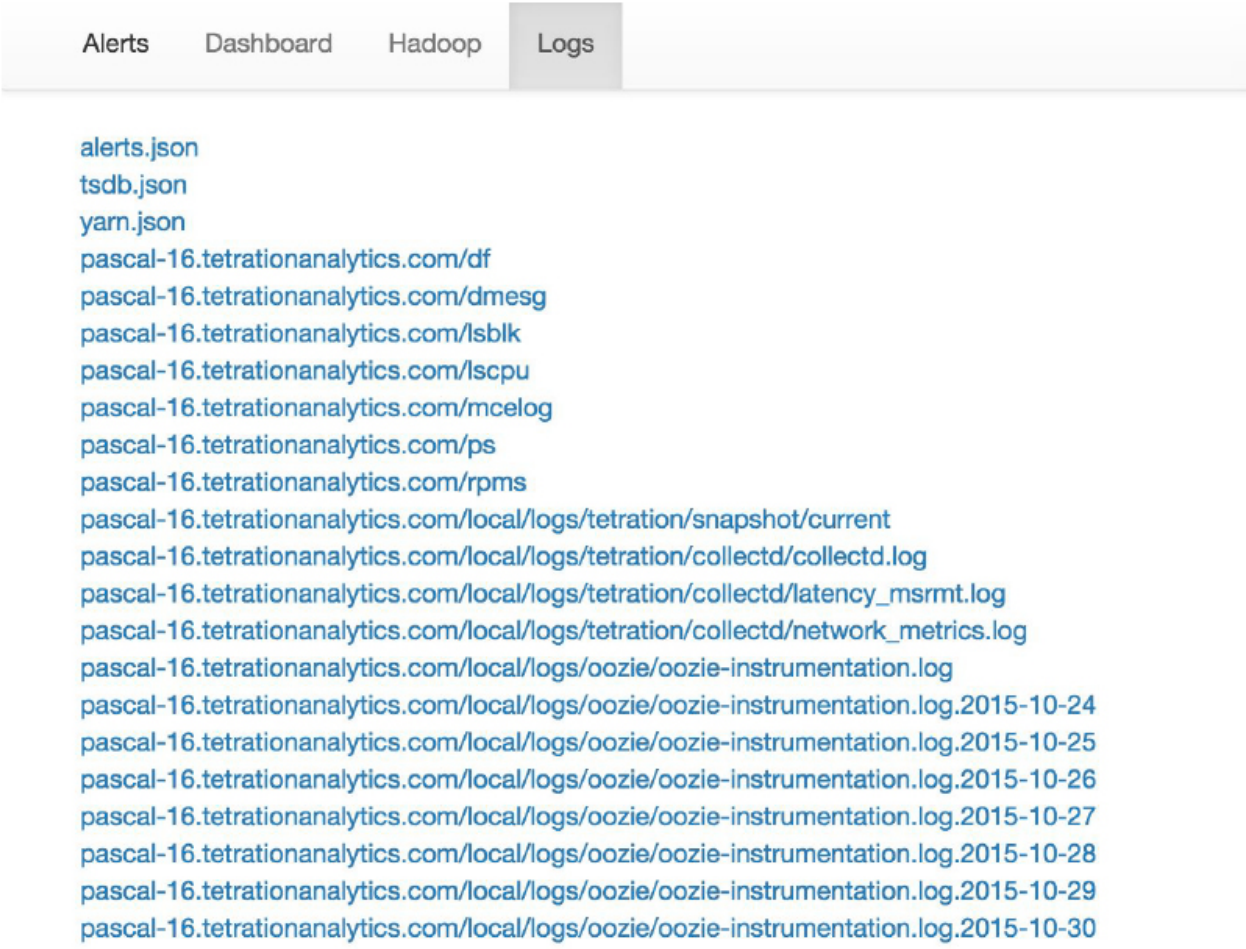


Fig. 11.7.4.5: List of all logs collected.

11.7.5 Using the Snapshot Service for Debugging and Maintenance

The snapshot service can be used to run service commands, but it requires Customer Support privileges.

Using the Explore tool, you can hit arbitrary URIs within the cluster:

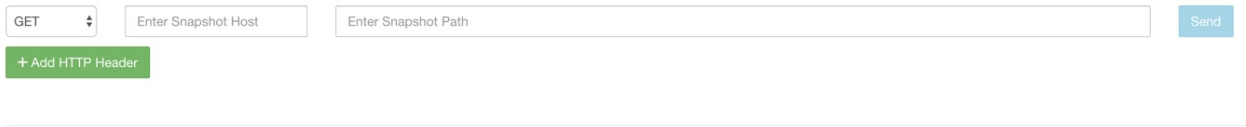


Fig. 11.7.5.1: Using the Snapshot Service for Debugging and Maintenance Example

The Explore tool only appears for users with Customer Support privileges.

The snapshot service runs on port 15151 of every node. It listens only on the internal network (not exposed externally) and has POST endpoints for various commands.

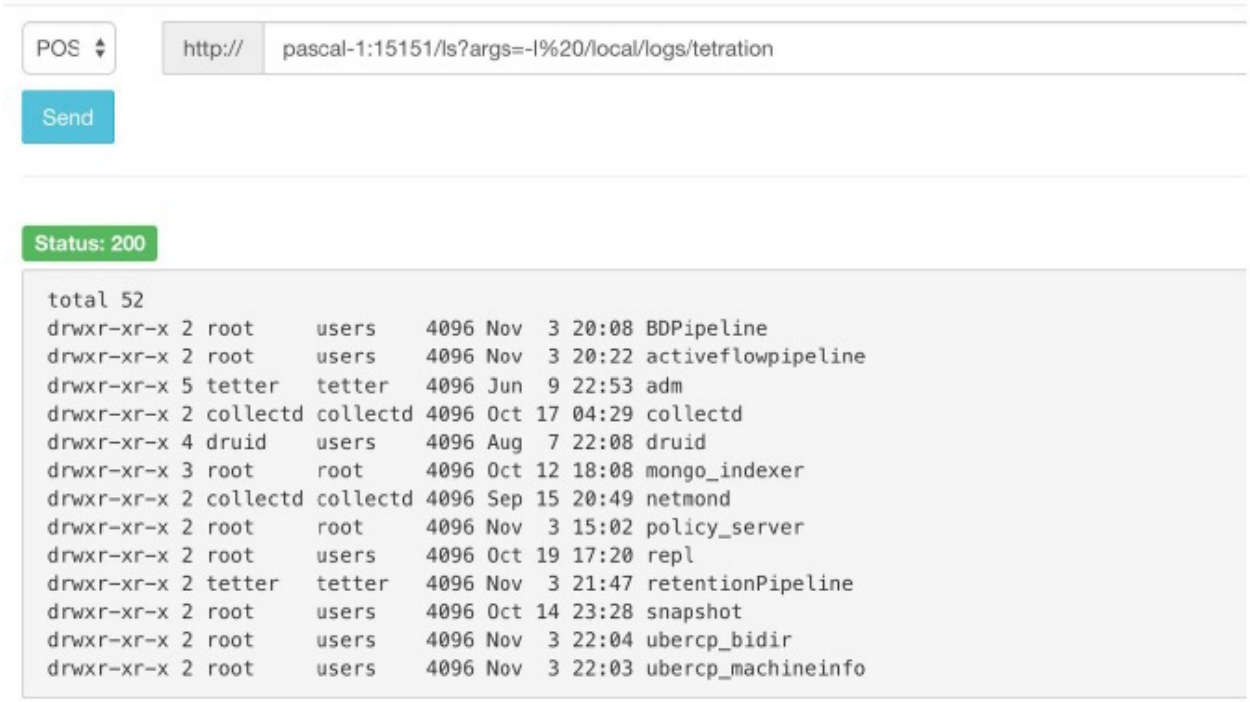


Fig. 11.7.5.2: Using the Snapshot Service for Debugging and Maintenance Example

The URI you must hit is **POST** http://<hostname>:15151/<cmd>?args=<args>, where args are space separated and URI encoded. It does **not** run your command with a shell. This would avoid allowing anything to be run.

Endpoints of a snapshot are defined for:

- **snapshot 0.2.5**
  - ls
  - **svstatus, svrestart - runs sv status, sv restart** Example: 1.1.11.15:15151/svrestart?args=snapshot
  - hadoopls runs hadoop fs -ls <args>
  - hadoopdu - runs hadoop fs -du <args>
  - **ps** Example: 1.1.11.31:15151/ps?args=eafux
  - du
  - ambari - runs ambari_service.py
  - monit
  - MegaCli64 (/usr/bin/MegaCli64)
  - service
  - hadoopfsck - runs hadoop -fsck
- **snapshot 0.2.6**
  - makecurrent - runs make -C /local/deploy-ansible current
  - netstat
- **snapshot 0.2.7 (run as uid "nobody")**

  - cat
  - head
  - tail
  - grep
  - ip -6 neighbor
  - ip address
  - ip neighbor

There is another endopint, POST /runsigned, which will run shell scripts signed by Tetration. It runs gpg -d on the POSTed data. If it can be verified against a signature, it will run the encrypted text under a shell. This means importing a public key on each server as part of the ansible setup and the need to keep the private key secure.

11.7.6 Run Book

Users with Customer Support privileges can use Run Book by selecting Maintenance in the Monitoring drop-down and then navigating to "Explore". Select **POST** from the drop-down menu. (Otherwise you will receive Page Not Found errors when running commands.)

Using the snapshot REST endpoint to restart services:

- **druid: 1.1.11.17:15151/service?args=supervisord%20restart**
  - druid hosts are all IPs .17 through .24; .17, .18 are coordinators, .19 is the indexer, and .20-.24 are brokers
- **hadoop pipeline launchers:**
  - 1.1.11.25:15151/svrestart?args=activeflowpipeline
  - 1.1.11.25:15151/svrestart?args=adm
  - 1.1.11.25:15151/svrestart?args=batchmover_bidir
  - 1.1.11.25:15151/svrestart?args=batchmover_machineinfo
  - 1.1.11.25:15151/svrestart?args=BDPipeline
  - 1.1.11.25:15151/svrestart?args=mongo_indexer
  - 1.1.11.25:15151/svrestart?args=retentionPipeline
- **policy engine**
  - 1.1.11.25:15151/svrestart?args=policy_server
- **wss**
  - 1.1.11.47:15151/svrestart?args=wss

11.8 Explore/Snapshot Endpoints Overview

To run any endpoint, you will need to go to the Maintenance -> Explore page from the top-level menu.

You can also view each endpoint overview in the explore page by running a **POST** command on any host as **<endpoint>?usage=true**.

For example: **makecurrent?usage=true**

11.8.1 GET commands

Endpoint	Description
bm_details	• Displays the baremetals information
endpoints	• Lists all the endpoints on the host
members	• Displays the current list of consul members, along with their status
port2cimc	• Lists the IPs that the port is connected to • Should be run on the **orchestrator hosts only**
status	• Displays the status of the snapshot service on the host
vm_info	• Displays the VM information of the location • Should be run on the **Baremetal hosts only** • Run endpoint as **vm_info?args=<vmname>**

11.8.2 POST commands

Endpoint	Description
add_data_export_license	• Grant data export license to a Tenant (using vrf-id) • To get vrf_id, visit Tenants page under gears icon on Tetration UI • Run endpoint as **add_data_export_license?args=<vrf-id>** • This command is supported only on TaaS cluster
cat	• wrapper command for unix 'cat' command
cimc_password_random	• Randomizes the CIMC password. • Should be run on the **orchestrator hosts only**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
cleancmdlogs	• Clears the logs in /local/logs/tetration/snapshot/cmdlogs/ snapshot_cleancmdlogs_log
clear_sel	• Clears the system event logs • Should be run on the **Baremetal hosts only**
cluster_fw_upgrade	• This is a **BETA** feature. • Run a UCS firmware upgrade across the whole cluster. • After this completes successfully each bare metal will need to be rebooted to activate the BIOS and other component firmware. • Run as: **cluster_fw_upgrade** • This endpoint will kick off and monitor the firmware upgrade and update the log file when a stage of the upgrade has been started or completed. • Please use the **cluster_fw_upgrade_status** endpoint to get the full upgrade status.
cluster_fw_upgrade_status	• This is a **BETA** feature. • Get the status of the full cluster UCS firmware upgrade. • Run as **cluster_fw_upgrade_status**
cluster_powerdown	• Powers down the cluster • USE WITH CAUTION, BRINGS THE CLUSTER DOWN • Run endpoint as **cluster_powerdown?args=–start**
create_data_export	• Create data export tasks for a Tenant (using vrf-id) and data source • Two supported data sources are: 'aggregated_flows' and 'active_inventory' • To get vrf_id, visit Tenants page under gears icon on Tetration UI • Run endpoint as **create_data_export?args=<vrf-id> <data-source>**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
collector_status	• Displays the status of the collector • Should be run on the **collector hosts only**
consul_kv_export	• Displays k-v pairs from consul in JSON format • Should be run on the **orchestrator hosts only**
consul_kv_recurse	• Displays k-v pairs from consul in tabular format • Should be run on the **orchestrator hosts only**
delete_data_export	• Delete data export task • Delete by task_id, run list_data_export and 'id' field in the response is task_id • Run endpoint as **delete_data_export?args=<task-id>**
df	• wrapper command for unix 'df' command
dig	• wrapper command for unix 'dig' command
dmesg	• wrapper command for unix 'dmesg' command
dmidecode	• wrapper command for unix 'dmidecode' command
druid_coordinator_v1	• Displays the druid stats.
du	• wrapper command for unix 'du' command
dusorted	• wrapper command for unix 'dusorted' command
externalize_change_tunnel	• Changes the collector IP that will be used to tunnel the CIMC UI • Run as: **externalize_change_tunnel?method=POST** • Pass **{"collector_ip" : "<IP>"}** in the Body • Should be run on the **orchestrator hosts only**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
externalize_mgmt	• Displays the current status of externalizing the CIMC UI's for each server • Displays the address and time remaining for externalization • Should be run on the **orchestrator hosts only**
externalize_mgmt_read_only_password	• Changes the read only password (ta_guest) for both the switch and CIMC UI • Changes only when they are externalized • Run as: **externalize_mgmt_read_only_password?method=POST** • Pass **{"password" : "<password>"}** in the Body • Should be run on the **orchestrator hosts only**
fsck	• wrapper command for unix 'fsck' command • Should be run on **Baremetal host only**
get_cimc_techsupport	• **INPUT** Internal IP address of BM. • Retrieves the CIMC techsupport. • Once it is completed it will be available for download from the snapshots page in the UI. • This can be run from any host on the cluster and requires the baremetal internal ip address as an argument. • Example: **get_cimc_techsupport?args=1.1.0.9**
syslog_endpoints	• Controls the syslog configurations for 1 or more of the ucs servers. • **Run the command with -h to get full list of parameters**
grep	• wrapper command for unix 'grep' command
hadoopbalancer	• Distributes HDFS data uniformly across all nodes • Should be run on **hosts that have hdfs** for example launcherhost
hadoopdu	• Prints the directory utilization of hdfs • Should be run on **hosts that have hdfs** for example launcherhost

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
hadoopfsck	• Runs hadoop fsck and reports the state of the provided hdfs file system • It also takes "-delete "as an argument to clear corrupt or missing blocks • Before deleting make sure all the DataNodes are up else we might lose data • Should be run on the **launcher hosts only** • To report state run as: **hadoopfsck?args=/raw** • To delete corrupt files run as: **hadoopfsck?args=/raw -delete**
hadoopls	• Lists the Hadoop File System • Should be run on **hosts that have hdfs** for example launcherhost
hbasehbck	• Checks for consistency and table integrity problems and repairing a corrupted HBase • Should be run on the **HBase hosts only** • To identify inconsistency, run as: **hbasehbck?args=-details** • To repair a corrupted HBase, run as: **hbasehbck?args=-repair** • **Output written to: /local/logs/tetration/snapshot/cmdlogs/** snapshot_hbasehbck_log.txt • **Repair with caution**
hdfs_safe_state_recover	• Removes HDFS from safe state • Required if HDFS is in READ_ONLY_STATE due full capacity and space has been cleared • Should be run on the **launcher hosts only** • Run as: **hadoop fs -rm '{{ hdfs_safe_state_marker_location }}/HDFS_READ_ONLY'**
initctl	• wrapper command for unix 'initctl' command
head	• wrapper command for unix 'head' command
internal_haproxy_status	• Prints the internal haproxy status and stats • Should be run on the **orchestrator hosts only**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
ip	• wrapper command for unix 'ip' command
ipmifru	• Prints Field Replaceable Unit (FRU) Information • Should be run on the **Baremetal hosts only**
ipmilan	• Prints the LAN configuration • Should be run on the **Baremetal hosts only**
ipmisel	• Prints System Event Log (SEL) entries • Should be run on the **Baremetal hosts only**
ipmisensorlist	• Prints the IPMI sensor information • Should be run on the **Baremetal hosts only**
jstack	• Prints Java stack traces of Java threads for a given Java process or core file
list_data_export	• Lists data export tasks for a Tenant (using vrf-id) • Run endpoint as **list_data_export?args=<vrf-id>**
list_data_export_license	• List Tenants with data export license granted • Run endpoint as **list_data_export_license** • This command is supported only on TaaS cluster
ls	• wrapper command for unix 'ls' command
lshw	• wrapper command for unix 'lshw' command
lsof	• wrapper command for unix 'lsof' command
lvdisplay	• wrapper command for unix 'lvdisplay' command
lvs	• wrapper command for unix 'lvs' command

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
lvscan	• wrapper command for unix 'lvscan' command
makecurrent	• Resets/fastforwards the pipeline processing the marker to the current timestamps • Should be run on the **orchestrator nodes only** • Run endpoint as **makecurrent?args=–start**
mongo_rs_status	• Displays the mongo replication status • Should be run or either the **mongodb or the enforcementpolicystore hosts**
mongo_stats	• Displays the mongo stats • Should be run or either the **mongodb or the enforcementpolicystore hosts**
mongodump	• Dumps the collections from the database • Should be run or either the **mongodb or the enforcementpolicystore hosts** • Run as: mongodump?args=<collection>[–db DB]
monit	• wrapper command for unix 'monit' command
namenode_jmx	• Displays the primary namenode jmx metrics
netstat	• wrapper command for unix 'netstat' command
ntpq	• wrapper command for unix 'ntpq' command
orch_reset	• Resets orchestrator state to IDLE • Run after commissioning or decommissioning failure • Should be run on the **orchestrator.service.consul host only** • **Do not use this command without consulting customer support**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
orch_stop	• Stops the orchestrator primary and trigger a switchover • Should be run on the **orchestrator.service.consul host only** • **USE WITH CAUTION**
ping	• wrapper command for unix 'ping' command
ps	• wrapper command for unix 'ps' command
pv	• wrapper command for unix 'pv' command
pvs	• wrapper command for unix 'pvs' command
pvdisplay	• wrapper command for unix 'pvdisplay' command
rebootnode	• Reboots the node • Should be run on the **Baremetal hosts only**
recover_rpmdb	• Recovers a corrupt RPMDB on a node • Can be run on Baremetals or VMs
recoverhbase	• Recovers Hbase and TSDB Service • Should be run on **orchestrator hosts only** • Should be run when HDFS is Healthy
recovervm	• Try to recover VM via stop/fsck/start • Should be run on **orchestrator hosts only** • Run endpoint as **recovervm?args=<vmname>**
remove_data_export_license	• Revoke data export license for a Tenant (using vrf-id) • To get vrf_id, visit Tenants page under gears icon on Tetration UI • Run endpoint as **remove_data_export_license?args=<vrf-id>** • This command is supported only on TaaS cluster

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
restartservices	• Stops and starts all non UI services • Should be run on the **orchestrator.service.consul host only** • **USE WITH CAUTION** • Run endpoint as **restartservices?args=–start**
runsigned	• Runs the signed script provided by cisco • **Follow the steps provided in the script guidelines**
service	• wrapper command for unix 'service' command
storcli	• wrapper command for unix 'storcli' command
sudocat	• wrapper for 'cat' command that works only under /var/log or /local/logs
sudogrep	• wrapper for 'grep' command that works only under /var/log or /local/logs
sudohead	• wrapper for 'head' command that works only under /var/log or /local/logs
sudols	• wrapper for 'ls' command that works only under /var/log or /local/logs
sudotail	• wrapper for 'tail' command that works only under /var/log or /local/logs
sudozgrep	• wrapper for 'zgrep' command that works only under /var/log or /local/logs
sudozcat	• wrapper for 'zcat' command that works only under /var/log or /local/logs
svrestart	• Restarts the service mentioned, run command as **svrestart?args=<servicename>**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
svstatus	• Prints the status of the service mentioned, run as **svstatus?args=<servicename>**
switchinfo	• Get the information about the cluster switches
switch_namenode	• Manually fail over namenode from primary or secondary • Should be run on the **orchestrator.service.consul** host only • Run while recommision or decommision of namenode hosts • Run endpoint as **switch_namenode?args=–start**
switch_secondarynamenode	• Manually fail over secondarynamenode from secondary to primary • Should be run on the **orchestrator.service.consul** host only • Run while recommision or decommision of namenode hosts • Run endpoint as **switch_secondarynamenode?args=–start**
switch_yarn	• Manually fail over resourcemanager from primary or secondary or vice versa • Should be run on the **orchestrator.service.consul host only** • Run while recommision or decommision of resourcemanager hosts • Run endpoint as **switch_yarn?args=–start**
tail	• wrapper command for unix 'tail' command
toggle_chassis_locator	• Toggle a chassis locator on a physical bare metal specified by the node serial number. • Run from any node as: **toggle_chassis_locator?method=POST** • Set the body to a JSON object that describes the host serial number (only one serial number is supported at a time), for example: **{"serials": ["FCH2308V0FH"]}**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
tnp_agent_logs	• Create a snapshot with all log files provided by Load Balancer agents registered as External Orchestrators • Should be run on the launcherhost hosts
tnp_datastream	• Create a snapshot with policy stream data consumed by Load Balancer policy enforcement agents registered as External Orchestrators • Should be run on the orchestrator hosts • In order to download policy status stream data run endpoint as **tnp_datastream?args=–ds_type datasink**
ui_haproxy_status	• Prints the haproxy stats and status for external haproxy
uptime	• wrapper command for unix 'uptime' command
userapps_kill	• Kills all the running user application • Should be run on the **launcherhost hosts only**
vgdisplay	• wrapper command for unix 'vgdisplay' command
vgs	• wrapper command for unix 'vgs' command
vmfs	• Lists the file system on a VM • Should be run on the **Baremetal hosts only** • Run endpoint as **vmfs?args=<vmname>**
vminfo	• Prints the VM information • Should be run on the **Baremetal hosts only** • Run endpoint as **vminfo?args=<vmname>**
vmlist	• Lists of all the VM on a baremetal • Should be run on the **Baremetal hosts only** • Run endpoint as **vmlist?args=<vmname>**

Continued on next page

Table 11.8.2.1 – continued from previous page

Endpoint	Description
vmreboot	• Reboots the VM • Should be run on the **Baremetal hosts only** • Run endpoint as **vmreboot?args=<vmname>**
vmshutdown	• Gracefully shutsdown the VM • Should be run on the **Baremetal hosts only** • Run endpoint as **vmshutdown?args=<vmname>**
vmstart	• Starts the VM • Should be run on the **Baremetal hosts only** • Run endpoint as **vmstart?args=<vmname>**
vmstop	• Force shutsdown the VM • Should be run on the **Baremetal hosts only** • Run endpoint as **vmstop?args=<vmname>**
yarnkill	• Kills a running Yarn application • Should be run on the **launcherhost hosts only** • Run endpoint as **yarnkill?args=<application id>** • To kill all the appplications run as **yarnkill?args=ALL**
yarnlogs	• Dumps the last 500 mb of yarn application logs • Should be run on the **launcherhost hosts only** • Run endpoint as **yarnlogs?args=<application id> <job user>**
zcat	• wrapper command for unix 'zcat' command
zgrep	• wrapper command for unix 'zgrep' command

11.9 Server Maintenance

Server Maintenance involves replacement of any faulty server component like Hard Disk, Memory or replacement of the entire server itself. **Note**: If there are multiple servers on the cluster that need maintenance then do server maintenance on them one at a time. Decommissioning multiple servers at the same time can lead to loss of data.

The **Cluster Status** page is used to perform all the steps involved in server maintenance. It can be accessed by all users but the actions can be carried out by **Customer Support** users only. It shows the status of all the physical servers in Cisco Tetration rack.

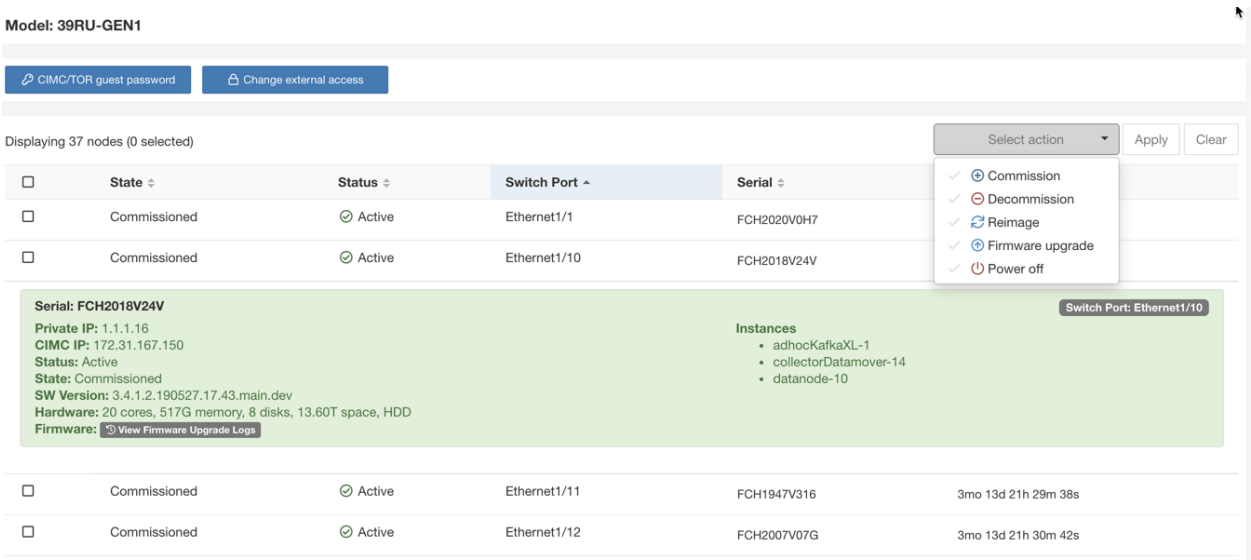


Fig. 11.9.1: Server Maintenance

Steps involved in server or component replacement

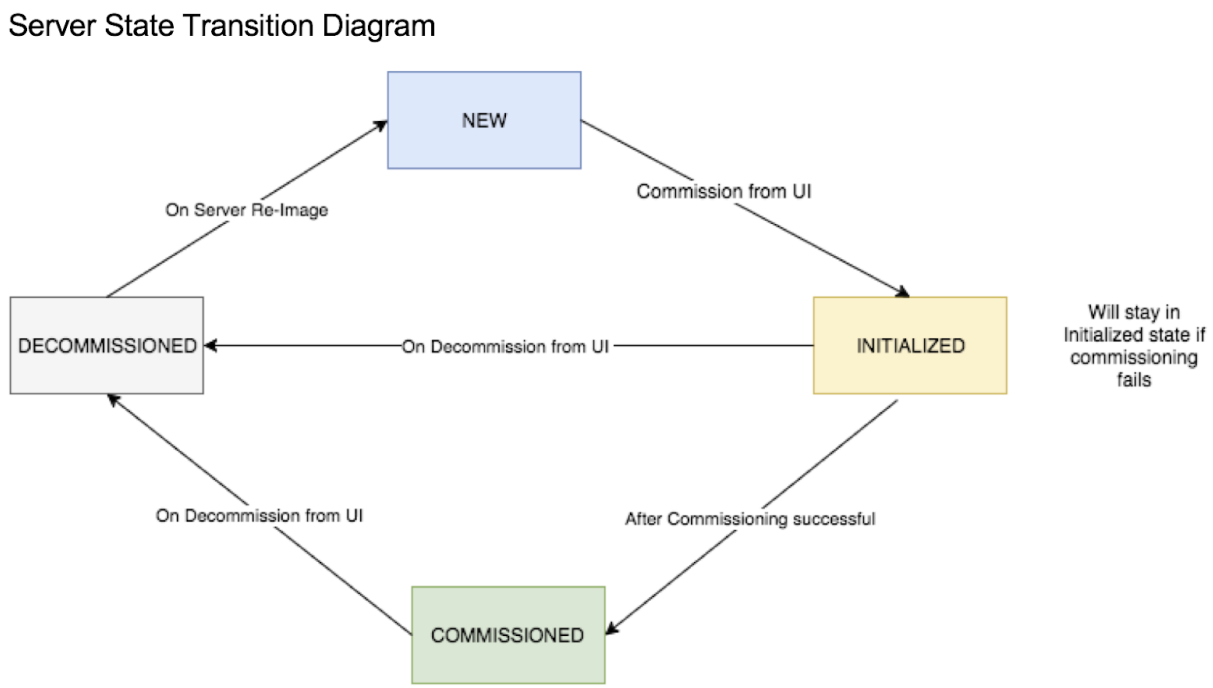


Fig. 11.9.2: Server Maintenance steps

1. **Determine the server that requires maintenance** : This can be done using the server *Serial* number or the *Switchport* the server is connected to , from the *Cluster Status* page. Note the CIMC IP of the server to be replaced. it would be shown in the server box on the *Cluster Status* page
2. **Check for actions for special VMs** : From the server box find out the VMs or instances present on the server and check if any special actions need to be carried out for those VMs. The next section lists out Actions for VMs during server maintenance.
3. **Decommission the server** : Once any pre-decommission actions are performed, use the **Cluster Status** page to decommission the server. Even if the server has failed and appears *Inactive* on the page , we still have to perform all the server maintenance steps. Decommission steps can be performed even if the server is powered off

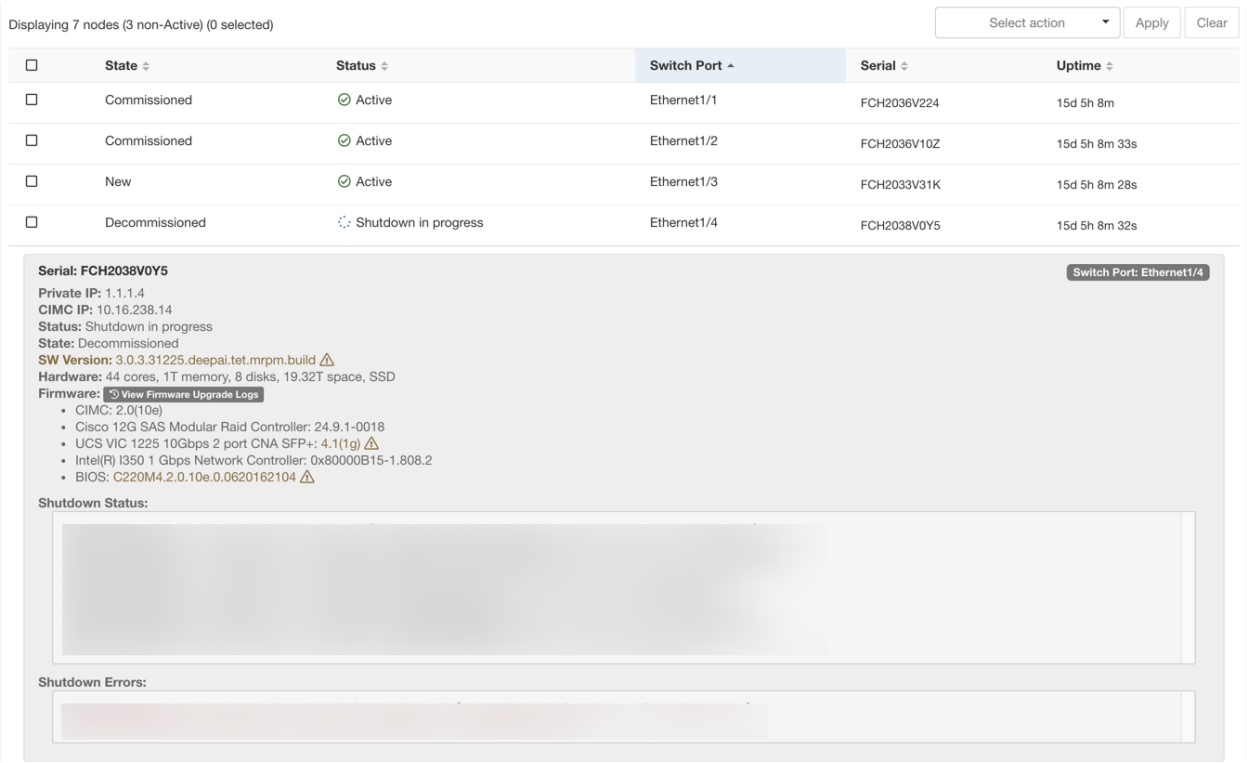


Fig. 11.9.3: Server Maintenance steps

4. **Perform server maintenance** : After the node is marked *Decommissioned* on the **Cluster Status** page perform any post decommission special actions for the VMs. Any component or server replacement can be carried out now. If the entire server is replaced, then change the CIMC IP of the new server to be same as that of the replaced server. The CIMC IP for each server is available on the **Cluster Status** page
5. **Reimage after component replacement** : Reimage the server after the component replacement using the **Cluster Status** page. Reimage takes about 30 mins and requires cimc access to servers. The Server is marked *NEW* after reimage is completed.
6. **Replacing entire server** : If the entire server is replaced, then the server would appear in *NEW* state on the **Cluster Status** page. The s/w version for the server can be seen on the same page. If the s/w version is different from the s/w version of the cluster then reimage the server.

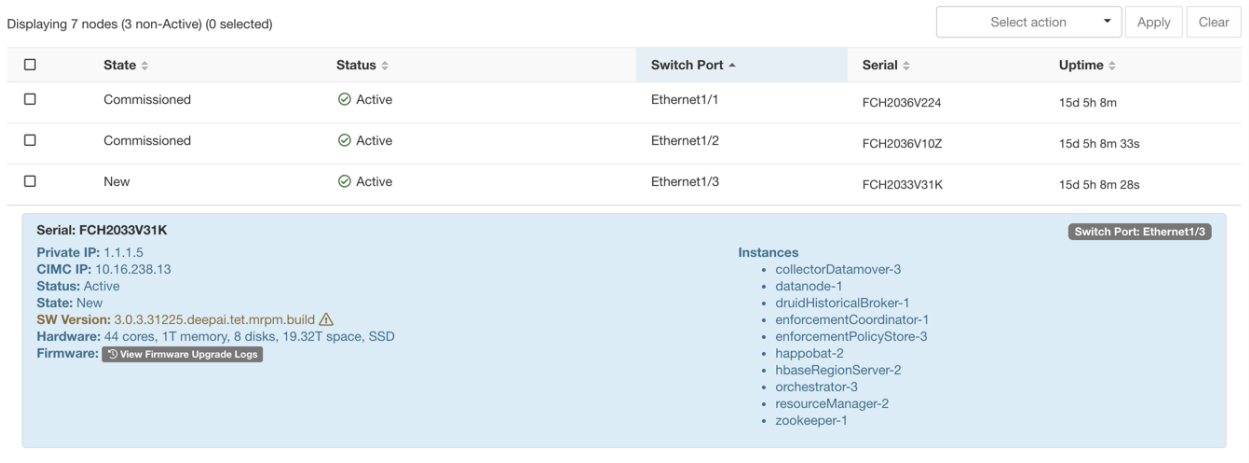


Fig. 11.9.4: Server Maintenance steps

7. **Commission the server** : After the server is marked *NEW* we can kick of the commissioning of the node from the **Cluster Status** page. This step will provision the VMs on the server. Commissioning of a server takes about 45 mins. The server will be marked *Commissioned* after commissioning completes.

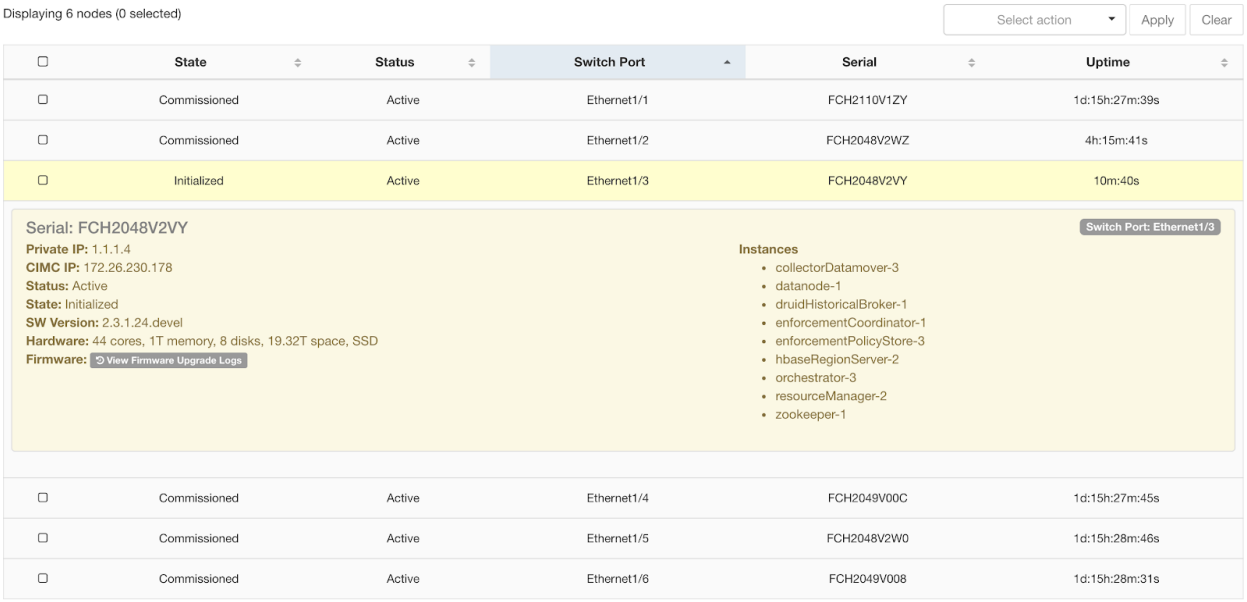


Fig. 11.9.5: Server Maintenance steps

Actions for VMs during server maintenance

Some of the VMs require special actions during the server maintenance procedure. These actions could be pre-decommission, post-decommission or post-commission.

1. **Orchestrator primary** : This is a pre-decommission action. If the server undergoing maintenance has primary orchestrator on it, then POST *orch_stop* command to orchestrator.service.consul from explore page before doing

decommission. This will switch the primary orchestrator.

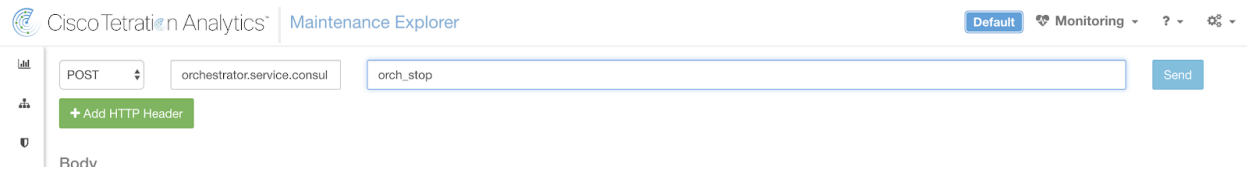


Fig. 11.9.6: Server Maintenance steps

If you try to decommission a server with primary orchestrator, you will see the following error

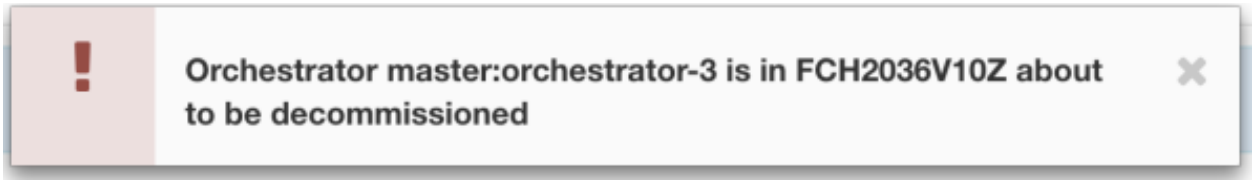


Fig. 11.9.7: Server Maintenance steps

To determine the orchestrator primary run the explore command "primaryorchestrator" on any host.

2. **Namenode** : If the server undergoing maintenance has namenode VM on it, then POST *switch_namenode* on orchestrator.service.consul from explore page after decommission and then POST *switch_namenode* on orchestrator.service.consul after commission. This is both post-decommission and post-commission action.
3. **Secondary namenode** : If the server undergoing maintenance has secondarynamenode VM on it, then POST *switch_secondarynamenode* on orchestrator.service.consul from explore page after decommission and then POST *switch_secondarynamenode* on orchestrator.service.consul after commission. This is both post-decommission and post-commission action.
4. **Resource manager primary** : If the server undergoing maintenance has resourcemanager primary on it, then POST *switch_yarn* on orchestrator.service.consul from explore page. This is both post-decommission and post-commission action.
5. **Datanode** : The cluster tolerates only one Datanode failure at a time. If multiple servers having Datanode VMs need servicing, then do server maintenance on them one at a time. After each server maintenance wait for the chart under Monitoring | hawkeye | hdfs-monitoring | Block Sanity Info, Missing blocks and Under replicated counts to be 0.

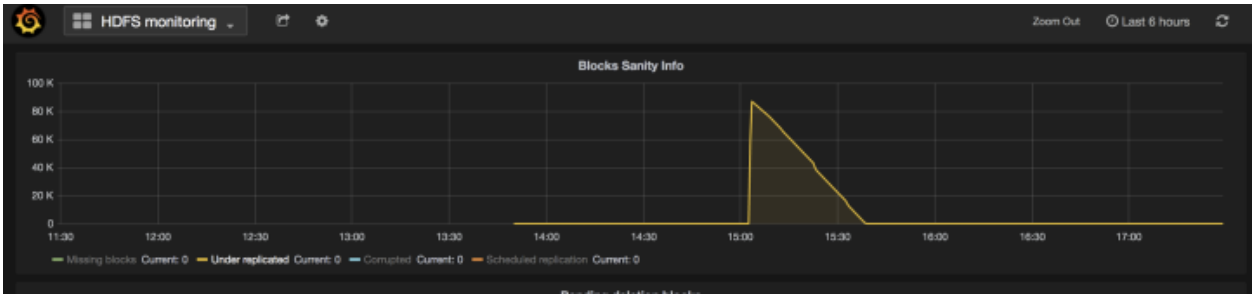


Fig. 11.9.8: Server Maintenance steps

Troubleshooting server maintenance

1. **Logs** : All the server maintenance logs are part of the orchestrator log. The location is /local/logs/tetration/orchestrator/orchestrator.log on orchestrator.service.consul.

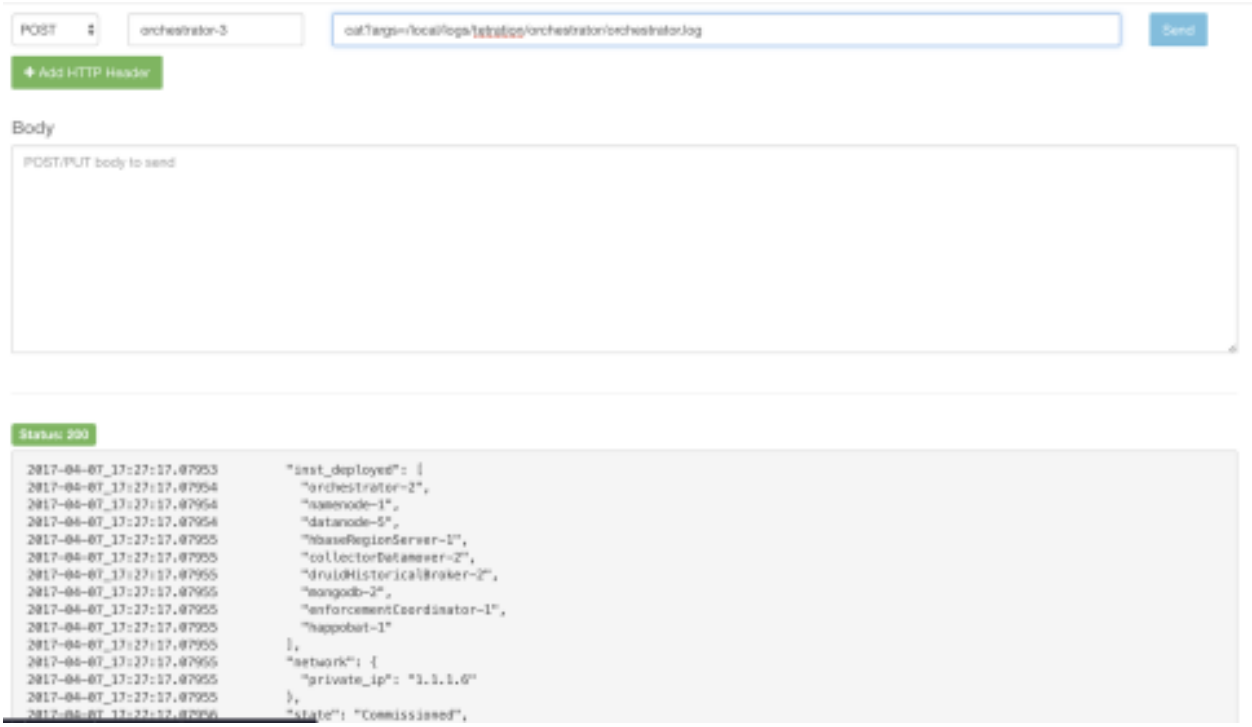


Fig. 11.9.9: Server Maintenance log

2. **Decommission :**
   (a) This step deletes the VMs/instances on the server.
   (b) It then deletes the entry of these instances in backend consul tables.
   (c) This step takes about 5 mins.
   (d) The server will be marked *Decommissioned* once the step completes. **Note**: Decommissioned does not mean the server is powered off. Decommissioning only deletes the tetration content on the server.
   (e) If the server is powered off it will be marked **Inactive**. We can still run Decommission on this server from the cluster status page. But the VMs deletion step will not run since the server is powered off. Make sure this server does not join back the cluster in decommissioned state. It needs to be reimaged and added back to the cluster.
3. **Reimage :**
   (a) This step installs the tetration base OS or Hypervisor OS on the server.
   (b) It also formats the hard drives and installs few tetration libraries on the server.
   (c) Reimage runs a script called **mjolnir** to initiate the server imaging. mjolnir run takes about 5 mins after which the actual imaging begins. Imaging takes about 30 mins. The logs during imaging can be seen only on the console of the server being reimaged. The user can use ta_dev key to check for

additional info regarding the reimage, like /var/log/nginx logs during pxe boot up, /var/log/messages to check for dhcp ip and pxe boot configs.

(d) Reimage requires CIMC connectivity from the orchestrator. The easiest way to check for cimc connectivity is to use explore page and POST ping?args=<cimc ip> from orchestrator.service.consul. **Remember** to change the CIMC IP incase the server is replaced and set the cimc password to the default password

(e) Also cimc network should have been set in site info when the cluster is deployed so that the switches get configured with the correct routes. In case the cluster cimc connectivity is not set correctly you will see the following result in the orchestrator logs.

4. **Commission:**

   (a) Commissioning schedules the VMs on the server and runs playbooks in the VMs to install tetration software

   (b) it takes about 45 mins for commissioning to finish.

   (c) The workflow is similar to deploy or upgrade.

   (d) The Logs will indicate any failures during commissioning

   (e) The server on the cluster status page will be marked initialized during commissioning and marked commissioned only after the step completes

11.9.1 Baremetal Exclude (bmexclude)

If a hardware failure is detected upon restart of a cluster after power shutdown, currently the cluster gets stuck in a state where we can neither run Reboot workflow to get services stable nor run Commission workflow as down services result in commissioning failure. This feature is expected to help in such scenarios by allowing user to reboot (upgrade) with a bad hardware, after which regular RMA process for the failed baremetal can be performed.

User is expected to use a post to explore endpoint with serial of the baremetal to be excluded.

1. Action: POST
2. Host: orchestrator.service.consul
3. Endpoint: exclude_bms?method=POST
4. Body: {"baremetal": ["BMSERIAL"]}

Orchestrator performs few checks to determine if the exclusion is feasible. In which case, it will setup few consul keys and return success message indicating which baremetal and VMs will be excluded in the next reboot/upgrade workflow. If the baremetals include certain vms, they can't be excluded as described in the Limitation section below, the explore endpoint will reply back with the message indicating why the exclusion is not possible. After successful post on the explore endpoint, user can initiate reboot/upgrade through main UI and proceed with reboot as usual. At the end of the upgrade, we remove the exclude bm list. If there is a need to run upgrade/reboot again with exclude BMs, users are expected to post to the bmexclude explore endpoint again.

Limitations We don't allow following VMs to be excluded currently. 1. namenode 2. secondaryNamenode 3. mongodb 4. mongodbArbiter

11.10 Disk Maintenance

Disk Maintenance involves replacement of any faulty Hard Disk(s) from the server(s). Orchestrator monitors the health of the disks as reported by bmmgr on every server in the cluster. If there are any faulty disk detected, the **Cluster Status** page will indicate this via a banner. This banner will show the number of disks that are in UNHEALTHY state. Clicking on *here* on that banner will lead user to a disk replacement wizard where all the steps for the disk maintenance will be performed. Like the **Cluster Status** page, the disk replacement page can be accessed by all users but the actions can be carried out by **Customer Support** users only.

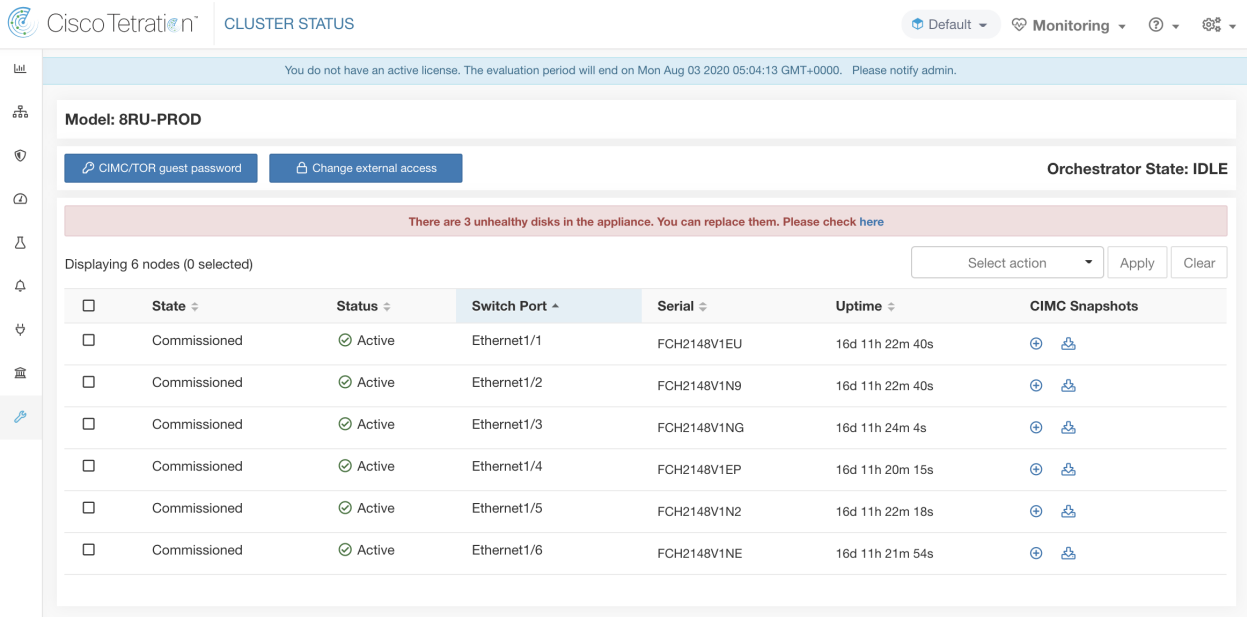


Fig. 11.10.1: Faulty Disk Banner

11.10.1 Disk Replacement Wizard

The landing page of Disk Replacement Wizard shows the details of the failed disks. These details include the size, the type, the make and the model for every disk that needs replacement. It also shows the slot id and lists all the vms that use each of these disks. Before the user starts the replacement process, they should have the replacement disks available.

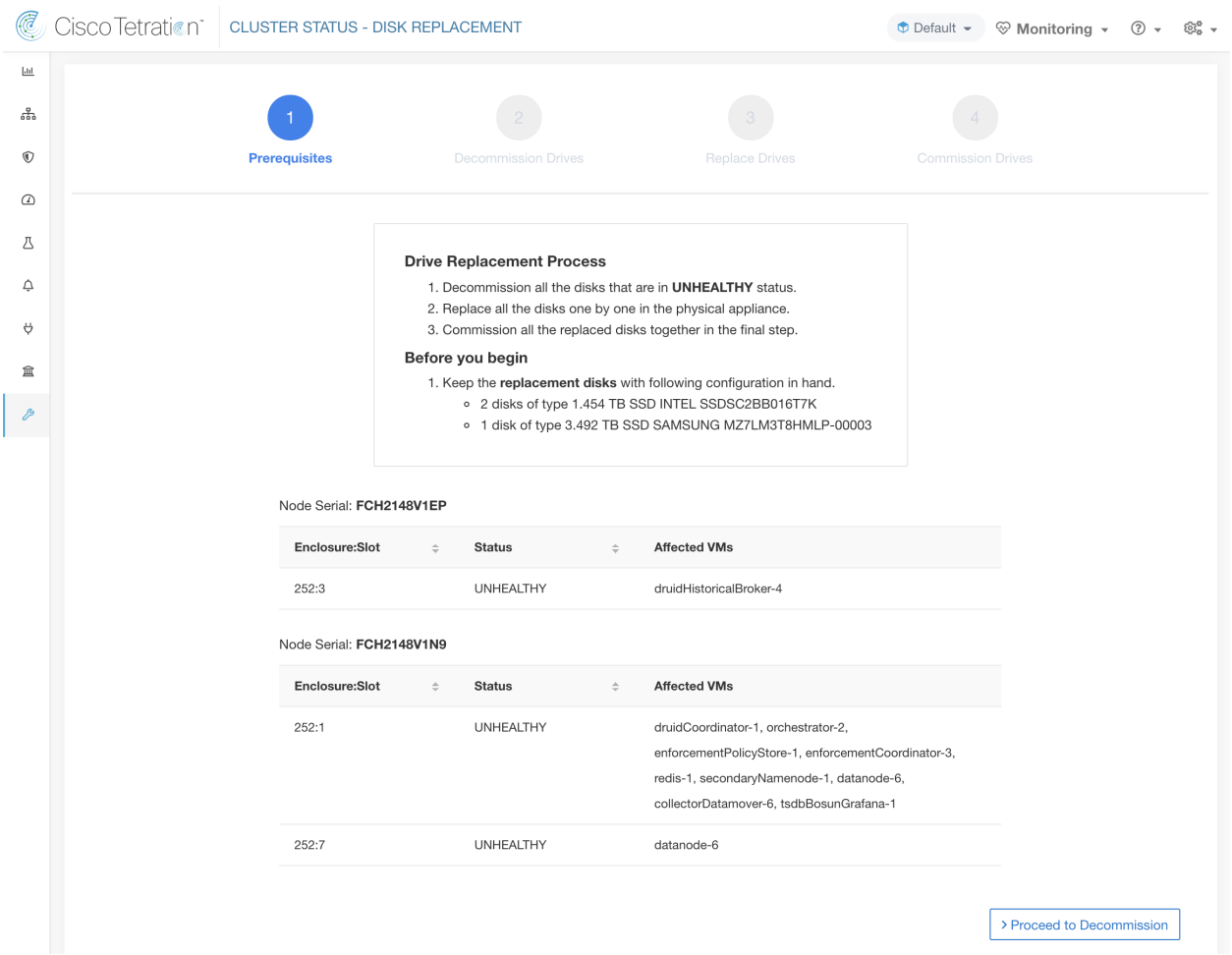


Fig. 11.10.1.1: Disk Replacement Wizard

11.10.2 Disk Status Transitions

In the cluster, Hard Disks can have 6 states. HEALTHY, UNHEALTHY, UNUSED, REPLACED, NEW and INITIALIZED. Upon deployment/upgrade, the status of every disk in the cluster is HEALTHY. Based of various error detection the status of one or more disk can become UNHEALTHY.

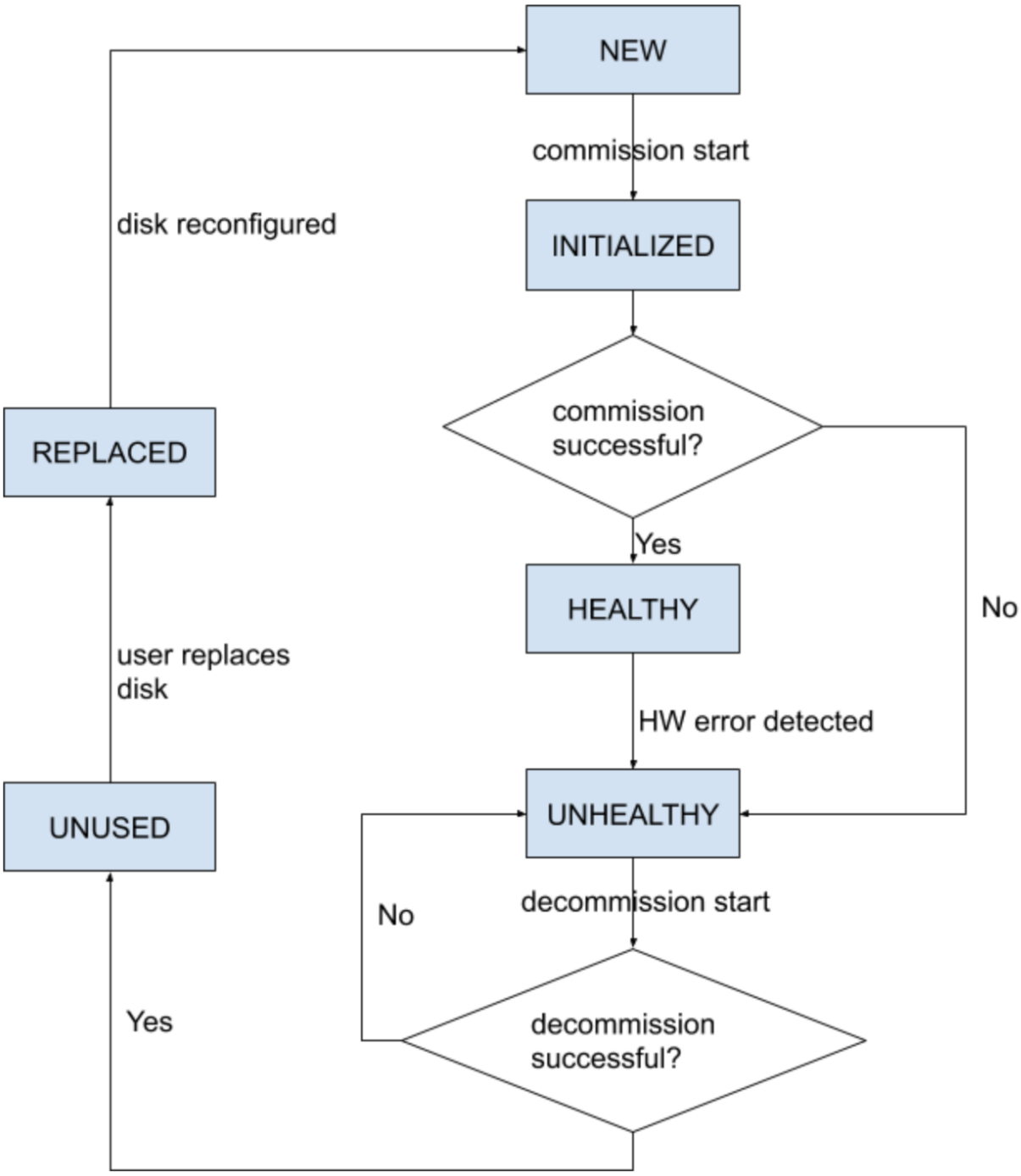


Fig. 11.10.2.1: Disk Status Transitions

The first step of the disk replacement process is decommission where all the vms that use these disks are removed from the cluster. The status of disks that are decommissioned become UNUSED. After decommission, the replacement disks should be inserted in their appropriate slots. Users will confirm that the disks are replaced, which will be the backend's signal to reconfigure the newly instered disks. This will change the status to REPLACED and after the next hardware scan these replaced disks' status will change to NEW. This transition can take 2-3 minutes..

Once all the disks have been replaced and reconfigured, user can proceed to commissioning which will deploy all the vms that were removed as part of decommission process. The start of commission will change the disk status to INITIALIZED. A successfull commission will make all disks' status HEALTHY. A failure in this step will make the status UNHEALTHY again so that we start the recovery from decommission again.

11.10.3 Requirement PreChecks

Before any of the decommission or commission step can take place, a requirement precheck must be performed. Backend performs various checks all of which must pass before user can proceed with the decommission or commission step. Any failed checks will be reported on the disk replacement wizard with the failure detail and suggested corrective action, which must be taken before the needed step can proceed

Example of such pre chesk are: namenode and secondaryNamenode can't be decommissioned together. only one datanode can be decommissioned at one time. namenode is healthy before commissioning.

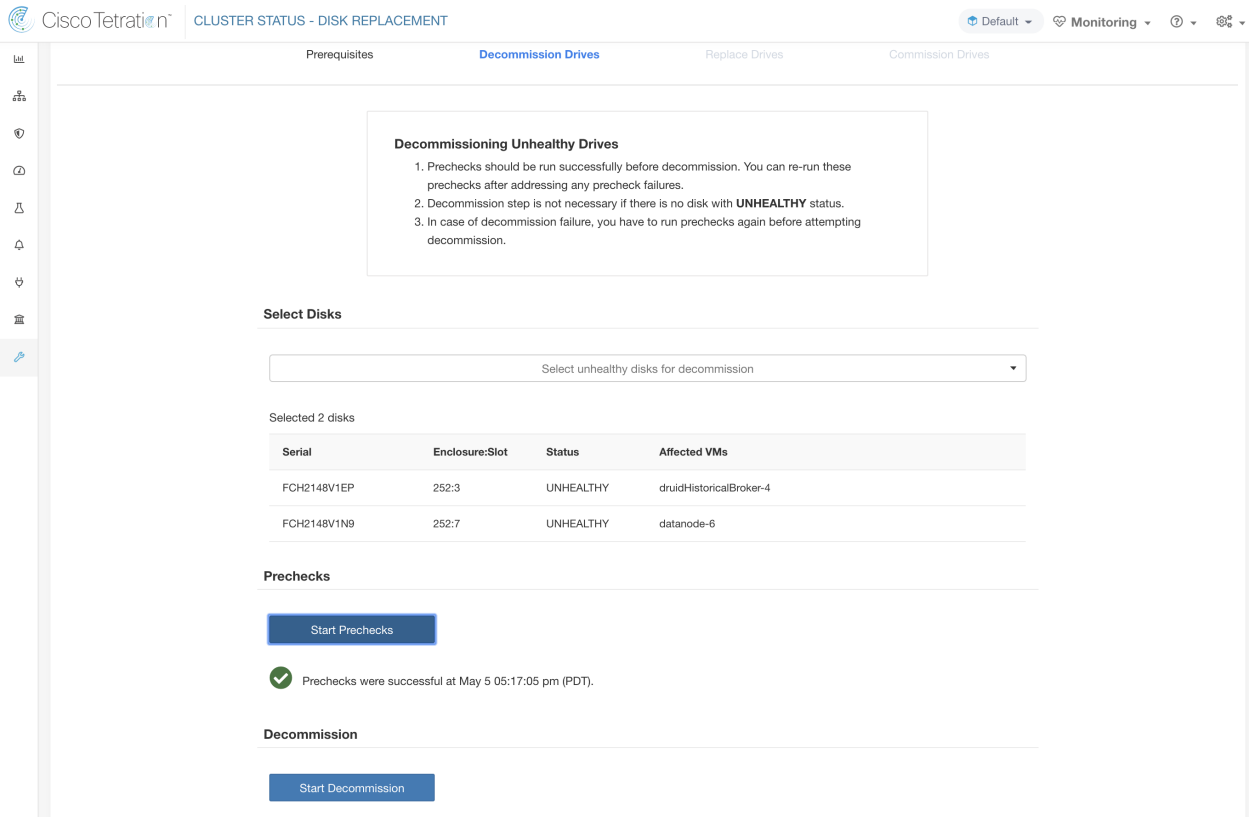


Fig. 11.10.3.1: Disk Replacement PreChecks

User can select any set of failed disks to be decommissioned together and start the decommission precheck. Changing the set of failed disk will require a rerun of the precheck. Same prechecks are checked again before the task (decommission/commission) starts to ensure that there are no new precheck failure between last precheck run and the start of the decommission task

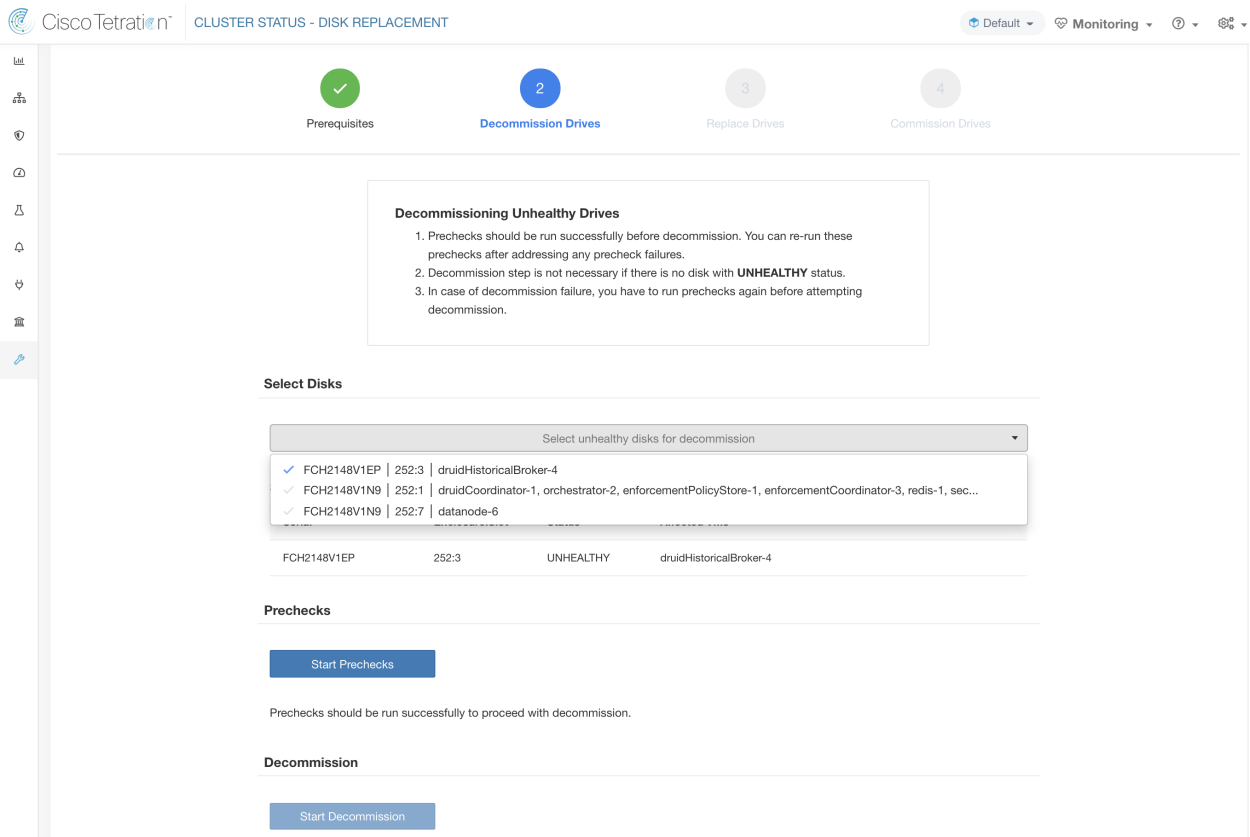


Fig. 11.10.3.2: Select one or all UNHEALTHY disks to decommission

Upon any failed precheck, a detailed message can be seen by clicking on the failure message as well as a suggested action will be shown in a pop-over when pointer hovers over the red cross button.

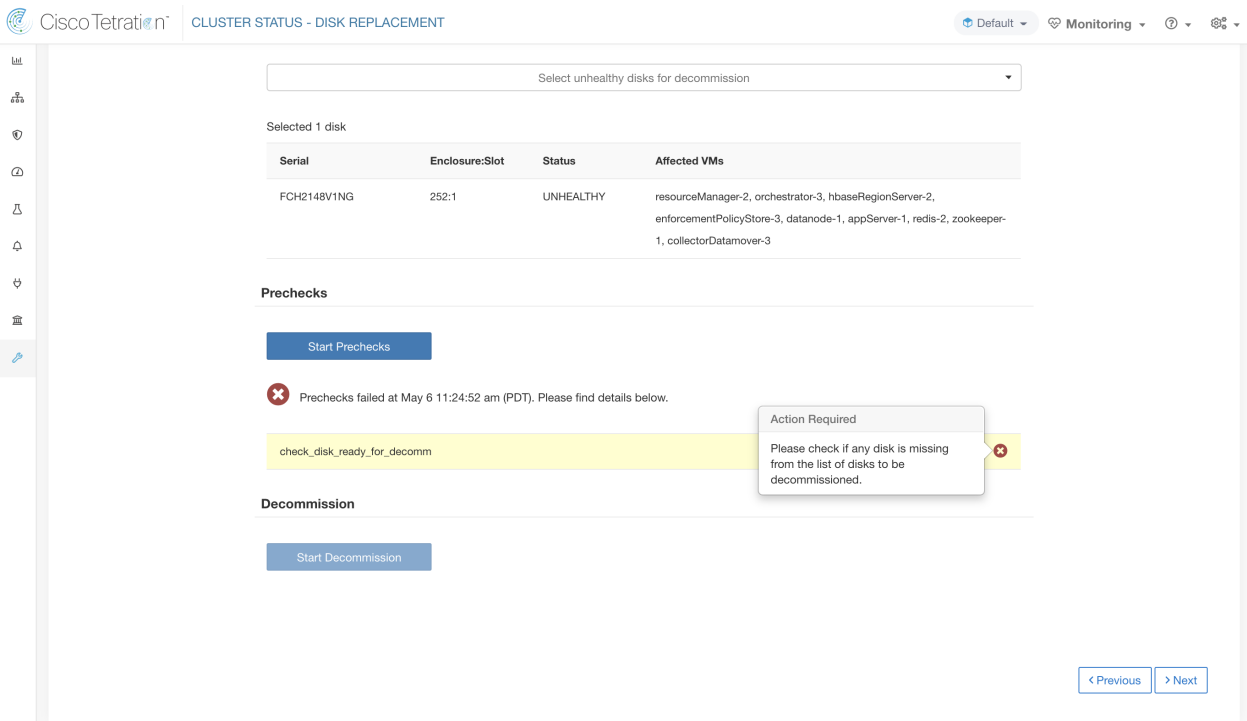


Fig. 11.10.3.3: Suggested action in pop-over for failed precheck

11.10.4 Decommission Disk

Once the prechecks passes, user can proceed to decommission disk. The progress of decommission will be shown on the disk replacement wizard. Once progress of decommission reach 100%, all the decommissioned disk status changes to UNUSED.

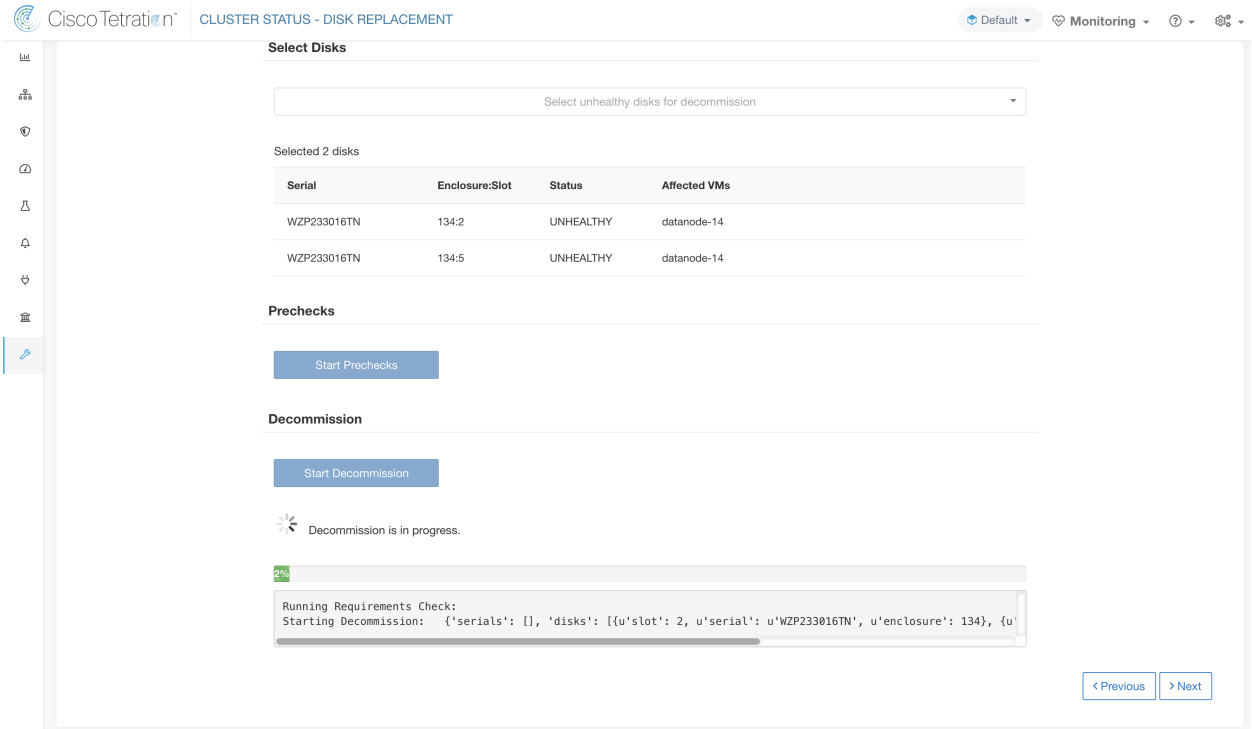


Fig. 11.10.4.1: Monitoring disk decommission progress

11.10.5 Replace Disk

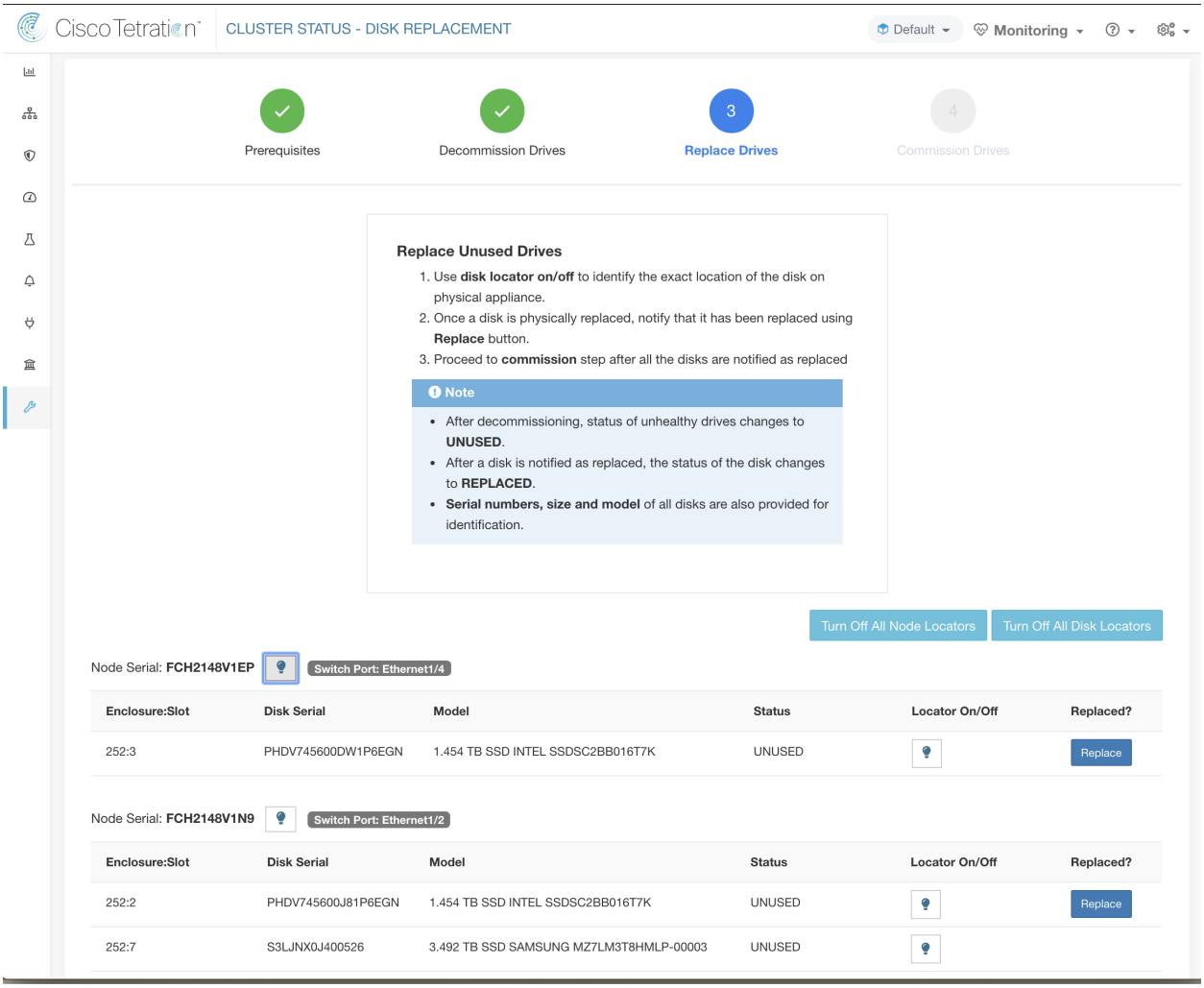


Fig. 11.10.5.1: Reconfigure newly added disks

After disk decommssion, user is expected to physically replace the disks. To assist in this process, we have added disk and server locator LED access on the replace page. There are buttons to switch off all the server and disks locator LEDs to take care any other process that might have left the locators on.

Disks can be physically replaced in any order but they must be reconfiured in smallest to largest slot numbers for a given server. This order is enforce through both UI and the backend. UI will have replace button active for disk with the lowest slot number with status UNUSED.

11.10.6 Commission Disk

When all the disks are replaced, we proceed to commission. Like decommission, we need to run a set of prechecks before we can continue to commission.

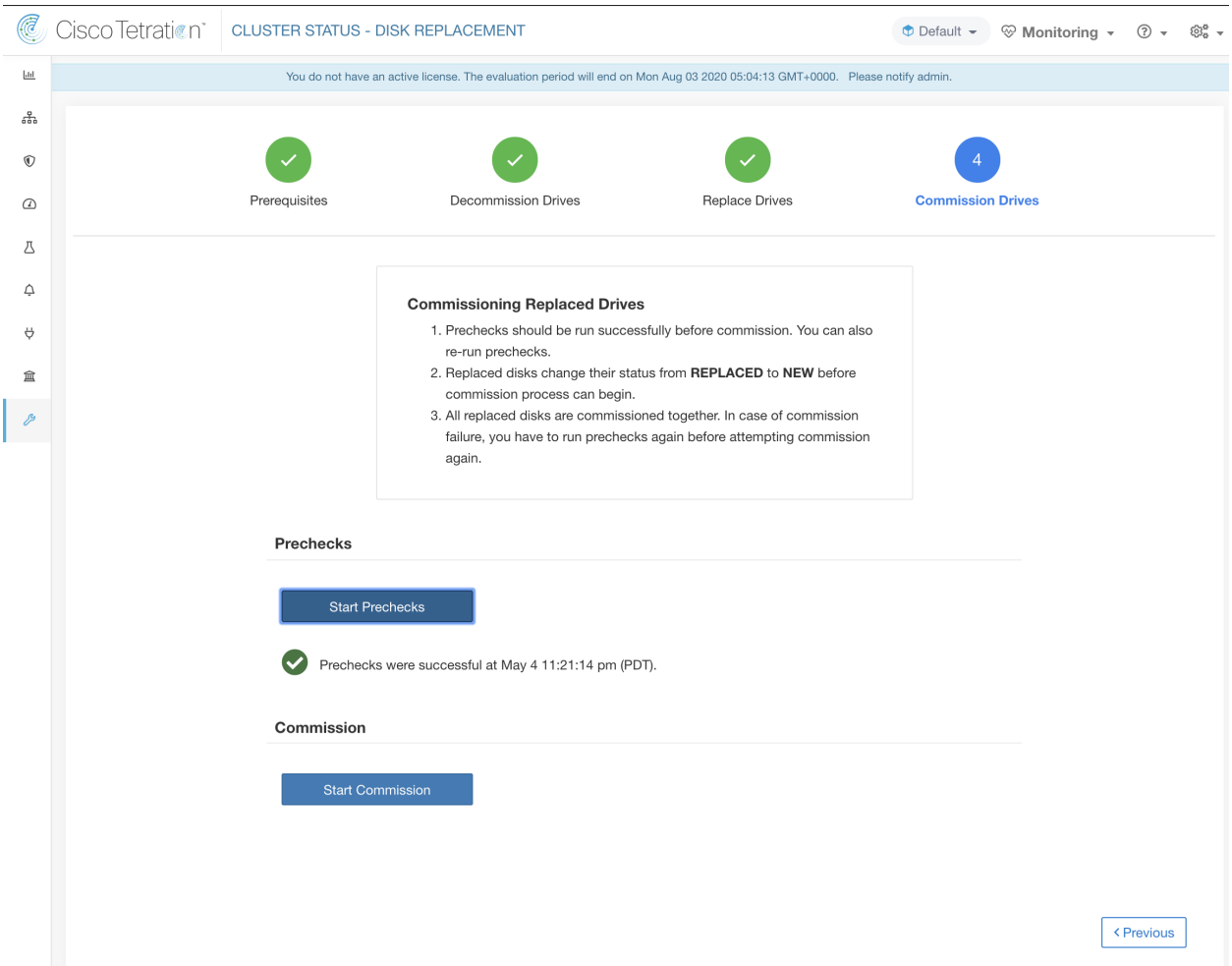


Fig. 11.10.6.1: Prechecks before commission

Progress of commission is monitored on the disk commission page. At the end successful commission, the status all disks change to HEALTHY.

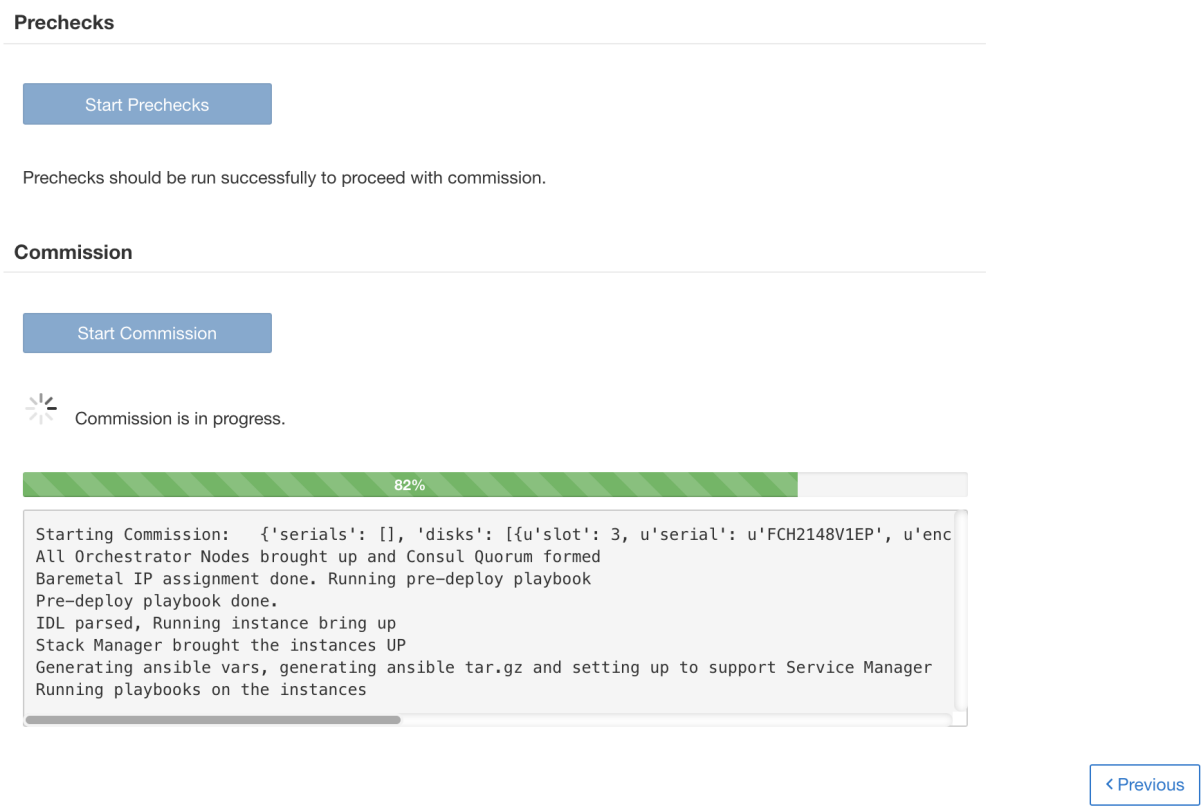


Fig. 11.10.6.2: Commission progress

Recovery from failure during commission

A failure after vms have been redeployed, can be recovered via resume. In case of such failures, a *Resume Commission* button will appear on the disk commission page, which can be clicked to continue commission by restarting the post deploy playbooks.

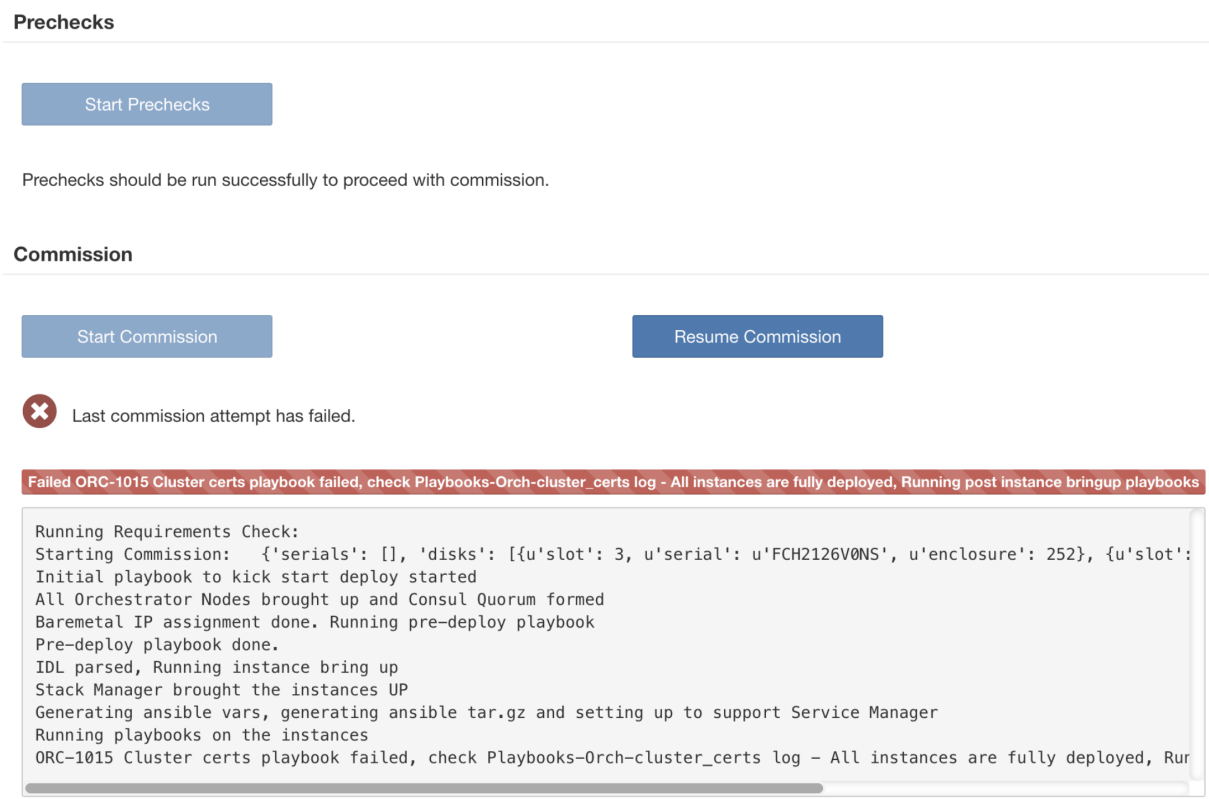


Fig. 11.10.6.3: Resume commission

In case of any failure before the vms have been redeployed, the disks that were being commissioned will have their status changed to UNHEALTHY. That will require us to restart the replacement process from the decommission of UNHEALTHY disks.

Additional disk failures during commission

In case of any other disks than the ones that are being replaced fails while disk commission is in progress, notice of this failure will be displayed on the disk replacement wizard after the ongoing commission process finishes, either in success or failure.

In cases of resumable failures, user will have two options in what next steps to take.

1. They can try to resume and complete current commission and perform the disk replacement process for the new failures later.

2. Alternatively, they can start decommission of newly failed disk and perform commission of all the disks together.

This second path will be the only path availble in cases of non-resumable failures. If the post deploy failure is caused due to the newly failed disks, the second path will again be only way forward, even though we will have resume button available.

11.10.7 Troubleshooting

Logs

1. All the disk commission/decommsion logs are part of orchestrator logs. Starting debug point should be /local/logs/tetration/orchestrator/orchestrator.log on orchestrator.service.consul.

2. Details of any failure during disk replace/reconfigure action can be found on the bmmgr log on the server in consideration. The log location on the server would be /local/logs/tetration/bmmgr/bmmgr.log

Limitations

1. Disk containing server's root volumes can't be replaced using this procedure. Such disk failure must be corrected using server maintainence process.

2. Disk commissioning can happen only when all servers are active and in commissioned state. See special handling section below to that describes how to proceed in the cases where a combination of disk and server replacement is needed.

11.10.8 Special handling

Disk and Server Replacement together

In the case of failure scenarios where a disk and a server needs to be commissioned together, user is expected to decommission and replace all the disks that can be decommissioned. Commission of those disk would be prevented by the precheck that ensure that

1. All non healthy disks have the status of NEW
2. All servers are in the *Commissioned* state with status *Active*

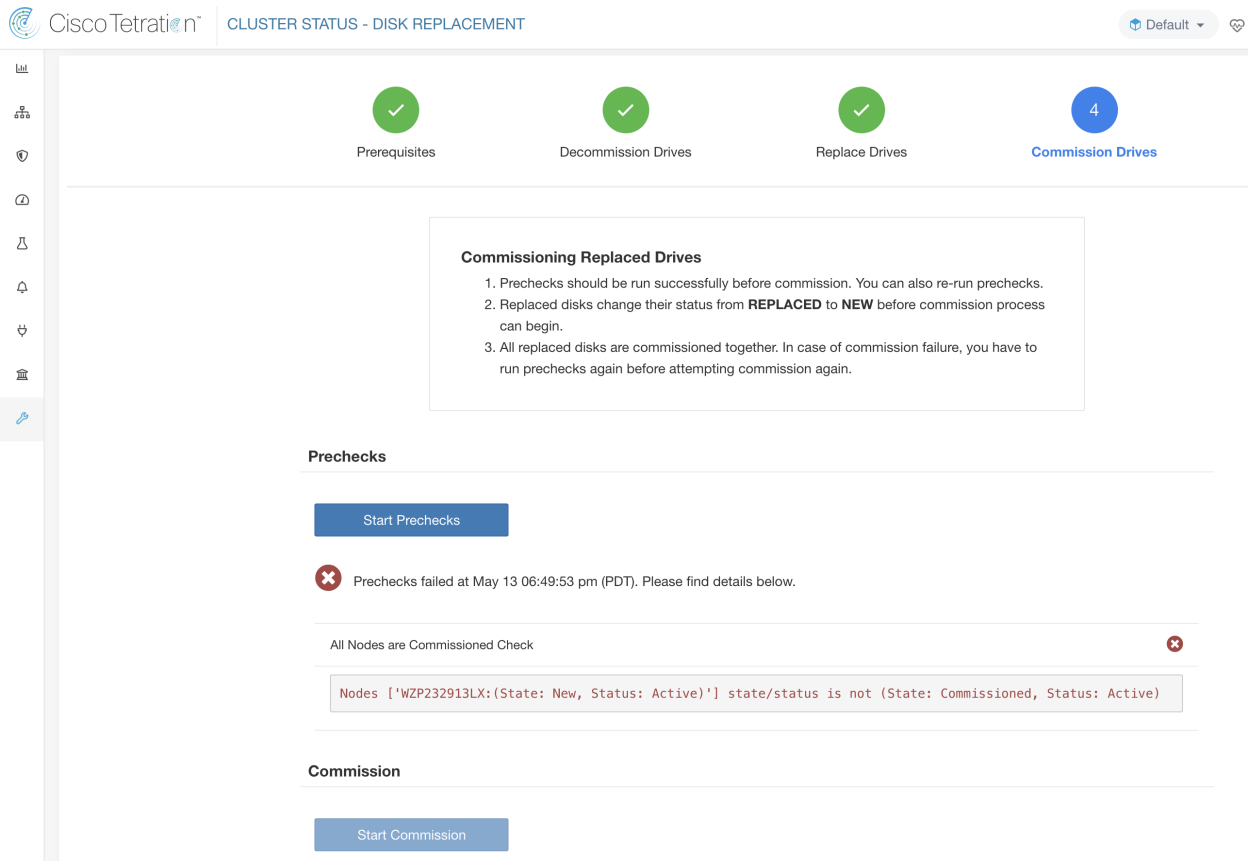


Fig. 11.10.8.1: Ensure the all servers are commissioned and active before disk commission

Once all the UNHEALTHY disks are in the NEW state, the faulty server is expected to be decommission/reimaged/commission back using server maintainence procedure.

Now server commission will be prevented if there are any disk without status HEALTHY or NEW. A successful server commission will also make the status of all disks HEALTHY.

Fig. 11.10.8.2: Ensure the all faulty disks are in NEW state before server commission

11.11 Cluster Maintenance - Cluster Shutdown and Reboot

In this section, we discuss two maintenance operations that affect the entire cluster.

1. Cluster Shutdown
2. Cluster Reboot

11.11.1 Cluster Shutdown

Cluster shutdown stops all running Tetration processes, and powers down all individual nodes.

Please follow the steps below for executing the shutdown.

11.11.1.1 Initiating Shutdown

Go to the upgrade page under maintenance as shown below.

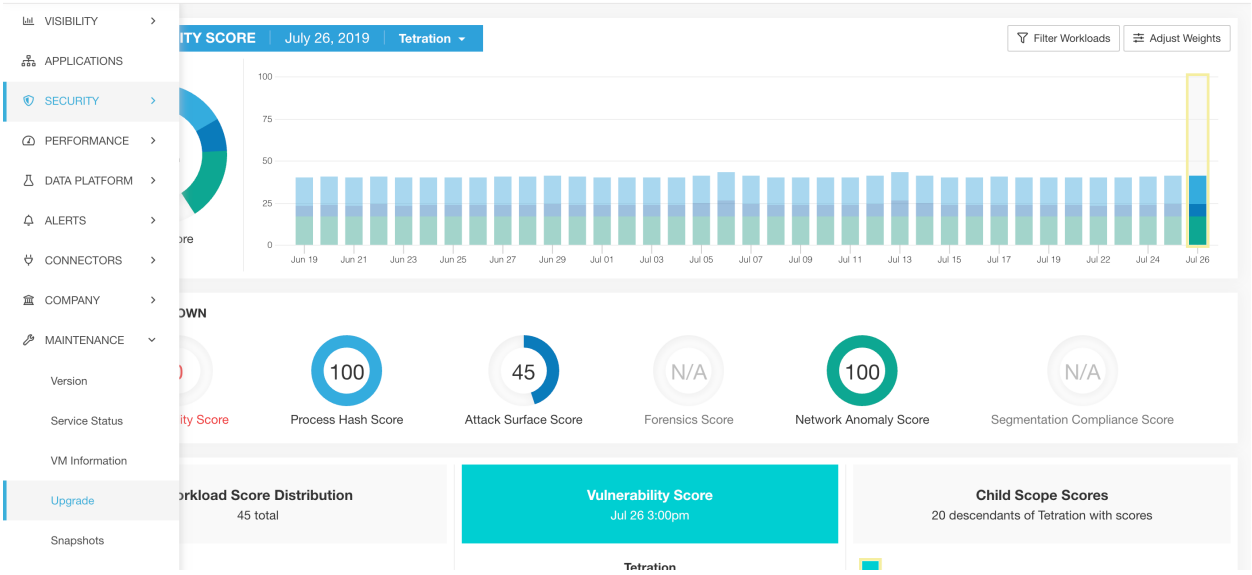


Fig. 11.11.1.1.1: Upgrade Tab

Next select the shutdown radio button on the shutdown/reboot tab from the upgrade page and click on Send Shutdown Link button.

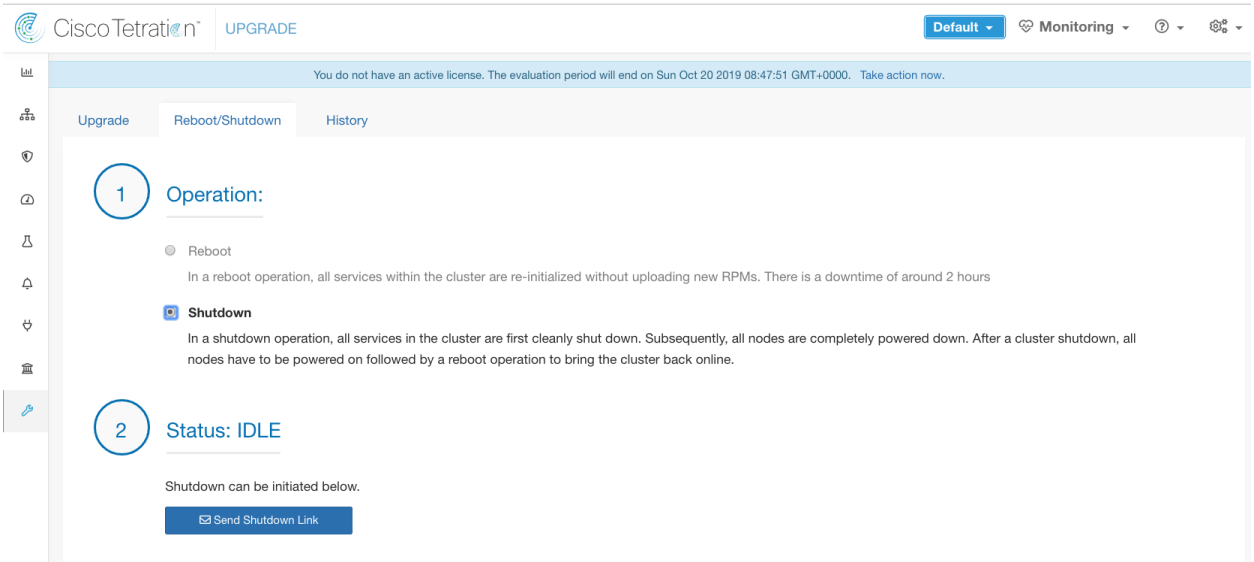


Fig. 11.11.1.1.2: Shutdown Radio Button

This sends the shutdown link in an email as shown below. The shutdown link is delivered to the email address of the user requesting the link.

Hello Site Admin!

We received a request that you intend to shutdown the cluster "98". You can do this through the link below.

Shutdown 98 (For best results, please use Google Chrome)

The above link expires by Jul 22 08:34:30 pm (PDT).

If you didn't request this, please ignore this email.

Shutdown will not be triggered until you actually click the above link.

Fig. 11.11.1.1.3: Shutdown email

Clicking on the link will take you to shutdown page. This has the shutdown button which can be clicked to initiate the shutdown. Clicking the button will bring up a confirmation box. This is an added check to make sure that shutdown initiation is intentional. Note that after clicking on this, the shutdown process starts on the cluster and cannot be cancelled midway.

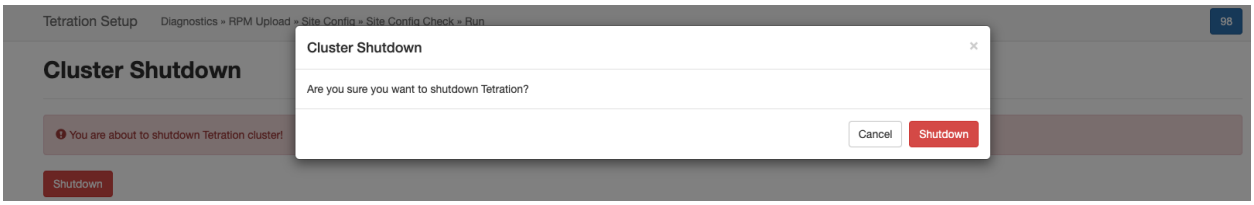


Fig. 11.11.1.1.4: Shutdown Dialog Box

11.11.1.2 Shutdown Progress

Once the shutdown starts, the page transitions to show a progress bar tracking the progress of the shutdown. This is similar to the upgrade progress page.

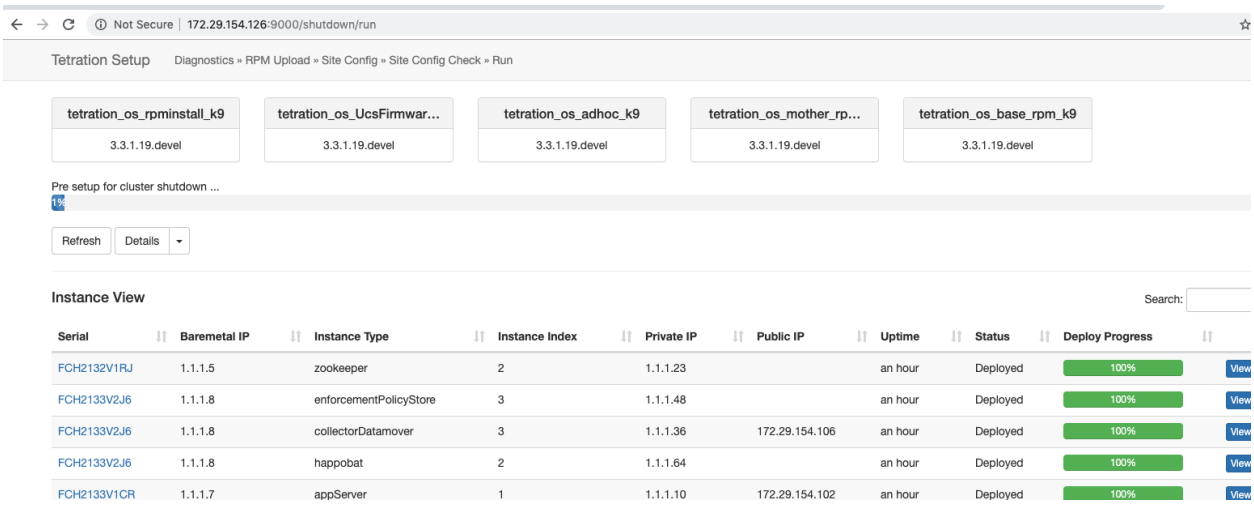


Fig. 11.11.1.2.1: Shutdown Progress

If an error occurs in the initial shutdown pre-checks, progress bar will turn red and a resume button will show up which can be clicked to restart shutdown after fixing the errors.

After pre-checks are complete, VMs are stopped. As the VMs progressively stop, their progress is shown in the lower portion of the page. This page is similar to the VM stop under upgrades - please refer to the upgrades section for more information on each field being displayed. Please note that stopping of VMs can take upto 30 minutes.

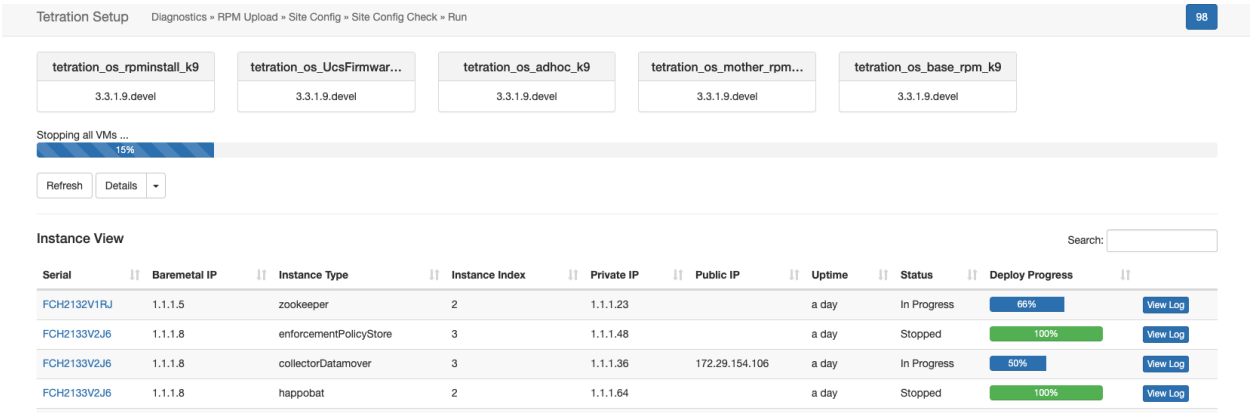


Fig. 11.11.1.2.2: VM stop

Eventually, as the cluster is completely ready to be shutdown, the progress bar will go to a 100% and indicate the time after which it is safe to poweroff the cluster. This is highlighted in the screenshot below.

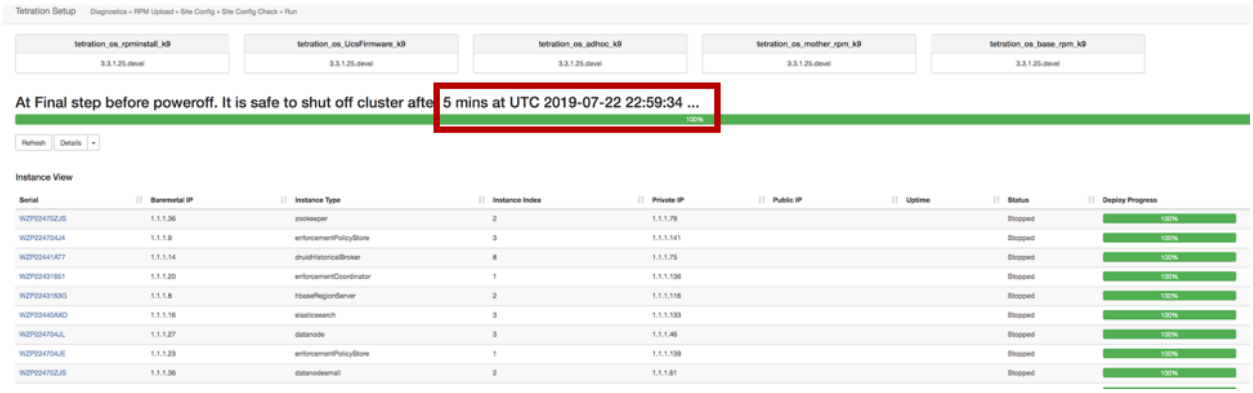


Fig. 11.11.1.2.3: Shutdown 100 Percent

Note: The user should poweroff the cluster only AFTER the time displayed on the progress bar.

11.11.2 Cluster Reboot

To recover the cluster after shutdown, user has to power on the baremetals When all the individual bare metals are up, the UI will become accessible again. After logging into the cluster, cluster reboot MUST be initiated to make the cluster fully operational again.

Note: The user must reboot the cluster after a shutdown to make it fully operational again.

11.11.2.1 Initiating Reboot

Go to the upgrade page under maintenance as shown below.

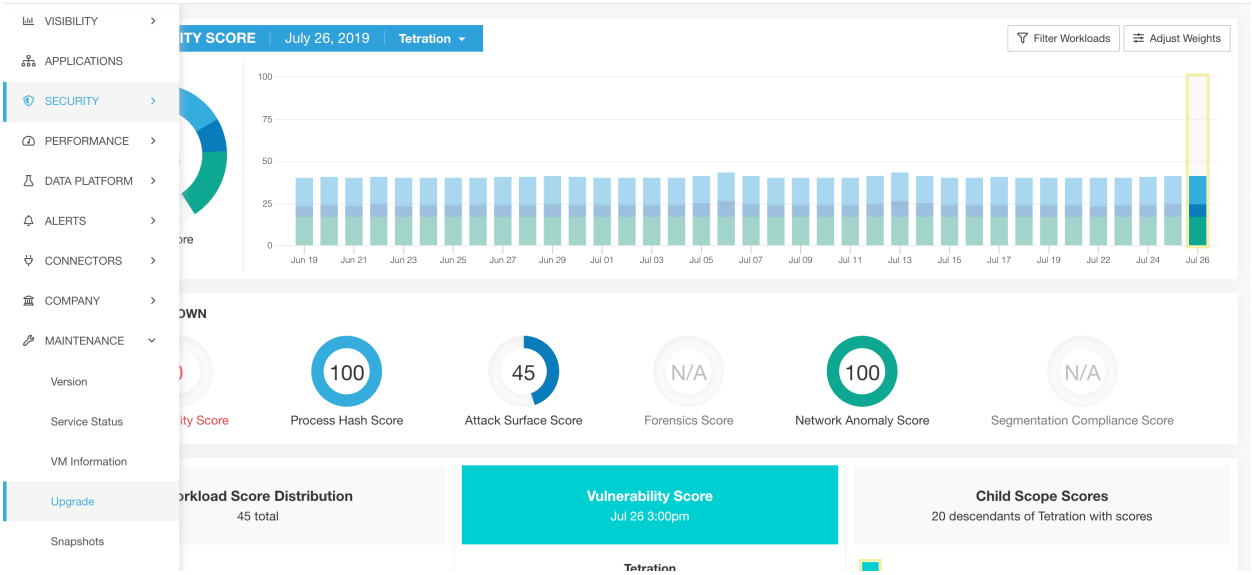


Fig. 11.11.2.1.1: Upgrade Tab

Next select the reboot radio button on the shutdown/reboot tab from the upgrade page and click on Send Reboot Link button.

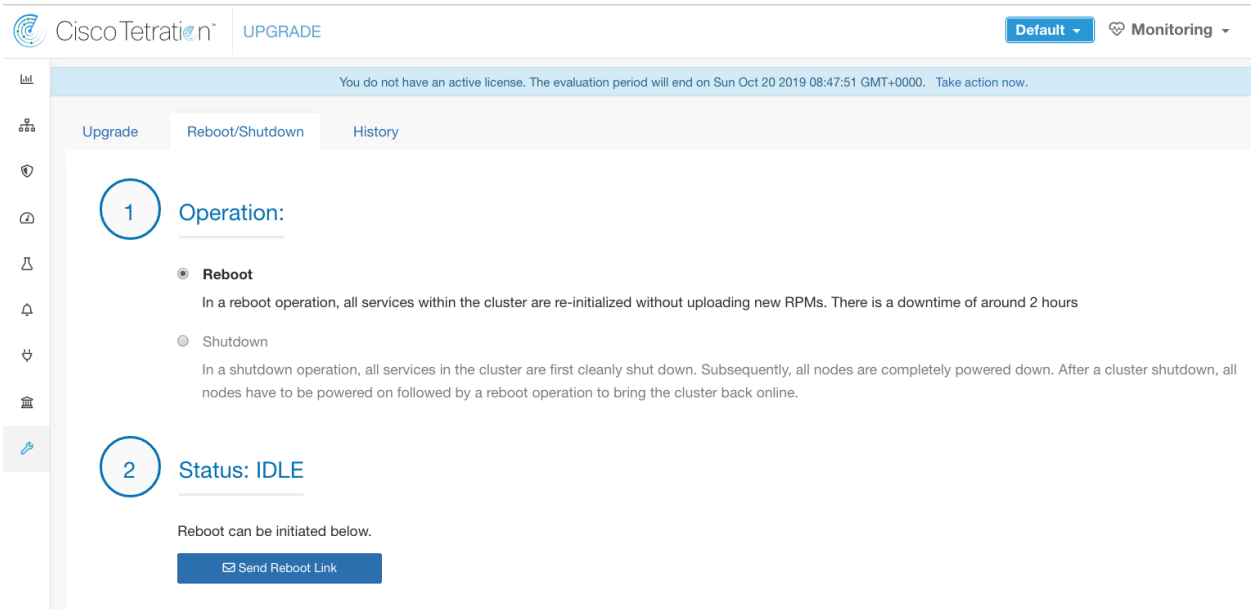


Fig. 11.11.2.1.2: Reboot Radio button

The reboot link is delivered to the email address of the user requesting the link.

Tetration services reboot performs a restricted upgrade operation. After clicking the reboot link in the email, the user is taken to the setup UI where the reboot can be initiated.

From here on, the progress is same as upgrades. Please refer to upgrade section for more details.

11.11.2.2 History of Shutdown and Reboot

The history of shutdown and reboots is shown under the history tab on the upgrade page as shown below.

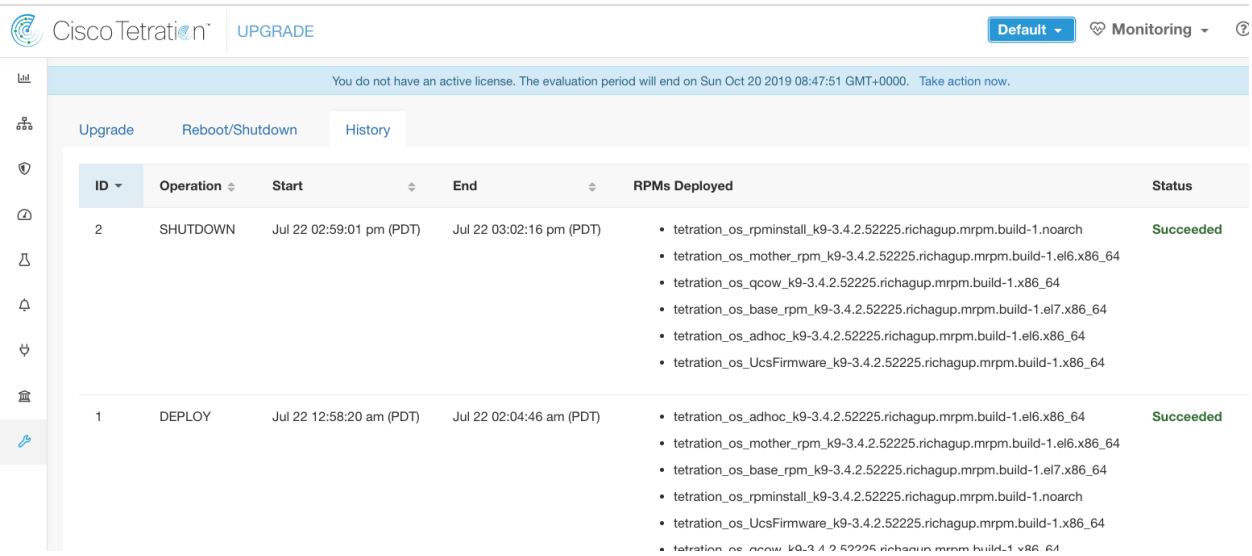


Fig. 11.11.2.2.1: History Table

CHAPTER

TWELVE

MONITORING

The **Monitoring** drop-down menu options can be accessed using the top-level menu item. The options available in the menu vary depending on your role.

12.1 Agents Overview

The page shows counts of all monitored agents in a cluster based on the currently selected root scope.

Note: Total Inventory count is the summation of all inventory observed on the network after applying collection rules.

12.1.1 Agents Overview

This page is only available for users that have **Site Admin** and **Customer Support** roles. **Scope owners** can see Inventory, Deep Visibility Agents, Enforcement Agents and Universal Visibility agents.

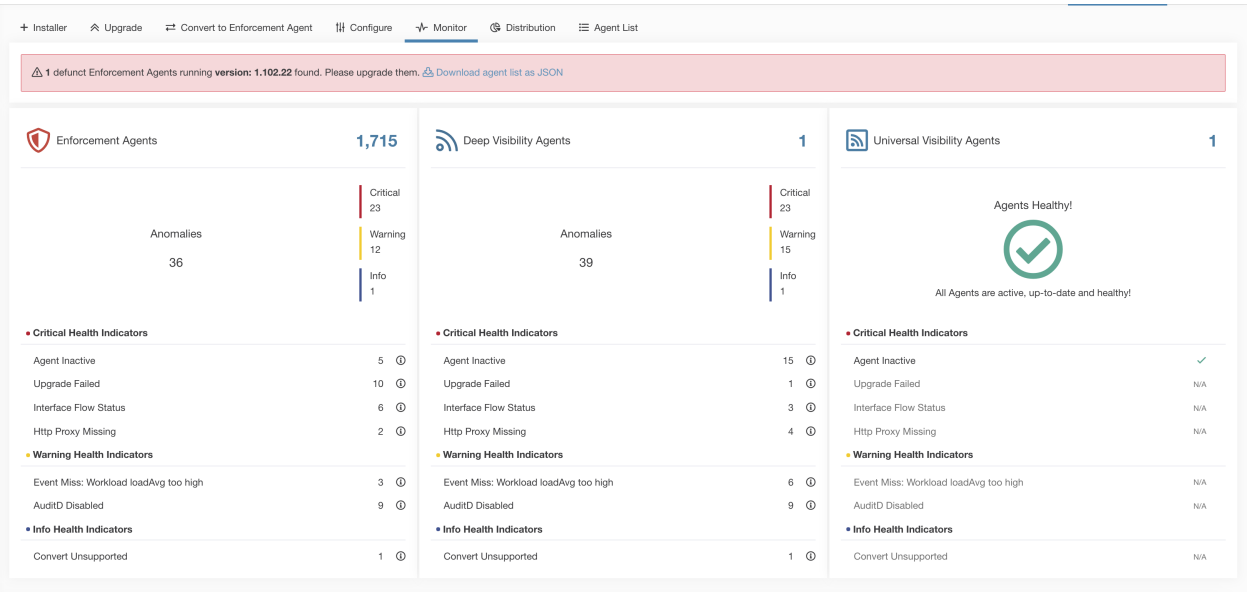


Fig. 12.1.1.1: Total Number of Installed Agents

The following table shows the differences between each agent type.

Agent Type	Description
Deep Visibility	Provides highest fidelity in terms of time series flow data, processes running on a host. Most Linux and Windows platforms are supported. See *Deploying Software Agents*
Enforcement	Provides all capabilities available in Deep Visibility Agents. In addition, Enforcement agents have capability to set firewall rules on the installed host.
Universal Visibility	Provides flexibility to be installed on almost any compute platform. Hosts that have an Universal Visibility Agent installed allows conversation analysis via ADM.
AnyConnect	Provides time series flow data on endpoints running AnyConnect Secure Mobility Agent with Network Visibility Module (NVM) without requiring any Tetration agent installation. IPFIX records generated by NVM are sent to Tetration AnyConnect Proxy connector. Windows, Mac, and certain smartphone platforms are supported.
ISE	Provides metadata about endpoints registered with Cisco ISE. Through ISE pxGrid, ISE connector collects the metadata, registers the ISE endpoints on Tetration as ISE agents pushes labels based on the attributes fetched from ISE appliance and LDAP attributes for the users logged in to the endpoints.
Hardware Switch	Provides the highest throughput flow analysis without requiring any per-host agent installation. Requires to be installed on Cisco N9K switch operating system.

The following table provides a brief summary of various appliance agents provided by Tetration.

Appliance Agents	Description
SPAN	Provides the flow analysis without requiring any per-host agent installation. It runs in the Tetration ERSPAN VM appliance. It consumes ERSPAN packets sourced by any Cisco switch.

Note: Appliance agents such as NetFlow, NetScaler, F5, AWS and AnyConnect Proxy are now supported as connectors. For more information on connectors, please refer to *What are Connectors*.

Any non-zero agent type button allows further drill-down into the distribution of each agent type.

12.1.1.1 Software Agents

All of the following charts are available for both Deep Visibility and Enforcement Agent types but only a subset is available for Universal Agent.

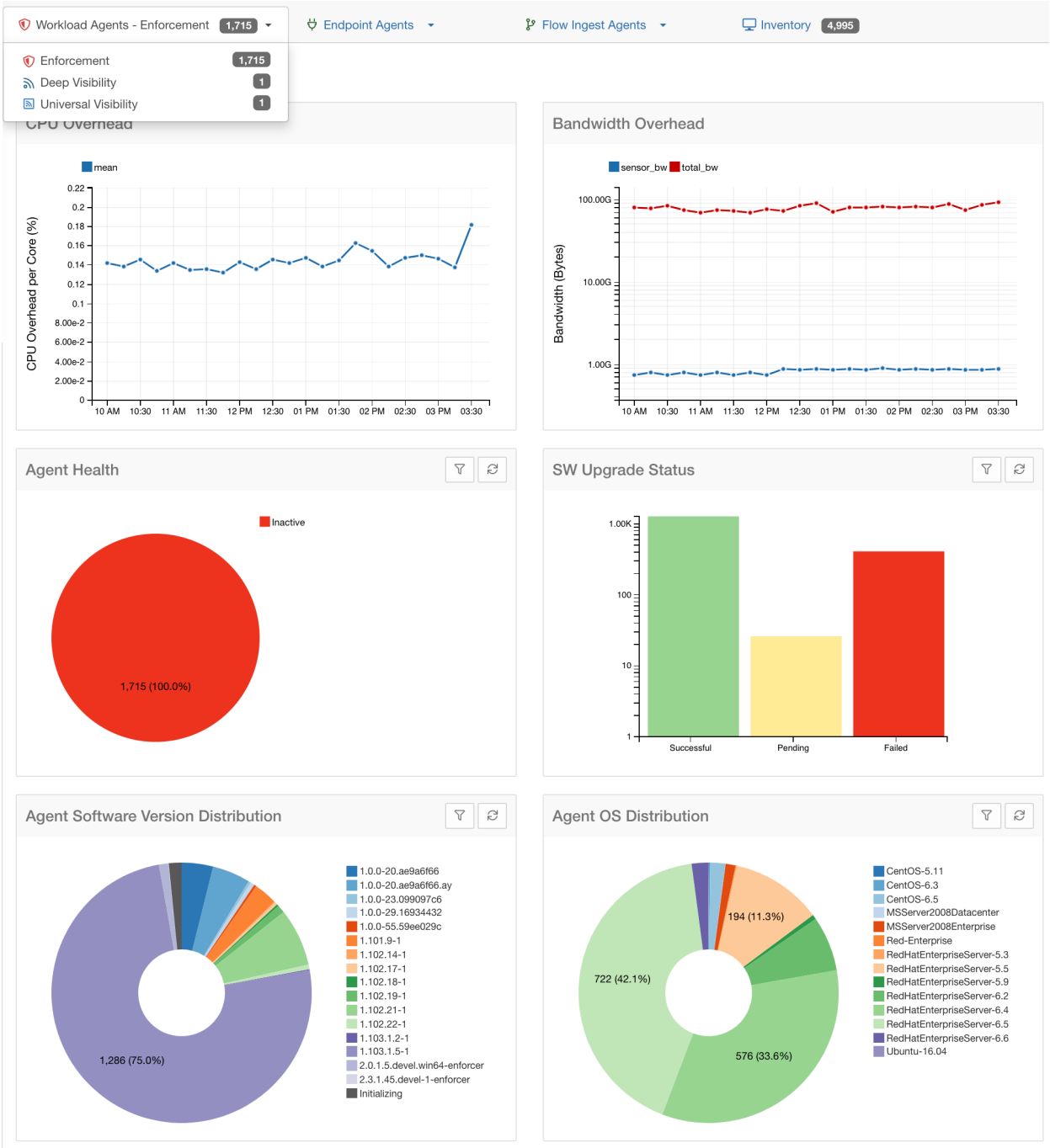


Fig. 12.1.1.1.1: Agents Distribution

For each agent type, this page provides an overview and the health of registered agents including overall CPU overhead, bandwidth overhead, missed packets, OS/version distribution and agent upgrade status.

CPU Overhead Chart

The CPU Overhead chart provides an aggregated view of CPU overhead per core from all agents. Per-agent CPU Overhead is provided as part of the *Workload Profile*. This chart is only available for Deep Visibility and Enforcement Agent Types.

Bandwidth Overhead Chart

The Bandwidth Overhead chart provides aggregated stats of total bandwidth and bandwidth used by agents. Per-agent bandwidth overhead is provided as part of the *Workload Profile*. This chart is only available for Deep Visibility and Enforcement Agent Types.

Agent Health Chart

The Agent Health chart provides number of active/inactive agents. Active agents are the one checking in with config server for upgrade on regular intervals. The checking interval is 30 minutes. If we see that an agent has missed more than 2 check-in periods from a agent, it would be declared as inactive agents.

Software Agent Updates to Latest Revision Chart

Every time an agent checks in with the config server, the agent would also provide its current RPM version. If an agent is configured to a specific version and is not able to update after 2 check-in periods, the agent would be declared as not able to upgrade to the latest version.

Agent Packet Missed Chart

In rare occasions when the traffic volume traversing a host is greater than the rate at which the agent is able to inspect, some packets will be skipped from being analyzed. The number of missed packets and the corresponding agent name will be shown in this chart.

Agent Software Version/OS Distribution Charts

These charts show the agent version distribution and parent OS platform of all agents registered with Tetration cluster.

12.1.2 Hardware Switch Agent

The **Hardware Switch Agents** tab shows the status of all registered switches to a given cluster.

Serial	IP Address	Name	Switch SW Ver	Agent SW Ver	Bootup Time	Last Check-in	First Check-in
[illegible]	[illegible]	[illegible]	n9000-12.0(0.128)	1.102.2	5:15 PM	10:00 PM	Apr 22, 11:10 AM
FOX0912034U	[illegible]	[illegible]	n9000-12.0(0.128)	1.102.2	6:43 PM	10:00 PM	Apr 22, 11:07 AM
[illegible]	[illegible]	[illegible]	n9000-12.0(0.128)	1.102.2	2:14 PM	10:00 PM	Apr 22, 11:07 AM
[illegible]	[illegible]	[illegible]	bootflash:/nxos.7.0.3.I4.1.bin	1.102.2	May 16, 1:26 PM	11:09 AM	Apr 22, 11:07 AM
[illegible]	[illegible]	[illegible]	nxos.7.0.3.I3.1.bin	1.102.2	Apr 6, 6:57 PM	10:13 AM	Apr 26, 3:31 PM
[illegible]	[illegible]	[illegible]	bootflash:/nxos.7.0.3.I3.1.bin	1.101.6	May 16, 1:54 PM	May 20, 3:35 PM	May 2, 11:27 AM
[illegible]	[illegible]	[illegible]	bootflash:/nxos.7.0.3.I3.0.175.bin	0.17.4	Apr 11, 10:34 AM	Apr 26, 3:10 PM	Apr 22, 11:07 AM
[illegible]	[illegible]	[illegible]	bootflash:/nxos.7.0.3.I3.1.bin	1.101.4	Feb 7, 2:38 AM	May 20, 1:14 PM	Apr 22, 11:12 AM

Fig. 12.1.2.1: Hardware Switch Agent Table

The **Last Check-in** time specifies the time when the config server received a message from that switch. For an active hardware agent, this should be within 5 minutes of the current time as the agent is expected to send periodic messages to the config server.

To see more detailed view of hardware agents you can click on the row to expand **Switch Details**.

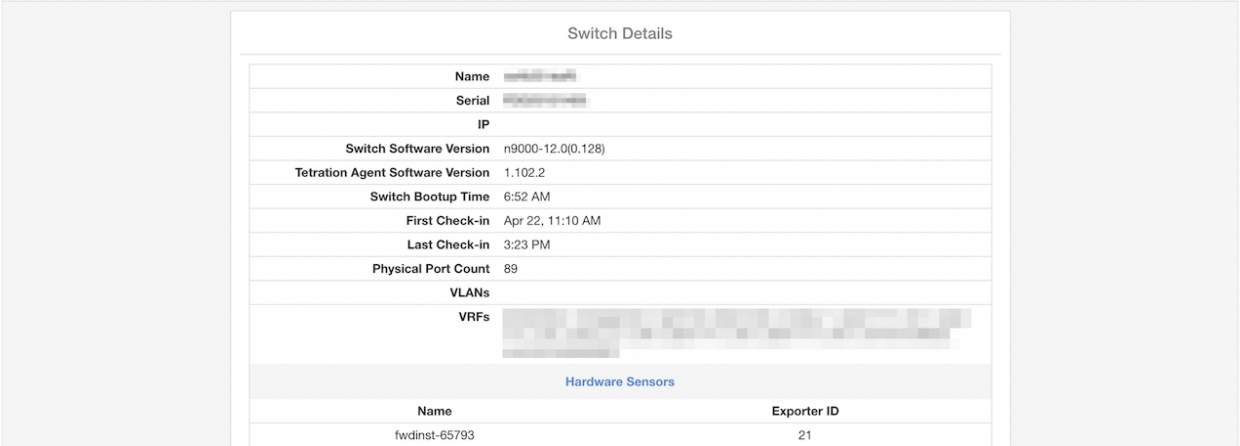


Fig. 12.1.2.2: Hardware Switch Agent Details

12.2 Enforcement Status

This page is available for site admin/customer support users and scope owners to get an overview of the current status of all the enforcement agents. For each agent, the current desired version of the concrete policies to be enforced is shown along with the last version that has been enforced. There are three ways to filter the status of agents:

1. Filter by the faceted filter
2. Filter by the distribution charts based on the status of enforcement enabled, policy config and concrete policy generation
3. Filter by root/child scope - SA/CS users have option to turn the scope filter on/off and scope owner users cannot turn off the scope filter

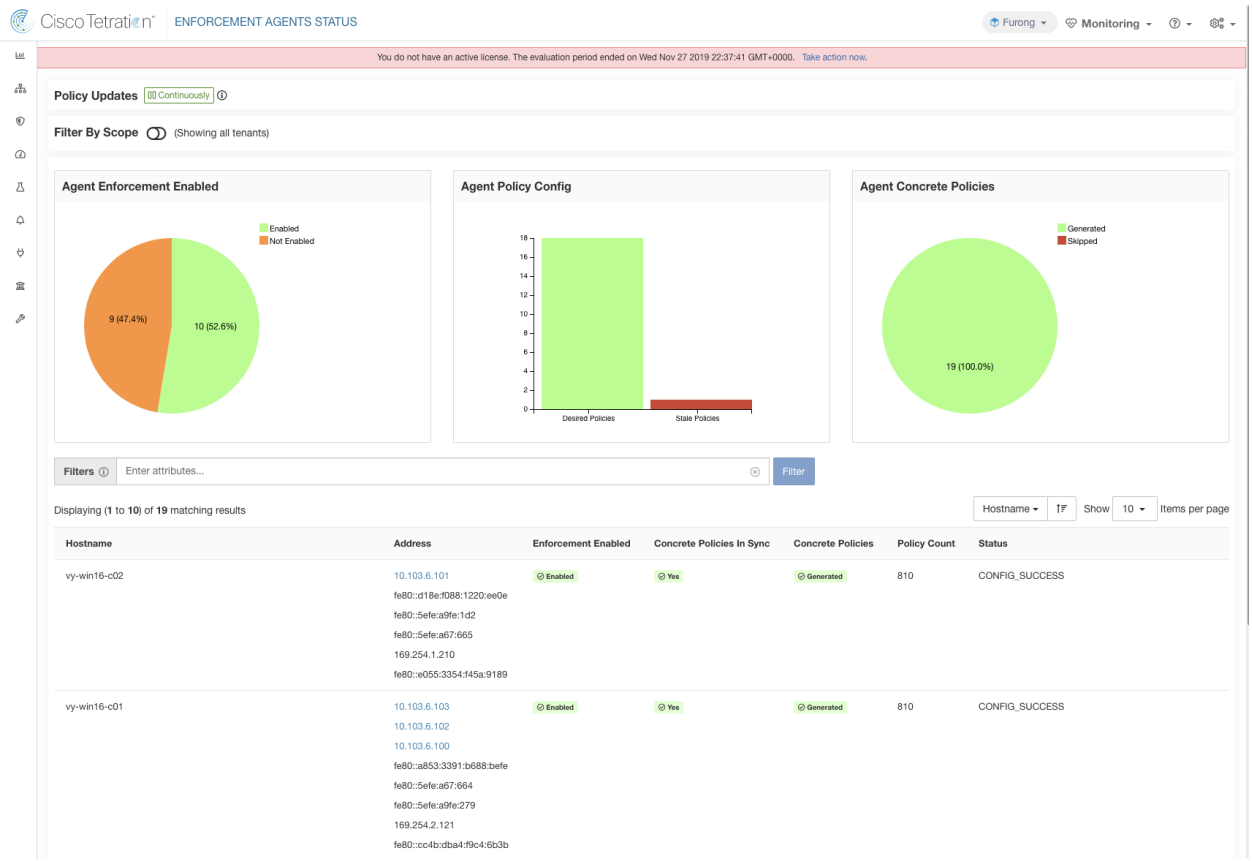


Fig. 12.2.1: Filter by all tenants - Site Admin

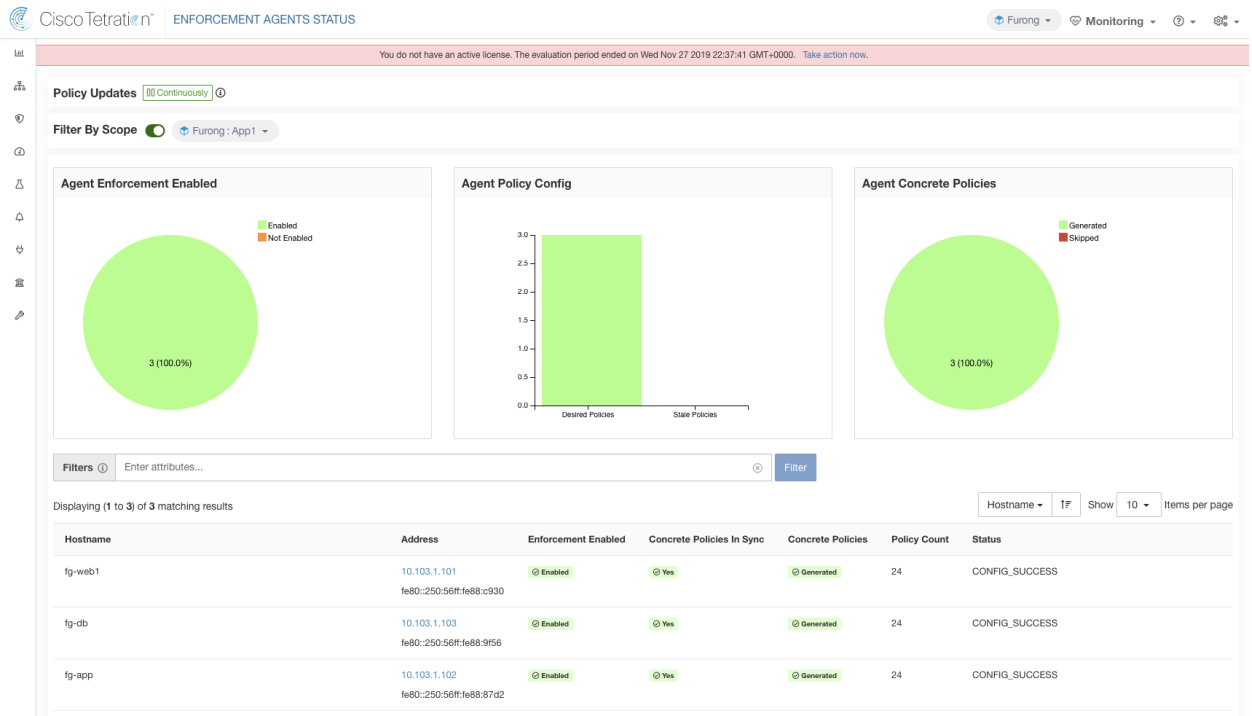


Fig. 12.2.2: Filter by root/child scope - Site Admin

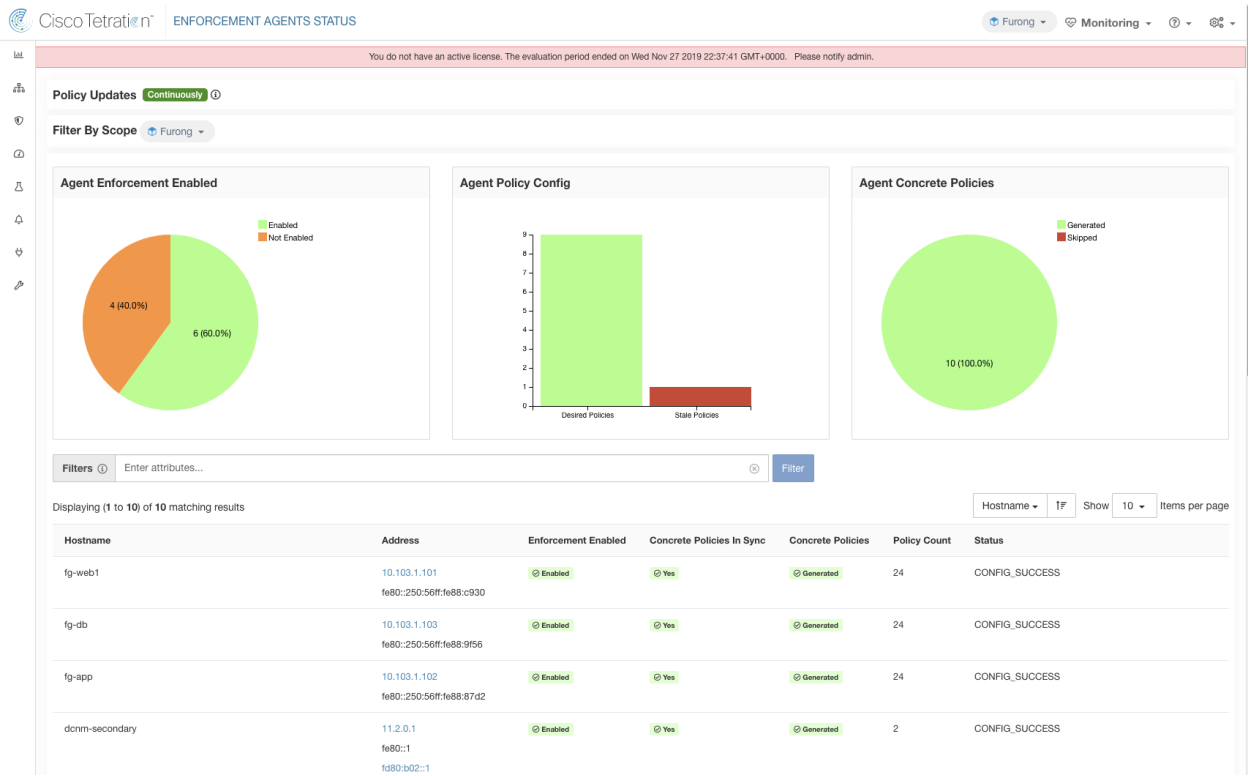


Fig. 12.2.3: Filter by root/child scope - Scope Owner

The following table describes the fields shown in enforcement status table.

Table 12.2.1: Enforcement Status Table

Field	Description
Host Name	Host name of the agent.
Address	IP addresses of all the interfaces on the agent. From these addresses, we can navigate to the host profile
Enforcement Enabled	Indicates whether enforcement is enabled or not on the agent.
Concrete Policies in Sync	This indicates whether the desired version of concrete policies are currently enforced on the agent.
Concrete Polices	This field indicates whether the generation of concrete policies is skipped for the host. This happens wh
Policy Count	The policy count of the agent.
Status	The status of the latest policy config enforcement. If the status is **CONFIG_SUCCESS**, it indicates that

12.2.1 Pausing policy update

Firewall rule update in all enforcement endpoints can be paused or un-paused through the toggle button. This feature is reserved for site admin and customer support. Please note that the pausing/un-pausing is a global configuration regardless of the user's current scope.

> **Warning:** Please exercise caution during this operation as pausing/un-pausing is an **appliance-wide configuration** regardless of the user's current scope and so can potentially affect policy enforcement on a wider set of workloads than the user's current scope.

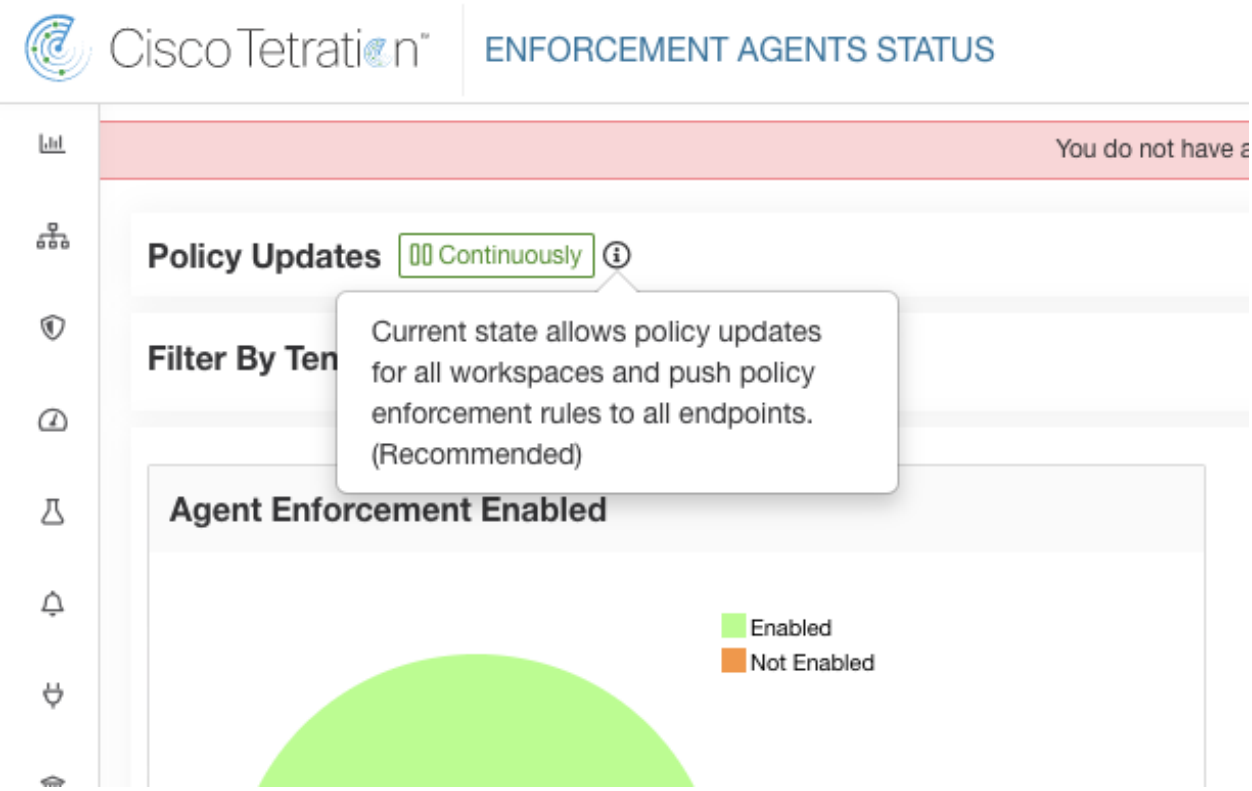


Fig. 12.2.1.1: Firewall rules are being updated continuously

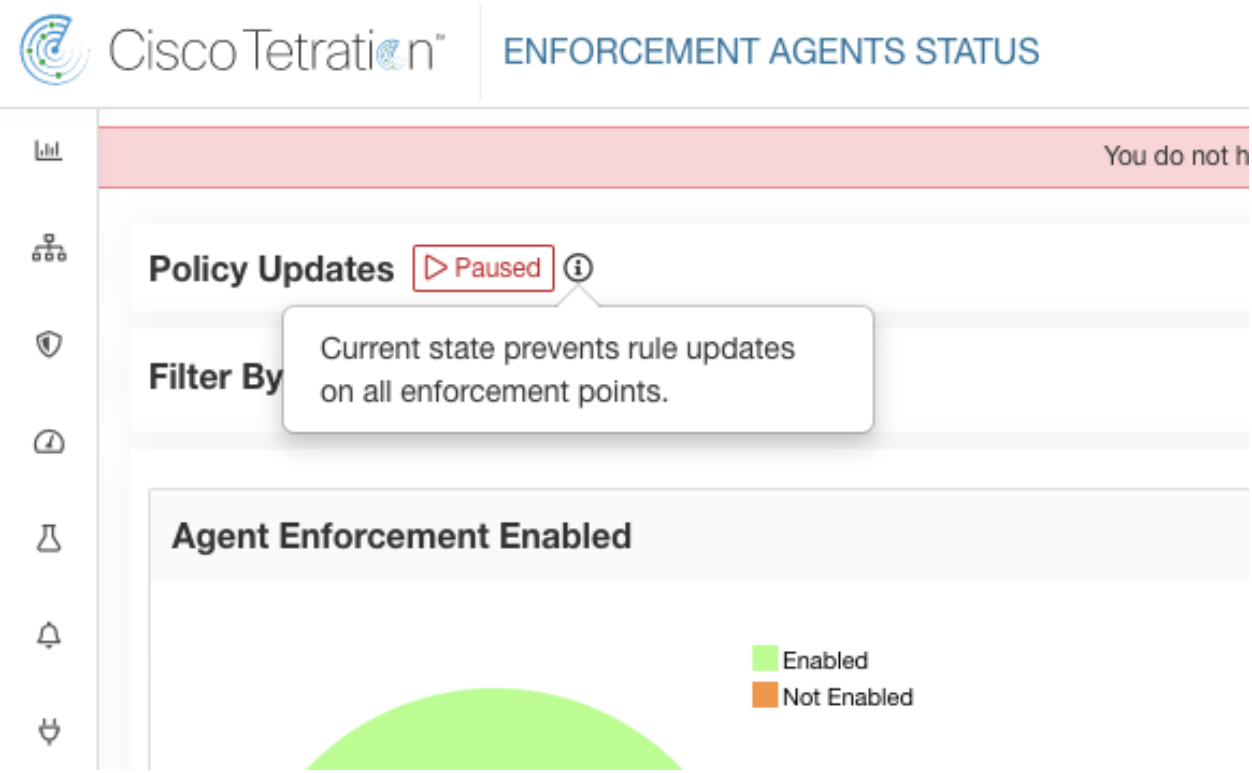


Fig. 12.2.1.2: Firewall rule updates are paused

12.3 Licenses

This page is available for site admin to get an overview of the current licensing status and license usages. In this release and forward, it is required to register the cluster for on-premises deployment. When you upgrade to or deploy a new cluster with this release, software will automatically enter a 90 days evaluation mode. A banner will be displayed and show the evaluation expiration date.

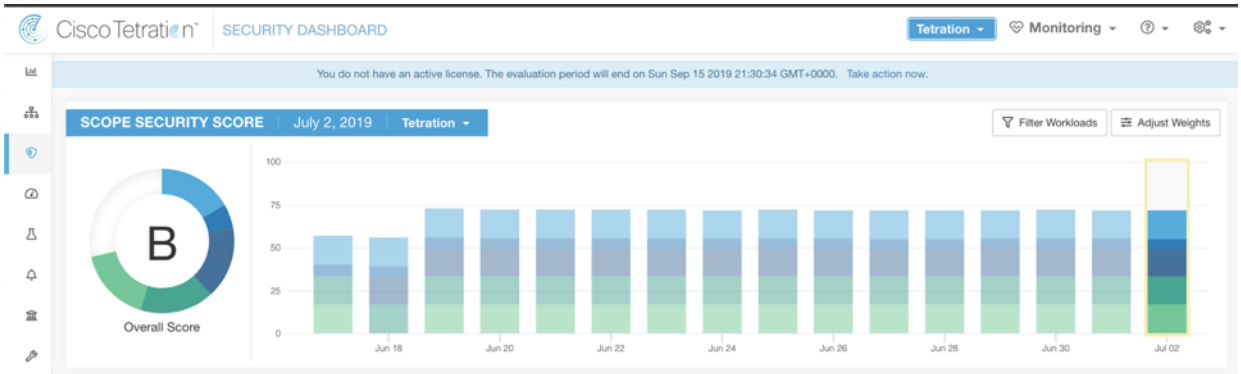


Fig. 12.3.1: License banner

Note: If the registration is not completed successfully within the 90 days period, the banner message will change to out-of-compliance. No feature or functionality will be blocked due to non-registration.

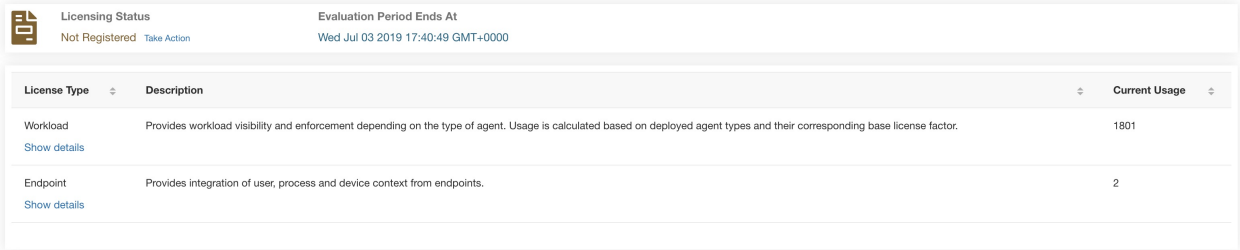


Fig. 12.3.2: In monitoring - licenses page, detailed license information is displayed

12.3.1 License Registratration

This section explains how to obtain a license.

Click **Take Action** in the license banner or in **Monitoring - Licenses** page to request a license. A **License Registratration Modal** will be displayed with instructions on how to download a cluster identify file and how to acquire a license.

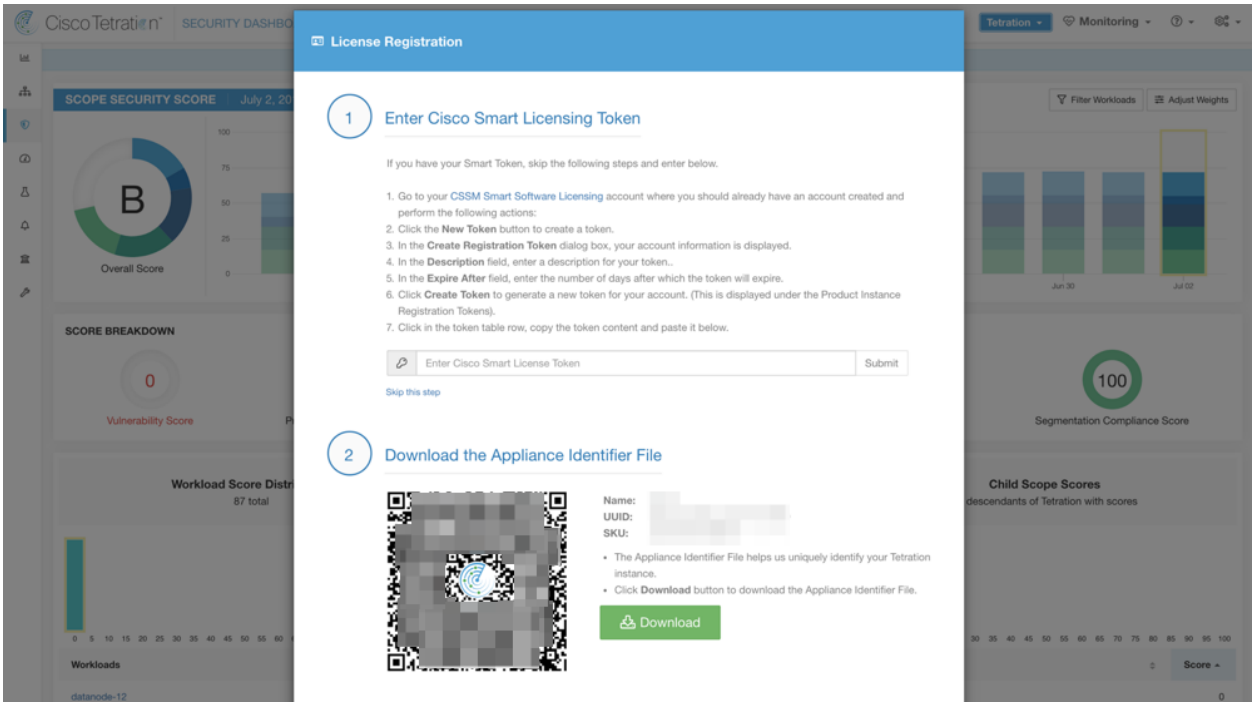


Fig. 12.3.1.1: License registration modal - Download cluster identify file

1. To complete **License Registratration Modal** it requires registration token generated through CSSM Smart software licensing portal. The steps to generate the token through CSSM is provided in the license modal itself. Once you have the registration token, copy and paste the token into the text box in the licensing modal and click the **Submit** button next to the text box.

2. Next, click the **Download** button to download the cluster identify file to local storage. File name format for the identify file is: **reg_id_<cluster_name>_<cluster_uuid>.gz**. The identity file does not contain any IP address information, specific workload details or PII information. This identity file needs to be sent to ta-entitlement@cisco.com. A response that contains the **license key file** will be sent to the same email address from which the identity file was received.
3. This **license key file** must be uploaded through the licensing model. Step 4 of the licensing modal should be used to upload the response file.

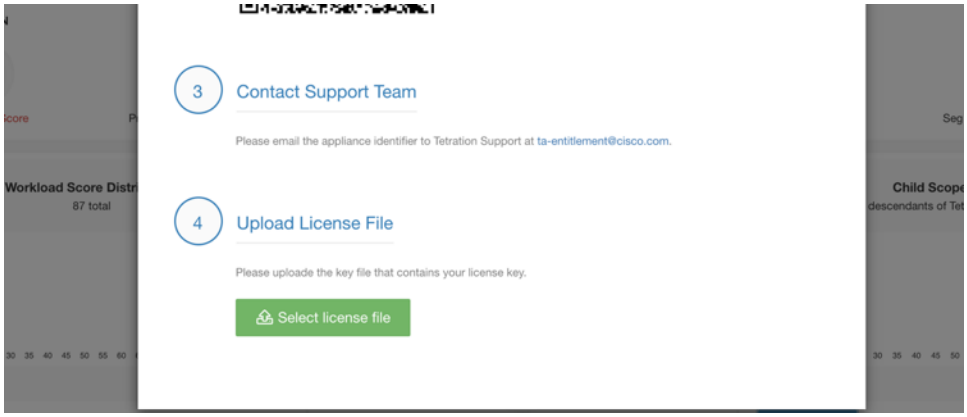


Fig. 12.3.1.2: License registration modal - Upload license key file

12.3.2 Check License Usage

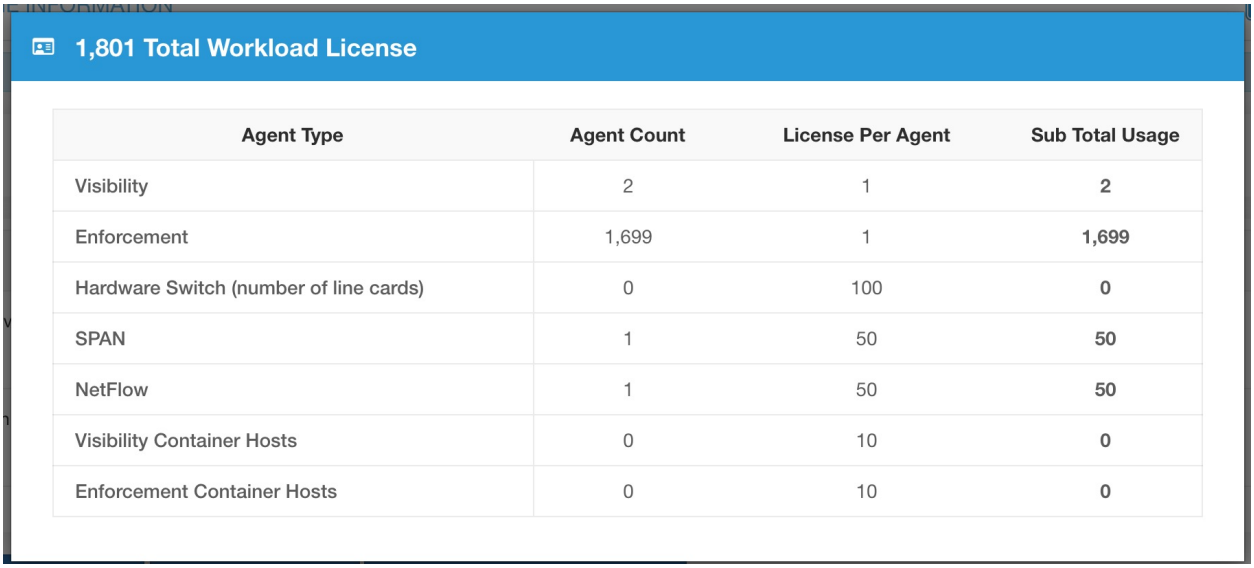
This section explains how to check the detailed license usage.

1. License usage can be fond in the license table in **Monitoring - Licenses** page.

License Type	Description	Current Usage	Entitlement
Workload Show details	Provides workload visibility and enforcement depending on the type of agent. Usage is calculated based on deployed agent types and their corresponding base license factor.	0	267
Endpoint Show details	Provides integration of user, process and device context from endpoints.	0	946

Fig. 12.3.2.1: License Table

2. Click **Show details** to see details on the license usage.



1,801 Total Workload License

Agent Type	Agent Count	License Per Agent	Sub Total Usage
Visibility	2	1	**2**
Enforcement	1,699	1	**1,699**
Hardware Switch (number of line cards)	0	100	**0**
SPAN	1	50	**50**
NetFlow	1	50	**50**
Visibility Container Hosts	0	10	**0**
Enforcement Container Hosts	0	10	**0**

Fig. 12.3.2.2: Detailed License Usage

Note: After the registration, if the license usage exceeds the entitlement (workload or endpoint), a non-compliant warning banner would be displayed in the UI. Exceeding the license usage does not block any feature or functionality including installing additional sensors. If the usage falls below the entitlement, then the compliance warning banner goes away. If additional licenses have been purchased, you can reach out to ta-entitlement@cisco.com along with the identity information (Download it again from the license modal) and request an updated license key file.

12.3.3 More on Cisco Smart Licensing

Cisco Smart Licensing is a unified license management system that manages all the software licenses across Cisco products. If you have a Cisco Smart Licensing account, you can associate the Cisco Smart Licensing Token with a Tetration license. If you don't have a Cisco Smart Licensing account, you can acquire/update a license without Cisco Smart Licensing.

1. If you already a valid Tetration license, you can click **Request A New License To Enroll** to acquire a new license with Cisco Smart Licensing Token.

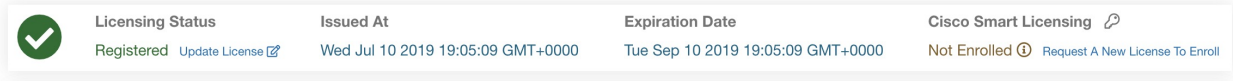


Fig. 12.3.3.1: Acquire a new license to associate Cisco Smart Licensing Token with a Tetration license

2. If you do not have a valid Tetration license, you can click **Take Action** to acquire a new license as described in the previous sections.

CHAPTER

THIRTEEN

THREAT INTELLIGENCE

The **Threat Intelligence** feature set provides the most up to date datasets for Tetration pipeline that identifies and quarantines threats by inspecting the datacenter workloads against externally-known malware command and control addresses, security flaws in processes and geographical location.

The Threat Intelligence dashboard shows the most update status of Threat Intelligence datasets. These datasets are updated automatically.

Warning: The Threat Intelligence feature requires a connection to Tetration servers to automatically update. Your enterprise outbound HTTP request may require:

1. Allow the following domain from enterprise firewall outbound rules:
   - uas.tetrationcloud.com
2. *Outbound HTTP Connection* configuration.

In environments without an outbound connection, these datasets can be uploaded directly. Please refer to *Manual Uploads*.

Datasets

Dataset	Description
NVD CVEs	Security related software flaws, CVSS base score, vulnerable product configuration, and weakness categorization
MaxMind Geo	Identification of the location and other characteristics of source IPs
NIST RDS	NIST Reference Data Set of digital signatures of known, traceable software applications
Team Cymru	Insight on over 3,000 botnet command and control IPs
Hash Verdict	Tetration's verdict on process hashes (only available via *Automatic Updates*)

Note: In case MaxMind Geo dataset was manually uploaded in an earlier release, please re-upload the corresponding RPM in order to view location and related information in flow visibility page.

Threat Intelligence topics

13.1 Automatic Updates

When Internet connection to Cisco Threat Intelligence server is avaialble, the Threat datasets will be updated periodically. The Threat Intelligence dashboard lists datasets and the date of the dataset's last update.

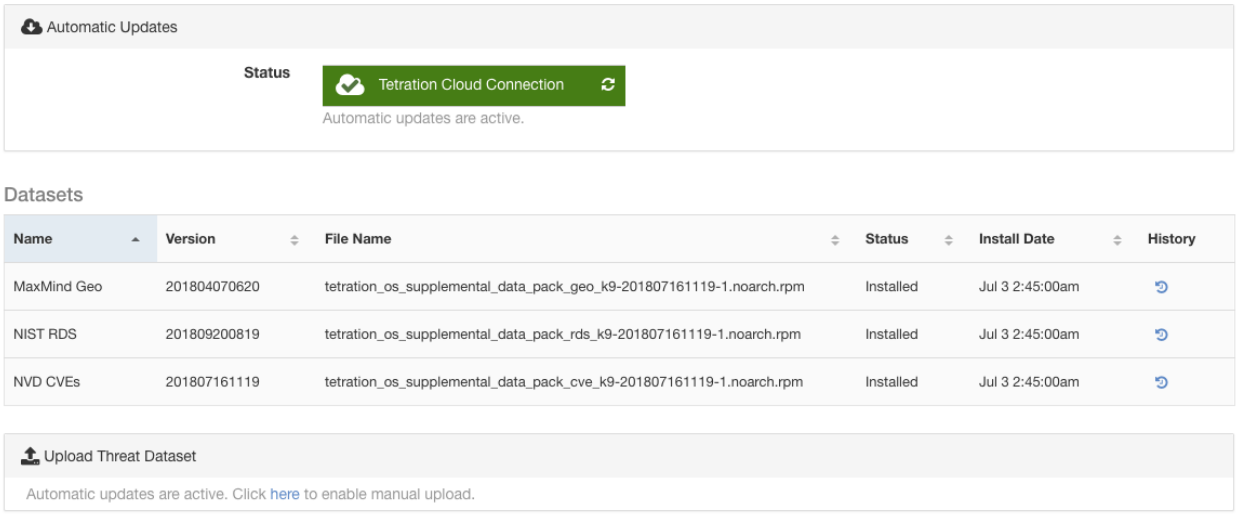


Fig. 13.1.1: Dashboard

13.2 Manual Uploads

> **Attention: Scheduling Manual Uploads**
>
> Dataset rpm files are published to Cisco Tetration Update Portal weekly. We recommend installing the latest releases periodically and setting a schedule for an adminstrator to do so.

13.2.1 Downloading updated Datasets

The datasets can be downloaded from Cisco Tetration Update Portal.

13.2.2 Uploading to Tetration

This section explains how to upload dataset rpm files.

Before You Begin

You must login as **Site Admin** or **Customer Support** in the system.

1. Click on **Security** link on the left panel.
2. Click on **Threat Intelligence** item. Threat Intelligence dashboard page appears.
3. Click on "Select Supplemental RPM" buttonm, under **Upload Threat Dataset**.

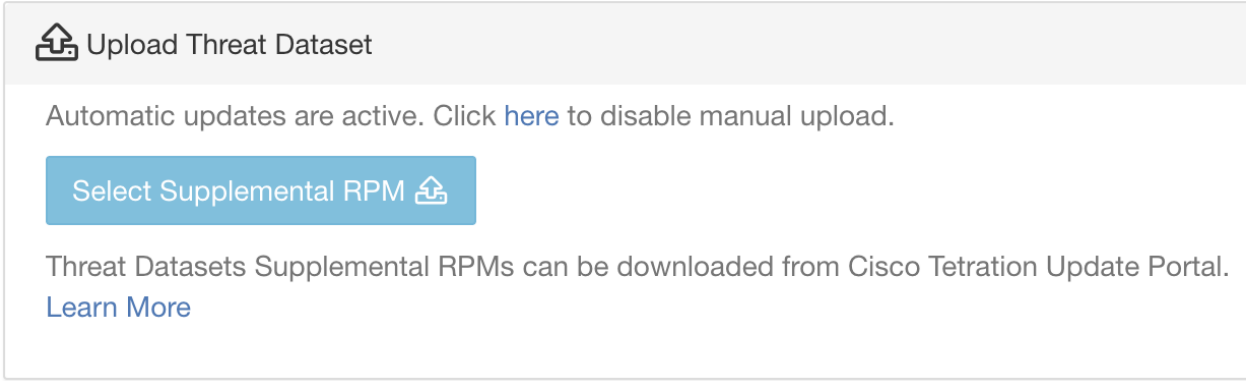


Fig. 13.2.2.1: Upload

4. Select a rpm file that you downloaded from Cisco Tetration Update Portal
5. Once ready, a confirmation dialog will appear. Click **Upload**.

Fig. 13.2.2.2: Upload ready

6. The rpm will then upload. Once uploaded the dialog will close.

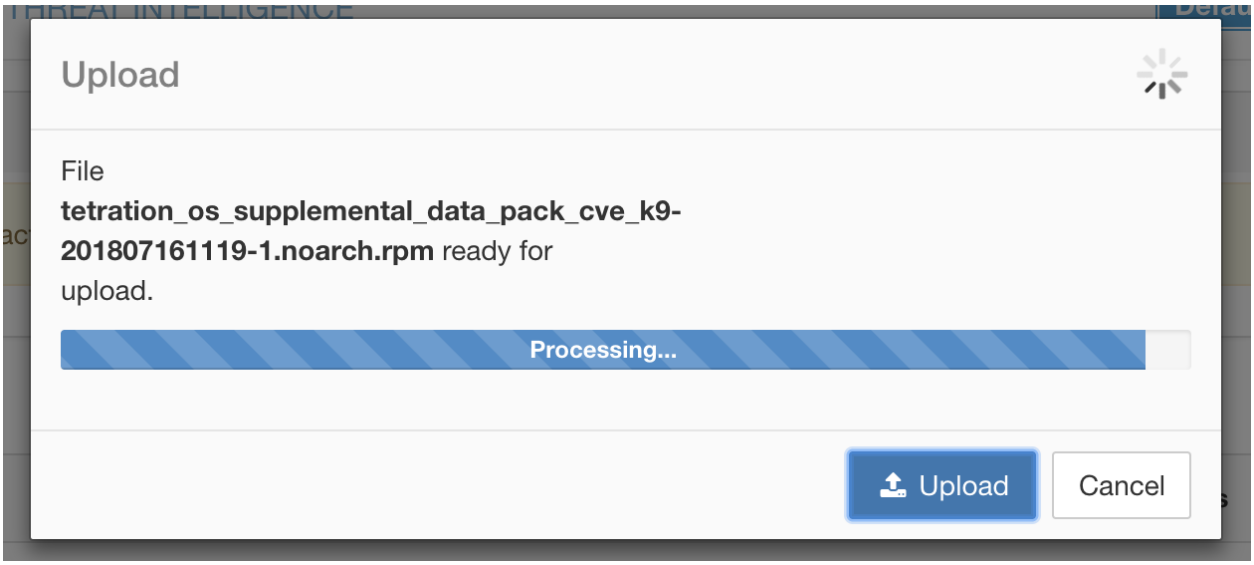


Fig. 13.2.2.3: Upload in progress

7. The rpm will then be processed and installed in the background. The table will update when this is complete.

Datasets

Name	Version	File Name	Status	Install Date	History
NVD CVEs	20180810024408	cve_data.download	Installed	Aug 9 7:44:34pm	
Team Cymru	20180810024407	threat_data.download	Installed	Aug 9 7:44:08pm	

Fig. 13.2.2.4: Updated table

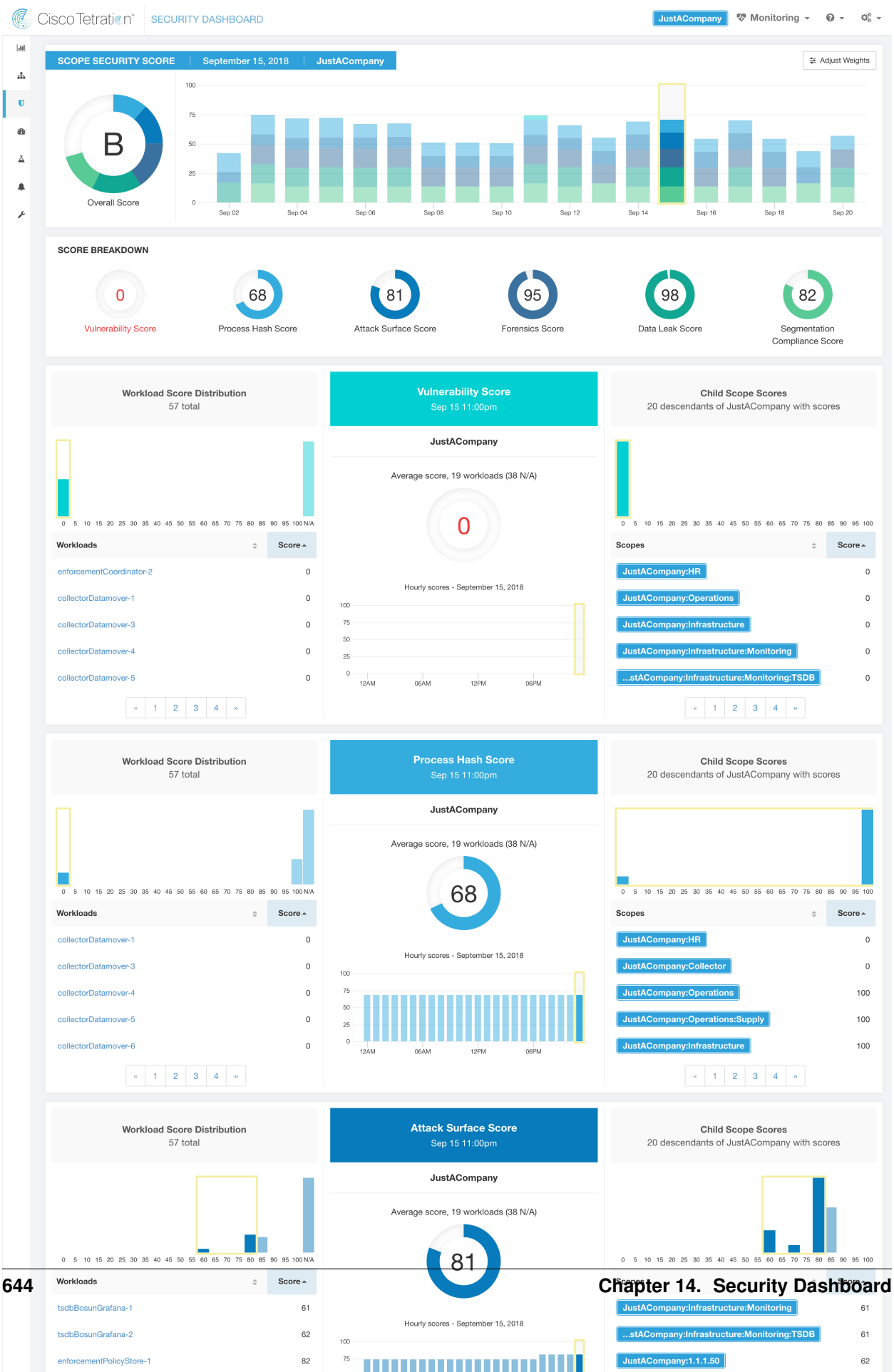
CHAPTER

FOURTEEN

SECURITY DASHBOARD

Security Dashboard presents actionable security scores by bringing together multiple signals available in Tetration. It helps in understanding the current security position and improving it.

Security Dashboard is acts as springboard to many richer drill-downs within Tetration such as Flow search, Inventory Search, ADM, Neighborhood, Forensics etc.


Cisco Tetration
SECURITY DASHBOARD
JustACompany
Monitoring
SCOPE SECURITY SCORE
September 15, 2018
JustACompany
Adjust Weights
B
Overall Score
SCORE BREAKDOWN
0
Vulnerability Score
68
Process Hash Score
81
Attack Surface Score
95
Forensics Score
98
Data Leak Score
82
Segmentation Compliance Score
Workload Score Distribution
57 total
Vulnerability Score
Sep 15 11:00pm
JustACompany
Average score, 19 workloads (38 N/A)
Hourly scores - September 15, 2018
Child Scope Scores
20 descendants of JustACompany with scores
Workloads
Score
Scopes
Process Hash Score
Attack Surface Score

14.1 Security Score

Security Score is a number between 0 and 100. It indicates the security position in category. A score of 100 is the best score, and a score of 0 is the worst. Scores closer to 100 are better.

The Security Score computation takes into account vulnerabilities in installed software packages, consistency of process hashes, open ports on different interfaces, forensic and network anomaly events, and compliance/non-compliance to policies.

14.2 Security Score Categories

There are 6 different score categories. Most security aspects of a workload are taken into account to come up with these categories.

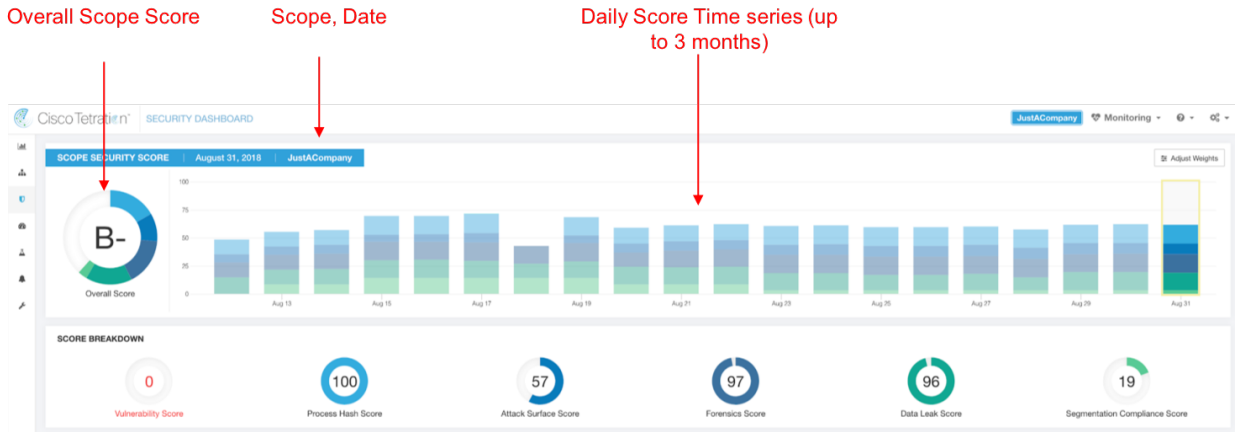
- **Vulnerability Score**: Vulnerabilities in the installed packages on a workload are used for scoring.
- **Process Hash Score**: Process hash consistency (and anomaly) along with Benign and Flagged process hashes is used for scoring.
- **Attack Surface Score**: Process may have one or more ports open on multiple interfaces to make services available. Unused open ports are used for scoring.
- **Forensics Score**: Severity of forensic events on a workload is used for scoring.
- **Network Anomaly Score**: Severity of network anomaly events on a workload is used for scoring.
- **Segmentation Compliance Score**: Compliance (permitted) and violations (escaped) to ADM policies is used for scoring.

14.3 High Level View

Security dashboard has scope level scores for the selected scope. There is overall score with time series and score breakdown. Score details for 6 score categories for selected scope appears down one by one.

14.4 Scope Level Score Details

Scope Level Score details is on top of the dashboard.

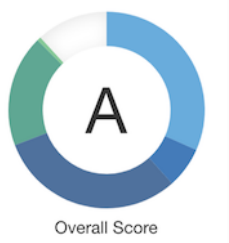


It has following:

- **Overall Scope Score**: Overall score for the selected scope.
- **Daily Score Time series**: Stacked time series that can go up to 3 months.
- **Score Breakdown**: Breakdown of category scores for the selected day on time series.

14.4.1 Overall Score

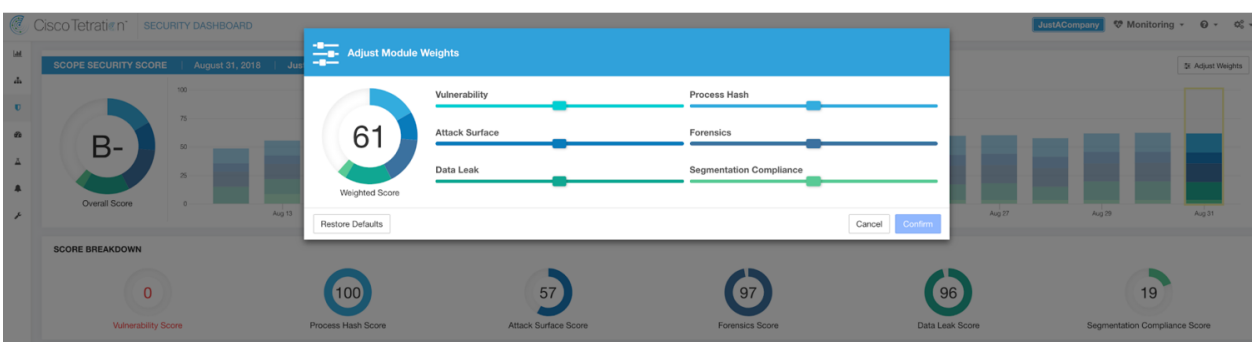
Overall score is letter from **A+**, **A**, **. . .** , **F**. **A+** is be best. **F** is the worst. It's a donut chart with each slice (color coded) representing a score category.



Overall score is the weighted average of 6 categories of scores. By default all weights are equal. If a score is **N/A**, it's considered as 0 in the overall score calculation.

$$Overall\ score = \frac{\sum W_{category} \times Score_{category}}{\sum W_{category}}$$

Weights can be adjusted using slides in the **Adjust Weights** module. Each user can set their own weight adjustments, which helps in aligning scores with user's priorities.

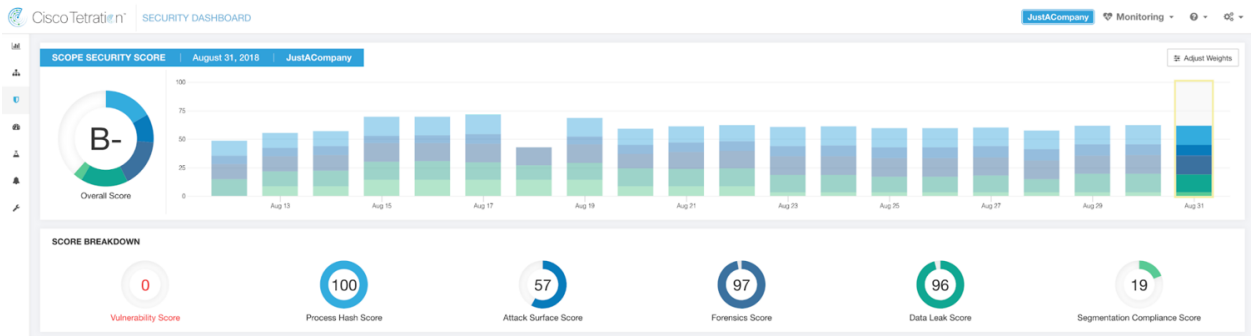


Important: If a score is **N/A**, it's considered as **0** in the overall score calculation.

14.4.2 Daily Time Series

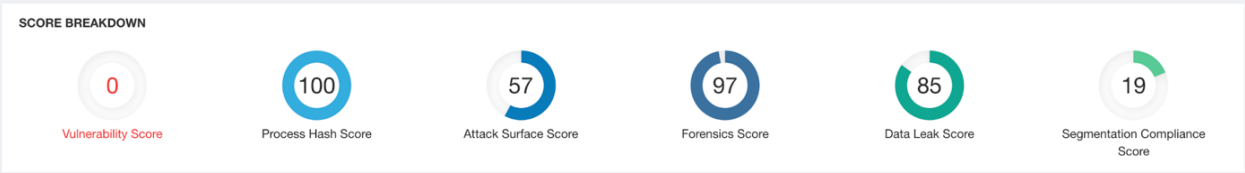
Stacked time series that can go up to 3 months. It helps in tracking security position over a long period. Each stack represents overall score for a day. Each segment in the stack is a category represented by a different color. You can

click on day to get score breakdown for the day.



14.4.3 Score Breakdown

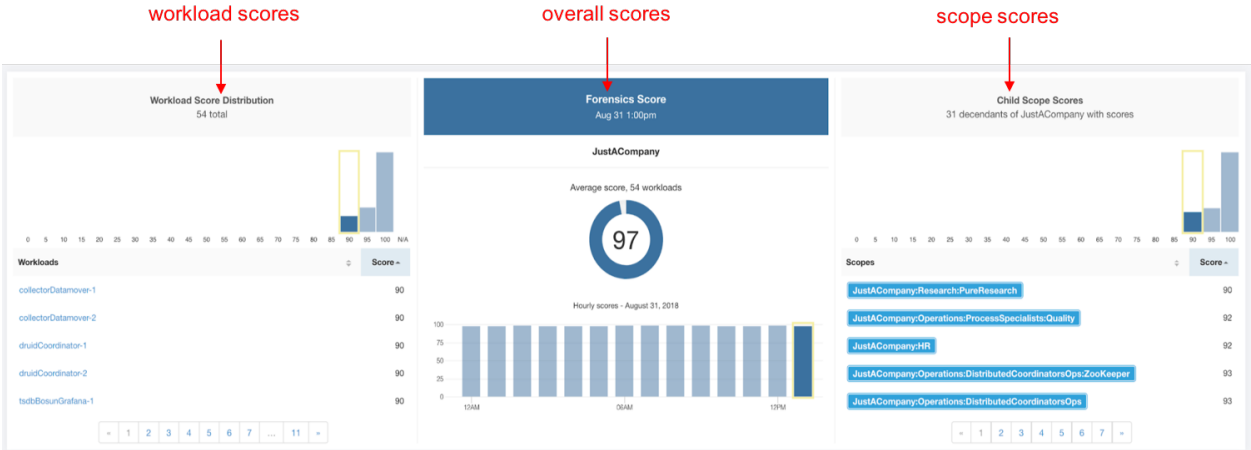
The Score Breakdown shows the score for all 6 categories for the day selected on the time series. Score **N/A** indicates that score is not available. It will be counted as 0 for overall score calculation.



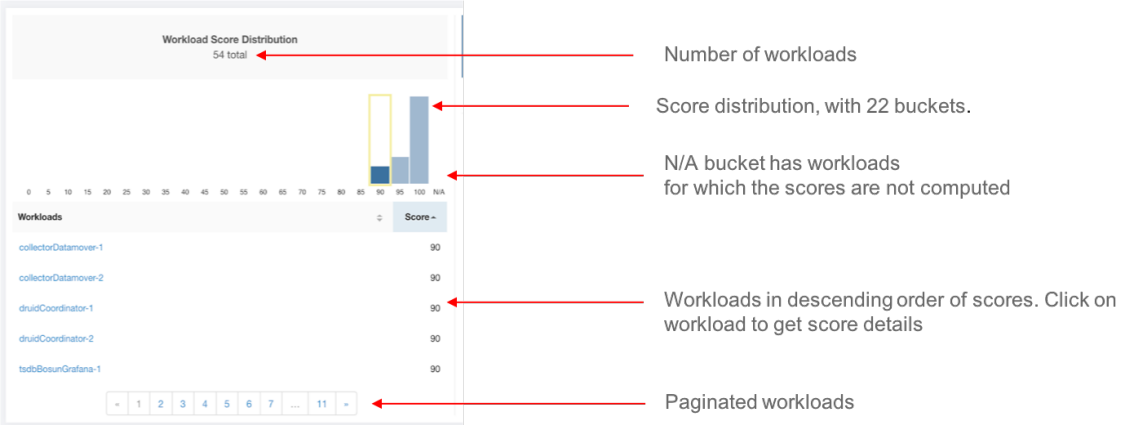
Important: If a score is **N/A**, it's considered as **0** for overall score calculation.

14.5 Score Details

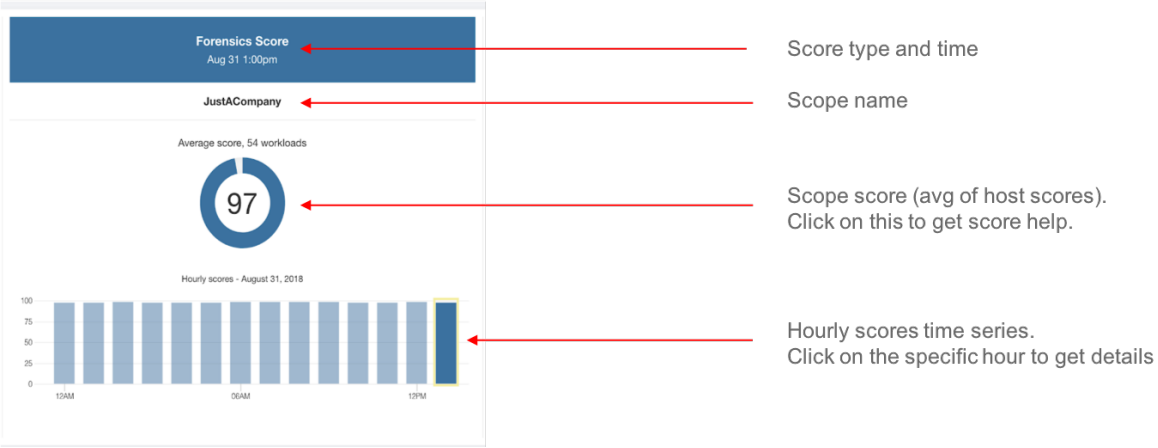
Each of the 6 categories follow the following template. It has workload score distribution, hourly time series and child scope score distribution.



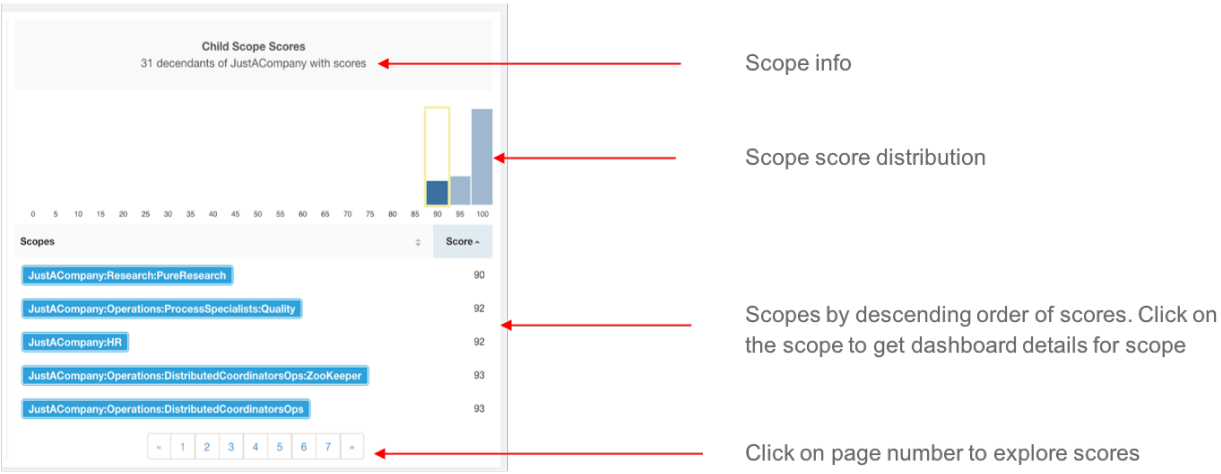
Workload score distribution provides insight into score contribution from workloads under the selected scope. It helps to bubble up lowest-scoring workloads to expedite corrective actions.



Hourly time series helps in getting hourly score over the course of a selected day. Selecting an hour in the hourly time series updates the workload score distribution and descendent scope distribution to show the selected hour.



Descendent scope distribution provides insight into score contribution of child scopes of the selected scope.



Details of each score category is explained in this section.

14.5.1 Vulnerability Security Score

Vulnerabilities in software packages installed on workloads is used for computing Vulnerability Security Score.

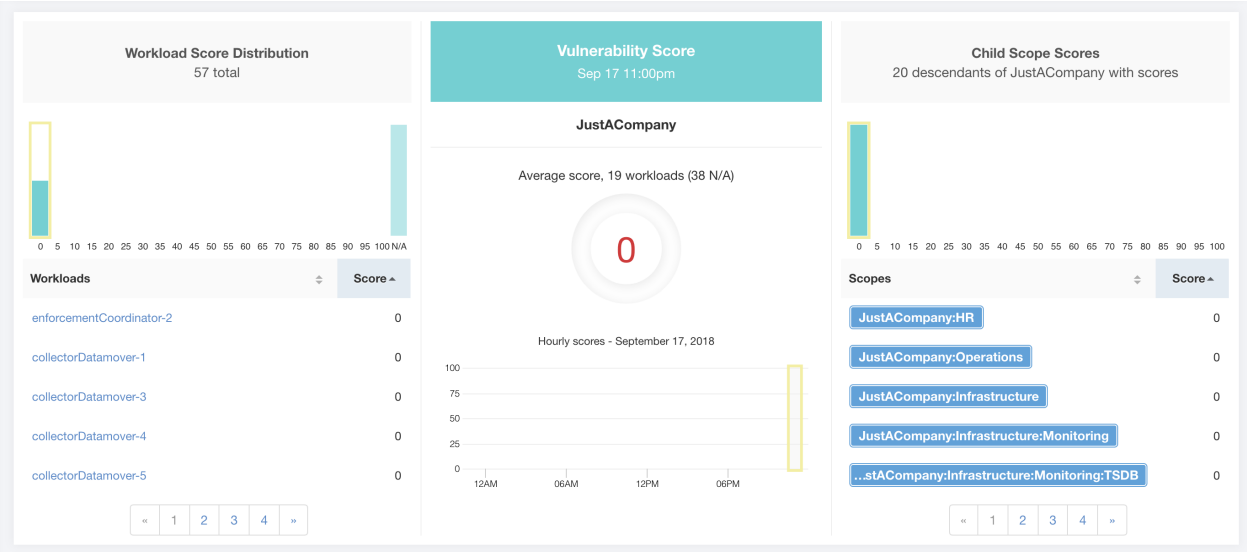


Fig. 14.5.1.1: Vulnerability Security Score Details

Lower score indicates:

- One or more installed software packages have serious vulnerabilities
- Apply patch or upgrade to reduce the chances of exposures/exploits

Software packages on a workload could potentially be associated with known vulnerabilities (CVE). CVSS (Common Vulnerability Scoring System) is used for assessing the impact of a CVE. CVSS score ranges from 0 to 10, with 10 being the most severe.

CVE can have CVSS v2 and CVSS v3 score. To compute Vulnerability score, CVSS v3 is considered if available, else CVSS v2 is considered.

Vulnerability score for a workload is derived from scores of vulnerable software detected on that workload. The Workload Vulnerability Score is calculated based on the CVSS scores, the vendor data, and may be adjusted by our security research team when data is missing or inaccurate (common for new vulnerabilities). This data is updated every 24 hours when the threat feed is configured. Higher the severity of the most severe vulnerability, lower is the score.

Scope score is average of workload scores in the scope. Improve the score by identifying workload/scopes with vulnerable software packages, and patch/upgrade with safer packages.

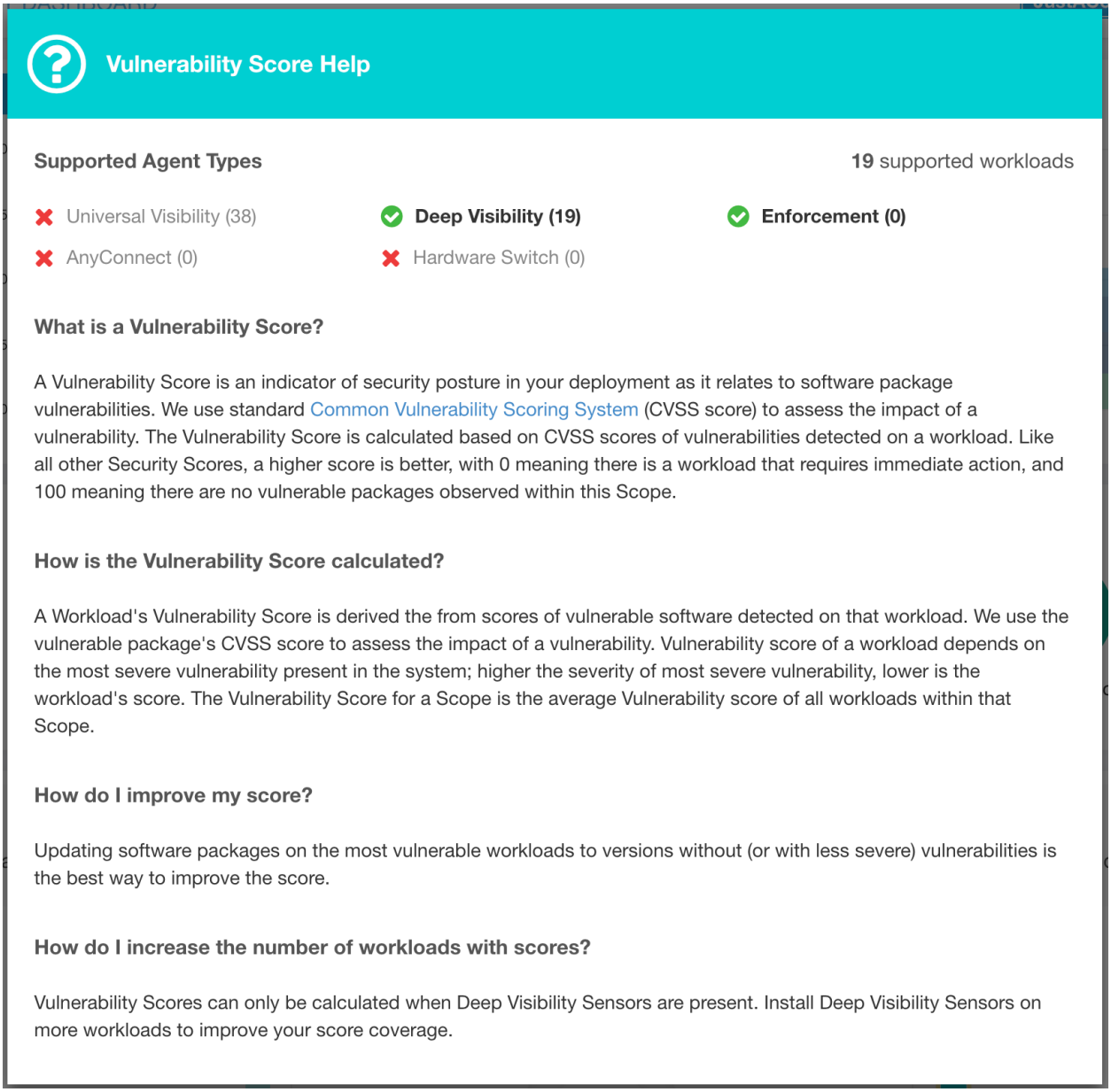


Fig. 14.5.1.2: Help for Vulnerability Security Score

14.5.2 Process Hash Score

Process hash score is assessment of process binary hash (file hash) consistency across workloads. For example: A web server farm running Apache cloned from the same setup config is expected to have same hash for httpd binaries on all servers. A mismatch is an anomaly.

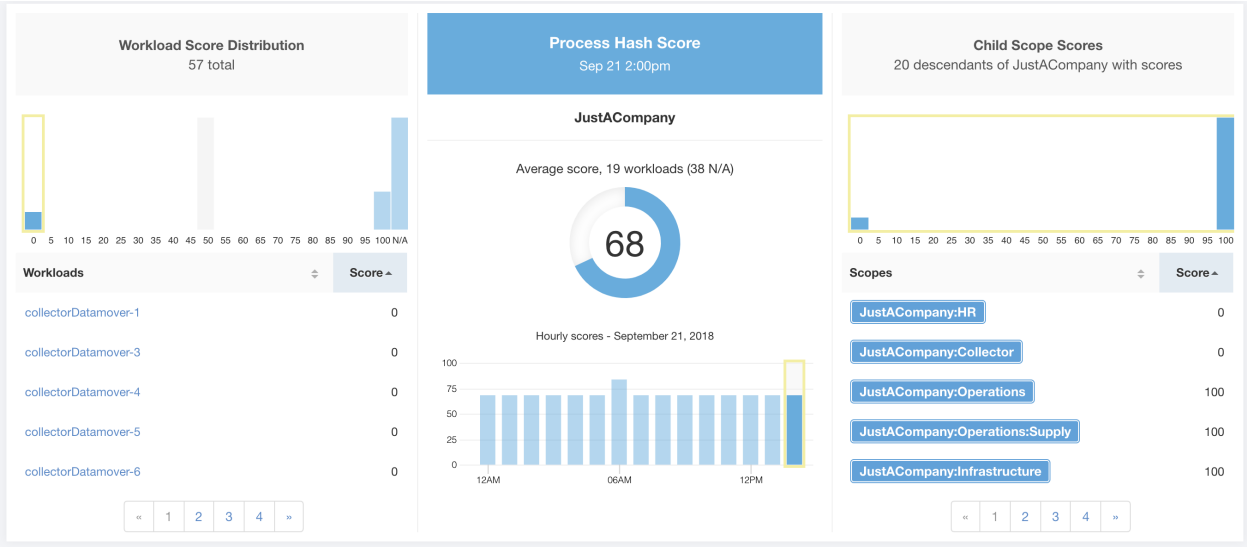


Fig. 14.5.2.1: Process Hash Score Details

Lower score indicates, at least one or both of:

- One or more process hashes are flagged
- One or more process hashes are anomalous

Refer to *Process hash anomaly detection* for more details.

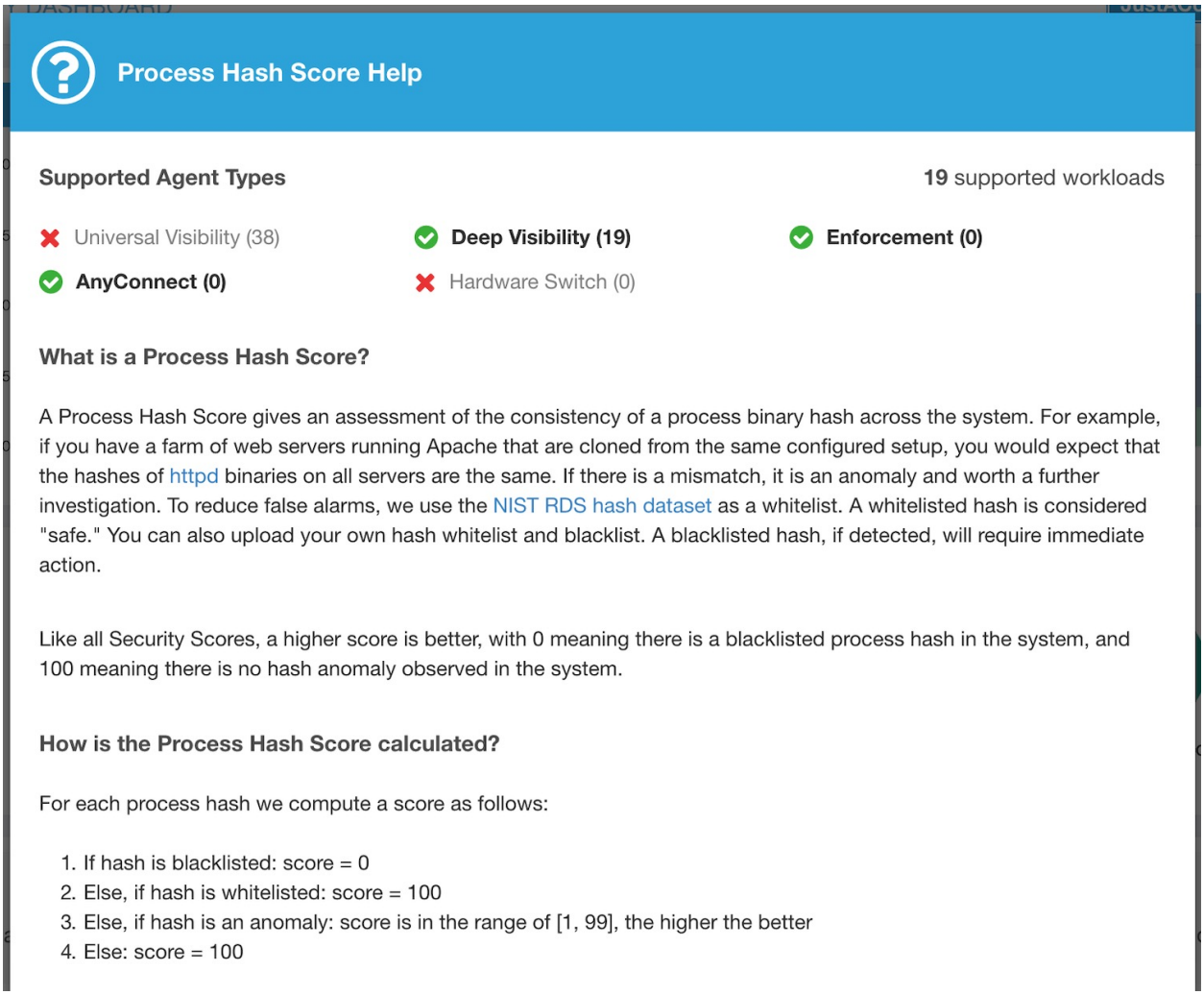


Fig. 14.5.2.2: Help for Process Hash Score

14.5.3 Attack Surface Score

Attack Surface Score highlights potential attack surface in a workload. Open unused ports (open ports without traffic) contribute to lowering this score.

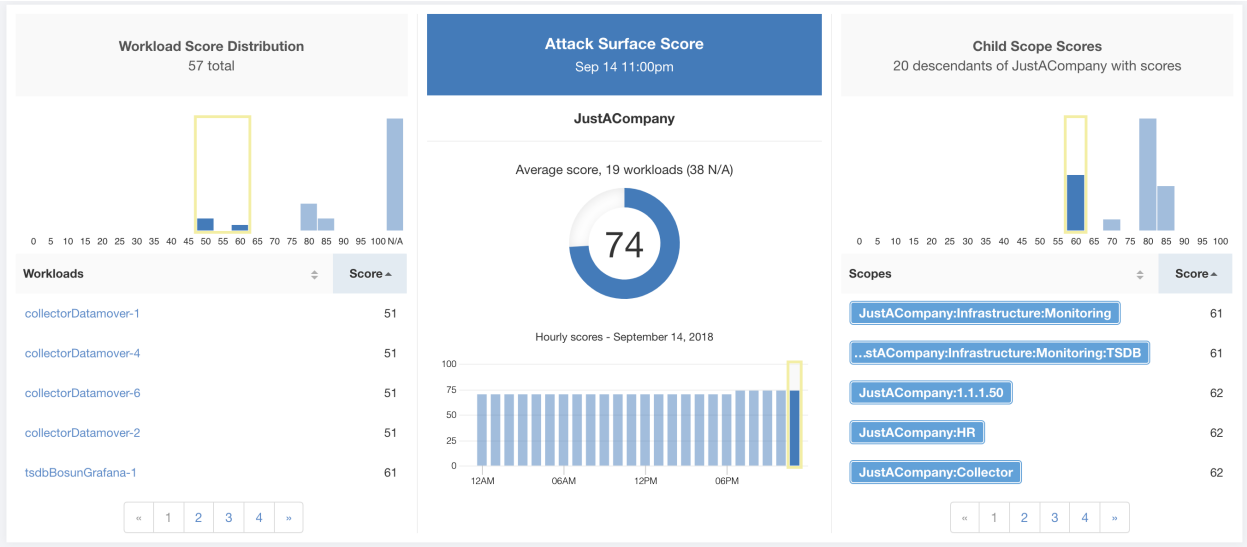


Fig. 14.5.3.1: Attack Surface Score Details

A lower score indicates:

- Many open ports without any traffic in the last 2 weeks
- Well known attack ports may be open and unused in last 2 weeks
- One or more open ports are attached with packages that have serious vulnerabilities

The attack surface score is a function of unused open ports relative to total ports, with a smoothing factor. Open ports without any traffic over the past 2 weeks are considered "unused open ports". An additional penalty is applied to unused open ports which are well known ports used in attacks (e.g., 21, 22, 8080 etc.).

$$Attack\ surface\ score = \frac{\alpha + \sum used\ open\ ports}{\alpha + \sum open\ ports + (\rho * \sum unused\ common\ attack\ ports\ ) + f_v(vulnerablity\ pkgs)}$$

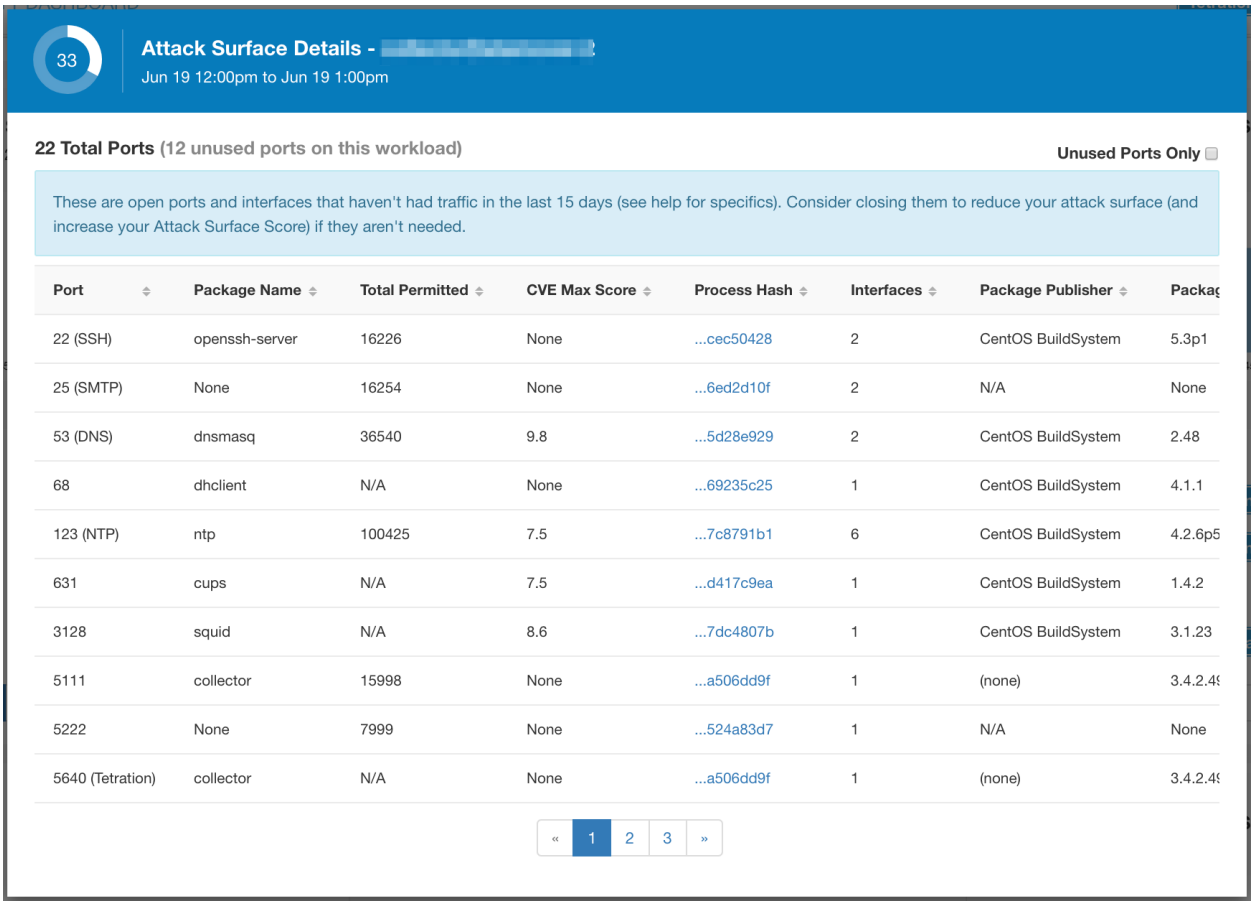
$$f_v = \max\left(\left\{cve_{score} = \begin{cases} CVSS_{V3}, & v3\ exist \\ CVSS_{V2}, & v3\ not\ exist \end{cases}\right\}\right)$$

Fig. 14.5.3.2: Attack Surface Score Formula

Laplace smoothing is used with a penalty factor based on heuristic data. Score is computed daily with the past 2 weeks of data.

Tenant score is average of workload scores in the scope. Improve the score by identifying workload/scopes with unused open ports, and closing the unused ports.

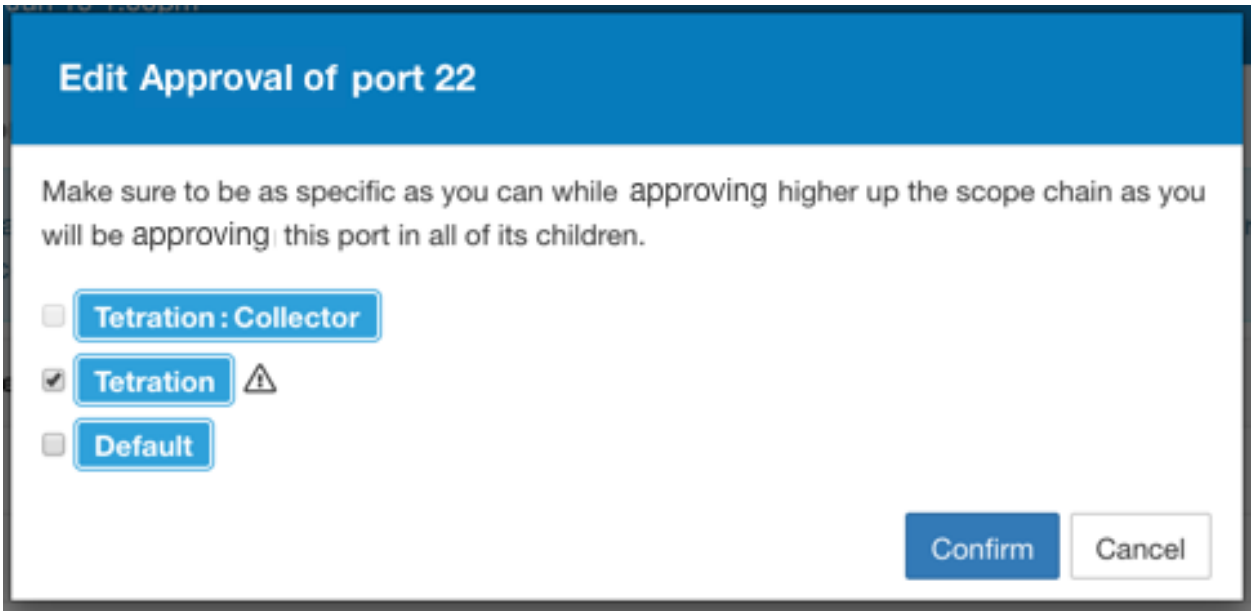
When a workload link is clicked an attack surface modal is opened with details on all available ports and interfaces within the context of that workload.



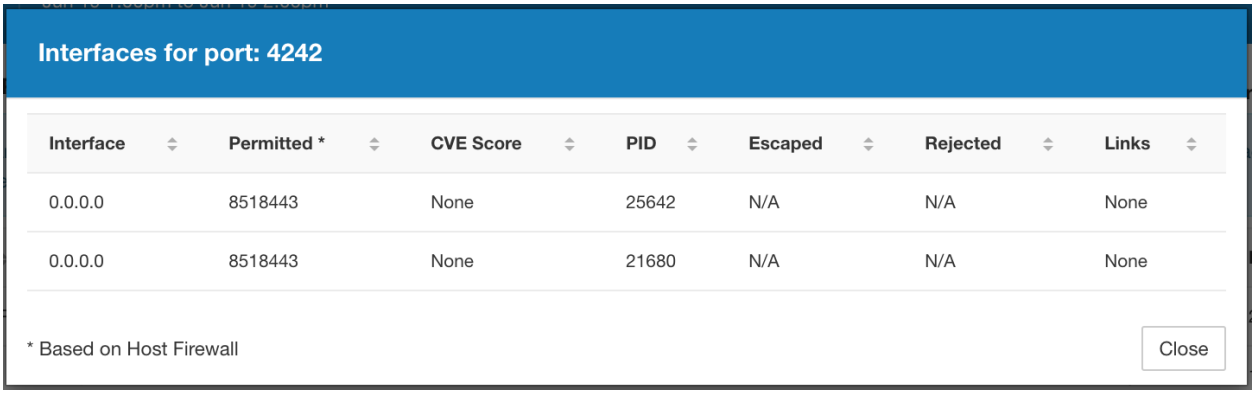
Features:

- Unused Ports Only: checkbox that when toggled filters out the ports that are used and only shows you the unused ports associated with the workload.
- Columns: Approved, port, package name, total permitted, CVE Max Score, Process Hash, Interfaces, Package Publisher, Package Version, Total Escaped, Total Rejected, Commonly Hacked Port, Links.
- Interfaces: If you click on any one of the line items in the Attack Surface table you can view the interfaces that are associated with each port inside of a modal. *please see screenshot below*
- Approved: checkbox that when toggled, allows you to intentionally set an "unused port" as "approved" on any one of the scopes on the scope chain that that workload has access to. Note: if a port is approved on a scope and that port is not explicitly approved on any of the children (if that scope has children), then the scope checkboxes are disabled as it is implied that any child scope that the parent scope has access to already is approved in that chain. *please see screenshot below*

Approval Modal:



Interfaces Modal:



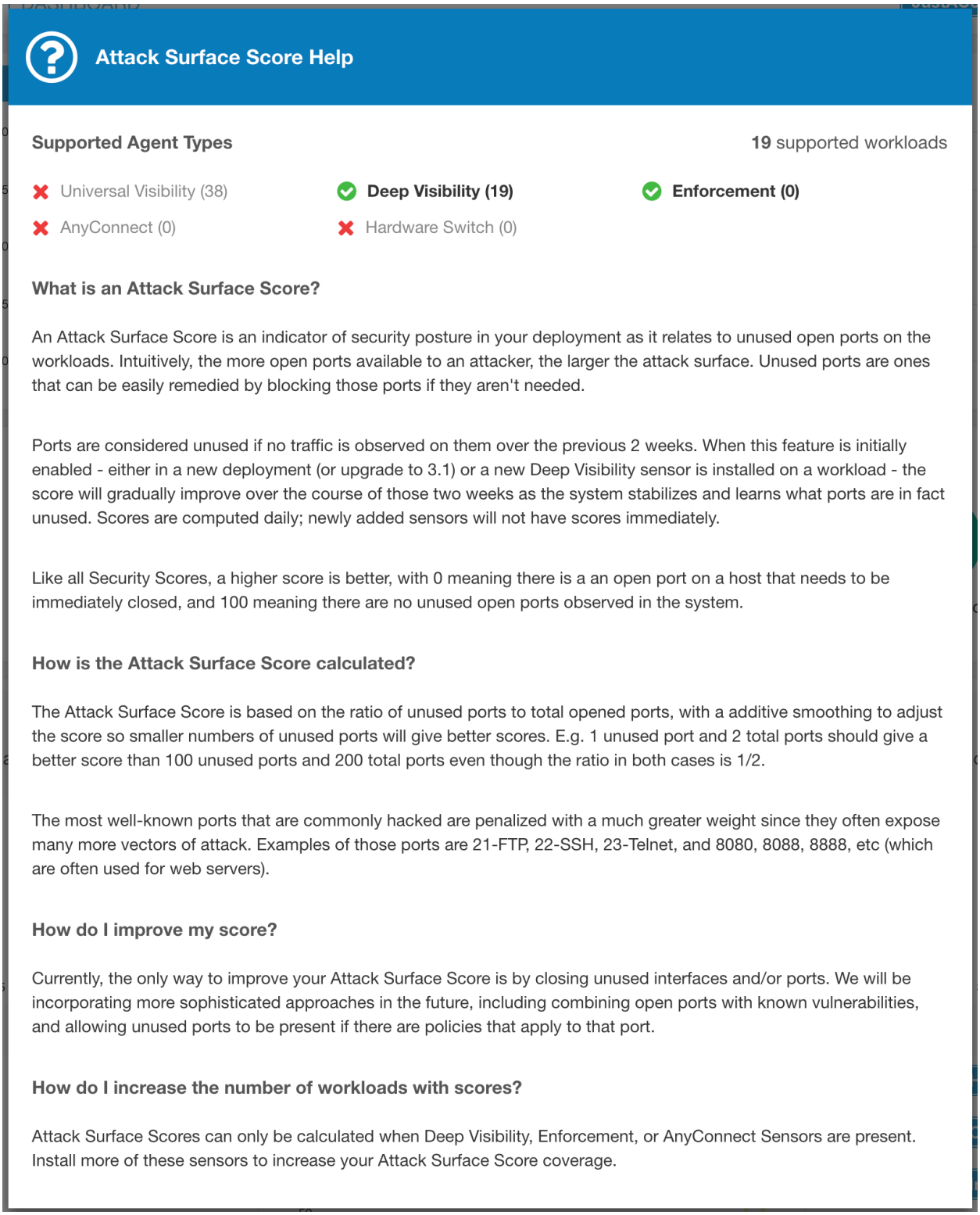


Fig. 14.5.3.3: Help for Attack Surface Score

14.5.4 Forensics Score

Severity of Forensics events on workloads is used for computing the scores.

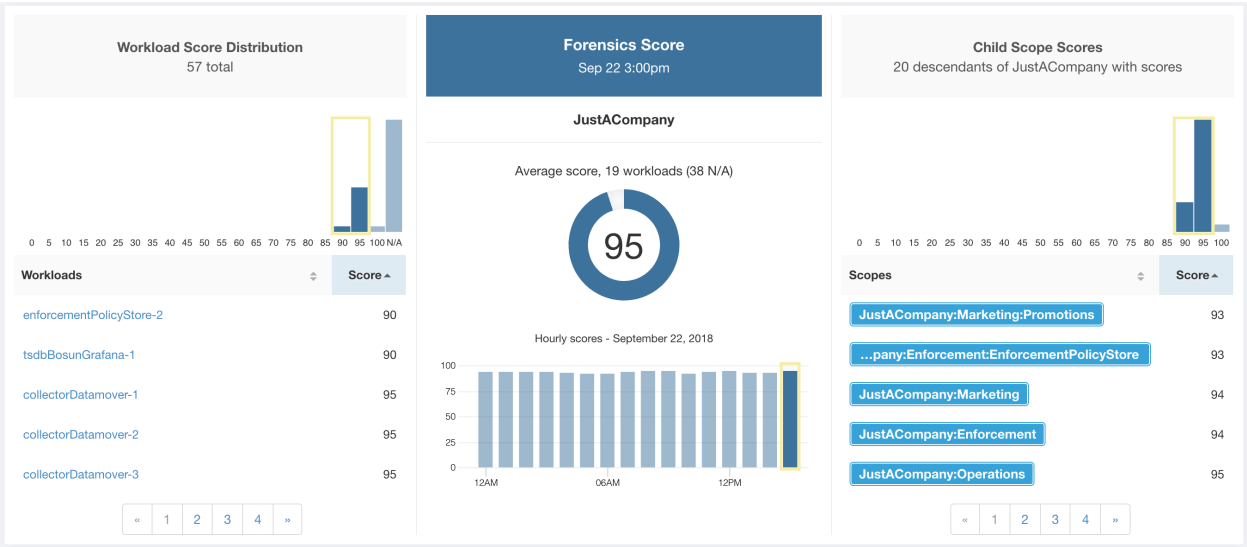


Fig. 14.5.4.1: Forensics Score Details

Lower score indicates:

- One or more forensics events were observed on the workload
- Or one/more forensics rules are noisy and/or incorrect

To improve the score:

- Fix the issue if any to reduce the chances of exposures/exploits
- Tweak forensics rules to reduce noise and false alarms

Forensics score for a workload is inverse function of total impact score of forensics events. Higher is the total impact score of forensics events, lower is the forensics score.

Severity	Impact Score
IMMEDIATE_ACTION	100
CRITICAL	10
HIGH	5
CRITICAL	3

$$forensics\ score = max\left(0, \left(100 - \sum forensics\ event\ impact\ score\right)\right)$$

Fig. 14.5.4.2: Forensics Score Formula

Refer to *Forensics* for more details.

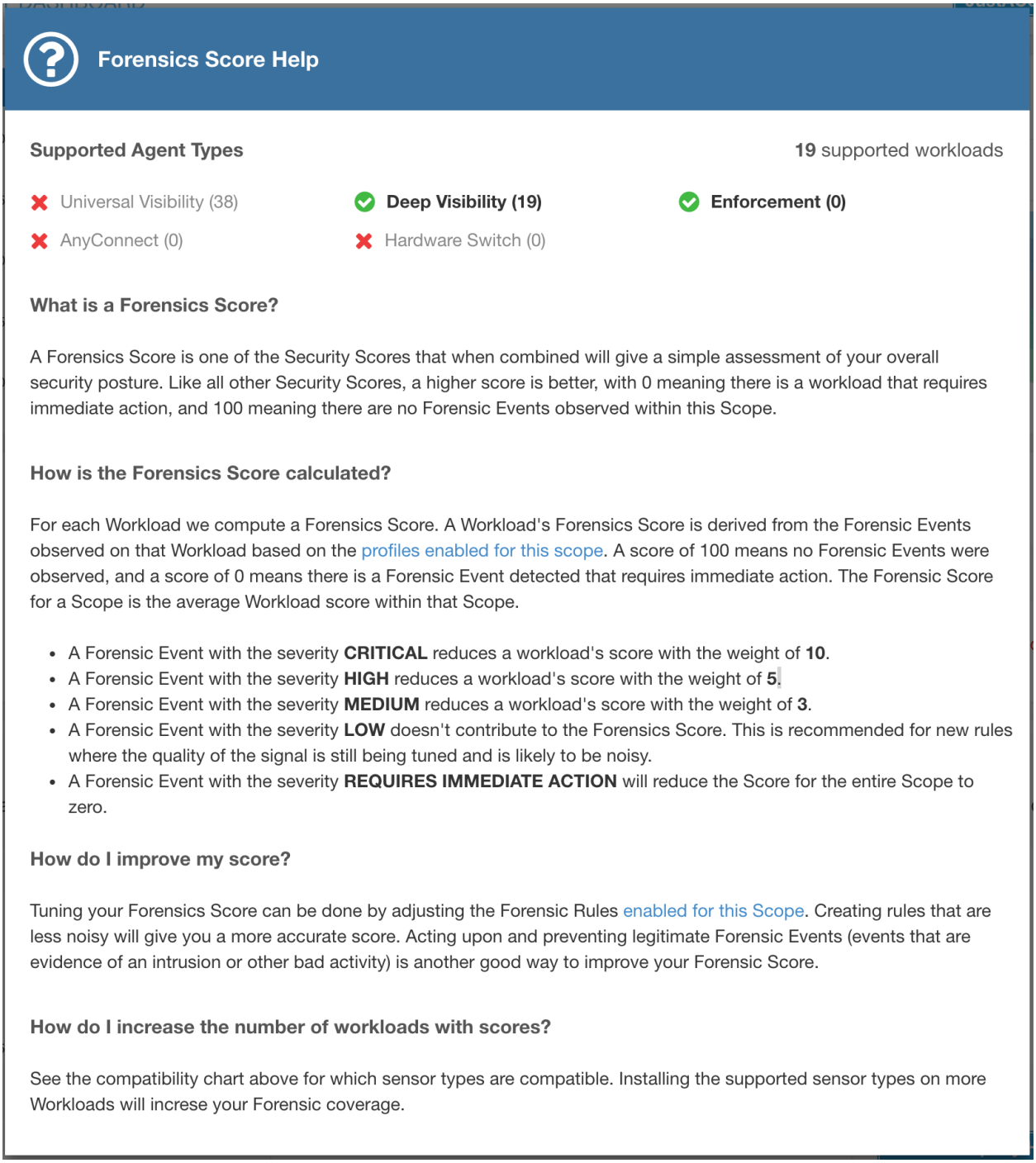


Fig. 14.5.4.3: Help for Forensics Score

14.5.5 Network Anomaly Score

Severity of Network Anomaly events on workloads is used for computing the scores.

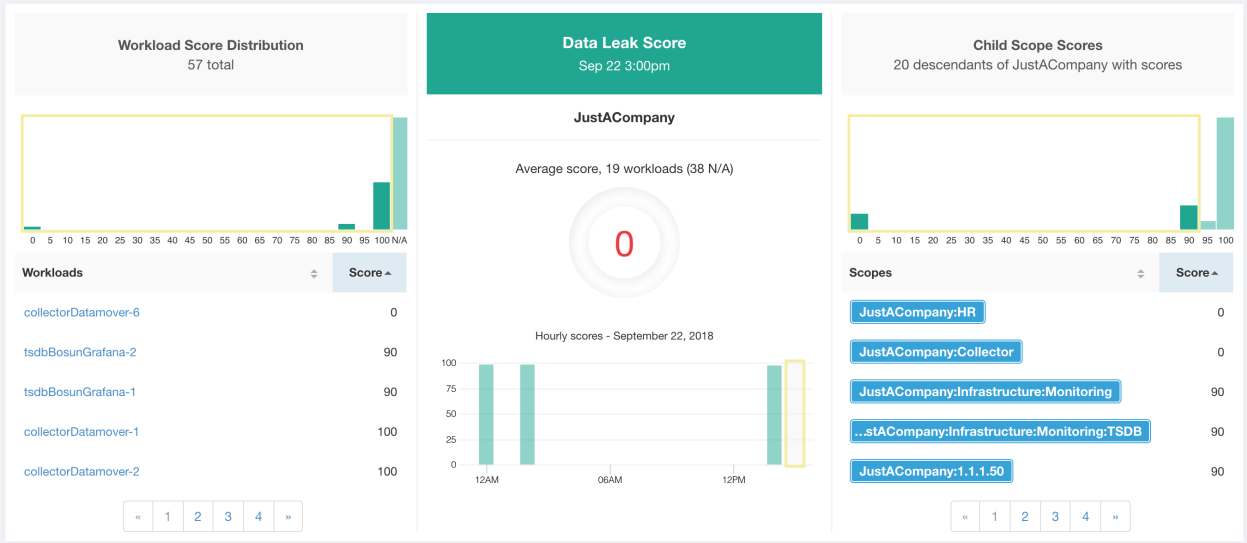


Fig. 14.5.5.1: Data Leak Score Details

Lower score indicates:

- Unusually high amount of data is being transferred out of workloads
- Or Network Anomaly forensic rule is incorrect or noisy

To improve the score:

- Fix the issue if any to reduce the chances of data exfiltration
- Adjust Network Anomaly rules to reduce noise and false alarms

Network Anomaly score for a workload is inverse function of total severity score of Network Anomaly events. Higher is the total severity score, lower is the Network Anomaly score.

Severity	Score
IMMEDIATE_ACTION	100
CRITICAL	10
HIGH	5
CRITICAL	3

$$data\ leak\ score = max\left(0, \left(100 - \sum data\ leak\ event\ severity\ score\right)\right)$$

Fig. 14.5.5.2: Data Leak Score Formula

Refer to *PCR-based Network Anomaly detection* for more details.

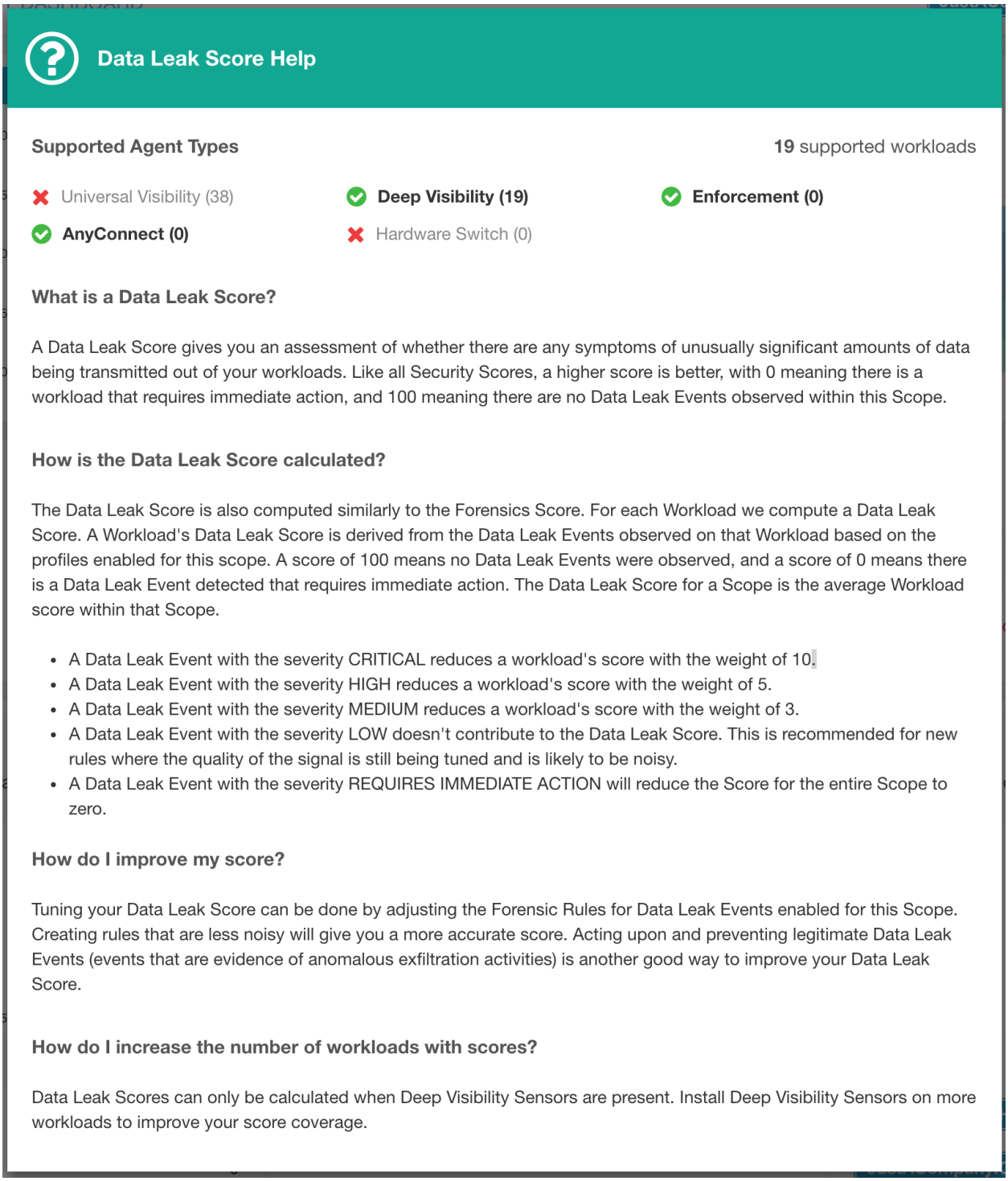


Fig. 14.5.5.3: Help for Data Leak Score

14.5.6 Segmentation Compliance Score

Segmentation Compliance Score presents a top-level view of policy violations and emphasizes which scopes and applications have the most violations.

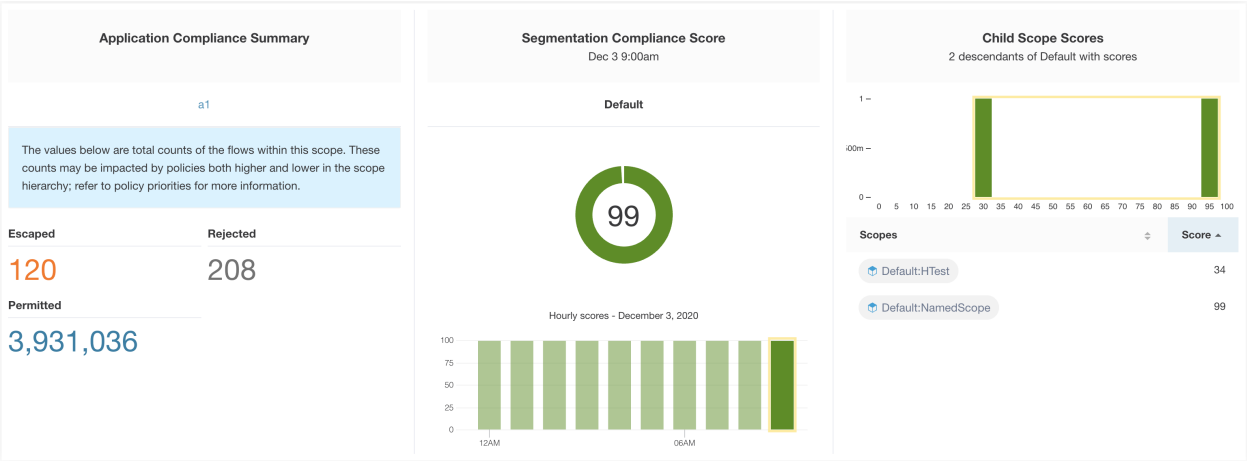


Fig. 14.5.6.1: Segmentation Compliance Score Details

Note: Escaped/Rejected/Permitted count displayed on security dashboard for root scope does not add up to all the counts respectively displayed for all child scopes. Escaped/Rejected/Permitted count is an evaluation on the policy and not just on source or destination.

Lower score indicates:

- Significant number of escaped flows (policy violations) relative to permitted
- Score will be 0 when more escaped flows than permitted

Segmentation Compliance Score is computed for scopes with an enforced primary workspace. For scopes without enforced applications, the score will be computed as the average of descendant scope scores with enforced policies.

Score is computed by using the ratio between escaped and permitted.

$$compliance\ score = \left\lfloor 100 - \frac{100 \times escaped}{permitted} \right\rfloor$$

Fig. 14.5.6.2: Segmentation Compliance Score Formula

Improve score by reducing number of policy violations

- Verify policies correctly cover desired behavior

- Verify policies are correctly being enforced

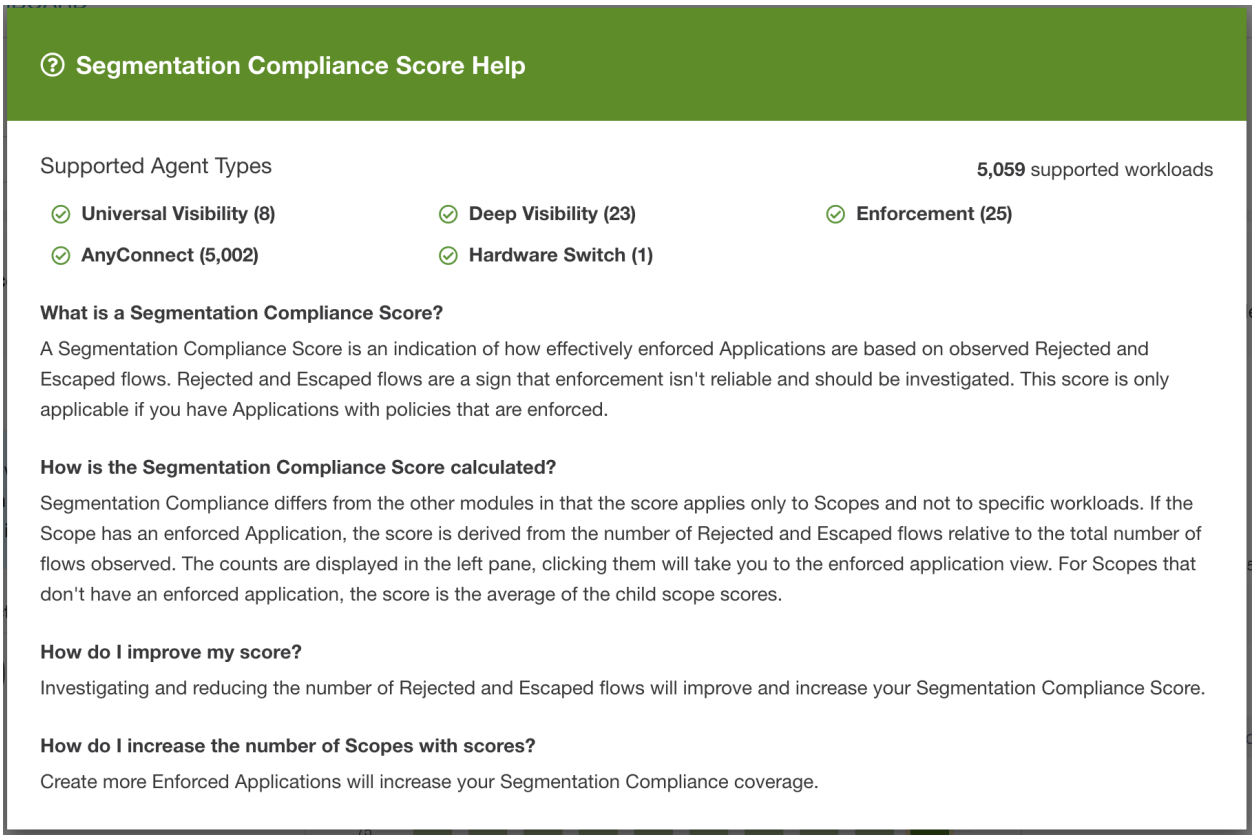


Fig. 14.5.6.3: Help for Segmentation Compliance Score Details

CHAPTER

FIFTEEN

VULNERABILITY DASHBOARD

Vulnerability Dashboard enables end users to focus their effort on critical vulnerabilities and workloads that need most attention. Users can select relevant scope at the top of this page as well as select the scoring system for vulnerabilities they want to view (Common Vulnerability Scoring System v2 or v3). The new page highlights the distribution of vulnerabilities in the chosen scope as well as displays vulnerabilities by different attributes, e.g. complexity of exploits, can the vulnerabilities be exploited over the network or does attacker need local access to the workload. Furthermore, there are statistics to quickly filter out vulnerabilities that are remotely exploitable and have lowest complexity to exploit.

There are three tabs that are available on this page – all of them adjust/filter based on user's click(s) on the widgets at the top of the page:

- CVEs tab highlights the vulnerabilities to focus on in the chosen scope.
- Packages tab shows the end users the packages that need to be patched.
- Workloads tab lists the workloads that need most attention in terms of patching in the chosen scope.

Clicking on any row in the above tabs display more information about that row, e.g. clicking on package row in the packages tab show which workloads that package/version is installed on and the associated vulnerabilities for that package. Similarly, clicking on the row in workloads tab shows packages installed on the chosen workload along with the associated vulnerabilities.

This page is intended to help the users identify workloads to focus on first and which packages to patch first.

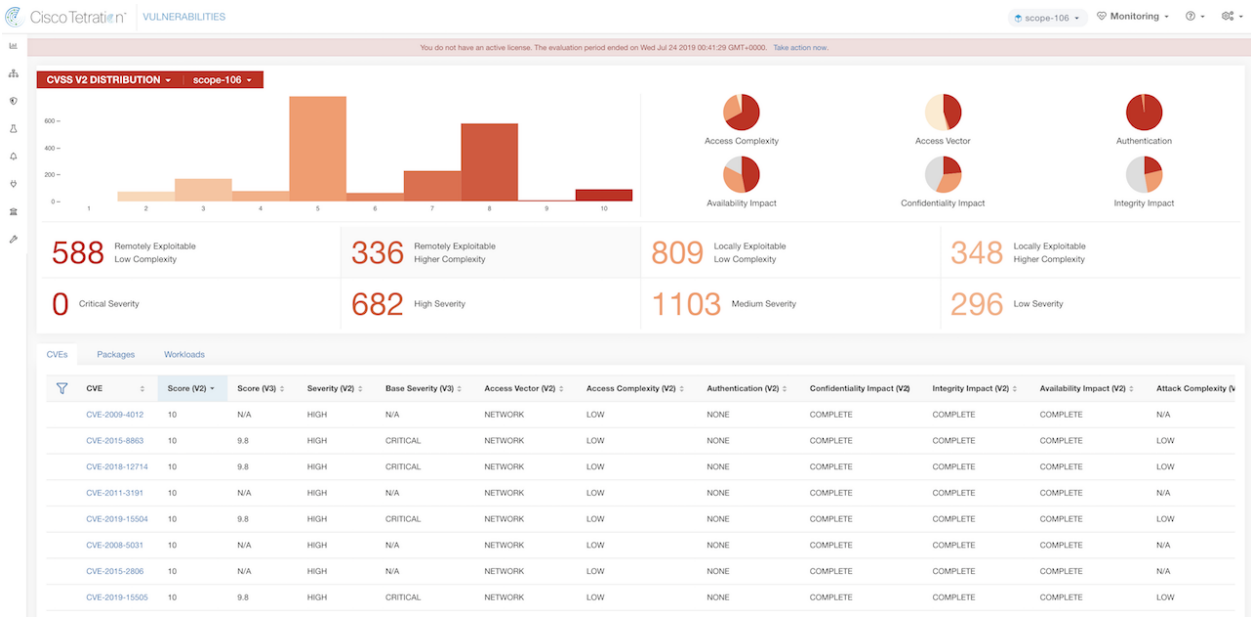


Fig. 15.1: Vulnerability dashboard

15.1 CVEs tab

Based on the scope selected at the top of the page as well as the scoring system (v2 or v3), CVE tab highlights the vulnerabilities (sorted by the scores) on workloads in the selected scopes that need attention.

For each CVE, besides basic impact metrics, exploit information based on our threat intelligence is displayed:

- Exploit Count: number of times CVE was seen exploited in the wild in the last year
- Last Exploited: last time CVE was seen exploited in the wild by our threat intelligence

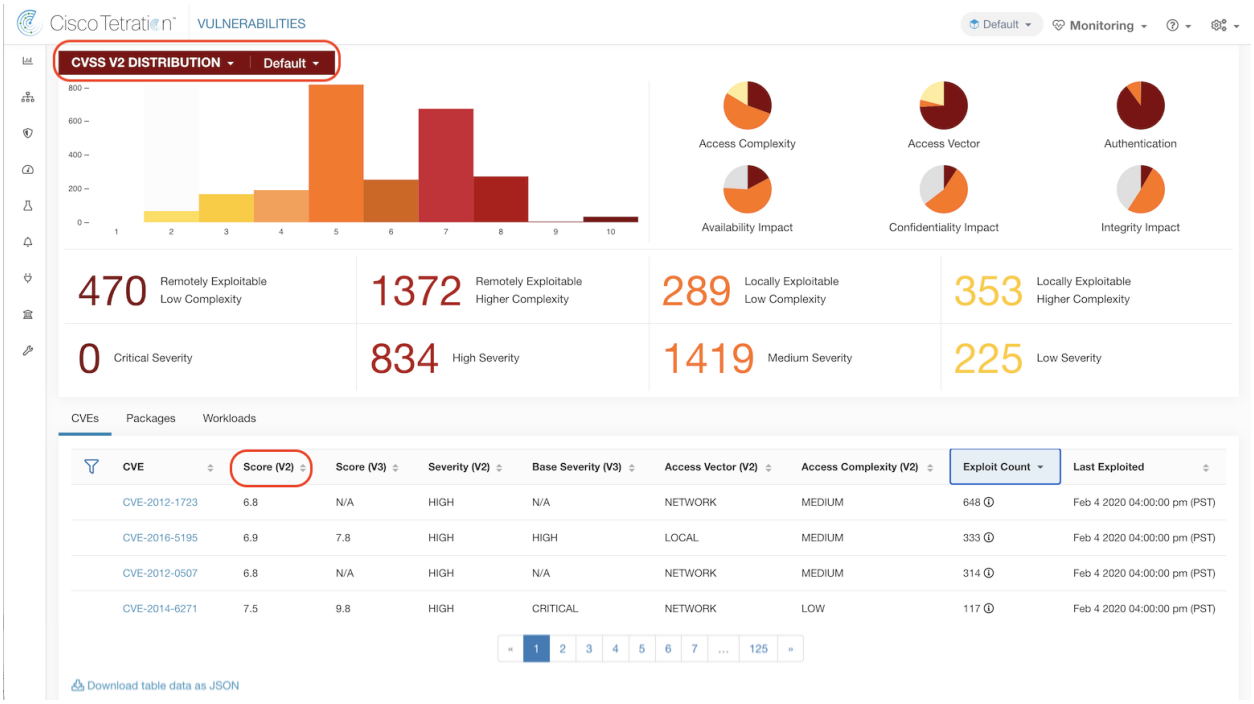


Fig. 15.1.1: CVEs tab listing vulnerabilities in specified scope

Clicking on any row in the CVEs table gives more details about that vulnerability and which workloads it affects.

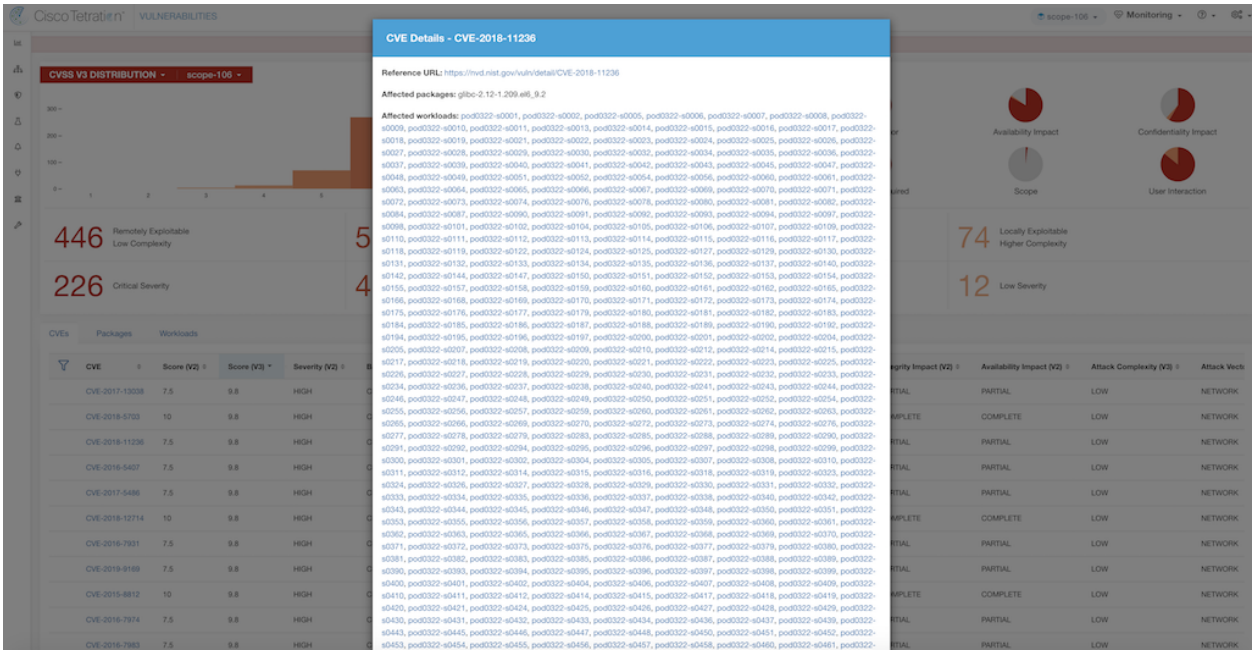


Fig. 15.1.2: Details for a CVE

15.2 Packages tab

Packages tab lists the software packages that users need to pay attention to and potentially upgrade in order to reduce their attack surface.

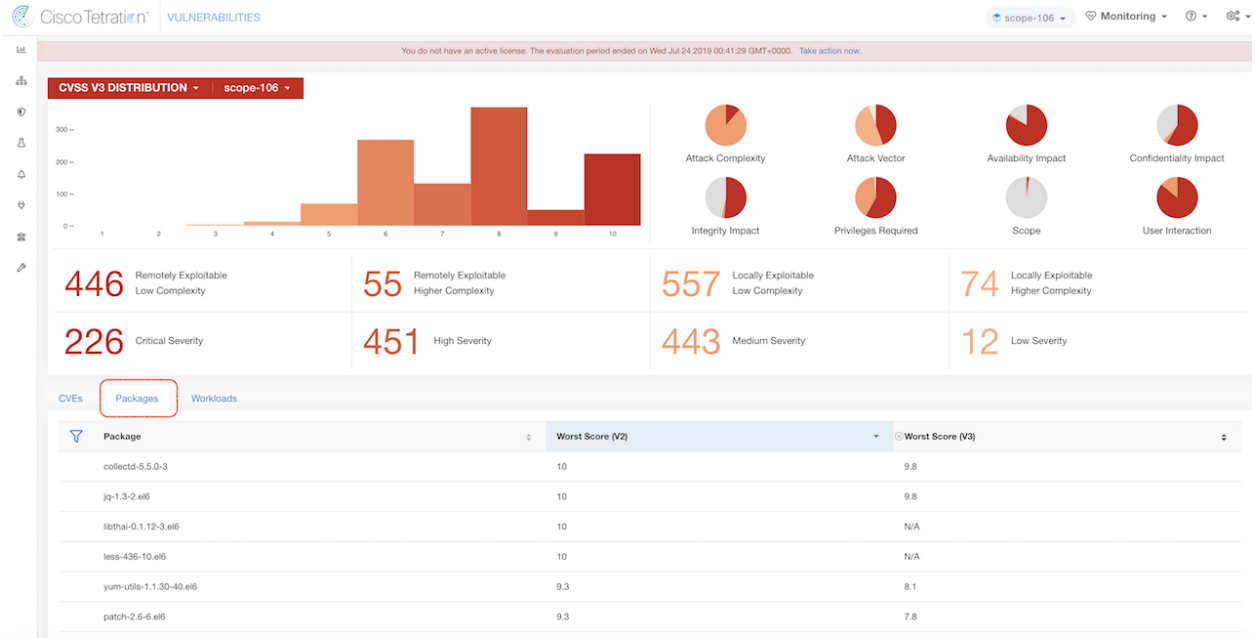


Fig. 15.2.1: Packages tab listing vulnerable software in specified scope

Clicking on any row in the packages table gives more details about which workloads that package is installed as well as the known CVEs for that package.

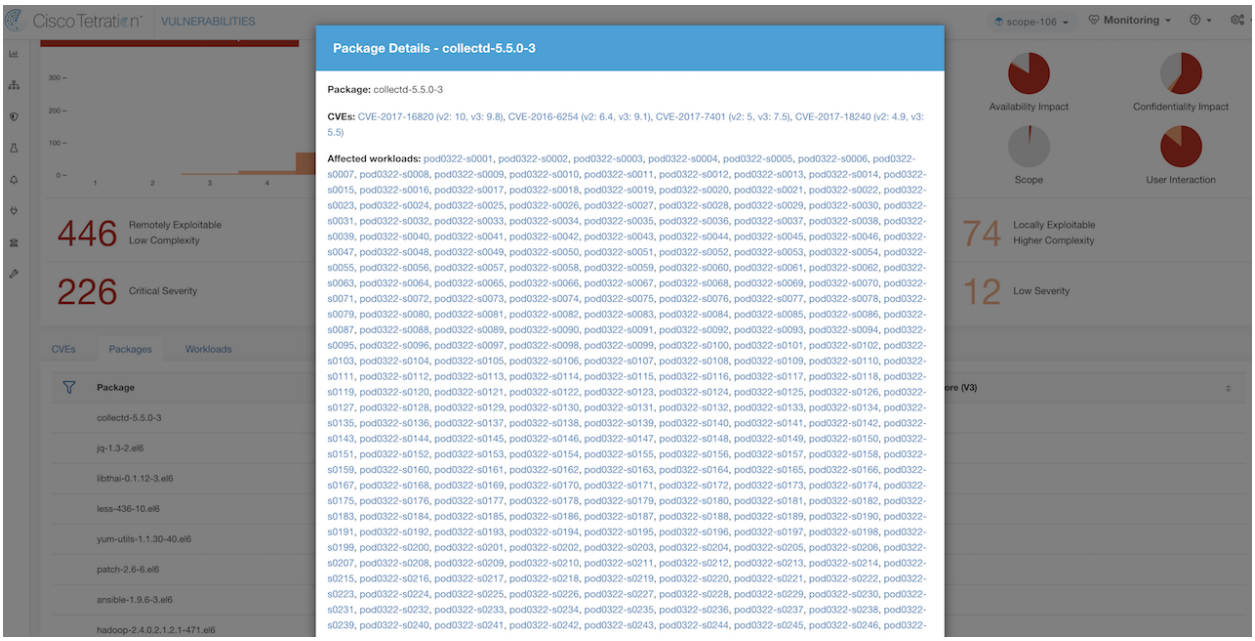


Fig. 15.2.2: Details of vulnerabilities and affected workloads for a package

15.3 Workloads tab

Workloads tab lists the workloads that need attention in terms of software updates or patches.

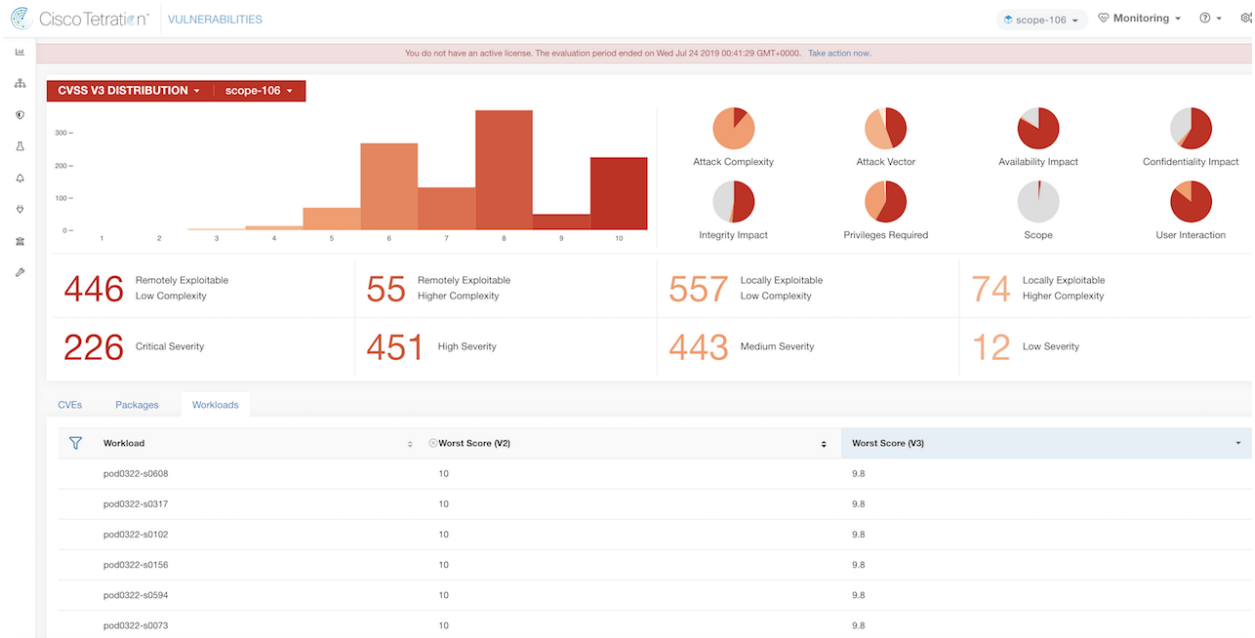


Fig. 15.3.1: Workloads tab listing vulnerable workloads in specified scope

Clicking on any row in the workloads table provides the list of packages with vulnerabilities on that workload.

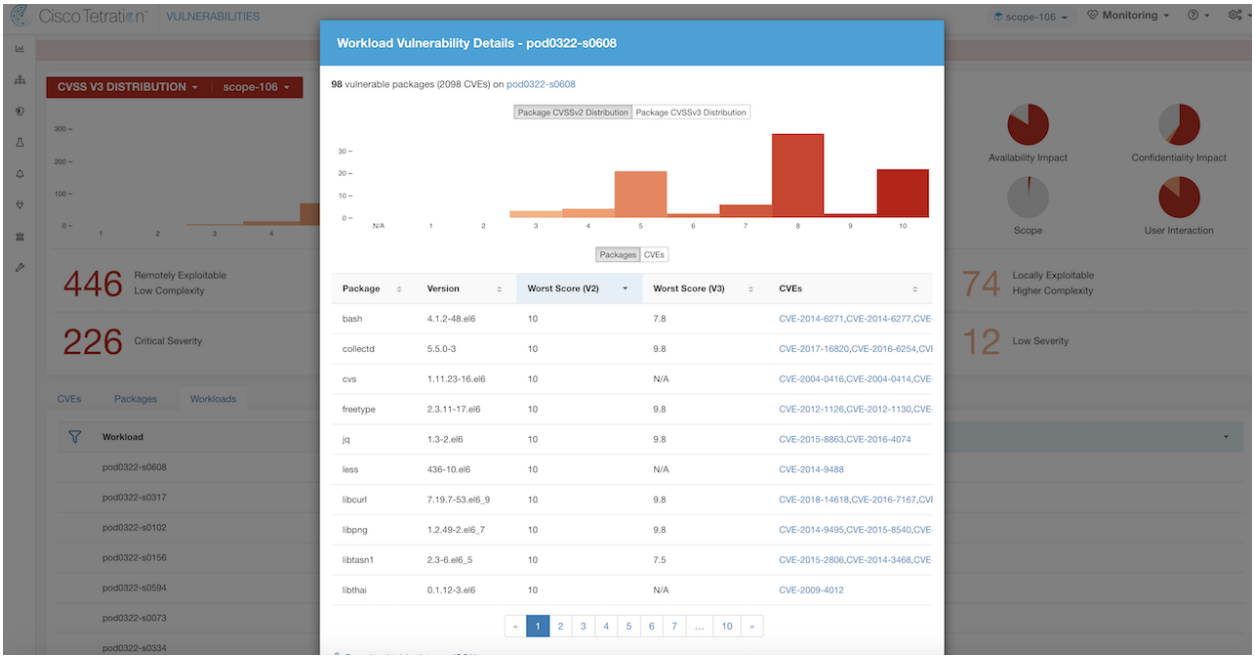


Fig. 15.3.2: Details of vulnerabilities for a workload

All of the above tables are downloadable by the user using the Download links at the bottom of the tables.

CHAPTER

SIXTEEN

SETTINGS

The **Settings** drop-down menu options can be accessed using the **Settings menu** icon in the far upper-right. The options include **Preferences**, **Company**, **Users**, **Tenants**, **Collectors**, and **Agent Config**. The options available in the menu vary depending on your role. For example, only users with **Site Admin** and **Customer Support user** role can see the **Users** option.

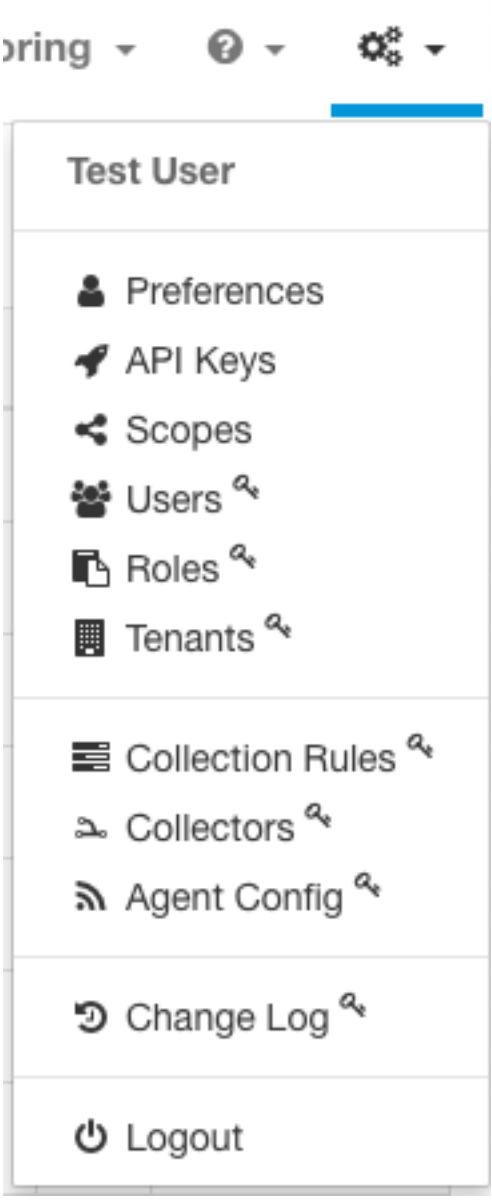


Fig. 16.1: Settings Drop-down Menu

This section explains how to specify preferences, such as changing a password or enabling/disabling two-factor authentication, how to manage user accounts, company/cluster attributes, tenants, and commission/decommission collectors.

16.1 Agent Config

16.1.1 Hardware Agent Config

Tetration switch agent (TaAgent) runs in the guestshell on the switch and acts as a proxy between Tetration Configuration Controller and the switch. This agent can program the switch to start sending analytics related exports to the Tetration cluster.

16.1.1.1 Obtaining TaAgent for a specific cluster

The TaAgent rpm is created for every release of the Tetration software package (mother_rpm) and is modified for every cluster that it is installed or upgraded on. These modifications mainly consist of the following changes:

- Providing hw_cfg_agent's parameters such as its ip address and port number it will be listening for TaAgents' to register
- Providing authentication information to the rpm such that TaAgent downloaded for one cluster can't connect to a different cluster.

This cluster specific TaAgent rpm file is available for download from the **Hardware Agent Download** tab of the **Agent config** page.

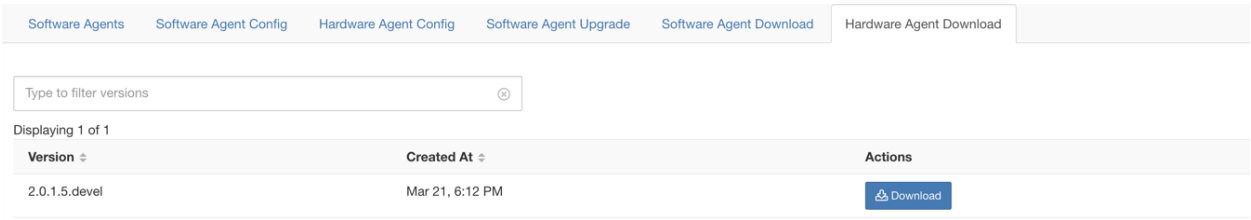


Fig. 16.1.1.1.1: Hardware Agent Download

16.1.1.2 Installation and Configuration for Standalone NXOS

Installing TaAgent

Once downloaded, the rpm needs to be transferred over to the switch using wget/scp. To install the agent on a standalone switch, follow the steps below:

1. First time on a given switch:

- `#guestshell resize rootfs 400` < wait for 30 seconds or so>
- `#guestshell disable` < wait for a minute or so>
- `#guestshell enable`

2. Enter guestshell by typing:

- `#guestshell`

3. Download the RPM via wget or scp and install the rpm:

- `[guestshell@guestshell ~]$ sudo rpm -ivh <file name>`

4. To uninstall an existing version of ta_agent:

- `[guestshell@guestshell ~]$ sudo rpm -e tet-agent-site`

Setting up access and connectivity

To be able to communicate to the cluster, we need to configure the switch with VRF and source interface.

`EXPORTER_SRC_INTERFACE` is the switch interface used for exporting flow info to the cluster, and `VRF` is the name of the vrf for the exporter source interface:: Enter code below in the NXOS switch configuration terminal:

- `Switch(config)# analytics cluster tetration vrf VRF srcIf EXPORTER_SRC_INTERFACE`

You should restart TaAgent after making changes.

Starting and Stopping the service

Normally users are not required to start/stop TaAgent. Upon install of RPM the agent starts immediately. We have a init.d process that checks whether the agent is running periodically and restart it if needed. However it can be started or stopped using the following commands:

Starting:

- `[guestshell@guestshell ~]$ sudo systemctl start ta_agent`

Stopping:

- `[guestshell@guestshell ~]$ sudo systemctl stop ta_agent`

Restarting:

- `[guestshell@guestshell ~]$ sudo systemctl restart ta_agent`

16.1.1.3 Installation and Configuration for ACI

Uploading Cisco Tetration Switch Agent

1. Log into APIC.
2. Click Admin > Firmware > Firmware Repository.
3. From the ACTIONS drop-down menu, click Upload Firmware to APIC.
4. Choose the RedHat Package Manager (RPM) file downloaded from Tetration UI.
5. Click and Submit.

At this point hardware sensor RPM binary is available to be installed on all the switches in the Cisco ACI cluster.

Configuring Analytics Policy

1. Define an analytics policy by choosing Fabric > Fabric Policies > Analytics Policies.
2. From the ACTIONS drop-down list, choose Create Analytics Policy. The Create Analytics Policy dialog box appears.
3. Enter the following values in the Create Analytics Policy dialog box:

- Enter the cluster name in the Cluster field.
- Enter the server name in the Name field.
- Enter the destination IP address of the cluster in the IP field. Use Tetration Web UI IP address.

- Use the up and down arrows on the stepper to choose the destination port.
- (Optional) Click the DSCP drop-down arrow and choose the Differentiated Services Code Point.
- Click Submit.

An Analytics Policy is created.

Setting Fabric Node Controls

1. Define an fabric node control by choosing Fabric > Fabric Policies > Switch Policies > Policies > Fabric Node Controls.
2. From the ACTIONS drop-down list, choose Create Fabric Node Control. The Create Fabric Node Control dialog box appears.
3. Enter the following values in the Create Fabric Node Controls dialog box:

- Enter the name in the Name field.
- (Optional) Enter description if required in the Description field.
- In the Feature Selection field, make sure "Analytics Priority" is selected.
- Click Submit.

Fabric Node Control is set for Tetration telemetry export.

Defining Leaf Switch Policy Group

1. From the Policies pane, choose Switch Policies > Policy Groups.
2. From the ACTIONS drop-down list, choose Create Leaf Switch Policy Group. The Create Leaf Switch Policy Group dialog box appears.
3. Enter the following values in the Create Leaf Switch Policy Group dialog box:

- Enter the policy group name in the Name field.
- Click the Node Control Policy drop-down arrow and choose the fabric node control policy.
- Click the Analytics Policy drop-down arrow and choose the analytics policy.
- Click Submit.

A Policy Group is created.

Creating Leaf Switch Profile

1. In the Policies pane, click Profiles.
2. From the ACTIONS drop-down list, choose Create Leaf Switch Profile. The Create Leaf Switch Profile dialog box appears.
3. Enter the following values in the Create Leaf Switch Profile dialog box:

- Enter a name in the Name field.
- Click the + icon.
- Enter a value in the Name field.

- Click the Blocks drop-down arrow and check the check box for the leafs to which you want to push this policy.
- Click the Policy Group drop-down and select the switch policy group.
- Click Update and Submit.

The configuration is pushed to the selected leafs.

16.1.1.4 Configuring TaAgent

Configure the following two parameters on a hardware agent via **Hardware Agent Config** page.

- **Export Interval**: Specifies the interval at which the hardware agent should export flow info to collectors

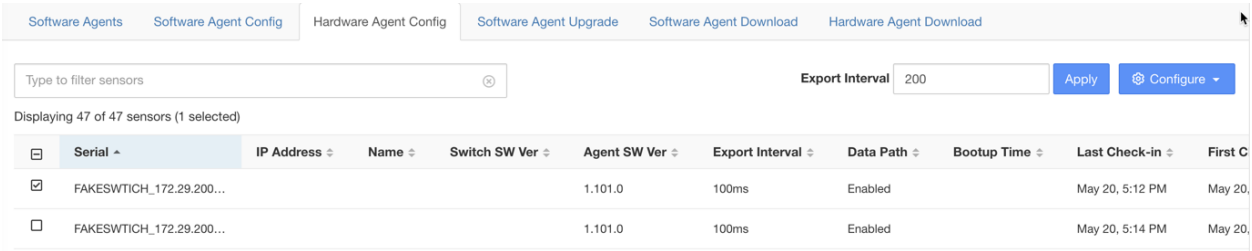


Fig. 16.1.1.4.1: Export Interval

- **Data Path** Enable/Disable: Disable all exports to the collectors altogether

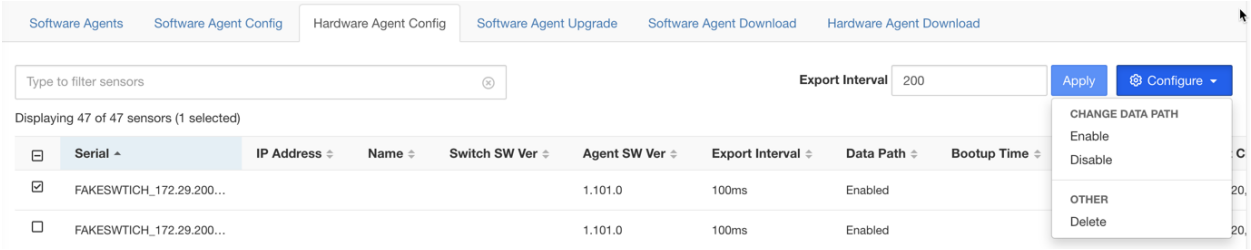
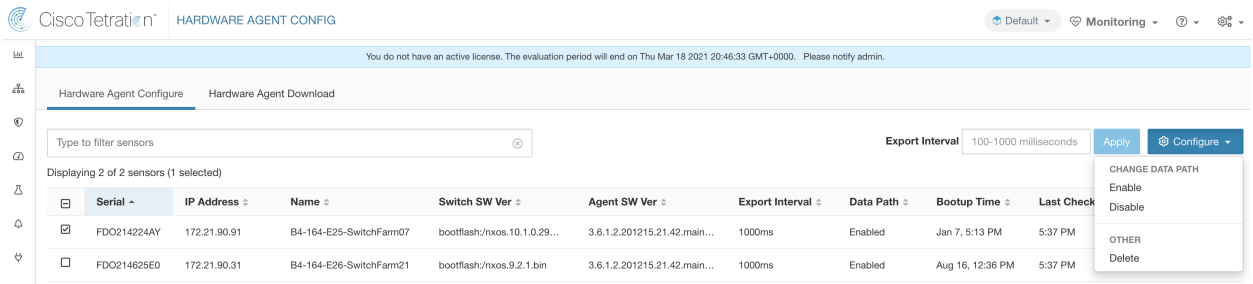


Fig. 16.1.1.4.2: Data Path

16.1.1.5 Removing TaAgent from Tetration UI

To remove the TaAgent from the Tetration UI after it has been removed/uninstalled from the switch/ACI:

1. Go to Settings->Hardware Agent Configure
2. Select the Hardware Agent Configuration tab
3. Click the box next to the switch to be removed
4. click on configure in the top right corner and select Delete in the drop down menu.



16.1.2 Software Agent Config

Site Admins and **Customer Support users** can access the **Agent Config** page under the **Settings menu**. Various agent attributes that dictate how a agent collects data can be changed using the **Agent Config** page. This section explains how to view software agent information and how to install, upgrade and configure agents.

16.1.2.1 Configuring Software Agents

Software agents are configured by creating **Agent Config Intents** that associate an **Agent Config Profile** with either an **Inventory Filter** or a **Scope**. The first matching intent will be applied to each agent. There is always a default agent config in Tetration deployment which is applied to all sensors that are not associated with any specific config profile.

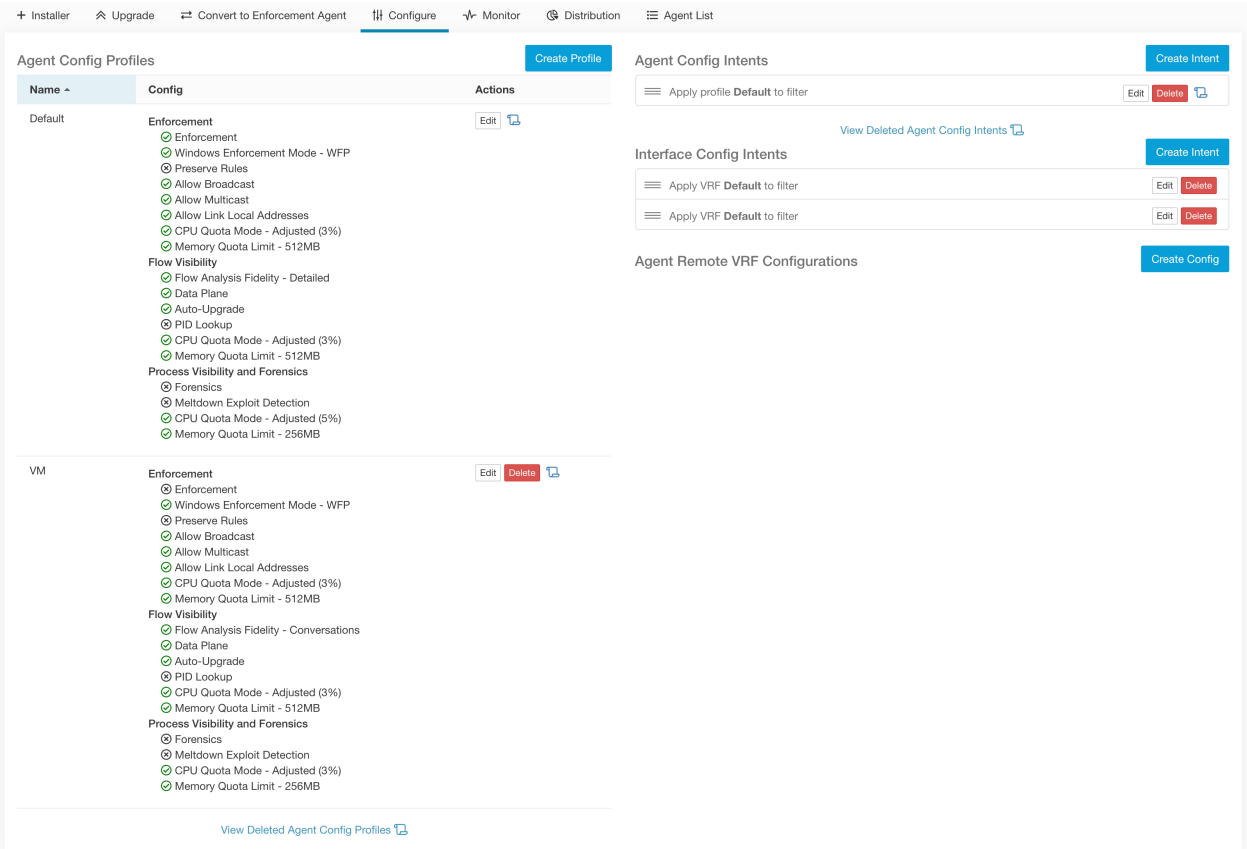


Fig. 16.1.2.1.1: Software Agent Config Page

Creating an Agent Config Profile

1. Click the **Settings menu** in the top-right corner.
2. Select **Agent Config**. The **Software Agents** page displays.
3. Select the **Software Agent Config** tab.
4. Click the **Create Profile** button.
5. Enter a name for the profile (required) and select a scope where profile will be available.
6. Enter the appropriate values in the fields listed in the tables below:

Note that in the tables below, the option marked with (*) is the default setting.

Table 16.1.2.1.1: Enforcement config

Field	Description
Enforcement	**Enable**-Enable policy enforcement on the agent. **Disable(*)**-Do not enable policy enforcement on the agent.

Continued on next page

Table 16.1.2.1.1 – continued from previous page

Preserve Rules	**Enable**-Preserves any existing firewall rules on agent. **Disable(*)**-Clears existing firewall rules before applying enforcement policy rules from Tetration.
Allow Broadcast	**Enable(*)**-Adds rules to the firewall to allow broadcast traffic on the workload. **Disable**-Does not add any rule. Broadcast traffic will be dropped if default policy is deny on Agent.
Allow Multicast	**Enable(*)**-Adds rules to the firewall to allow multicast traffic on the workload. **Disable**-Does not add any rule. Multicast traffic will be dropped if default policy is deny on Agent.
Allow Link Local	**Enable(*)**-Adds rules to the firewall to allow link local addresses' traffic on the workload. **Disable**-Does not add any rule. Multicast traffic will be dropped if default policy is deny on Agent.

Continued on next page

Table 16.1.2.1.1 – continued from previous page

CPU Quota Mode for enforcement process	**Adjusted(*)**-The CPU limit is adjusted according to the number of CPUs on the system. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system, selecting this mode means that agent is allowed to use a total of 30% (measured by **top**). **Top**-The CPU limit value would match the **top** view on average. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system. The cpu usuage would still be 3%. This is a fairly restrictive mode and should be used only when necessary. **Disable**-The CPU limit feature is disabled. The agent will use CPU resources permitted by the OS. See agent_cpu_sla.pdf for more information.
CPU Quota Limit (%)	Specify the actual limit in percentage of the system processing power the agent can use.
Memory Quota Limit (MB)	Specify the memory limit in MB that the process is allowed to use. If the process hits this limit, it will restart.
Windows Enforcement Mode	**WFP**-Enable Windows Filtering Platform for enforcement on Windows agents. **WAF(*)**-Enable Windows Advanced Firewall for enforcement on Windows agents.

Table 16.1.2.1.2: Flow Visibility config

Field	Description

Continued on next page

Table 16.1.2.1.2 – continued from previous page

Data Plane	**Enable(*)**-Enable the agent to send reports to the cluster. **Disable**-Disable the agent's reports.
Auto-Upgrade	**Enable(*)**-Automatically upgrade the agent when a new package is available. **Disable**-Do not automatically upgrade the agent.
PID Lookup	**Enable**-Enable PID lookups on the agent. When enabled, the agent will make best-effort attempts to associate network flows with running processes in the workload. This operation might be expensive, therefore the agent will throttle the number of operations done in each export cycle to keep the CPU overhead under control. It is possible that some flows are not asssociated with any processes even when the config is enabled. **Disable(*)**-Do not enable PID lookups on the agent.

Continued on next page

Table 16.1.2.1.2 – continued from previous page

CPU Quota Mode	**Adjusted(*)**-The CPU limit is adjusted according to the number of CPUs on the system. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system, selecting this mode means that agent is allowed to use a total of 30% (measured by **top**). **Top**-The CPU limit value would match the **top** view on average. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system. The cpu usuage would still be 3%. This is a fairly restrictive mode and should be used only when necessary. **Disable**-The CPU limit feature is disabled. The agent will use CPU resources permitted by the OS. See agent_cpu_sla.pdf for more information.
CPU Quota Limit (%)	Specify the actual limit in percentage of the system processing power the agent can use.
Memory Quota Limit (MB)	Specify the memory limit in MB that the process is allowed to use. If the process hits this limit, it will restart.
Flow Analysis Fidelity	**Conversations**-Enable conversations mode on all sensors. **Detailed(*)**-Enable detailed mode on all sensors

Table 16.1.2.1.3: Process Visibility and Forensics config

Field	Description

Continued on next page

Table 16.1.2.1.3 – continued from previous page

Forensics	**Enable**-Enable forensics on the agent. Note that this feature may consume additional CPU cycles specified in the **CPU limit** below. For example, if the cpu limit is 3% and this feature is enabled, the agent assumes it could use up to 6% in total. **Disable(*)**-Disable forensics on the agent.
Meltdown Exploit Detection	**Enable**-Enable Meltdown exploit detection on the agent. This feature requires Forensics to be enabled. For more information, see **Side Channel** in *Compatibility*. **Disable(*)**-Disable Meltdown exploit detection on the agent.
CPU Quota Mode	**Adjusted(*)**-The CPU limit is adjusted according to the number of CPUs on the system. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system, selecting this mode means that agent is allowed to use a total of 30% (measured by **top**). **Top**-The CPU limit value would match the **top** view on average. For example, if the CPU limit is set to 3% and there are 10 CPUs in the system. The cpu usuage would still be 3%. This is a fairly restrictive mode and should be used only when necessary. **Disable**-The CPU limit feature is disabled. The agent will use CPU resources permitted by the OS. See agent_cpu_sla.pdf for more information.

Continued on next page

Table 16.1.2.1.3 – continued from previous page

CPU Quota Limit (%)	Specify the actual limit in percentage of the system processing power the agent can use.
Memory Quota Limit (MB)	Specify the memory limit in MB that the process is allowed to use. If the process hits this limit, it will restart.

7. Click **Save**.

Create Profile

Name	Enter a name (required)
Ownership Scope	Tetration

Enforcement

Enforcement	○ Enable ◉ Disable (Default)
Windows Enforcement Mode	◉ WAF (Default) ○ WFP BETA
Preserve Rules	○ Enable ◉ Disable (Default)
Allow Broadcast	◉ Enable (Default) ○ Disable
Allow Multicast	◉ Enable (Default) ○ Disable
Allow Link Local Addresses	◉ Enable (Default) ○ Disable
CPU Quota Mode	○ Disable ◉ Adjusted (Default) ○ Top
CPU Quota Limit (%)	3
Memory Quota Limit (MB)	512

Flow Visibility

Flow Analysis Fidelity	○ Conversations BETA ◉ Detailed (Default)
Data Plane	◉ Enable (Default) ○ Disable
Auto-Upgrade	◉ Enable (Default) ○ Disable
PID Lookup	○ Enable ◉ Disable (Default)
CPU Quota Mode	○ Disable ◉ Adjusted (Default) ○ Top
CPU Quota Limit (%)	3
Memory Quota Limit (MB)	512

Creating an Agent Config Intent

1. Click the **Settings menu** in the top-right corner.
2. Select **Agent Config**. The **Software Agents** page displays.
3. Select the **Software Agent Config** tab.
4. Click the **Create Intent** button next to the **Agent Config Intent** heading.
5. Enter the appropriate values in the fields listed in the table below:

Field	Description
Profile	Enter the name of an existing profile and select it from the dropdown menu (required).
Filter	Enter the name of an existing filter or scope or select *Create new filter* from the dropdown menu (required). See *Filters* for more information on creating filters.

6. Click **Save**.

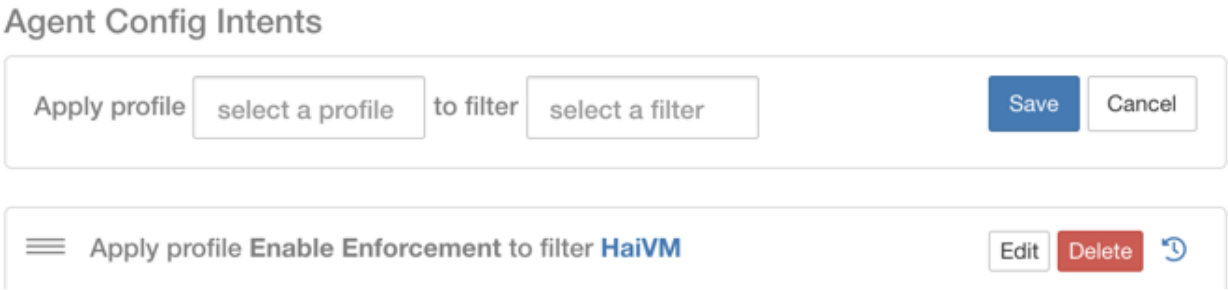


Fig. 16.1.2.1.3: Agent Config Intents

Creating a Remote VRF configuration for agents

This is the recommended way to assign VRFs for Tetration software agents. Using this configuration, Tetration appliance assigns VRFs to software sensors based on the source IP address and source port seen for those agent on connections to Tetration appliance.

1. Click the **Settings menu** in the top-right corner.
2. Select **Agent Config**. The **Software Agents** page displays.
3. Select the **Software Agent Config** tab.
4. Click the **Create Config** button next to the **Agent Remote VRF Configurations** heading.
5. Enter the appropriate values in the fields and click **Save**.

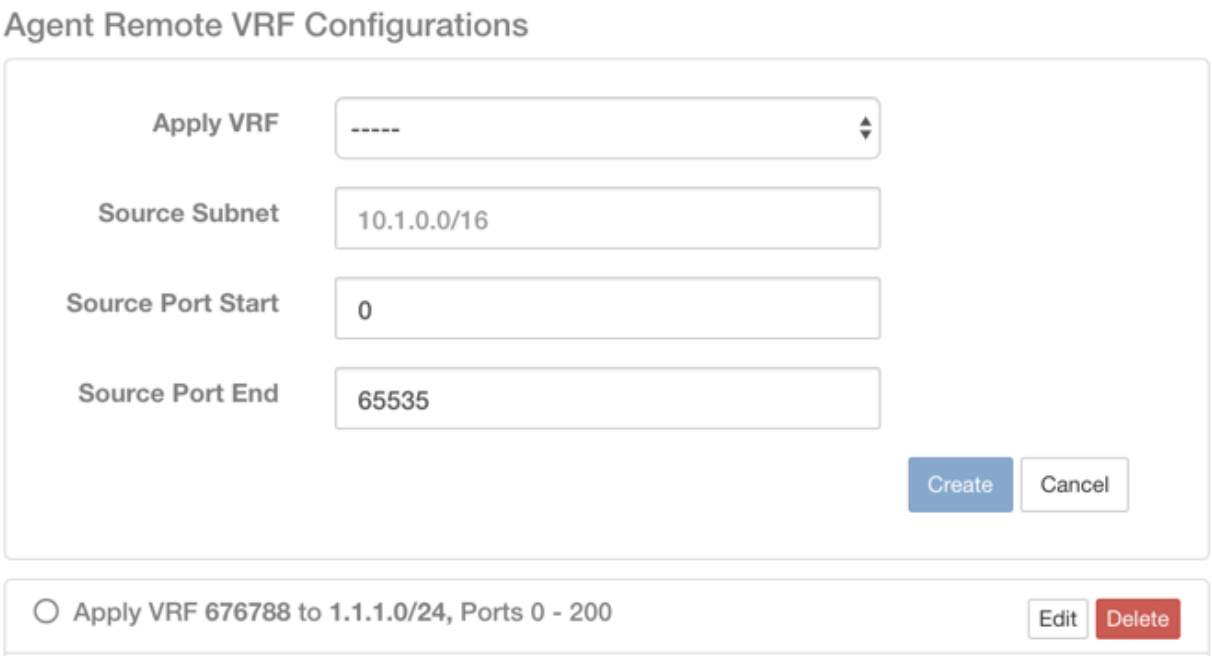


Fig. 16.1.2.1.4: Remote VRF configuration

Creating an Interface Config Intent

Recommended way to assign VRFs to agents is using Remote VRF configuration settings. In rare cases, when agent hosts may have multiple interfaces that need to be assigned different VRFs, users can choose to assign them VRFs using Interface Config Intents.

1. Click the **Settings menu** in the top-right corner.
2. Select **Agent Config**. The **Software Agents** page displays.
3. Select the **Software Agent Config** tab.
4. Click the **Create Intent** button next to the **Interface Config Intent** heading.
5. Enter the appropriate values in the fields listed in the table below:

Field	Description
VRF	Select a VRF from the dropdown menu (required).
Filter	Enter the name of an existing filter or scope or select *Create new filter* from the dropdown menu (required). See *Filters* for more information on creating filters.

6. Click **Save**.

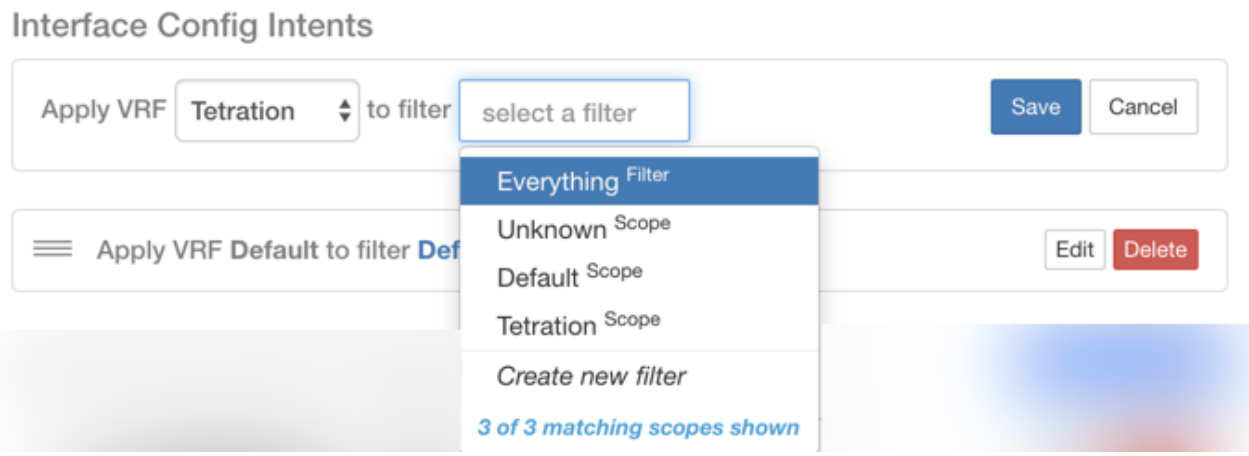


Fig. 16.1.2.1.5: Interface Config Intents

Note: There is a known issue where catch all interface config intent does not get applied. It is only applicable when users delete a higher priority interface config intent; in those cases, agents will not fall back to default catch all intent.

User Roles and Access to Agent Config

1. A Root scope owner has access only to "Config Profile" creation and "Config Intent" specification.
2. A Root scope owner can create config profiles associated with owned scopes only and impose them only on agents that fall under owned filters/scopes.

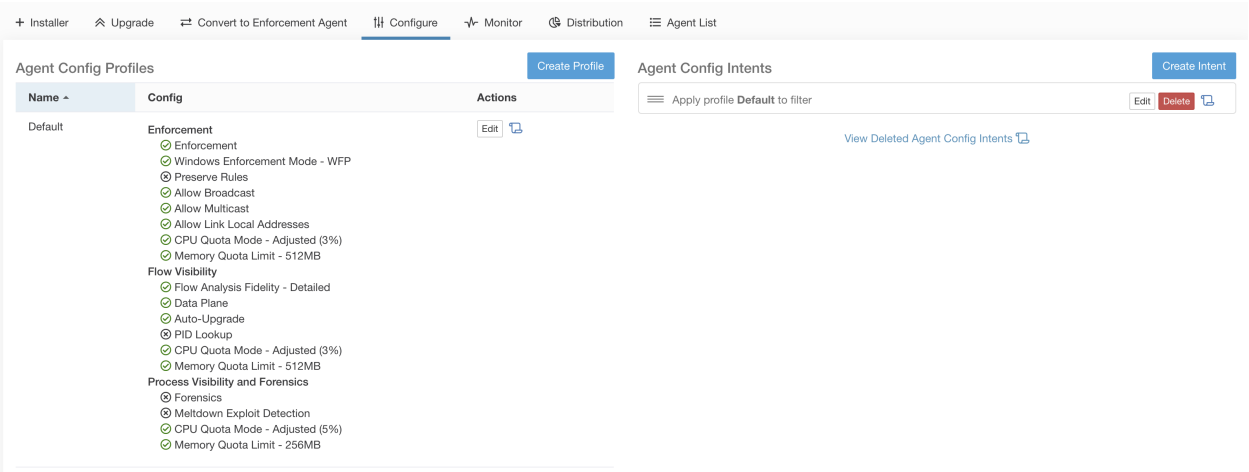


Fig. 16.1.2.1.6: Software Agent Config tab for Scope Owner Users

3. A Site admin user has access to all the components in Agent Config page which include specifying interface config intents and remote vrf configurations

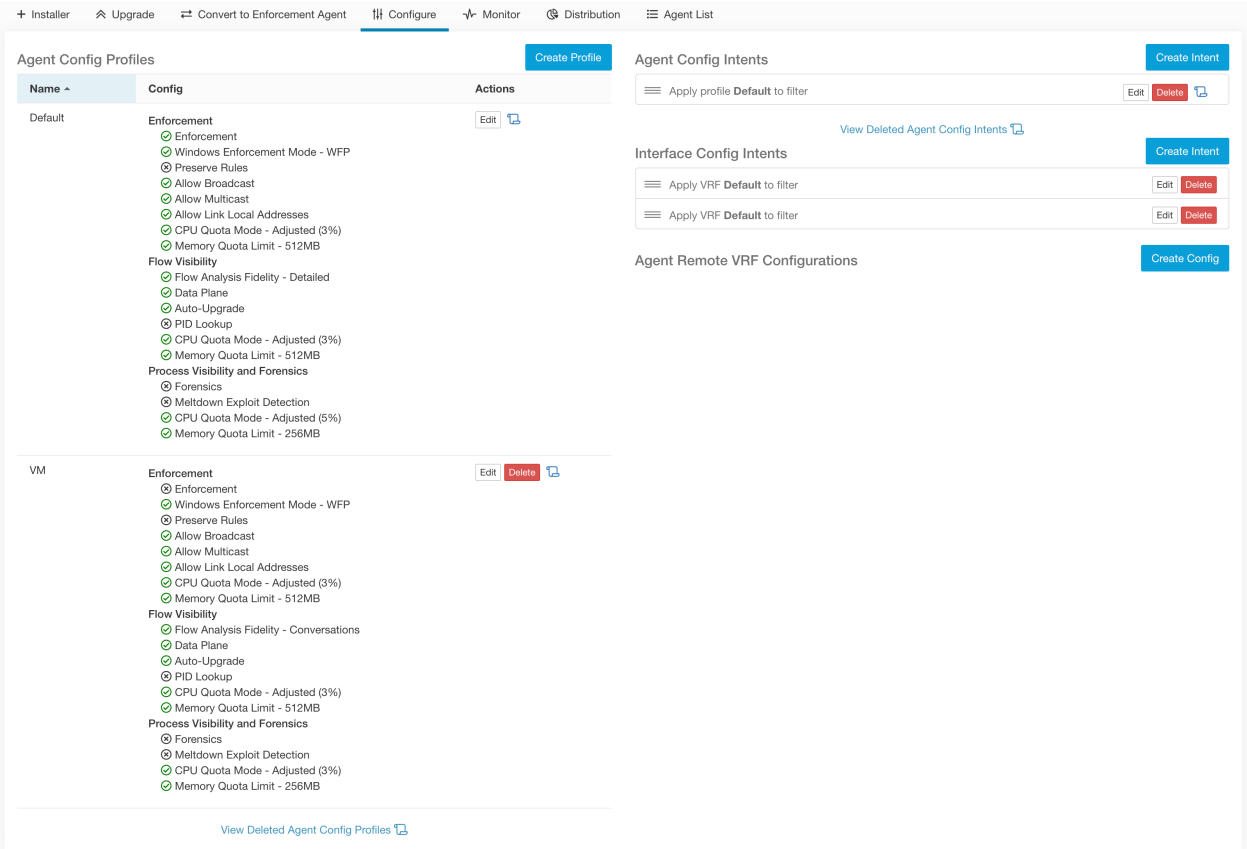


Fig. 16.1.2.1.7: Software Agent Config tab for Site Admin Users

16.1.2.2 Change Log

Site Admins and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each profile and intent by clicking on the icon next to the item as shown below.

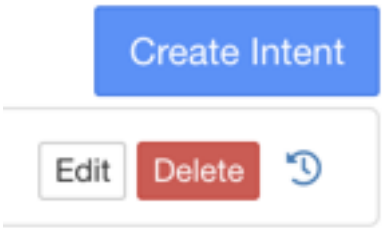


Fig. 16.1.2.2.1: Change Log

These users can also view a list of deleted profiles and intent by clicking on the **View Deleted Profile/Intent** link below each corresponding table.

For more information on the **Change Log** see *Change Log*. Root scope owners are restricted to viewing change log entries for entities belonging to their scope.

16.2 Change Log

Site Admins can access the **Change Log** page under the **Settings menu**. This page shows all of the most recent changes made within Cisco Tetration.

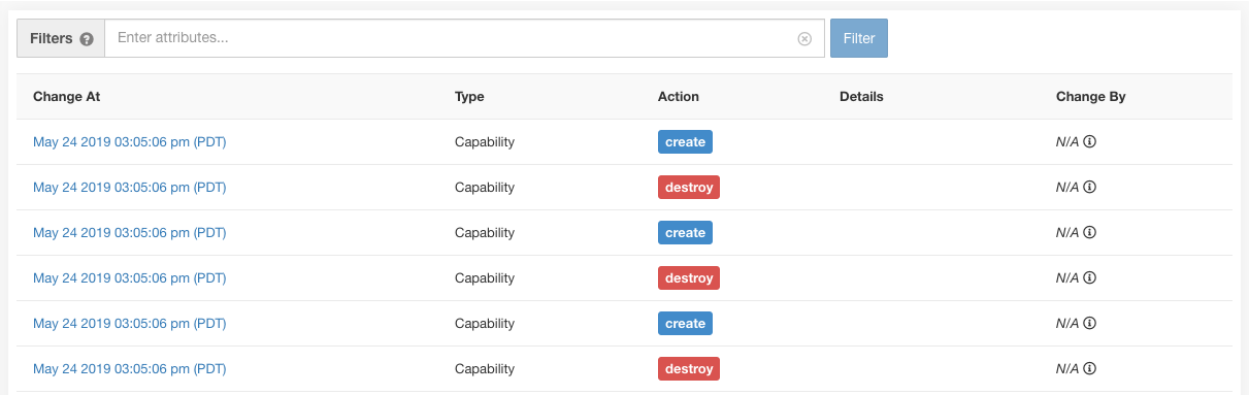


Fig. 16.2.1: Change Log Page

The details of each change log entry can be viewed by clicking on the link in the **Change At** column. This page will include a **Before** and **After** snapshot of the fields changed. The fields may include technical names that require some interpretation to understand how they're surfaced elsewhere throughout Cisco Tetration.

Fig. 16.2.2: Change Log Details Page

The complete list of changes for an entity can be viewed by clicking the button in the upper-right corner, titled **Full log for this <entity type>**. This page will show the details of each change. It also includes the **Current State** of the entity, when available.

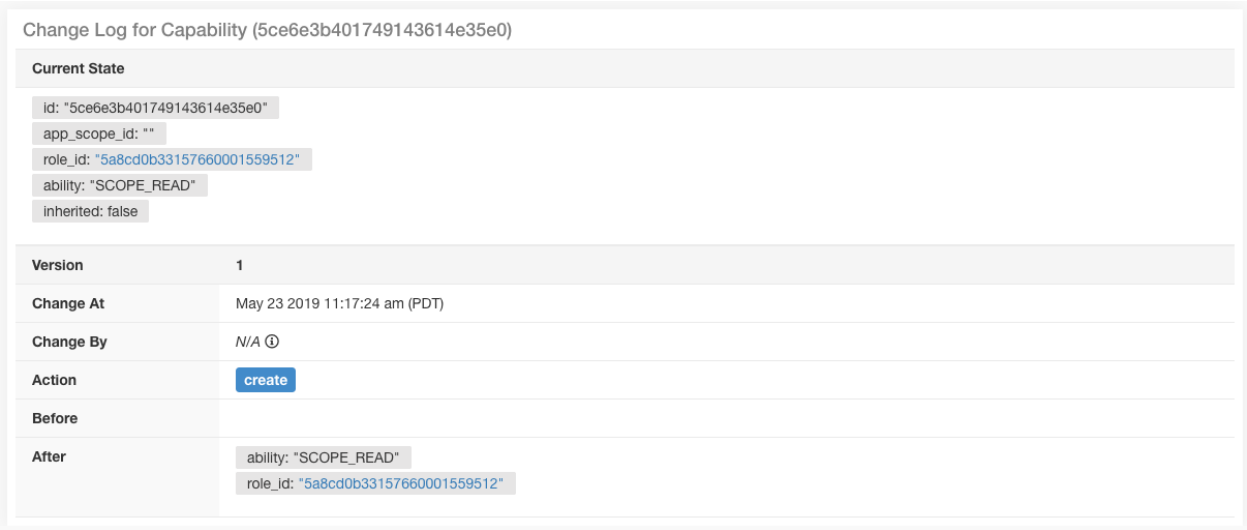


Fig. 16.2.3: Full Change Log for Entity

16.3 Collection Rules

Site Admins and **Customer Support users** can access the **Collection Rules** page under the **Settings menu**. This page shows all of the hardware collection rules by VRF that will be used by switches running the Cisco Tetration agent. There is a row in the table for each VRF.

Fig. 16.3.1: Collection Rules by VRF

16.3.1 Apply to Switches

Depending on the hardware version of your switches, they may not support rules for more than one VRF. If this is the case, please select the **Apply to Switches** checkbox on only one VRF and define all of your rules under this VRF. If your switches support rules for multiple VRFs, select the **Apply to Switches** checkbox on all VRFs that you would like monitored.

16.3.2 Rules

Click the **Edit** button on a VRF to modify its collection rules. By default, every VRF will be configured with two default catch-all rules, one for IPv4 (`0.0.0.0/0 INCLUDE`) and one for IPv6 (`::/0 INCLUDE`). *These default rules can be removed, but do so with caution.*

Additional include and exclude rules can be added. Just enter a valid subnet, select include or exclude and click **Add Rule**. The priority of these rules can be adjusted via drag-and-drop. Just click-and-hold on a rule in the list and drag it to adjust the order.

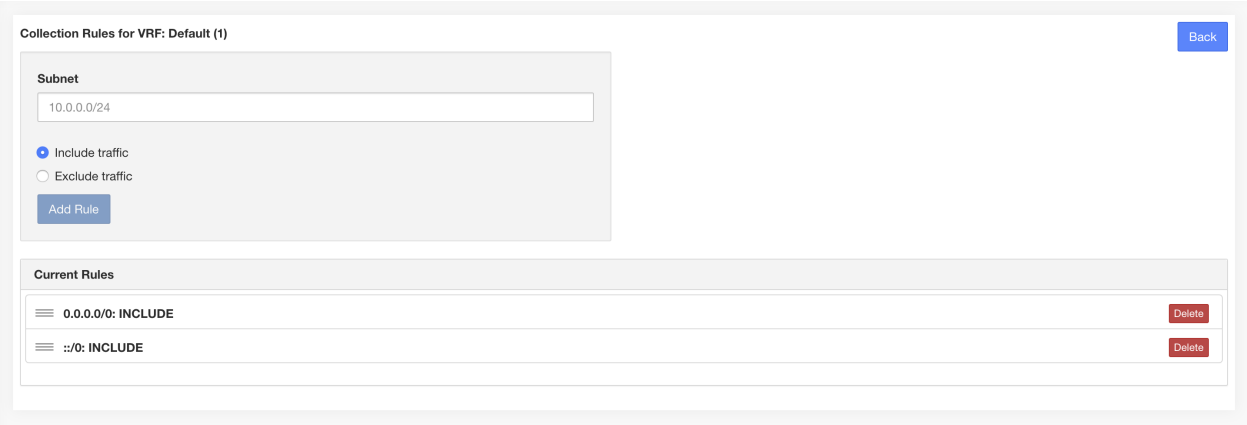


Fig. 16.3.2.1: Edit Collection Rules

Changes may take several minutes to propagate to your switches. Click the **Back** button in the upper-right corner to return to the VRF list.

16.3.3 Priority

Collection Rules are ordered in decreasing order priority. No longest prefix match is done to determine the priority. The rule appearing first has higher priority over all the subsequent rules. Example:

1. 1.1.0.0/16 INCLUDE
2. 1.0.0.0/8 EXCLUDE
3. 0.0.0.0/0 INCLUDE

In the above example, all addresses belonging to 1.0.0.0/8 subnet are excluded except subnet 1.1.0.0/16 which is included.

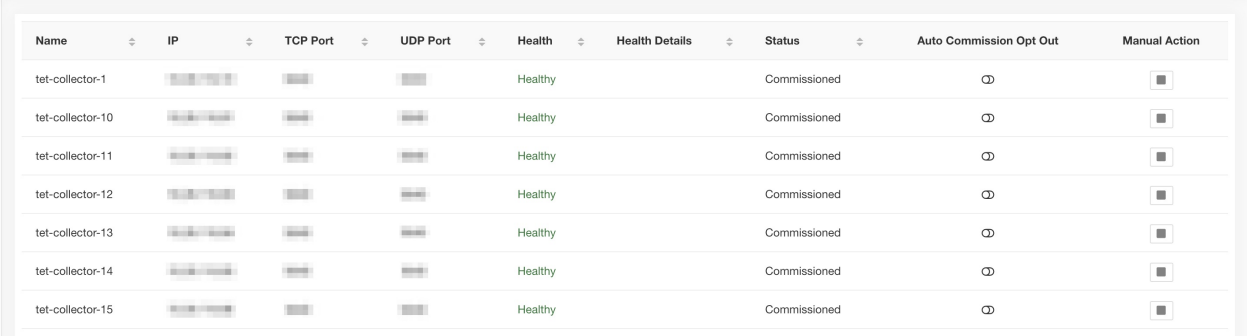
Another Example with changed order:

1. 1.0.0.0/8 EXCLUDE
2. 1.1.0.0/16 INCLUDE
3. 0.0.0.0/0 INCLUDE

In the above example, all addresses belonging to 1.0.0.0/8 subnet are excluded. Rule number-2 does not get exercised here because of a higher order rule already defined for its subnet.

16.4 Collectors

Site Admins and **Customer Support users** can access the **Collectors** page under the **Settings menu**. This page shows all of the currently configured collectors. The Cisco Tetration agents will send flow data to the commissioned collectors, so it's important for all of the commissioned collectors to be available. By default, all collectors are periodically checked for their health and they are either commissioned or decommissioned based on their health. You can opt out of this automated process using the toggle **Auto Commission Opt Out**. With this toggle on, The **Play** and **Stop** icons under the far right column can be used to commission and decommission respectively.

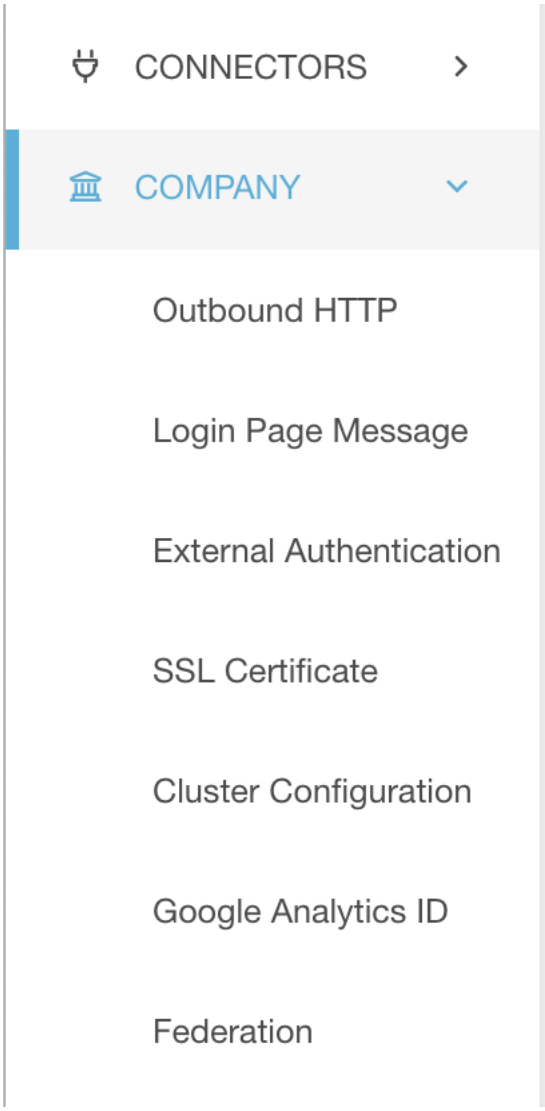


Name	IP	TCP Port	UDP Port	Health	Health Details	Status	Auto Commission Opt Out	Manual Action
tet-collector-1				Healthy		Commissioned		
tet-collector-10				Healthy		Commissioned		
tet-collector-11				Healthy		Commissioned		
tet-collector-12				Healthy		Commissioned		
tet-collector-13				Healthy		Commissioned		
tet-collector-14				Healthy		Commissioned		
tet-collector-15				Healthy		Commissioned		

Fig. 16.4.1: Collectors Page

16.5 Company

Site Admins and **Customer Support users** can access the **Company** page from the menu on the left pane.



Company-wide (per cluster) configurations can be specified here.

16.5.1 Outbound HTTP Connection

To ensure the latest Threat Intelligence Datasets are retrieved from Cisco Cloud, we highly recommend you to set up Outbound HTTP Connection.

> **Warning:** Your enterprise outbound HTTP request may require allowing traffic to **periscope.tetrationcloud.com** and **uas.tetrationcloud.com** from enterprise firewall outbound rules in addition to setting up the HTTP Proxy as shown below.
>
> The TLS connection to **periscope.tetrationcloud.com** is used to transport Threat Intelligence Data for identifying known vulnerabilities. Therefore, it is essential for Tetration to verify the authenticity of the domain name by verifying the domain's X.509 certificate's signing CA cert against reputable root CA certificates included with Tetration. Tampering with the X.509 trust chain will prevent the feature from working correctly.

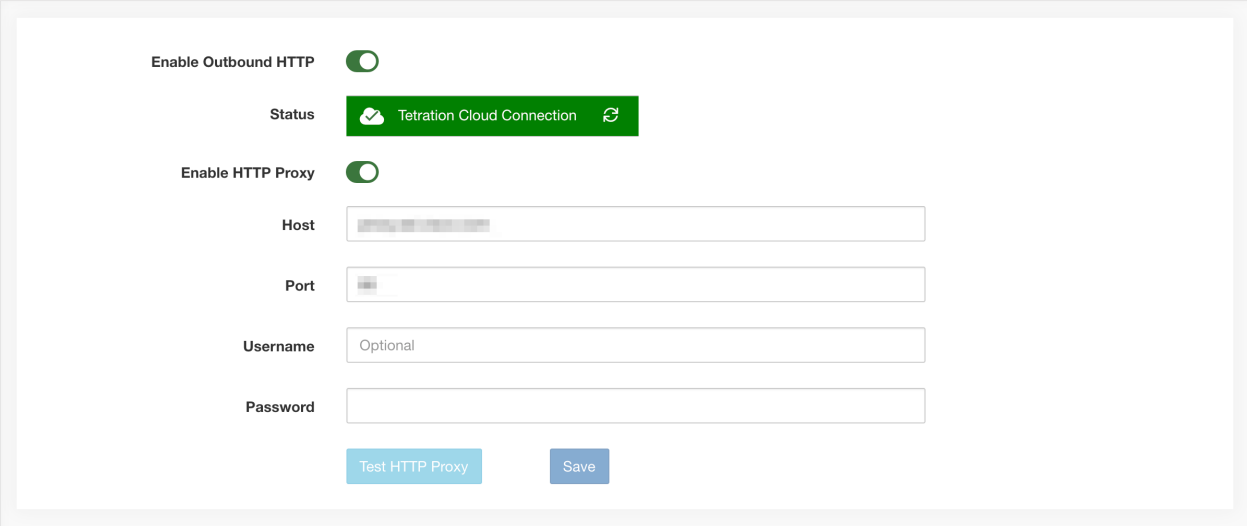


Fig. 16.5.1.1: Outbound HTTP Connection

Field	Description
Status	Indicates whether Tetration appliance can reach to Tetration Cloud to retrieve Threat Intelligence Dataset updates. The status check can be re-triggered by clicking on the refresh button. The following HTTP proxy settings can be used to configure HTTP Proxy settings based on your Tetration deployment.
Enable HTTP Proxy	All external HTTP connections will use HTTP proxy if this option is enabled
Host	HTTP proxy host address
Port	HTTP proxy port number
Username	Required only if your HTTP proxy server uses basic authentication
password	Required only if your HTTP proxy server uses basic authentication

16.5.2 Login Page Message

An optional message of the day can be configured which will be displayed to users on the sign in page once saved. The message is limited to a maximum of 1600 characters.

Fig. 16.5.2.1: Login Page Message

16.5.3 Session Configuration

UI User Authentication idle session timeout can be configured here. The default idle session duration is 1 hour. The idle session duration can be set within the range of 5 minutes to 24 hours. The session timeout will take effect on a user's authenticated session as soon as this value is saved.

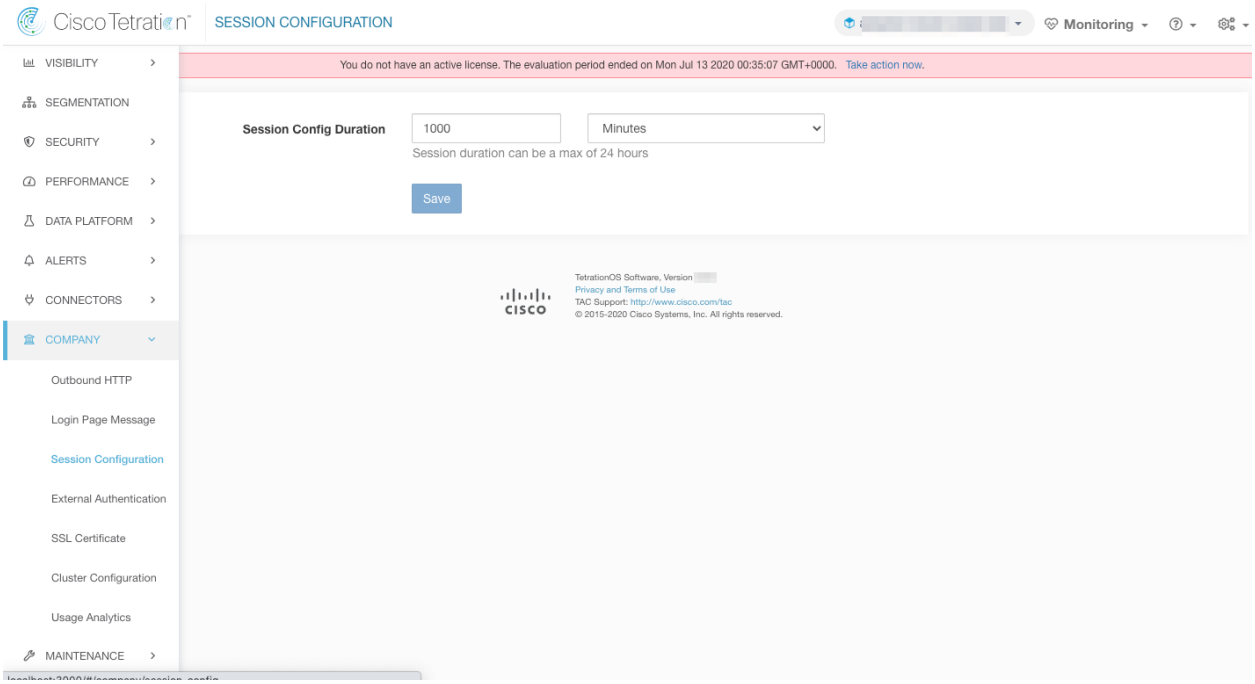


Fig. 16.5.3.1: Configuring Idle Session Timeout

16.5.4 Configuring External Authentication

If this option is enabled, authentication can be handed off to an external system. The current options for authentication are Lightweight Directory Access Protocol (LDAP) and Single Sign-On (SSO). This means that once this is enabled all users[1] signing in will use the chosen mechanism to authenticate. It is important to establish that the LDAP connection is configured correctly, especially if no users are on the *'Use Local Authentication' option*. The recommended approach is to have at least one locally authenticated user with **Site Admin** credentials by turning on the *'Use Local Authentication' option*. This user can make sure that the LDAP configuration is setup correctly. Once the connection is successfully set up, this user can also be transitioned to external authentication by unchecking the 'Use Local Authentication' option in the user edit flow.

Site Admin can enable additional debug messages which is useful to debug external connection issues, user sign in failures etc. This can be enabled by checking the 'External Auth Debug' option. Once this is turned on, additional descriptive log messages are written into a separate log file titled 'external_auth_debug.log'. The recommendation is to turn 'External Auth Debug' off once debugging is done to prevent extra logs being written into the log file.

[1] Users can bypass external authentication once it is enabled on a per user basis as indicated in *'Use Local Authentication' option*. This option can also be enabled by going to the user edit flow from link though the warning message when external auth is enabled as well.

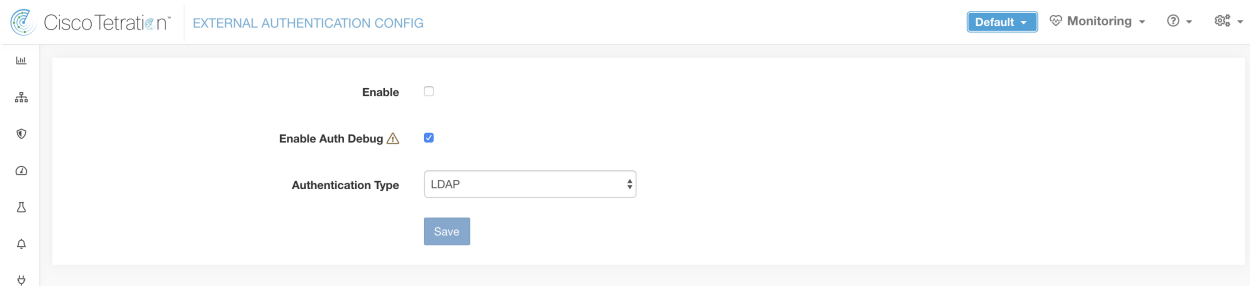


Fig. 16.5.4.1: Configuring External Authentication

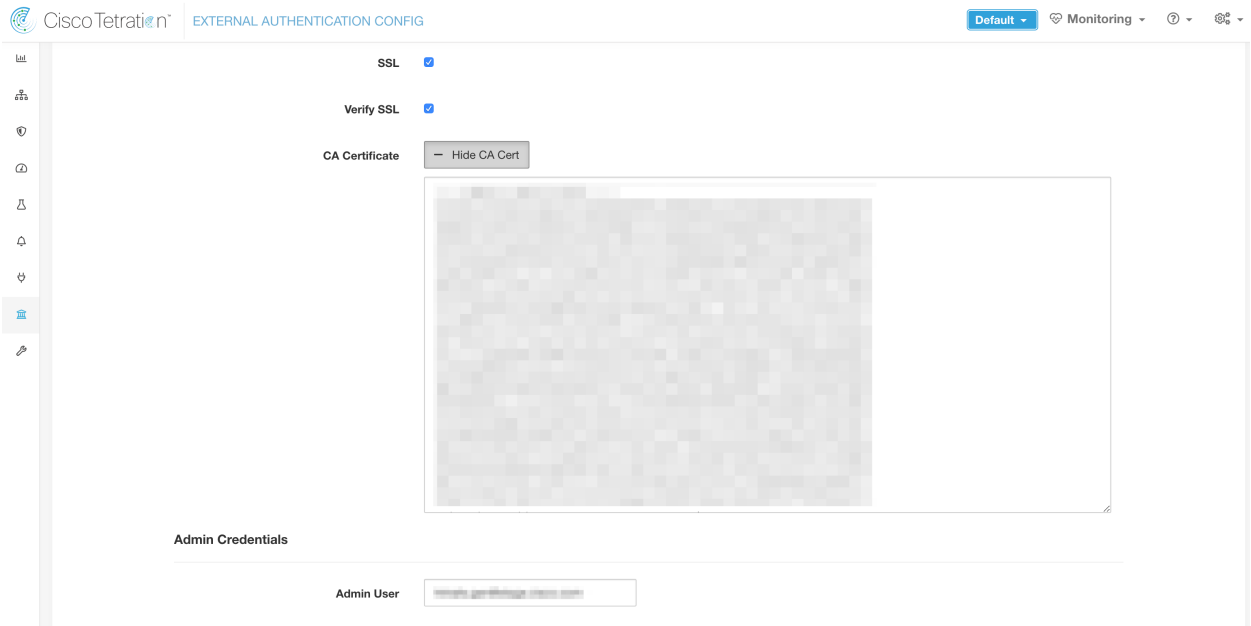


Fig. 16.5.4.2: Configuring External Authentication Continued

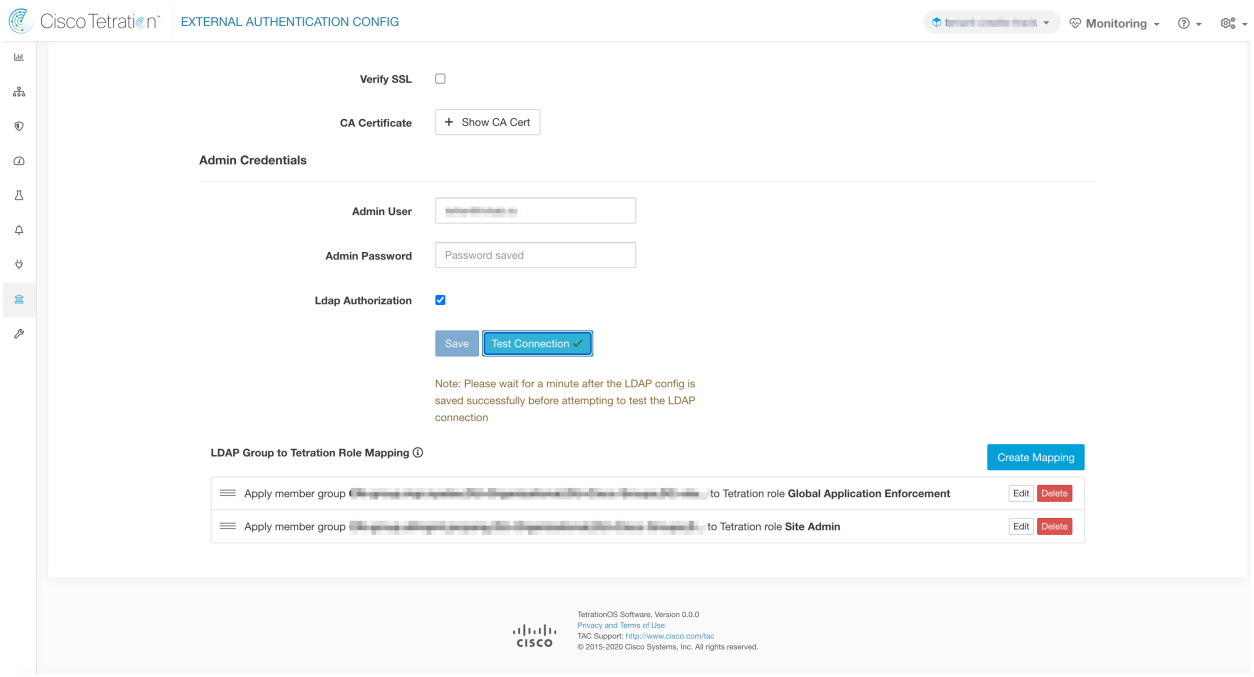


Fig. 16.5.4.3: Configuring External Authentication Continued

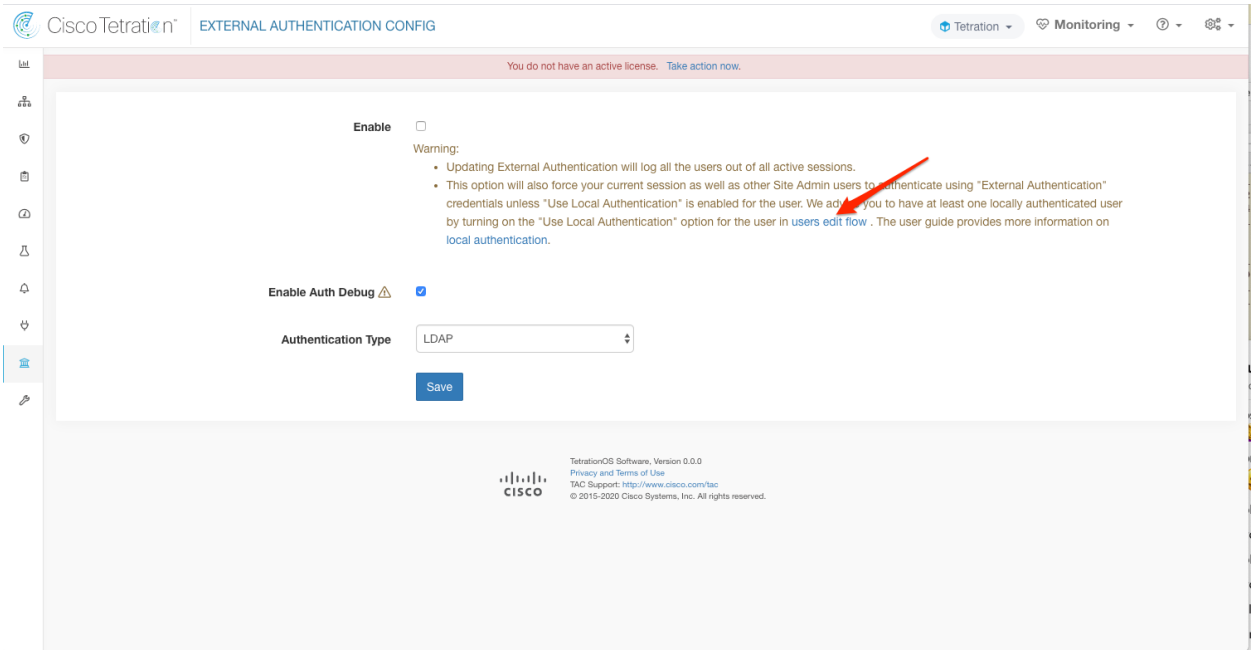


Fig. 16.5.4.4: External Authentication Warning

16.5.5 Configuring Lightweight Directory Access Protocol (LDAP)

If this option is selected, LDAP can be used to authenticate users. This means that once this is enabled all users will be logged out and subsequent signing in will use their LDAP email and password to authenticate.

If LDAP is enabled the recommended workflow for new user creation is as follows.

Site Admins are encouraged to first create new users with their emails and assign the appropriate roles by *Configuring LDAP Authorization (AD authorization)* before new users logs in via LDAP for the first time. If a new user logs in via LDAP without the appropriate role, no default role is assigned to the user.

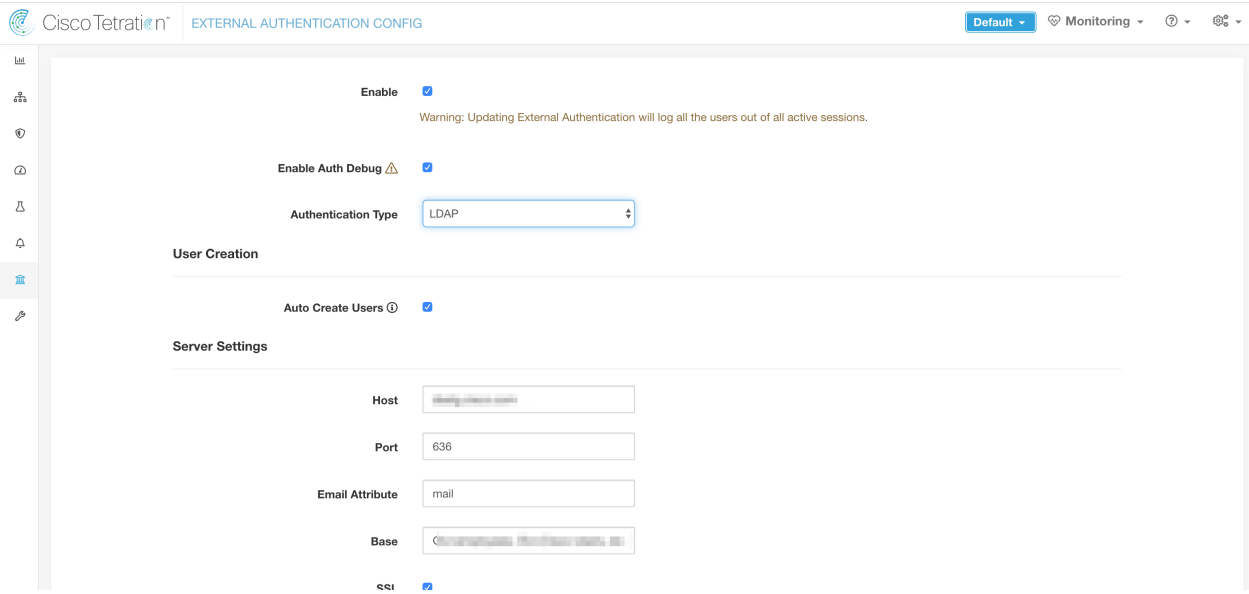


Fig. 16.5.5.1: Configuring Lightweight Directory Access Protocol (LDAP)

Field	Description
Auto Create Users	Turning on 'Auto Create Users' will create users if they don't exist at first login. This will save the
Host	LDAP Host which will be used for authentication.
Port	LDAP Port which will be used for authentication.
Email Attribute	LDAP attribute name which represents email for the organization.
Base	LDAP base dn from where users will be searched.
SSL	Enable encryption and use 'ldaps://'.
SSL Verify	Verify server's SSL attributes such as Fully Qualified Domain Name (FQDN) based on server's ce
SSL Certificate Authority Cert	Signing cert for LDAP server's SSL Cert. Required if server cert chain cannot be publicly verified
Admin User	LDAP Admin user (not Tetration user) name used to bind against the LDAP server. Eg: [User]@[
Admin Password	LDAP Admin password used to bind against the LDAP server.
Ldap Authorization	LDAP Authorization can be enabled and configured as explained in *Configuring LDAP Authorizat*

Once the LDAP config is enabled all users except users with *'Use Local Authentication' option* enabled will be logged out of their sessions.

The LDAP config can be saved once the 'Save' button is clicked. We recommend that you wait for a minute after the LDAP config is saved successfully before attempting to test the LDAP connection.

The LDAP connection can be tested out after the LDAP config has been saved using the 'Test Connection' button. This tries a bind against the LDAP server with the admin credentials entered.

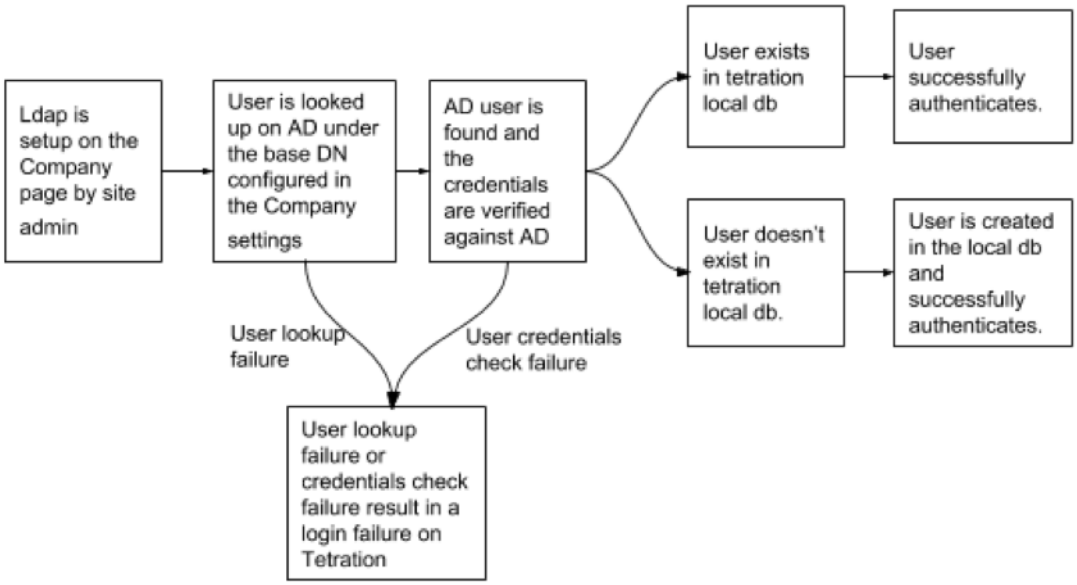


Fig. 16.5.5.2: Authentication Workflow

16.5.5.1 Debugging LDAP issues

If an error is raised when you test the ldap connection, please check the following:

- Check whether the LDAP admin credentials are correct.
- Check the connection params such as host, port, ssl etc.
- Check whether the LDAP server can be reached from Tetration UI VIPs.
- Check whether the AD server is up.
- Use command line tools such as **'ldapsearch'** with the connection details to see whether a bind can be made.

If an error is raised during login for a user, please check the following:

- Check whether the user can login with their LDAP credentials to other company websites which use LDAP authentication.
- Check whether the 'base' dn specified in the Company LDAP settings is correct. This can be done by using command line tools such as **'ldapsearch'** to lookup the user against the base dn.

Example **'ldapsearch'** query to search a user by email:

```
ldapsearch -H "ldap://<host>:<port>" -b "<base-dn>" -D "<ldap-admin-user>" -w <ldap-
↪admin-password> "(mail=<users-email-address>)"
```

16.5.6 Configuring LDAP Authorization (AD authorization)

Active Directory Authorization can be configured by enabling the 'LDAP Authorization' checkbox in the 'Admin Credentials' section of the External Authentication LDAP configuration. Once this setting is enabled, Site Admin needs to set up mappings of LDAP 'MemberOf' groups to Tetration Roles in the section below. By default, without this configuration, Active Directory users need to be pre-configured with one or more Tetration roles prior to a login attempt.

LDAP MemberOf Group to Tetration Role Mapping must be setup if LDAP external authentication is enabled. 'Create Mapping' allows setting up an LDAP MemberOf group value to be mapped to a Tetration Role. The roles in the role dropdown are pre-populated based on the scope selected in the scope selector. Once these mappings are saved, all users[2] will get authorized based on these values on their subsequent login.

These mappings can be reordered, edited or deleted. Any modifications to the mappings will be reflected on the roles assigned to users on their subsequent login. A maximum of 50 LDAP MemberOf Group to Tetration Role Mappings can be created.

Duplicate LDAP MemberOf group names are not allowed. However multiple LDAP MemberOf groups can map to the same role. If more than one group maps to the same role, the last mapping will be stored in the user as the matched LDAP MemberOf to Tetration role.

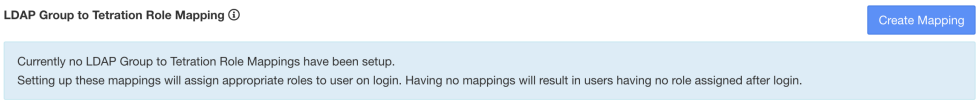


Fig. 16.5.6.1: LDAP Group to Tetration Role SetUp

Fig. 16.5.6.2: LDAP Group to Tetration Role Mapping

A site admin user can reconcile the assignment of roles based on the above role mapping with the help of external user's info obtained from the user's last successful login

[2] Users can bypass external authentication once it is enabled on a per user basis as indicated in *'Use Local Authentication' option*. These users will also bypass the authorization process set up for AD authorization.

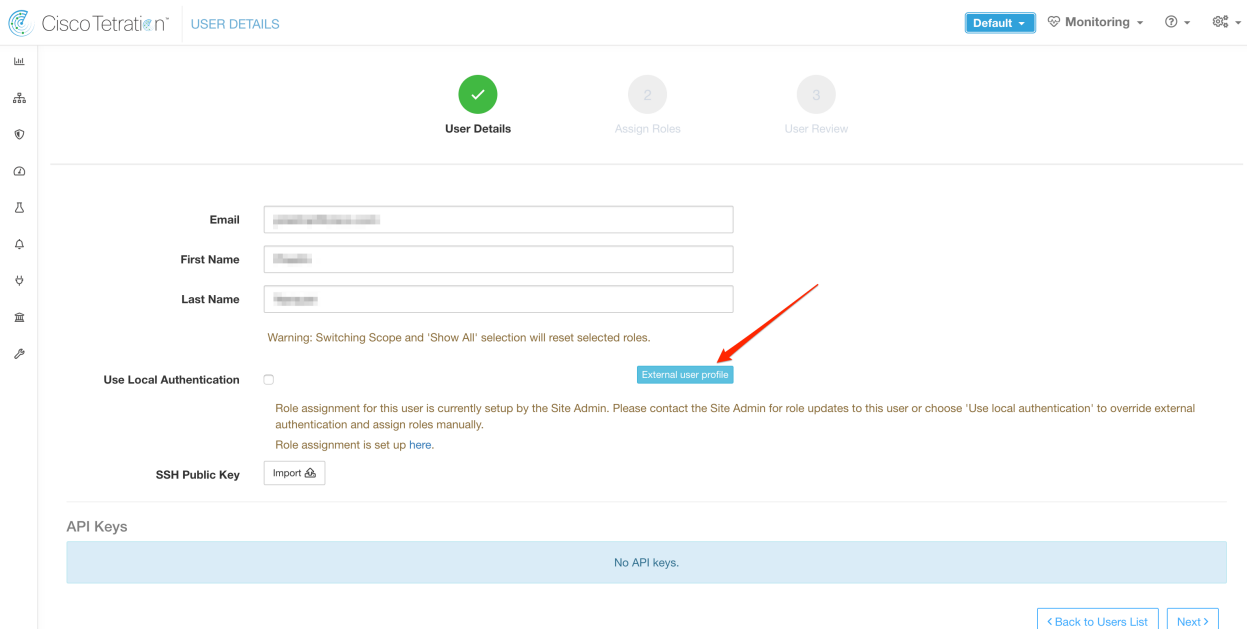


Fig. 16.5.6.3: External User Information

Once authorization is enabled, manual Tetration Role selection in the user creation (*Adding a New User Account*) and user edit flows (*Editing a User Account*) is **disallowed**.

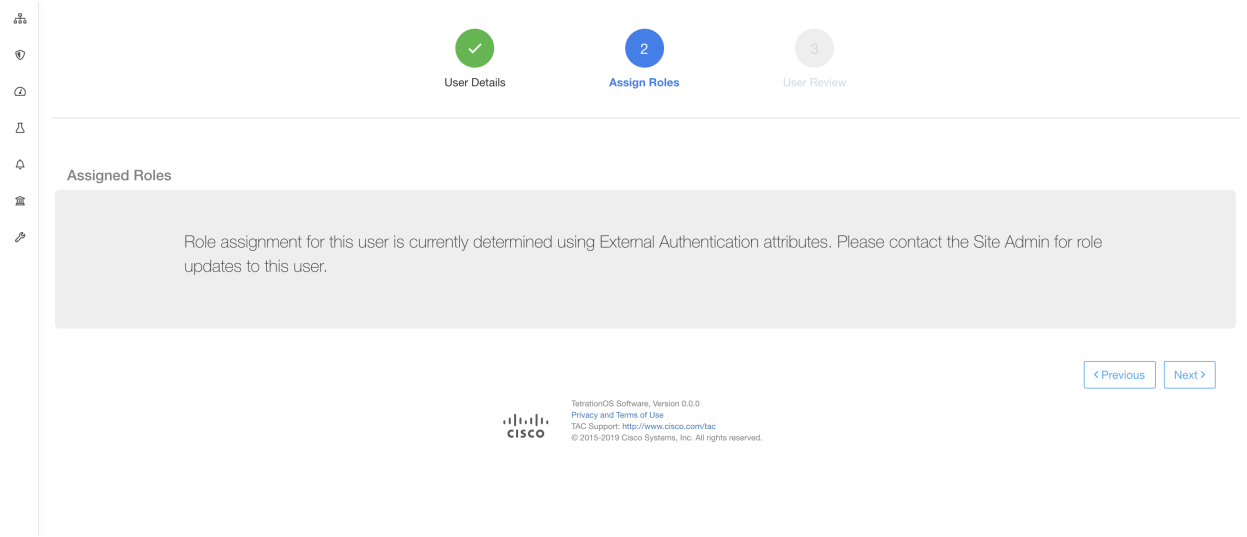


Fig. 16.5.6.4: Users Page

The mapped LDAP MemberOf groups to Tetration Roles are visible on the user profile page.

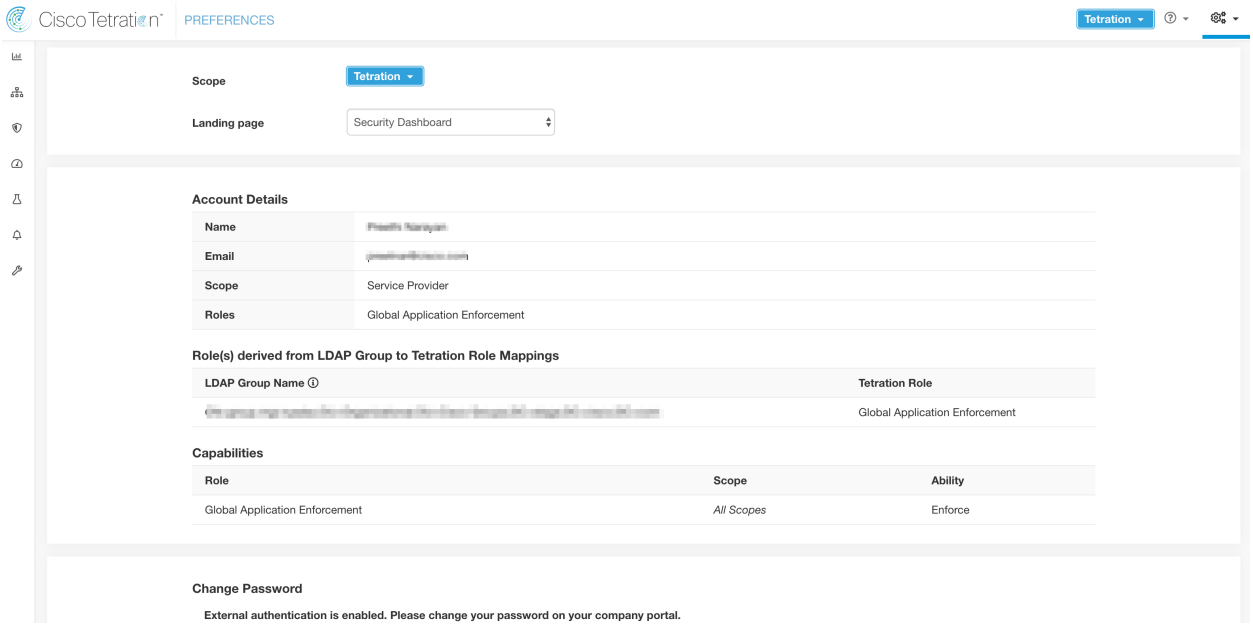


Fig. 16.5.6.5: User Profile Page

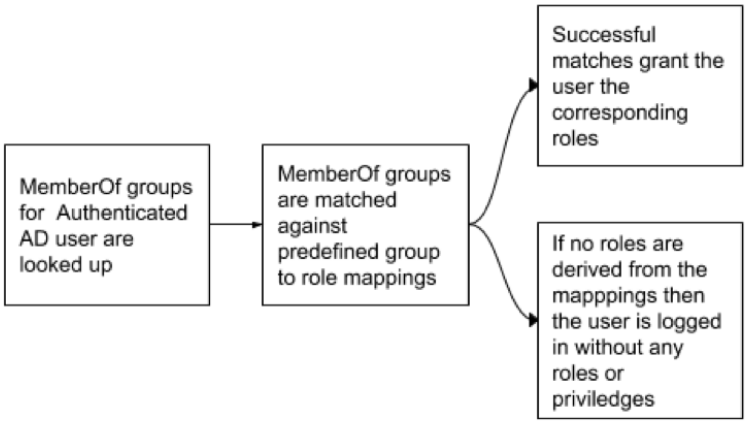


Fig. 16.5.6.6: Authorization Workflow

If LDAP Authorization is enabled, access to OpenAPI via API Keys will cease to work seamlessly because Tetration Roles derived from LDAP MemberOf groups are reassessed once the user session terminates. Hence to ensure uninterrupted OpenAPI access, we recommend that any user with API Keys have *'Use Local Authentication' option* enabled.

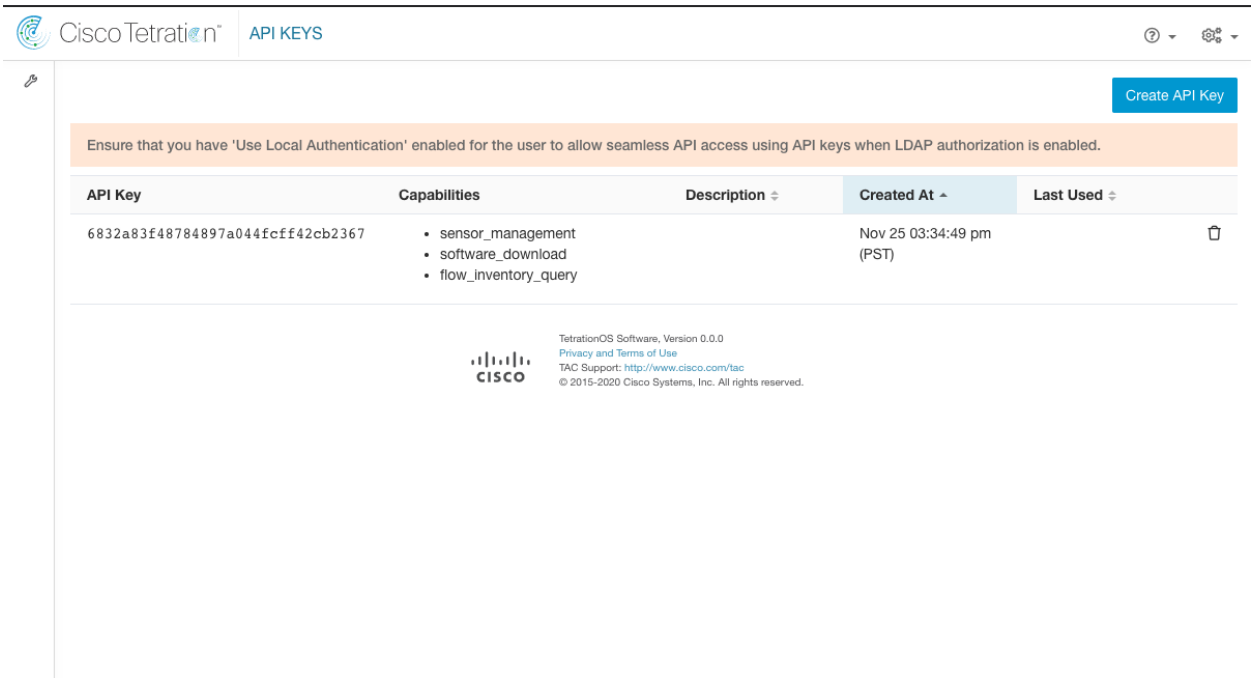


Fig. 16.5.6.7: LDAP Authorization API Key Warning

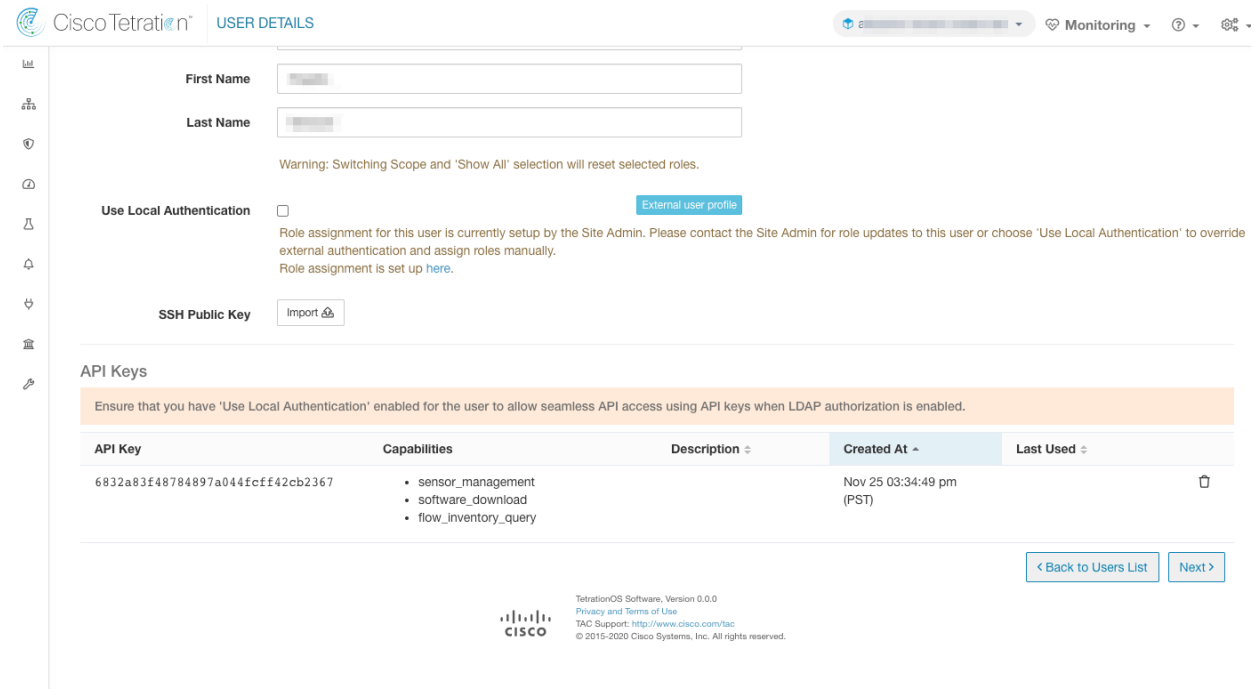


Fig. 16.5.6.8: LDAP Authorization API Key Warning on Users Page

16.5.6.1 Debugging LDAP Authorization issues

If the roles are not getting assigned to users based on mappings defined in the 'External Authentication', 'LDAP Group to Role Mappings' section, check the role mappings setup and format once more.

- Group string should be of the string format. Eg: CN=group.jacpang,OU=Organizational,OU=Cisco Groups,DC=stage,DC=cisco,DC=com
- Group names must be exact from what is present in AD with no spaces or extra characters.
- Role mapping for the group should be selected from the role selector

User Role Mapping Debug Steps

- You should have 2 users, one that is Site Admin, the email of this user shouldn't be the same as the AD user.
- This user will be called 'SA User' for the steps below.
  - SA user has previously set up the role mapping configs on the Company page External Auth Config as described above. Let's assume 'SA User' will be logging in with [site-admin]@[Domain].
  - We'll assume that 'AD User' is [ad-user]@[Domain]. We'll assume that the LDAP setup is done and the AD user is able to login but not getting his role assigned.
- As AD User, login using incognito browser session. This splits the browser state from SA User session.
- As SA User, login and go to Users page.
- Click on the Edit Icon for the AD User that needs to have Role Mapping configured.
- Click on the 'External User Profile' icon as shown in the User Profile Page figure.
- You'll see a table like this. Notice the 'memberof' section.

Fig. 16.5.6.1.1: External Auth Profile Table

- This is one of the 'memberof' values you can use for role mapping under Company page, External Auth Config, Ldap Group to Role Mapping section.
- You need to provide the whole 'memberof' per-line string to match. Once you create this role mapping, anyone who has the same attribute 'memberof' will be assigned the mapped role.
- For the AD User to be granted the newly mapped role, the user need to log out then log back in to allow re-evaluation of this mapping profile.
- Once a user logs in and has roles assigned successfully as a result of group role mappings, the matching rules are visible on the 'Preferences' page for that user.

16.5.7 Configuring Single Sign-On (SSO)

If this option is selected, SSO can be used to authenticate users. This means that once this is enabled all users will be redirected to the identity provider sign in page to authenticate. Users with *'Use Local Authentication' option* enabled can use the email and password sign in form in the sign in page to authenticate.

It is important to establish that the SSO configuration is set up correctly, especially if no users are on the *'Use Local Authentication' option*. The recommended approach is to have at least one locally authenticated user with **Site Admin**

credentials by turning on the *'Use Local Authentication' option*. This user can make sure that the SSO configuration is setup correctly. Once the connection is successfully set up, this user can also be transitioned to external authentication by unchecking the 'Use Local Authentication' option in the user edit flow.

If SSO is enabled the recommended workflow for new user creation is as follows.

Site Admins and **Scope Owners** are encouraged to first create new users with their emails and assign the appropriate roles and scopes before the new user logs in via SSO for the first time. If a new user logs in via SSO without the appropriate role, no default role is assigned to the user.

The following table describes the fields that need to be setup in order to configure SSO on Tetration. Tetration is the Service Provider (SP) in this case.

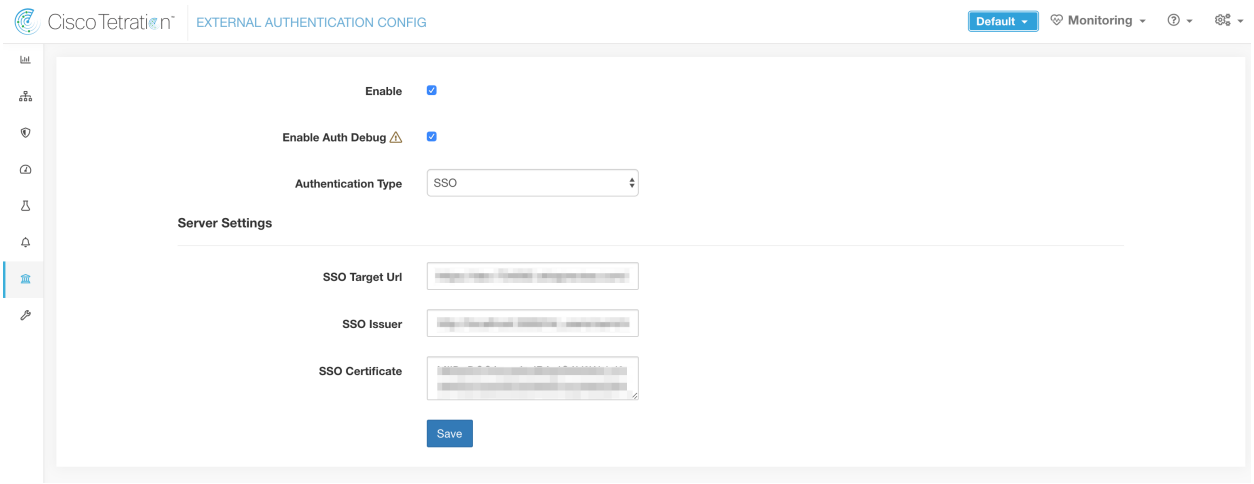


Fig. 16.5.7.1: Configuring Single Sign-On

Field	Description
SSO Target Url	SSO IdP target url to which users will be redirected to for sign in.
SSO Issuer	SSO Entity Id of your SP, a URL that uniquely identifies your SP. This is generally the metadata for the SP. In this case it is: `https://<tetration-cluster-fqdn>/h4_users/saml/metadata`
SSO Certificate	SSO certificate provided by the Identity Porvider (IdP).

Once the SSO config is enabled all users except users with *'Use Local Authentication' option* enabled will be logged out of their sessions.

The SSO config can be saved once the 'Save' button is clicked.

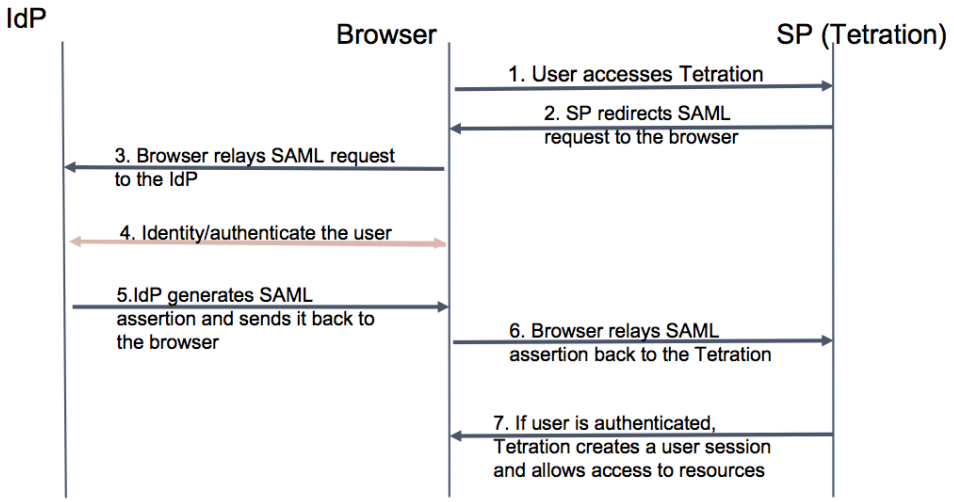


Fig. 16.5.7.2: Authentication Workflow

16.5.7.1 Information to be provided to the Identity Provider (IdP)

The IdP will need some information from Tetration (SP) in order to set up SSO for authentication. The following table describes the fields that need to be setup.

Field	Description
SSO Url	The authentication endpoint (url) which will consume the SAML assertion (response from the IdP). In our case it will be – `https://<tetration-cluster-fqdn>/h4_users/saml/auth`
Entity Id	This is the metadata for the SP. In this case it is: `https://<tetration-cluster-fqdn>/h4_users/saml/metadata`
Name ID format	NameId is email i.e `'urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress'`
Attributes	User attributes are fetched from the IDP. We fetch these attributes as part of authentication: • email • firstName • lastName Please make sure that the attribute names are as specified above.

16.5.7.2 Debugging SSO issues

- Set up some downtime for this SSO config setup since the only way to verify authentication works (from the Service Provider) it is after setting it up.
- Check and validate the IdP metadata generated.

- **Check all configuration parameters exchanged between IdP and SP.**
  - Config at the IdP - SSO url, Audience, Name ID, attributes etc
  - Config on Tetration Company page - SSO Target url, SSO issuer and SSO certificate.
- Get a sample SAML assertion returned from the IdP from the server app logs. Validate it against a SAML validator to make sure it is a valid SAML response.
- Errors in the SP SSO setup may result in an error generated from the IdP. Using the browser inspect element, you can see the network requests being made.
- If a user has issues logging in, have the IdP admin check whether the user has access to the Tetration app.

16.5.8 'Use Local Authentication' option

Once the config is setup, it is possible for site admins to allow users not to use external authentication. This can be done on a per user basis by enabling a flag 'Use Local Authentication' in the user edit section. Selecting this field for the user will log that user out of all sessions.

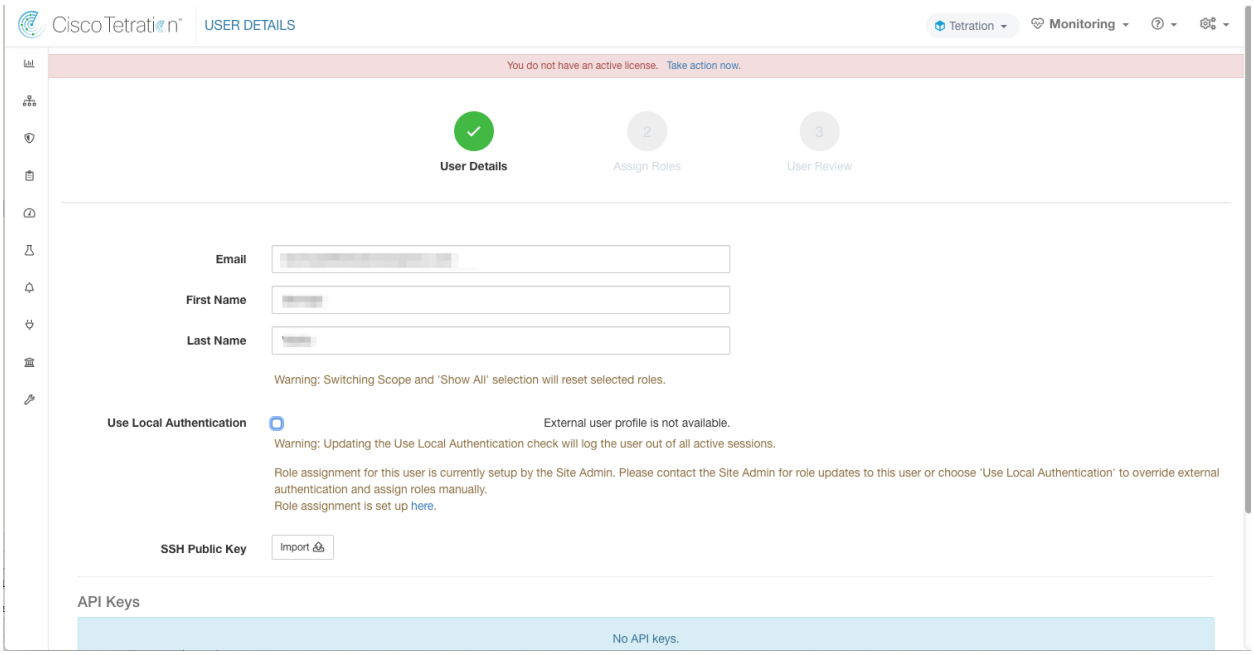


Fig. 16.5.8.1: Use Local Authentication

If the 'Use Local Authentication' option is removed i.e unchecked for a user and this user happens to be the last user with the option, a warning message comes up to indicate this. Any disruption with the external authentication system such as config issues, connectivity issues etc will mean that no user user has local authentication access to signin to Tetration.

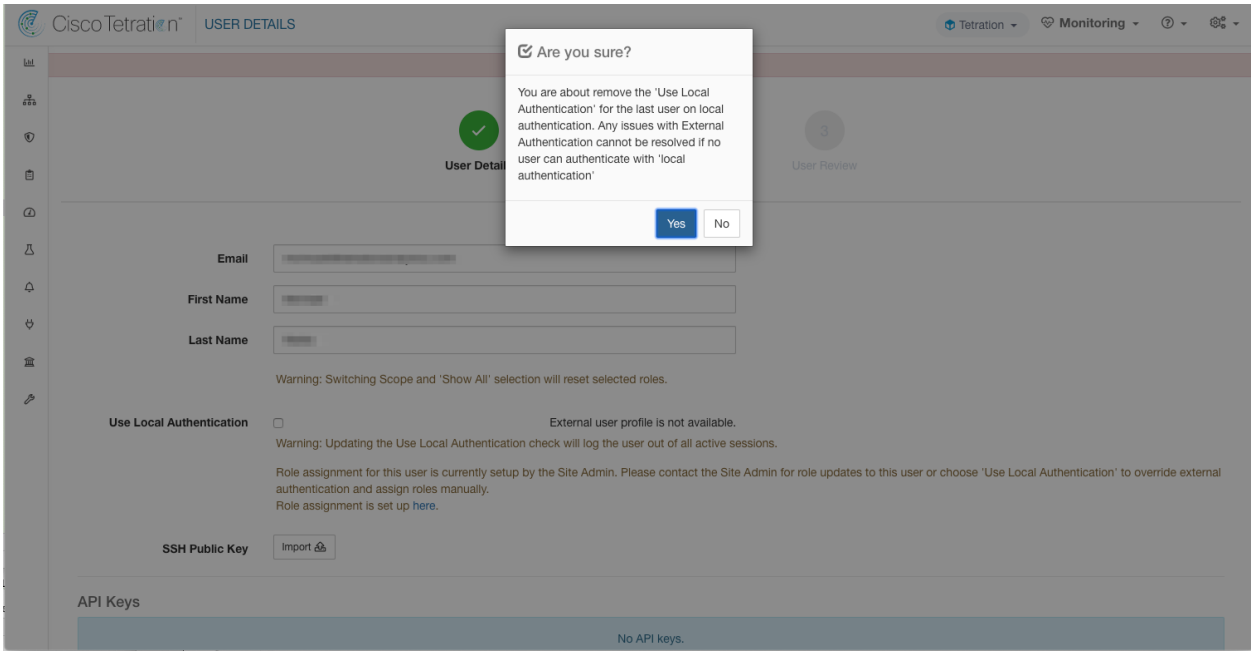


Fig. 16.5.8.2: Use Local Authentication Warning

Users logging via external authentication will have shorter sessions and will be prompted to log in when the session expires. Users logging via external authentication cannot reset their password on the site (they will have to do it on their company website). However if the 'Use Local Authentication' flag is set for the user, password reset is possible.

16.5.9 SSL Certificate and Key

To enable fully verifiable HTTPS access to the Tetration UI, an SSL certificate specific to the UI's domain name and the RSA private key that matches the SSL certificate's public key can be uploaded into the cluster.

An SSL Certificate can be obtained in two ways depending on the format of the Fully Qualified Domain Name (FQDN) used to refer to the Tetration UI Virtual IP (VIP) address. If the Tetration FQDN is based on an enterprise domain name such as `tetration.cisco.com`, your enterprise Certificate Authority (CA) who owns the base domain will issue you a SSL Certificate. Otherwise, you may use a reputable SSL Certificate vendor to issue you a SSL Certificate for your FQDN.

Note: It is important to note that although the Tetration UI supports Server Name Indication (SNI), subject alternative names (SANs) specified in the certificate will not be matched. For instance, if the common name (CN) of the certificate is tetration.cisco.com and the certificate includes a SAN for tetration1.cisco.com, HTTPS requests sent with an SNI-compatible browser to the cluster with tetration1.cisco.com as the hostname will not be served with that certificate. HTTPS requests made to the cluster with a hostname other than the hostname specified in the CN will be served using the default, self-signed cerficate installed on the cluster. These requests will result in browser warnings.

To import the certificate and key, click on the **Import New Certifcate and Key** button.

Note: The first import of SSL certification and the private key should be performed through a trusted network connection to the cluster so that the private key cannot be intercepted by malicious parties who has access to the transport layer.

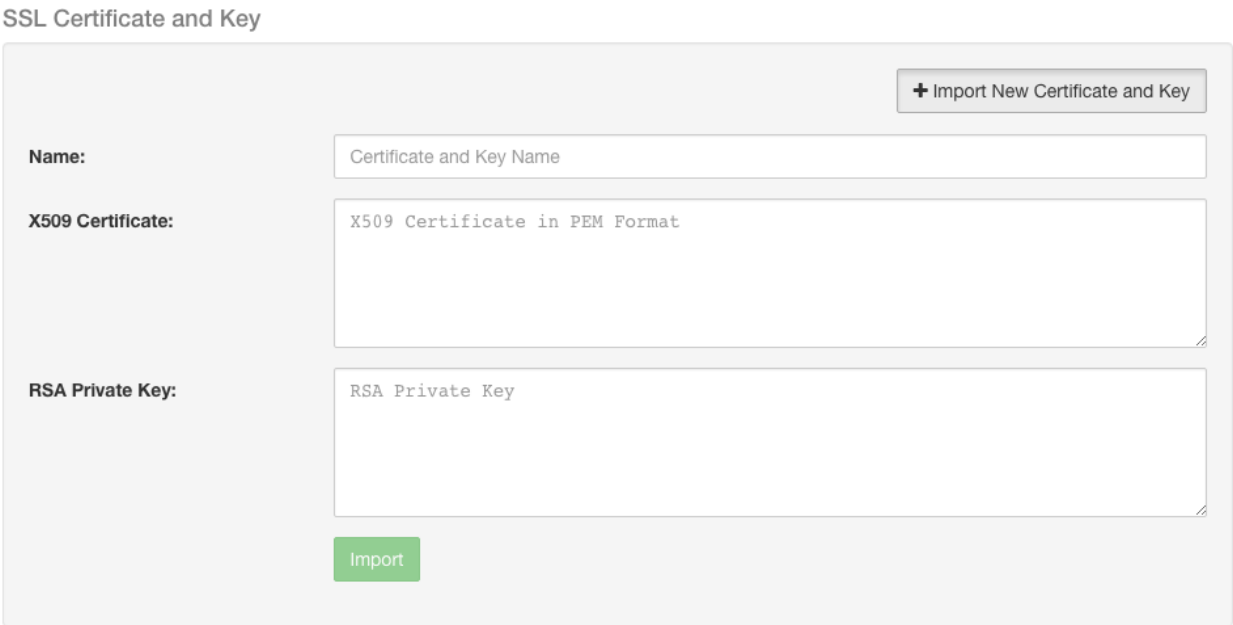


Fig. 16.5.9.1: SSL Certificate and Key

NAME This can be any name for the certificate key pair. This name is for your benefit when looking at which SSL certificate is installed.

X509 Certificate field accepts SSL certificate string in Privacy Enhanced Mail (PEM) format. If your SSL certificate requires intermediary CA bundle, concatenate the CA bundle after your cert so that the SSL certificate for your Tetration FQDN is in the beginning of the certificate file.

It should have the following format:

```
-----BEGIN CERTIFICATE-----
< Certificate for Tetration FQDN >
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
< Intermediary CA 1 content >
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
< Intermediary CA 2 content >
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
< Root CA content >
-----END CERTIFICATE-----
```

RSA Private Key field should be RSA private key of the public key signed in the the certificate above. It should have the following format:

```
-----BEGIN RSA PRIVATE KEY-----
< private key data >
-----END RSA PRIVATE KEY-----
```

Note: RSA Private Key is required to be unencrypted. It will cause a "500 Internal Server Error" if RSA Private Key is encrypted.

Once the import button is pressed, we run verification steps to ensure that public key signed in the certificate and the private key are indeed RSA key pair. If the verification is successful, we will display the SHA1 digest of the certificate bundle.

Fig. 16.5.9.2: SHA1 digest

Reload the browser to see that your SSL connection to the Tetration UI is now using the newly imported SSL certificate.

16.5.10 Cluster Configuration

This section display the running configuration of the Tetration cluster with respect to the customer network and administrative contacts. Some of these configuration items are editable. After the configuration is edited, it takes some time for the new configuration to get applied through out the cluster and it is indicated by highlighting the particular config.

Fig. 16.5.10.1: Editable cluster configuration

Note: a. Strong SSL Ciphers for Agent Connections: When this option is enabled, following connections will honor it and use strong ciphers during the TLS handshake:

1. All API and UI connections to Tetration
2. All visibility and enforcement agent connections to Tetration

Please note older SSL libraries may not support this option.

16.5.11 Usage Analytics

We collect data that Cisco used only to improve the Tetration user interface. Collected data is anonymized thorugh one-way hashing before being sent to the server. Data collection is enabled by default and can be toggled on this page. The configurability of this privacy setting is on per-appliance basis for on-prem appliances and per-tenant basis for TaaS

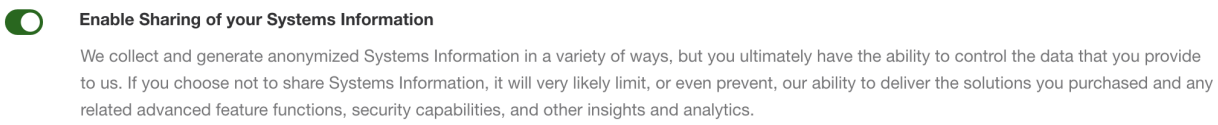


Fig. 16.5.11.1: Toggle Usage Analytics

16.6 Idle Session

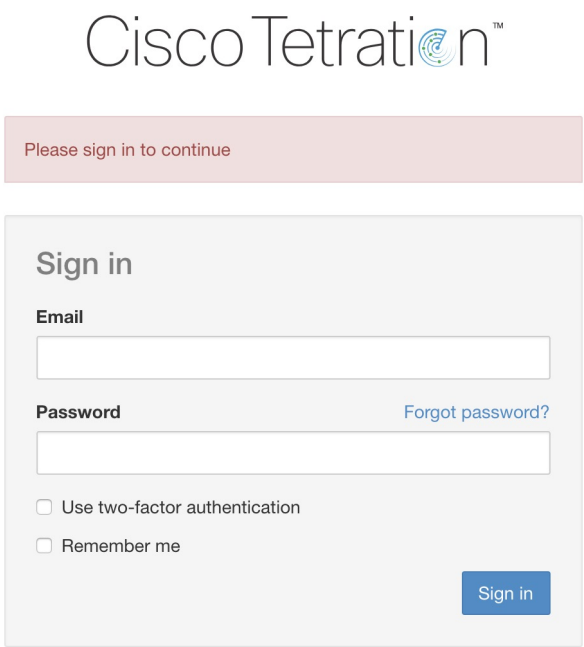


Fig. 16.6.1: Email and password login interface

For those who are authenticating using local database, this section explains how failed login attempts may lock the user account:

1. Five failed login attempts using email and password will result in locking the account.

Note: As a security measure against probing, no specific message indicating the lock will be provided in the login interface when trying to sign in a locked account.

2. Lock out interval is set at 30 minutes. After the account is unlocked, use correct password to login or initiate password recovery by clicking *Forgot password?*

Note: Once a user is successfully signed in, one hour of inactivity will log out the user. This timeout is configured from `Settings > Company > Session Configuration`

16.7 Preferences

The **Preferences** page displays your account details and enables you to update your display preferences, change your landing page, change your password, and configure two-factor authentication.

16.7.1 Changing Your Landing Page Preference

This section explains how to change your landing page.

Note: Security dashboard is set as default landing page. To change it to another page, follow steps as described below.

1. In the **Preferences** pane, open dropdown with label landing page.
2. Choose between **Applications**, **Dashboard**, **Fabric**, **Flow search**, **Performance Dashboard**, **Security Dashboard** dropdown options. Your preference is saved as soon as the menu option is selected. To see the change, click on the Tetration logo at the top left corner of the page.

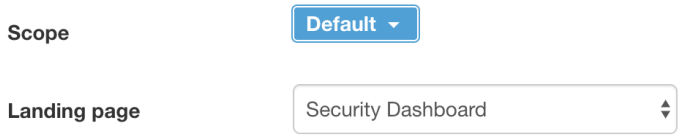


Fig. 16.7.1.1: Change landing page pane

16.7.2 Changing a Password

This section explains how to change your password.

1. Click on the **gear menu** in the top-right corner.
2. Select **Preferences**. The **Preferences** page appears.
3. In the **Change Password** pane, enter your current password in the **Old Password** field.
4. Enter your new password in the **Password** field.
5. Re-enter your new password in the **Confirm Password** field.
6. Click **Change Password** to submit the change.

Note:

Password must be between 8 and 128 characters and contain at least one of the each following:

- Lower case letters (a b c d . . .)
- Upper case letters (A B C D . . .)
- Numbers (0 1 2 3 4 5 6 7 8 9)
- Special characters (! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ‘ { | } ~), space included

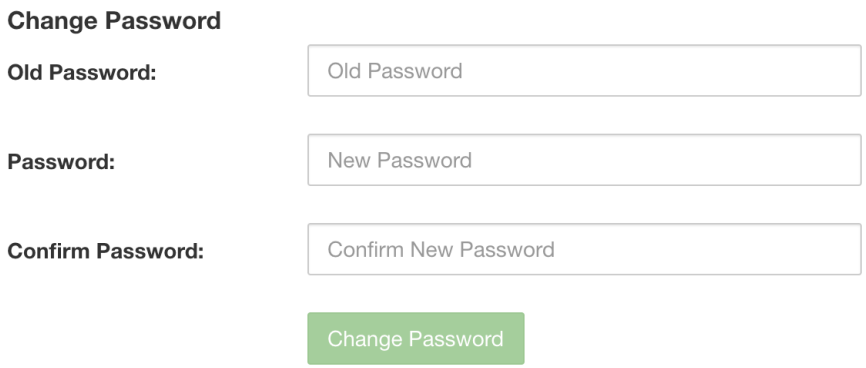


Fig. 16.7.2.1: Change Password Pane

16.7.3 Recovering Passwords

This section explains how to recover your password.

Before You Begin

To reset a password you must first have an account. A new account can be added by **Site Admins** and **Customer Support users**.

1. Point your browser to the Cisco Tetration URL and click the **Forgot Password** link. The **Forgot your password?** dialog appears.

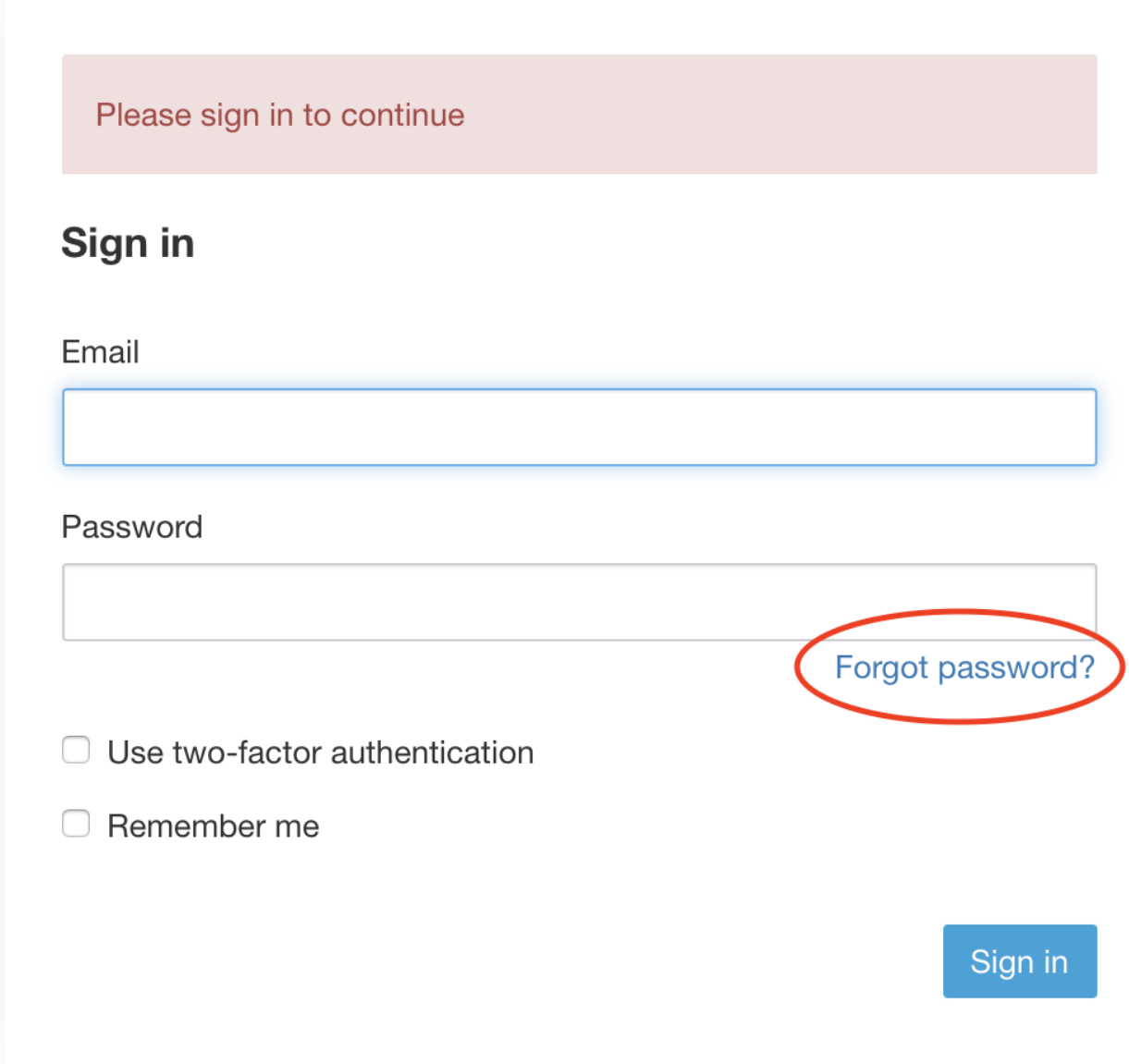


Fig. 16.7.3.1: Sign-in Dialog

2. Enter your email address in the **Email** field.
3. Click **Reset Password**.

Password reset instructions will be sent to your email.

Note: The password recovery procedure for two-factor authentication requires contacting Cisco Tetration Customer Support because the email-based password recovery cannot contain the one-time password.

16.7.4 Enabling Two-Factor Authentication

This section explains how to enable two-factor authentication.

1. Click on the **gear menu** in the top-right corner.

2. Select **Preferences**. The **Preferences** page appears.
3. In the **Two-Factor Authentication** pane, click the **Enable** button. A new **Two-Factor Authentication** pane appears.
4. Enter your password.
5. Scan the QR code displayed under the **Current Password** field using any time-based one-time password (TOTP) app, such as Google Authenticator (for Android or iOS) or Authenticator (for Windows Phone).
6. Enter the validation code shown by your chosen TOTP app.
7. Click **Enable**.

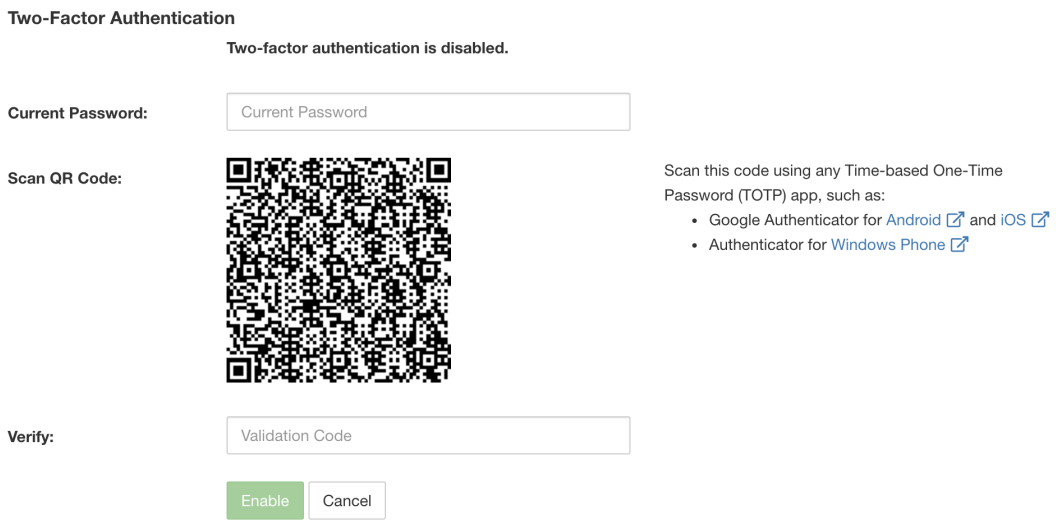


Fig. 16.7.4.1: Two-Factor Authentication Pane

The next time you log into the system, you will need to select the **Use two-factor authentication** check box and enter the verification code shown in your TOTP app to sign in.

Note: The password recovery procedure for two-factor authentication requires contacting Cisco Tetration Customer Support because the email-based password recovery cannot contain the one-time password.

16.7.5 Disabling Two-Factor Authentication

This section explains how to disable two-factor authentication.

1. Click on the **gear menu** in the top-right corner.
2. Select **Preferences**. The **Preferences** page is displayed.
3. Under two-factor authentication, click the **Disable** button. The **Two-Factor Authentication** pane appears.
4. Enter your password.
5. Click the **Disable** button again.

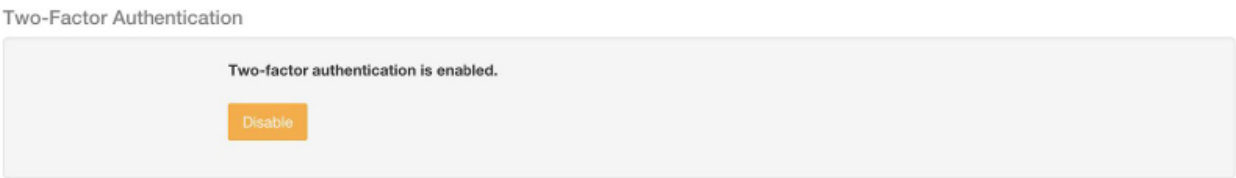


Fig. 16.7.5.1: Two-Factor Authentication Disable Button

You will no longer be required to enter a two-factor verification code during login.

16.8 Roles

Roles are used to implemented a role-based access control (RBAC) model so features and data can be restricted to sets of users.

- User - someone with login access to Cisco Tetration.
- Role - user created set of capabilities that can be assigned to a user
- Capability - a scope + ability pair
- Ability - collections of actions
- Action - low level user action such as "change application name"

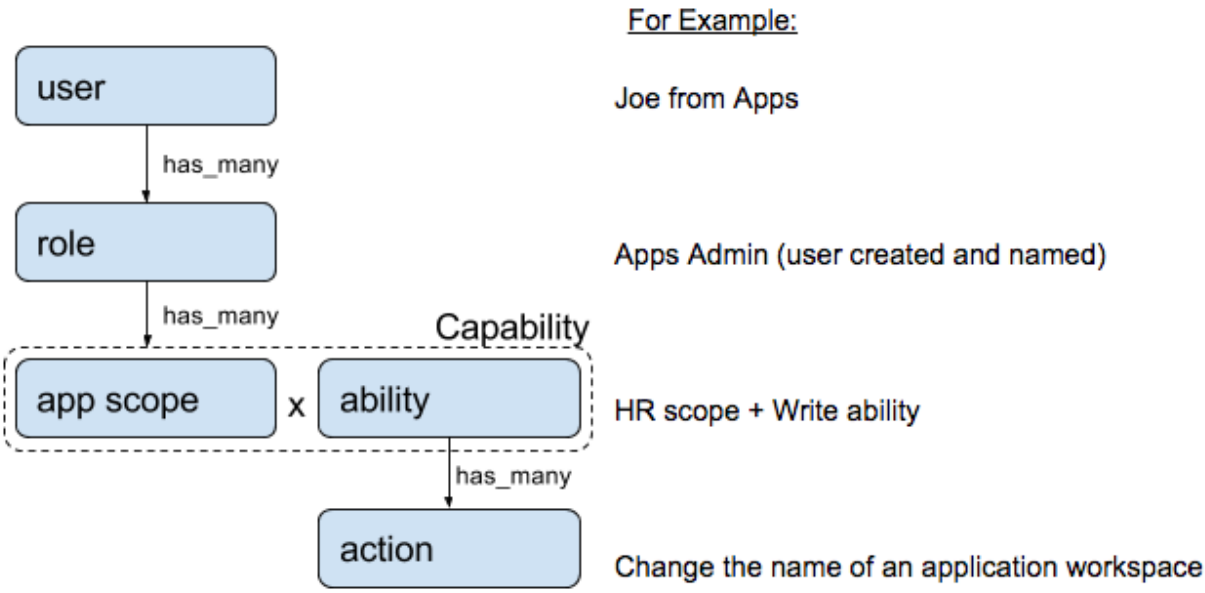


Fig. 16.8.1: Role model

A user can have any number of roles. Roles can have any number of capabilities. For example, the "HR Search Engineer" role could have two capabilities: "Read on the HR Scope" to give visibility and context and "Execute on "HR:Search" capability to allow the engineers assigned this role to make specific changes to their application.

Roles contain sets of Capabilities and are assigned to users on the **Users** page. A user can have any number of roles. Roles can have any number of capabilities.

Six system roles are defined to allow users to get started more quickly. They define different levels of access to **all Scopes**, ie. all data on the system. These system roles are defined below.

Role	Description
Agent Installer	Provide the ability to manage agents life cycle including install, monitor, upgrade and convert, but **can not** delete agents and access agent config profile.
Customer Support	For Technical Support or Advanced Services. Provides access to cluster maintenance features. Allows the same access as Site Admin, but **can not** modify users.
Site Admin	Provides the ability to manage users, agents, etc. Can view and edit all features and data. There must be at least one site admin.
Global Application Enforcement	Provides the Enforce ability on every scope.
Global Application Management	Provides the Execute ability on every scope.
Global Read Only	Provides the Read ability on every scope.

16.8.1 Abilities and Capabilities

Roles are made up of Capabilities which include a Scope and an Ability. These define the allowed actions and the set of data they apply to. For example, the (HR, Read) capability should be read and interpreted as "Read ability on the HR scope". This capability would allow access to the HR scope and all of its children.

Ability	Description
Read	Read all data including flows, application and inventory filters.
Write	Make changes to applications and inventory filters.
Execute	Perform ADM runs and publish policies for analysis.
Developer	Access to Data Platform features such as creating and running User Apps, scheduling Jobs, and uploading data to the Da
Enforce	Enforce policies defined in application workspaces associated with the given scope.
Owner	Required to toggle an application workspace from secondary to primary. Access to Data Tap Admin abilities such as ma

Important: Abilities are inheritied, eg. the Execute ability allows all the Read, Write and Execute actions.

Important: Abilities apply to the scope and all of the scope's children.

16.8.2 Creating a New Role

This section explains how **Site Admins** and **Customer Support users** can create new roles.

Before You Begin

You must already have scope owner capability, Site Admin or Customer Support role.

1. Click on the **gear menu** in the top-right corner.
2. Select **Roles**. The **Roles** page is displayed with a table containing the list of roles.
3. Click the **Create New Role** button. The **Roles** panel appears.

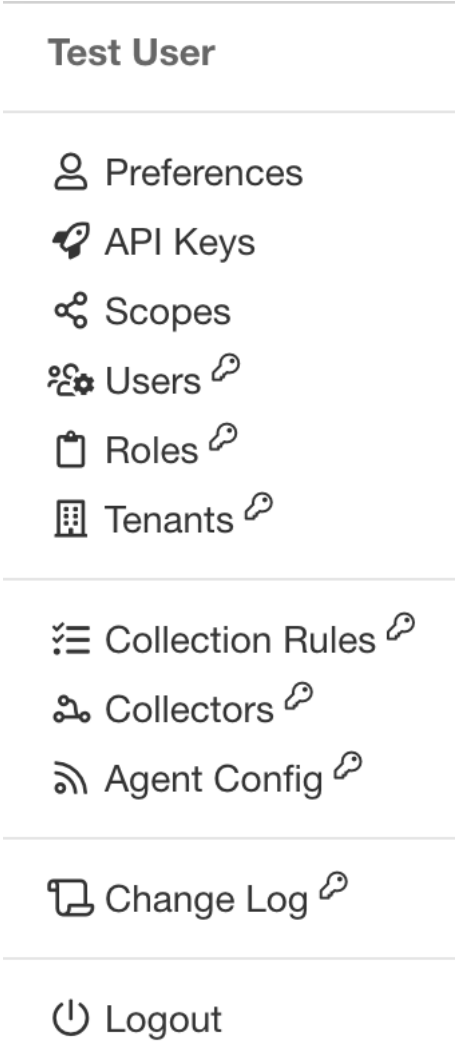


Fig. 16.8.2.1: Menu

Creating a role using the Create Role Wizard is a three-step process.

Step 1:

1. Enter the appropriate values in the following fields:

Field	Description
Name	The name to identify the role.
Description	A short description to add context about the role.

2. Click the **Next** button to move to the next step or **Back to Roles Page** to go back to Roles Page.

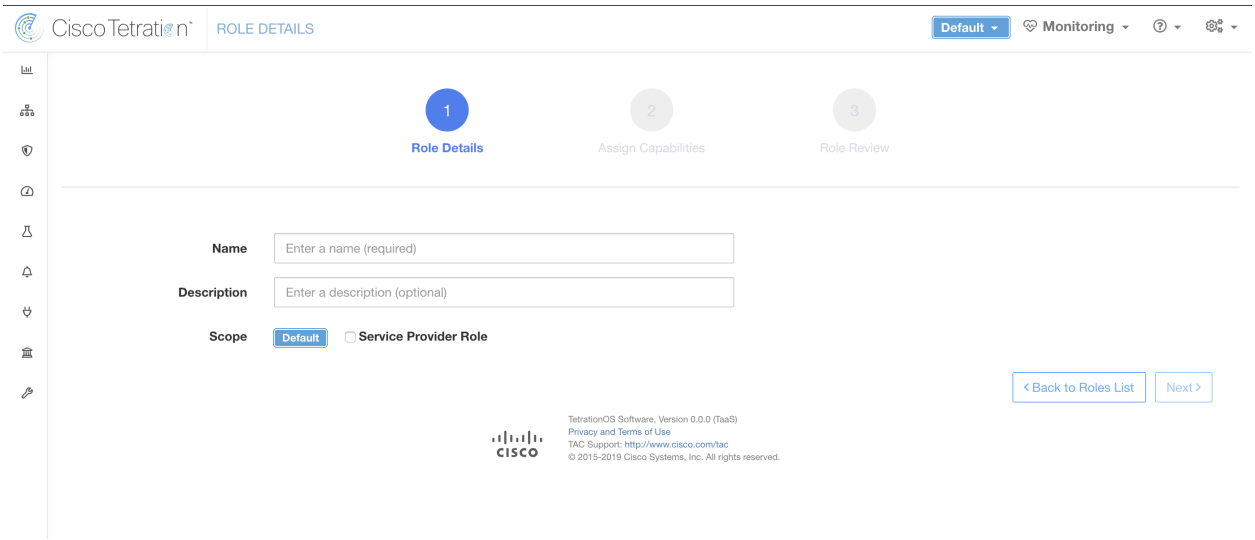


Fig. 16.8.2.2: Role Details

Step 2:

1. Click the **Add Capability** button to show a creation form in the top row.
2. Select a scope and ability.
3. Click the **Checkmark** button to create a new capability or **Cancel** button to cancel.
4. Click **Next** button to review role details or **Previous** to go back and edit.

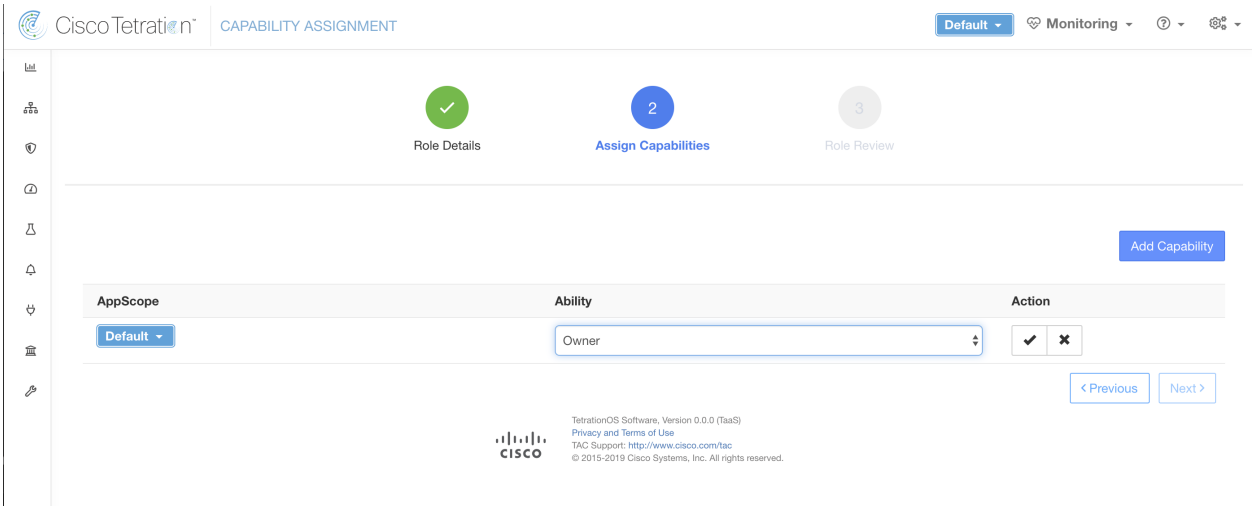


Fig. 16.8.2.3: Capability Assignment

Step 3:

1. Review the role details and capabilities.
2. Click **Create** to create role.

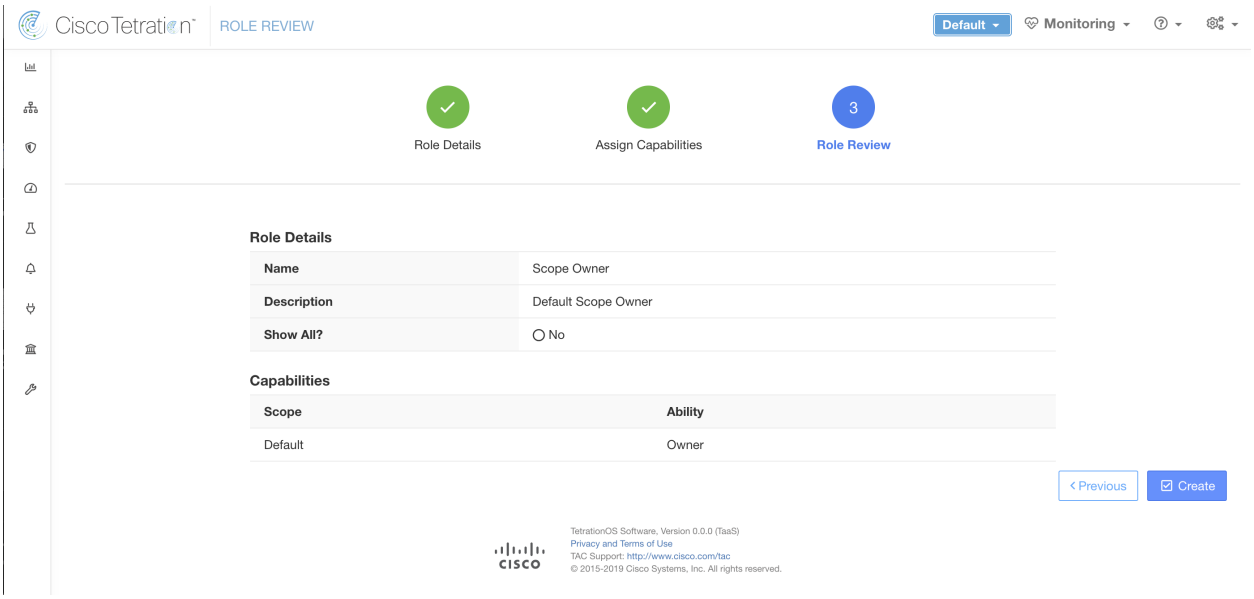


Fig. 16.8.2.4: Role Review

16.8.3 Editing a Role

This section explains how **Site Admins** and **Customer Support users** can edit roles.

Before You Begin

You must be Site Admin or Customer Support User.

1. Click on the **gear menu** in the top-right corner.
2. Select **Roles**. The **Roles** page is displayed with a table containing the list of roles.
3. In the row of the role to edit, click the **Edit** button in the right hand column. The **Roles** panel appears.

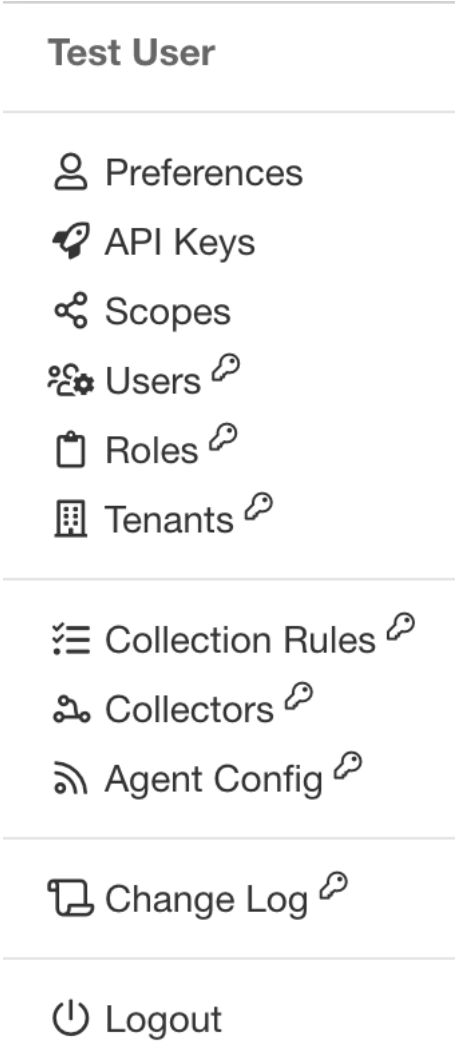


Fig. 16.8.3.1: Menu

Editing a role using the Edit Role Wizard is a three-step process.

Step 1:

1. Update the name or description if desired.
2. Click the **Next** button to move to the next step or **Back to Roles Page** to go back to Roles Page.

Step 2:

1. Remove any capability as needed. In the row of the capability to delete, click the **Delete** icon in the right hand column.
2. To add, click the **Add Capability** button to show a creation form in the top row.
3. Select a scope and ability.
4. Click **Next** button to review role details or **Previous** to go back and edit.

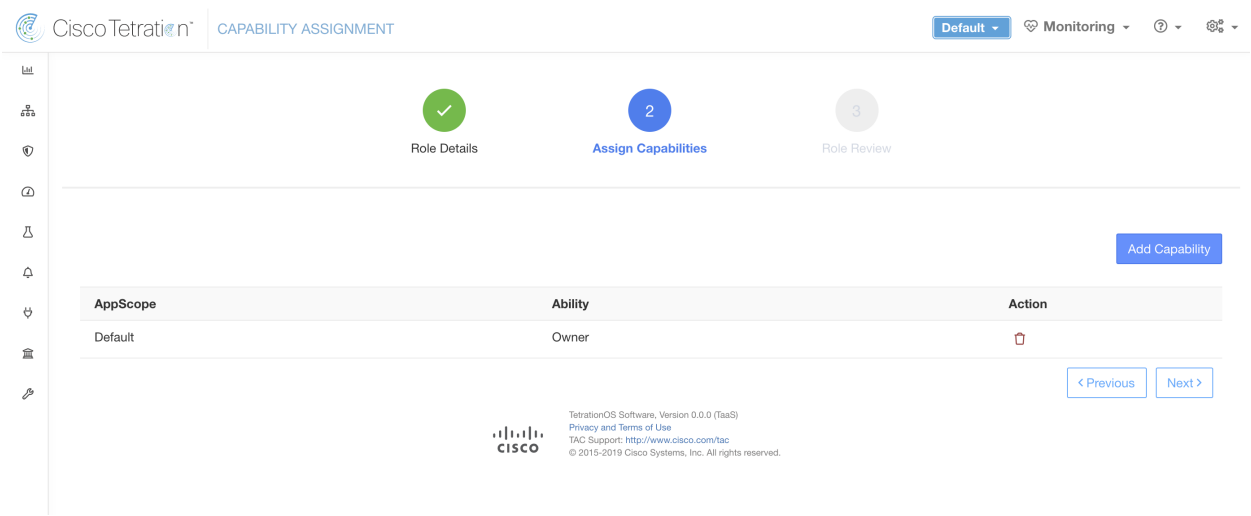


Fig. 16.8.3.2: Capability Assignment

Step 3:

1. Review the role details and capabilities.
2. Click **Update** to create the role or **Previous** to go back and edit. Changes to role details and capability assignment are saved after **Update**.

Note: Capabilities can not be edited, they must be deleted and recreated.

16.8.4 Change Log

Site Admins and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each role by clicking on the icon in the **Action** column as shown below.

Fig. 16.8.4.1: Change Log

These users can also view a list of deleted roles by clicking on the **View Deleted Roles** link below the table.

For more information on the **Change Log** see *Change Log*. Root scope owners are restricted to viewing change log entries for entities belonging to their scope.

16.9 Scopes

Note: The **Scopes** page has been merged with **Inventory Search**. See the **Scopes and Inventory** page for help with the link below.

Scopes and Inventory

16.10 Tenants

Site Admins and **Customer Support users** can access the **Tenants** page under the **gears menu**. This page displays all of the currently configured Tenants and VRFs. The system comes preconfigured with one or more Tenants and VRFs. Tenants can be added, edited, and deleted.

Note: These values will affect the results of the cluster output. We recommend consulting Cisco TAC before making changes to these values to understand the system impact

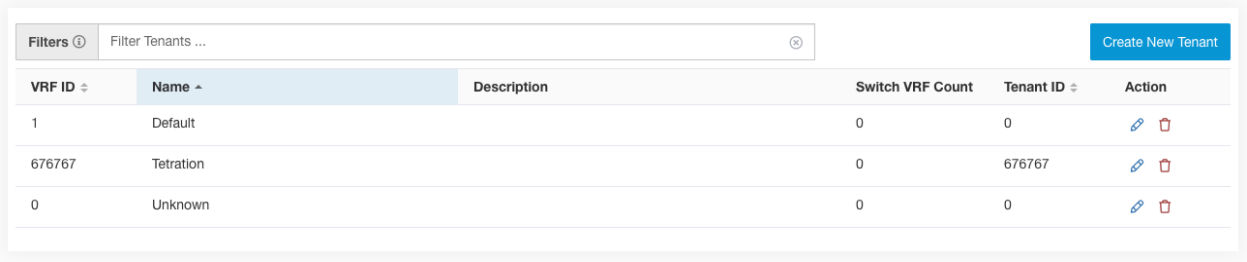


Fig. 16.10.1: Tenants Page

16.10.1 Adding a Tenant

This section explains how **Site Admins** and **Customer Support users** add new tenants.

Before You Begin

You must be **Site Admin** or **Customer Support user**.

1. Click the **Settings menu** in the top-right corner.
2. Select **Tenants**. The **Tenants** page appears.
3. Click **Create New Tenant**. The **Create Tenant** modal opens.
4. Enter the appropriate values in the following fields:

Field	Description
Name	Enter a desired name for the tenant.
Description	(optional) The description field contains additional information about the tenant.
Switch VRFs	(optional) Configure this feature to map multiple hardware (switch) VRFs to one Tetration Root Scope/VRF. Detailed explanation below

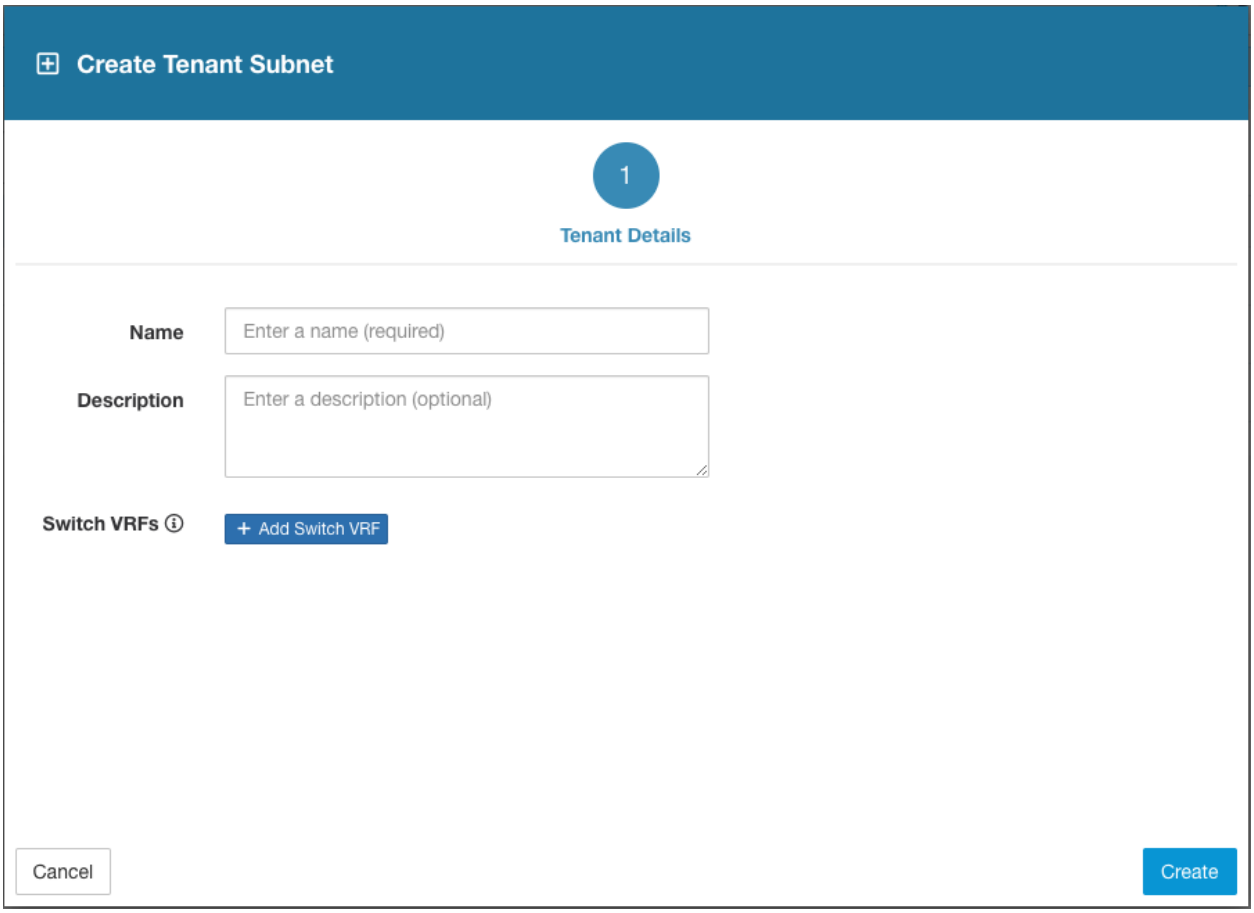


Fig. 16.10.1.1: Add Tenant Modal

5. Click **Create**.

16.10.2 Editing a Tenant

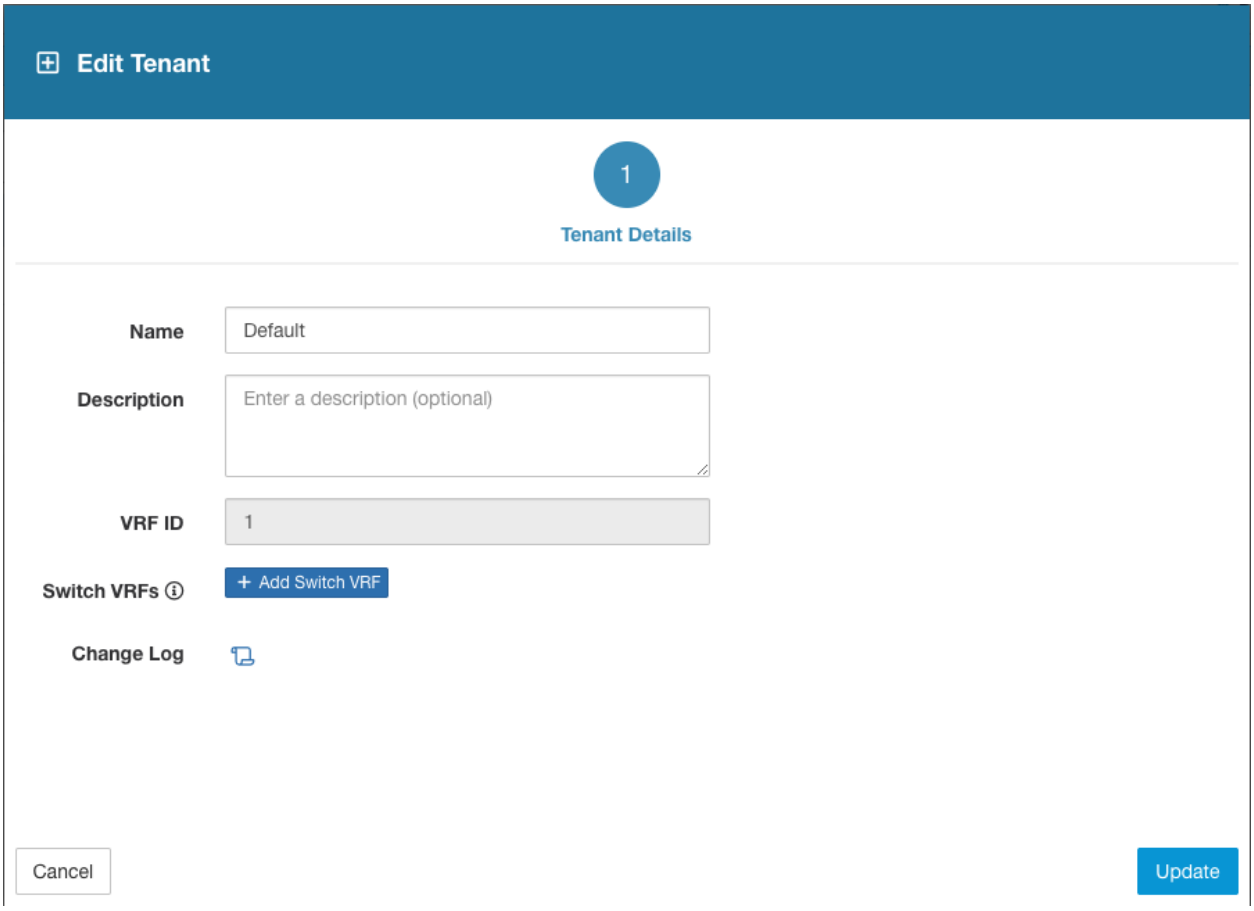
This section explains how **Site Admins** and **Customer Support users** edit tenants.

Before You Begin

You must be **Site Admin** or **Customer Support user**.

1. Click the **Settings menu** in the top-right corner.
2. Select **Tenants**. The **Tenants** page appears.
3. Find the tenant you want to edit and click the **pencil** icon in the column on the right.

Field	Description
Name	Update a name for the tenant.
Description	(optional) Update the description field contains additional information about the tenant.
VRF ID	Displays the ID for this particular Tenant/VRF.
Switch VRFs	(optional) Update configuration to map multiple hardware (switch) VRFs to one Tetration Root Scope/VRF. Detailed explanation below
Change log	Clicking on change log icons takes you to a new page which shows all the change log for the Tenant/VRF.



4. Click **Update**.

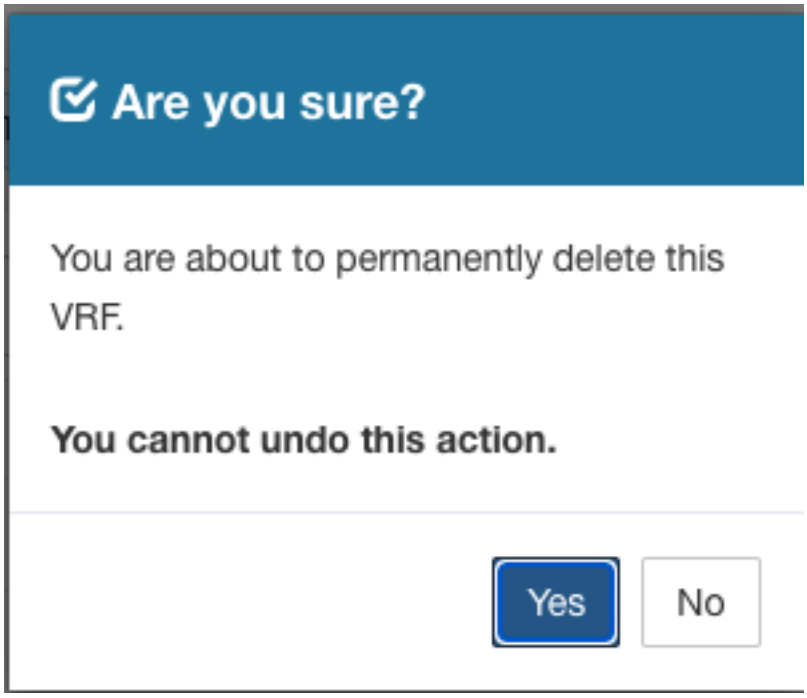
16.10.3 Deleting a Tenant

This section explains how **Site Admins** and **Customer Support users** delete tenants.

Before You Begin

You must be **Site Admin** or **Customer Support user**.

1. Click the **Settings menu** in the top-right corner.
2. Select **Tenants**. The **Tenants** page appears.
3. Find the Tenant you want to delete and click the **trash can** icon in the column on the right.



4. Click **Yes** on the confirm dialog prompt.

16.10.4 Adding Switch VRFs to a Tenant

Configure this feature to map multiple hardware (switch) VRFs to one Tetration Root Scope/VRF. Tetration's ingest data path (collectors) will map the hardware VRFs to the one Tetration VRF.

> **Warning:** This feature works when all the hardware VRFs being mapped have no overlapping IPs. If the switch VRFs have overlapping IPs this feature should not be used.

Switch VRFs can be added to a VRF by entering Switch VRF name and clicking the **Checkmark** icon in the **Add/Edit Tenant Modal as shown below**.

1. Enter the switch vrf name and click the check button shown below.

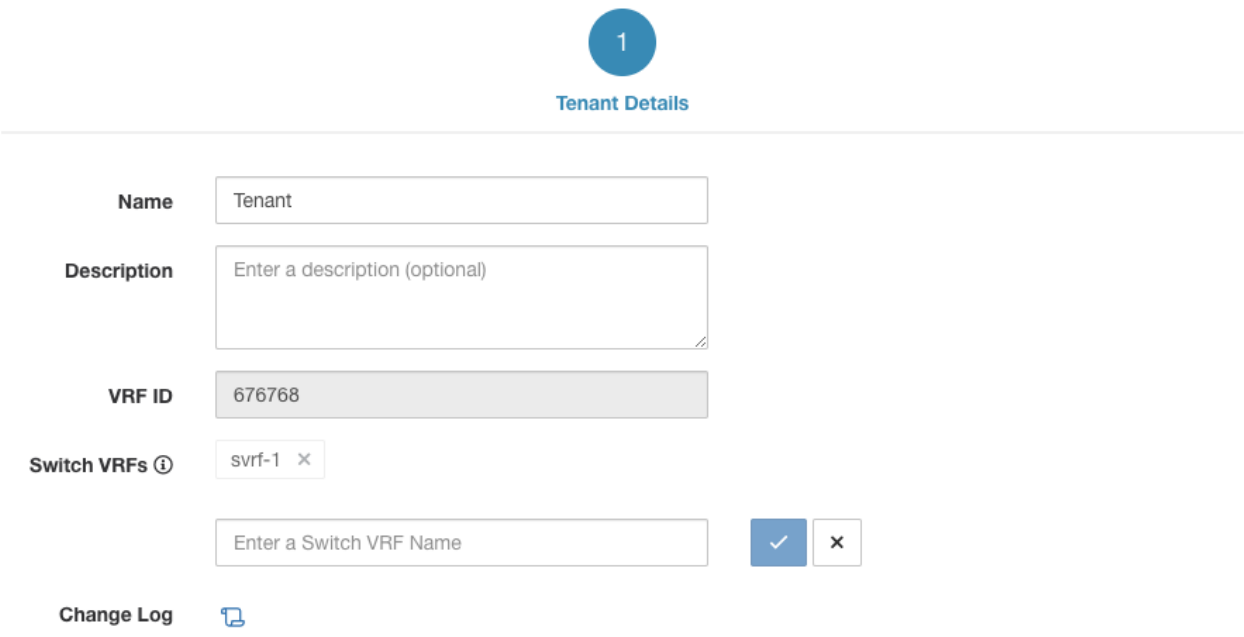


Fig. 16.10.4.1: Add Switch VRFs to a VRF

2. Click **Create/Update** button to save the switch VRF.

16.10.5 Removing Switch VRFs

Switch VRFs can be removed from a VRF by clicking the **x** button next to the switch VRF label in the **Add/Edit VRF Dialog**.

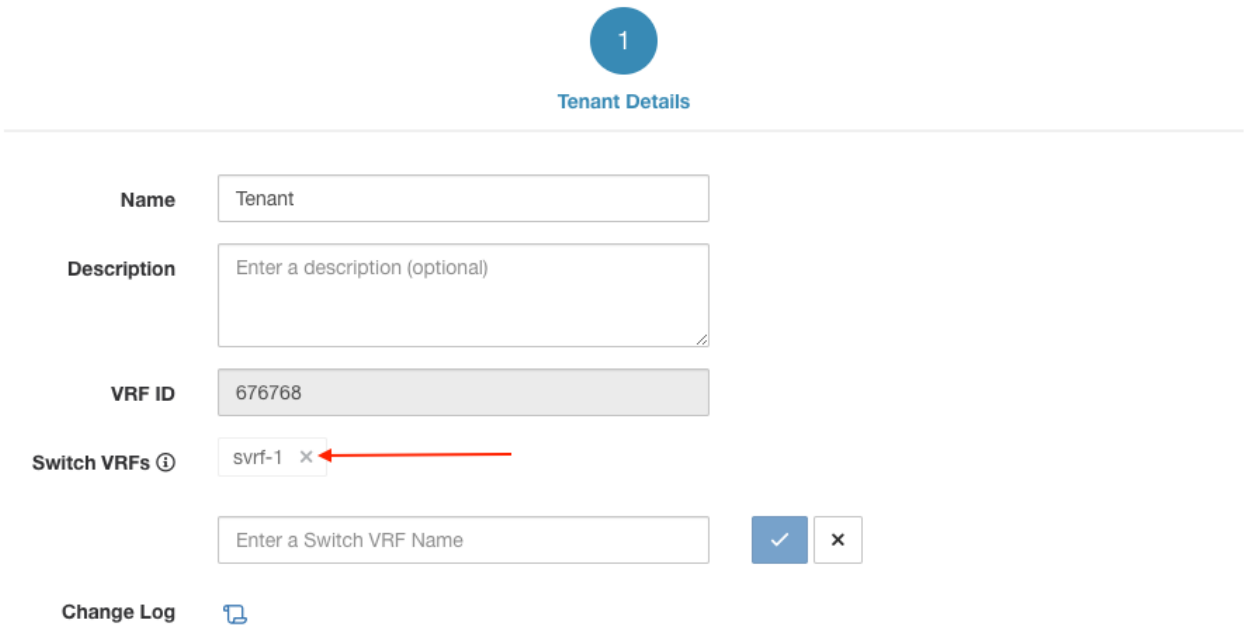


Fig. 16.10.5.1: Removing Switch VRFs

Click **Create/Update** button to save changes.

16.11 Users

Site Admins and Root Scope Owners can access the **Users** page under the **Settings menu**.

This page will show all Service Provider users and those associated with the scope selected in the page header.

Multitenancy

To support multitenancy, users can be assigned to a root scope. These users can be managed by users with the 'Owner' ability on the root scope and can only be assigned roles associated with the same scope.

Users without a scope are called 'Service Providers' and they can be assigned any role allowing them to preform actions across root scopes.

16.11.1 Adding a New User Account

This section explains how **Site Admins** and Users with the **''SCOPE_OWNER''** ability on the root scope can add new user accounts.

If a user is assigned a scope for the purpose of multitenancy, only roles assigned to the same scope may be selected.

Note: This page is filtered by the scope preference selected in the page header.

Before You Begin

1. Click on the **gear menu** in the top-right corner.
2. Select **Users**. The **Users** page is displayed.
3. Click the **Add New User** button. The **Users** wizard appears.

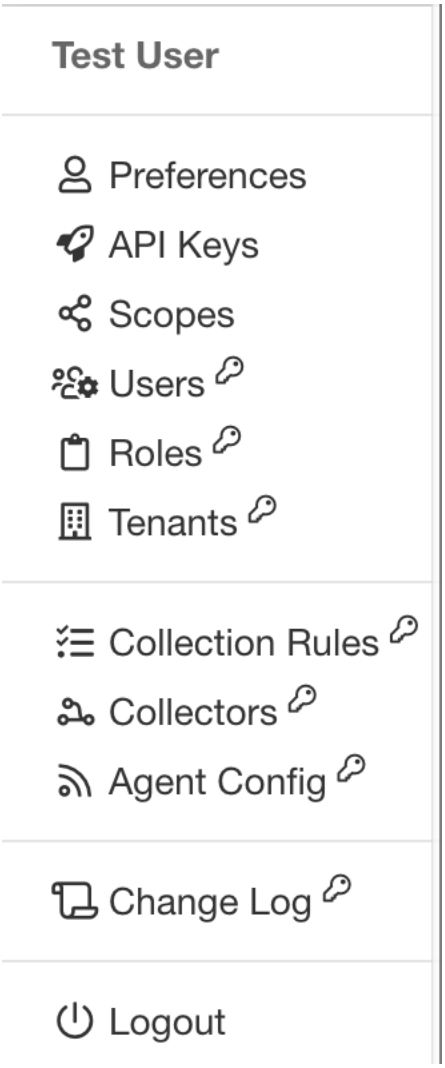


Fig. 16.11.1.1: Menu

User creation is a three-step process.

Step 1:

1. Enter the appropriate values in the following fields:

Field	Description
Email	Enter the new user's email address, it is non case-sensitive. We will use the lower cased version of your email if it contains letters.
First Name	Enter the new user's first name.
Last Name	Enter the new user's last name.
Scope	Root Scope assigned to the user for multitenancy.

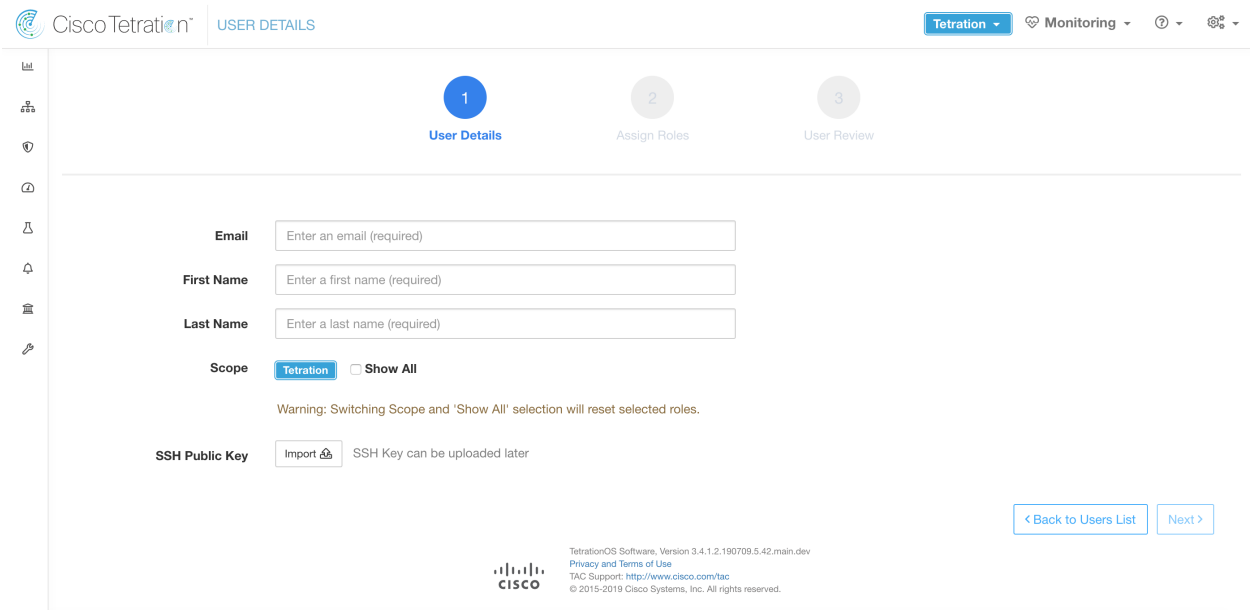


Fig. 16.11.1.2: User Details

2. Click the **Next** button to move to the next step or **Back to Users List** to go back to the Users Page.

Step 2: In this view you can 'Add Roles', 'Delete Roles' or 'Select Roles':

1. Click on **Add Roles** to assign Roles.

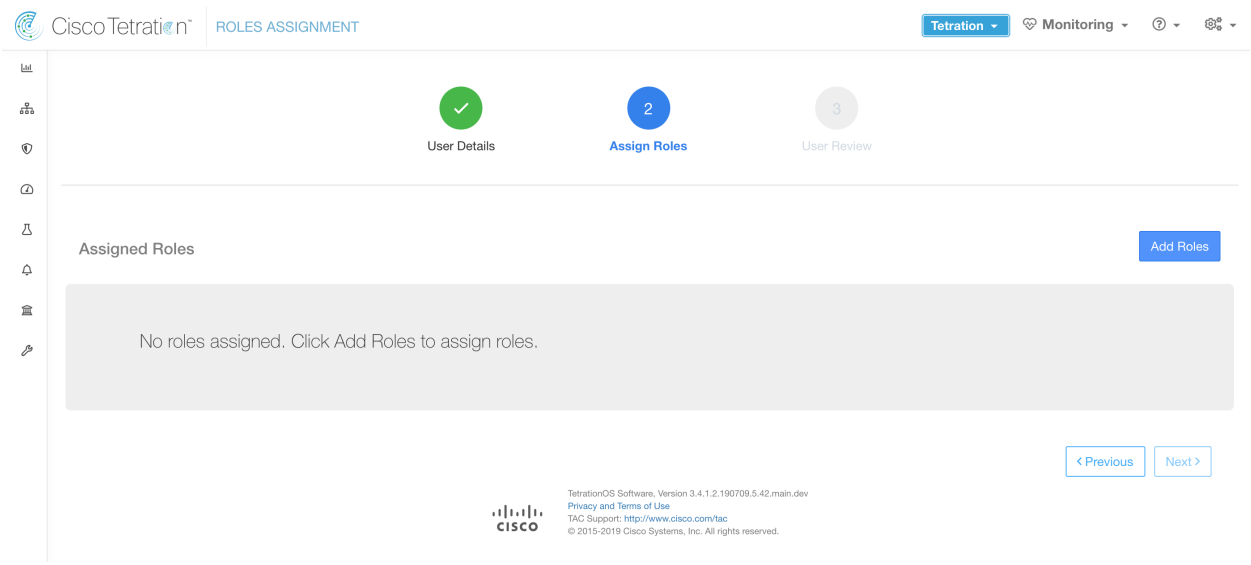


Fig. 16.11.1.3: Assigned Roles

2. Click on **Edit Assigned Roles** to delete them.

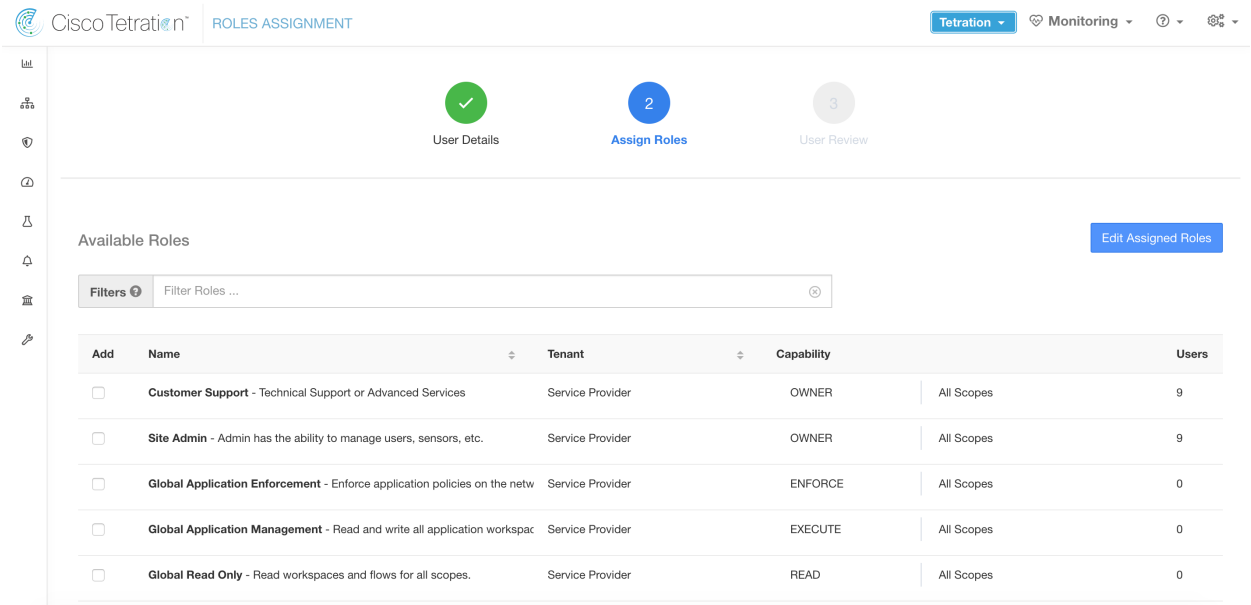


Fig. 16.11.1.4: Available Roles

3. Filter roles by Name and Tenant.

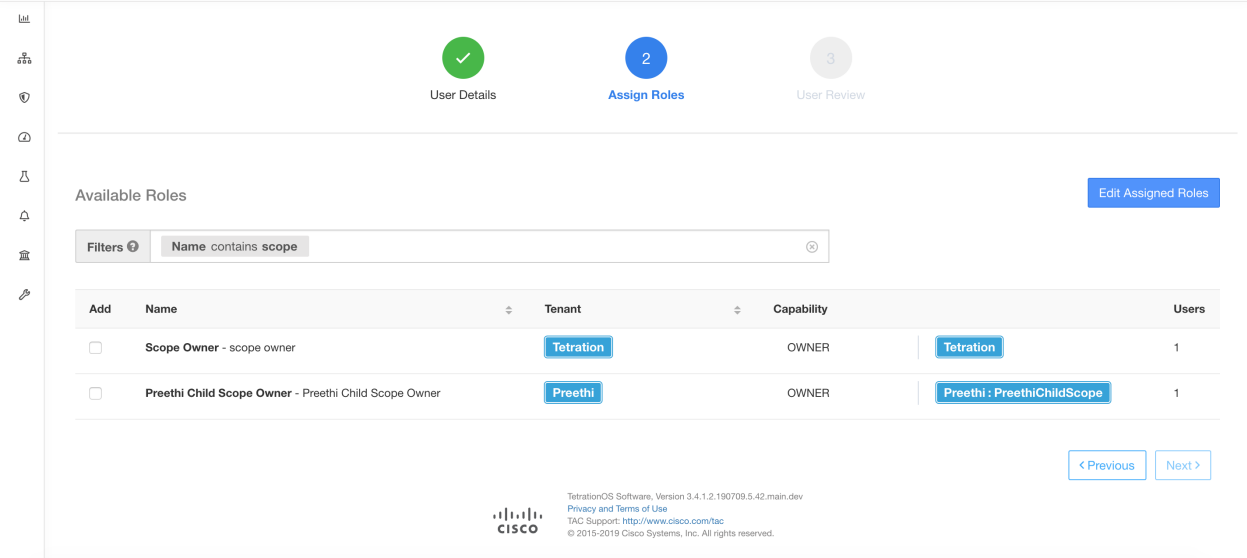


Fig. 16.11.1.5: Filter Roles

4. Click **Next** button to review the user details and role assignment or **Previous** button to go back and edit details.

Step 3: Review selections and click **Create**.

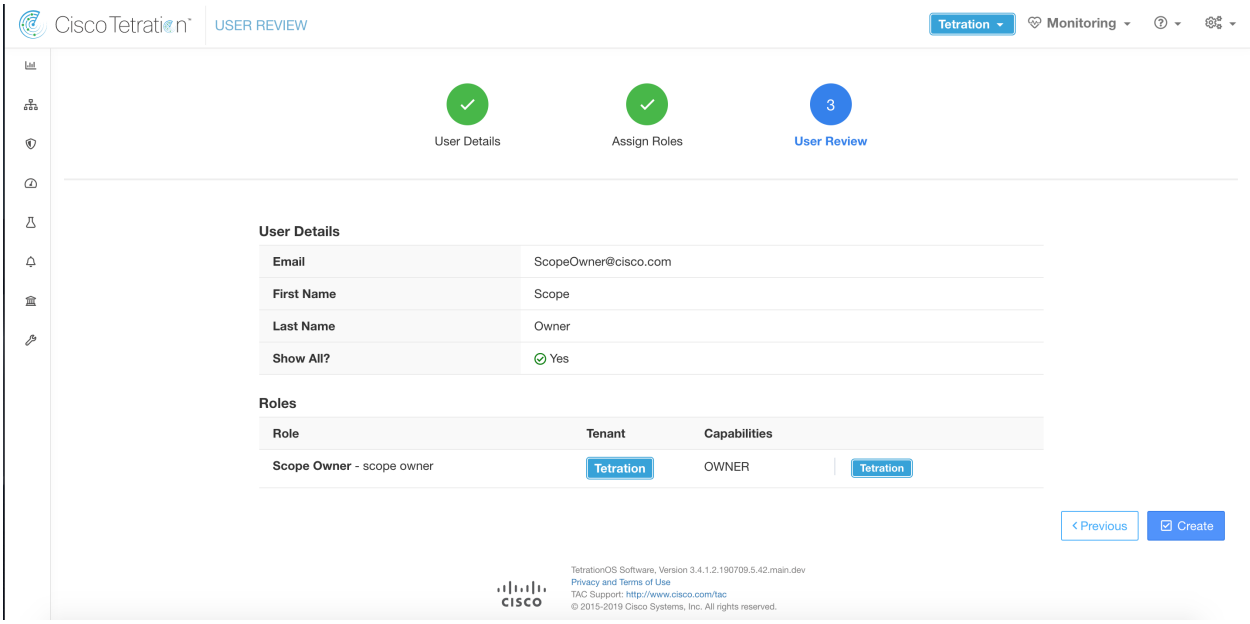


Fig. 16.11.1.6: Review User Details

If external auth is enabled, authentication details are displayed.

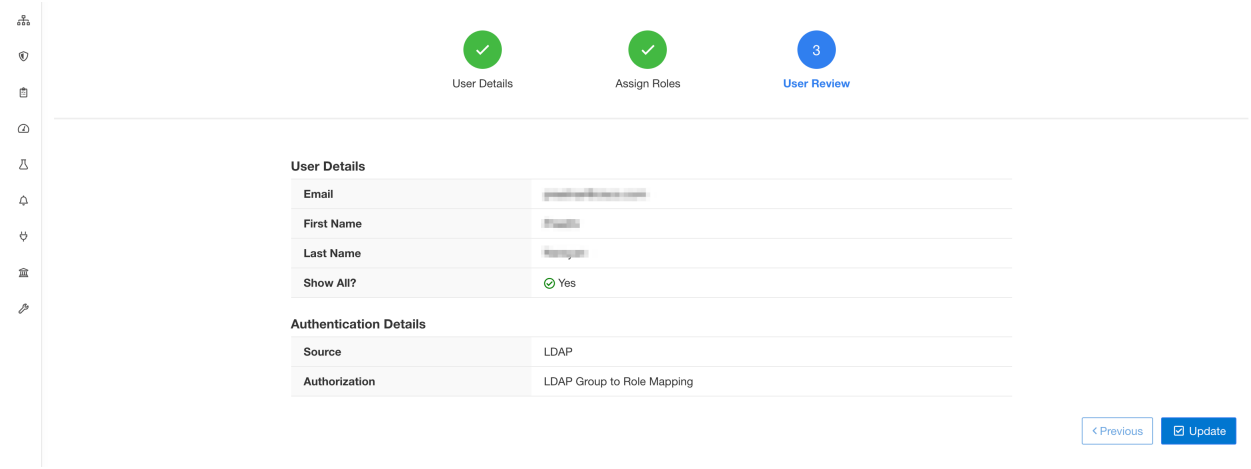


Fig. 16.11.1.7: User Details with Authentication if External Authentication is enabled

Note: After user creation, the user will receive an email to set up password.

16.11.2 Editing a User Account

This section explains how **Site Admins** or **Root Scope Owners** can edit user accounts.

Note: This page is filtered by the scope preference selected in the page header.

Before You Begin

1. Click on the **gear menu** in the top-right corner.
2. Select **Users**. The **Users** page is displayed with a table containing the list of registered users.
3. In the row of the account you want to edit, click **Edit** button in the right hand column. The **Users** Wizard appears.

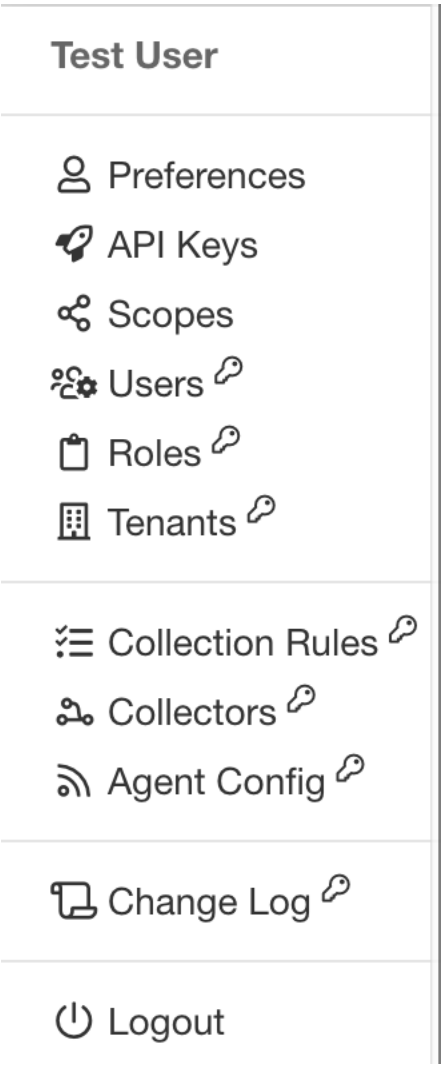


Fig. 16.11.2.1: Menu

Editing user using the wizard is a three-step process.

Step 1:

1. Update the following fields, if desired:

Field	Description
Email	Update the new user's email addres
First Name	Update the new user's first name.
Last Name	Update the new user's last name.
Scope	Root Scope assigned to the user for multitenancy. (available to site admins)

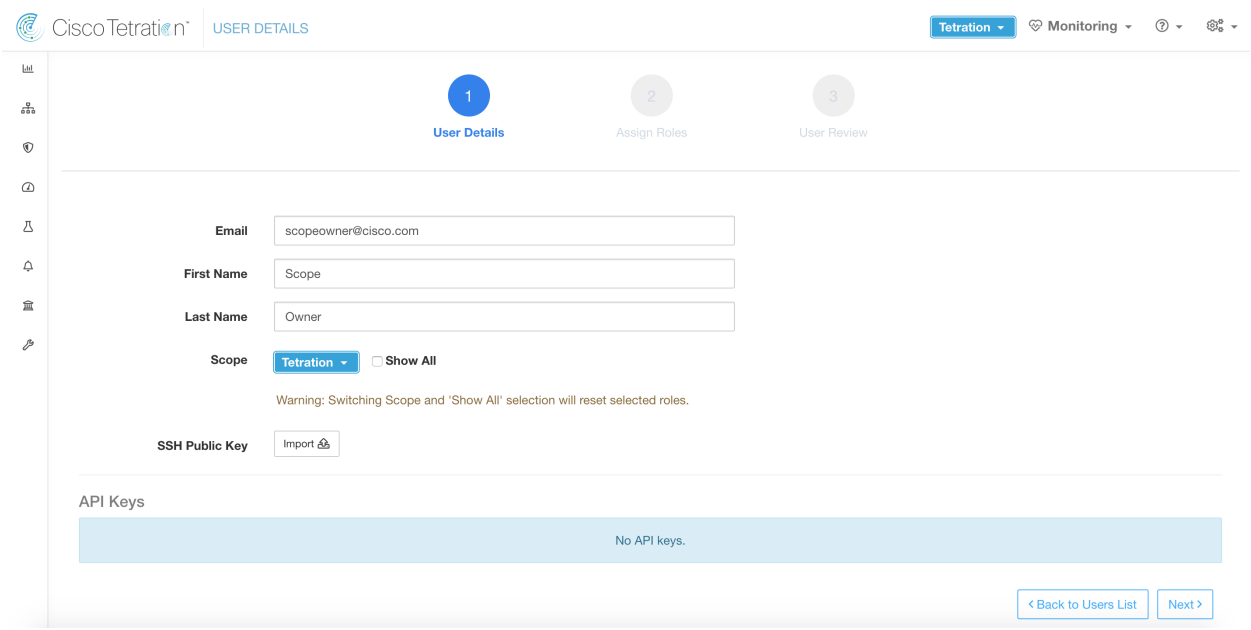


Fig. 16.11.2.2: User Details

2. Click **Next** button to go to Role Assignment.

Step 2:

1. In this view, assigned roles can be removed.
2. Click on **Add Roles** to assign new roles.

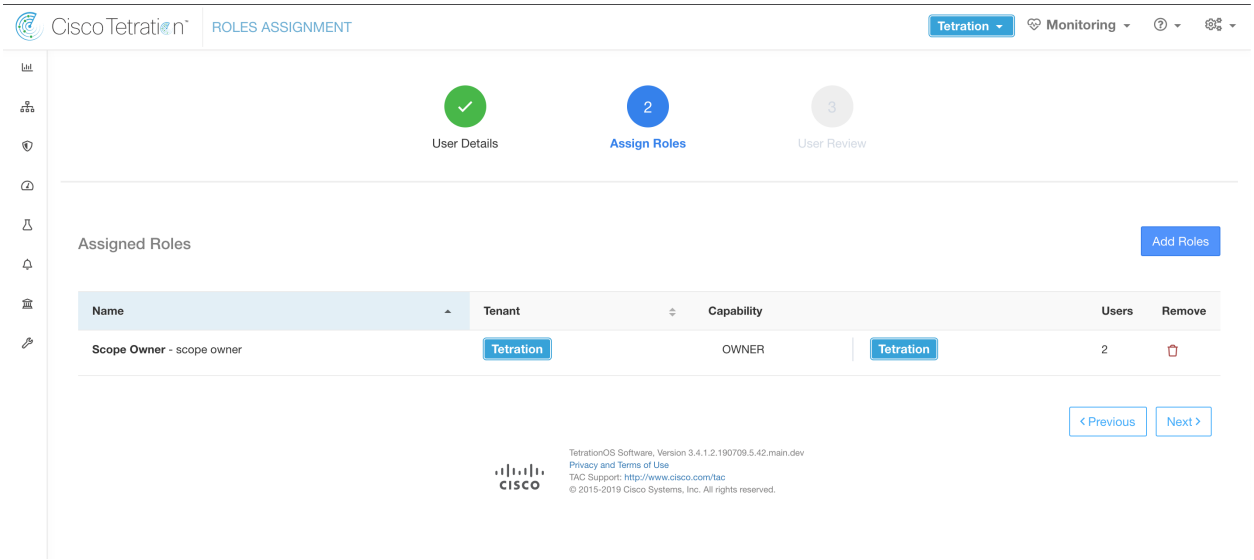


Fig. 16.11.2.3: Remove Assigned Roles

3. Click **Next** button to review the user details and role assignment or **Previous** button to go back and edit details.

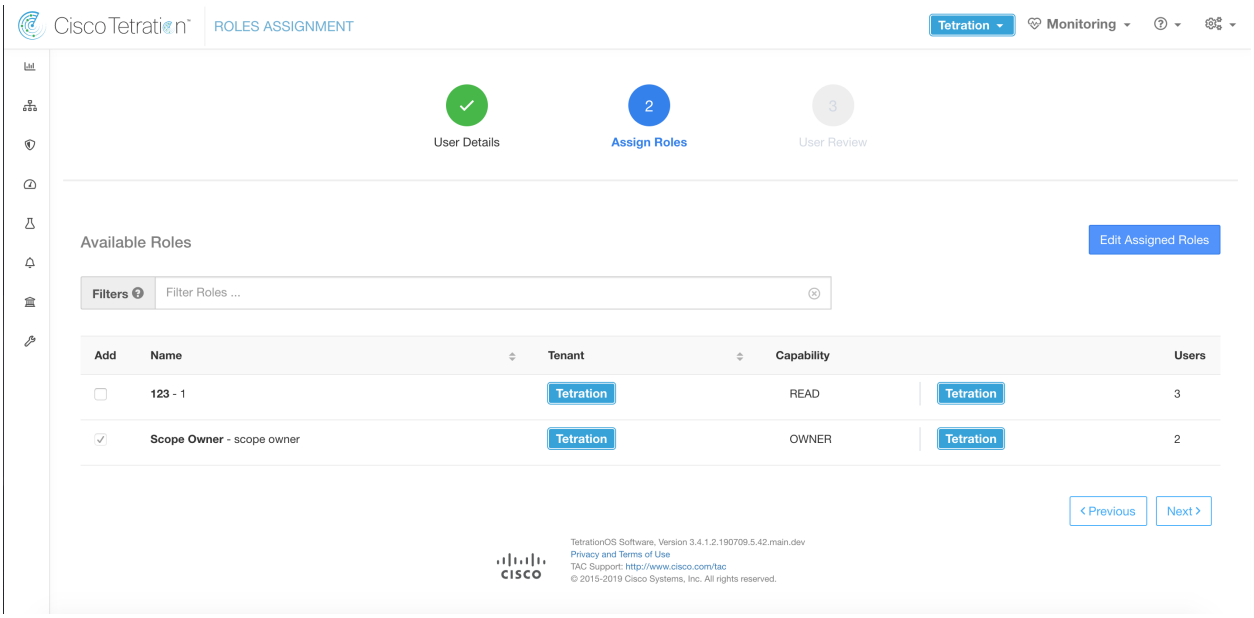


Fig. 16.11.2.4: Assign New Roles

Step 3:

1. Review the user details and role assignment.
2. Click **Update** to update the user or **Previous** to go back and edit roles. Changes to user details and role assignment are saved.

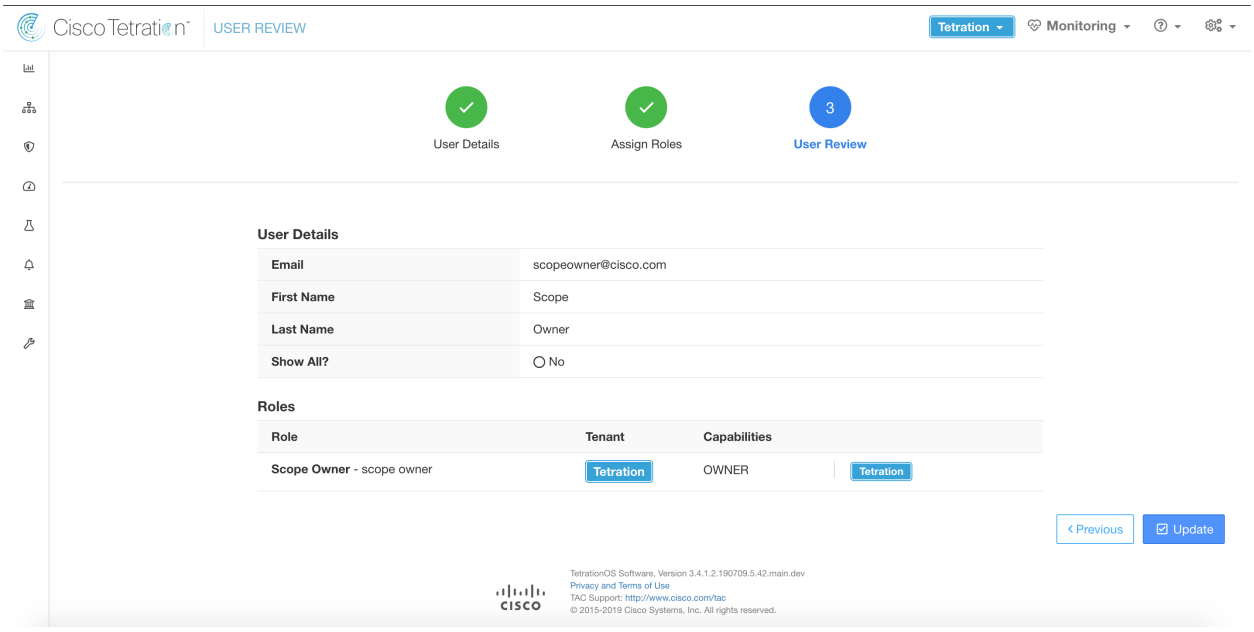


Fig. 16.11.2.5: User Review

If external auth is enabled, authentication details are displayed.

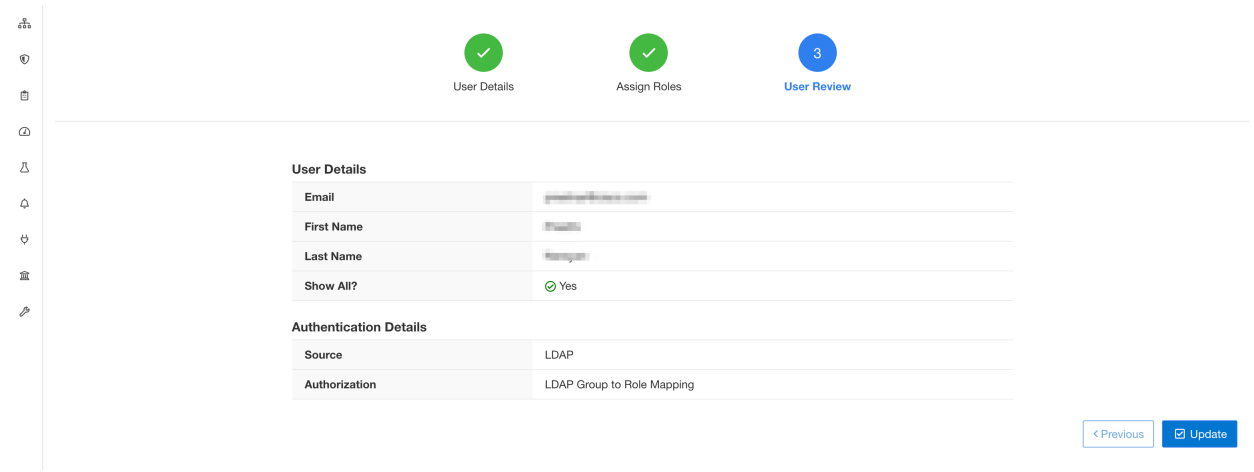


Fig. 16.11.2.6: User Details with Authentication if External Authentication is enabled

16.11.3 Importing SSH Public Key

To enable SSH access as **ta_guest** user via one of the collector IP addresses, SSH public key can be imported for each user. This menu will only be available to **Site Admins** and users with the `SCOPE_OWNER` ability on the root scope . The SSH Public Key will automatically expire in 7 days.

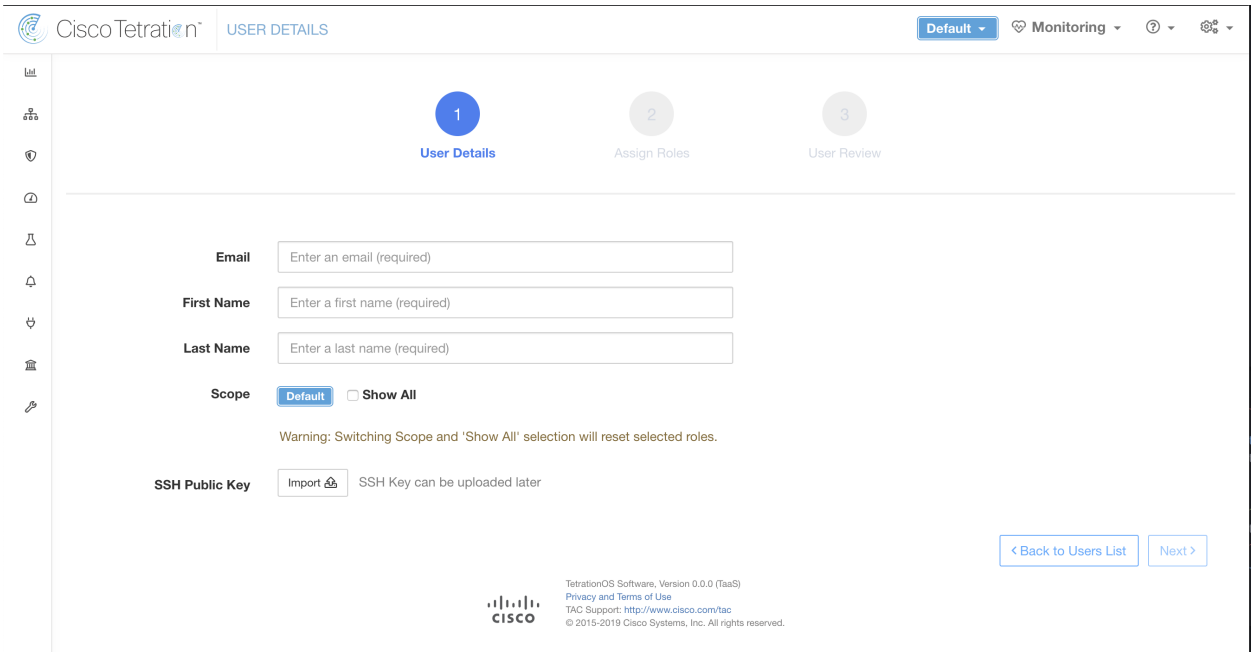


Fig. 16.11.3.1: Import SSH Public Key

16.11.4 Site config in Tetration Setup

This section explains how **Site Admins** can setup a site during the Tetration Setup process.

Field	Description
UI Admin Email	The email address of the individual who will be responsible for administering Tetration within your organization
UI Primary Customer Support Email	The email address of primary support. Must be different from UI Admin Email
Admiral Alert Email	This email address will receive alerts related to the cluster health. Must be different from UI Admin Email and UI Primary Customer Support Email

The email addresses are non case-sensitive. We will use the lower cased version of your email if it contains letters.

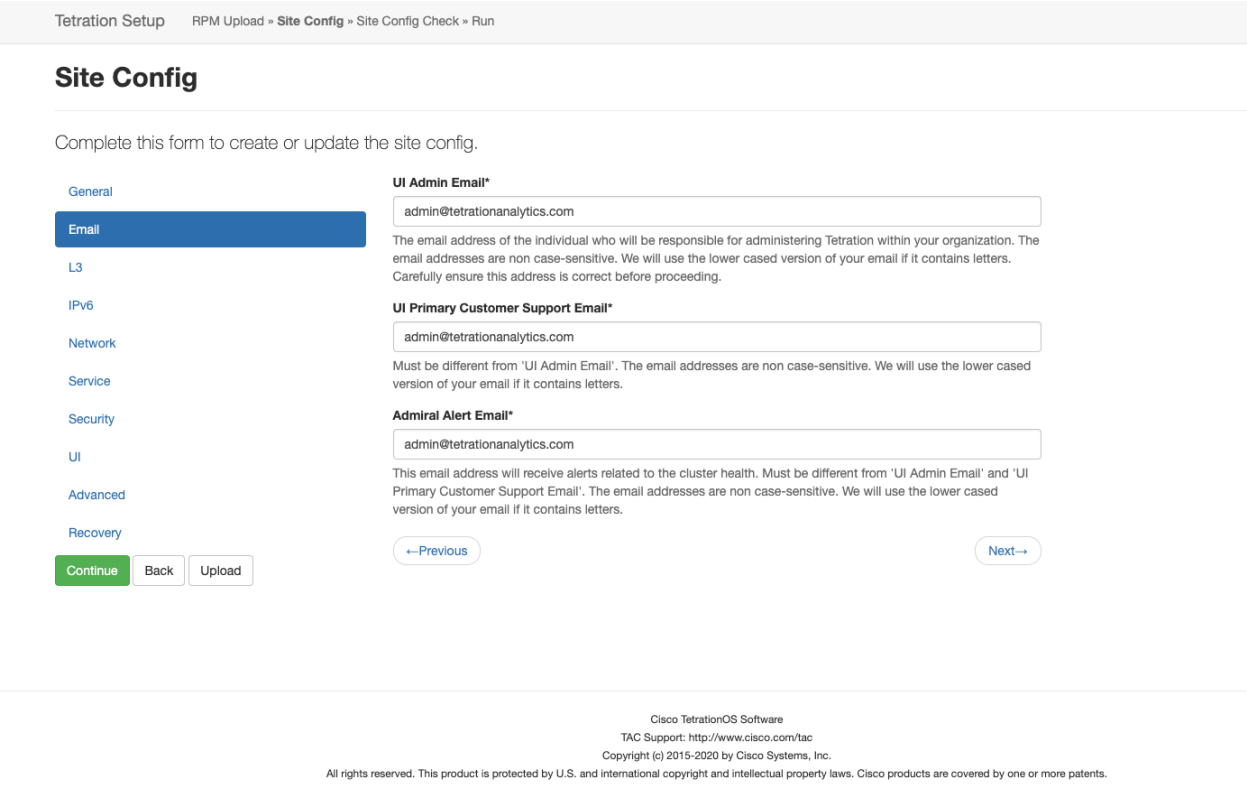


Fig. 16.11.4.1: Configure UI Admin, Primary customer support and Admiral admin alert emails.

16.11.5 Change Log

Site Admins and users with the `SCOPE_OWNER` ability on the root scope can view the change logs for each user by clicking on the icon in the **Actions** column as shown below.

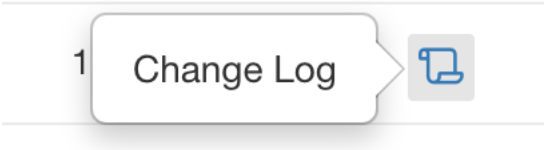


Fig. 16.11.5.1: Change Log

For more information on the **Change Log** see *Change Log*. Root scope owners are restricted to viewing change log entries for entities belonging to their scope.

CHAPTER

SEVENTEEN

OPENAPI

OpenAPI provides a REST API for Tetration features.

17.1 OpenAPI Authentication

OpenAPI uses a digest based authentication scheme. The workflow is as follows:

1. Log into the Tetration UI Dashboard
2. Generate an API key and an API secret with the desired capabilities.
3. Use Tetration API sdk to send REST requests in json format.
4. To use python sdk, user would install the sdk using `pip install tetpyclient`.
5. Once python sdk is installed, here is some boilerplate code for instantiating the RestClient:

```
from tetpyclient import RestClient

API_ENDPOINT="https://<UI_VIP_OR_DNS_FOR_TETRATION_DASHBOARD>"

# ``verify`` is an optional param to disable SSL server authentication.
# By default, |product| appliance dashboard IP uses self signed cert after
# deployment. Hence, ``verify=False`` might be used to disable server
# authentication in SSL for API clients. If users upload their own
# certificate to |product| appliance (from ``Settings > Company`` Tab)
# which is signed by their enterprise CA, then server side authentication
# should be enabled.
# credentials.json looks like:
# {
#   "api_key": "<hex string>",
#   "api_secret": "<hex string>"
# }

restclient = RestClient(API_ENDPOINT,
                credentials_file='<path_to_credentials_file>/credentials.json',
                verify=True)
# followed by API calls, for example API to retrieve list of agents.
# API can be passed /openapi/v1/sensors or just /sensors.
resp = restclient.get('/sensors')
```

17.1.1 Generate API Key and Secret

In the UI dashboard, click on the gears icon in the upper right hand corner, a menu will appear, navigate to API keys.

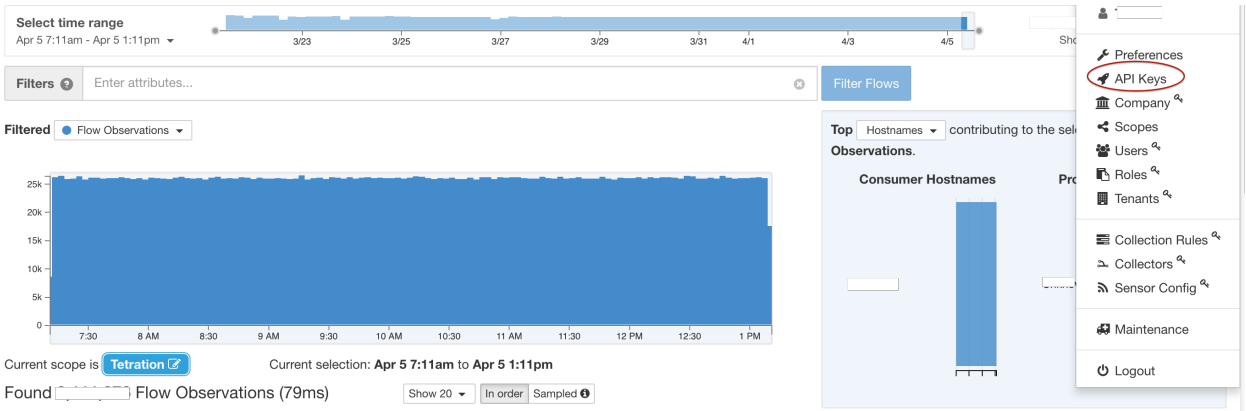


Fig. 17.1.1.1: OpenAPI Dashboard

Click the Create API Key button

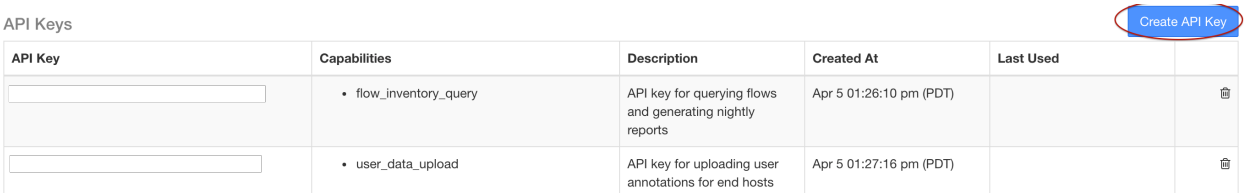


Fig. 17.1.1.2: CreateAPI Key button

Specify the desired capabilities for the key and secret. User must choose the limited set of capabilities that they intend to use the API Key+Secret pair for. Note, the API capabilities available to the user varies based on user's roles, e.g. Site Admin users can generate keys to manage software agents but this capability is not available to not non Site Admin users.

List of API capabilities include:

- SW agent management (`sensor_management`): able to configure and monitor status of SW agents
- HW agent management (`hw_sensor_management`): able to configure and monitor status of HW agents (available only to Site Admin users)
- Tetration software download (`software_download`): able to download software packages for Tetration agents/virtual appliances
- Flow and inventory search (`flow_inventory_query`) : able to query flows and inventory items in Tetration cluster
- Users, roles and scope management (`user_role_scope_management`): able to read/add/modify/remove users, roles and scopes
- User data upload (`user_data_upload`): allow user to upload data for annotating flows and inventory items or upload good/bad file hashes
- Applications and policy management (`app_policy_management`): able to manage applications and enforce policies

- External system integration: able to allow integration with external systems like vCenter, kubernetes etc
- Tetration appliance management: able to manage Tetration appliance (available only to Site Admin users)

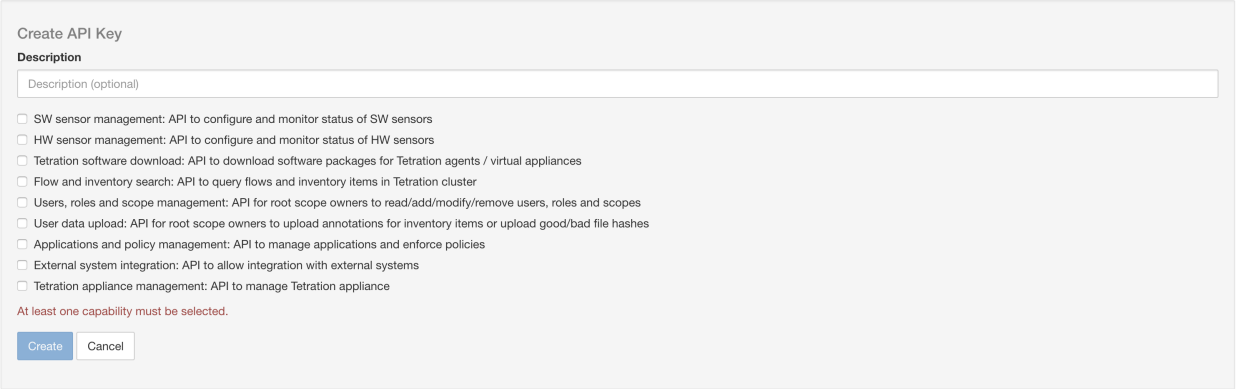


Fig. 17.1.1.3: API capabilities

Copy and paste the key and secret and save it in a safe location. Alternatively, download the API Credentials file.

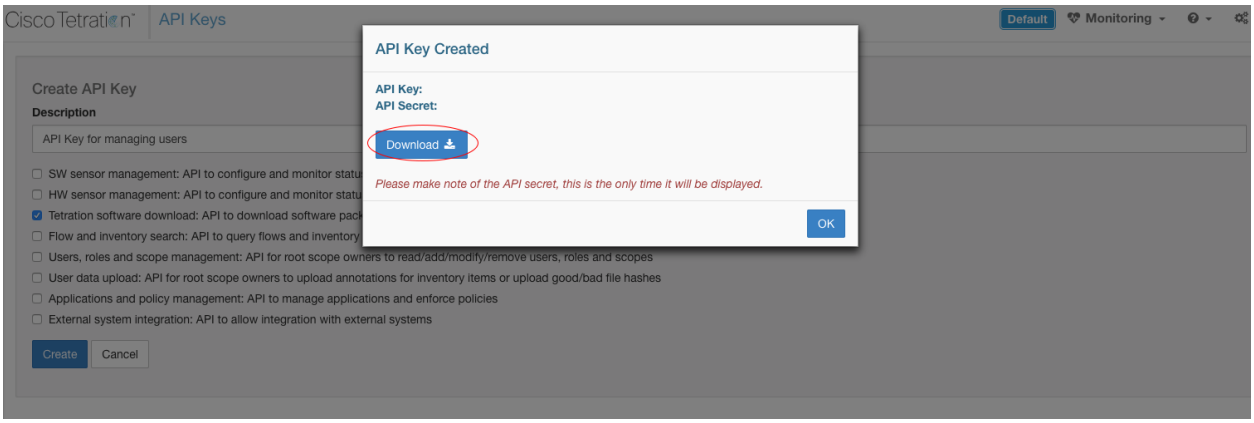


Fig. 17.1.1.4: API Credentials

Note: If External Auth with LDAP and LDAP Authorization are enabled, access to OpenAPI via API Keys will cease to work seamlessly because Tetration Roles derived from LDAP MemberOf groups are reassessed once the user session terminates. Hence to ensure uninterrupted OpenAPI access, we recommend that any user with API Keys have 'Use Local Authentication' option enabled in the Edit User Details flow for the user.

17.2 Applications and Security Policies

The following pages describe the OpenAPI endpoints to manage *Segmentation*

17.2.1 Applications

Application workspaces are the containers for defining, analyzing and enforcing policies for a particular application. For more information about how they work see the *Navigating to Applications* documentation. This set of APIs requires the `app_policy_management` capability associated with the API key.

17.2.1.1 Application Object

The application JSON object is returned as a single object or an array of objects depending on the API endpoint. The object's attributes are described below:

Attribute	Type	Description
id	string	A unique identifier for the application.
name	string	User specified name of the application.
description	string	User specified description of the application.
app_scope_id	string	ID of the scope assigned to the application.
author	string	First and last name of the user who created the application.
primary	boolean	Indicates if the application is primary for its scope.
alternate_query_mode	boolean	Indicates if 'dynamic mode' is used for the application. In the dynamic mode, an ADM run creates one or more candidate queries for each cluster. Default value is true.
created_at	integer	Unix timestamp of when the application was created.
latest_adm_version	integer	The latest adm (v*) version of the application.
enforcement_enabled	boolean	Indicates if enforcement is enabled on the application
enforced_version	integer	The enforced p* version of the application.

17.2.1.2 List applications

This endpoint will return an array of applications that are visible to the users.

```
GET /openapi/v1/applications
```

Parameters: None

Response object: Returns an array of application objects.

Sample python code

```
restclient.get('/applications')
```

17.2.1.3 Retrieve a single application

This endpoint will return the requested application as a single JSON object.

```
GET /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Response object: Returns the application object for the specified ID.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s' % application_id)
```

17.2.1.4 Create an application

This endpoint creates an application. It is possible to define policies by posting a JSON body containing the cluster and policy definitions.

Note: If a primary application exists for the same scope and new policies are provided, the policies will be added as a new version to the existing application.

```
POST /openapi/v1/applications
```

Parameters: The JSON query body contains the following keys

Name	Type	Description
app_scope_id	string	The scope ID to assign to the application.
name	string	(optional) A name for the application.
description	string	(optional) A description for the application.
alternate_query_mode	boolean	(optional) Indicates if 'dynamic mode' is used for the application. In the dynamic mode, an ADM run creates one or more candidate queries for each cluster. Default value is true.
strict_validation	boolean	(optional) Will return an error if there are unknown keys/attributes in the uploaded data. Useful for catching misspelled keys. Default value is false.
primary	string	(optional) Set to 'true' to indication this application should be primary for the given scope. **Default is true**

Extra optional parameters may be included describing policies to be created within the application.

Note: The scheme corresponds to that returned during export from the UI and the **Details** endpoint.

Name	Type	Description
clusters	array of clusters	Groups of nodes to be used to define policies.
inventory_filters	array of inventory filters	Filters on datacenter assets.
absolute_policies	array of policies	Ordered policies to be created with the absolute rank.
default_policies	array of policies	Ordered policies to be created with the default rank.
catch_all_action	string	"ALLOW" or "DENY"

Cluster object attributes:

Name	Type	Description
id	string	Unique identifier to be used with policies.
name	string	Displayed name of the cluster.
description	string	Description of the cluster.
nodes	array of nodes	Nodes or endpoints that are part of the cluster.
consistent_uuid	string	Must be unique to a given application. After an ADM run, the similar/same clusters in the next version will maintain the consistent_uuid.

Node object attributes:

Name	Type	Description
ip	string	IP or subnet of the node. eg 10.0.0.0/8 or 1.2.3.4
name	string	Displayed name of the node.

Inventory Filter object attributes:

Name	Type	Description
id	string	Unique identifier to be used with policies.
name	string	Displayed name of the cluster.
query	object	JSON object representation of an inventory filter query.

Policy object attributes:

Name	Type	Description
consumer_filter_id	string	ID of a cluster, user inventory filter or app scope.
provider_filter_id	string	ID of a cluster, user inventory filter or app scope.
action	string	"ALLOW" or "DENY"
l4_params	array of l4params	List of allowed ports and protocols.

L4Params object attributes:

Name	Type	Description
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000].
approved	boolean	(optional) Indicates if the policy is approved. Default is False.

Response object: Returns the newly created application object.

Sample python code

```
name = 'test'
scope_id = '5ce480cc497d4f1b4b9a9e8d'
filter_id = '5ce480cd497d4f1b4b9a9ea4'
application = {
    'app_scope_id': scope_id,
    'name': name,
    'absolute_policies': [
        {
            # consumer/provider filter IDs can be ID of an ADM cluster,
            # user inventory filter or app scope.
            'provider_filter_id': filter_id,
            'consumer_filter_id': filter_id,
            'action': 'ALLOW',
            # ALLOW policy for TCP on port 80.
            'l4_params': [
                {
                    'proto': 6, # TCP
                    'port': [80, 80],  # port range
                }
            ],
        }
    ],
    'catch_all_action': 'ALLOW'
}
restclient.post('/applications', json_body=json.dumps(application))
```

17.2.1.5 Delete an application

Removes an application.

```
DELETE /openapi/v1/applications/{application_id}
```

Enforcement must be disabled on the application before it can be deleted.

If the application, or its Clusters, are used on by other Applications (via a Provided Service relationship) this endpoint will return `422 Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of dependent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Response object: None

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.delete('/applications/%s' % application_id)
```

17.2.1.6 Update an application

This end point updates an existing application.

```
PUT /openapi/v1/applications/{application_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

The JSON query body contains the following keys

Name	Type	Description
name	string	(optional) The updated name for the application.
description	string	(optional) The updated description for the application.
primary	string	(optional) Set to 'true' to make the application a primary one. Set to 'false' to make the application a secondary one.

Response object: The updated application object for the specified ID.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'name': 'Updated Name',
      'description': 'Updated Description',
      'primary': 'true'
}
resp = restclient.put('/applications/%s' % application_id,
                      json_body=json.dumps(req_payload))
```

17.2.1.7 Retrieve application Details

This endpoint returns a full export JSON file for the application. This will include policy and cluster definitions.

```
GET /openapi/v1/applications/{application_id}/details
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.
version	string	(optional) A version in the form of 'v10' or 'p10', defaults to 'latest'.

Response object: Returns the clusters and policies for the given application version.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
# For v* version v10 and for p* version p10
version = 'v10'
resp = restclient.get('/applications/%s/details?version=%s' % (application_id,␣
↪version))
```

17.2.1.8 List application Versions

This endpoint will return a list of all the versions for a given applications.

```
GET /openapi/v1/applications/{application_id}/versions
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.
created_before	integer	(optional) For pagination, set to 'created_at' of the last version from previous response.
limit	integer	(optional) Max results to return, default is 50.

Response object: An array of objects with the following attributes:

Attribute	Type	Description
version	string	A version in the form of 'v10' or 'p10'.
created_at	integer	Unix timestamp of when the application was created.
description	string	User provided description.
name	string	Displayed name.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
created_before = 1612325705
limit = 10
resp = restclient.get('/applications/%s/versions?created_before=%s&limit=%s' %
                      (application_id, created_before, limit))
```

17.2.1.9 Delete application Version

This endpoint will remove the given version including clusters and policies. Enforced or Analyzed versions can not be deleted. If members are referenced by another application, through an external policy, the response will return error with a list of the references.

```
DELETE /openapi/v1/applications/{application_id}/versions/{version}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.
version	string	A version in the form of 'v10' or 'p10'.

Response object: None

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
version = 'v10'
resp = restclient.delete('/applications/%s/versions/%s' %
                         (application_id, version))
```

17.2.1.10 Analyze latest policies

Enable analysis on the latest set of policies in the application.

```
POST /openapi/v1/applications/{application_id}/enable_analysis
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Parameters: The optional JSON query body contains the following keys

Name	Type	Description
action_note	string	(optional) Reason for the publish policies action.
name	string	(optional) Name for the published policy version.
description	string	(optional) description for the published policy version.

Response object: Returns an object with the following attributes:

Attribute	Type	Description
data_set	object	JSON object representation of the data set.
analyzed_policy_version	integer	The analyzed p* version of the application.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'action_note': 'Policy analysis',
  'name': 'Test run 1',
  'description': 'New workloads added.'
}
resp = restclient.post('/applications/%s/enable_analysis' % application_id,
                       json_body=json.dumps(req_payload))
```

17.2.1.11 Disable policy analysis on a single application

Disable policy analysis on the application.

```
POST /openapi/v1/applications/{application_id}/disable_analysis
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Response object: Returns an object with the following attributes:

Attribute	Type	Description
data_set	object	JSON object representation of the data set.
analyzed_policy_version	integer	Last analyzed p* version of the application.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_analysis' % application_id)
```

17.2.1.12 Enforce a single application

Enable enforcement on the latest set of policies in the application.

```
POST /openapi/v1/applications/{application_id}/enable_enforce
```

Warning: New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.
version	string	(optional) The policy version to enforce.

If a `version` is not provided the latest policies of the application will be enforced. `versions` is preferred to be of the form 'p*', if just an integer is provided the corresponding 'p*' version will be enforced.

Response object: Returns an object with the following attributes:

Name	Type	Description
epoch	string	Unique identifier for the latest enforcement profile.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
  'version': 'p10'
}
resp = restclient.post('/applications/%s/enable_enforce' % application_id,
                       json_body=json.dumps(req_payload))
```

17.2.1.13 Disable enforcement for a single application

Disable enforcement on the application.

```
POST /openapi/v1/applications/{application_id}/disable_enforce
```

Warning: New host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Response object: Returns an object with the following attributes:

Name	Type	Description
epoch	string	Unique identifier for the latest enforcement profile.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.post('/applications/%s/disable_enforce' %
                       application_id)
```

17.2.1.14 Submit an ADM run

Submit an ADM run for the application.

```
POST /openapi/v1/applications/{application_id}/submit_run
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Parameters: The JSON query body contains the following keys

Name	Type	Description
start_time	string	Start time of the ADM run input time interval.
end_time	string	End time of the ADM run input time interval.
clustering_granularity	string	(optional) *Clustering Granularity* allows the user to have a control on the size of the generated clusters by ADM algorithms. Expected values: `VERY_FINE`, `FINE`, `MEDIUM`, `COARSE`, or `VERY_COARSE`
port_generalization	string	(optional) *Port Generalization* controls the level of statistical significance required when performing port generalization. Expected values: `DISABLED`, `CONSERVATIVE`, `MODERATE`, `AGGRESSIVE`, or `VERY_AGGRESSIVE`
policy_compression	string	(optional) *Policy Compression* when enabled, policies that are sufficiently frequent, i.e. they use the same provider port, among the generated clusters inside a workspace may be 'factored out' to the parent, that is, replaced with one or more policies applicable to the entire parent scope. Expected values: `DISABLED`, `CONSERVATIVE`, `MODERATE`, `AGGRESSIVE`, or `VERY_AGGRESSIVE`
auto_accept_policy_connectors	boolean	(optional) *Auto accept policy connectors* any outgoing policy requests created during the ADM run will be auto accepted.
enable_exclusion_filter	boolean	(optional) Enable exclusion filter option provides the flexibility to ignore all conversations matching any of the user defined exclusion filters (if any). Please see *Exclusion Filters* for more info.
enable_default_exclusion_filter	boolean	(optional) Enable default exclusion filter option provides the flexibility to ignore all conversations matching any of the default exclusion filters (if any). Please see *Default Exclusion Filters* for more info.
enable_service_discovery	boolean	(optional) When *Enable service discovery on agent* is set, ephemeral port-range information regarding services present on the agent node are reported. Policies are then generated based on the reported port-range information.
carry_over_policies	boolean	(optional) When *Carry over Approved Policies* is set, all the policies that are marked as approved by the user via UI or OpenAPI will be preserved.
skip_clustering	boolean	(optional) When *Skip clustering* is set, no new clusters are generated, and policies are generated from any existing approved clusters or inventory filters and otherwise involve the entire application scope.
deep_policy_generation	boolean	(optional) Deep policy generation is useful specially when one is interested in global policy generation. Please See *Deep policy generation* for more info.
use_default_config	boolean	(optional) When this option is set the ADM

Note: Unspecified optional parameter default values will be taken from the previous ADM run config if an ADM run was performed earlier in the workspace or else the default values will be taken from the Default ADM run config.

Response object: Returns an object with the following attributes:

Name	Type	Description
message	string	Message regarding success/failure in submission of ADM run.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
req_payload = {
      'start_time': '2020-09-17T10:00:00-0700',
      'end_time': '2020-09-17T11:00:00-0700',
      # Optional Parameters.
      'clustering_granularity': 'FINE',
      'port_generalization': 'AGGRESSIVE',
      'policy_compression': 'AGGRESSIVE',
      'auto_accept_policy_connectors': False,
      'enable_exclusion_filter': True,
      'enable_default_exclusion_filter': True,
      'enable_service_discovery': True,
      'carry_over_policies': True,
      'skip_clustering': False,
      'deep_policy_generation': True,
      'use_default_config': False
}
resp = restclient.post('/applications/%s/submit_run' % application_id,
                       json_body=json.dumps(req_payload))
```

17.2.1.15 Get ADM Run Status

Query ADM run status of the application

```
GET /openapi/v1/applications/{application_id}/adm_run_status
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

Response object: Returns an object with the following attributes:

Name	Type	Description
status	string	Status of the ADM run. Values: `PENDING`, `COMPLETE`, or `FAILED`

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
resp = restclient.get('/applications/%s/adm_run_status' % application_id)
```

17.2.2 Policies

This set of APIs can be used to manage add, edit or delete Policies. `version` parameter is required for create and update catch all actions. They require the `user_role_scope_management` capability associated with the API key.

17.2.2.1 Policy object

The policy object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the policy.
application_id	string	The id for the Application to which the policy belongs.
consumer_filter_id	string	ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	ID of a defined filter. Currently, any cluster, user defined filter or scope can be used as the provider of a policy.
version	string	Indicates the version of the Application to which the policy belongs.
rank	string	Policy rank, possible values: DEFAULT, ABSOLUTE or CATCHALL.
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic should be allowed or dropped for the given service port/protocol between the consumer and provider.
priority	integer	Used to sort policy.
l4_params	array of l4params	List of allowed ports and protocols.

L4Params object attributes:

Name	Type	Description
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000].
description	string	Short string about this proto and port.
approved	boolean	If the policy has been approved by the user.

17.2.2.2 Get Policies

This endpoint returns a list of policies for a particular application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/policies
```

Parameters: The request URL contains the following parameters

Name	Type	Description
version	string	Indicates the version of the Application for which to get the policies.
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the consumer filter id.

Returns an object of all policies for this particular application as shown below

```
{
  absolute_policies: [ ... ],
  default_policies: [ ... ],
  catch_all_action:
}
```

Sample python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/policies' % application_id, params={'version': '1'})
```

Get Default Policies

This endpoint returns a list of Default policies for a given application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/default_policies
```

Parameters:

Name	Type	Description
id	string	Unique identifier for the policy.
version	string	Indicates the version of the Application for which to get the policies.
limit	integer	Limits the number of policies per request.
offset	integer	(optional) Offset number received from previous response, should always be used along with `limit`.
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the provider filter id.

Returns a list of default policies for the provided version of this application. The response contains the requested number of polices and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

Sample python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/default_policies' % application_id, params={'version
↪': '1', 'limit': 3, 'offset': 3})
```

Sample response

```
{
  "results": [
  PolicyObject4,
  PolicyObject5,
  PolicyObject6
  ],
  "offset": 6
}
```

Get Absolute Policies

This endpoint returns a list of Absolute policies for a given application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/absolute_policies
```

Parameters:

Name	Type	Description
version	string	Indicates the version of the Application for which to get the policies.
limit	integer	Limits the number of policies per request.
offset	integer	(optional) Offset number received from previous response, should always be used along with `limit`.
consumer_filter_id	string	(optional) Filters the output by the consumer filter id.
provider_filter_id	string	(optional) Filters the output by the provider filter id.

Returns a list of absolute policies for the provided version of this application. The response contains the requested number of polices and an `offset`, to get the next set policies use this `offset` in the subsequent requests. Absence of an `offset` in the response indicates that all the policies are already retrieved.

Sample python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/absolute_policies' % application_id, params={'version
↪': '1', 'limit': 3})
```

Sample response

```
{
  "results": [
  PolicyObject1,
  PolicyObject2,
  PolicyObject3
  ],
  "offset": 3
}
```

Get Catch All Policies

This endpoint returns a Catch All policy for a given application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

Name	Type	Description
version	string	Indicates the version of the Application for which to get the policies.

Returns a single policy object representing the catch all policy for the given version of the application

Sample python code

```
application_id = '5f88c996755f023f3bafe163'
restclient.get('/applications/%s/catch_all' % application_id, params={'version': '1'})
```

17.2.2.3 Get Specific Policy

This endpoint returns an instance of a policy.

```
GET /openapi/v1/policies/{policy_id}
```

Returns the policy object associated with the specified ID.

Sample python code

```
policy_id = '5f88ca1e755f0222f85ce85c'
restclient.get('/policies/%s' % policy_id)
```

17.2.2.4 Create a Policy

This endpoint is used to create new policies.

```
POST /openapi/v1/applications/{application_id}/policies
```

Parameters:

Attribute	Type	Description
consumer_filter_id	string	ID of a defined filter.
provider_filter_id	string	ID of a defined filter.
version	string	Indicates the version of the Application for which to update the policies.
rank	string	values can be DEFAULT, ABSOLUTE or CATCHALL for ranking
policy_action	string	values can be ALLOW or DENY: means whether we should allow or drop traffic from consumer to provider on the given service port/protocol
priority	integer	Used to sort policy.

Sample python code

```
req_payload = {
  "version": "v1",
  "rank" : "DEFAULT",
  "policy_action" : "ALLOW",
  "priority" : 100,
  "consumer_filter_id" : "123456789",
  "provider_filter_id" : "987654321",
}
resp = restclient.post('/openapi/v1/applications/{application_id}/policies', json_
↪body=json.dumps(req_payload))
```

Create a Default Policy

This endpoint is used to create new default policies. This endpoint creates a default policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/default_policies
```

Create a Absolute Policy

This endpoint is used to create new absolute policies. This endpoint creates a absolute policy similar to the create a policy endpoint.

```
POST /openapi/v1/applications/{application_id}/absolute_policies
```

17.2.2.5 Update a Policy

This endpoint updates a policy.

```
PUT /openapi/v1/policies/{policy_id}
```

Parameters:

Attribute	Type	Description
consumer_filter_id	string	ID of a defined filter.
provider_filter_id	string	ID of a defined filter.
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic should be allowed or dropped for the given service port/protocol between the consumer and provider.
priority	integer	Used to sort policy priorities

Returns the modified policy object associated with specified ID.

Update a Catch All

This endpoint updates Catch All for a particular Application.

```
PUT /openapi/v1/applications/{application_id}/catch_all
```

Parameters:

Attribute	Type	Description
version	string	Indicates the version of the Application for which to update the policies.
policy_action	string	Possible values can be ALLOW or DENY. Indicates whether traffic not matching any of the policies in this application will allowed or drooped.

17.2.2.6 Adding Service Ports to a Policy

This endpoint is used to create service ports for a specific policy.

```
POST /openapi/v1/policies/{policy_id}/l4_params
```

Parameters:

Attribute	Type	Description
version	string	Indicates the version of the Application for which to get the policies.
start_port	integer	Start port of the range.
end_port	integer	End port of the range.
proto	integer	Protocol Integer value (NULL means all protocols).
description	string	(optional) Short string about this proto and port.

17.2.2.7 Updating Service Ports of a Policy

This endpoint updates the specified service port of a Policy.

```
PUT /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

Attribute	Type	Description
approved	bool	Marks the policy as approved.

17.2.2.8 Deleting Service Ports of a Policy

This endpoint deletes the specified service port of a Policy. (optional) see *Exclusion Filters* for more details.

```
DELETE /openapi/v1/policies/{policy_id}/l4_params/{l4_params_id}
```

Parameters:

Attribute	Type	Description
create_exclusion_filter	bool	(optional) If true, creates an exclusion filter matching the policy. Flows matching this filter will be excluded from future ADM runs. see *Exclusion Filters* for more details.

17.2.2.9 Deleting a Policy

This endpoint deletes the specified Policy. No exclusion filters are created.

```
DELETE /openapi/v1/policies/{policy_id}
```

17.2.2.10 Policy Quick Analysis

This endpoint can be used to find matching set of policies for any hypothetical flow against the analyzed/enforced polices in a root scope. For more details refer *Quick Analysis*

This API is only available to users with a minimum read access to root scope and requires `app_policy_management` capability associated with the API key.

```
POST /openapi/v1/policies/{rootScopeID}/quick_analysis
```

The query body consists of a JSON body with the following schema:

Name	Type	Description
consumer_ip	string	IP Address of the client / consumer.
provider_ip	string	IP Address of the server / provider.
provider_port	integer	(optional) Provider Port, only relevant for TCP or UDP flows.
protocol	string	Protocol of the flow, e.g. TCP.
analysis_type	string	Analysis type can be either **analyzed** or **enforced**. Analysis type "analyzed" makes the flow decision by matching the flow against all the analyzed polices in the root scope. Analysis type "enforced" makes the flow decision by matching the flow against all enforced policies in the root scope.
application_id	string	(optional) The ID of the primary application, always accompanied by the application 'v' version, if specified, makes the flow decision by using the policies from the specified version along with analyzed/enforced policies from other applications in the root scope. If this field is skipped, the flow decision is made by considering all the analyzed/enforced polices in the root scope.
version	integer	(optional) The 'v' version of the application mentioned above. This must be specified if the application_id is specified and must be skipped otherwise.

Sample Request

The body of the request should be a JSON formatted query.

An example of a query body where the flow decision is based on all analyzed polices looks like

```
req_payload = {
    "consumer_ip": "4.4.1.1",
    "provider_ip": "4.4.2.1",
    "provider_port": 9081,
    "protocol": "TCP",
    "analysis_type": "analyzed"
}
resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis', json_
↪body=json.dumps(req_payload))
```

An example of a query body where the flow decision is based on the policies from the applications 'v' version along with the analyzed polices from all other applications in the root scope looks like

```
req_payload = {
    "consumer_ip": "4.4.1.1",
    "provider_ip": "4.4.2.1",
    "provider_port": 9081,
    "protocol": "TCP",
    "analysis_type": "analyzed",
    "application_id": "5e7e5f56497d4f0bc26c7bb3",
    "version": 1
}
resp = restclient.post('/openapi/v1/policies/{rootScopeID}/quick_analysis', json_
↪body=json.dumps(req_payload))
```

Sample Response

The response is a JSON object in the body with the following properties.

Keys	Values
policy_decision	The decision of the hypothetical flow whether is allowed or denied.
outbound_policy	The policy on the consumer thats allowing/denying the outgoing traffic
inbound_policy	The policy on the provider thats allowing/denying the incoming traffic

```
{
    "policy_decision": "ALLOW",
    "outbound_policy": {
        "policy_rank": "DEFAULT",
        "start_port": 9082,
        "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
        "src_filter_id": "5e7e600e497d4f7341f4f459",
        "end_port": 9082,
        "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
        "dst_filter_id": "5e7d0efc497d4f44b6b09351",
        "action": "ALLOW",
        "protocol": "TCP",
        "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
    },
```

(continues on next page)

(continued from previous page)

```
    "inbound_policy": {
        "policy_rank": "DEFAULT",
        "start_port": 9082,
        "l4_detail_id": "5e7e600f497d4f7341f4f6d0",
        "src_filter_id": "5e7e600e497d4f7341f4f459",
        "end_port": 9082,
        "cluster_edge_id": "5e7e600f497d4f7341f4f6d1",
        "dst_filter_id": "5e7d0efc497d4f44b6b09351",
        "action": "ALLOW",
        "protocol": "TCP",
        "app_scope_id": "5e7e5f3a497d4f0bc26c7bb0"
    }
}
```

17.2.3 Clusters

This set of APIs can be used to add, edit or delete Clusters, which are members of Applications. They require the `user_role_scope_management` capability associated with the API key.

17.2.3.1 Cluster object

The cluster object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the cluster.
consistent_uuid	string	A consistent id that is maintained across ADM runs.
application_id	string	The id for the Application to which the cluster belongs.
version	string	The version of the Application to which the cluster belongs
name	string	The name of the cluster.
description	string	The description of the cluster.
approved	boolean	If the cluster has been 'approved' by the user.
query	JSON	Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes.
short_query	JSON	Filter (or match criteria) associated with the filter.
alternate_queries	array of queries	Alternate suggested queries generated by an ADM run in dynamic mode.

17.2.3.2 Get Clusters

This endpoint returns a list of clusters for a particular application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The id for the Application to which the cluster belongs.
version	string	Indications the version of the Application for which to get the clusters.

Response object: Returns an array of all clusters for this particular application and version.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
restclient.get('/applications/%s/clusters' % application_id)
```

17.2.3.3 Get Specific Cluster

This endpoint returns an instance of a cluster.

```
GET /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.

Response object: Returns the cluster object associated for the specified ID.

Sample python code

```
cluster_id = '5d02d021497d4f0949ba74e4'
restclient.get('/clusters/%s' % cluster_id)
```

17.2.3.4 Create a Cluster

This endpoint is used to create a new cluster.

```
POST /openapi/v1/applications/{application_id}/clusters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The id for the Application to which the cluster belongs.

The JSON query body contains the following keys

Attribute	Type	Description
name	string	The name of the cluster.
version	string	Indications the version of the Application the cluster will be added to.
description	string	(optional) The description of the cluster.
approved	boolean	(optional) An approved cluster will not be updated during an ADM run. Default false.
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the application, otherwise ignored.
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the application, otherwise ignored.
nodes	Array	List of ip addresses or endpoints. Will be used to create the query matching these ips unless a query is provided and the application is in Dynamic Mode.

Nodes object attributes:

Name	Type	Description
ip	string	IP address
name	string	(optional) The name of the node.
prefix_len	integer	(optional) Subnet mask.

Note: The nodes will be used to create a query unless a query is provided and the application is in Dynamic Mode.

Response object: Returns the newly created cluster object.

Sample python code

```
application_id = '5d02b493755f0237a3d6e078'
payload = {
  'name': 'test_cluster',
  'version': 'v2',
  'description': 'basic granularity',
  'approved': False,
  'query': {
      'type': 'eq',
      'field': 'host_name',
      'value': 'centos6001'
  }
}
restclient.post('/applications/%s/clusters' % application_id)
```

17.2.3.5 Update a Cluster

This endpoint updates a cluster.

```
PUT /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.

The JSON query body contains the following keys

Attribute	Type	Description
name	string	The name of the cluster.
description	string	(optional) The description of the cluster.
approved	boolean	An approved cluster will not be updated during an ADM run.
query	JSON	Filter (or match criteria) associated with the filter. Alternate Query Mode (also called Dynamic Mode) must be enabled on the application, otherwise ignored.

Response object: Returns the modified cluster object associated with specified ID.

Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
payload = {
  'name': 'new_test_cluster',
}
restclient.put('/clusters/%s' % cluster_id, json_body=json.dumps(payload))
```

17.2.3.6 Deleting a Cluster

This endpoint deletes the specified Cluster. If the cluster is used by any policies the cluster will not be deleted and a list of dependents will be returned.

```
DELETE /openapi/v1/clusters/{cluster_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
cluster_id	string	Unique identifier for the cluster.

Response object: None

Sample python code

```
cluster_id = '5d02d2a4497d4f5194f104ef'
restclient.delete('/clusters/%s' % cluster_id)
```

17.2.4 Conversations

Conversations are aggregated flows in the time range of the ADM run where the consumer port is removed. More detailed description about the conversations can be found in *Conversations*.

This API enables user to search the conversations generated during the ADM run on a given application. It requires `app_policy_management` capability associated with the API key to invoke this API.

17.2.4.1 Search conversations for an ADM run

This end point enables the user to search the conversations for an ADM run for a given application. The user can also specify a subset of supported dimensions and metrics which they may want to see as part of the downloaded conversations. Optionally, the user can query for subset of conversations using filters on supported dimensions and metrics.

```
POST /openapi/v1/conversations/{application_id}
```

The query consists of a JSON body with the following keys.

Name	Type	Description
version	integer	Version of the ADM run
filter	JSON	(optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters refer to *filters*.
dimensions	array	(optional) List of dimensions to be returned for the downloaded conversations. The list of supported dimension can be found *here*.
metrics	array	(optional) List of metrics to be returned for the downloaded conversations. The list of supported metrics can be found *here*.
limit	integer	(optional) Number of conversations to be returned in a single API response.
offset	string	(optional) Offset received from previous response – useful for pagination.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "version": 1,
    "filter": {
        "type": "and",
        "filters":[
            {
                "type": "eq",
                "field": "excluded",
                "value": False
            },
            {
                "type": "eq",
                "field": "protocol",
```

(continues on next page)

(continued from previous page)

```
                "value": "TCP"
            },
        ]
    },
    "dimensions": ["src_ip", "dst_ip", "port"],
    "metrics": ["byte_count", "packet_count"],
    "limit" : 2,
    "offset": <offset-object>
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

```
req_payload = {"version": 1,
               "limit": 10,
               "filter": {"type": "and",
                   "filters": [
                       {"type": "eq", "field": "excluded", "value": False},
                       {"type": "eq", "field": "protocol", "value": "TCP"}
                    ]
               }
 }

 resp = restclient.post('/conversations/{application_id}', json_body=json.dumps(req_
↪payload))
 print resp.status_code
 if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.2.4.2 Top N conversations for an ADM run

This end point enables the user to search the top conversations for an ADM run for a given application based on a metric and grouped by a dimension. The current supported metrics are *here* and the current supported group by dimensions are *here* The user can query for subset of conversations using filters on supported dimensions and metrics. One example may be to find the source IP address with the most byte traffic conversations so a query with the `src_ip` dimension with the `byte_count` metric.

```
POST /openapi/v1/conversations/{application_id}/topn
```

The query consists of a JSON body with the following keys.

Name	Type	Description
version	integer	Version of the ADM run
dimension	string	The dimension for the conversations to be grouped by for the top N query The list of supported dimension can be found *here*.
metric	string	The metric to be sorted by for the top N conversations. The list of supported metrics can be found *here*.
filter	JSON	(optional) Query filter. If filter is empty (i.e. {}), then query matches all the conversations. More specific conversations can be downloaded using filters on supported dimensions and metrics. For the syntax on filters refer to *filters*.
threshold	integer	(optional) Number of top N results to be returned in a single API response.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "version": 1,
    "dimension": "src_ip",
    "metric": "byte_count",
    "filter": {
        "type": "and",
        "filters":[
            {
                "type": "eq",
                "field": "excluded",
                "value": False
            },
            {
                "type": "eq",
                "field": "protocol",
                "value": "TCP"
            },
        ]
    },
    "threshold" : 10
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
results	List with one JSON object with a results key and a value of a list of results objects with keys matching the query dimension and metric.

```
[ {"result": [
  {
    "byte_count": 1795195565,
```

(continues on next page)

(continued from previous page)

```
    "src_ip": "192.168.1.6"
  },
  {
    "byte_count": 1781002379,
    "src_ip": "192.168.1.28"
  },
  ...
] } ]
```

```
req_payload = {"version": 1, "dimension": "src_ip", "metric": "byte_count",
  "filter": {"type": "and",
    "filters": [
      {"type": "eq", "field": "excluded", "value": False},
      {"type": "eq", "field": "protocol", "value": "TCP"},
      {"type": "eq", "field": "consumer_filter_id", "value": "16b12a5614c5af5b68afa7ce
↪"},
      {"type": "subnet", "field": "src_ip", "value": "192.168.1.0/24"}
    ]
  },
  "threshold" : 10
}

resp = restclient.post('/conversations/{application_id}/topn', json_body=json.
↪dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
    parsed_resp = json.loads(resp.content)
    print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.2.4.3 Supported dimensions

Name	Type	Description
src_ip	string	IP address of the consumer
dst_ip	string	IP address of the provider
protocol	string	Protocol used in the communication. Ex: "TCP", "UDP" etc.
port	integer	Port of the provider.
address_type	string	"IPv4" or "IPv6"
consumer_filter_id	string	Cluster ID of the cluster if the consumer IP belongs to a cluster, else the Scope ID the consumer IP belongs to.
provider_filter_id	string	Cluster ID of the cluster if the provider IP belongs to a cluster, else the Scope ID the provider IP belongs to.
excluded	boolean	Whether this conversation is excluded while generating policies.
confidence	double	The confidence level of consumer and provider classification. The value varies from 0.0 to 1.0 with 1.0 being more confident about classification.

17.2.4.4 Supported metrics

Name	Type	Description
byte_count	integer	Total number of bytes in the conversation
packet_count	integer	Total number of packets in the conversation

17.2.5 Exclusion Filters

This set of APIs can be used to add, edit or delete Exclusion Filters and require the `user_role_scope_management` capability associated with the API key.

Exclusion Filters exclude flows from the ADM clustering algorithm. See *Exclusion Filters* for more information.

17.2.5.1 Exclusion Filter object

The exclusion filter object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the cluster.
application_id	string	The id for the Application to which the exclusion filter belongs.
version	string	The version of the Application to which the exclusion filter belongs.
consumer_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the provider of a policy.
proto	integer	Protocol Integer value (NULL means all protocols).
port	array	Inclusive range of ports. eg [80, 80] or [5000, 6000]. NULL means all ports.
updated_at	integer	Unix timestamp of when the exclusion filter was updated.

17.2.5.2 Get Exclusion Filters

This endpoint returns a list of exclusion filters for a particular application. This API is available to API keys with `app_policy_management` capability.

```
GET /openapi/v1/applications/{application_id}/exclusion_filters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.
version	string	Indicates the version of the Application for which to get the exclusion filters.

Response object: Returns a list of exclusion filter objects for the specified application and version.

Sample python code

```
application_id = '<application-id>'
params = {'version': 'v10'}
restclient.get('/applications/%s/exclusion_filters' % application_id,
               params=params)
```

17.2.5.3 Get Specific Exclusion Filter

This endpoint returns an instance of an exclusion filters.

```
GET /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: Returns the exclusion filter object with the specified ID.

Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.get('/exclusion_filters/%s' % exclusion_filter_id)
```

17.2.5.4 Create an Exclusion Filter

This endpoint is used to create a new exclusion filter.

```
POST /openapi/v1/applications/{application_id}/exclusion_filters
```

Parameters: The request URL contains the following parameters

Name	Type	Description
application_id	string	The unique identifier for the application.

The JSON request body contains the following keys

Attribute	Type	Description
version	string	The version of the Application to which the exclusion filter belongs.
consumer_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the provider of a policy.
proto	integer	(optional) Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.
end_port	integer	(optional) End port of the range.

Missing optional parameters will be considered as wildcards (match any).

Response object: Returns the created exclusion filter object.

Sample python code

```
provider_filter_id = '<provider-filter-id>'
consumer_filter_id = '<consumer-filter-id>'
payload = {'version': 'v0',
           'consumer_filter_id': consumer_filter_id,
           'provider_filter_id': provider_filter_id,
           'proto': 6,
           'start_port': 800,
           'end_port': 1000}
application_id = '<application-id>'
restclient.post('/applications/%s/exclusion_filters' % application_id,
                json_body=json.dumps(payload))
```

17.2.5.5 Update an Exclusion Filter

This endpoint updates an exclusion filter.

```
PUT /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

The JSON request body contains the following keys

Attribute	Type	Description
consumer_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the consumer of a policy.
provider_filter_id	string	(optional) ID of a defined filter. Currently, any cluster belonging to the application, user defined filter or scope can be used as the provider of a policy.
proto	integer	Protocol Integer value (NULL means all protocols).
start_port	integer	(optional) Start port of the range.
end_port	integer	(optional) End port of the range.

Response object: Returns the modified exclusion filter object with the specified ID.

Sample python code

```
payload = {'proto': 17}
exclusion_filter_id = '<exclusion-filter-id>'
restclient.post('/exclusion_filters/%s' % exclusion_filter_id,
                json_body=json.dumps(payload))
```

17.2.5.6 Deleting an Exclusion Filter

This endpoint deletes the specified exclusion filter.

```
DELETE /openapi/v1/exclusion_filters/{exclusion_filter_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
exclusion_filter_id	string	The unique identifier for the exclusion filter.

Response object: None

Sample python code

```
exclusion_filter_id = '<exclusion-filter-id>'
restclient.delete('/exclusion_filters/%s' % exclusion_filter_id)
```

17.2.6 Live Analysis

Live analysis or Policy Analysis is an important aspect of generating security policies for applications. It allows users to evaluate the impact of a set of policies – where generated by ADM or manually added by users – before actually enforcing those policies on the workloads. Live analysis allows users to run what-if analysis on live traffic without disrupting any application traffic.

The set of APIs available in this section allow downloading flows and the effect of current set of published policies for an application on those flows. It requires `app_policy_management` capability associated with the API key to invoke these set of APIs.

Flows available via Live Analysis have some attributes (dimensions and metrics) and the download API allows user to filter flows by different criteria on dimensions.

17.2.6.1 Flow dimensions available in Live Analysis

This endpoint is useful to know the columns on which search criteria (or *filters*) can be specified for downloading flows available via Live Analysis. Most common use case would be to download *permitted*, *escaped* or *rejected* flows – this can be achieved by passing a search criteria on `category` dimension to the download API.

```
GET /openapi/v1/live_analysis/dimensions
```

17.2.6.2 Flow metrics available in Live Analysis

This endpoint returns the list of metrics (e.g. byte count, packet count) associated with live analysis. One use case for this endpoint would be to project a subset of metrics in the download API, i.e. instead of downloading all the metrics, users can specify a small subset of metrics they are interested in.

```
GET /openapi/v1/live_analysis/metrics
```

17.2.6.3 Download flows available via Live Analysis

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of live analysis dimensions (returned by the live analysis dimensions API above) as well as the live analysis metrics (returned by the live analysis metrics API above). Optionally, user can also specify a small subset of dimensions or metrics if they are not interested in the full set of available dimensions and metrics – this projection of a smaller subset of dimensions or metrics also have the side effect of making API calls fast.

```
POST /openapi/v1/live_analysis/{application_id}
```

The query body consists of a JSON body with the following keys.

Name	Type	Description
t0	integer or string	Start of time interval (epoch or ISO 8601)
t1	integer or string	End of time interval (epoch or ISO 8601)
filter	JSON	Query filter. If filter is empty (i.e. {}), then query matches all flows. Refer to section on *Filters* in Flow Search regarding syntax of filters.
dimensions	array	(optional) List of flow dimensions to be returned for the downloaded flows available through Live Analysis. If unspecified, all available dimensions are returned.
metrics	array	(optional) List of flow metrics to be returned for the downloaded flows available through Live Analysis.
limit	integer	(optional) Number of flows to be returned in a single API response.
offset	string	(optional) Offset received from previous response – useful for pagination.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "eq",
                "field": "category",
                "value": "escaped"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "limit": 100,
    "offset": <offset-object>
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

Sample python code

```
req_payload = {"t0": "2016-11-07T09:00:00-0700",
               "t1": "2016-11-07T19:00:00-0700",
               "limit": 10,
               "filter": {"type": "and",
                   "filters": [
                       {"type": "eq", "field": "category", "value": "escaped"},
                       {"type": "regex", "field": "src_hostname", "value": "web*"}
                    ]
               }
 }

 resp = restclient.post('/live_analysis/{application_id}', json_body=json.dumps(req_
↪payload))
 print resp.status_code
 if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.3 Scopes

This set of APIs can be used to manage Scopes (or AppScopes) in Tetration cluster deployment. They require the `user_role_scope_management` capability associated with the API key. The API to get the list of scopes is also available to API keys with `app_policy_management` or `sensor_management` capability.

17.3.1 Scope object

The scope object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the scope.
short_name	string	User specified name of the scope.
name	string	Fully qualified name of the scope. This is a fully qualified name, i.e. it has name of parent scopes (if applicable) all the way to the root scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.
query	JSON	Filter (or match criteria) associated with the scope in conjunction with the filters of the parent scopes (all the way to the root scope).
vrf_id	integer	ID of the VRF to which scope belongs to.
parent_app_scope_id	string	ID of the parent scope.
child_app_scope_ids	array	An array of scope children's ids.
policy_priority		Used to sort application priorities. See *Semantics and Viewing*.
dirty	bool	Indicates a child or parent query has been updated and that the changes need to be committed.
dirty_short_query	JSON	Non-null if the query for this scope has been updated but not yet committed.

17.3.2 Get scopes

This endpoint returns a list of scopes known to Tetration appliance. This API is available to API keys with either `app_policy_management` or `user_role_scope_management` capability.

```
GET /openapi/v1/app_scopes
```

Parameters: None

Returns a list of scope objects.

17.3.3 Create a scope

This endpoint is used to create new scopes.

```
POST /openapi/v1/app_scopes
```

Parameters:

Name	Type	Description
short_name	string	User specified name of the scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.
parent_app_scope_id	string	ID of the parent scope.
policy_priority	integer	Default is 'last'. Used to sort application priorities. See Policy Ordering under *Policies*.

Sample python code

```
req_payload = {
    "short_name": "App Scope Name",
    "short_query": {
        "type":"eq",
        "field":"ip",
        "value": <....>
    },
    "parent_app_scope_id": <parent_app_scope_id>
}
resp = restclient.post('/app_scopes', json_body=json.dumps(req_payload))
```

To create a scope based on subnet, use the following short_query:

```
"short_query":
  {
    "type":"subnet",
    "field":"ip",
    "value": "1.0.0.0/8"
  },
```

17.3.4 Get specific scope

This endpoint returns an instance of a scope.

```
GET /openapi/v1/app_scopes/{app_scope_id}
```

Returns the scope object associated with the specified ID.

17.3.5 Update a scope

This endpoint updates a scope. Changes to the `name` and `description` are applied immediately. Changes to the `short_query` mark the scope as 'dirty' and set the `dirty_short_query` attribute. Once all scope query changes, under a given root scope, are made, one needs to ping the *Commit Scope Query Changes* endpoint to commit all the required updates.

```
PUT /openapi/v1/app_scopes/{app_scope_id}
```

Parameters:

Name	Type	Description
short_name	string	User specified name of the scope.
description	string	User specified description of the scope.
short_query	JSON	Filter (or match criteria) associated with the scope.

Returns the modified scope object associated with specified ID.

17.3.6 Delete a specific scope

This endpoint deletes the specified scope.

```
DELETE /openapi/v1/app_scopes/{app_scope_id}
```

If the Scope is assigned to an Application, Policy, User Inventory Filter, etc. this endpoint will return `422 Unprocessable Entity`. The returned Error object will contain a `details` attribute with the count of dependent objects along with the ids of the first 10 of each type. This information can be used to locate and remove the blocking dependencies.

17.3.7 Get scopes in policy priority order

This endpoint lists the scopes in the order that their corresponding primary Applications will be enforced.

```
GET /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Returns an array of scope objects.

17.3.8 Update the policy order

This endpoint will update the order at which policies are applied. See *Semantics and Viewing* for more details.

> **Warning:** This endpoint changes the order at which policies are applied. As a result new host firewall rules will be inserted and any existing rules will be deleted on the relevant hosts.

```
POST /openapi/v1/app_scopes/{root_app_scope_id}/policy_order
```

Parameters:

Name	Type	Description
root_app_scope_id	string	Root scope or which the order is being changed.
ids	array	array of scope id strings in the order they should be enforced.

The `ids` array parameter must include all members of the root scope, including the root.

17.3.9 Commit scope query changes

This endpoint triggers an asynchronous background job to update all 'dirty' children under a given root scope. This job updates scopes and applications, see *Scopes* for more details.

```
POST /openapi/v1/app_scopes/commit_dirty
```

Parameters:

Name	Type	Description
root_app_scope_id	string	ID for a root scope for which all children will be updated.
sync	boolean	(optional) Indicate if the request should be synchronous.

Returns 202 to indicate the job has been enqueued. To check if the job has completed, poll the root scope's 'dirty' attribute to see if it has been set to false.

Users may pass the `sync` parameter to have the job run immediately. The request will return when done with a 200 status code. This request may take some time if many updates need to be applied.

17.4 Roles

This set of APIs can be used to manage user roles. They require the `user_role_scope_management` capability associated with the API key.

Note: These APIs are only available to site admins and owners of root scopes.

17.4.1 Role object

The role object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the role.
app_scope_id	string	Scope to which the scope is defined, maybe empty for "Service Provider Roles".
name	string	User specified name for the role.
description	string	User specified description for the role.

17.4.2 Get roles

This endpoint returns a list of roles accessible to the user. Roles can be filtered to a given root scope. If no scope if provided, all roles, for all scopes the user has access to, are returned. Service provider roles will only be returned if the user is a site admin.

```
GET /openapi/v1/roles
```

Parameters:

Name	Type	Description
app_scope_id	string	(optional) ID of a root scope to return roles only assigned to that scope.

Response object: Returns a list of user role objects.

Sample python code

```
resp = restclient.get('/roles')
```

17.4.3 Create a role

This endpoint is used to create a new role.

```
POST /openapi/v1/roles
```

Parameters:

Name	Type	Description
name	string	User specified name for the role.
description	string	User specified description for the role.
app_scope_id	string	(optional) The scope ID under which the role is created If no scope ID mentioned the role is considered as service provider role.

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may create them.

Response object: Returns the newly created role object.

Sample python code

```
app_scope_id = '<app-scope-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
      'app_scope_id': app_scope_id
}
restclient.post('/roles', json_body=json.dumps(req_payload))
```

17.4.4 Get specific role

This endpoint returns a specific role object.

```
GET /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
role_id	string	Uniquely identifies the role.

Response object: Returns a role object associated with the specified ID.

Sample python code

```
role_id = '<role-id>'
restclient.get('/roles/%s' % role_id)
```

17.4.5 Update a role

This endpoint is used to update an existing role.

```
PUT /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
role_id	string	Uniquely identifies the role.

The JSON request body contains the following parameters

Name	Type	Description
name	string	User specified name for the role.
description	string	User specified description for the role.

The requesting user must have access to the provided scope. A role without a scope is called a 'Service Provider Role' and only site admin may update them.

Response object: The updated role object with the specified ID.

Sample python code

```
role_id = '<role-id>'
req_payload = {
      'name': 'Role Name',
      'description': 'Role Description',
}
restclient.put('/roles/%s' % role_id, json_body=json.dumps(req_payload))
```

17.4.6 Give a role access to scope

This endpoint gives a role the specified access level to a scope.

```
POST /openapi/v1/roles/{role_id}/capabilities
```

Capabilities can only be added to the roles that the user has access to. If the roles is assigned to a scope, capabilities must correspond to that scope or its children. Service provider roles (those not assigned to a scope) can add capabilities for any scope.

Parameters: The request URL contains the following parameters

Name	Type	Descrption
role_id	string	Uniquely identifies the role.

The JSON request body contains the following parameters

Name	Type	Description
app_scope_id	string	ID of the scope to which access is provided.
ability	string	Possible values are `SCOPE_READ`, `SCOPE_WRITE`, `EXECUTE`, `ENFORCE`, `SCOPE_OWNER`, `DEVELOPER`

For more description of abilities, refer to *Roles*.

Response object:

Name	Type	Description
app_scope_id	string	ID of the scope to which access is provided.
role_id	string	ID of the role.
ability	string	Possible values are `SCOPE_READ`, `SCOPE_WRITE`, `EXECUTE`, `ENFORCE`, `SCOPE_OWNER`, `DEVELOPER`
inherited	boolean	

Sample python code

```
role_id = '<role-id>'
req_payload = {
  'app_scope_id': '<app-scope-id>',
  'ability': 'SCOPE_READ'
}
restclient.post('/roles/%s/capabilities' % role_id,
                json_body=json.dumps(req_payload))
```

17.4.7 Delete specific role

This endpoint deletes the specified role.

```
DELETE /openapi/v1/roles/{role_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
role_id	string	Uniquely identifies the role.

Response object: None.

Sample python code

```
role_id = '<role-id>'
restclient.delete('/roles/%s' % role_id)
```

17.5 Users

This set of APIs manages users. They require the `user_role_scope_management` capability associated with the API key.

Note: These APIs are only available to site admins and owners of root scopes.

17.5.1 User object

The user object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the user role.
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	The scope to which the user is assigned. Maybe empty if the user is a "Service Provider User".
role_ids	list	List of IDs of roles assigned to the user account.
bypass_external_auth	boolean	True for local users and false for external auth users (ldap or sso).
disabled_at	integer	Unix timestamp of when the user has been disabled. Zero or null, otherwise.

17.5.2 Get users

This endpoint returns a list of user objects known to the Tetration appliance.

```
GET /openapi/v1/users
```

Parameters: The request URL contains the following parameters

Name	Type	Description
include_disabled	boolean	(optional) To include disabled users, defaults to false.
app_scope_id	string	(optional) Return only users assigned to the provided scope.

Response object: Returns a list of user objects. Only site admins can see 'Service provider users', i.e. those not assigned to a scope.

Sample python code

```
resp = restclient.get('/users')
```

17.5.3 Create a new user account

This endpoint is used to create a new user account.

```
POST /openapi/v1/users
```

Parameters: The JSON request body contains the following parameters

Name	Type	Description
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	(optional) Root scope to which user belongs.
role_ids	list	(optional) The list of roles that should be assigned to the user.

The app_scope_id is the ID of the root scope to which the user is to be assigned. If the `app_scope_id` is not present then the user is a 'Service Provider user.' Only site admins can create service provider users. The `role_ids` are the ids of the roles that were created under the specified app scope.

Response object: Returns the newly created user object.

Sample python code

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
    "email": "foo@bar.com"
    "app_scope_id": "root_appscope_id",
    "role_ids": ["roleid1", "roleid2"]
}
resp = restclient.post('/users', json_body=json.dumps(req_payload))
```

17.5.4 Get specific user

This endpoint returns specific user object.

```
GET /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object.

Response object: Returns a user object associated with specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.get('/users/%s' % user_id)
```

17.5.5 Update a user

This endpoint updates an existing user.

```
PUT /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being updated.

The JSON request body contains the following parameters

Name	Type	Description
email	string	Email associated with user account.
first_name	string	First name.
last_name	string	Last name.
app_scope_id	string	Root App Scope ID (only allowed for site admins)

Response object: Returns the newly updated user object.

Sample python code

```
req_payload = {
    "first_name": "fname",
    "last_name": "lname",
    "email": "foo@bar.com"
    "app_scope_id": "root_appscope_id",
}
restclient.put('/users', json_body=json.dumps(req_payload))
```

17.5.6 Enable/reactivate a deactivated user

This endpoint is used to re-enable a deactivated user.

```
POST /openapi/v1/users/{user_id}/enable
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being enabled.

Response object: Returns the reactivated user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.post('/users/%s/enable' % user_id)
```

17.5.7 Add role to the user account

This endpoint is used to add a role to a user account.

```
PUT /openapi/v1/users/{user_id}/add_role
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being modified.

The JSON request body contains the following parameters

Name	Type	Description
role_id	string	ID of the role object to be added.

Response object: Returns the modified user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.put('/users/%s/add_role' % user_id,
                      json_body=json.dumps(req_payload))
```

17.5.8 Remove role from the user account

This endpoint is used to remove a role from a user account.

```
DELETE /openapi/v1/users/{user_id}/remove_role
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being deleted.

The JSON request body contains the following parameters

Name	Type	Description
role_id	string	ID of the role object to be removed.

Response object: Returns the modified user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
req_payload = {
    "role_id": "5ce480d4497d4f1c155d0cef",
}
resp = restclient.delete('/users/%s/remove_role' % user_id,
                         json_body=json.dumps(req_payload))
```

17.5.9 Delete specific user

This endpoint deletes the specified user account.

```
DELETE /openapi/v1/users/{user_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
user_id	string	ID of the user object being deleted.

Response object: Returns the deleted user object associated with the specified ID.

Sample python code

```
user_id = '5ce480db497d4f1ca1fc2b2b'
resp = restclient.delete('/users/%s' % user_id)
```

17.6 Inventory filters

Inventory filters encode the match criteria for inventory search queries. This set of APIs provide functionality similar to what is described in Inventory *Filters*. They require either `sensor_management` or `app_policy_management` capability associated with the API key.

17.6.1 Inventory Filter Object

The inventory filter JSON object is returned as a single object or an array of objects depending on the API endpoint. The object's attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the inventory filter.
name	string	User specified name of the inventory filter.
app_scope_id	string	ID of the scope associated with the filter.
short_query	JSON	Filter (or match criteria) associated with the filter.
primary	boolean	When 'true' the filter is restricted to the ownership scope.
public	boolean	When 'true' the filter provides a service for its scope. Must also be primary/scope restricted.
query	JSON	Filter (or match criteria) associated with the filter in conjunction with the filters of the parent scopes. These conjunctions take effect if 'restricted to ownership scope' checkbox is checked. If 'primary' field is false then query is same as short_query.

17.6.2 Get inventory filters

This endpoint returns a list of inventory filters visible to the user.

```
GET /openapi/v1/filters/inventories
```

Parameters: None

17.6.3 Create an inventory filter

This endpoint is used to create an inventory filter.

```
POST /openapi/v1/filters/inventories
```

Parameters:

Name	Type	Description
name	string	User specified name of the application scope.
query	JSON	Filter (or match criteria) associated with the scope.
app_scope_id	string	ID of the scope associated with the filter.
primary	boolean	When 'true' the filter is restricted to the ownership scope.
public	boolean	When 'true' the filter provides a service for its scope. Must also be primary/scope restricted.

Sample python code

```
req_payload = {
  "app_scope_id": <app_scope_id>,
  "name": "sensor_config_inventory_filter",
  "query": {
    "type": "eq",
    "field": "ip",
    "value": <sensor_interface_ip>
  },
}
resp = restclient.post('/filters/inventories', json_body=json.dumps(req_payload))
```

17.6.4 Get specific inventory filter

This endpoint returns an instance of an inventory filter.

```
GET /openapi/v1/filters/inventories/{inventory_filter_id}
```

Returns an inventory filter object associated with specified ID.

17.6.5 Update specific inventory filter

This endpoint is used to update an inventory filter.

```
PUT /openapi/v1/filters/inventories/{inventory_filter_id}
```

Parameters:

Name	Type	Description
name	string	User specified name of the application scope.
query	JSON	Filter (or match criteria) associated with the scope.
app_scope_id	string	ID of the scope associated with the filter.
primary	boolean	When 'true' the filter is restricted to the ownership scope.
public	boolean	When 'true' the filter provides a service. May be used as part of policy generation. Must also be primary/scope restricted.

17.6.6 Delete specific application scope

This endpoint deletes the specified inventory filter.

```
DELETE /openapi/v1/filters/inventories/{inventory_filter_id}
```

17.7 Flow Search

The flow search feature provides similar functionality as described in *Flows*. These set of APIs require the `flow_inventory_query` capability associated with the API key.

17.7.1 Query for flow dimensions

This endpoint returns the list of flow columns on which search criteria (or *filters*) can be specified for flow search queries (below). For more description of columns, refer to *Columns and Filters*.

```
GET /openapi/v1/flowsearch/dimensions
```

Parameters: None

Response object:

Name	Type	Description
dimensions	List of strings	List of user uploaded and orchestrator dimensions.

Sample python code

```
restclient.get('/flowsearch/dimensions')
```

17.7.2 Query for flow metrics

This endpoint returns the list of metrics (e.g. byte count, packet count) associated with flow observations.

```
GET /openapi/v1/flowsearch/metrics
```

Parameters: None

Response object:

Name	Type	Description
metrics	List of strings	List of available metrics

Sample python code

```
restclient.get('/flowsearch/metrics')
```

17.7.3 Query for flows

This endpoint returns the list of flows matching the filter criteria. Each flow object in the result has attributes that are a union of flow dimensions (returned by the flow dimensions API above) as well as the flow metrics (returned by the flow metrics API above).

```
POST /openapi/v1/flowsearch
```

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API.

Parameters: The query body consists of a JSON body with the following keys.

Name	Type	Description
t0	integer or string	Flow search start time (epoch or ISO 8601)
t1	integer or string	Flow search end time (epoch or ISO 8601)
filter	JSON	Query filter. If filter is empty (i.e. {}), then query matches all flows.
scopeName	string	Full name of the scope to which query is restricted.
dimensions	array	(optional) List of dimension names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions.
metrics	array	(optional) List of metric names to be returned in the result of flowsearch API. This is an optional parameter. If unspecified, flowsearch results return all the available metrics. This option is useful to specify a subset of the available metrics when caller does not care about the rest of the metrics.
limit	integer	(optional) Number of response flows limit.
offset	string	(optional) Offset object received from previous response.
descending	boolean	(optional) If this parameter is false or left unspecified, results are in ascending order of timestamps. If parameter value is true, results are in descending order of timestamps.

The body of the request should be a JSON formatted query. An example of a query body is shown below.

```
{
    "t0": "2016-06-17T09:00:00-0700",
    "t1": "2016-06-17T17:00:00-0700",
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
```

(continues on next page)

(continued from previous page)

```
                "field": "dst_hostname",
                "value": "prod"
            },
            {
                "type": "in",
                "field": "dst_port",
                "values": ["80", "443"]
            }
        ]
    },
    "scopeName": "Default:Production:Web",
    "limit": 100,
    "offset": <offset-object>
}
```

17.7.3.1 Filters

The filter supports primitive filters and logical filters ("not", "and", "or") comprised of one or more primitive filters.

Format of primitive filter is as follows:

```
{"type" : "<OPERATOR>", "field": "<COLUMN_NAME>", "value": "<COLUMN_VALUE>"}
```

For primitive filters, operator can be a comparison operator like `eq`, `ne`, `lt`, `lte`, `gt` or `gte`. Operator could also be `in`, `regex`, `subnet`, `contains` or `range`.

Some examples of primitive filters might include:

```
{"type": "eq", "field": "src_address", "value": "7.7.7.7"}

{"type": "regex", "field": "src_hostname", "value": "prod.*"}

{"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}

# Note, 'in' clause uses 'values' key instead of 'value'
{"type": "in", "field": "src_port", "values": [80, 443]}
```

User can also specify complex filters using boolean operations like `not`, `and` or `or`. Following are some examples of these type of filters:

```
# "and" and "or" operators need to specify list of "filters"
{"type": "and",
    "filters": [
        {"type": "in", "field": "src_port", "values": [80, 443]},
        {"type": "regex", "field": "src_hostname", "value": "prod.*"}
    ]
}

# "not" operator needs to specify a "filter"
{"type": "not",
    "filter": {"type": "subnet", "field": "src_addr", "value": "1.1.11.0/24"}
}
```

More formally, schema of `filter` in the flow search request is as follows:

Keys	Values
type	Filter type
field	Filter field column for primitive filters
filter	Filter object (only used for `not` filter type)
filters	List of filter objects (used for `and` and `or` filter types)
value	Value for primitive filters
values	List of values for primitive filters with filter type `in` or `range`

17.7.3.2 Primitive Filter Types

eq, ne Searches flows for equality or inequality respectively in column specified by `"field"` with value specified by `"value"`. Supports the following fields: `src_hostname`, `dst_hostname`, `src_address`, `dst_address`, `src_port`, `dst_port`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`, `proto`. These operators also work on user labelled columns.

lt, lte, gt, gte Searches flows where values of column specified by `"field"` are less than, less than equal to, greater than or greater than equal to (as applicable) the value specified by `"value"`. Supports the following fields: `[src_port, dst_port]`.

range Searches flows for values of column specified by `"field"` between range start and range end specified by `"values"` list (this list must be of size 2 for *"range"* filter type – first value is the range start and second is the range end). Supports the following fields: `[src_port, dst_port]`.

in Searches flows for membership in column specified by `"field"` with membership list specified by `"values"`. Supports the following fields: `src_hostname`, `dst_hostname`, `src_address`, `dst_address`, `src_port`, `dst_port`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`, `proto`. This operator also works on user labelled columns.

regex, contains Searches flows for regex matches or containment matches respectively in column specified by `"field"` with regex specified by `"value"`. Supports the following fields: `src_hostname`, `dst_hostname`, `src_scope_name`, `dst_scope_name`, `vrf_name`, `src_enforcement_epg_name`, `dst_enforcement_epg_name`. These operators also work on user labelled columns. Filters with `regex` type must use Java style regex patterns as `"value"`.

subnet Searches flows for subnet membership specified by `"field"` as a string in CIDR notation. Supports the following fields: `["src_address", "dst_address"]`

17.7.3.3 Logical Filter Types

not Logical not filter of object specified by `"filter"`.

and Logical and filter of list of filter objects specified by `"filters"`.

or Logical or filter of list of filter objects specified by `"filters"`.

Response object:

Keys	Values
offset	Response offset to be passed for the next page of results
results	List of results

To generate the next page of results, take the object received by the response in `offset` and pass it as the value for the `offset` of the next query.

Sample python code

```
req_payload = {"t0": "2016-11-07T09:00:00-0700",
               "t1": "2016-11-07T19:00:00-0700",
               "scopeName": "Default:Prod:Web",
               "limit": 10,
               "filter": {"type": "and",
                   "filters": [
                       {"type": "subnet", "field": "src_address", "value": "1.1.11.0/
↪24"},
                       {"type": "regex", "field": "src_hostname", "value": "web*"}
                    ]
               }
 }

 resp = restclient.post('/flowsearch', json_body=json.dumps(req_payload))
 print resp.status_code
 if resp.status_code == 200:
     parsed_resp = json.loads(resp.content)
     print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.7.4 TopN query for flows

This endpoint returns a top n sorted list of values of specified dimension where rank in the list is determined by the aggregate of specified metric.

```
POST /openapi/v1/flowsearch/topn
```

Parameters:

The list of columns that can be specified in the filter criteria can be obtained by `/openapi/v1/flowsearch/dimensions` API. The body of the request should be a JSON formatted query. An example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in iso8601 format. TopN API only allows querying maximum time range of 1 day. The dimension on which the grouping has to be done should be specified through `dimension`. The metric by which top N results need to ranked should be specified in `metric` field in the JSON body. Users should specify a `threshold` with a minimum value of 1 which signifies the 'N' in 'TopN'. The maximum value of this `threshold` is 1000. Even if the user specifies more than 1000 the API returns only a maximum of 1000 results. In addition, user needs to specify a parameter called `scopeName` which is the full name of the application scope to which user wants to restrict the search. The `filter` is same as that of filter of Flow Search *Filters*. If the `filter` is not mentioned, then the topN is applied on all the flow entries.

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "dimension": "src_address",
  "metric": "fwd_pkts",
  "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},
↪#optional
  "threshold": 5,
  "scopeName": "Default"
}
```

The query body consists of a JSON body with the following keys.

Keys	Values
t0	Start time of the Flow (epoch or ISO 8601)
t1	End time of the Flow (epoch or ISO 8601)
filter	Query filter. If filter is empty (i.e. {}),or filter is absent (optional) then topN query is applied on all flow entries
scopeName	Full name of the scope to which query is restricted to
dimension	The dimension is a field on which we are grouping.
metric	The metric is the total count of values of the dimension.
threshold	Threshold is 'N' in the topN.

Response object:

Keys	Values
result	Array of the top N entries

Sample python code

```
req_payload = {
    "t0": "2017-06-07T08:20:00-07:00",
    "t1": "2017-06-07T14:20:00-07:00",
    "dimension": "src_address",
    "metric": "fwd_pkts",
    "filter": {"type": "ne", "field": "src_address", "value": "172.29.203.193"},
    "threshold": 5,
    "scopeName": "Default"
}
resp = rc.post('/flowsearch/topn',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
[
  { "result": [
    {"src_address": "172.31.239.163", "fwd_pkts": 23104},
    {"src_address": "172.31.239.162", "fwd_pkts": 22410},
    {"src_address": "172.31.239.166", "fwd_pkts": 16185},
    {"src_address": "172.31.239.168", "fwd_pkts": 15197},
    {"src_address": "172.31.239.169", "fwd_pkts": 15116}
    ]
  }
]
```

17.7.5 Flow Count

This endpoint returns the number of flow observations matching the specified criteria.

```
POST /openapi/v1/flowsearch/count
```

Parameters:

The body of the request should be a JSON formatted query. An example of a query body is shown below. Parameters `t0` and `t1` in the request body can be in epoch format or in iso8601 format. This API only allows querying maximum time range of 1 day. In addition, user needs to specify `scopeName` parameter which is the full name of the application scope to which user wants to restrict the search. If this parameter is not specified, flow observation count API request applies to all scopes to which user has read access to. The `filter` is same as that of filter of Flow Search *Filters*.

```
{
  "t0": "2016-06-17T09:00:00-0700",    # t0 can also be 1466179200
  "t1": "2016-06-17T17:00:00-0700",    # t1 can also be 1466208000
  "filter": {"type": "eq", "field": "src_address", "value": "172.29.203.193"},
  "scopeName": "Default"
}
```

The query body consists of a JSON body with the following keys.

Keys	Values
t0	Start time of the Flow (epoch or ISO 8601)
t1	End time of the Flow (epoch or ISO 8601)
filter	Query filter. If filter is empty (i.e. {}), then query matches all flows.
scopeName	Full name of the scope to which query is restricted to

Response object:

Keys	Values
count	The number of flow observations matching flow search criteria.

Sample python code

```
req_payload = {
      "t0": "2017-07-20T08:20:00-07:00",
      "t1": "2017-07-20T10:20:00-07:00",
      "scopeName": "Tetration",
      "filter": {
                  "type": "eq",
                  "field": "dst_port",
                  "value": "5642"
          }
}
resp = rc.post('/flowsearch/count',
               json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
{"count":508767}
```

17.8 Inventory

The inventory search APIs provide similar functionality as described in Inventory ../inventory/search. These set of APIs require the `flow_inventory_query` capability associated with the API key.

17.8.1 Query for inventory dimensions

This endpoint returns the list of inventory columns on which search criteria (or *filters*) can be specified for inventory search queries.

```
GET /openapi/v1/inventory/search/dimensions
```

17.8.2 Inventory search

This endpoint returns the list of inventory items matching the specified criteria.

```
POST /openapi/v1/inventory/search
```

The list of columns that can be specified in the filter criteria can be obtained with the `/openapi/v1/inventory/search/dimensions` API.

Parameters:

Name	Type	Description
filter	JSON	A filter query.
scopeName	string	(optional) Name of the scope by which to limit results.
limit	integer	(optional) Max number of results to return.
offset	integer	(optional) Offset from the previous request to get the next page.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
    "filter": {
        "type": "contains",
        "field": "hostname",
        "value": "collector"
    },
    "scopeName": "Default:Production:Web",  # optional
    "limit": 100,
    "offset": <offset-object>               # optional
}
```

To get the different types of filters supported refer to *Filters*.

The query body consists of a JSON body with the following keys.

Keys	Values
filter	Query filter. If filter is empty (i.e. {}), then query matches all inventory items.
scopeName	Full name of the scope to which query is restricted to (optional)
dimensions	List of dimension names to be returned in the result of inventory search API. This is an optional parameter. If unspecified, results return all the available dimensions. This option is useful to specify a subset of the available dimensions when caller does not care about the rest of the dimensions.
limit	Number of response items limit (optional)
offset	Offset object received from previous response (optional)

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
offset	integer	Response offset to be passed for the next page of results.
results	array of objects	List of results.

The response may contain an `offset` field for paginated responses. Users will need to specify the same offset in the subsequent request to get the next set of results.

Sample Python code

```
req_payload = {
  "scopeName": "Tetration", # optional
  "limit": 2,
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
  }
}

resp = restclient.post('/inventory/search', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.8.3 Inventory Statistics

This endpoint returns statistics for inventory items.

```
GET /openapi/v1/inventory/{id}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Path Parameter	Description
id	Inventory item id as `{ip}-{vrf_id}` such as `1.1.1.1-123`

Query Parameter	Description
t0	Start time for statistics in epoch time
t1	End time for statistics in epoch time
td	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as "minute", "hour", and "day".

Sample Python code

```
resp = restclient.get('/inventory/1.1.1.1-123/stats?t0=1483228800&t1=1485907200&td=day
↪')
```

17.8.4 Inventory count

This endpoint returns the count of inventory items matching the specified criteria.

```
POST /openapi/v1/inventory/count
```

The list of columns that can be specified in the filter criteria can be obtained with the `/openapi/v1/inventory/search/dimensions` API.

Parameters:

Name	Type	Description
filter	JSON	A filter query.
scopeName	string	(optional) Name of the scope by which to limit results.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
    "filter": {
        "type": "and",
        "filters": [
            {
                "type": "contains",
                "field": "hostname",
                "value": "prod"
            },
            {
                "type": "subnet",
                "field": "ip"
                "value": "6.6.6.0/24"
            }
        ]
    },
    "scopeName": "Default:Production:Web",  # optional
}
```

Response

The response is a JSON object in the body with the following properties.

Keys	Values
count	Number of inventory items matching the filter Criteria

Sample python code

```
req_payload = {
  "scopeName": "Tetration", # optional
  "filter": {"type": "and",
     "filters": [
        {"type": "eq", "field": "vrf_name", "value": "Tetration"},
        {"type": "subnet", "field": "ip", "value": "1.1.1.0/24"},
        {"type": "contains", "field": "hostname", "value": "collector"}
     ]
  }
}

resp = restclient.post('/inventory/count', json_body=json.dumps(req_payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.8.5 Inventory vulnerability

This endpoint returns CVEs corresponding to IP addresses associate with vulnerable workloads.

This API is only available to users with a minimum read access to root scope.

```
POST /openapi/v1/inventory/cves/{rootScopeID}
```

Parameters:

Name	Type	Description
ips	list of strings	List of IPs to fetch CVE information.

The body of the request must be a JSON formatted query. An example of a query body is shown below.

```
{
    "ips": {
        "10.18.187.72",
        "10.18.187.73"
    }
}
```

Response

The response is an array of JSON objects in the body with the following properties.

Name	Type	Description
ip	string	IP address
cve_ids	list of strings	List of CVE IDs on the inventory with the ip address.

Sample Python code

```
root_scope_id = "5fa0d242497d4f7d968c669b"
req_payload = {
  "ips":["10.18.187.72", "10.18.187.73"]
}
```

(continues on next page)

(continued from previous page)

```
resp = restclient.post('/inventory/cves/' + root_scope_id, json_body=json.dumps(req_
↪payload))
print resp.status_code
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp, indent=4, sort_keys=True)
```

17.9 Workload

The workload APIs provides programmatic access to the contents of the *Workload Profile* page. This set of APIs requires `sensor_management` or `flow_inventory_query` capability associated with the API key.

17.9.1 Workload details

This endpoint returns the specific workload given agent UUID.

```
GET /openapi/v1/workload/{uuid}
```

Path Parameter	Description
uuid	Agent UUID

Response

The response is a workload object associated with the specified UUID. The workload object's attributes schema is described below:

Attribute	Type	Description
agent_type	string	Agent type
auto_upgrade_opt_out	boolean	If true, agents do not get automatically upgraded on cluster upgrade
cpu_quota_mode	integer	CPU quota control
cpu_quota_us	integer	CPU quota usage
current_sw_version	string	Version of agent software running on the workload
data_plane_disabled	boolean	If true, flow telemetry data is not exported from the agent to the cluster
desired_sw_version	string	Version of agent software intended to be running on the workload
enable_conversation_mode	boolean	If true, conversation mode is enabled
enable_cache_sidechannel	boolean	If true, side channel attack detection is enabled
enable_forensics	boolean	If true, forensics is enabled
enable_meltdown	boolean	If true, meltdown exploit detection is enabled
enable_pid_lookup	boolean	If true, process lookup is enabled
forensics_cpu_quota_mode	integer	Forensics CPU quota control
forensics_cpu_quota_us	integer	Forensics quota usage
forensics_mem_quota_bytes	integer	Forensics memory quota in bytes
host_name	string	Host name on the workload
interfaces	array	Array of *Interface* objects
kernel_version	string	Kernel version
last_config_fetch_at	integer	Last config fetched at
last_software_update_at	integer	Last software is the timestamp at which agent reported its current version
max_rss_limit	integer	Max memory limit
platform	string	Platform of the workload
uuid	string	Unique ID of the agent
windows_enforcement_mode	string	Type of Windows enforcement mode, WAF(Windows Advanced Firewall) or WFP(Windows Filtering Platform)

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s' % (agent_uuid))
```

17.9.2 Workload Statistics

This endpoint returns statistics for a workload.

```
GET /openapi/v1/workload/{uuid}/stats?t0=<t0>&t1=<t1>&td=<td>
```

Path Parameter	Description
uuid	Agent UUID

The query URL contains the following parameters

Query Parameter	Description
t0	Start time for statistics in epoch time
t1	End time for statistics in epoch time. The end time cannot exceed the start time by more . than a day.
td	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as "minute", "hour", and "day".

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
timestamp	string	Time at which metrics were gathered (epoch or ISO 8601)
results	object	Metrics

Metrics is a JSON object with the following properties

Name	Type	Description
flow_count	integer	Number of flows.
rx_byte_count	integer	Number of received bytes.
rx_packet_count	integer	Number of received packets.
tx_byte_count	integer	Number of transmitted bytes.
tx_packet_count	integer	Number of transmitted packets.

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
td = 15 * 60 # 15 minutes
resp = restclient.get('/workload/%s/stats?t0=1483228800&t1=1485907200&td=%d' % (agent_
↪uuid, td))

# This code queries workload statistics for a week
t0 = 1483228800
for _ in range(7):
  t1 = t0 + 24 * 60 * 60
  resp = restclient.get('/workload/%s/stats?t0=%d&t1=%d&td=day' % (agent_uuid, t0,␣
↪t1))
  t0 = t1
```

17.9.3 Installed Software Packages

This endpoint returns list of packages installed on the workload.

```
GET /openapi/v1/workload/{uuid}/packages
```

Path Parameter	Description
uuid	Agent UUID

Response

Thw respone is an array of package JSON objects. The package object's schema is described below:

Attribute	Type	Description
architecture	string	Architecture of the package
name	string	Name of the package
publisher	string	Publisher of the package
version	string	Version of the package

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/packages' % (agent_uuid))
```

17.9.4 Workload Vulnerabilities

This endpoint returns list of vulnerabilities observed on the workload.

```
GET /openapi/v1/workload/{uuid}/vulnerabilities
```

The vulnerabilities object consists of a JSON body with the following keys.

Path Parameter	Description
uuid	Agent UUID

Response

The respone is an array of vulnerability JSON objects. The vulnerability object's schema is described below:

Attribute	Type	Description
cve_id	string	Common Vulnerability Exposure ID
package_infos	array	Array of *Package Info* objects
v2_score	float	CVSS V2 Score
v2_access_complexity	string	CVSS V2 Access Compleixty
v2_access_vector	string	CVSS V2 Access Vector
v2_authentication	string	CVSS V2 Authentication
v2_availability_impact	string	CVSS V2 Availability Impact
v2_confidentiality_impact	string	CVSS V2 Confidentiality Impact
v2_integrity_impact	string	CVSS V2 Intergrity Impact
v2_severity	string	CVSS V2 Severity
v3_score	float	CVSS V3 Score
v3_attack_complexity	string	CVSS V3 Attack Compleixty
v3_attack_vector	string	CVSS V3 Attack Vector
v3_availability_impact	string	CVSS V3 Availability Impact
v3_base_severity	string	CVSS V3 Base Severity
v3_confidentiality_impact	string	CVSS V2 Confidentiality Impact
v3_integrity_impact	string	CVSS V3 Intergrity Impact
v3_privileges_required	string	CVSS V3 Privileges Required
v3_scope	string	CVSS V3 Scope
v3_user_interaction	string	CVSS V3 User Interaction

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/workload/%s/vulnerabilities' % (agent_uuid))
```

17.9.5 Workload Long Running Processes

This endpoint returns list of long running processes on the workload. Long running processes are defined as processes that have at least 5 minutes uptime.

```
GET /openapi/v1/workload/{uuid}/process/list
```

Path Parameter	Description
uuid	Agent UUID

Response

The respone is a list of processes JSON objects.

Attribute	Type	Description
cmd	string	Command string of the process
binary_hash	string	Sha256 of the process binary in hex
ctime	long	ctime of the process binary in us
mtime	long	mtime of the process binary in us
exec_path	string	Process executable path
exit_usec	long	Time when the process exited in us
num_libs	integer	Number of libs the process loads
pid	integer	Process ID
ppid	integer	Parent process ID
pkg_info_name	string	Name of the package associated with the process
pkg_info_version	string	Version of the package associated with the process
proc_state	string	Process state
uptime	long	Uptime of the process in us
username	string	Username of the process
resource_usage	array	Array of *Resource Usage* object

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
resp = restclient.get('/openapi/v1/workload/%s/process/list' % (agent_uuid))
```

17.9.6 Workload Process Snapshot Summary

This endpoint returns process snapshot summary on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. The endpoint supports POST method with empty payload to enable easier future expansion.

```
POST /openapi/v1/workload/{uuid}/process/tree/ids
```

Path Parameter	Description
uuid	Agent UUID

Response

The respone is a list of process snapshot summary JSON objects.

Attribute	Type	Description
sensor_uuid	string	Agent UUID
handle	string	Handle to the process snapshot to be retrieved
process_count	integer	Number of processes in the snapshot
ts_usec	integer	Timestamp when the snapshot is captured

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
                       agent_uuid, json_body=json.dumps(payload))
```

17.9.7 Workload Process Snapshot

This endpoint returns process snapshot on this workload. A process snapshot contains all the processes that are captured by the workload at a given time. Currently one copy of the latest process snapshot is retained. This endpoint needs to be used together with the workload process snapshot summary endpoint.

```
POST /openapi/v1/workload/{uuid}/process/tree/details
```

Path Parameter	Description
uuid	Agent UUID

Payload Field	Type	Description
handle	string	Handle to the process snapshot to be retrieved

Response

The respone is a list of processes belonging to the snapshot in JSON.

Attribute	Type	Description
command_string	string	Tokenized command string
command_string_raw	string	Raw command string
binary_hash	string	Sha256 of the process binary in hex
ctime	long	ctime of the process binary in us
mtime	long	mtime of the process binary in us
exec_path	string	Process executable path
process_id	integer	Process ID
parent_process_id	integer	Parent process ID
process_key	integer	Unique key to the process
parent_process_key	integer	Unique key to the parent process
pkg_info_name	string	Name of the package associated with the process
pkg_info_version	string	Version of the package associated with the process
proc_state	string	Process state
uptime	long	Uptime of the process in us
username	string	Username of the process
cve_ids	array	Array of CVEID object

Sample Python code

```
agent_uuid = 'aa28b304f5c79b2f22d87a5af936f4a8fa555894'
payload = {
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/ids' %
            agent_uuid, json_body=json.dumps(payload))
handle = json.loads(resp.text)['process_summary'][0]['summary'][0]['handle']
payload = {
  "handle": handle,
}
resp = restclient.post('/openapi/v1/workload/%s/process/tree/details' %
            agent_uuid, json_body=json.dumps(payload))
```

17.9.8 JSON Object Definitions

17.9.8.1 Interface

Attribute	Type	Description
ip	string	IP Address of the interface
mac	string	Mac Address of the interface
name	string	Name of the interface
netmask	string	Netmask of the interface
pcap_opened	boolean	If false, packet captures are not enabled for the interface
tags_scope_id	array	Scope IDs associated with the interface
vrf	string	VRF Name
vrf_id	integer	VRF ID

17.9.8.2 Package Info

Attribute	Type	Description
name	string	Package name
version	string	Package version

17.9.8.3 Resource Usage

Attribute	Type	Description
cpu_usage	float	CPU usage
memory_usage_kb	integer	Memory usage
ts_usec	long	Timestamp in us when the resource usage is captured

17.9.8.4 CVE ID

Attribute	Type	Description
cve_id	string	cve ID
impact__cvss_v2__access_complexity	string	CVE access complexity
impact__cvss_v2__access_vector	string	CVE access vector

17.10 Enforcement

Policy enforcement is the feature where generated policies are pushed to the assets in the scope of an application and new firewall rules are written. More information can be found in the *Enforcement* documentation. This set of APIs requires the `app_policy_management` capability associated with the API key.

17.10.1 Agent Network Policy Config

This endpoint returns an *Agent* object according to the agent ID. It is useful for fetching the network policy, agent configuration, its version, etc.

```
GET /openapi/v1/enforcement/agents/{aid}/network_policy_config
```

Parameters:

The request URL contains the following parameters

Name	Type	Description
aid	string	Agent UUID for network policy config.

The JSON query body contains the following keys

Name	Type	Description
include_filter_names	boolean	Includes filter names and ID's in network policies.
inject_versions	boolean	Includes ADM workspace versions in network policies.

Response

The response of this endpoint is an *Agent* object.

17.10.2 Concrete Policy Statistics

This endpoint returns statistics for concrete policies given the agent ID and the concrete policy ID. The endpoint returns an array of *Timeseries Concrete Policy Result* objects.

```
GET /openapi/v1/enforcement/agents/{aid}/concrete_policies/{cid}/stats?t0=<t0>&t1=<t1>
↪&td=<td>
```

Parameters:

The request URL contains the following parameters

Name	Type	Description
aid	string	Agent UUID for statistics.
cid	string	Concrete Policy UUID for statistics.

The JSON query body contains the following keys

Name	Type	Description
t0	integer	Start time for statistics in epoch time
t1	integer	End time for statistics in epoch time
td	integer or string	Granularity for statistic aggregations. An integer specifies number of seconds. Strings may be passed such as "minute", "hour", and "day".

17.10.3 JSON Object Definitions

17.10.3.1 Agent

Attribute	Type	Description
agent_uuid	string	Agent UUID.
agent_config	object	*Agent Config*
agent_config_status	object	*Agent Config Status*
desired_network_policy_config	object	*Network Policy Configuration*
provisioned_network_policy_config	object	*Provisioned Network Policy Config*
provisioned_state_update_timestamp	integer	epoch timestamp in seconds when agent acknowledged the above provisioned policy.
desired_policy_update_timestamp	integer	epoch timestamp in seconds when desired_network_policy_config is generated.
agent_info	object	*Agent Info*
skipped	boolean	true, when concrete policy generation is skipped.
message	string	Reason why concrete policy generation is skipped.

17.10.3.2 Agent Config

Attribute	Type	Description
agent_uuid	string	Agent UUID.
enforcement_enabled	boolean	Config stating is enforcement is enabled on Agent.
fail_mode	string	Fail Mode.
version	number	Agent config version number.
control_tet_rules_only	boolean	Control tet rules only config.
allow_broadcast	boolean	Allow Broadcast config.
allow_multicast	boolean	Allow Multicast config.
allow_link_local	boolean	Allow Link Local config.
enforcement_cpu_quota_mode	string	Enforcement Agent CPU quota mode.
enforcement_cpu_quota_us	string	Enforcement Agent CPU quota micros sec.
enforcement_max_rss_limit	number	Enforcement Agent Max RSS limit.

17.10.3.3 Network Policy Configuration

Attribute	Type	Description
version	string	Version number.
network_policy	array	Array of *Network Policy* objects.
address_sets	array	Array of *Address Set* objects for IP set feature.
container_network_policy	array	Array of *ContainerNetworkPolicy* objects.

17.10.3.4 Network Policy

Attribute	Type	Description
priority	string	Priority of concrete policy.
enforcement_intent_id	string	Enforcement Intent ID.
concrete_policy_id	string	Concrete Policy ID.
match	object	*Match* criteria for policy. This field is deprecated.
action	object	*Action* for policy match.
workspace_id	string	ID for ADM/enforcement workspace.
adm_data_set_id	string	ADM data set id of workspace.
adm_data_set_version	string	ADM data set version of the workspace. Set only when inject_versions=true is passed in params.
cluster_edge_id	string	Cluster Edge ID.
policy_intent_group_id	string	Policy intent group ID.
match_set	object	*Match Set* object for IP set support. Exactly one of match or match_set will be present.
src_filter_id	string	Source inventory filter ID. This will be set when include_filter_names=true passed as params.
src_filter_name	string	Source inventory filter name. This will be set when include_filter_names=true passed as params.
dst_filter_id	string	Destination inventory filter ID. This will be set when include_filter_names=true passed as params.
dst_filter_name	string	Destination Inventory filter name. This will be set when include_filter_names=true passed as params.

17.10.3.5 ContainerNetworkPolicy

Attribute	Type	Description
pod_id	string	POD ID.
network_policy	array	Array of *Network Policy* objects.
deployment	string	Deployment Name.
service_endpoint	array	List of service endpoint names.

17.10.3.6 Match

Attribute	Type	Description
src_addr	object	*Subnet* object for source address.
dst_addr	object	*Subnet* object for destination address.
src_port_range_start	int	Source port range start.
src_port_range_end	int	Source port range end.
dst_port_range_start	int	Destination port range start.
dst_port_range_end	int	Destination port range end.
ip_protocol	string	IP Protocol.
address_family	string	IPv4 or IPv6 address family.
direction	string	Direction of match, INGRESS or EGRESS.
src_addr_range	object	*Address Range* object for source address.
dst_add_range	object	*Address Range* object for destination address.

17.10.3.7 Action

Attribute	Type	Description
type	string	Action type.

17.10.3.8 Match Set

Attribute	Type	Description
src_set_id	string	Source set ID of *Address Set* object in the *Network Policy Configuration* `address_sets` array.
dst_set_id	string	Destination set ID of *Address Set* object in the *Network Policy Configuration* `address_sets` array.
src_ports	array	Array of *Port Range* objects for source ports.
dst_ports	array	Array of *Port Range* objects for destination ports.
ip_protocol	string	IP Protocol.
address_family	string	IPv4 or IPv6 address family.
direction	string	Direction of match, INGRESS or EGRESS.

17.10.3.9 Address Set

Attribute	Type	Description
set_id	string	Address set ID.
addr_ranges	array	Array of *Address Range* objects.
subnets	array	Array of *Subnet* objects.
addr_family	string	IPv4 or IPv6 address family.

17.10.3.10 Subnet

Attribute	Type	Description
ip_addr	string	IP address.
prefix_length	int	Prefix length for subnet.

17.10.3.11 Address Range

Attribute	Type	Description
start_ip_addr	string	Start IP address for range.
end_ip_addr	string	End IP address for range.

17.10.3.12 Port Range

Attribute	Type	Description
start_port	int	Start port for range.
end_port	int	End port for range.

17.10.3.13 Agent Config Status

Attribute	Type	Description
disabled	boolean	Config stating is enforcement is disabled on Agent.
current_version	number	Current Agent config version applied on Agent.
highest_seen_version	number	Highest version of agent config received by Agent.

17.10.3.14 Provisioned Network Policy Config

Attribute	Type	Description
version	string	Network policy config version provisioned by Agent.
error_reason	string	CONFIG_SUCCESS when Agent successfully applied policies else error reason.
disabled	boolean	Config stating is enforcement is disabled on Agent.
current_version	number	Current NPC version applied on Agent.
highest_seen_version	number	Highest version of NPC received by Agent.
policy_status	object	Every network policy status.

17.10.3.15 Agent Info

Attribute	Type	Description
agent_info_supported	boolean	Agent capability if agent_info is supported.
ipset_supported	boolean	Agent capability if ipsets are supported.

17.10.3.16 Concrete Policy Result

Attribute	Type	Description
byte_count	int	Byte count for concrete policy hits.
pkt_count	int	Packet count for concrete policy hits.

17.10.3.17 Timeseries Concrete Policy Result

Attribute	Type	Description
timestamp	string	Timestamp string for aggregation of results.
result	object	*Concrete Policy Result*

17.11 Client Server configuration

Detecting client and server relationships is central to various features in Tetration which is why we recommend using the Software Agent whenever possible as it can report the ground truth. Any telemetry monitoring point in the network cannot guarantee to observe every packet for a given flow - due to a wide range of circumstances, for example: two unidirectional halves of a TCP flow may take unique paths through the network - therefore will always unavoidably affected by a level of error.

Tetration attempts to detect and minimise these errors without any user interaction by applying machine learning algorithms to each flow, building a statistical model which provides a judgement when inconsistent telemetry is reported. For the majority of cases, users do not need to worry about this set of APIs. However, in some minority of cases the client server detection algorithm does not get the flow direction correct. Features which rely on flow direction, for example, ADM, may exhibit undesired behaviour like opening unnecessary ports.

A set of APIs are provided that can be used to provide hints about known server ports to Tetration algorithms. This set of APIs is available to users with root scope ownership role and requires the `app_policy_management` capability associated with the API key for those users.

There are 2 options for Client Server configuration:

17.11.1 Host Config

Configuration of known server ports that are applicable to a specific subset of IP addresses within a root scope

17.11.1.1 Add server port configuration

This API can be used to provide hints to Tetration algorithms about known server ports for a given root scope. Users can provide a list of known TCP/UDP server ports for a set of IP addresses belonging to a root scope to aid Tetration algorithms with figuring out client server direction correct in flows.

```
POST /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
root_scope_id	string	Unique identifier for the root scope.

Additionally, a text file provided as input to this API contains the endpoint server port configuration in the following format:

17.11.1.2 Endpoint server port configuration

Attribute	Type	Description
ip_address	string	IP Address (can be ipv4 or ipv6 address). Subnets are not allowed.
tcp_server_ports	List of int	List of known TCP server ports corresponding to the ip_address.
udp_server_ports	List of int	List of known UDP server ports corresponding to the ip_address.

17.11.1.3 Bulk server port configuration

Attribute	Type	Description
host_config	List of *Endpoint server port configuration* objects.	List of IP addresses with associated known server ports.

Sample python code

```
# contents of below file:
# {"host_config": [
#    {"ip_address": "1.1.1.1",
#     "tcp_server_ports": [100, 101, 102],
#     "udp_server_ports": [103]
#    },
#    {"ip_address": "1.1.1.2",
#     "tcp_server_ports": [200, 201, 202]
#    }
#  ]
# }

file_path = '/<path_to_file>/server_ports.txt'
root_scope_id = '<root-scope-id>'
restclient.upload(file_path,
                  '/adm/%s/server_ports' % root_scope_id,
                  timeout=200) # seconds
```

Note: Above API overwrites the full state of known server port configuration in the backend. If user needs to modify anything, they need re-upload the full configuration after modifications.

17.11.1.4 Get server port configuration

This API returns list of known server ports for endpoints in a root scope uploaded by the user.

```
GET /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
root_scope_id	string	Unique identifier for the root scope.

Response object: A list of ref:*ServerPortConfig* objects.

Sample python code

```
root_scope_id = '<root-scope-id>'
restclient.get('/adm/%s/server_ports' % root_scope_id)
```

17.11.1.5 Delete server port configuration

This API deletes server port configuration for specified root scope.

```
DELETE /openapi/v1/adm/{root_scope_id}/server_ports
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
root_scope_id	string	Unique identifier for the root scope.

Response object: None.

Sample python code

```
root_scope_id = '<root-scope-id>'
restclient.delete('/adm/%s/server_ports' % root_scope_id)
```

17.11.2 Port Config

Configuration of known server ports that are applicable to all IP addresses that belong to a root scope

17.11.2.1 Push server port configuration

This API can be used to provide hints to Tetration algorithms about known server ports for a given root scope. Users can provide a list of known TCP/UDP server ports for a given root scope to aid Tetration algorithms with figuring out client server direction correct in flows. Users also have the option of specifying a service name associated with each server port.

There is also a default list of known services that are applicable to all root scopes(hereafter referred to as global services). This list can be overridden at any point by the user.

17.11.2.2 Service configuration

A service is defined to be a (port, name) pair.

Attribute	Type	Description
port	int	TCP/UDP server port number
name	string	Service name associated with this port (optional)
override_in_conflicts	boolean	Force host to be provider in case of a conflict (optional)

17.11.2.3 Bulk service configuration

Attribute	Sub-Attribute	Type	Description
server_ports_config	tcp_service_list	List of *Service configuration* objects.	List of known TCP services
	udp_service_list	List of *Service configuration* objects.	List of known UDP services

```
Push services per root scope:
POST /openapi/v1/adm/{root_scope_id}/server_ports_config
```

Sample python code

```
# contents of below file:
#{"server_ports_config":
#    {
#        "tcp_service_list": [
#                {
#                        "port": 80,
#                        "name" : "http"
#
#                },
#                {
#                        "port": 53,
#                        "name" : "dns"
#                },
#                {
#                        "port": 514,
#                        "name" : "syslog",
#                        "override_in_conflicts": true
#                }
#        ],
#        "udp_service_list": [
#                {
#                        "port": 161
#                },
#                {
#                        "port": 53,
#                        "name": "dns"
#                }
#        ]
#    }
#}

file_path = '/<path_to_file>/server_ports.json'

# Updating service list for a given root scope
```

(continues on next page)

(continued from previous page)

```
#restclient.upload(file_path,
#                   '/openapi/v1/adm/{root_scope_id}/server_ports_config',
#                   timeout=200) # seconds
```

Note: Above API overwrites the full state of known server port configuration in the backend. If user needs to modify anything, they need re-upload the full configuration after modifications.

17.11.2.4 Retrieve server port configuration

This API returns list of known server ports in a root scope uploaded by the user. Response is *Bulk service configuration*.

```
Retrieve configured services per root scope:
    GET /openapi/v1/adm/{root_scope_id}/server_ports_config

Retrieve configured global services:
    GET /openapi/v1/adm/server_ports_config
```

17.11.2.5 Remove server port configuration

This API deletes server port configuration for specified root scope.

```
Remove configured services per root scope:
    DELETE /openapi/v1/adm/{root_scope_id}/server_ports_config
```

17.12 Software Agents

17.12.1 Agent APIs

The software agents APIs are associated with managing Tetration software agents. These set of APIs require the `sensor_management` capability associated with the API key. *GET* APIs below are also available with `flow_inventory_query` capability associated with the API key.

17.12.1.1 Get software agents

This endpoint returns a list of software agents.

```
GET /openapi/v1/sensors
```

Parameters:

Name	Type	Description
limit	integer	Limits the number of results returned (optional)
offset	string	Offset is used for paginated requests. If response returns offset then subsequent request must use the same offset to get more results in the next page. (optional)

17.12.1.2 Get specific software agent

This endpoint returns attributes for the agent whose UUID is part of the URI.

```
GET /openapi/v1/sensors/{uuid}
```

17.12.1.3 Deleting software agent

This endpoint is used to decommission a software agent given its UUID. This API must be used with caution; once an agent is deleted, it does not show up in the Tetration dashboard and if the agent is active, flow exports from the agent are not allowed in Tetration.

```
DELETE /openapi/v1/sensors/{uuid}
```

17.12.2 Software agent configuration using Intents

This API workflow uses few REST endpoints defined below.

17.12.2.1 Creating an inventory filter

This endpoint is used to specify criteria that match agent hosts on which user wants to configure software agents.

```
POST /openapi/v1/filters/inventories
```

Parameters:

Name	Type	Description
app_scope_id	string	The scope ID to assign to the inventory filter.
name	string	A name for the inventory filter.
query	json	Filter or match criteria for agent host.

Sample python code

```
# app_scope_id can be retrieved by /app_scopes API
req_payload = {
   "app_scope_id": <app_scope_id>,
   "name": "sensor_config_inventory_filter",
   "query": {
      "type": "eq",
      "field": "ip",
      "value": <sensor_interface_ip>
```

(continues on next page)

(continued from previous page)

```
    }
}
resp = restclient.post('/filters/inventories',
                       json_body=json.dumps(req_payload)))
print resp.status_code
# returned response will contain the created filter and it's ID.
```

17.12.2.2 Creating a software agent configuration profile

This endpoint is used to specify the set of configuration options to apply to target set of software agents.

```
POST /openapi/v1/inventory_config/profiles
```

Following configuration options can be specified as part of agent configuration profile:

- allow_broadcast: option to allow/disallow broadcast traffic (default value of this option is True).
- allow_multicast: option to allow/disallow multicast traffic (default value of this option is True).
- allow_link_local: option to allow/disallow link local traffic (default value of this option is True).
- auto_upgrade_opt_out: if true, agents are not auto-upgraded during upgrade of Tetration cluster.
- cpu_quota_mode & cpu_quota_usec: these options are used to police the amount of CPU quota to give to agent on the end host.
- data_plane_disabled: if true, agent stops reporting flows to Tetration.
- enable_conversation_mode: option to enable conversation mode on all sensors.
- enable_forensics: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- enable_meltdown: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- enable_pid_lookup: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- enforcement_disabled: can be used to disable enforcement on hosts running enforcement agents.
- preserve_existing_rules: option to specify whether to preserve existing iptable rules.
- windows_enforcement_mode: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to *Software Agent Config*

Sample python code

```
# Define profile to disable data_plane on agent
req_payload = {
   "root_app_scope_id": <root_app_scope_id>,
   "data_plane_disabled": True,
   "name": "sensor_config_profile_1",
   "enable_pid_lookup": True,
   "enforcement_disabled": False
}
resp = restclient.post('/inventory_config/profiles',
                       json_body=json.dumps(req_payload))
print resp.status_code
```

(continues on next page)

(continued from previous page)

```
# returned response will contain the created profile and it's ID.
parsed_resp = json.loads(resp.content)
```

17.12.2.3 Get software agent configuration profiles

This endpoint returns a list of software agent configuration profiles visible to the user.

```
GET /openapi/v1/inventory_config/profiles
```

Parameters: None

17.12.2.4 Get specific software agent configuration profile

This endpoint returns an instance of software agent configuration profile.

```
GET /openapi/v1/inventory_config/profiles/{profile_id}
```

Returns the software agent configuration profile object associated with the specified ID.

17.12.2.5 Update a software agent configuration profile

This endpoint updates a software agent configuration profile.

```
PUT /openapi/v1/inventory_config/profiles/{profile_id}
```

Following configuration options can be specified as part of agent configuration profile:

- allow_broadcast: option to allow/disallow broadcast traffic (default value of this option is True).
- allow_multicast: option to allow/disallow multicast traffic (default value of this option is True).
- allow_link_local: option to allow/disallow link local traffic (default value of this option is True).
- auto_upgrade_opt_out: if true, agents are not auto-upgraded during upgrade of Tetration cluster.
- cpu_quota_mode & cpu_quota_usec: these options are used to police the amount of CPU quota to give to agent on the end host.
- data_plane_disabled: if true, agent stops reporting flows to Tetration.
- enable_conversation_mode: option to enable conversation mode on all sensors.
- enable_forensics: option to enable collection of forensic events on the workload (agent uses more CPU as a result).
- enable_meltdown: enables Meltdown Exploit detection on the workload (agent uses more CPU as a result).
- enable_pid_lookup: if true, agent tries to attach process information to flows. Note this config option uses more CPU on the end host.
- enforcement_disabled: can be used to disable enforcement on hosts running enforcement agents.
- preserve_existing_rules: option to specify whether to preserve existing iptable rules.
- windows_enforcement_mode: option to use WAF (Windows Advanced Firewall) or WFP (Windows Filtering Platform) (default option is WAF).

For more details about the configuration options, refer to *Software Agent Config*

Returns the modified software agent configuration profile object associate with the specified ID.

17.12.2.6 Delete a software agent configuration profile

This endpoint deletes the specified software agent configuration profile.

```
DELETE /openapi/v1/inventory_config/profiles/{profile_id}
```

17.12.2.7 Creating a software agent configuration intent

This endpoint is used to specify the intent to apply set of configuration options to specified set of software agents. This will create the intent and updates the intent order by adding the newly created intent to the order.

```
POST /openapi/v1/inventory_config/intents
```

Sample python code

```
req_payload = {
  "inventory_config_profile_id": <>,
  "inventory_filter_id": <>
}
resp = restclient.post('/inventory_config/intents',
               json_body=json.dumps(req_payload))
print resp.status_code
# returned response will contain the created intent object and it's ID.
```

17.12.2.8 Specifying order of intents

This endpoint is used to specify the ordering of various software agent configuration intents. For example, there could be two intents – one to enable process ID lookup on development machines and second one to disable process ID lookup on windows machines. If the first intent has higher priority, then development windows machines will have process ID lookup enabled. NOTE: By default, when intent is created, it is added to the beginning of intent orders list. This endpoint is only to be used if end user needs to modify the existing order of intents.

```
POST /openapi/v1/inventory_config/orders
```

Sample python code

```
# Read the agent config intents ordered list
resp = restclient.get('/inventory_config/orders')
order_result_json = json.loads(resp.content)

# Modify the list by prepending the new intent in the list
order_rslt_json['intent_ids'].insert(0,<intent_id>)

# Post the new ordering back to the server
resp = restclient.post('/inventory_config/orders',
               json_body=json.dumps(order_rslt_json))
```

17.12.2.9 Remove agent config intent

This endpoint is used to remove a specific agent configuration intent.

```
DELETE /openapi/v1/inventory_config/intents/{intent_id}
```

Sample python code

```
intent_id = '588a51dcb5b30d0ee6da084a'
resp = restclient.delete('/inventory_config/intents/%s' % intent_id)
```

17.12.3 Interface Config Intents

Recommended way to assign VRFs to agents is using Remote VRF configuration settings. In rare cases, when agent hosts may have multiple interfaces that need to be assigned different VRFs, users can choose to assign them VRFs using Interface Config Intents. See Settings > Agent Config > Software Agent Config for more details.

17.12.3.1 Inventory Config Intent Object

The GET and POST methods return an array of inventory config intent JSON objects. The object's attributes are described below:

Attribute	Type	Description
vrf_id	integer	VRF ID integer
vrf_name	string	VRF Name
inventory_filter_id	string	Inventory Filter ID
inventory_filter	JSON	Inventory filter. See OpenAPI > Inventory Filters for more details.

17.12.3.2 Get Interface Config Intents

This endpoint returns a list of inventory config intents to the user.

```
GET /openapi/v1/inventory_config/interface_intents
```

Parameters: None

17.12.3.3 Create or Update list of Interface Config Intents

This endpoint is used to create or modify list of interface config intents. The API takes an ordered list of intents. To remove an intent in this list, users would need to read the existing list of intents, modify the list and write the modified list back.

```
POST /openapi/v1/inventory_config/interface_intents
```

Parameters:

Name	Type	Description
inventory_filter_id	string	Inventory filter ID to match interface
vrf_id	integer	VRF ID to assign interface

Sample python code

```
req_payload = {
  "intents": [
    {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_1>},
    {"inventory_filter_id": <inventory_filter_id_1>, "vrf_id": <vrf_id_2>}
  ]
}
resp = restclient.post('/inventory_config/interface_intents', json_body=json.
↪dumps(req_payload))
```

17.12.4 VRF configuration for agents behind NAT

Following set of APIs are useful to specify policies to assign VRFs to agents behind NAT boxes. These set of APIs require the `sensor_management` capability associated with the API key and are only available to site admin users.

17.12.4.1 List VRF configuration rules for agents behind NAT

This endpoint returns a list of VRF configuration rules applicable to agents behind NAT.

```
GET /openapi/v1/agentnatconfig
```

17.12.4.2 Create a new VRF configuration applicable to agents behind NAT

This endpoint is used to specify criteria for VRF labeling for hosts based on their source IP and source port as seen by Tetration appliance.

```
POST /openapi/v1/agentnatconfig
```

Parameters:

Name	Type	Description
src_subnet	string	Subnet to which source IP can belong to (CIDR notation).
src_port_range_start	integer	Lower bound of source port range (0-65535).
src_port_range_end	integer	Upper bound of source port range (0-65535).
vrf_id	integer	VRF ID to use for labeling flows for agents whose source address and port falls in the above specified range.

Sample python code

```
req_payload = {
    src_subnet: 10.1.1.0/24,          # src IP range for sensors
    src_port_range_start: 0,
    src_port_range_end: 65535,
    vrf_id: 676767                    # VRF ID to assign
}

resp = rc.post('/agentnatconfig', json_body=json.dumps(req_payload))
print resp.status_code
```

17.12.4.3 Delete existing VRF configuration

```
DELETE /openapi/v1/agentnatconfig/{nat_config_id}
```

17.13 Tetration software download

The Tetration software download feature provides a way to download software packages for Tetration agents. These set of APIs require the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners and users with agent installer roles.

17.13.1 API to get supported platforms

This end point returns the list of supported platforms.

```
GET /openapi/v1/sw_assets/platforms
```

Parameters: None

Reponse object: Returns the list of supported platforms.

Sample python code

The sample code below retrieves all the supported platforms.

```
resp = restclient.get('/sw_assets/platforms')
if resp.status_code == 200:
   parsed_resp = json.loads(resp.content)
   print json.dumps(parsed_resp)
```

Sample response

```
{"results": [{"platform": "OracleServer-6.3", "agent_type": "enforcer", "arch": "x86_
↪64"}, {"platform": "MSWindows8Enterprise", "agent_type": "legacy_sensor", "arch":
↪"x86_64"}]}
```

17.13.2 API to get supported software version

This endpoint returns the list of supported software version for specified "agent_type", "package_type", "platform" and "architecture".

```
GET /openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_
↪type=<pkg_type>&arch=<arch>&list_version=<list_version>
```

where <agent_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg_type> can be either "sensor_w_cfg" or "sensor_bin_pkg". Both <pkg_type> and <agent_type> are optional but at least one of them should be specified. <list_version> must be "True" to enable this API.

Parameters: The request URL contains the following parameters

Name	Type	Description
platform	string	Specify the platform.
agent_type	string	(optional) Specify the agent type.
pkg_type	string	(optional) Specify the package type, the value can be either "sensor_w_cfg" or "sensor_bin_pkg".
arch	string	Specify the architecture.
list_version	string	Set to "True" to enable software version search.

Response object: Returns a list of supported software version.

Sample python code

```
resp = restclient.get('/sw_assets/download?platform=OracleServer-6.3&pkg_type=sensor_
↪w_cfg&arch=x86_64&list_version=True')
if resp.status_code == 200:
   print resp.content
```

Sample response

```
3.3.1.30.devel
3.3.1.31.devel
```

17.13.3 API to download Tetration software

This endpoint enables clients to download the software for specified "agent_type", "package_type", "platform", "architecture" and "sensor_version".

```
GET /openapi/v1/sw_assets/download?platform=<platform>&agent_type=<agent_type>&pkg_
↪type=<pkg_type>&arch=<arch>&sensor_verion=<sensor_version>
```

where <agent_type>, <platform> and <arch> can be any one of the results retrieved from the **API to get supported platforms**, and <pkg_type> can be either "sensor_w_cfg" or "sensor_bin_pkg". Both <pkg_type> and <agent_type> are optional but at least one of them should be specified. <sensor_version> can be any one of the results retrieved from the **API to get supported software version**. If "sensor_version" is not specified, it will download the **latest** software.

Parameters: The request URL contains the following parameters

Name	Type	Description
platform	string	Specify the platform.
agent_type	string	(optional) Specify the agent type.
pkg_type	string	(optional) Specify the package type, the value can be either "sensor_w_cfg" or "sensor_bin_pkg".
arch	string	Specify the architecture.
sensor_version	string	(optional) Specify the software version, defaults to empty string.

Response object: Returns the Tetration software for the given parameters.

Sample python code

```
resp = restclient.download('<download_path>/<file_name>','/sw_assets/download?
↪platform=OracleServer-6.3&pkg_type=sensor_w_cfg&arch=x86_64&sensor_version=3.3.1.30.
↪devel')
if resp.status_code == 200:
   print 'file downloaded successfully'
```

17.14 Tetration Agents Upgrade

The Tetration agents upgrade feature provides a way to upgrade installed Tetration agents to specific version. It only updates the metadata, actual upgrade will happen during next check-in. The API requires the `software_download` capability associated with the API key. This capability is only available to site admin users, root scope owners or users with agent installer roles.

17.14.1 API to upgrade an agent to specific version

This end point triggers the agent given its "UUID" upgrade to specific "sensor_version", the latest version will be applied if "sensor_version" is not provided. This API won't proceed downgrade requests.

```
POST /sensor_config/upgrade/{UUID}?sensor_version=<sensor_version>
```

where <sensor_version> can be any one of the results retrieved from the *API to get supported software version*.

Parameters: The request URL contains the following parameters

Name	Type	Description
sensor_version	string	(optional) Specify the desired version, the latest version will be applied by default

Returns the status for this upgrade request.

Sample python code

```
resp = restclient.post('/sensor_config/upgrade/{UUID}?sensor_version=3.4.1.1.devel')
if resp.status_code == 200:
  print 'agent upgrade was triggered successfully and in progress'
elif resp.status_code == 304:
  print 'provided version is not newer than current version'
elif resp.status_code == 400:
  print 'provided version is invalid'
elif resp.status_code == 403:
  print 'user does not have required capability'
elif resp.status_code == 404:
  print 'agent with {UUID} does not exist'
```

17.15 Switches

The switch related APIs are associated with managing Tetration hardware agents. These set of APIs require the `hw_sensor_management` capability associated with the API key.

Note: These APIs are only available to site admin users.

17.15.1 Switch object

The switch object attributes are described below:

Attribute	Type	Description
serial	string	Serial number of the switch.
last_checkin_epoch	integer	Unix timestamp of when the switch last checked in.
name	string	Switch name.
ip	string	Switch IP address.
nxos_version	string	Switch SW version.
agent_version	string	Agent SW version.
bootup_time_epoch	integer	Unix timestamp of when the switch booted up.
export_interval_ms	integer	Export interval to Tetration cluster.
datapath_disabled	boolean	If true, switch stops reporting flows to Tetration.
hw_sensors	JSON	Array of `HW sensor objects`.
catchall_vrf_id	integer	ID of catchall VRF.
role	string	Role associated with the switch.
gateway_uuid	string	Gateway UUID.
deleted_at	integer	If the switch was deleted, then this parameter provides the timestamp at which the object was deleted.

The `HW sensor object` attributes are described below:

Attribute	Type	Description
name	string	Name of the HW Sensor.
decommissioned	boolean	Set to true for decommissioned HW sensors.
exporter_id	integer	Exporter ID.

17.15.2 Get switches

This endpoint returns a list of switches known to Tetration appliance.

```
GET /openapi/v1/switches
```

Parameters: None

Response object: Array of switch objects.

Sample python code

```
restclient.get('/switches')
```

17.15.3 Configure switch

This endpoint is used to configure a switch given its serial number.

```
PUT /openapi/v1/switches/{serial}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
serial	string	Serial number of the switch.

The query body consists of a json body with the following keys used to configure one or more of the following configuration options for a switch with specified serial number.

Keys	Values
datapath_disabled	Optional parameter. If true, switch stops reporting flows to Tetration
export_interval_ms	Optional parameter. Export interval to Tetration cluster
catchall_vrf_id	Optional parameter. Default Catch All Vrf Id

Response object: None

Sample python code

```
req_payload = {'export_interval_ms': 60000}
resp = restclient.put('/switches/%s' % switch_serial,
                      json_body=json.dumps(req_payload))
```

17.15.4 Delete switches

This endpoint deletes a switch given its serial number. This API must be used with caution.

```
DELETE /openapi/v1/switches/{serial}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
serial	string	Serial number of the switch.

Response object: None

Sample python code

```
serial = '<serial>'
restclient.delete('/switches/%s' % serial)
```

17.16 Collection Rules

These set of APIs can be used to manage collection rules. Collection rules in Tetration appliance are means for user to specify what IP addresses or subnets are interesting for their deployment. If the deployment has any switches that support Tetration analytics, then these collection rules are sent to the switches (user needs to check the 'Apply to

switches' checkbox on the dashboard). On receiving these collection rules, switches only extract traffic signals for IP addresses that match these sets of collection rules. These APIs require the `hw_sensor_management` capability associated with the API key.

Note: These APIs are only available to site admin users.

17.16.1 Collection rule object

The collection rule object attributes are described below:

Attribute	Type	Description
subnet	string	Subnet or IP address in CIDR format.
action	string	Possible values are 'INCLUDE' or 'EXCLUDE'.

17.16.2 Update new collection rules for a VRF

This endpoint can be used to update the ordered list of collection rules for the specified VRF. Note, the list of collection rules in the POST request is treated as an ordered list.

```
POST /openapi/v1/collection_rules/{vrf_name}
```

Parameters:

Ordered list of collection rule objects in the POST body. **The last two rules must be catch all rules for IPv4 and IPv6.** The rules may specify the subnets 0.0.0.0/0 and ::/0 respectively, similar to the example below.

Response object: Updated ordered list of collection rules for the VRF.

Sample python code

```
req_payload = [
    {
       "subnet": "10.10.10.0/24",
       "action": "INCLUDE"
    },
    {
       "subnet": "11.11.11.0/24",
       "action": "INCLUDE"
    },
    {
       "subnet": "0.0.0.0/0",    # catch all rule for IPV4 addresses
       "action": "EXCLUDE"
    },
    {
       "subnet": "::/0",         # catch all rule for IPV6 addresses
       "action": "EXCLUDE"
    }
]
resp = restclient.post('/collection_rules/test_vrf', json_body=json.dumps(req_
↪payload))
```

17.16.3 Get collection rules for a VRF

This endpoint returns an ordered list of collection rules for a specified VRF.

```
GET /openapi/v1/collection_rules/{vrf_name}
```

Parameters: None

Response object: Ordered list of collection rules for a specified VRF.

Sample python code

```
resp = restclient.get('/collection_rules/test_vrf')
```

17.16.4 Impact of Collection Rules

There are 2 kinds of inventory items:

- Sensor learnt (*Workload Profile*): Includes all IP addresses that belong to workloads running Tetration sensors
- Flow learnt (*Inventory Profile*): Includes all IP addresses that were seen in flow signals collected by Tetration but are not associated with any workloads running Tetration agents.

EXCLUDE/INCLUDE collection rules control what inventory items are tracked. Sensor learnt inventory items are always tracked, irrespective of collection rules. For flow learnt inventory items, if they are excluded by collection rules, inventory item will not exist. Therefore, inventory search will not return any result for such inventories.

Flow search is unaffected by collection rules, except the labels column, which will not be populated for the IP excluded by collection rules. Collection rules have no bearing on determination of client-server for any given flow.

ADM results may be affected as we do not track labels for IPs excluded by collection rules.

17.17 User Uploaded Filehashes

Users can upload a list of filehashes to Tetration and specify whether those hashes are benign or flagged. Tetration will flag processes with the respective binary hashes accordingly.

This set of APIs can be used to upload or remove list of filehashes to Tetration. To call these APIs, use an API key with the `user_data_upload` capability.

Note: You can have up to 1 million file hashes per root scope. 500000 for both benign and flagged hashes each.

The following APIs are available to scope owners and site admins and are used to upload/download/remove filehashes in a single root scope on the |product| appliance.

17.17.1 User filehash upload

This endpoint is used to upload a CSV file with filehash for a root scope on the Tetration appliance. The column headers `HashType` and `FileHash` must appear in the CSV file. `HashType` should be `SHA-1` or `SHA-256`, `FileHash` must not be empty and must be in the format of 40-hex SHA1 or 64-hex SHA256.

`FileName` and `Notes` headers are optional. Given file name should not exceed a maximum length of 150 characters and given notes should not exceed a maximum length of 1024 characters.

```
POST /openapi/v1/assets/user_filehash/upload/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of `benign` or `flagged`.

Response object: None

Sample python code

```
# Sample CSV File
# HashType,FileHash,FileName,Notes
# SHA-1,1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29,application_1.exe,Sample Notes
# SHA-256,8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4,
↪application_2.exe,Sample Notes

file_path = '/<path_to_file>/user_filehash.csv'
root_app_scope_name = 'Tetration'
restclient.upload(file_path, '/assets/user_filehash/upload/%s/benign' % root_app_
↪scope_name)
```

17.17.2 User filehash delete

This endpoint is used to upload a CSV file to delete filehashes from root scope on the Tetration appliance. CSV file must have `FileHash` as a header.

```
POST /openapi/v1/assets/user_filehash/delete/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of `benign` and `flagged`.

Response object: None

Sample python code

```
# Sample CSV File
# FileHash
# 1AF17E73721DBE0C40011B82ED4BB1A7DBE3CE29
# 8F434346648F6B96DF89DDA901C5176B10A6D83961DD3C1AC88B59B2DC327AA4

file_path = '/<path_to_file>/user_filehash.csv'
root_app_scope_name = 'Tetration'
restclient.upload(file_path, '/assets/user_filehash/delete/' + root_app_scope_name +
↪'/benign')
```

17.17.3 User filehash download

This endpoint returns the user file hash for the given root scope on the Tetration appliance as a CSV file. The CSV file will have the headers `HashType`, `FileHash`, `FileName` and `Notes` in the respective order.

```
GET /openapi/v1/assets/user_filehash/download/{rootAppScopeNameOrID}/{benignOrflagged}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeNameOrID	string	Root scope name or ID.
benignOrflagged	string	Can be one of `benign` or `flagged`.

Response object: None

Sample python code

```
file_path = '/<path_to_file>/output_user_filehash.csv'
root_app_scope_name = 'Tetration'
restclient.download(file_path, '/assets/user_filehash/download/%s/benign' % root_app_
↪scope_name)
```

17.18 User defined labels

These APIs are used to add or remove user defined labels that label flows and inventory items on the Tetration appliance. To call these APIs, use an API key with the `user_data_upload` capability. Please refer to the *Label schema* section of the UI user guide for guidelines governing keys and values used for labeling flows and inventory items.

Note: Refer to Inventory > ../inventory/upload for instructions on accessing this functionality via the UI.

Note: Refer to *Label limits* for limits on the number of IPv4/IPv6 addresses/subnets that can be uploaded.

17.18.1 Scope dependent APIs

The following APIs are used to get/set/delete labels in a single root scope on the Tetration appliance. They are available to root **scope owners** and **site admins**. Additionally, the GET API calls are available to users with **read access** to the root scope.

17.18.1.1 Get Inventory Label

This endpoint returns labels for an IPv4/IPv6 address or subnet in a root scope on the Tetration appliance. The address/subnet used to query this endpoint must exactly match the one used for uploading labels.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object:

Name	Type	Description
attributes	JSON	Key/value map for labeling matching flows and inventory items

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.get('/inventory/tags/%s' % root_app_scope_name, params={'ip': '10.1.1.1/24
↪'})
```

17.18.1.2 Search Inventory Label

This endpoint allows for searching labels for an IPv4/IPv6 address or subnet in a root scope on the Tetration appliance.

```
GET /openapi/v1/inventory/tags/{rootAppScopeName}/search?ip={IPorSubnet}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
IPorSubnet	string	IPv4/IPv6 address or subnet.

Response object: This API returns a list of objects of the following format

Name	Type	Description
key	string	IPv4/IPv6 address or subnet.
updatedAt	integer	Unix timestamp of when the labels were updated.
value	JSON	Key/value map of attributes for the key.

Sample python code

```
root_app_scope_name = 'Tetration Scope'
encoded_root_app_scope_name = urllib.quote(root_app_scope_name, safe='')
restclient.get('/inventory/tags/%s/search' % encoded_root_app_scope_name, params={'ip
↪': '10.1.1.1/24'})
```

17.18.1.3 Set Inventory Label

This endpoint is used to set labels for labeling flows and inventory items in a root scope on the Tetration appliance.

```
POST /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet.
attributes	JSON	Key/value map for labeling matching flows and inventory items

Response object:

Name	Type	Description
warnings	JSON	Key/value map containing warnings encountered while setting labels.

Sample python code

```
root_app_scope_name = 'Tetration'
req_payload = {'ip': '10.1.1.1/24', 'attributes': {'datacenter': 'SJC', 'location':
↪'CA'}}
restclient.post('/inventory/tags/%s' % root_app_scope_name, json_body=json.dumps(req_
↪payload))
```

17.18.1.4 Delete Inventory Label

This endpoint deletes labels for an IPv4/IPv6 address or subnet in a root scope on the Tetration appliance.

```
DELETE /openapi/v1/inventory/tags/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

The JSON query body contains the following keys

Name	Type	Description
ip	string	IPv4/IPv6 address or subnet

Sample python code

```
root_app_scope_name = 'Tetration'
req_payload = {'ip': '10.1.1.1/24'}
restclient.delete('/inventory/tags/%s' % root_app_scope_name, json_body=json.
↪dumps(req_payload))
```

17.18.1.5 Upload labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items in a root scope on the Tetration appliance. A column header with name `IP` must appear in the CSV file. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items. To use non-English characters in labels, the uploaded csv file must be in UTF-8 format.

```
POST /openapi/v1/assets/cmdb/upload/{rootAppScopeName}
```

Parameters:

User needs to provide an operation type (`X-Tetration-Oper`) as a parameter to this API. `X-Tetration-Oper` can be one of the following:

- add: Appends labels to new and existing addresses/subnets. Resolves conflicts by selecting new labels over existing ones. For example, if labels for an address in the database are *{"foo": "1", "bar": "2"}*, and the CSV file contains *{"z": "1", "bar": "3"}*, *add* sets labels for this address to *{"foo": "1", "z": "1", "bar": "3"}*.
- overwrite: inserts labels for new addresses/subnets and replaces labels for existing ones. For example, if labels for an address in the database are *{"foo": "1", "bar": "2"}* and the CSV file contains *{"z": "1", "bar": "3"}*, *overwrite* sets labels for this address to *{"z": "1", "bar": "3"}*.
- delete: removes labels for an address/subnet.

Response object:

Name	Type	Description
warnings	JSON	Key/value map containing warnings encountered while setting labels.

Sample python code

```
file_path = '/<path_to_file>/user_annotations.csv'
root_app_scope_name = 'Tetration'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload/%s' % root_app_scope_name, req_
↪payload)
```

17.18.1.6 Download user labels

This endpoint returns user uploaded labels for a root scope on the Tetration appliance as a CSV file.

```
GET /openapi/v1/assets/cmdb/download/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response:

Content-Type: *text/csv*

CSV file containing user uploaded labels for the scope.

Sample python code

```
file_path = '/<path_to_file>/output.csv'
root_app_scope_name = 'Tetration'
restclient.download(file_path, '/assets/cmdb/download/%s' % root_app_scope_name)
```

17.18.1.7 Get column headers

This endpoint returns a list of column headers for a root scope on the Tetration appliance.

```
GET /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: An array of facets available for a label.

Sample python code

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/attributenames/%s' % root_app_scope_name)
```

17.18.1.8 Delete column header

This endpoint deletes a column header in a root scope on the Tetration appliance. Deleting a column header drops it from the list of labelled facets and removes it from existing labels.

```
DELETE /openapi/v1/assets/cmdb/attributenames/{rootAppScopeName}/{attributeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.
attributeName	string	Attribute being deleted.

Response object: None

Sample python code

```
root_app_scope_name = 'Tetration'
attribute_name = 'column1'
resp = restclient.delete('/assets/cmdb/attributenames/%s/%s' % (root_app_scope_name,␣
↪attribute_name))
```

17.18.1.9 Get list of labelled facets

This endpoint returns a list of labelled facets for a root scope on the Tetration appliance. Labelled facets are a subset of column headers in the uploaded CSV file used for annotating flows and inventory items in that scope.

```
GET /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: An array of labelled facets for the root scope.

Sample python code

```
root_app_scope_name = 'Tetration'
resp = restclient.get('/assets/cmdb/annotations/%s' % root_app_scope_name)
```

17.18.1.10 Update list of labelled facets

This endpoint updates list of facets used for annotating flows and inventory items in a root scope on the Tetration appliance.

```
PUT /openapi/v1/assets/cmdb/annotations/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: None

Sample python code

```
# the following list is a subset of column headers in the
# uploaded CSV file
req_payload = ['location', 'region', 'detail']
root_app_scope_name = 'Tetration'
restclient.put('/assets/cmdb/annotations/%s' % root_app_scope_name,
               json_body=json.dumps(req_payload))
```

17.18.1.11 Flush user uploaded labels

This endpoint flushes labels for flows and inventory items in a root scope on the Tetration appliance. The changes affect new data; older labelled data remains unaltered.

```
POST /openapi/v1/assets/cmdb/flush/{rootAppScopeName}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
rootAppScopeName	string	Root scope name.

Response object: None

Sample python code

```
root_app_scope_name = 'Tetration'
restclient.post('/assets/cmdb/flush/%s' % root_app_scope_name)
```

The following APIs are available to users with read access to a scope, scope owners and site admins:

17.18.2 Scope independent APIs

The following APIs are only available to **site admins** and can span multiple scopes on the Tetration appliance.

17.18.2.1 Upload labels

This endpoint is used to upload a CSV file with labels for labeling flows and inventory items on the Tetration appliance. Column headers with names `IP` and `VRF` must appear in the CSV file and `VRF` should match the root scope for a label. Of the remaining column headers, up to 32 can be used to annotate flows and inventory items.

```
POST /openapi/v1/assets/cmdb/upload
```

Parameters:

User needs to provide an operation type (`X-Tetration-Oper`) as a parameter to this API to specify the *operation* to be performed.

Response object:

Name	Type	Description
warnings	JSON	Key/value map containing warnings encountered while setting labels.

Sample python code

```
file_path = '/<path_to_file>/user_annotations.csv'
req_payload = [tetpyclient.MultiPartOption(key='X-Tetration-Oper', val='add')]
restclient.upload(file_path, '/assets/cmdb/upload', req_payload)
```

17.18.2.2 Download user labels

This endpoint returns the user uploaded labels for all scopes on the Tetration appliance as a CSV file.

```
GET /openapi/v1/assets/cmdb/download
```

Parameters: None

Response:

Content-Type: *text/csv*

CSV file containing user uploaded labels for the scope.

Sample python code

```
file_path = '/<path_to_file>/output.csv'
restclient.download(file_path, '/assets/cmdb/download')
```

17.19 Virtual Routing and Forwarding (VRF)

This set of APIs manages VRFs.

Note: These APIs are only available to site admins.

17.19.1 VRF object

The VRF object attributes are described below:

Attribute	Type	Description
id	int	Unique identifier for the VRF.
name	string	User specified name of the VRF.
tenant_id	int	ID of parent tenant.
switch_vrfs	list of strings	List of switch vrf names that map to this Tetration VRF.
root_app_scope_id	string	ID of associated root scope.
created_at	integer	Unix timestamp when the VRF was created.
updated_at	integer	Unix timestamp when the VRF was last updated.

17.19.2 Get VRFs

This endpoints returns a list of VRFs. This API is available to API keys with `sensor_management`, `flow_inventory_query` or `hw_sensor_management` capability.

```
GET /openapi/v1/vrfs
```

Parameters: None

Response object: Returns a list of VRF objects.

Sample python code

```
resp = restclient.get('/vrfs')
```

17.19.3 Create a VRF

This endpoint is used to create new VRFs. An associated root scope will automatically be created with a query matching the VRF ID. This API is available to API keys with `sensor_management` capability.

```
POST /openapi/v1/vrfs
```

Parameters:

Name	Type	Descrption
id	int	(optional) Unique identifier for the VRF. If unspecified, Tetration cluster will generate a unique ID for the newly created VRF. Best practice is to let Tetration generate these IDs instead of caller explicitly specifying unique IDs.
tenant_id	int	(optional) ID of parent tenant.
name	string	User specified name of the VRF.
switch_vrfs	list of strings	(optional) List of switch vrf names that map to this Tetration VRF.
apply_monitoring_rules	boolean	(optional) Whether or not collection rules should be applied for the VRF. Defaults to 'false'. See *Collection Rules* for more information.

The `tenant_id` is optional. If not provided, the VRF will be added to the tenant with the same id as the VRF, auto-creating if necessary. If the `tenant_id` is provided, the tenant will not be auto created and an error will be returned if the tenant does not exist.

Response object: Returns the newly created VRF object.

Sample python code

```
req_payload = {
    "tenant_id": <tenant_id>,
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.post('/vrfs', json_body=json.dumps(req_payload))
```

17.19.4 Get specific VRF

This endpoints returns information for the specified vrf ID. This API is available to API keys with `sensor_management`, `flow_inventory_query` or `hw_sensor_management` capability.

```
GET /openapi/v1/vrfs/{vrf_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
vrf_id	int	Unique identifier for the VRF.

Response object: Returns a VRF object associated with specified ID.

Sample python code

```
vrf_id = 676767
resp = restclient.get('/vrfs/%d'% vrf_id)
```

17.19.5 Update a VRF

This endpoint updates a VRF. This API is available to API keys with `sensor_management` capability.

```
PUT /openapi/v1/vrfs/{vrf_id}
```

Parameters: The request URL contains the following parameters

Name	Type	Descrption
vrf_id	int	Unique identifier for the VRF.

The JSON request body contains the following parameters

Name	Type	Description
name	string	User specified name of the VRF.
switch_vrfs	list of strings	(optional) List of switch vrf names that map to this Tetration VRF.
apply_monitoring_rules	boolean	(optional) Whether or not collection rules should be applied to the VRF.

Response object: Returns the modified VRF object associated with specified ID.

Sample python code

```
vrf_id = 676767
req_payload = {
    "name": "Test",
    "apply_monitoring_rules": True
}
resp = restclient.put('/vrfs/%d'% vrf_id,
                      json_body=json.dumps(req_payload))
```

17.19.6 Delete specific VRF

This endpoint deletes a VRF. It will fail if there are is an associated root scope. This API is available to API keys with `sensor_management` capability.

```
DELETE /openapi/v1/vrfs/{vrf_id}
```

Parameters: The following parameter is part of the URL

Name	Type	Descrption
vrf_id	int	Unique identifier for the VRF.

Sample python code

```
vrf_id = 676767
resp = restclient.delete('/vrfs/%d'% vrf_id)
```

17.20 Orchestrators

This set of APIs can be used to manage external Orchestrator inventory learning in Tetration cluster deployment. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator types are 'vcenter' (VCenter 6.5 and later), 'aws', 'kubernetes', 'dns', 'f5', 'netscaler' and 'infoblox'. Supported user interface located at *External Orchestrators*.

17.20.1 Orchestrator Object

The orchestrator object attributes are described below - some of the fields are applicable only for specific orchestrator types; restrictions are mentioned in the table below.

17.20.2 Ingress Controller

Attribute	Type	Description
pod_selector	object	*Pod Selector*
controller_config	object	*Controller Config*

17.20.3 Pod Selector

Attribute	Type	Description
namespace	string	Namespace where the Ingress controller pod is running.
labels	Array	Array of {"key", "value"} pairs that specify the labels of ingress controller pods.

17.20.4 Controller Config

Attribute	Type	Description
ingress_class	string	Name of the ingress class which ingress controller satisifies.
namespace	string	Namespace is the name of the namespace which ingress controller satisfies.
http_ports	Array	Array of http ports.
https_ports	Array	Array of https ports.

** Read-only status fields in the Orchestrator object **

Attribute	Type	Description
authentication_failure	bool	Status of the connection to the Tetration Orchestrator - *true* indicates a successful connection to the orchestrator. If this field is *false*, the *authentication_failure_error* field will provide a detailed error message explaining the reason for the failure
authentication_failure_error	string	Detailed error message to help debug connectivity or credential failures with orchestrators
scope_id	string	Tenant Root scope id where the inventory will be published and visible

17.20.5 Get orchestrators

This endpoint returns a list of orchestrators known to Tetration appliance. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}
```

Parameters: None

Returns a list of orchestrator objects for the provided root scope. The *scope* MUST be a root scope id.

17.20.6 Create a orchestrator

This endpoint is used to create new orchestrators.

```
POST /openapi/v1/orchestrator/{scope}
```

Sample python code for VCenter orchestrators

```
req_payload = {
    "name": "VCenter Orchestrator"
    "type": "vcenter",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 443}],
    "username":"admin",
    "password":"admin"
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for DNS orchestrators

```
req_payload = {
    "name": "DNS Server"
    "type": "dns",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "dns_zones": [ "lab.corp.com.", "dev.corp.com." ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators

```
req_payload = {
    "name": "k8s"
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators with Ingress Controller

Please refer external_orchestrator_k8s for creating authentication details.

```
req_payload = {
    "name": "k8s"
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
    "ingress_controllers": [
        {
            "pod_selector": {
```

(continues on next page)

(continued from previous page)

```
                "namespace": "ingress-nginx",
                "labels": [{ "key": "app", "value": "nginx-ingress"}],
            }
        }
    ]
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

Sample python code for Kubernetes orchestrators with Multiple Ingress Controllers

Please refer external_orchestrator_k8s for creating authentication details.

```
req_payload = {
    "name": "k8s"
    "type": "kubernetes",
    "hosts_list": [ { "host_name": "8.8.8.8", "port_number": 53}],
    "certificate": "",
    "key": "",
    "ca_certificate": "",
    "ingress_controllers": [
        {
            "pod_selector": {
                "namespace": "ingress-nginx",
                "labels": [{ "key": "app", "value": "nginx-ingress"}],
            },
            "controller_config": {
                "ingress_class": "nginx-class",
            }
        },
        {
            "pod_selector": {
                "namespace": "ingress-haproxy",
                "labels": [{ "key": "app", "value": "haproxy-ingress"}],
            },
            "controller_config": {
                "ingress_class": "haproxy-class",
                "http_ports": [8080],
                "https_ports": [8443],
                "namespace": "haproxy-watching-namespace"
            }
        }
    ],
}
resp = restclient.post('/orchestrator/Default', json_body=json.dumps(req_payload))
```

17.20.7 Get specific orchestrator

This endpoint returns an instance of a orchestrator.

```
GET /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Returns the orchestrator object associated with the specified ID.

17.20.8 Update an orchestrator

This endpoint updates a orchestrator.

```
PUT /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

Parameters:

Same as POST parameters

17.20.9 Delete specific orchestrator

This endpoint deletes the specified orchestrator.

```
DELETE /openapi/v1/orchestrator/{scope}/{orchestrator_id}
```

17.21 Orchestrator Golden Rules

This set of APIs can be used to manage Golden Rules for external Kubernetes Orchestrators. Golden Rules are necessary to ensure Kubernetes control plane connectivity in allow list enforcement mode. They require the `external_integration` capability associated with the API key.

Currently supported Orchestrator type for Golden Rules is 'kubernetes' only. Requests to this endpoint for non-Kubernetes orchestrators will fail.

17.21.1 Orchestrator Golden Rules object

The orchestrator object attributes are described below:

Attribute	Type	Description
kubelet_port	integer	Kubelet node-local API port
services	Array	Array of Kubernetes Services objects

17.21.2 Get orchestrator golden rules

This endpoint returns the golden rules associated with an orchestrator. This API is available to API keys with the `external_integration` capability.

```
GET /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters: None

Returns a single Golden Rules object

17.21.3 Create/Update Golden Rules

This endpoint is used to create or update golden rules for an existing orchestrator.

```
POST /openapi/v1/orchestrator/{scope}/{id}/gr
```

Parameters:

Attribute	Type	Description
kubelet_port	integer	Kubelet node-local API port
services	Array	Array of Kubernetes Services objects

Sample python code

```
req_payload = {
            "kubelet_port":10255,
            "services": [
                      {
                              "description": "kube-dns",
                              "addresses":   [ "10.0.1.1:53/TCP", "10.0.1.1:53/UDP" ],
                              "consumed_by":  [ "NODES", "PODS"],
                      }
           ]
}
resp = restclient.post('/orchestrator/{scope_id}/{orchestrator_id}/gr', json_
↪body=json.dumps(req_payload))
```

17.22 RBAC (Role Based Access Control) Considerations

Access to orchestrators under a root scope requires that the API Key used for the request has the requisite privileges. All orchestrator API calls are scoped and always require the root scope id as part of the URL. Orchestrators always reside at the root scope level and cannot be created under sub-scopes. Orchestrators created (and inventory learnt by these orchestrators) under a specific tenant root scope are invisible to other tenants.

In the case of F5 load balancers that may have multiple route domains (vrfs) configured, the F5 Route Domain filtering logic will scan all entities on the F5 across all partitions but discard entities (services, snat pools, pools and backends) that do not evaluate to the route domain specified in the F5 orchestrator *route_domain* field.

17.23 High Availability and Failover Considerations

The hosts_list parameter allows configuration of multiple server addresses for an orchestrator. Tetration server selection logic in the case of multiple server addresses varies for each orchestrator type.

For *Vcenter, Kubernetes, DNS, F5, Netscaler,Infoblox*, the selection is on a first healthy endpoint basis. Connections are not persistent (except for *kubernetes*) and thus, every poll period, Tetration Orchestrator Manager will scan the hosts and poll the first healthy endpoint encountered in the hosts_list. For *kubernetes*, a persistent event channel is maintained and upon connection failure, a scan of all hosts and subsequent full poll will be performed using the next healthy endpoint.

For *aws*, the hosts_list is ignored since the AWS SDK has its own robust REST endpoint selection logic.

17.24 Kubernetes RBAC Resource Considerations

The Kubernetes client attempts to GET/LIST/WATCH the following resources.

The provided Kubernetes authentication credentials should have a minimum set of privileges to the following resources:

Resources	Verbs
daemonsets	[get list watch]
deployments	[get list watch]
endpoints	[get list watch]
namespaces	[get list watch]
nodes	[get list watch]
pods	[get list watch]
replicasets	[get list watch]
replicationcontrollers	[get list watch]
services	[get list watch]
statefulsets	[get list watch]
daemonsets.apps	[get list watch]
deployments.apps	[get list watch]
endpoints.apps	[get list watch]
namespaces.apps	[get list watch]
nodes.apps	[get list watch]
pods.apps	[get list watch]
replicasets.apps	[get list watch]
replicationcontrollers.apps	[get list watch]
services.apps	[get list watch]
statefulsets.apps	[get list watch]
daemonsets.extensions	[get list watch]
deployments.extensions	[get list watch]
endpoints.extensions	[get list watch]
namespaces.extensions	[get list watch]
nodes.extensions	[get list watch]
pods.extensions	[get list watch]
replicasets.extensions	[get list watch]
replicationcontrollers.extensions	[get list watch]
services.extensions	[get list watch]
statefulsets.extensions	[get list watch]

17.25 Site Infos

This API can be used to get cluster information such as cluster state, cluster type, external IPs, and emails.

Note: This API is only available to site admin users.

17.25.1 Get site infos

This endpoint returns a JSON object with cluster site infos information.

```
GET /openapi/v1/site_infos
```

Parameters: None

Response object: JSON object with cluster site infos information

Sample Python code

```
resp = restclient.get('/site_infos')
```

Sample response

```
{
  "cluster_state": "Enabled till 2020-12-31 23:59:59 UTC",
  "cluster_uuid": "00000000-0000-0000-0000-000000000000",
  "site_bosun_email": "customer-support@company.com",
  "site_cluster_type": "physical",
  "site_external_ips": [
      "1.1.1.1",
      "1.1.1.2",
      ...
      "1.1.1.7"
  ],
  "site_name": "cluster_name",
  "site_sensor_vip_ip": "2.1.1.1",
  "site_ui_admin_email": "site-admin@company.com",
  "site_ui_fqdn": "cluster.company.com",
  "site_ui_primary_customer_support_email": "customer-support@company.com"
}
```

17.26 Cluster Health

This API can be used to get status of all the physical servers in Cisco Tetration.

Note: This API is only available to site admin users.

17.26.1 Get Cluster Health

This endpoint returns a JSON object with cluster health information.

```
GET /openapi/v1/cluster_nodes
```

Parameters: None

Response object: JSON object with cluster health information

Sample Python code

```
resp = restclient.get('/cluster_nodes')
```

17.27 Service Health

This API can be used to get the health of all services that are used in Cisco Tetration cluster along with their dependencies..

Note: This API is only available to site admin users.

17.27.1 Get Service Health

This endpoint returns a JSON object with service health information.

```
GET /openapi/v1/service_status
```

Parameters: None

Response object: JSON object with service health information

Sample Python code

```
resp = restclient.get('/service_status')
```

17.28 Secure Connector

OpenAPI exposes the endpoints to manage the functions of the *Tetration Secure Connector*. These endpoints require the `external_integration` capability to be associated with the API key.

Note: The Secure Connector APIs cannot be used at site level. They can only be used at the root scope level.

17.28.1 Get Status

This endpoint returns the current status of the Secure Connector Tunnel for the specified root scope.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/status
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/status
```

READ permission to the specified root scope is required.

The returned status is a json object with the following schema:

Key	Type	Value
active	boolean	A Secure Connector tunnel is currently active
peer	string	<ip>:<port> of the Secure Connector client end of the tunnel
start_time	int	Timestamp at which the tunnel was started (epoch time in seconds)
last_heartbeat	int	Timestamp of last heartbeat from the client (epoch time in seconds)

17.28.2 Get Token

This endpoint returns a new single-use limited-time token to be used for bootstrapping a Secure Connector client for the specified root scope.

```
GET /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/token
GET /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/token
```

OWNER permission to the specified root scope is required.

The returned token is a string which contains a cryptographically signed token that is valid for one hour. A valid token can be used only once to bootstrap a Secure Connector client.

17.28.3 Rotate Certificates

This endpoint forces the creation of a new certificate for the specified root scope. The new certificate will be used by the Secure Connector server and will be used to sign the certificate signing requests from clients for this root scope.

```
POST /openapi/v1/secureconnector/name/{ROOT_SCOPE_NAME}/rotate_certs?invalidate_old=
↪{true|false}
POST /openapi/v1/secureconnector/{ROOT_SCOPE_ID}/rotate_certs?invalidate_old=
↪{true|false}
```

OWNER permission to the specified root scope is required.

Once this endpoint is called, communication between the client and server for this root scope will immediately transition to using the new certificate.

If *invalidate_old* is set to false, any existing clients will automatically create a new public/private key pair and use their existing certificates to sign a new certificate for the new public key.

If *invalidate_old* is set to true, the existing certificate will be immediately invalidated. Any existing clients will not be able to connect to the server and will have to be bootstrapped once again using a new token. See *Secure Connector Deployment* for more information.

17.29 Policy Enforcement Status for external orchestrators

This set of APIs is used to provide policy enforcement status for load balancer external orchestrators such as *F5 BIG-IP* or *Citrix Netscaler*.

Note: In order to use these APIs, user should have access to the scope attached to the VRF.

17.29.1 Get policy enforcement status for all external orchestrators

This endpoint returns policy enforcement status for all external orchestrators belonging to the given VRF.

This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/tnp_policy_status/{vrfID}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrfID	integer	VRF ID for the root scope.

Response object: Returns a list of network policies with the Status as `ENFORCED` or `FAILED` or `IGNORED`.

Sample python code

```
vrf_id = 676767
restclient.get('/tnp_policy_status/%d' % vrf_id)
```

17.29.2 Get policy enforcement status for an external orchestrator

This endpoint returns policy enforcement status for an external orchestrator belonging to the given VRF.

This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/tnp_policy_status/{vrfID}/{orchestratorID}
```

Parameters: The request URL contains the following parameters

Name	Type	Description
vrfID	integer	VRF ID for the root scope.
orchestratorID	string	External orchestrator ID.

Response object: Returns a list of network policies with the Status as `ENFORCED` or `FAILED` or `IGNORED`.

Sample python code

```
vrf_id = 676767
orchestrator_id = '5ee3c991497d4f3b00f1ee07'
restclient.get('/tnp_policy_status/%d/%s' % (vrf_id, orchestrator_id))
```

17.30 Download Certificates for Managed Data Taps and Datasinks

This set of APIs is used to download the certificates for the Managed Data Taps and Datasinks.

Note: In order to use these APIs, user should have access to the scope attached to the VRF.

17.30.1 Get List of Managed Data Taps for a given VRF ID.

This endpoints returns a list of Managed Data Taps in a given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/mdt/{vrfID}
```

Parameters: None

Returns a list of Managed Data Taps with attributes like Managed Data Tap ID.

17.30.2 Download Managed Data Tap certificates for a given MDT ID.

This endpoint is used to download the certificates for a given Managed Data Tap ID. The MDT ID can be obtained by using **/openapi/v1/mdt/{vrfID}** endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/mdt/{vrfID}/{mdtID}/certs
```

Parameters: None

Returns a tar.gz file which contains the following files:- **KafkaConsumerCA.cert**, **KafkaConsumerPrivateKey.key**, **kafkaCA.cert**, **kafkaBrokerIps.txt**, **topic.txt**.

KafkaConsumerCA.cert is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **kafkaCA.cert** has the CA certificate and **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. **topic.txt** file has the name of the topic which should be used to fetch data from MDT.

17.30.3 Get List of DataSinks for a given VRF ID.

This endpoints returns a list of DataSinks in a given VRF. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}
```

Parameters: None

Returns a list of DataSinks with attributes like DataSink ID.

17.30.4 Download DataSink certificates for a given DataSink ID.

This endpoint is used to download the certificates for a given DataSink ID. The DataSink ID can be obtained by using **/openapi/v1/datasinks/{vrfID}** endpoint as explained in the above documentation. This API is available to API keys with `external_integration` capability.

```
GET /openapi/v1/datasinks/{vrfID}/{dsID}/certs
```

Parameters: None

Returns a tar.gz file which contains the following files:- **userCA.cert**, **userPrivateKey.key**, **intermediateCA.cert**, **kafkaCA.cert**, **kafkaBrokerIps.txt**, **topic.txt**.

userCA.cert is the Public certificate file and **KafkaConsumerPrivateKey.key** file has the private key. **intermediateCA.cert** and **kafkaCA.cert** has the CA certificate for intermediate and root CA respectively. **kafkaBrokerIps.txt** has the list of the Kafka brokers IP Addresses and Ports. **topic.txt** file has the name of the topic which should be used to fetch data from datasink.

17.31 Change Logs

This API provides read access to change log items. This API requires the `user_role_scope_management` capability associated with the API key.

Note: This API is only available to site admins and owners of root scopes.

17.31.1 Change log object

The change log object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the change log item.
association_chain	array of objects	List of names and ids associated with this change.
scope	string	Scope of change (not the same as a Tetration scope).
action	string	Change action.
details	string	Further action details, when available.
created_at	integer	Unix timestamp of when change log item was created.
modifier	object	User responsible for change.
modified	object	Modified fields and values.
original	object	Fields and values before modification.
version	integer	Version identifier.

17.31.2 Search

This endpoint returns the list of change log items matching the specified criteria.

```
GET /openapi/v1/change_logs
```

Parameters: The request URL contains the following parameters

Name	Type	Description
root_app_scope_id	string	(optional) Required for root scope owners. Filter results by root scope.
association_name	string	(optional) Required for root scope owners. The item type to return. For example: "H4Users"
history_action	string	(optional) Change action. For example: "update"
details	string	(optional) Action details. For example: "soft-delete"
before_epoch	integer	(optional) Include results created before this unix timestamp.
after_epoch	integer	(optional) Include results created after this unix timestamp.
offset	integer	(optional) Number of results to skip.
limit	integer	(optional) Limit number of results.

Response object: Returns a list of change log objects.

Response

The response is a JSON object in the body with the following properties.

Name	Type	Description
total_count	integer	Total number of items matching before applying offset or limit.
items	array of objects	List of results.

Sample python code

Fetch last 100 scope object changes within a given root scope within the last day.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24*60*60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

Further refine these results to only show new scope creations.

```
root_app_scope_id = '5ce480db497d4f1ca1fc2b2b'
one_day_ago = int(time.time() - 24*60*60)
resp = restclient.get('/change_logs', params={'root_app_scope_id': root_app_scope_id,
                                              'association_name': 'AppScope',
                                              'history_action': 'create',
                                              'after_epoch': one_day_ago,
                                              'limit': 100})
```

A site admin could use limit and offset to iteratively fetch all changes across all scopes.

```
resp = restclient.get('/change_logs', params={'offset': 100, 'limit': 100})
```

17.32 Non Routable Endpoints

This set of API is used to manage Non Routable Endpoints, to mark an ip/subnet as non routable or get a list of non routable endpoints that are marked by an user or to unmark an ip/subent as non routbale endpoint. They require `user_data_upload` capability associated with the API key.

17.32.1 Non Routable Endpoint Object

The Non Routable Endpoint Object attributes are described below:

Attribute	Type	Description
id	string	Unique identifier for the non routable endpoint.
name	string	User specified name of the non routable endpoint.
subnet	string	IPv4/IPv6 subnet.
vrf_id	long	ID of the VRF to which non routable endpoint belongs to.
address_type	string	IPV4/IPV6 based upon subnet address type
host_uuid	string	Unique ID of the agent
description.	string	User specified description of the non routable endpoint.

17.32.2 GET non routable endpoints

This endpoint returns a list of non routable endpoints in the given tenant.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters: None

17.32.3 Create a non routable endpoint

This endpoint is used to create a non routable endpoint.

```
POST /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

Attribute	Type	Description
name	string	User specified name of the non routable endpoint.
subnet	string	IPv4/IPv6 subnet.
address_type(optional)	string	IPV4/IPV6 based upon subnet address type
host_uuid(optional)	string	Unique ID of the agent
description(optional)	string	User specified description of the non routable endpoint.

*if optional fields are not specified, null values will get populated

Sample python code

```
req_payload = {
  "name": "nre-1",
  "subnet": "1.1.1.1/30",
  "address_type": IPV4,
  "description": "sample parameters test"
}
resp = restclient.post('/openapi/v1/non_routable_endpoints/Default', json_body=json.
↪dumps(req_payload))
```

17.32.4 GET specific non routable endpoints with name

This endpoint returns a non routable endpoint for the specified name.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

Parameters: None

17.32.5 GET specific non routable endpoints with id

This endpoint returns a non routable endpoint for the specified id.

```
GET /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

Parameters: None

17.32.6 Update specific non routable endpoint's name

This endpoint is used to update a non routable endpoint. It uses either id or name of the existing non routable endpoint to update its name.

```
PUT /openapi/v1/non_routable_endpoints/{rootScopeName}
```

Parameters:

Attribute	Type	Description
id	string	Unique identifier for the non routable endpoint.
name	string	User specified name of the non routable endpoint.
new_name	string	new name to update

Sample python code

```
req_payload = {
  "name": "nre-1",
  "new_name": "nre-updated",
}
resp = restclient.put('/openapi/v1/non_routable_endpoints/Default', json_body=json.
↪dumps(req_payload))


req_payload = {
  "id": "5f706964a5b5f16ed4b0aacb",
  "new_name": "nre-updated",
}
resp = restclient.put('/openapi/v1/non_routable_endpoints/Default', json_body=json.
↪dumps(req_payload))
```

17.32.7 DELETE specific non routable endpoint with name

This endpoint deletes the specific non routable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/name/{name}
```

17.32.8 DELETE specific non routable endpoint with name

This endpoint deletes the specific non routable endpoint.

```
DELETE /openapi/v1/non_routable_endpoints/{rootScopeName}/id/{id}
```

CHAPTER

EIGHTEEN

CONNECTORS

18.1 What are Connectors

Connectors are integrations that Tetration supports for a variety of use cases, including flow ingestion, inventory enrichment and alert notifications. Please refer *List of connectors* supported in Tetration. Some of the connectors are agents that ingest flow observations to Tetration through standard protocols such as NetFlow v9 and IPFIX. Examples of such connectors are NetFlow, Citrix NetScaler, F5 BIG-IP, and AnyConnect. And, some of the connectors are alert notifiers. Examples of such connectors include Slack, Email, Syslog, PagerDuty and Kinesis. Prior to 3.3.1.x release, these connectors are deployed on specific appliances. For example, NetFlow agent is deployed on NetFlow appliance. And, all alert notifiers are deployed on Tetration Alert Notifier appliance. In 3.3.1.x release, connectors are enabled and managed (including configuration management) directly through Tetration. Each connector is enabled on one of three types of virtual appliances, namely: (1) *Tetration Ingest*, (2) *Tetration Edge*, and (3) *Tetration Export*. Please refer to the *Tetration Virtual Appliances for Connectors* for more information on appliances.

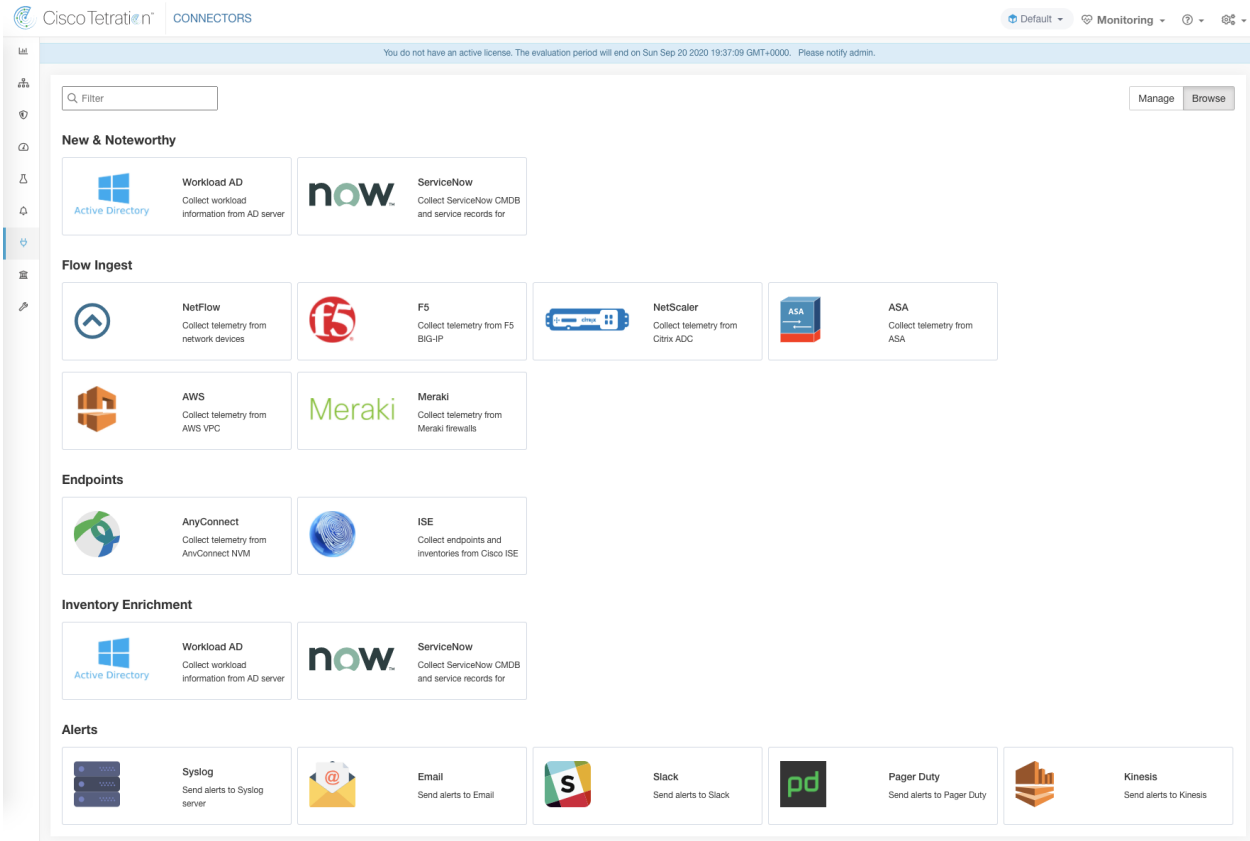


Fig. 18.1.1: List of connectors

18.1.1 Connectors for Flow Ingestion

Connectors for flow ingestion stream flow observations from different Network switches, routers, and other middleboxes (such as load balancers and firewalls) to Tetration. Tetration supports flow ingestion through NetFlow v9, IPFIX and custom protocols. In addition to flow observations, middlebox connectors stitch client-side and server-side flows, in order to understand which client flows are related to which server flows. Furthermore, flow ingestion from other sources such AWS VPC flow logs is also supported.

Connector	Description	Deployed on Virtual Appliance
NetFlow	Collect NetFlow V9 and/or IP-FIX telemetry from network devices such as routers and switches.	Tetration Ingest
F5 BIG-IP	Collect telemetry from F5 BIG-IP, stitch client and server side flows, enrich client inventory with user attributes.	Tetration Ingest
Citrix NetScaler	Collect telemetry from Citrix ADC, stitch client and server side flows.	Tetration Ingest
ASA	Collect telemetry data from Cisco ASA, stitch client and server side flows.	Tetration Ingest
AWS	Collect telemetry data from AWS for the configured VPC.	Tetration Ingest
Meraki	Collect telemetry data from Meraki firewalls.	Tetration Ingest

18.1.1.1 NetFlow Connector

NetFlow connector allows Tetration to ingest flow observations from routers and switches in the network. Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay NetFlow records to NetFlow connector hosted in a Tetration Ingest appliance for processing.

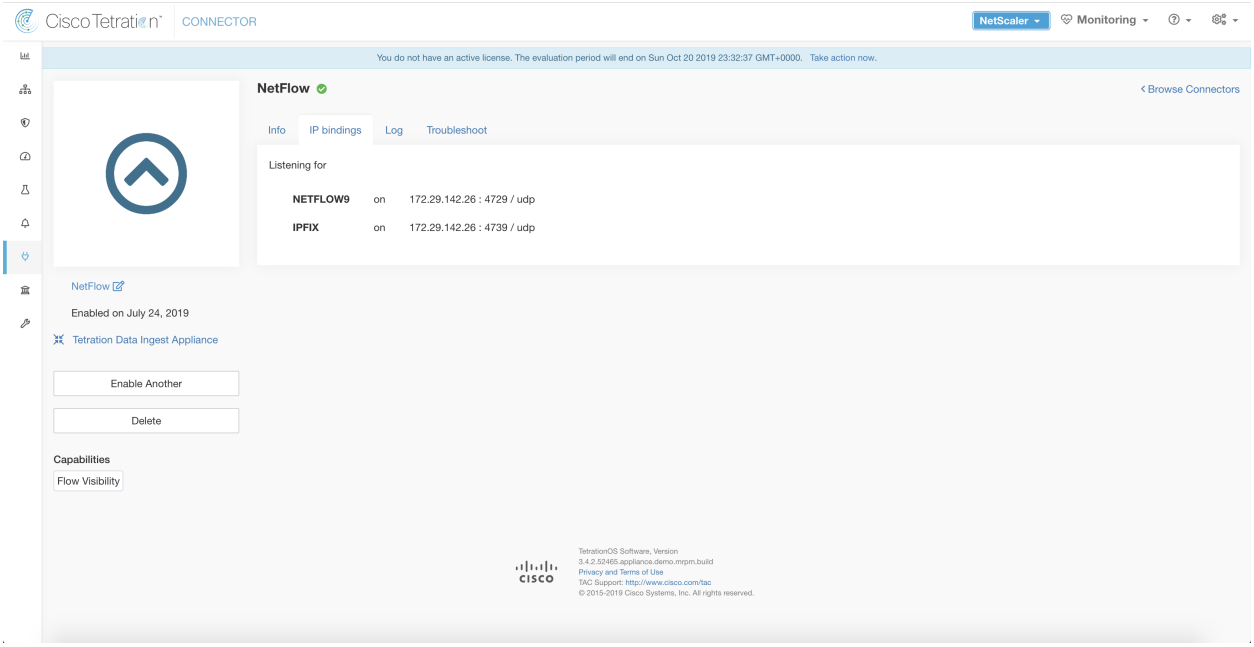


Fig. 18.1.1.1.1: NetFlow connector

What is NetFlow

NetFlow protocol allows routers and switches to aggregate traffic that passes through them into flows and export these flows to a flow collector. The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis. NetFlow is supported in most Cisco routers and switches.

Typically, the setup involves the following steps:

1. Enable NetFlow feature on one or more network devices and configure the flow templates that devices should export.
2. Configure the NetFlow collector endpoint information on the remote network devices. This NetFlow collector will be listening on configured endpoint to receive and process NetFlow flow records.

Flow Ingestion to Tetration

NetFlow connector is essentially a NetFlow collector. The connector receives the flow records from the network devices and forwards them to Tetration for flow analysis. A NetFlow connector can be enabled on a Tetration Ingest appliance and runs as a Docker container.

NetFlow connector also registers with Tetration as a Tetration NetFlow agent. NetFlow connector decapsulates the NetFlow protocol packets (i.e., flow records); then processes and reports the flows like a regular Tetration agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.

Note: NetFlow connector supports NetFlow v9 and IPFIX protocols.

Note: Each NetFlow connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

Rate Limiting

NetFlow connector accepts up to 15000 flows per second. Note that a given NetFlow v9 or IPFIX packet could contain one or more flow and template records. NetFlow connector parses the packets and identifies the flows. If the connector parses more than 15000 flows per second, it will drop the additional flow records.

Please also note that Tetration customer support will support NetFlow connector only if the flow rate is within this acceptable limit. If ever the flow rate is higher than 15000 flows per second, first, we recommend adjusting the flow rate to fall within the limits and stay at this level for at least 3 days (to rule out issues related to higher incoming flow rate). If the original issue persists then customer support will start investigating the issue and identify proper workaround and/or solution.

Supported Information Elements

NetFlow connector *only* supports the following information elements in NetFlow v9 and IPFIX protocols. For more information about these elements, please refer to IP Flow Information Export (IPFIX) Entities document.

Element ID	Name	Description	Mandatory
1	octetDeltaCount	Number of octets in incoming packets for this flow.	Yes
2	packetDeltaCount	Number of incoming packets for this flow.	Yes
4	protocolIdentifier	The value of the protocol number in the IP packet header.	Yes
6	tcpControlBits	TCP control bits observed for packets of this flow. Only FIN, SYN, RST, PSH, ACK, and URG flags are handled by the agent.	No
7	sourceTransportPort	The source port identifier in the transport header.	Yes
8	sourceIPv4Address	The IPv4 source address in the IP packet header.	Either 8 or 27
11	destinationTransportPort	The destination port identifier in the transport header.	Yes
12	destinationIPv4Address	The IPv4 destination address in the IP packet header.	Either 12 or 28
27	sourceIPv6Address	The IPv6 source address in the IP packet header.	Either 8 or 27
28	destinationIPv6Address	The IPv6 destination address in the IP packet header.	Either 12 or 28
150	flowStartSeconds	The absolute timestamp of the first packet of the flow (in seconds).	No
151	flowEndSeconds	The absolute timestamp of the last packet of the flow (in seconds).	No
152	flowStartMilliseconds	The absolute timestamp of the first packet of the flow (in milliseconds).	No
153	flowEndMilliseconds	The absolute timestamp of the last packet of the flow (in milliseconds).	No
154	flowStartMicroseconds	The absolute timestamp of the first packet of the flow (in microseconds).	No
155	flowEndMicroseconds	The absolute timestamp of the last packet of the flow (in microseconds).	No
156	flowStartNanoseconds	The absolute timestamp of the first packet of the flow (in nanoseconds).	No
157	flowEndNanoseconds	The absolute timestamp of the last packet of the flow (in nanoseconds).	No

How to configure NetFlow on the Switch

The following steps are for a Nexus 9000 switch. The configurations may slightly differ for other Cisco platforms. In any case, please also refer to the official Cisco configuration guide for the Cisco platform you are configuring.

Step 1: Enter global configuration mode.

```
switch# configure terminal
```

Step 2: Enable NetFlow feature.

```
switch(config)# feature netflow
```

Step 3: Configure a flow record.

The following example configuration shows how to generate 5 tuple information of a flow in a NetFlow record.

```
switch(config)# flow record ipv4-records
switch(config-flow-record)# description IPv4Flow
switch(config-flow-record)# match ipv4 source address
switch(config-flow-record)# match ipv4 destination address
switch(config-flow-record)# match ip protocol
switch(config-flow-record)# match transport source-port
switch(config-flow-record)# match transport destination-port
switch(config-flow-record)# collect transport tcp flags
switch(config-flow-record)# collect counter bytes
switch(config-flow-record)# collect counter packets
```

Step 4: Configure a flow exporter.

The following example configuration specifies the NetFlow protocol version, NetFlow template exchange interval, and NetFlow collector endpoint details. Please specify the IP and port on which NetFlow connector is enabled on a Tetration Ingest appliance.

```
switch(config)# flow exporter flow-exporter-one
switch(config-flow-exporter)# description NetFlowv9ToNetFlowConnector
switch(config-flow-exporter)# destination 172.26.230.173 use-vrf management
switch(config-flow-exporter)# transport udp 4729
switch(config-flow-exporter)# source mgmt0
switch(config-flow-exporter)# version 9
switch(config-flow-exporter-version-9)# template data timeout 20
```

Step 5: Configure a flow monitor.

Create a flow monitor and associate it with a flow record and flow exporter.

```
switch(config)# flow monitor ipv4-monitor
switch(config-flow-monitor)# description IPv4FlowMonitor
switch(config-flow-monitor)# record ipv4-records
switch(config-flow-monitor)# exporter flow-exporter-one
```

Step 6: Apply the flow monitor to an interface.

```
switch(config)# interface Ethernet 1/1
switch(config-if)# ip flow monitor ipv4-monitor input
```

The above steps configure NetFlow on Nexus 9000 to export NetFlow v9 protocol packets for ingress traffic going through interface 1/1. The flow records will be sent to 172.26.230.173:4729 over UDP protocol. Each flow record includes 5 tuple information of the traffic and the byte/packet count of the flow.

The following screenshot shows running configuration of NetFlow on a Nexus 9000 switch.

```
switch# show running-config netflow

!Command: show running-config netflow
!Time: Wed Mar 21 04:25:21 2018

version 7.0(3)I7(1)
feature netflow

flow timeout 60
flow exporter flow-exporter-173
  destination 172.26.230.173 use-vrf management
  transport udp 4729
  source mgmt0
  version 9
    template data timeout 20
flow record ipv4-records
  match ipv4 source address
  match ipv4 destination address
  match ip protocol
  match transport source-port
  match transport destination-port
  collect transport tcp flags
  collect counter bytes
  collect counter packets
  collect timestamp sys-uptime first
  collect timestamp sys-uptime last
flow monitor ipv4-monitor
  record ipv4-records
  exporter flow-exporter-173


interface Ethernet1/1
  ip flow monitor ipv4-monitor input

interface Ethernet1/2
  ip flow monitor ipv4-monitor input

switch#
```

Fig. 18.1.1.1.2: Running configuration of NetFlow on Cisco Nexus 9000 Switch

How to Configure the Connector

The following configurations are allowed on the connector.

- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of NetFlow v9 and IPFIX protocols on the connector can be updated on the Docker container in Tetration Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of NetFlow connectors on one Tetration Ingest appliance	3
Maximum number of NetFlow connectors on one Tenant (rootscope)	10
Maximum number of NetFlow connectors on Tetration	100

18.1.1.2 F5 Connector

F5 connector allows Tetration to ingest flow observations from F5 BIG-IP ADCs. It allows Tetration to remotely monitor flow observations on F5 BIG-IP ADCs, and stitch client-side and server-side flows, and annotate users on the client IPs (if user information is available). Using this solution, the hosts do not need to run software agents, because F5 BIG-IP ADCs will be configured to export IPFIX records to F5 connector for processing.

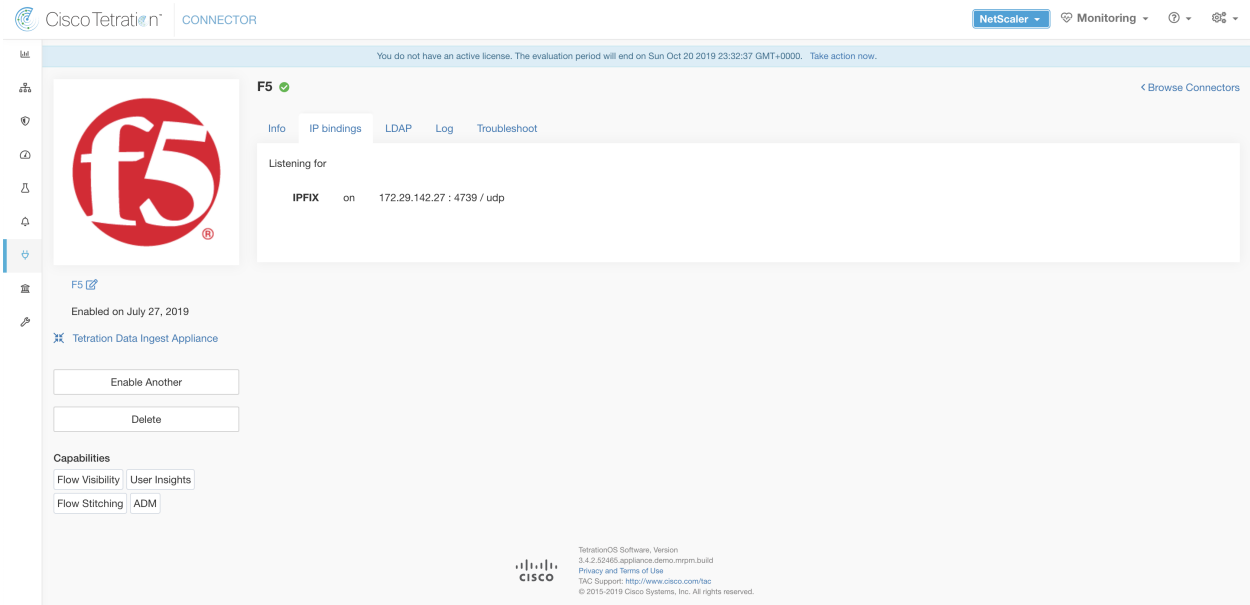


Fig. 18.1.1.2.1: F5 connector

What is F5 BIG-IP IPFIX

F5 BIG-IP IPFIX logging collects flow data for traffic going through the F5 BIG-IP and exports IPFIX records to flow collectors.

Typically, the setup involves the following steps:

1. Create IPFIX Log-Publisher on F5 BIG-IP appliance.
2. Configure the IPFIX Log-Destination on the F5 BIG-IP appliance. This log-destination will be listening on configured endpoint to receive and process flow records.
3. Create an F5 iRule that publishes IPFIX flow records to the log-publisher.
4. Add the F5 iRule to the virtual server of interest.

Note: F5 connector supports F5 BIG-IP software version 12.1.2 and above.

Flow Ingestion to Tetration

F5 BIG-IP connector is essentially an IPFIX collector. The connector receives the flow records from F5 BIG-IP ADCs, stitch the NATed flows and forwards them to Tetration for flow analysis. In addition, if LDAP configuration is provided to F5 connector, it determines values for configured LDAP attributes of user associated with the transaction (if F5 authenticates the user before processing the transaction). The attributes are associated to the client IP address where the flow happened.

Note: F5 connector supports only IPFIX protocol.

Note: Each F5 connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

How to configure IPFIX on F5 BIG-IP

The following steps are for F5 BIG-IP load balancer. (Ref: Configuring F5 BIG-IP for IPFIX)

Purpose	Description
1. Create a pool of IPFIX collectors	On F5 BIG-IP appliance, create the pool of IPFIX collectors. These are the IP addresses associated with F5 connectors on a Tetration Ingest appliance. F5 connectors run in Docker containers on the VM listen on port 4739 for IPFIX packets.
2. Create a log-destination.	The log destination configuration on F5 BIG-IP appliance specifies the actual pool of IPFIX collectors that should be used.
3. Create a log-publisher.	A log publisher specifies where F5 BIG-IP sends the IPFIX messages. The publisher is bound with a log-destination.
4. Add a F5 and Tetration approved iRule	Tetration and F5 developed iRules that will export flow records to F5 connectors. These iRules will export complete information about a given transaction: including all the endpoints, byte and packet counts, flow start and end time (in milliseconds). F5 connectors will create 4 independent flows and match each flow with its related flow.
5. Add the iRule to the virtual server.	In the iRule settings of a virtual server, add the Tetration. approved iRule to the virtual server.

The above steps configures IPFIX on F5 BIG-IP load balancer to export IPFIX protocol packets for traffic going through the appliance. Here is a sample config of F5.

```
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmos)# show running-config ltm virtual vip-1 rules
ltm virtual vip-1 {
    rules {
        ipfix-rule-1
    }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmos)# show running-config ltm  pool ipfix-pool-1
ltm pool ipfix-pool-1 {
    members {
        10.28.118.6:ipfix {
            address 10.28.118.6
            session monitor-enabled
            state up
        }
    }
    monitor gateway_icmp
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmos)# show running-config ltm virtual vip-1 rules
ltm virtual vip-1 {
    rules {
        ipfix-rule-1
    }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmos)# show running-config sys  log-config
sys log-config destination ipfix ipfix-collector-1 {
    pool-name ipfix-pool-1
    transport-profile udp
}
sys log-config publisher ipfix-pub-1 {
    destinations {
        ipfix-collector-1 { }
    }
}
root@(localhost)(cfg-sync Standalone)(Active)(/Common)(tmos)#
```

Fig. 18.1.1.2.2: Running configuration of IPFIX on F5 BIG-IP load balancer

In the example above, flow records will be published to *ipfix-pub-1*. *ipfix-pub-1* is configured with log-destination *ipfix-collector-1* which sends the IPFIX messages to IPFIX pool *ipfix-pool-1*. *ipfix-pool-1* has 10.28.118.6 as one of the IPFIX collectors. The virtual server *vip-1* is configured with IPFIX iRule *ipfix-rule-1* which specifies the IPFIX template and how the template gets filled and sent.

F5 and Tetration approved iRule for TCP virtual server can be found in the following file

See L4 iRule for TCP virtual server.

F5 and Tetration approved iRule for UDP virtual server can be found in the following file.

See L4 iRule for UDP virtual server.

F5 and Tetration approved iRule for HTTPS virtual server with authentication enabled can be found in the following file.

See iRule for HTTPS virtual server.

Note: Before using the iRule downloaded from this guide, please update the **log-publisher** to point to the log-publisher configured in the F5 connector where the iRule will be added.

Note: F5 has published a GitHub repository, f5-tetration to help users get started with flow-stitching. The iRules for publishing IPFIX records to F5 connector for various protocol types are available at: f5-tetration/irules. Please visit

this site for latest iRule definitions. In addition, F5 also developed a script to: (1) install the correct iRule for the virtual servers, (2) add a pool of IPFIX collector endpoints (where F5 connectors listen for IPFIX records), (3) configure the log-collector and log-publisher, and (4) bind the correct iRule to the virtual servers. This tool minimizes manual configuration and user error while enabling flow-stitching use-case. The script is available at f5-tetration/scripts.

How to Configure the Connector

The following configurations are allowed on the connector.

- *LDAP*: LDAP configuration supports discovery of LDAP attributes and provide a workflow to pick the attribute that corresponds to username and a list of up to 6 attributes to fetch for each user. Please refer to *Discovery* for more details.
- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Tetration Ingest appliance using a command that is allowed to be run on the container. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of F5 connectors on one Tetration Ingest appliance	3
Maximum number of F5 connectors on one Tenant (rootscope)	10
Maximum number of F5 connectors on Tetration	100

18.1.1.3 NetScaler Connector

NetScaler connector allows Tetration to ingest flow observations from Citrix ADCs (Citrix NetScalers). It allows Tetration to remotely monitor flow observations on Citrix ADCs and stitch client-side and server-side flows. Using this solution, the hosts do not need to run software agents, because Citrix ADCs will be configured to export IPFIX records to NetScaler connector for processing.

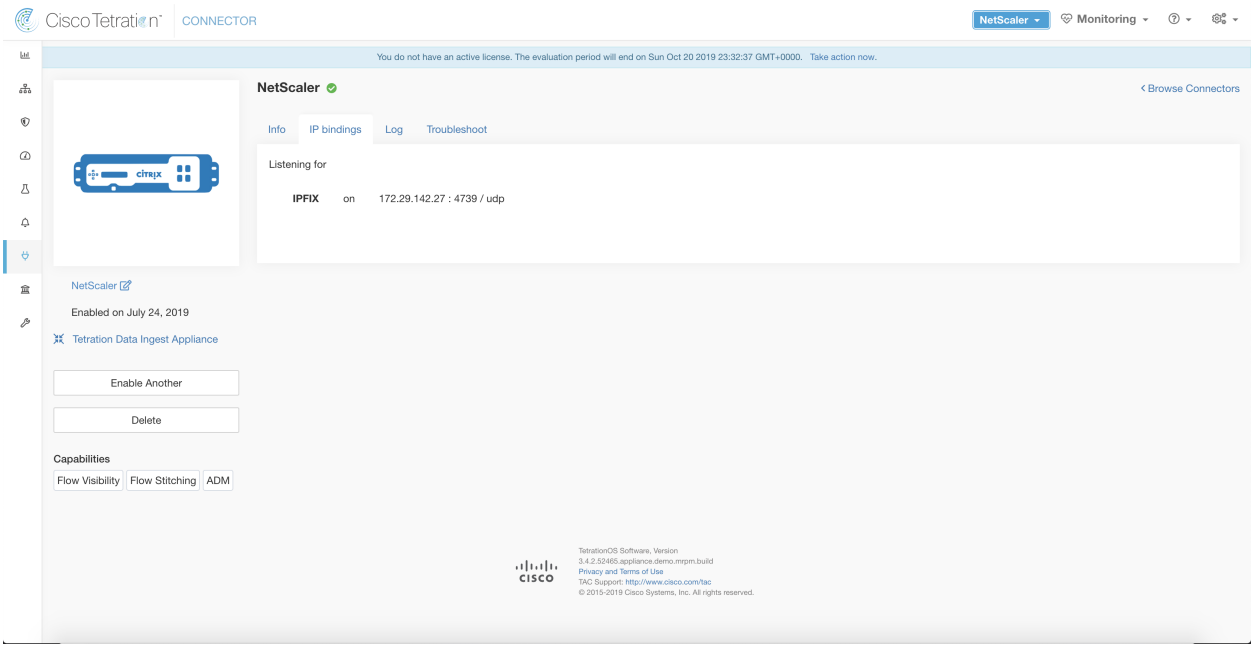


Fig. 18.1.1.3.1: NetScaler connector

What is Citrix NetScaler AppFlow

Citrix NetScaler AppFlow collects flow data for traffic going through the NetScaler and exports IPFIX records to flow collectors. Citrix AppFlow protocol uses IPFIX to export the flows to flow collectors. Citrix AppFlow is supported in Citrix NetScaler load balancers.

Typically, the setup involves the following steps:

1. Enable AppFlow feature on one or more Citrix NetScaler instances.
2. Configure the AppFlow collector endpoint information on the remote network devices. This AppFlow collector will be listening on configured endpoint to receive and process flow records.
3. Configure AppFlow actions and policies to export flow records to AppFlow collectors.

Note: NetScaler connector supports Citrix ADC software version 11.1.51.26 and above.

Flow Ingestion to Tetration

NetScaler connector is essentially a Citrix AppFlow (IPFIX) collector. The connector receives the flow records from Citrix ADCs, stitch the NATed flows and forwards them to Tetration for flow analysis. A NetScaler connector can be enabled on a Tetration Ingest appliance and runs as a Docker container. NetScaler connector also registers with Tetration as a Tetration NetScaler agent.

Note: NetScaler connector supports only IPFIX protocol.

Note: Each NetScaler connector should report only flows for one VRF. The flows exported by the connector is put

in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

How to configure AppFlow on NetScaler

The following steps are for NetScaler load balancer. (Ref: Configuring AppFlow)

Step 1: Enable AppFlow on NetScaler.

```
enable ns feature appflow
```

Step 2: Add AppFlow collector endpoints.

The collector receives the AppFlow records from NetScaler. Please specify the IP and port of NetScaler connector enabled on a Tetration Ingest appliance as an AppFlow collector.

```
add appflow collector c1 -IPAddress 172.26.230.173 -port 4739
```

Step 3: Configure an AppFlow action.

This lists the collectors that will get AppFlow records if the associated AppFlow policy matches.

```
add appflow action a1 -collectors c1
```

Step 4 Configure an AppFlow policy.

This is a rule that has to match for an AppFlow record to be generated.

```
add appflow policy p1 CLIENT.TCP.DSTPORT(22) a1
add appflow policy p2 HTTP.REQ.URL.SUFFIX.EQ("jpeg") a1
```

Step 5: Bind AppFlow policy to Virtual Server.

Traffic hitting the IP of the virtual server (VIP) will be evaluated for AppFlow policy matches. On a match, a flow record is generated and sent to all collectors listed in the associated AppFlow action.

```
bind lb vserver lb1 -policyname p1 -priority 10
```

Step 6: Optionally, bind AppFlow policy globally (for all virtual servers).

An AppFlow policy could also be bound globally to all virtual servers. This policy applies to all traffic that flows through Citrix ADC.

```
bind appflow global p2 1 NEXT -type REQ_DEFAULT
```

Step 7: Optionally, template refresh interval.

Default value for template refresh is 60 seconds.

```
set appflow param -templatereferesh 60
```

The above steps configures AppFlow on Citrix NetScaler load balancer to export IPFIX protocol packets for traffic going through NetScaler. The flow records will be sent to either 172.26.230.173:4739 (for traffic going through vserver lb1) and to 172.26.230.184:4739 (for all traffic going through the NetScaler). Each flow record includes 5 tuple information of the traffic and the byte/packet count of the flow.

The following screenshot shows a running configuration of AppFlow on a Citrix NetScaler load balancer.

```
MAARUMUG-M-M1PB:~ maarumug$ ssh nsroot@172.26.231.131
################################################################################
#                                                                              #
#        WARNING: Access to this system is for authorized users only          #
#         Disconnect IMMEDIATELY if you are not an authorized user!           #
#                                                                              #
################################################################################

Password:
Last login: Fri Dec 15 12:32:45 2017 from 10.128.140.136
 Done
> sh run | grep appflow
add appflow collector c1 -IPAddress 172.26.230.174
add appflow collector c2 -IPAddress 172.26.230.173
set appflow param -templateRefresh 60 -connectionChaining ENABLED
add appflow action act1 -collectors c1 c2
add appflow policy pol1 true act1
bind appflow global pol1 1 NEXT -type REQ_DEFAULT
>
```

Fig. 18.1.1.3.2: Running configuration of AppFlow on Citrix NetScaler load balancer

How to Configure the Connector

The following configurations are allowed on the connector.

- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Tetration Ingest appliance using a an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of NetScaler connectors on one Tetration Ingest appliance	3
Maximum number of NetScaler connectors on one Tenant (rootscope)	10
Maximum number of NetScaler connectors on Tetration	100

18.1.1.4 ASA Connector

ASA connector allows Tetration to ingest flow observations from Cisco Adaptive Security Appliance (ASA) firewall. Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay NetFlow Secure Event Logging (NSEL) records to ASA connector hosted in a Tetration Ingest appliance for processing.

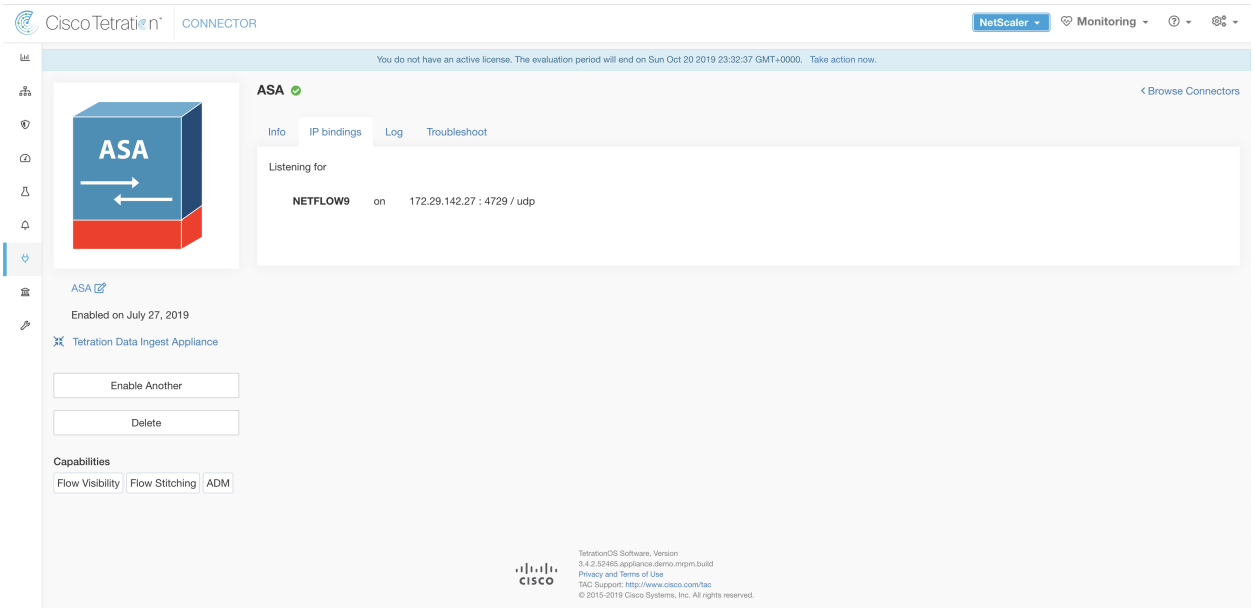


Fig. 18.1.1.4.1: ASA connector

What is ASA NSEL

Cisco ASA NSEL provides a stateful, IP flow monitoring that exports significant events in a flow to a NetFlow collector. When an event causes a state change on a flow, an NSEL event is triggered that sends the flow observation along with the event that caused the state change to the NetFlow collector. The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis.

Typically, the setup involves the following steps:

1. Enable NSEL feature on Cisco ASA firewall.
2. Configure the ASA connector endpoint information on Cisco ASA. ASA connector will be listening on configured endpoint to receive and process NSEL records.

Flow Ingestion to Tetration

ASA connector is essentially a NetFlow collector. The connector receives the NSEL records from Cisco ASA and forwards them to Tetration for flow analysis. ASA connector can be enabled on a Tetration Ingest appliance and runs as a Docker container.

ASA connector also registers with Tetration as a Tetration ASA agent. ASA connector decapsulates the NSEL protocol packets (i.e., flow records); then processes and reports the flows like a regular Tetration agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.

Note: ASA connector supports NetFlow v9 protocol.

Note: Each ASA connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

Handling NSEL Events

The following table shows how various NSEL events are handled by ASA connector. For more information about these elements, please refer to IP Flow Information Export (IPFIX) Entities document.

Flow Event Element ID: 233 Element Name: *NF_F_FW_EVENT*	Extended Flow Event Element ID: 33002 Element Name: *NF_F_FW_EXT_EVENT*	Action on ASA connector
0 (default, ignore this value)	Don't care	No op
1 (Flow created)	Don't care	Send flow to Tetration
2 (Flow deleted)	> 2000 (indicates the termination reason)	Send flow to Tetration
3 (Flow denied)	1001 (denied by ingress ACL)	Send flow with disposition marked as rejected to Tetration
	1002 (denied by egress ACL)	
	1003 (denied connection by ASA interface or denied ICMP(v6) to device)	
	1004 (first packet on TCP is not SYN)	
4 (Flow alert)	Don't care	No op
5 (Flow updated)	Don't care	Send flow to Tetration

Based on the NSEL record, ASA connector sends flow observation to Tetration. NSEL flow records are bidirectional. So, ASA connector sends 2 flows: forward flow and reverse flow to Tetration.

Here are the details about flow observation sent by ASA connector to Tetration.

Forward Flow observation

Field	NSEL Element ID	NSEL Element Name
Protocol	4	*NF_F_PROTOCOL*
Source Address	8	*NF_F_SRC_ADDR_IPV4*
	27	*NF_F_SRC_ADDR_IPV6*
Source Port	7	*NF_F_SRC_PORT*
Destination Address	12	*NF_F_DST_ADDR_IPV4*
	28	*NF_F_DST_ADDR_IPV6*
Destination Port	11	*NF_F_DST_PORT*
Flow Start Time	152	*NF_F_FLOW_CREATE_TIME_MSEC*
Byte Count	231	*NF_F_FWD_FLOW_DELTA_BYTES*
Packet Count	298	*NF_F_FWD_FLOW_DELTA_PACKETS*

Reverse Flow Information

Field	NSEL Element ID	NSEL Element Name
Protocol	4	*NF_F_PROTOCOL*
Source Address	12	*NF_F_DST_ADDR_IPV4*
	28	*NF_F_DST_ADDR_IPV6*
Source Port	11	*NF_F_DST_PORT*
Destination Address	8	*NF_F_SRC_ADDR_IPV4*
	27	*NF_F_SRC_ADDR_IPV6*
Destination Port	7	*NF_F_SRC_PORT*
Flow Start Time	152	*NF_F_FLOW_CREATE_TIME_MSEC*
Byte Count	232	*NF_F_REV_FLOW_DELTA_BYTES*
Packet Count	299	*NF_F_REV_FLOW_DELTA_PACKETS*

NAT

If the client to ASA flow is NATed, NSEL flow records indicate the NATed IP/port on the server side. ASA connector uses this information to stitch server to ASA and ASA to client flows.

Here is the NATed flow record in the forward direction.

Field	NSEL Element ID	NSEL Element Name
Protocol	4	*NF_F_PROTOCOL*
Source Address	225	*NF_F_XLATE_SRC_ADDR_IPV4*
	281	*NF_F_XLATE_SRC_ADDR_IPV6*
Source Port	227	*NF_F_XLATE_SRC_PORT*
Destination Address	226	*NF_F_XLATE_DST_ADDR_IPV4*
	282	*NF_F_XLATE_DST_ADDR_IPV6*
Destination Port	228	*NF_F_XLATE_DST_PORT*
Flow Start Time	152	*NF_F_FLOW_CREATE_TIME_MSEC*
Byte Count	231	*NF_F_FWD_FLOW_DELTA_BYTES*
Packet Count	298	*NF_F_FWD_FLOW_DELTA_PACKETS*

The forward flow will be marked as related to the NATed flow record in the forward direction (and vice versa).

Here is the NATed flow record in the reverse direction.

Field	NSEL Element ID	NSEL Element Name
Protocol	4	*NF_F_PROTOCOL*
Source Address	226	*NF_F_XLATE_DST_ADDR_IPV4*
	282	*NF_F_XLATE_DST_ADDR_IPV6*
Source Port	228	*NF_F_XLATE_DST_PORT*
Destination Address	225	*NF_F_XLATE_SRC_ADDR_IPV4*
	281	*NF_F_XLATE_SRC_ADDR_IPV6*
Destination Port	227	*NF_F_XLATE_SRC_PORT*
Flow Start Time	152	*NF_F_FLOW_CREATE_TIME_MSEC*
Byte Count	232	*NF_F_REV_FLOW_DELTA_BYTES*
Packet Count	299	*NF_F_REV_FLOW_DELTA_PACKETS*

The reverse flow will be marked as related to the NATed flow record in the reverse direction (and vice versa).

Note: Only NSEL element IDs listed in this section are supported by ASA connector.

How to configure NSEL on Cisco ASA

The following steps are guidelines on how to configure NSEL and export NetFlow packets to a collector (i.e., ASA connector). Please also refer to the official Cisco configuration guide at Cisco ASA NSEL for more details.

Here is an example NSEL configuration.

```
flow-export destination outside 172.29.142.27 4729
flow-export template timeout-rate 1
!
policy-map flow_export_policy
  class class-default
  flow-export event-type flow-create destination 172.29.142.27
  flow-export event-type flow-teardown destination 172.29.142.27
  flow-export event-type flow-denied destination 172.29.142.27
  flow-export event-type flow-update destination 172.29.142.27
  user-statistics accounting
service-policy flow_export_policy global
```

In this example, ASA appliance is configured to sent NetFlow packets to *172.29.142.27* on port *4729*. In addition, *flow-export* actions are enabled on *flow-create*, *flow-teardown*, *flow-denied*, and *flow-update* events. When these flow events occur on ASA, a NetFlow record is generated and sent to the destination specified in the configuration.

Assuming an ASA connector is enabled on Tetration and listening on *172.29.142.27:4729* in a Tetration Ingest appliance, the connector will receive NetFlow packets from ASA appliance. The connector processes the NetFlow records as discussed in *Handling NSEL Events* and exports flow observations to Tetration. In addition, for NATed flows, the connector stitches the related flows (client-side and server-side) flows.

How to Configure the Connector

The following configurations are allowed on the connector.

- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Tetration Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of ASA connectors on one Tetration Ingest appliance	1
Maximum number of ASA connectors on one Tenant (rootscope)	10
Maximum number of ASA connectors on Tetration	100

18.1.1.5 AWS Connector

AWS connector allows Tetration to ingest flow observations from AWS VPC flow logs. Using this solution, AWS workloads do not need to run software agents; rather VPC Flow Logs are configured to be exported to S3 buckets and AWS connector will download these files, process them and export flow records to Tetration.

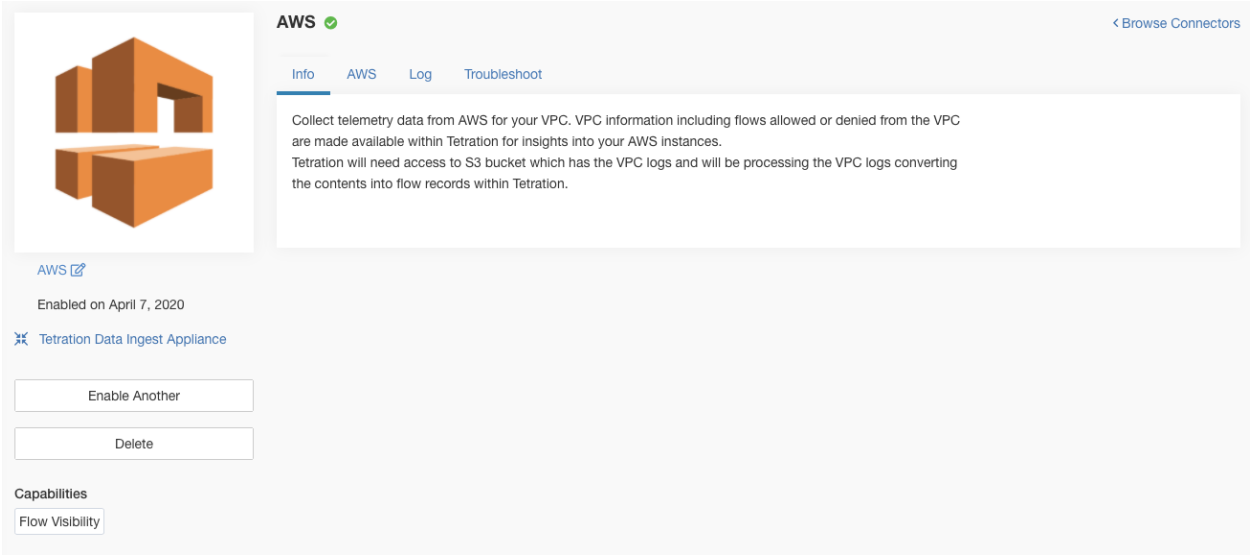


Fig. 18.1.1.5.1: AWS connector

What is AWS VPC Flow Log

AWS VPC Flow Logs collects flow data for traffic going through AWS VPC. VPC Flow Logs are collected in Amazon CloudWatch and can then be exported to Amazon S3 bucket.

Typically, setting up AWS VPC Flow Logs involves the following steps:

1. Create an IAM role for flow logs.
2. Create VPC flow log for VPC of interest.
3. Create Amazon S3 bucket.
4. [Optional] Create an export task to export data from CloudWatch to Amazon S3 bucket.

Flow Ingestion to Tetration

AWS connector fetches the flow logs from AWS S3 buckets, parses the flow logs and generates flow records that Tetration cluster can consume. AWS connector runs two services:

1. AWS VPC flow log downloader: is responsible for downloading the log files from S3 buckets specified in a configuration.
2. AWS VPC flow log exporter: parses log files downloaded by the downloader service, generates flow records and exports them to Tetration.

Note: Each AWS connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster. For TaaS cluster, Agent VRF configuration is not needed. Flows will be exported to the tenant for which the AWS connector is enabled.

How to configure AWS VPC Flow Logs

The following steps enable users to configure AWS VPC Flow Logs. (Ref: Enabling AWS VPC Flow Logs)

Step 1: Create an IAM role for flow logs.

Go to IAM Console, and create an IAM user with permissions to create, delete, view flow log events.

Step 2: Create Amazon S3 Bucket.

Go to Amazon S3 Console, and create an Amazon S3 bucket for the flow logs to be published.

Step 3: Create Flow Logs for VPC.

Go to Amazon VPC Console, and enable flow logs for the VPC of interest. Flow logs destination can be the Amazon S3 bucket created in previous step or Amazon CloudWatch Logs. If the S3 bucket was used as flow log destination, the log files will be published to the Amazon S3 bucket at 5-minute intervals. If CloudWatch Logs was selected, please refer to the following steps to export flow logs to S3 bucket.

Note: Only *AWS default format* is supported by the AWS connector.

[Optional] Step 4: Set Permissions to S3 Bucket.

If CloudWatch Logs was used as flow log destination, enable AWS CloudWatch to to export log records to the S3 Bucket created in the previous step. Set the Principal of the end point of of the region where you are exporting the logs from.

[Optional] Step 5: Create an Export task.

If CloudWatch Logs was used as flow log destination, go to Amazon CloudWatch Console, select Logs pane and choose *Export data to Amazon S3*. A dialog box will pop up to select the name of S3 bucket and the time range of data that should be exported. Once all the required data is provided and export is enabled, the logs will start showing up in S3 buckets.

Detailed configuration steps can be found in Amazon Documentation.

1. Flow Logs documentation: <https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html>
2. Amazon S3 documentation: <https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/S3ExportTasksConsole.html>

Once the logs are exported to S3 Buckets, AWS VPC Flow Log downloader in the Tetration AWS VPC Flow Logs Collector virtual appliance will start downloading the log files into the shared volume. This triggers AWS VPC Flow Log agent to start processing the log files and export the flow records to Tetration cluster. Once a log file is processed and all flow logs are exported, the log file is purged from the shared volume.

Note: AWS would aggregate flow log using the *Maximum aggregation interval* configured during flow log creation. After flow log data is captured within an aggregation interval, it takes additional time to process and publish the data to CloudWatch Logs or Amazon S3. This additional time could be up to 5 minutes to publish to CloudWatch Logs, and up to 10 minutes to publish to Amazon S3. The AWS connector would take some additional time to download the new flow log file. The overall delay is approximately 20 minutes.

How to Configure the Connector

The following configurations are allowed on the connector.

- *AWS*: Please refer to *AWS Configuration* for more details.

- *Log*: Please refer to *Log Configuration* for more details.

Limits

Metric	Limit
Maximum number of AWS connectors on one Tetration Ingest appliance	1
Maximum number of AWS connectors on one Tenant (rootscope)	10
Maximum number of AWS connectors on Tetration	100

18.1.1.6 Meraki Connector

Meraki connector allows Tetration to ingest flow observations from Meraki firewalls (included in Meraki MX security appliances and wireless access points). Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay NetFlow records to Meraki connector hosted in a Tetration Ingest appliance for processing.

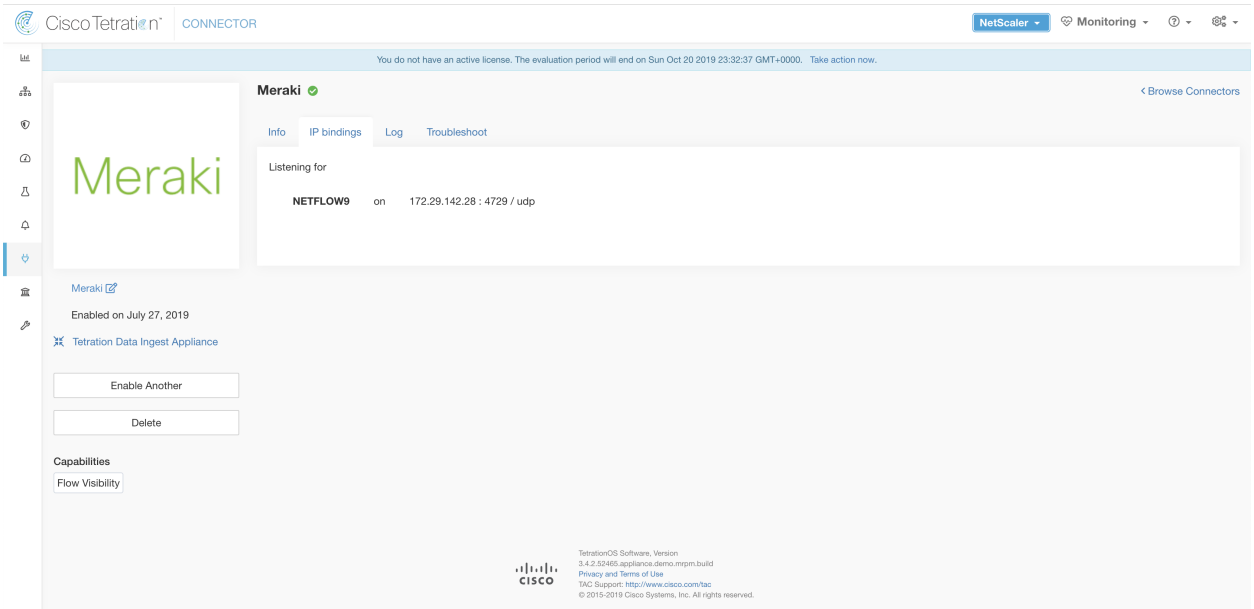


Fig. 18.1.1.6.1: Meraki connector

What is NetFlow

NetFlow protocol allows network devices such as Meraki Firewall to aggregate traffic that passes through them into flows and export these flows to a flow collector. The flow collector receives these flow records and stores them in their flow storage for offline querying and analysis.

Typically, the setup involves the following steps:

1. Enable NetFlow statistics reporting on Meraki Firewall
2. Configure the NetFlow collector endpoint information on Meraki Firewall.

Flow Ingestion to Tetration

Meraki connector is essentially a NetFlow collector. The connector receives the flow records from the Meraki firewalls that are configured to export NetFlow traffic statistics. It processes the NetFlow records and sends the flow observations reported by Meraki firewalls to Tetration for flow analysis. A Meraki connector can be enabled on a Tetration Ingest appliance and runs as a Docker container.

Meraki connector also registers with Tetration as a Tetration Meraki agent. Meraki connector decapsulates the NetFlow protocol packets (i.e., flow records); then processes and reports the flows like a regular Tetration agent. Unlike a Deep Visibility Agent, it does not report any process or interface information.

Note: Meraki connector supports NetFlow v9 protocol.

Note: Each Meraki connector should report only flows for one VRF. The flows exported by the connector is put in the VRF based on the Agent VRF configuration in Tetration cluster. To configure the VRF for the connector, go to: Settings menu > Agent Config > Software Agent Config. In this page, under *Agent Remote VRF Configurations* section, click *Create Config* and provide the details about the connector. The form requests the user to provide: the name of the VRF, IP subnet of the connector, and range of port numbers that can potentially send flow records to the cluster.

Handling NetFlow Records

Based on the NetFlow record, Meraki connector sends flow observation to Tetration. Meraki NetFlow flow records are bidirectional. So, Meraki connector sends 2 flows: forward flow and reverse flow to Tetration.

Here are the details about flow observation sent by Meraki connector to Tetration.

Forward Flow observation

Field	Element ID	Element Name
Protocol	4	*protocolIdentifier*
Source Address	8	*sourceIPv4Address*
Source Port	7	*sourceTransportPort*
Destination Address	12	*destinationIPv4Address*
Destination Port	11	*destinationTransportPort*
Byte Count	1	*octetDeltaCount*
Packet Count	2	*packetDeltaCount*
Flow Start Time		Set based on when the NetFlow record for this flow is received on the connector

Reverse Flow Information

Field	Element ID	Element Name
Protocol	4	*protocolIdentifier*
Source Address	8	*sourceIPv4Address*
Source Port	7	*sourceTransportPort*
Destination Address	12	*destinationIPv4Address*
Destination Port	11	*destinationTransportPort*
Byte Count	23	*postOctetDeltaCount*
Packet Count	24	*postPacketDeltaCount*
Flow Start Time		Set based on when the NetFlow record for this flow is received on the connector

How to configure NetFlow on Meraki Firewall

The following steps show how to configure NetFlow reporting on Meraki Firewall.

1. Login to Meraki UI console.
2. Navigate to **Network-wide > General**. In *Reporting* settings, enable `NetFlow traffic reporting` and make sure the value is set to *Enabled: send NetFlow traffic statistics*.
3. Set `NetFlow collector IP` and `NetFlow collector port` to the IP and port on which Meraki connector is listening in Tetration Ingest appliance. Default port on which Meraki connector listens for NetFlow records is 4729.
4. Save the changes.

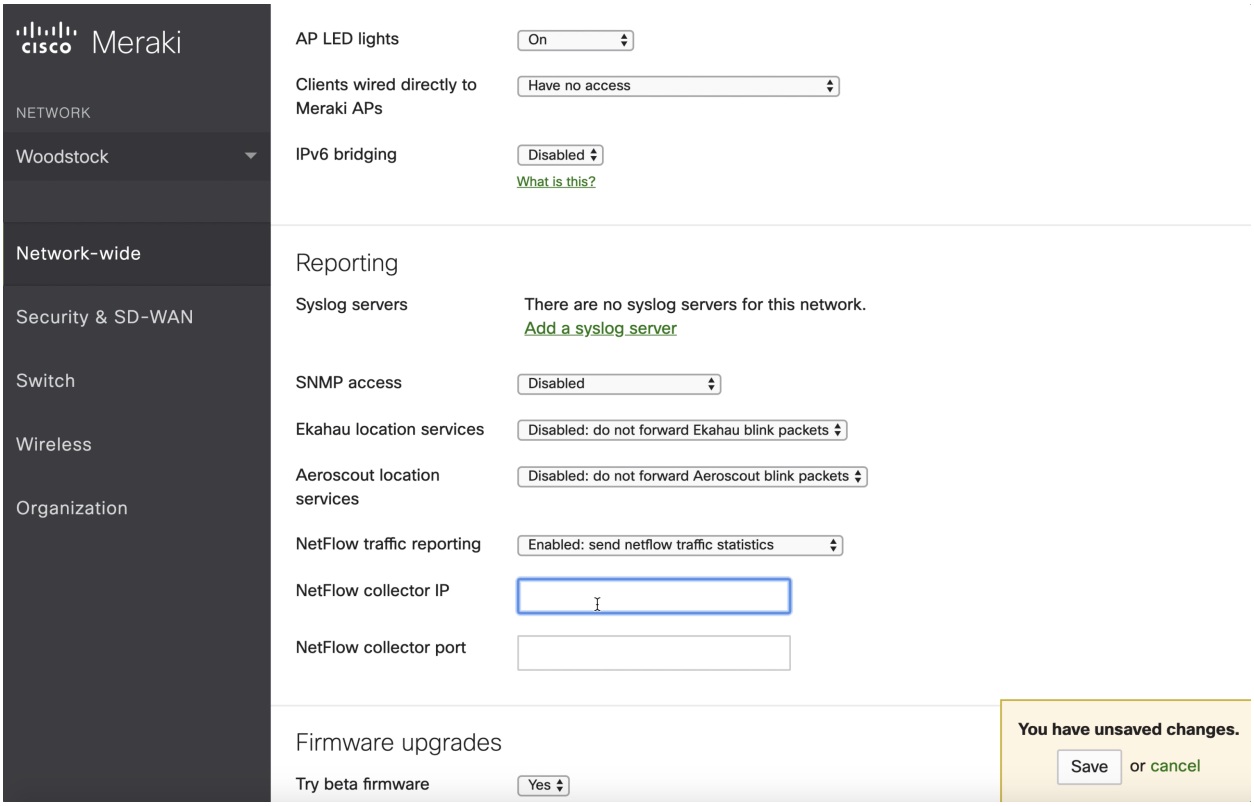


Fig. 18.1.1.6.2: Enabling NetFlow on a Meraki Firewall

How to Configure the Connector

The following configurations are allowed on the connector.

- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of NetFlow v9 protocol on the connector can be updated on the Docker container in Tetration Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of Meraki connectors on one Tetration Ingest appliance	1
Maximum number of Meraki connectors on one Tenant (rootscope)	10
Maximum number of Meraki connectors on Tetration	100

18.1.2 Connectors for Endpoints

Connectors for endpoints provide endpoint context for Tetration.

Connector	Description	Deployed on Virtual Appliance
AnyConnect	Collect telemetry data from Cisco AnyConnect Network Visibility Module (NVM) and enrich endpoint inventories with user attributes	Tetration Ingest
ISE	Collect information about endpoints and inventories managed by Cisco ISE appliances and enrich endpoint inventories with user attributes and secure group labels (SGL).	Tetration Edge

18.1.2.1 AnyConnect Connector

AnyConnect connector monitors endpoints that run Cisco AnyConnect Secure Mobility Client with Network Visibility Module (NVM). Using this solution, the hosts do not need to run any software agents on endpoints, because NVM sends host, interface, and flow records in IPFIX format to a collector (e.g., AnyConnect connector).

AnyConnect connector does the following high-level functions.

1. Register each endpoint (supported user devices such as a desktop, a laptop, or a smartphone) on Tetration as an AnyConnect agent.
2. Update interface snapshots from these endpoints with Tetration.
3. Send flow information exported by these endpoints to Tetration collectors.
4. Periodically send process snapshots for processes that generate flows on the endpoints tracked by the AnyConnect connector.
5. Label endpoint interface IP addresses with Lightweight Directory Access Protocol (LDAP) attributes corresponding to the logged-in-user at each endpoint.

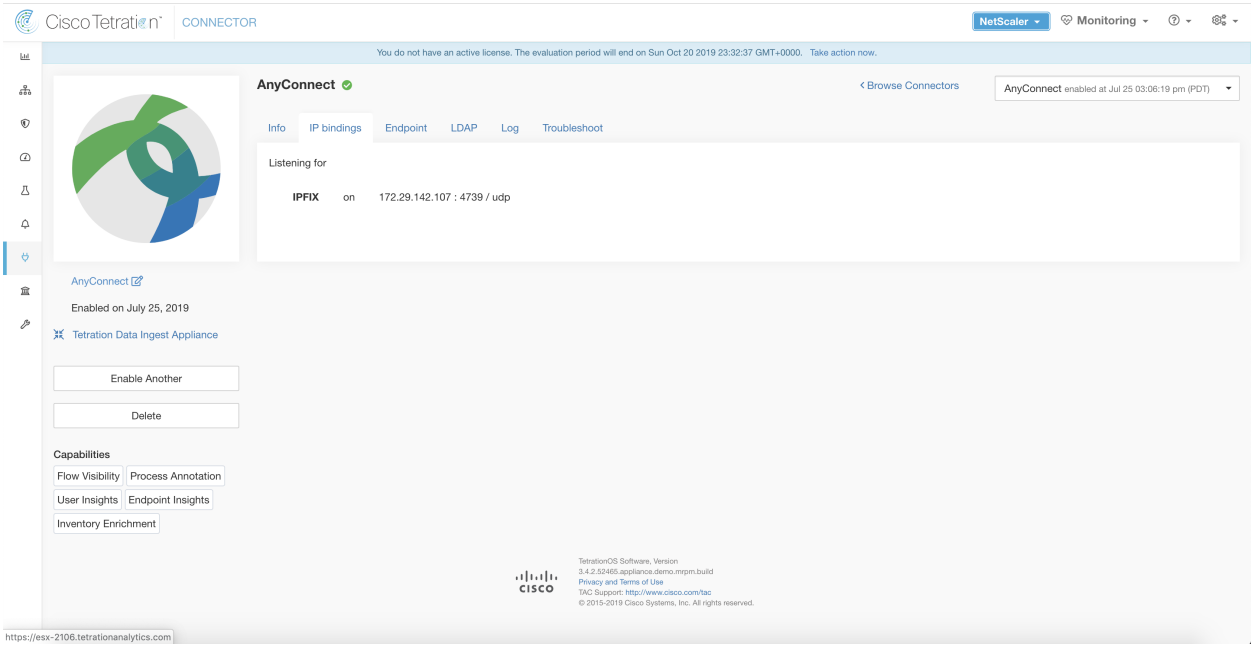


Fig. 18.1.2.1.1: AnyConnect connector

What is AnyConnect NVM

AnyConnect NVM provides visibility and monitoring of endpoint and user behavior both on and off premises. It collects information from endpoints that includes the following context.

1. **Device/Endpoint Context**: device/endpoint specific information.
2. **User Context**: users associated with the flow.
3. **Application Context**: processes associated with the flow.
4. **Location Context**: location specific attributes -if available.
5. **Destination Context**: FQDN of the destination.

AnyConnect NVM generates 3 types of records.

NVM Record	Description
Endpoint Record	device/endpoint information including unique device identifier (UDID), hostname, OS name, OS version and manufacturer.
Interface Record	information about each interface in the endpoint including the endpoint UDID, interface unique identifier (UID), interface index, interface type, interface name, and MAC address.
Flow Record	information about flows seen on the endpoint including endpoint UDID, interface UID, 5-tuple (source/destination ip/port and protocol), in/out byte counts, process information, user information, and fqdn of the destination.

Each record is generated and exported in IPFIX protocol format. When the device is in a trusted network (on-premise/VPN), AnyConnect NVM exports records to a configured collector. AnyConnect connector is an example IPFIX collector that can receive and process IPFIX stream from AnyConnect NVM.

Note: AnyConnect connector supports AnyConnect NVM from 4.2+ versions of Cisco AnyConnect Secure Mobility Client.

How to configure AnyConnect NVM

See How to Implement AnyConnect NVM document for step by step instructions on how to implement AnyConnect NVM using either Cisco Adaptive Security Appliance (ASA) or Cisco Identity Services engine (ISE). Once NVM module is deployed, an NVM profile should be specified and pushed to and installed on the endpoints running Cisco AnyConnect Secure Mobility Client. When specifying NVM profile, the IPFIX collector should be configured to point to AnyConnect connector on port 4739.

AnyConnect connector also registers with Tetration as a Tetration AnyConnect Proxy agent.

Processing NVM records

AnyConnect connector processes AnyConnect NVM records as shown below.

Endpoint Record

Upon receiving an endpoint record, AnyConnect connector registers that endpoint as AnyConnect agent on Tetration. AnyConnect connector uses the endpoint specific information present in the NVM record along with AnyConnect connector's certificate to register the endpoint. Once an endpoint is registered, data-plane for the endpoint is enabled by creating a new connection to one of the collectors in Tetration. Based on the activity (flow records) from this endpoint, AnyConnect connector checks-in the AnyConnect agent corresponding to this endpoint with the cluster periodically (20-30 minutes).

AnyConnect NVM starts to send agent version from 4.9. By default, the AnyConnect endpoint would be registered as version 4.2.x on Tetration. This version indicates the minimum supported AnyConnect NVM version. For the AnyConnect endpoints with version 4.9 or newer, the corresponding AnyConnect agent on Tetration would show the actual version installed.

Note: The AnyConnect agent installed version is not controlled by Tetration. Attempting to upgrade the AnyConnect endpoint agent on Tetration UI would not take effect.

Interface Record

IP address for an interface is not part of the AnyConnect NVM interface record. IP address for an interface is determined when flow records start coming from the endpoint for that interface. Once IP address is determined for an interface, AnyConnect connector sends a complete snapshot of all interfaces of that endpoint whose IP address is determined to config server of Tetration. This associates the VRF with the interface data and flows coming in on these interfaces will now be marked with this VRF.

Flow Record

Upon receiving a flow record, AnyConnect connector translates the record to the format that Tetration understands and sends FlowInfo over the dataplane corresponding to that endpoint. Furthermore, it stores process information included in the flow record locally. In addition, if LDAP configuration is provided to AnyConnect connector, it determines values for configured LDAP attributes of the logged-in-user of the endpoint. The attributes are associated to the endpoint IP address where the flow happened. Periodically, process information and user labels are pushed to Tetration.

Note: Each AnyConnect connector will report only endpoints/interfaces/ flows for one VRF. The endpoints and interfaces reported by AnyConnect connector are associated with the VRF based on the Agent VRF configuration in Tetration. The flows exported by the AnyConnect connector agent on behalf of the AnyConnect endpoint belong to the same VRF. To configure the VRF for the agent, go to: Settings menu > Agent Config > Software Agent Config. In this page, under "Agent Remote VRF Configurations" section, click "Create Config" and provide the details about the AnyConnect connector. The form requests the user to provide: the name of the VRF, IP subnet of the host on which the agent is installed, and range of port numbers that can potentially send flow records to the cluster.

Duplicate UDIDs in Windows Endpoints

If endpoint machines are cloned from the same golden image, it is possible that the UDID of all cloned endpoints are identical. In such cases, AnyConnect connector receives endpoint records from these endpoints with identical UDID and registers them on Tetration with same UDID. When interface/flow records are received by the connector from these endpoints, it is impossible for the connector to determine the correct AnyConnect agent on Tetration to associate the data. The connector associates all the data to one endpoint (and it is not deterministic).

To deal with this problem, AnyConnect NVM 4.8 release ships a tool called *dartcli.exe* to find and regenerate UDID on the endpoint.

- *dartcli.exe -u* retrieves the UDID of the endpoint.
- *dartcli.exe -nu* regenerates the UDID of the endpoint.

To run this tool, please use the following steps.

```
C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe␣
↪-u
UDID : 8D0D1E8FA0AB09BE82599F10068593E41EF1BFFF

C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe␣
↪-nu
Are you sure you want to re-generate UDID [y/n]: y
Adding nonce success
UDID : 29F596758941E606BD0AFF49049216ED5BB9F7A5

C:\Program Files (x86)\Cisco\Cisco AnyConnect Secure Mobility Client\DART>dartcli.exe␣
↪-u
UDID : 29F596758941E606BD0AFF49049216ED5BB9F7A5
```

Periodic Tasks

Periodically, AnyConnect connector sends process snapshots and user labels on AnyConnect endpoint inventories.

1. **Process Snapshots**: every 5 minutes, AnyConnect connector walks through the processes it maintains locally for that interval and sends process snapshot for all the endpoints that had flows during that interval.

2. **User Labels**: every 2 minutes, AnyConnect connector walks through the LDAP user labels it maintains locally and updates User Labels on those IP addresses.

For user labels, AnyConnect connector creates a local snapshot of LDAP attributes of all users in the organization. When AnyConnect connector is enabled, configuration for LDAP (server/port information, attributes to fetch for a user, attribute that contains the username) may be provided. In addition, the LDAP user credentials to access LDAP server may be provided. LDAP user credentials are encrypted and never revealed in the AnyConnect connector. Optionally, an LDAP certificate may be provided for securely accessing LDAP server.

Note: AnyConnect connector creates a new local LDAP snapshot every 24 hours. This interval is configurable in LDAP configuration of the connector.

How to Configure the Connector

The following configurations are allowed on the connector.

- *LDAP*: LDAP configuration supports discovery of LDAP attributes and provide a workflow to pick the attribute that corresponds to username and a list of up to 6 attributes to fetch for each user. Please refer to *Discovery* for more details.
- *Endpoint*: Please refer to *Endpoint Configuration* for more details.
- *Log*: Please refer to *Log Configuration* for more details.

In addition, the listening ports of IPFIX protocol on the connector can be updated on the Docker container in Tetration Ingest appliance using an allowed command. This command can be issued on the appliance by providing the connector ID of the connector, type of the port to be update, and the new port information. The connector ID can be found on the connector page in Tetration UI. Please refer to *Update Listening Ports of Connectors* for more details.

Limits

Metric	Limit
Maximum number of AnyConnect connectors on one Tetration Ingest appliance	1
Maximum number of AnyConnect connectors on one Tenant (rootscope)	50
Maximum number of AnyConnect connectors on Tetration	500

18.1.2.2 ISE Connector

ISE connector connects with Cisco Identity Services Engine using Cisco Platform Exchange Grid (pxGrid), to get contextual information regarding endpoints reported by Cisco ISE. Using this solutions, we can get enriched metadata for endpoints.

ISE connector does the following high-level functions.

1. Register each endpoint seen by ISE on Tetration as ISE agent.
2. Update metadata information regarding these endpoints to Tetration including MDM details, authentication, Security Group labels etc.

3. Periodically take a snapshot and update cluster with active endpoints seen on ISE.

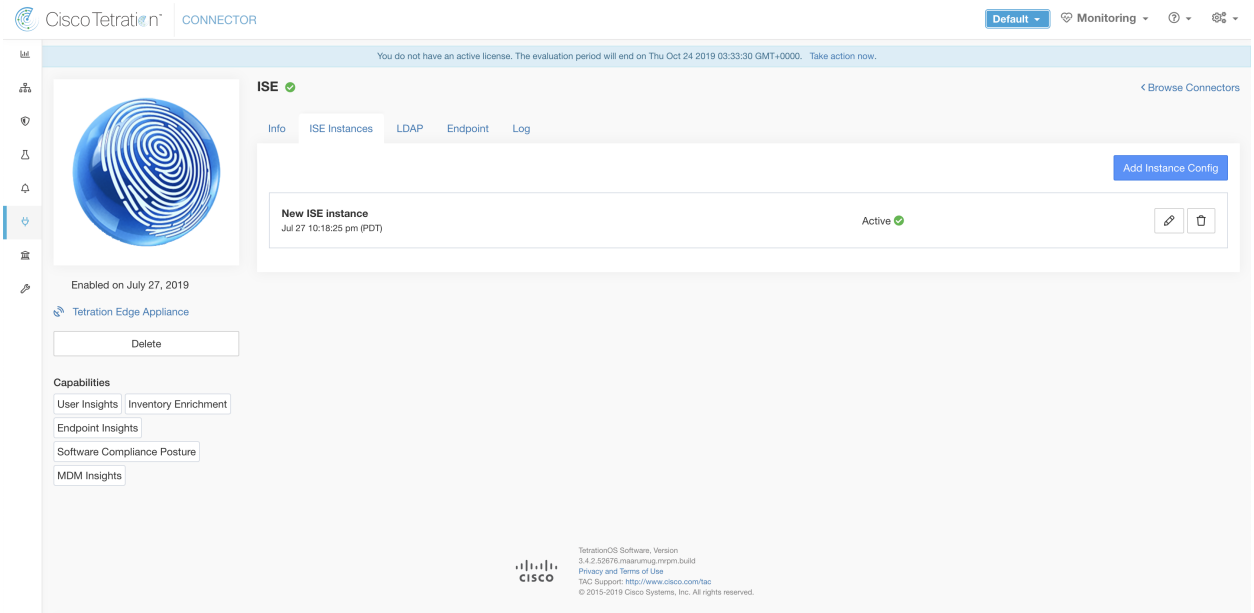


Fig. 18.1.2.2.1: ISE connector

Note: Each ISE connector will register only endpoints and interfaces for one VRF. The endpoints and interfaces reported by ISE connector are associated with the VRF based on the Agent VRF configuration in Tetration. To configure the VRF for the agent, navigate to: **Settings menu > Agent Config > Software Agent Config**. In this page, under "Agent Remote VRF Configurations" section, click "Create Config" and provide the details about the ISE connector. The form requests the user to provide: the name of the VRF, IP subnet of the host on which the agent is installed, and range of port numbers that can potentially register ISE endpoints and interfaces on Tetration.

How to Configure the Connector

Note: We need ISE version 2.4+ for this integration.

The following configurations are allowed on the connector.

- *ISE Instance*: ISE connector can connect to multiple instances of ISE using provided configs. Each instance requires ISE certificate credentials along with hostname and nodename to connect to ISE. Please refer to *ISE Instance Configuration* for more details.
- *LDAP*: LDAP configuration supports discovery of LDAP attributes and provide a workflow to pick the attribute that corresponds to username and a list of up to 6 attributes to fetch for each user. Please refer to *Discovery* for more details.
- *Endpoint*: Please refer to *Endpoint Configuration* for more details.
- *Log*: Please refer to *Log Configuration* for more details.

ISE Instance Configuration

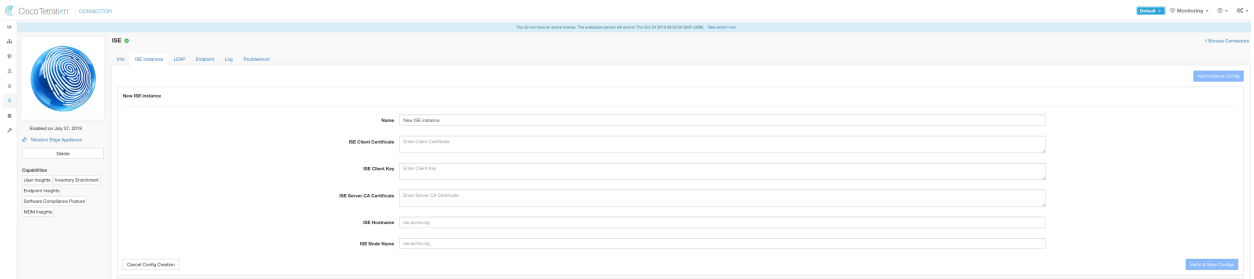


Fig. 18.1.2.2.2: ISE instance config

To fill the ISE config columns you need to do the following to get certs from ISE.

1. Go to pxGrid on ISE as shown below

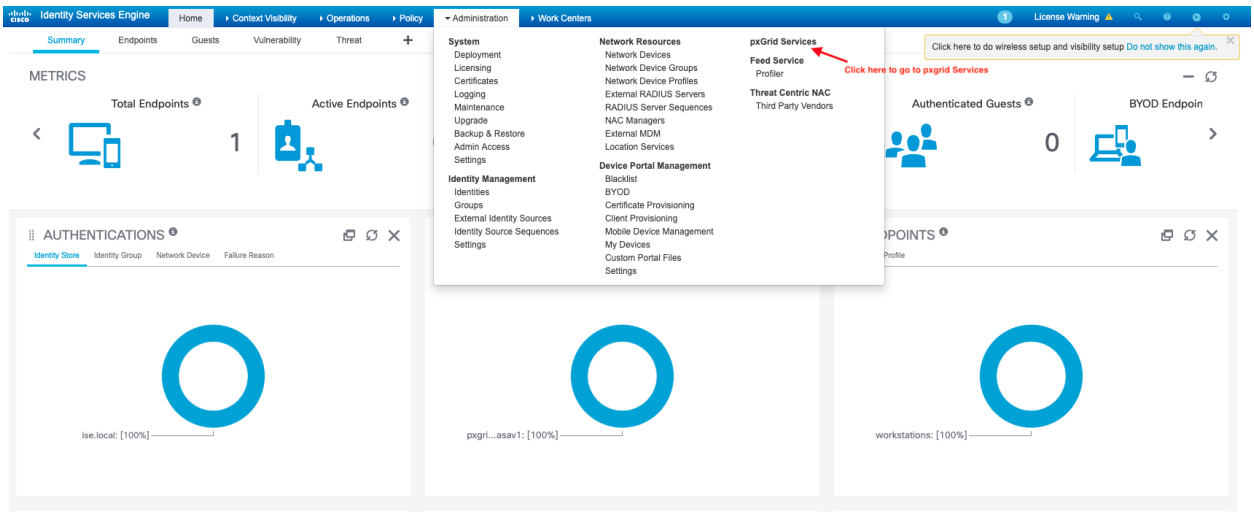


Fig. 18.1.2.2.3: ISE pxGrid integration illustration, browse to pxGrid tab.

2. Click on certificates tab.

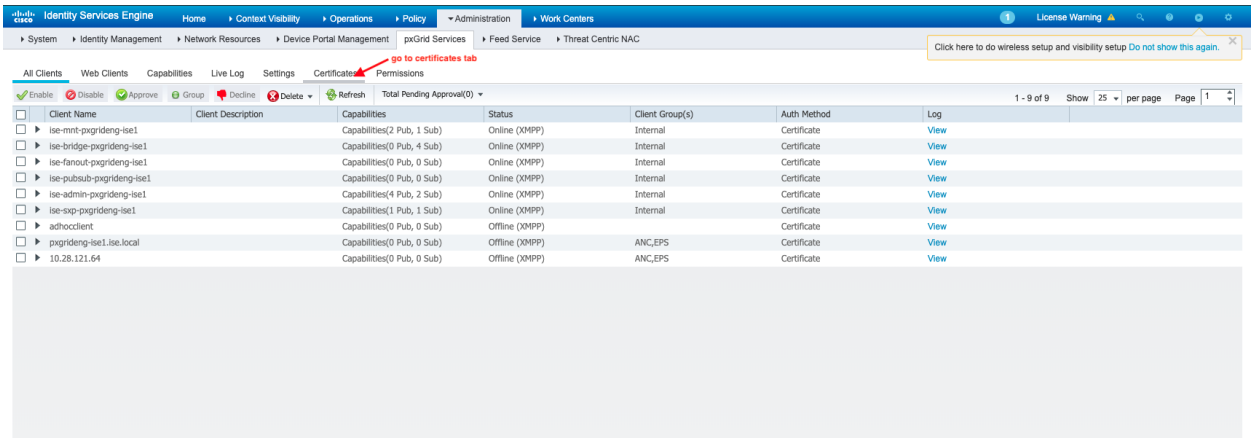


Fig. 18.1.2.2.4: ISE pxGrid integration illustration. Click on certificates tab.

3. Generate certificates as shown below

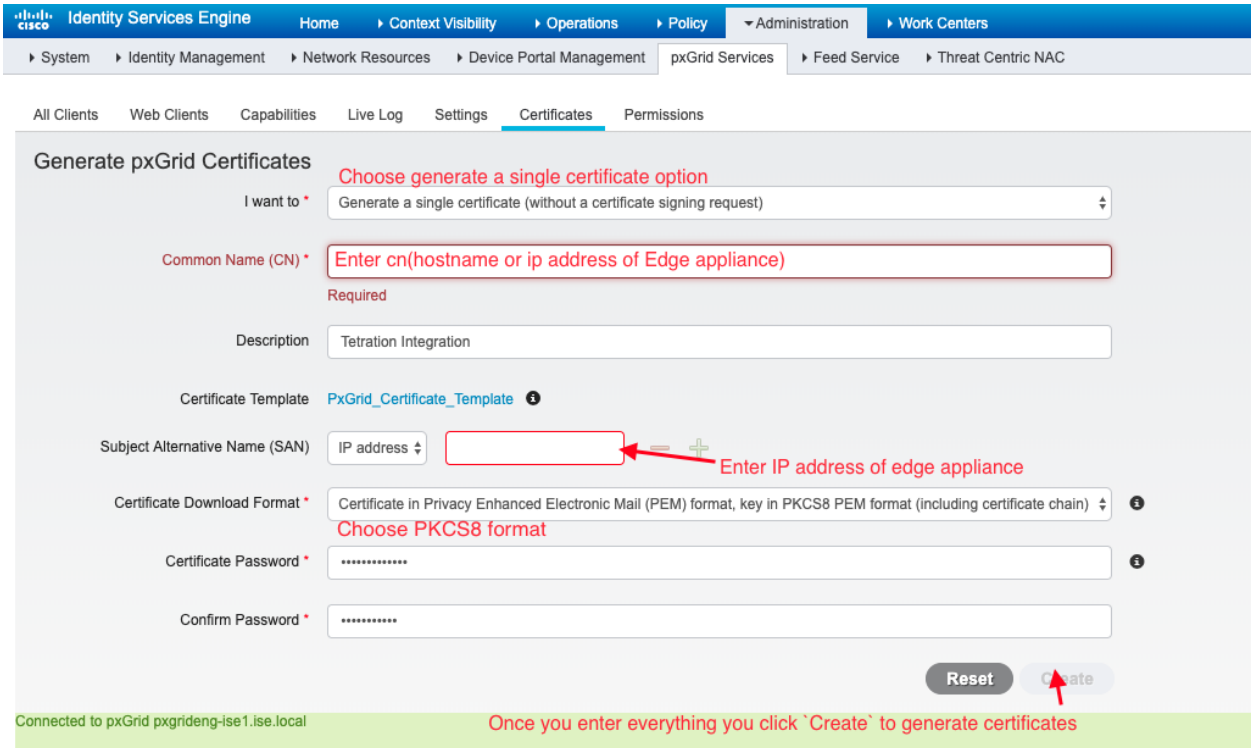


Fig. 18.1.2.2.5: ISE pxGrid integration illustration. Fill in the details as illustrated above.

Note: For the ISE integration to work, on ISE **pxGrid settings** we need to allow **Automatically approve new certificate-based accounts**

4. Unzip the zip file for certificate. Generate a decrypted key use the following command

openssl pkcs8 -in client.key > client.key.clear

5. Copy the client cert, client clear key and CA into the respective fields on the ISE configuration page on Tetration as shown below.

Note: Picking the certificates for connecting to ISE might differ based on ISE deployment.

- If *external CA* is used for certificates on ISE, same should be used to generate the certificates for connecting to ISE from Tetration.
- For multi-node ISE deployment with pxGrid, it is required that the all pxGrid nodes trust the Certs used for Tetration ISE Connector.

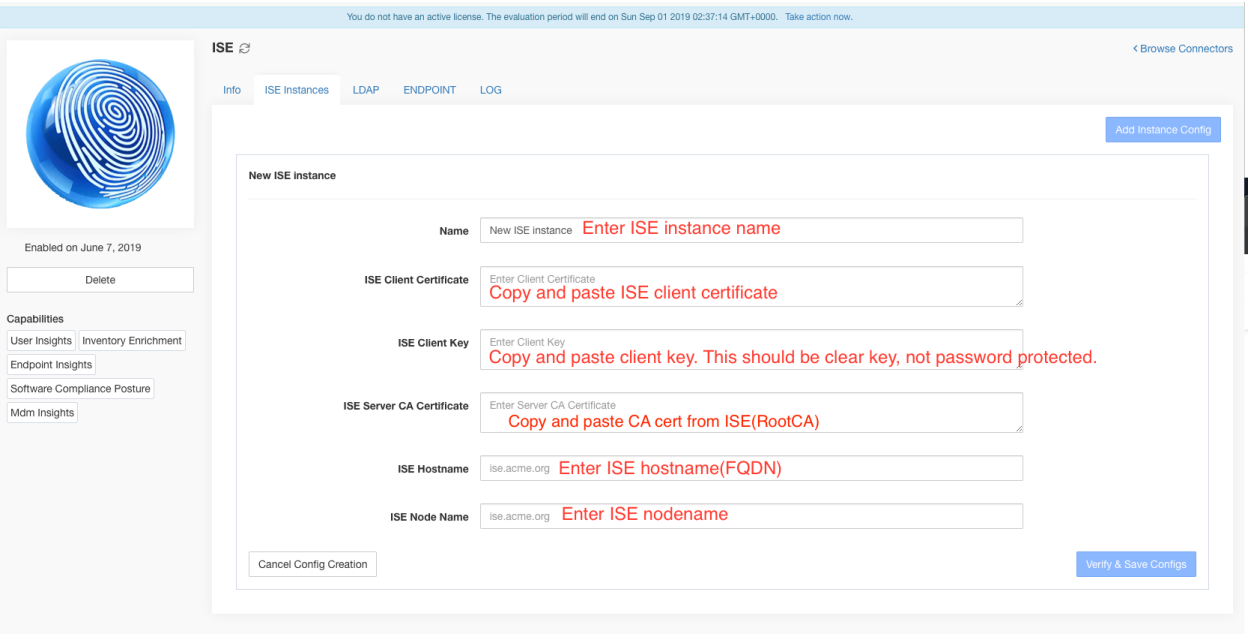


Fig. 18.1.2.2.6: ISE Connector configuration

Note: In case if IP Address is used instead of FQDN for ISE Hostname then it is required to have the IP address in the ISE CA certificate SAN, otherwise you might see connection failures.

Note: Number of active endpoints on ISE is not a snapshot and depends on configurations on ISE (wrt how long the aggregation duration is for computing the metric). The agent count on Tetration is always a snapshot based on last pull from ISE and pxgrid updates, typically the active device count over last one day (default refresh frequency for full snapshots is a day). Due to the difference in the way these numbers are depicted, it is possible that these two numbers will not always match.

Processing ISE records

ISE connector processes records as described below.

Endpoint Record

ISE connector connects to ISE instance and subscribes for any updates for endpoints over pxGrid. Upon receiving an endpoint record, ISE connector registers that endpoint as ISE agent on Tetration. ISE connector uses the endpoint specific information present in endpoint record along with ISE connector's certificate to register the endpoint. Once an endpoint is registered. ISE connector uses the endpoint object for inventory enrichment by sending this as user labels on Tetration. When ISE connector gets a disconnected endpoint from ISE, it deletes the inventory enrichment from Tetration.

Security Group Record

ISE connect also subscribes for updates about Security Group Labels change via pxGrid. On receiving this record, ISE connectors maintains a local database. It uses this database to map SGT name with value on receiving an endpoint record.

Periodic Tasks

Periodically, ISE connector sends user labels on ISE endpoint inventories.

1. **Endpoint Snapshots**: every 20 hours, ISE connector fetches a snapshot of endpoints and security group labels from ISE instance and updates the cluster if any change is detected. This call does not compute for endpoints that are disconnected in case we do not see endpoints on Tetration coming from ISE.
2. **User Labels**: every 2 minutes, ISE connector walks through the LDAP user and ISE endpoint labels it maintains locally and updates User Labels on those IP addresses.

For user labels, ISE connector creates a local snapshot of LDAP attributes of all users in the organization. When ISE connector is enabled, configuration for LDAP (server/port information, attributes to fetch for a user, attribute that contains the username) may be provided. In addition, the LDAP user credentials to access LDAP server may be provided. LDAP user credentials are encrypted and never revealed in the ISE connector. Optionally, an LDAP certificate may be provided for securely accessing LDAP server.

Note: ISE connector creates a new local LDAP snapshot every 24 hours. This interval is configurable in LDAP configuration of the connector.

Note: On upgrading Cisco ISE device, ISE connector will need to be re-configured with new certifcates generated by ISE after upgrade.

Limits

Metric	Limit
Maximum number of ISE instances that can be configured on one ISE connector	20
Maximum number of ISE connectors on one Tetration Edge appliance	1
Maximum number of ISE connectors on one Tenant (rootscope)	1
Maximum number of ISE connectors on Tetration	150

Note: Maximum number of ISE agents supported per connector is 400000.

18.1.3 Connectors for Inventory Enrichment

Connectors for inventory enrichment provides additional meta-data and context about the inventories (IP addresses) monitored by Tetration.

Connector	Description	Deployed on Virtual Appliance
ServiceNow	Collect endpoint information from ServiceNow instance and enrich the inventory with ServiceNow attributes	Tetration Edge

18.1.3.1 ServiceNow Connector

ServiceNow connector connects with ServiceNow Instance to get all the ServiceNow CMBD related labels for the endpoints in ServiceNow inventory. Using this solutions, we can get enriched metadata for the endpoints in Tetration.

ServiceNow connector does the following high-level functions.

1. Update ServiceNow metadata in Tetration's inventory for these endpoints.
2. Periodically take snapshot and update the labels on these endpoints.

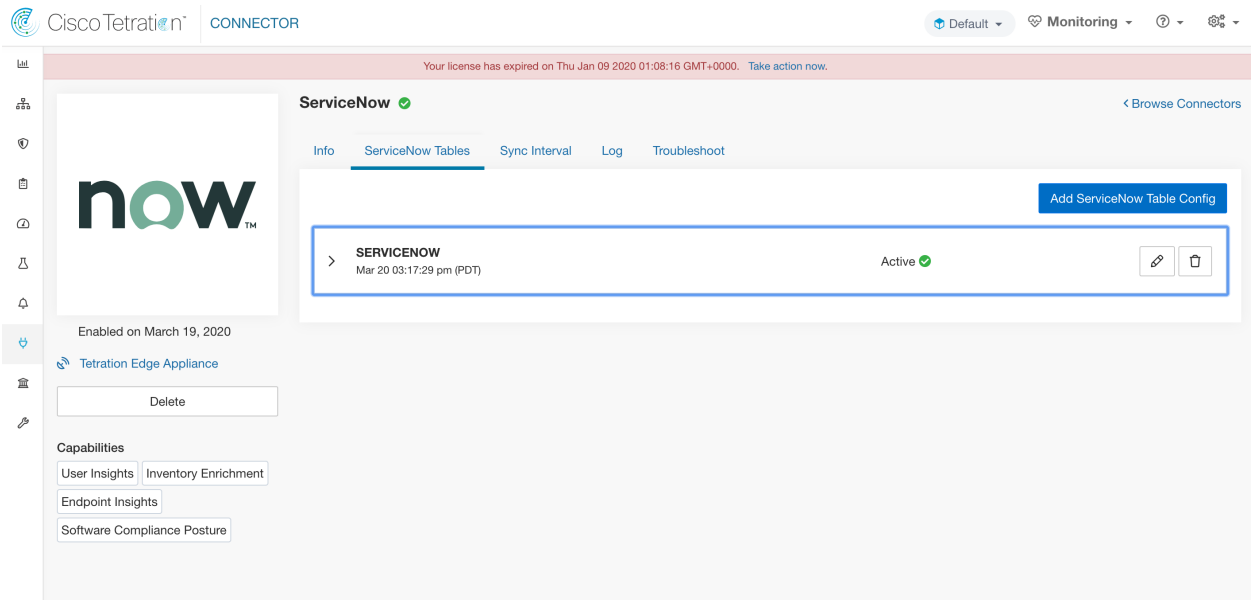


Fig. 18.1.3.1.1: ServiceNow connector

How to Configure the ServiceNow Connector

The following configurations are allowed on the connector.

- *ServiceNow Tables*: ServiceNow Tables configures the ServiceNow instance with it's credentials, and the information about ServiceNow tables to fetch the data from.
- *Sync Interval*: Sync Interval configuration allows to make change the periodicity at which Tetration should query ServiceNow instance for updated data.
- *Log*: Please refer to *Log Configuration* for more details.

ServiceNow Instance Configuration

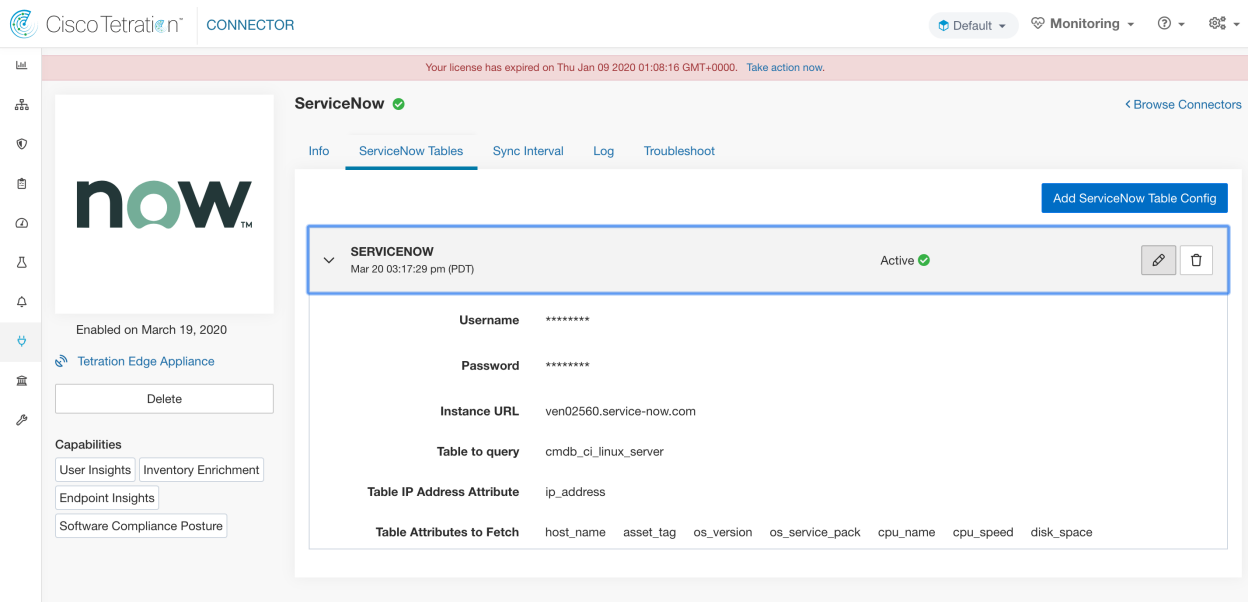


Fig. 18.1.3.1.2: ServiceNow instance config

You will need the following items to successfully configure a ServiceNow instance.

1. ServiceNow username
2. ServiceNow password
3. ServiceNow Instance URL

Subsequently, Tetration performs a discovery of all the tables from the ServiceNow Instance, and presents user with the list of tables to chose from. Once a user selects table, Tetration fetches all the list of attributes from that table for the user to select. User has to chose the ip_address attribute from the table as the key. Subsequently, user can chose upto 10 unique attributes from the table. Please see the following figures for each step.

Note: ServiceNow Connector can only support integrating with tables having **IP Address** field.

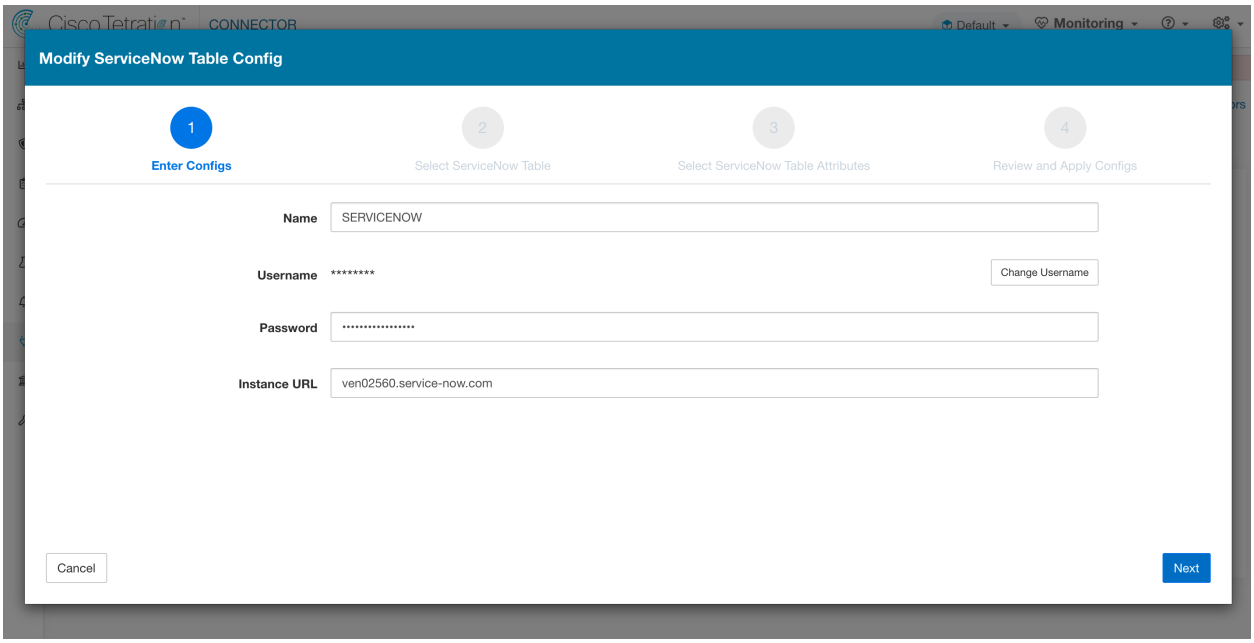


Fig. 18.1.3.1.3: ServiceNow instance config first step

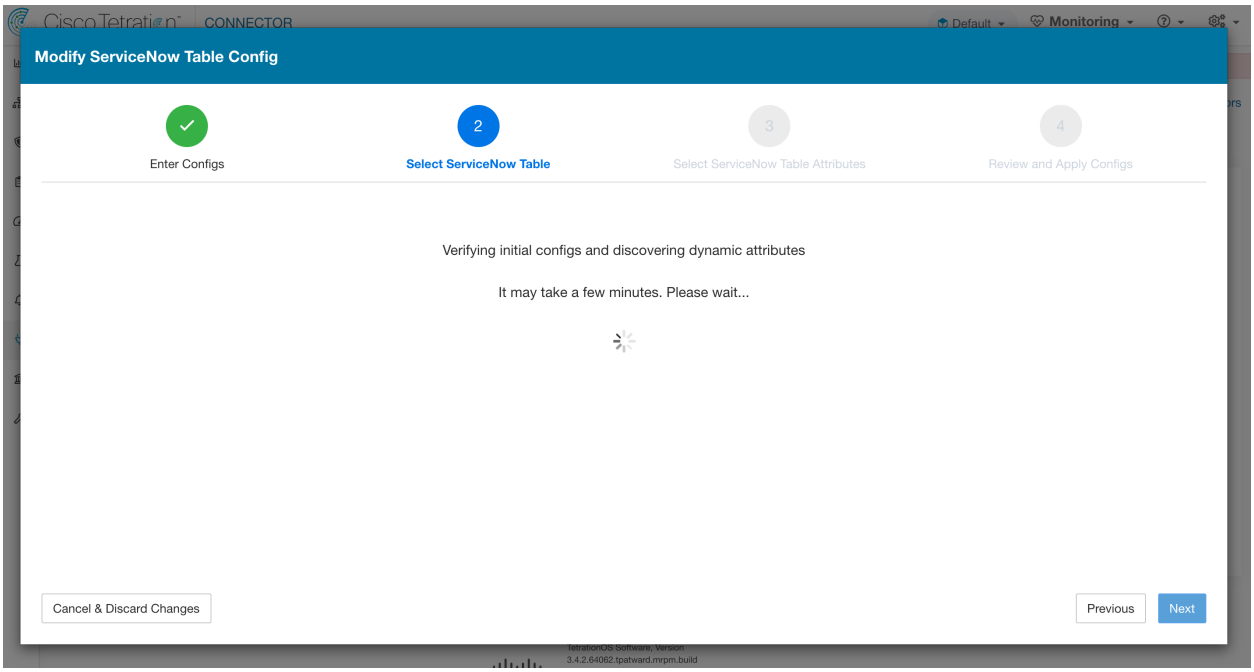


Fig. 18.1.3.1.4: Tetration Fetches the Table Info from ServiceNow Instance

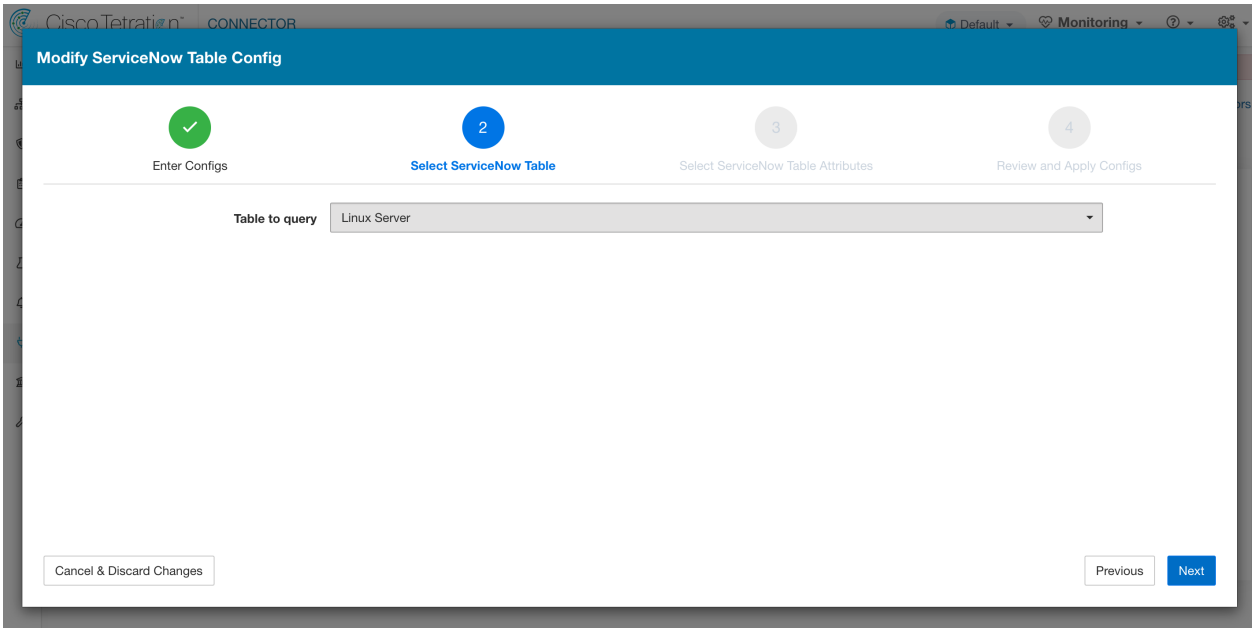


Fig. 18.1.3.1.5: Tetration presents the list of tables

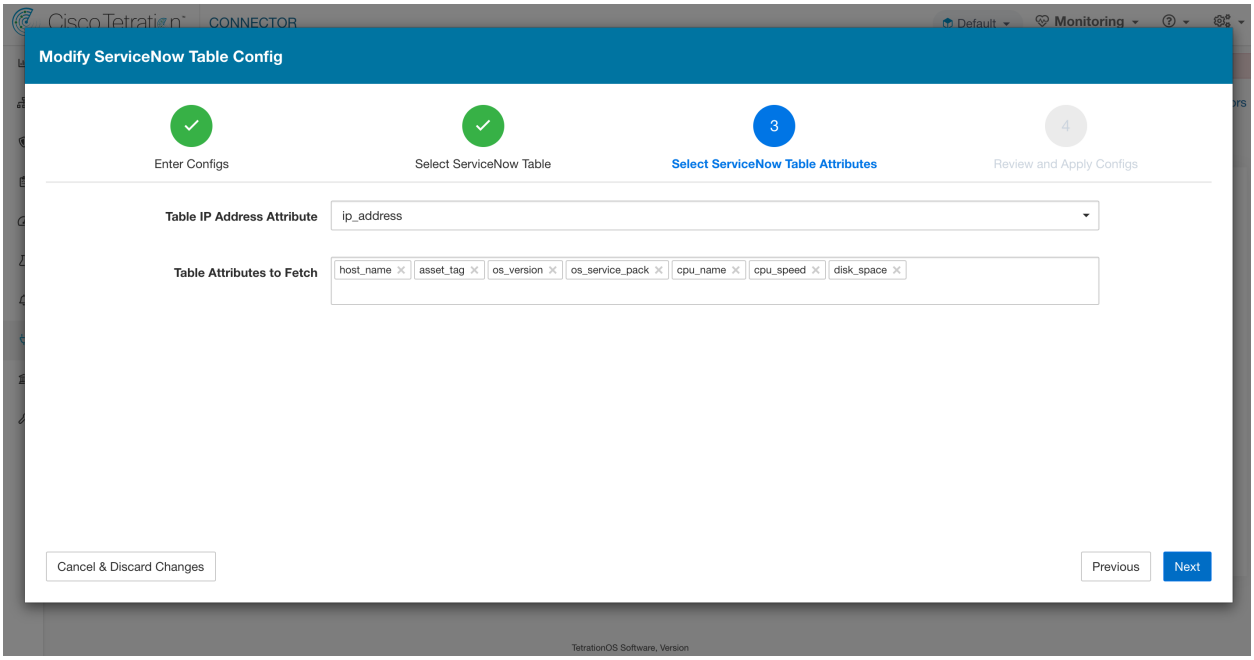


Fig. 18.1.3.1.6: User selects the ip_address attribute, and other attribute in the table

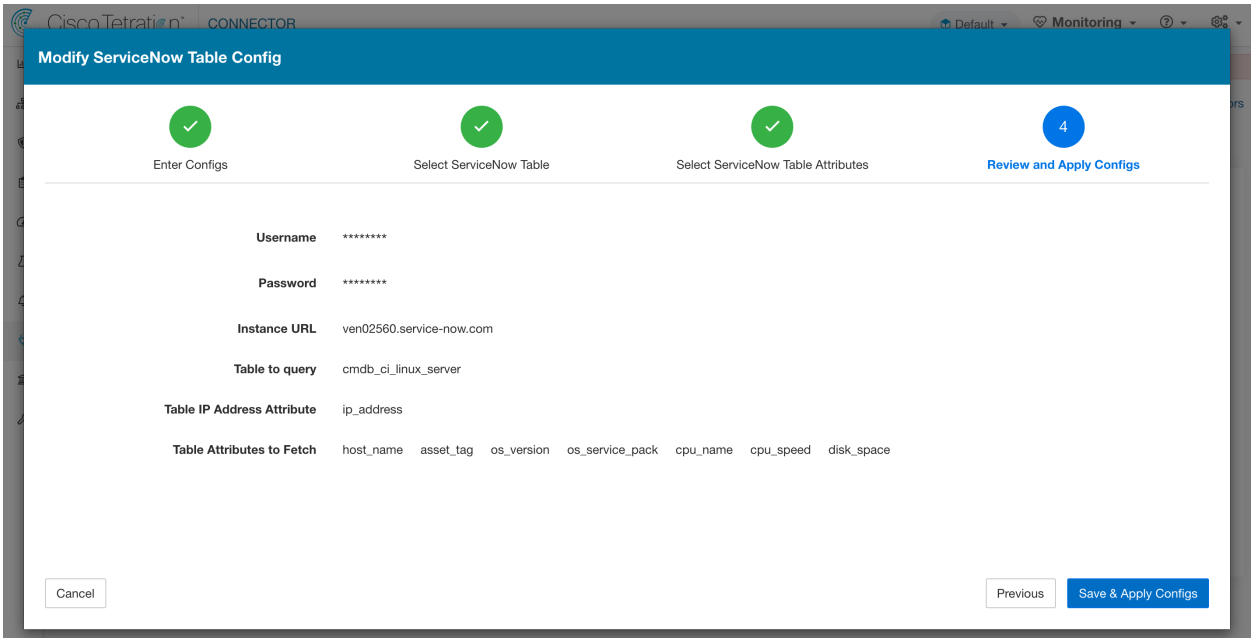


Fig. 18.1.3.1.7: User finalizes the ServiceNow config

Processing ServiceNow records

ServiceNow connector connects to ServiceNow Instance, and based on the configured Tables, it queries those tables to fetch the ServiceNow labels/metadata. Tetration annotates the ServiceNow labels to IP addresses in its inventory. ServiceNow connector periodically fetches new labels and updates Tetration inventory.

Note: Tetration fetches records from ServiceNow tables periodically. This is configurable under SyncInterval tab in the ServiceNow connector. The default sync interval is 60 minutes. For cases where integrating with ServiceNow table with large number of entries, this sync interval should be set to a higher value.

Note: Tetration will delete any entry not seen for 10 continuous sync intervals. In case the connection to ServiceNow instance is down for that long that could result in cleaning up of all labels for that instance.

Sync Interval Configuration

1. Tetration ServiceNow connector provides a way to configure the frequency of sync between Tetration and ServiceNow instance. By default the sync interval is set to 60 minutes, but it can be changed under the sync interval configuration as **Data fetch frequency**.
2. For detecting deletion of a record, Tetration ServiceNow connector relies on syncs from ServiceNow instances. If an entry is not seen in 48 consecutive sync intervals, we go ahead and delete the entry. This can be configured under sync interval config as **Delete entry interval**.

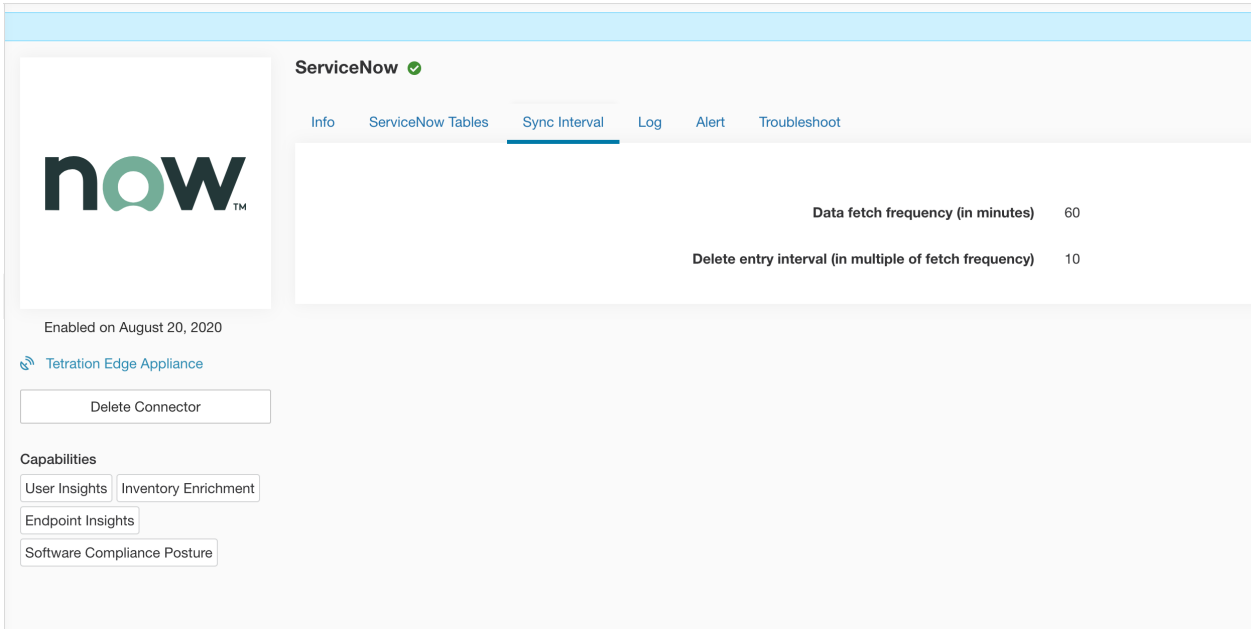


Fig. 18.1.3.1.8: Sync Interval Configuration

Explore command to delete the labels

In case user wants to cleanup the labels for a particular IP for a given instance immediately, without waiting for delete interval, they can do so using an explore command. Here are the steps to run the command.

1. Finding vrf ID for a Tenant
2. Getting to Explore command UI
3. Running the commands

For TaaS cluster, contact TaaS Operation team to cleanup labels for ServiceNow labels.

Finding VRF ID for a Tenant

Site Admins and **Customer Support users** can access the **Tenants** page under the **gears menu**. This page displays all of the currently configured Tenants and VRFs. Please refer to *Tenants* for more details.

On Tenants page, ID field of `Tenants` table is vrf ID for the Tenant.

Getting to Explore command UI

To reach Explore UI page, click on Maintenance in sidebar menu on the Tetration UI landing page. Maintenance menu drop down will have the explore tab.

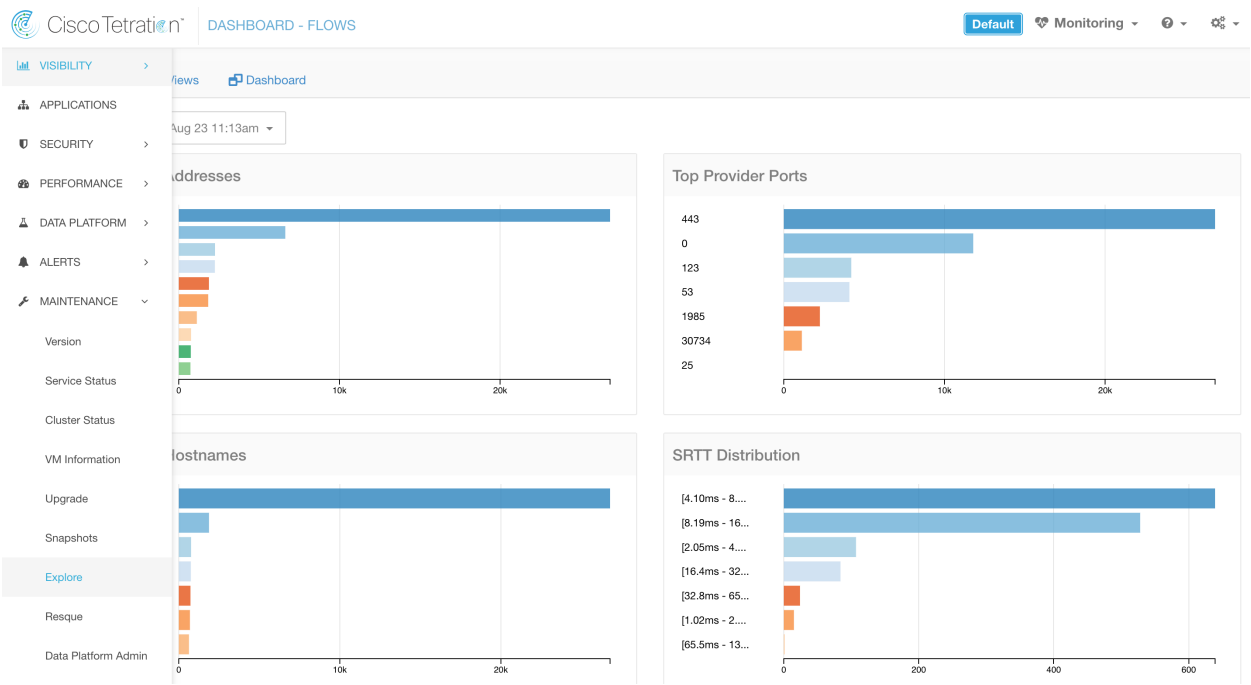


Fig. 18.1.3.1.9: Maintenance Explorer tab

Note: Customer Support privileges are required to access explore menu. If explore tab does not show up, the account may not have needed permissions.

Click on explore tab in the drop down menu to get to the Maintenance Explorer page.

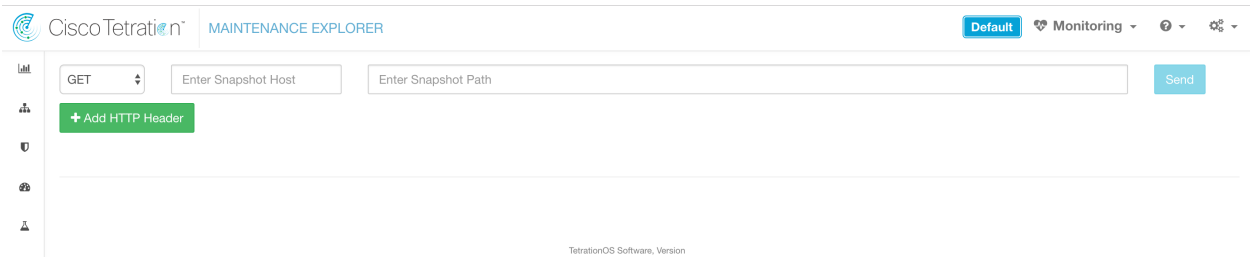


Fig. 18.1.3.1.10: Maintenance Explorer tab

Running the commands

- Choose the action as `POST`
- Enter snapshot host as `orchestrator.service.consul`
- Enter snapshot path
  - To delete the labels for a particular IP for a servicenow instance: `servicenow_cleanup_annotations?args=<vrf-id> <ip_address> <instance_url> <table_name>`
- Click Send

Note: If after deleting using explore command, we see the record show up in ServiceNow instace, it will be repopulated.

Frequently Asked Questions

1. **What if ServiceNow CMDB table does not have IP address.**

   In such case, the recommendation is to create a View on ServiceNow which will have desired fields from current table along with IP address (potentially coming from a JOIN operation with another table). Once such a view is created, it can be used in place of table name.

2. **What if ServiceNow instance requires MFA**

   Currently we do not support integrating with ServiceNow instance with MFA.

Limits

Metric	Limit
Maxumum number of ServiceNow instances that can be configured on one ServiceNow connector	20
Maximum number of attributes that can be fetched from one ServiceNow instance	10
Maximum number of ServiceNow connectors on one Tetration Edge appliance	1
Maximum number of ServiceNow connectors on one Tenant (rootscope)	1
Maximum number of ServiceNow connectors on Tetration	150

18.1.4 Connectors for Alert Notifications

Connectors for alert notifications enable Tetration to publish Tetration alerts on various messaging and logging platforms. These connectors run on TAN service on Tetration Edge Appliance.

Connector	Description	Deployed on Virtual Appliance
Syslog	Send Tetration alerts to Syslog server.	Tetration Edge
Email	Send Tetration alerts on Email.	Tetration Edge
Slack	Send Tetration alerts on Slack.	Tetration Edge
Pager Duty	Send Tetration alerts on Pager Duty.	Tetration Edge
Kinesis	Send Tetration alerts on Amazon Kinesis.	Tetration Edge

18.1.4.1 Syslog Connector

When enabled, TAN service on Tetration Edge appliance can send alerts to Syslog server using configuration.

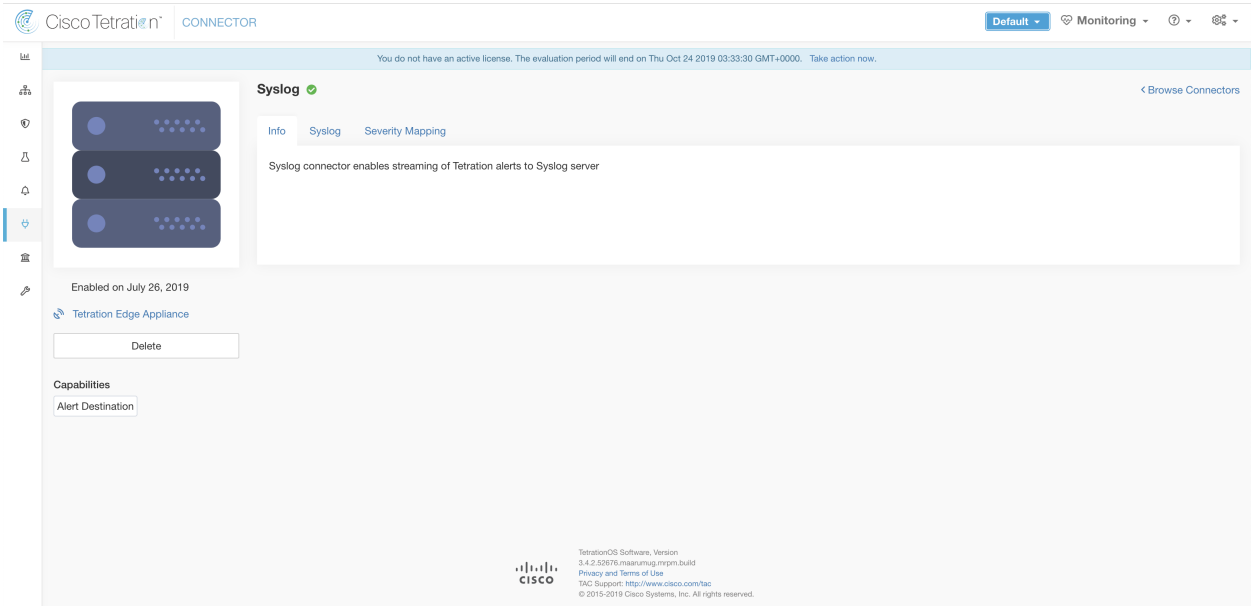


Fig. 18.1.4.1.1: Syslog connector

The following table explains the configuration details for publishing Tetration alerts on Syslog server. Please refer to *Syslog Notifier Configuration* for more details.

Parameter Name	Type	Description
Protocol	dropdown • *UDP* • *TCP*	Protocol to use to connect to server
Server Address	string	IP address or hostname of the Syslog server
Port	number	Listening port of Syslog server. Default port value is 514.

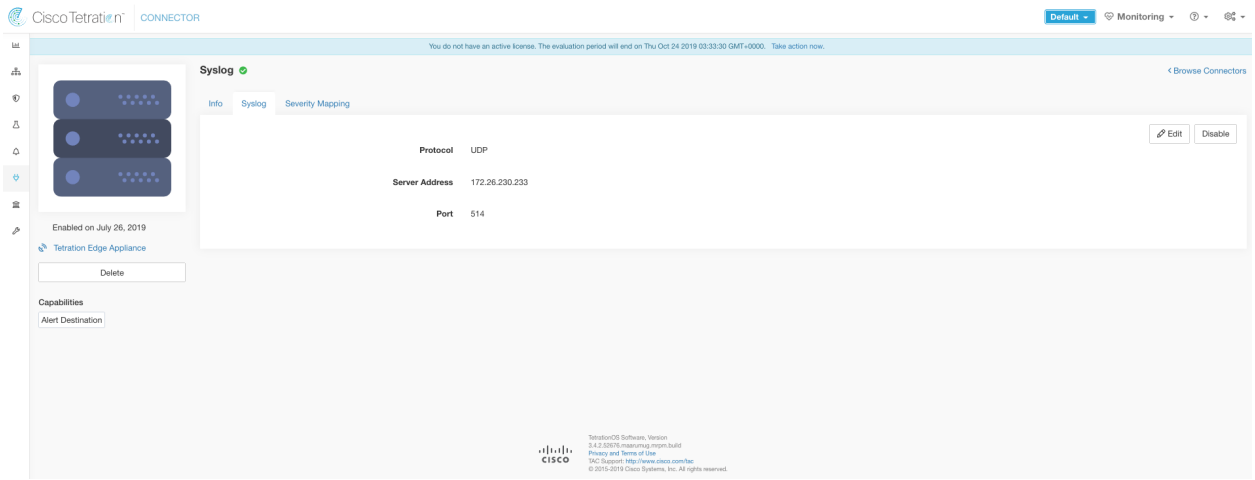


Fig. 18.1.4.1.2: Sample configuration for Syslog Connector.

Fig. 18.1.4.1.3: Sample alert.

Syslog Severity Mapping

The following table shows the default severity mapping for Tetration alerts on Syslog.

Tetration Alerts Severity	Syslog Severity
LOW	LOG_DEBUG
MEDIUM	LOG_WARNING
HIGH	LOG_ERR
CRITICAL	LOG_CRIT
IMMEDIATE ACTION	LOG_EMERG

This setting can be modified using **Severity Mapping** configuration under Syslog Connector. You can choose any corresponding Syslog priority for each Tetration Alert Severity and change the Severity Mapping. Please refer to *Syslog Severity Mapping Configuration* for more details.

Parameter Name	Dropdown of mappings
IMMEDIATE_ACTION	• *Emergency* • *Alert* • *Critical* • *Error* • *Warning* • *Notice* • *Informational* • *Debug*
CRITICAL	
HIGH	
MEDIUM	
LOW	

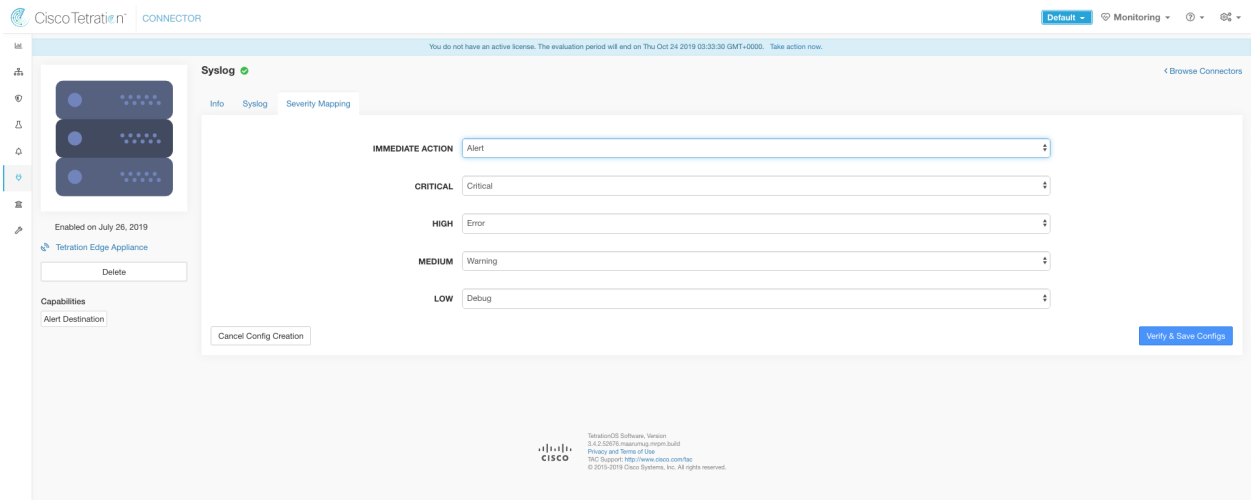


Fig. 18.1.4.1.4: Sample config for Syslog Severity Mapping.

Limits

Metric	Limit
Maximum number of Syslog connectors on one Tetration Edge appliance	1
Maximum number of Syslog connectors on one Tenant (rootscope)	1
Maximum number of Syslog connectors on Tetration	150

18.1.4.2 Email Connector

When enabled, TAN service on Tetration Edge Appliance can send alerts to given configuration.

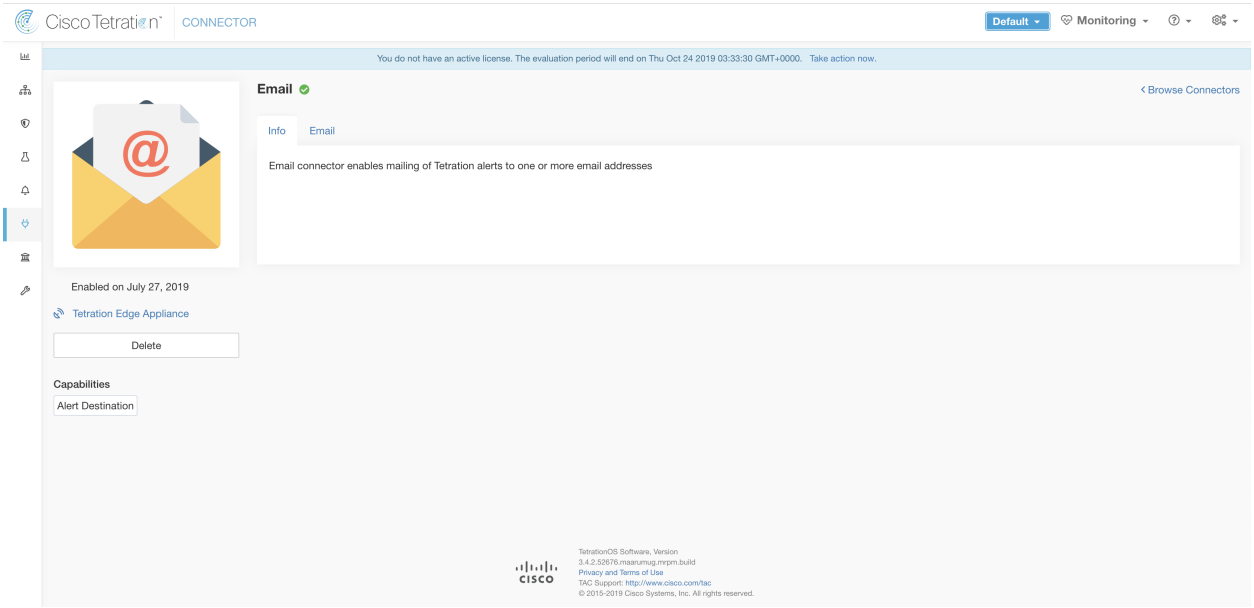


Fig. 18.1.4.2.1: Email connector

The following table explains the configuration details for publishing Tetration alerts on Email. Please refer to *Email Notifier Configuration* for more details.

Parameter Name	Type	Description
SMTP Username	string	SMTP server username. This parameter is optional.
SMTP Password	string	SMTP server password for the user (if given). This parameter is optional.
SMTP Server	string	IP address or hostname of the SMTP server
SMTP Port	number	Listening port of SMTP server. Default value is 587.
Secure Connection	checkbox	Should SSL be used for SMTP server connection?
From Email Address	string	Email address to use for sending alerts
Default Recipients	string	Comma separated list of recipient email addresses

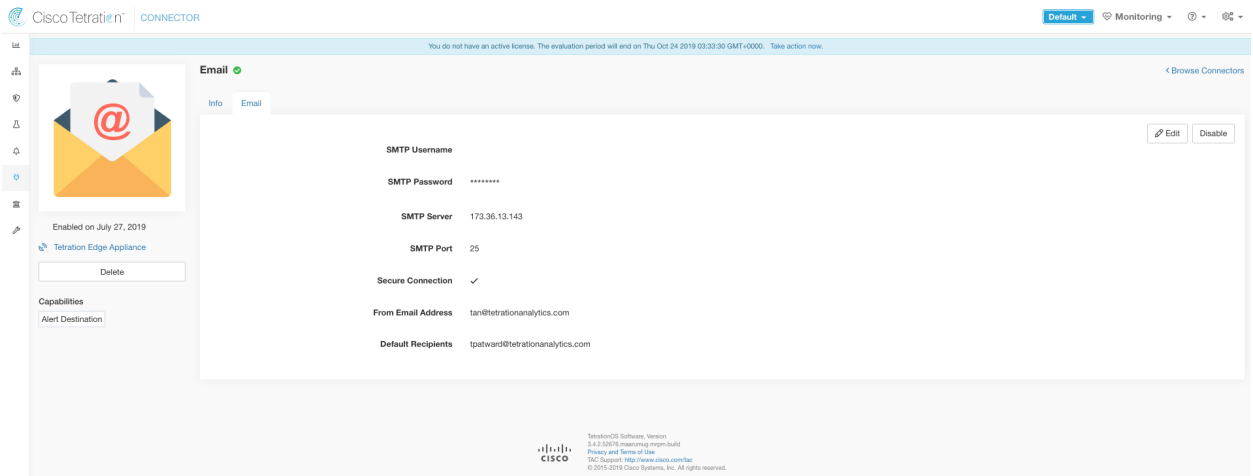


Fig. 18.1.4.2.2: Sample configuration for Email Connector.

Fig. 18.1.4.2.3: Sample alert.

Notes:

- SMTP username/password is optional. If no username is provided, we try to connect to SMTP server without any auth.
- If secure connection box is not checked, we will send alerts notification over non-secure connection.
- Default Recipients list is used to send alert notifications. This can be overridden per alert if required in Alert configuration.

Limits

Metric	Limit
Maximum number of Email connectors on one Tetration Edge appliance	1
Maximum number of Email connectors on one Tenant (rootscope)	1
Maximum number of Email connectors on Tetration	150

18.1.4.3 Slack Connector

When enabled, TAN service on Tetration Edge appliance can send alerts to Slack using configuration.

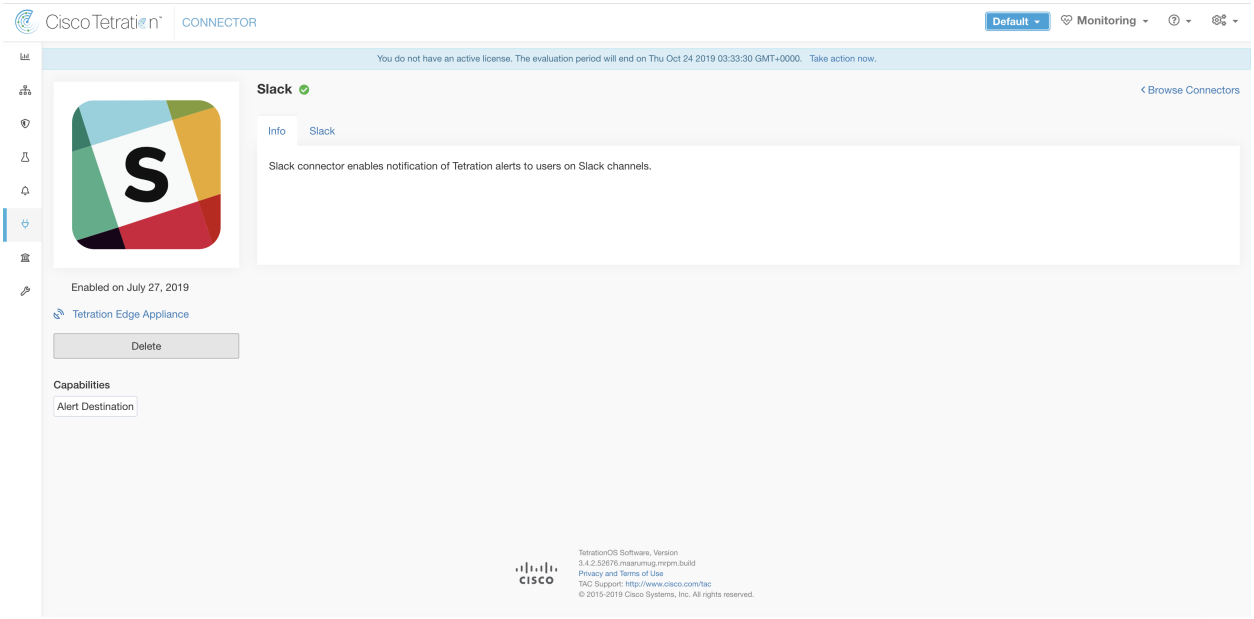


Fig. 18.1.4.3.1: Slack connector

The following table explains the configuration details for publishing Tetration alerts on Slack. Please refer to *Slack Notifier Configuration* for more details.

Parameter Name	Type	Description
Slack Webhook URL	string	Slack webhook on which Tetration alerts should be published

Note:

- To generate slack webhook go here.

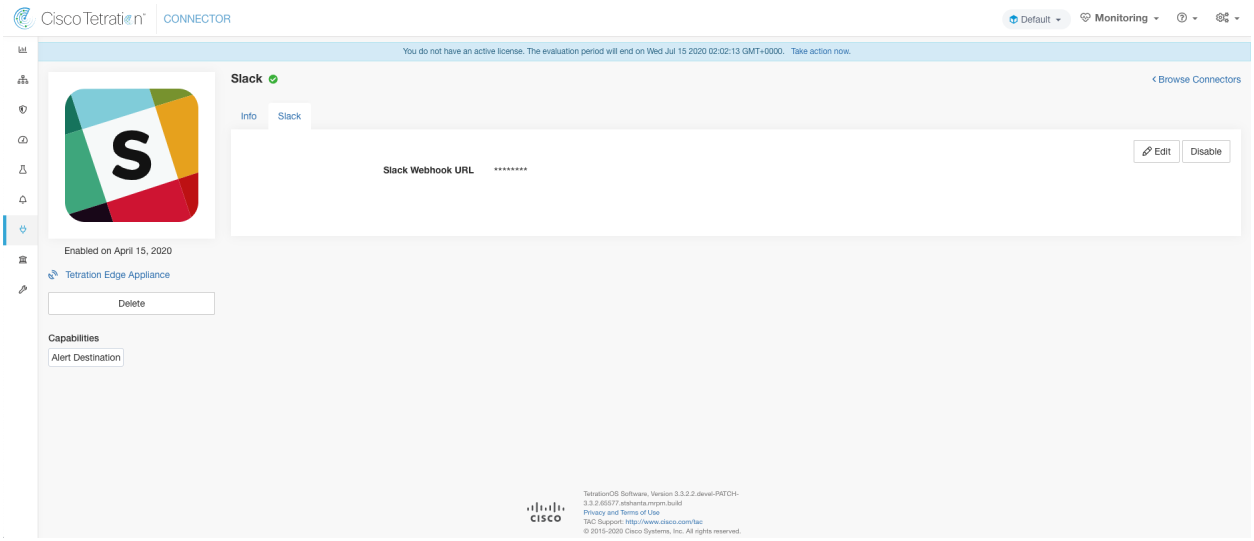


Fig. 18.1.4.3.2: Sample configuration for Slack Connector.

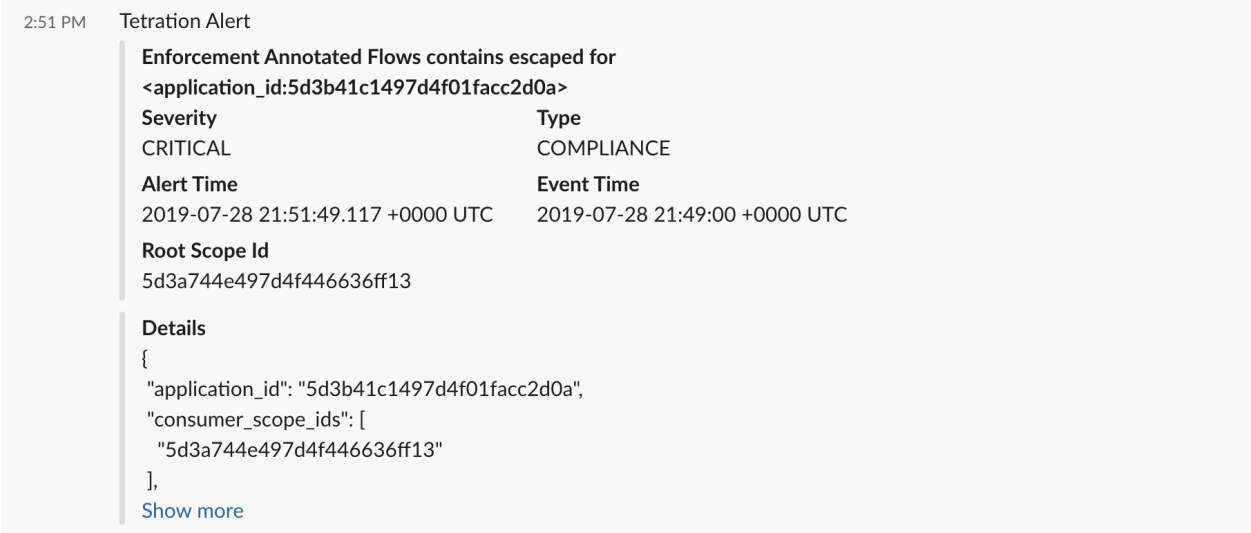


Fig. 18.1.4.3.3: Sample alert.

Limits

Metric	Limit
Maximum number of Slack connectors on one Tetration Edge appliance	1
Maximum number of Slack connectors on one Tenant (rootscope)	1
Maximum number of Slack connectors on Tetration	150

18.1.4.4 PagerDuty Connector

When enabled, TAN service on Tetration Edge appliance can send alerts to PagerDuty using configuration.

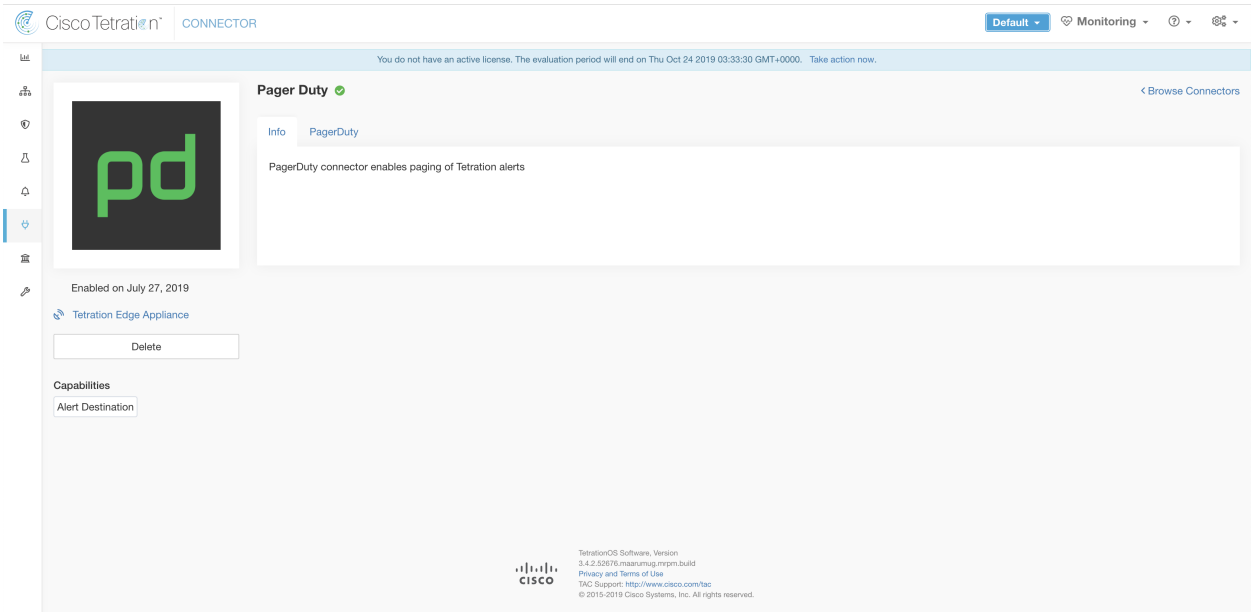


Fig. 18.1.4.4.1: PagerDuty connector

The following table explains the configuration details for publishing Tetration alerts on PagerDuty. Please refer to *PagerDuty Notifier Configuration* for more details.

Parameter Name	Type	Description
PagerDuty Service Key	string	PagerDuty service key for pushing Tetration alerts on PagerDuty

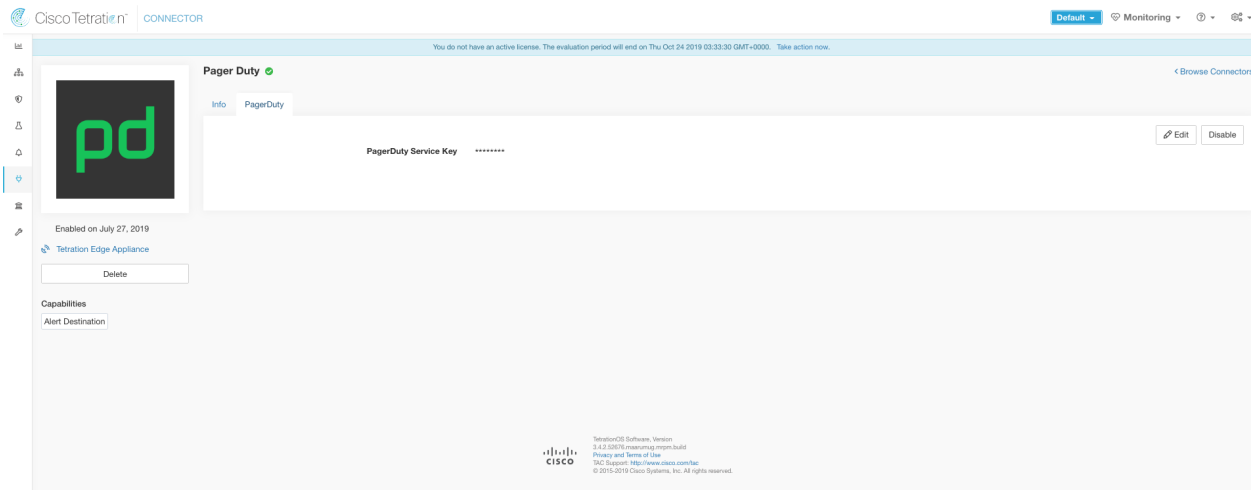


Fig. 18.1.4.4.2: Sample configuration for PagerDuty Connector.

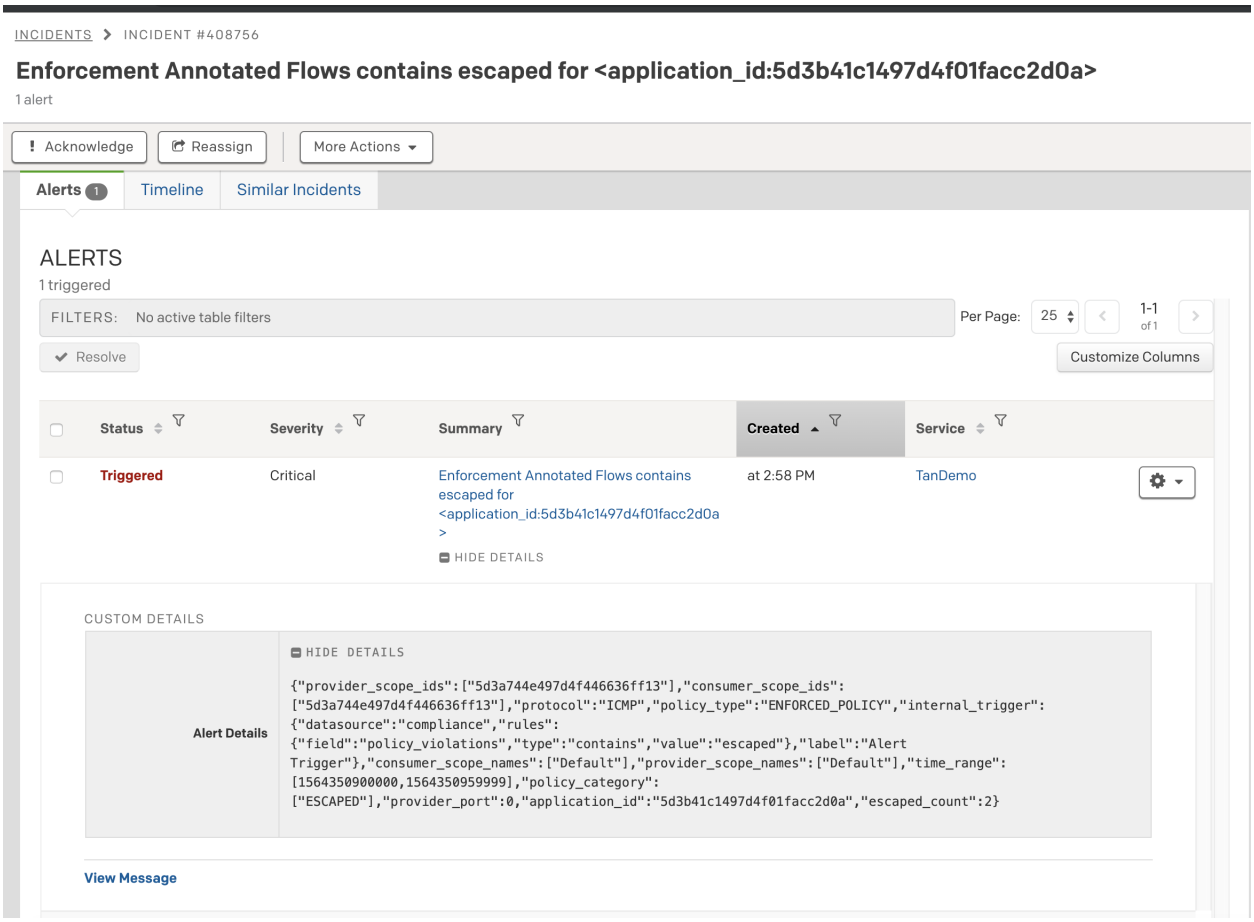


Fig. 18.1.4.4.3: Sample alert.

Limits

Metric	Limit
Maximum number of PagerDuty connectors on one Tetration Edge appliance	1
Maximum number of PagerDuty connectors on one Tenant (rootscope)	1
Maximum number of PagerDuty connectors on Tetration	150

18.1.4.5 Kinesis Connector

When enabled, TAN service on Tetration Edge appliance can send alerts using configuration.

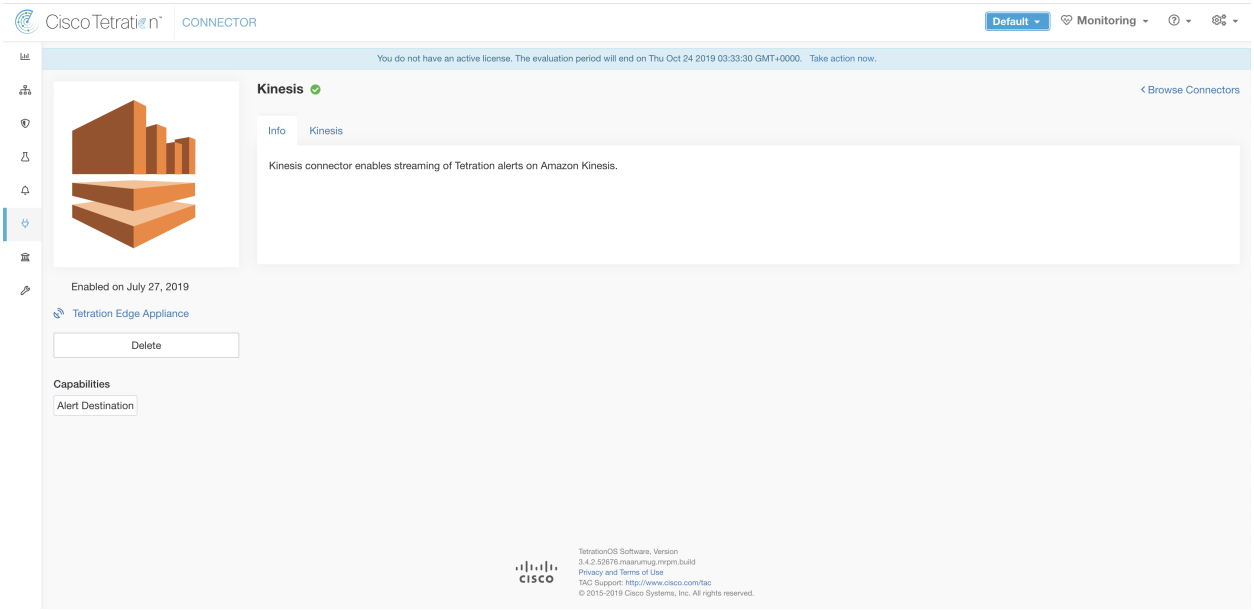


Fig. 18.1.4.5.1: Kinesis connector

The following table explains the configuration details for publishing Tetration alerts on Amazon Kinesis. Please refer to *Kinesis Notifier Configuration* for more details.

Parameter Name	Type	Description
AWS Access Key ID	string	AWS access key ID to communicate with AWS
AWS Secret Access Key	string	AWS secret access key to communicate with AWS
AWS Region	dropdown of AWS regions	Name of the AWS region where Kinesis stream is configured
Kinesis Stream	string	Name of the Kinesis stream
Stream Partition	string	Partition Name of the stream

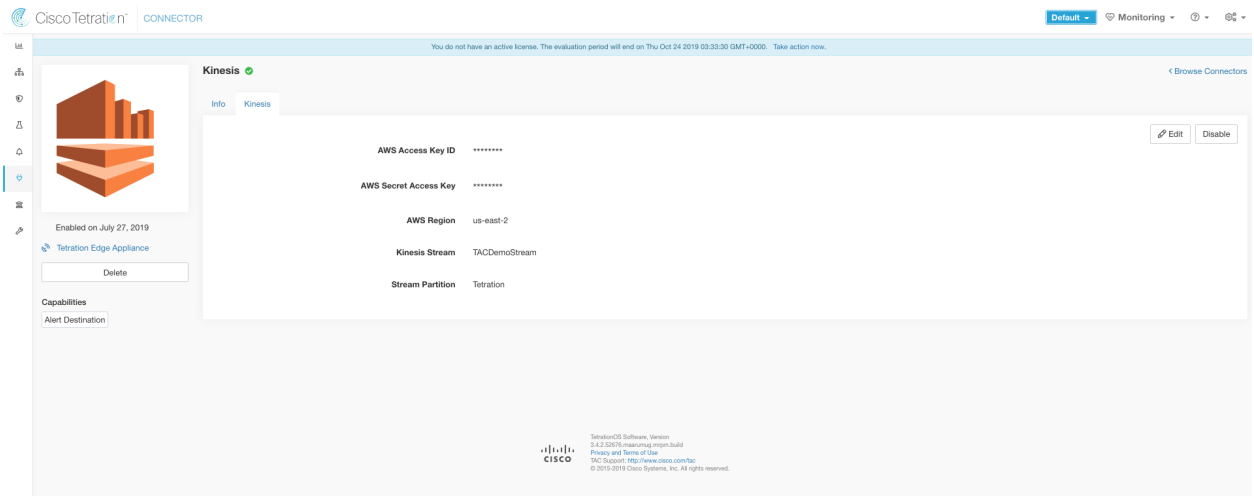


Fig. 18.1.4.5.2: Sample configuration for Kinesis Connector.

Limits

Metric	Limit
Maximum number of Kinesis connectors on one Tetration Edge appliance	1
Maximum number of Kinesis connectors on one Tenant (rootscope)	1
Maximum number of Kinesis connectors on Tetration	150

18.1.5 Connectors for Custom Visualizations and Dashboards

Connectors for custom visualizations and dashboards allow Tetration to stream annotated flow observations and active inventories to an external Logstash/Elasticsearch/Grafana stack for custom visualizations.

Connector	Description	Deployed on Virtual Appliance
Tetration Export	Export and visualize flows and inventories discovered and monitored by Tetration.	Tetration Export

Note: Tetration Export connector and appliance are both Alpha releases.

18.1.5.1 Tetration Export Connector

Data export connector enables exporting flows and inventory data from Tetration in near real-time through DataExport Managed Data Tap.

Managed Data Tap clients (Kafka Consumers) can consume/import this data to user's data lake. Alternatively, user's can be also be visualize flows and inventory data using the Tetration Export appliance.

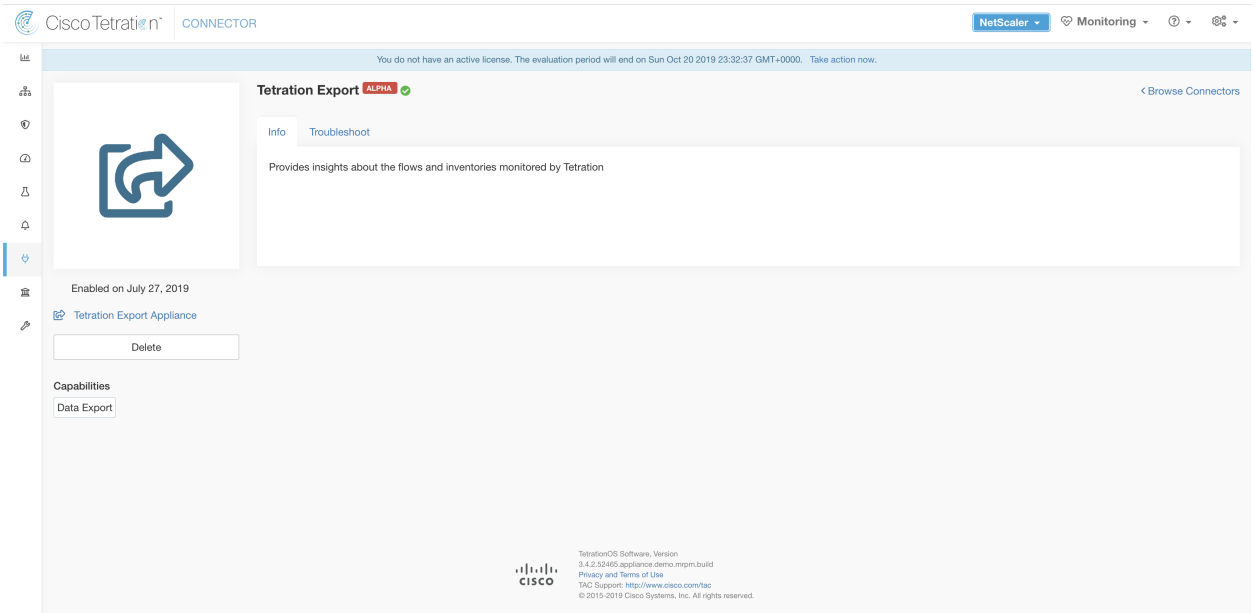


Fig. 18.1.5.1.1: Tetration Export connector

Pre-Requisites

Data export is a licensed feature. Obtain the right license for Data Export by following the instructions in the licensing page of the cluster.

For TaaS cluster, contact TaaS Operation team to enable the license. TaaS operation team can use explore commands to grant, revoke and list the data export licenses.

Data Export Tasks

Data export tasks define what will exported out of the Tetration cluster. Data is continuously exported once the data export task is created. To stop it, data export task has to be deleted. To make any changes, delete an existing data export task and create a new task with the changes.

Data export tasks can be managed through Explore commands. Here are the steps to run a command

1. Finding vrf ID for a Tenant
2. Getting to Explore command UI
3. Running the commands

For TaaS cluster, contact TaaS Operation team to manage the data export tasks

Finding VRF ID for a Tenant

Site Admins and **Customer Support users** can access the **Tenants** page under the **gears menu**. This page displays all of the currently configured Tenants and VRFs. Please refer to *Tenants* for more details.

On Tenants page, ID field of `Tenants` table is vrf ID for the Tenant.

Getting to Explore command UI

To reach Explore UI page, click on Maintenance in sidebar menu on the Tetration UI landing page. Maintenance menu drop down will have the explore tab.

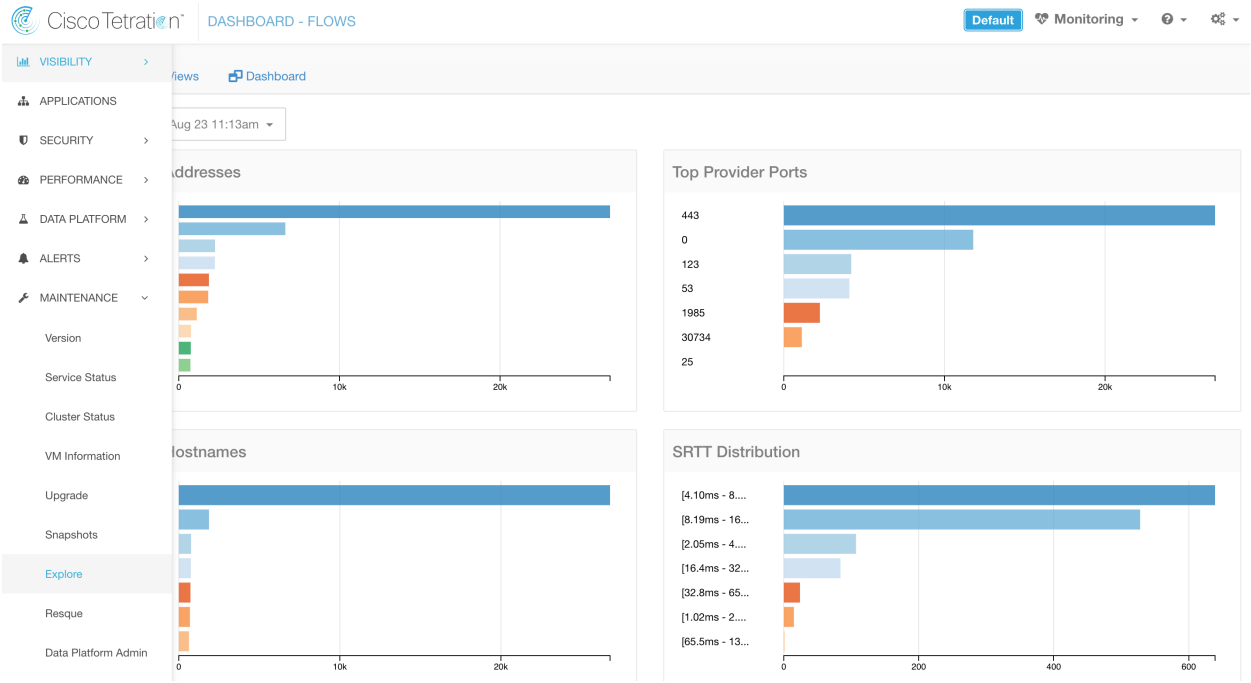


Fig. 18.1.5.1.2: Maintenance Explorer tab

Note: Customer Support privileges are required to access explore menu. If explore tab does not show up, the account may not have needed permissions.

Click on explore tab in the drop down menu to get to the Maintenance Explorer page.

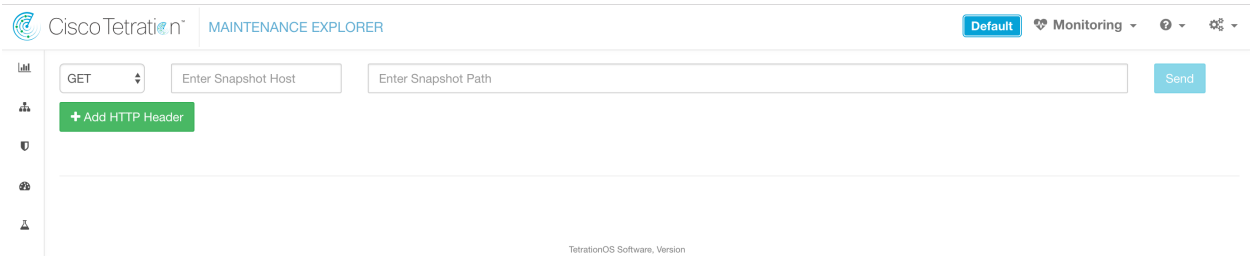


Fig. 18.1.5.1.3: Maintenance Explorer tab

Running the commands

- Choose the action as `POST`

- Enter snapshot host as `orchestrator.service.consul`
- Enter snapshot path
  - To list data export tasks: `list_data_export?args=<vrf-id>`
  - To create a data export task: `create_data_export?args=<vrf-id> <data-source>`. Two supported data sources are: `aggregated_flows` and `active_inventory`
  - To delete a data export task: `delete_data_export?args=<task-id>` (delete is based on the task-id, run `list_data_export` command and `id` field in the response is `task_id`)
- Click Send

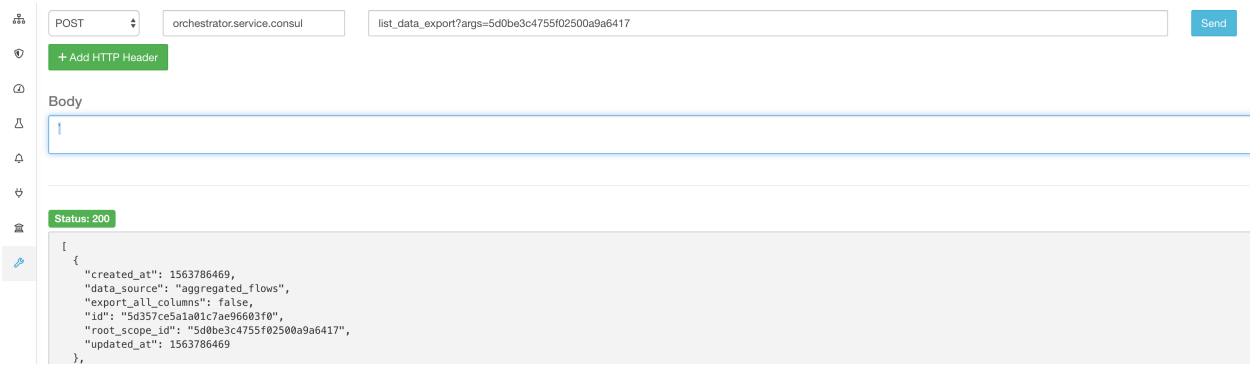


Fig. 18.1.5.1.4: List Data Export Tasks

Tetration Export Connector and Virtual Appliance

Note: The **Tetration Export Connector** and **Tetration Export Virtual Appliance** deployment is optional. To consume data exported from Tetration into an existing ingest pipeline, read data directly from the *Data Export Managed Data Tap (MDT)*.

Description

The **Tetration Export Connector** and **Tetration Export Virtual Appliance** together provide a turn-key dashboard platform that enables customizable insight into Tetration learned and enriched data. The **Tetration Export Virtual Appliance** hosts **Elasticsearch**, **Logstash**, and **Grafana**. Logstash is automatically configured to consume data from the **Data Export Managed DataTap (MDT)**.

Specifications

- 32 vCPU's
- 64GB Memory
- 250GB Storage (SSD preferred)
- vSphere 5.5+

Limitations

- The **Tetration Export Connector** runs on a dedicated **Tetration Export Virtual Appliance** that may not be shared with any other connectors.
- One **Tetration Export Virtual Appliance** may be deployed per **Tenant**.
- The rate at which data is exported may not exceed `1.5 million` events per minute.
- Data retention within the **Tetration Export Virtual Appliance** is limited to `65 minutes`.

Tetration Export Connector Deployment

1. Navigate to the **Connectors** page and **Browse** available connectors.
2. Locate and click the `Tetration Export` connector, then hit: `Enable`.

You will be prompted to first deploy a **Tetration Export Virtual Appliance**. After the **Tetration Export Virtual Appliance** has been deployed the **Tetration Export Connector** will automatically be deployed on the **Tetration Export Virtual Appliance**.

Tetration Export Appliance Deployment

1. Use the OVA image downloaded from Cisco Software Download page to deploy a new OVF template on the designated ESXi host.

- vSphere 5.5+ is supported. You may follow the instructions here for deploying an OVF template.
- Make sure the VM network settings match the recommended configuration for Tetration Export appliance.
- **Do not boot the VM yet!**

1. Upload the VM Configuration Bundle (iso image) to the datastore corresponding to the target ESXi host.
2. Go to **Edit VM settings** and mount the Configuration Bundle (iso image) from the datastore. Please make sure to select **Connect at Power On** checkbox.
3. Power on the deployed VM
4. Wait for the virtual appliance to connect back to Tetration. This may take a few minutes. The appliance status should change from **Pending Registration** to **Active**.

You may monitor the appliance status on the virtual appliance management page after clicking **Done** button below.

Note: Please refer to *Tetration Virtual Appliances for Connectors* for more information on deploying, monitoring, and troubleshooting an external appliance.

Accessing the Tetration Export Virtual Appliance

After the **Tetration Export Virtual Appliance** is **Active**, the Grafana interface is accessible via **HTTP** on the IP assigned to the appliance at **TCP/3000**.

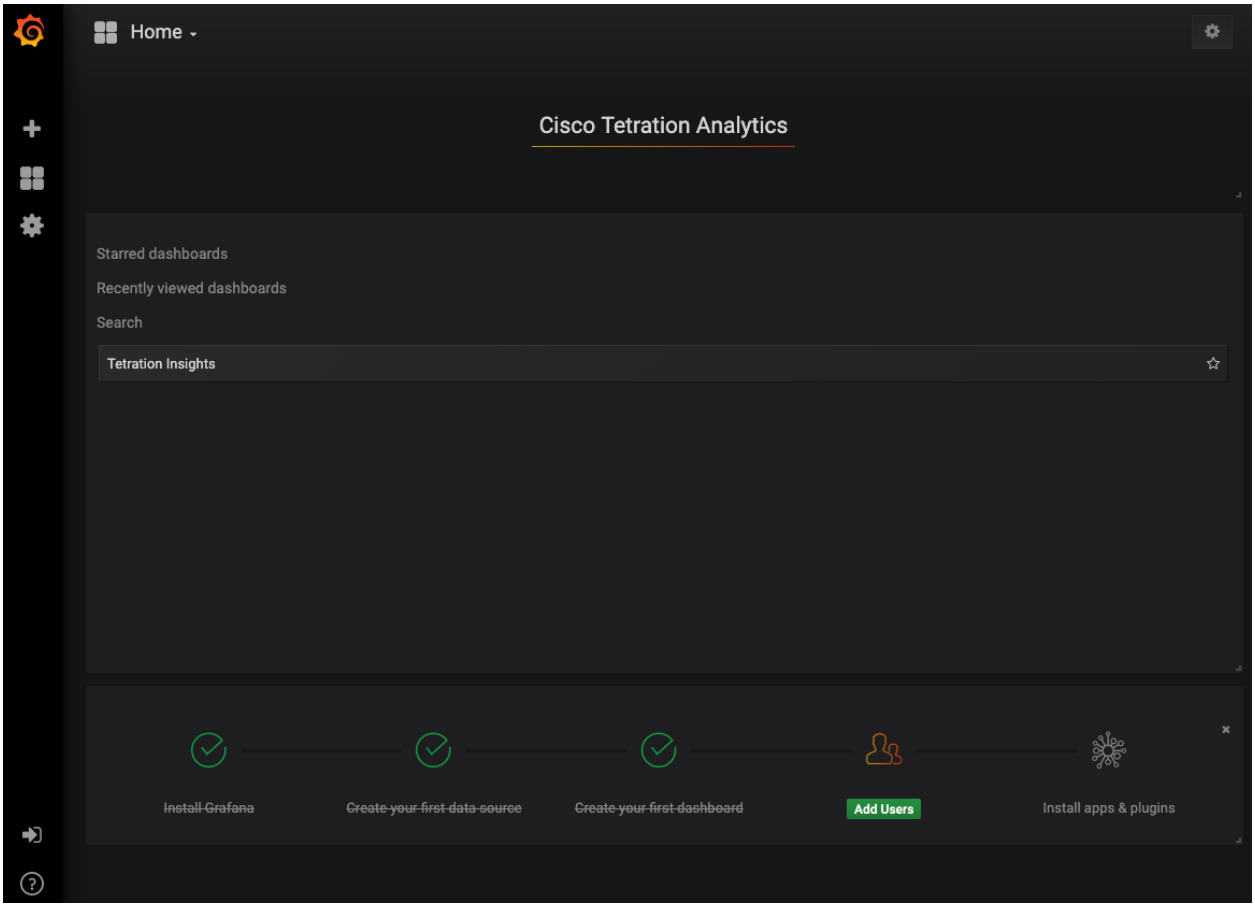


Fig. 18.1.5.1.5: Grafana landing page on Tetration Export appliance

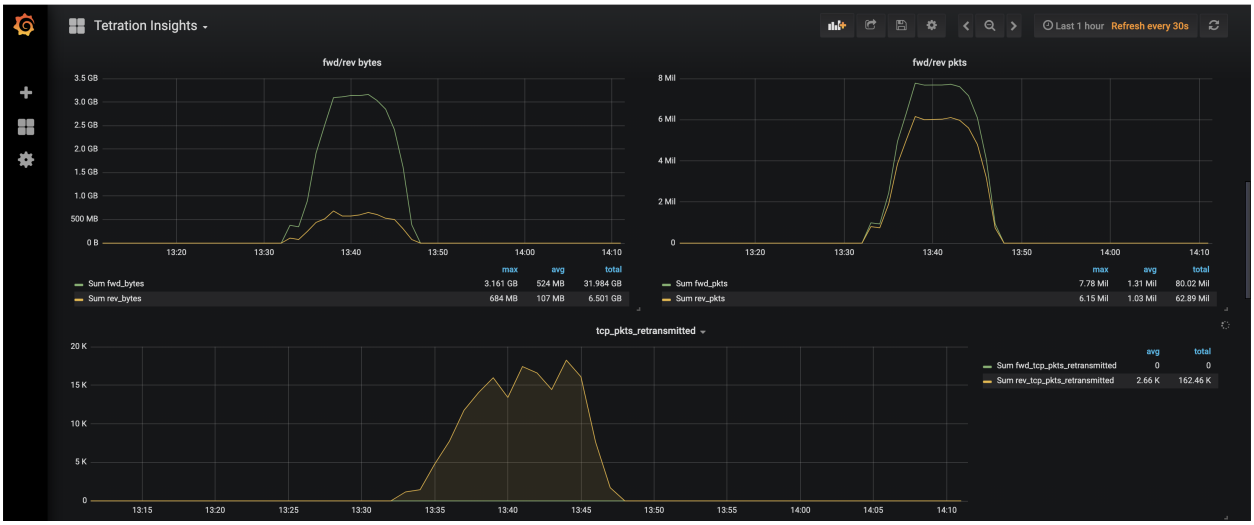


Fig. 18.1.5.1.6: Grafana charts showing data from Tetration

Reading data from Data Export Managed Data Tap (MDT)

Data can be read from DataExporter Managed Data Tap (MDT) by using the Tetration provided Certs. Please refer to *Managed Data Taps* for more details. User can download the certificates from Tetration UI. The certificates are available in two formats: (1) Java Keystore (JKS) based certs that can be used by Java and Scala clients and (2) regular cert files which can be used by clients written in Golang, Python, and other languages. Click on the download button to download the file

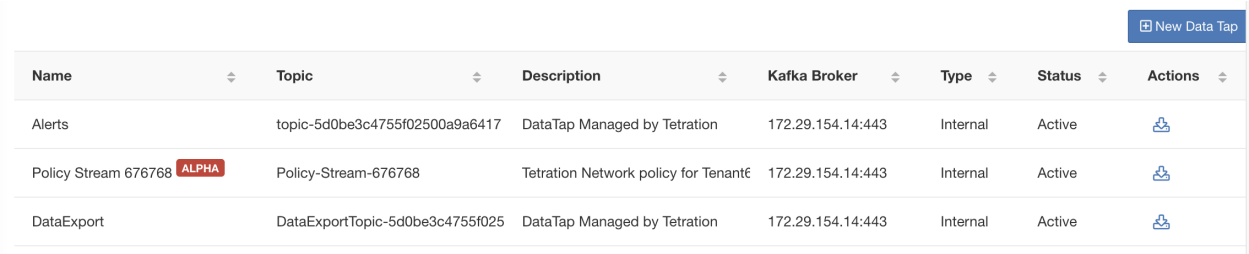


Fig. 18.1.5.1.7: List of Managed Data Taps

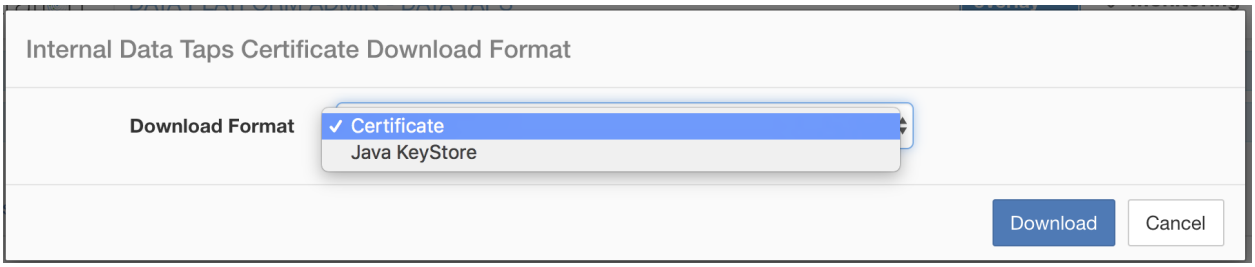


Fig. 18.1.5.1.8: Download the certificate for DataExport MDT

Java Keystore

Upon downloading the DataExportCerts.jks.tar.gz, user you should see the following files that contain information to connect to Tetration MDT to receive messages:

1. kafkaBrokerIps.txt - This file contains the IP address string, that Kafka client should use to connect to Tetration MDT.
2. topic.txt - This file contains the name of the topic from which client can read the messages.
3. keystore.jks - Keystore the Kafka Client should use in the connection settings shown below.
4. truststore.jks - Truststore the Kafka Client should use in the connection settings shown below.
5. passphrase.txt - This file contains the password to be used for #3 and #4.

Following the Kafka settings should be used while setting up Consumer.properties (Java client) that uses the keystore and truststore:

```
security.protocol=SSL
ssl.truststore.location=<location_of_truststore_downloaded>
ssl.truststore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.keystore.location=<location_of_truststore_downloaded>
ssl.keystore.password=<passphrase_mentioned_in_passphrase.txt>
ssl.key.password=<passphrase_mentioned_in_passphrase.txt>
```

Following set of Properties should be used while setting up the Kafka Consumer in Java code:

```
Properties props = new Properties();
props.put("bootstrap.servers", brokerList);
props.put("group.id", DataExportConsumerGroup-<id>);
props.put("key.deserializer",   "org.apache.kafka.common.serialization.
↪StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.
↪StringDeserializer");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("security.protocol", "SSL");
props.put("ssl.truststore.location", "<filepath_to_truststore.jks>");
props.put("ssl.truststore.password", passphrase);
props.put("ssl.keystore.location", <filepath_to_keystore.jks>);
props.put("ssl.keystore.password", passphrase);
props.put("ssl.key.password", passphrase);
props.put("zookeeper.session.timeout.ms", "500");
props.put("zookeeper.sync.time.ms", "250");
props.put("auto.offset.reset", "earliest");
```

Certificate

If end user wants to use Certificates, they can use Go clients using *Sarama* Kafka library to connect to Tetration MDT. Upon downloading DataExportCert.cert.tar.gz, user should see the following files:

1. kafkaBrokerIps.txt - This file contains the IP address string that Kafka Client should use to connect to Tetration MDT
2. topic.txt - This file contains the name of the topic from which client can read the messages.
3. KafkaConsumerCA.cert - This file contain the KafkaConsumer certificate.
4. KafkaConsumerPrivateKey.key - This file contains the Private Key for the Kafka Consumer.
5. KafkaCA.cert - This file should be used in the root CA certs listing in the Go client.

See the following example of Go Client to connect to Tetration MDT for consuming alerts from Tetration.

Sample Go Client to consume alerts from MDT

Limits

Metric	Limit
Maximum number of Export connectors on one Tetration Export appliance	1
Maximum number of Export connectors on one Tenant (rootscope)	1
Maximum number of Export connectors on Tetration	35

18.2 Tetration Virtual Appliances for Connectors

Connectors are deployed on Tetration virtual appliances. The virtual appliance OVA templates can be downloaded from Cisco Software Download page. Virtual appliances can be deployed from these templates on an ESXi host in VMware vCenter.

18.2.1 Types of Virtual Appliances

Each connector supported by Tetration can be deployed on one of three types of virtual appliances.

18.2.1.1 Tetration Ingest

Tetration Ingest appliance is a software appliance that can export flow observations to Tetration from various connectors.

Specification

- Number of CPU cores: 8
- Memory: 8 GB
- Storage: 250 GB
- Number of network interfaces: 3
- Number of connectors on one appliance: 3

Note: Each root scope on Tetration can have at most 100 Tetration Ingest appliances deployed.

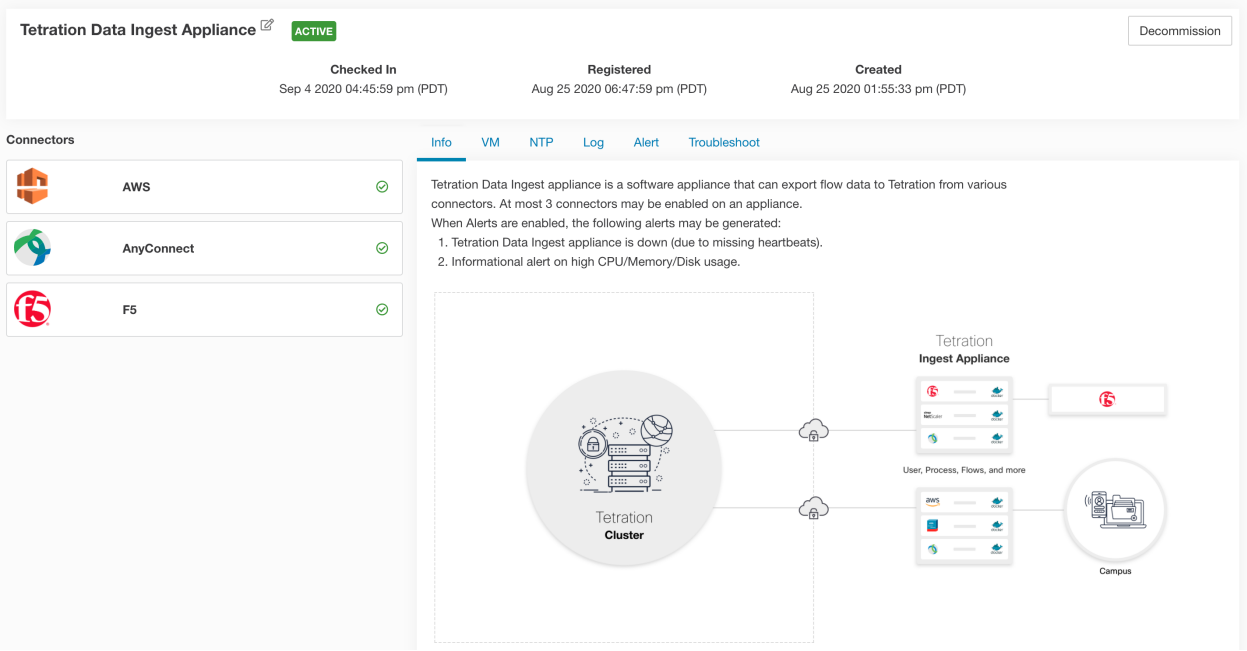


Fig. 18.2.1.1.1: Tetration Ingest appliance

Tetration Ingest appliance allows at most 3 connectors to be enabled on an appliance. There can be more than one instance of the same connector enabled on the same appliance. Many of the connectors deployed on Ingest appliance collects telemetry from various points in the network, these connectors need to listen on specific ports on the appliance. Each connector is therefore bound to one of the IP address and the default ports on which the connector should be listening to collect telemetry data. As a result, each IP address is essentially a slot that a connector occupies on the appliance. When a connector is enabled, a slot is taken (thereby, the IP corresponding to the slot). And, when a connector is disabled, the slot occupied by the connector is released (thereby, the IP corresponding to the slot). Please refer to *Tetration Ingest appliance slots* for a description of how Ingest appliance maintains the state of the slots.

```
[root@beretta-ingest-1 tetter]# cat /local/tetration/appliance/appliance.conf
{
  "type": "TETRATION_DATA_INGEST",
  "slots": [
    {
      "available": false,
      "index": 0,
      "mapped_ip": "172.29.142.26",
      "share_volume": true,
      "count": 1,
      "service_containers": {
        "5d379fac6e37d85f2bdeff45": {
          "connector_id": "5d379fac6e37d85f2bdeff44",
          "service_id": "5d379fac6e37d85f2bdeff45",
          "container_id": "2c7a7ed4f853e85f3d620c663f1c7f5395b53b9dd6696276ac439d34fe142bf1",
          "image_name": "netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45",
          "container_name": "nf-5d379fac6e37d85f2bdeff45",
          "service_type": "NETFLOW_SENSOR",
          "ip_bindings": [
            {
              "ip": "172.29.142.26",
              "port": "4729",
              "protocol": "udp"
            },
            {
              "ip": "172.29.142.26",
              "port": "4739",
              "label": 1,
              "protocol": "udp"
            }
          ],
          "volume_id": "373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439"
        }
      }
    },
    {
      "available": true,
      "index": 1,
      "mapped_ip": "172.29.142.27",
      "share_volume": true,
      "count": 0,
      "service_containers": null
    },
    {
      "available": true,
      "index": 2,
      "mapped_ip": "172.29.142.28",
      "share_volume": true,
      "count": 0,
      "service_containers": null
    }
  ]
}[root@beretta-ingest-1 tetter]#
```

Fig. 18.2.1.1.2: Tetration Ingest appliance slots

Allowed Configurations

- *NTP*: Configure NTP on the appliance. Please refer to *NTP Configuration* for more details.
- *Log*: Configure Logging on the appliance. Please refer to *Log Configuration* for more details.

18.2.1.2 Tetration Edge

Tetration Edge is a control appliance that streams alerts to various notifiers and collects inventory metadata from network access controllers such as Cisco ISE. In a Tetration Edge appliance, all alert notifier connectors (such as Syslog, Email, Slack, PagerDuty and Kinesis), ServiceNow connector, Workload AD connector and ISE connector can be deployed.

Specification

- Number of CPU cores: 8
- Memory: 8 GB
- Storage: 250 GB
- Number of network interfaces: 1
- Number of connectors on one appliance: 8

Note: Each root scope on Tetration can have at most one Tetration Edge appliance deployed.

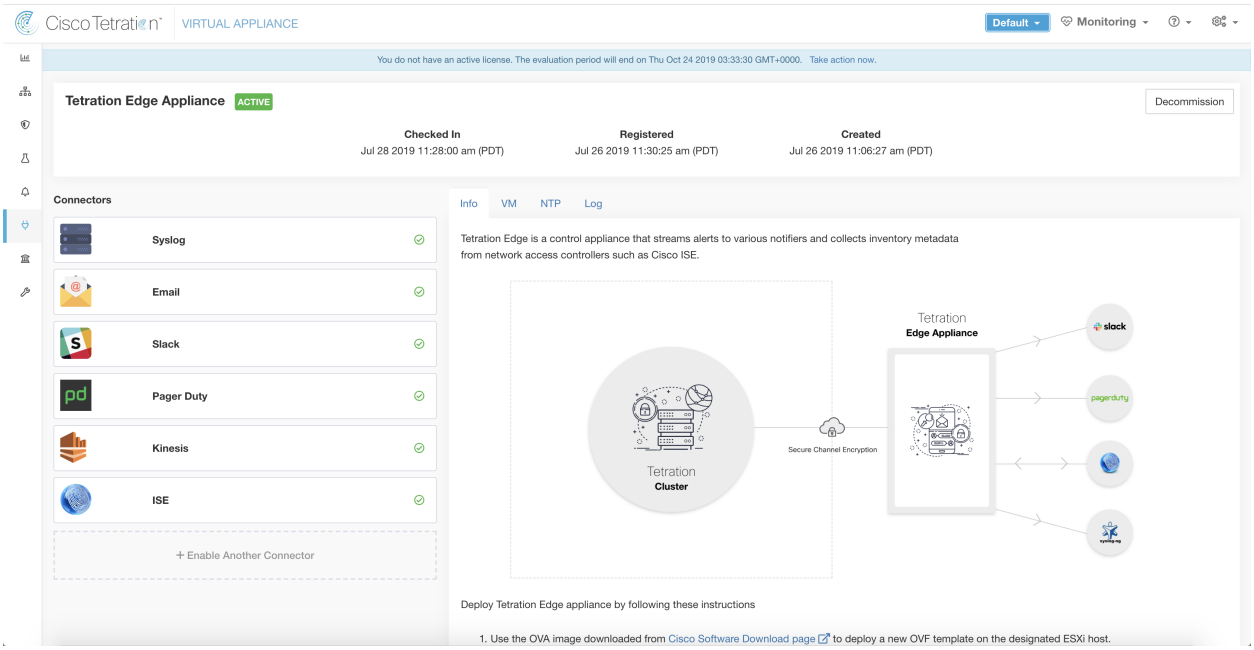


Fig. 18.2.1.2.1: Tetration Edge appliance

The connectors deployed on Tetration Edge appliance do not listen on ports. Therefore, the Docker containers instantiated for the connectors on Tetration Edge appliance do not expose any ports to the host.

Allowed Configurations

- *NTP*: Configure NTP on the appliance. Please refer to *NTP Configuration* for more details.
- *Log*: Configure Logging on the appliance. Please refer to *Log Configuration* for more details.

18.2.1.3 Tetration Export

Tetration Data Export is a software appliance for building/visualizing specialized dashboards for flow/inventory data enriched by Tetration.

Specification

- Number of CPU cores: 32
- Memory: 64 GB
- Storage: 250 GB
- Number of network interfaces: 1
- Number of connectors on one appliance: 1

Note: Each root scope on Tetration can have at most one Tetration Export appliance deployed.

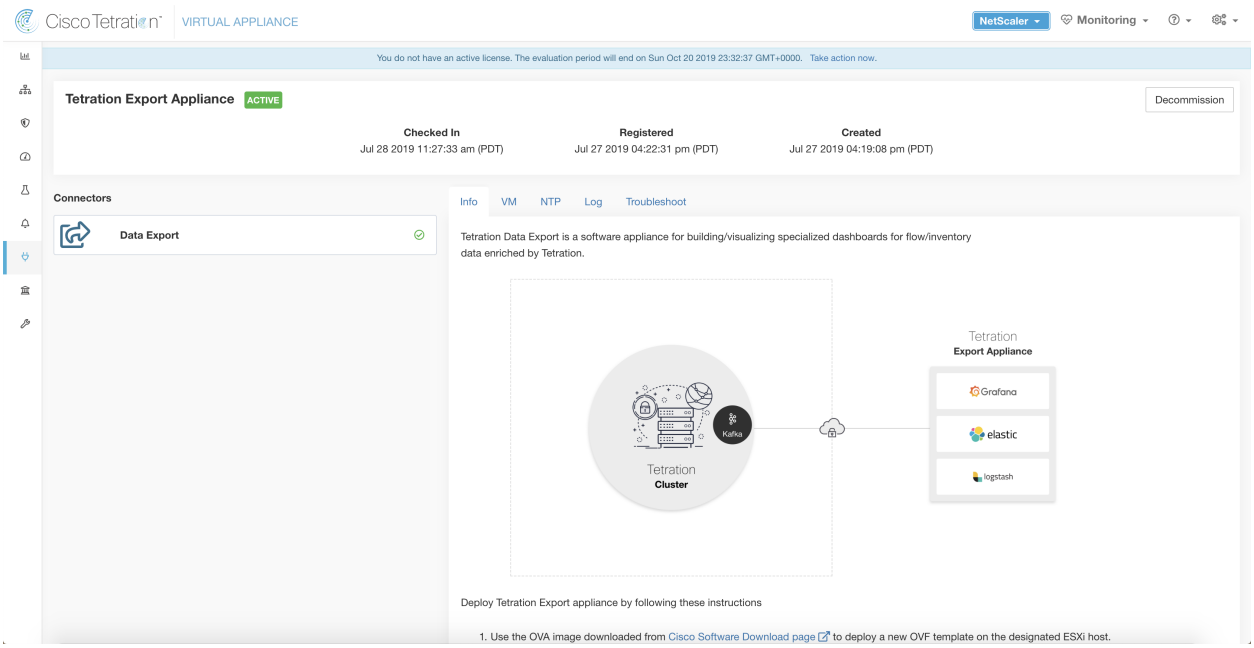


Fig. 18.2.1.3.1: Tetration Export appliance

Tetration Export connector is the only connector deployed on Tetration Export appliance. This connector is deployed implicitly on the appliance when the appliance comes up.

Allowed Configurations

- *NTP*: Configure NTP on the appliance. Please refer to *NTP Configuration* for more details.
- *Log*: Configure Logging on the appliance. Please refer to *Log Configuration* for more details.

18.2.2 Deploying a Virtual Appliance

Attention: To deploy a Tetration external appliance, the ESXi host where the appliance is created should have the following specifications.

- **vSphere**: version 5.5 or better.
- **CPU**: at least 2.2 GHz per core, and has enough reservable capacity for the appliance.
- **Memory**: at least enough space to fit the appliance.

The following steps outline how to deploy a virtual appliance.

1. Enable a connector on a new virtual appliance and follow the workflow to configure the appliance and download a VM configuration bundle (ISO file).
2. Download the OVA template for the virtual appliance from Cisco Software Download page. Please refer to the screenshot for *Deploying a Tetration Ingest appliance*.
3. Use the OVA downloaded to deploy a new OVF template on a designated ESXi host.
   - Please follow Deploy an OVF Template for instructions on how to deploy an OVA on a vSphere Web Client.
   - Ensure that the deployed VM settings match the recommended configuration for the virtual appliance type.
   - **Do not power on the deployed VM** yet.
4. Configure the virtual appliance by providing IP address(es), gateway(s), hostname, DNS, proxy server settings and docker bridge subnet configuration. Please refer to the screenshot for *Configuring the VM with network parameters*.
   - If the appliance needs to use proxy server to reach Tetration, please check the box *Use proxy server to connect to Tetration*. If this is not set correctly, connectors may not be able to communicate with Tetration for control messages, register connectors, and send flow data to Tetration collector.
   - If the IP address(es) and gateways(s) of the appliance conflict with the default docker bridge subnet (*172.17.0.1/16*), the appliance can be configured with a customized docker bridge subnet specified in *Docker Bridge (CIDR format)* field. This requires appliance OVA 3.3.2.16 or later.
5. In the next step, a VM configuration bundle will be generated and available for download. Download the VM configuration bundle. Please refer to the screenshot for *Download the VM configuration bundle*.
6. Upload the VM configuration bundle to the datastore corresponding to the target ESXi host.
7. Edit the VM settings and mount the VM configuration bundle from the datastore to the CD/DVD drive. Please make sure to select **Connect at Power On** checkbox.
8. Power on the deployed VM.
9. Once the VM boots up and configures itself, it will connect back to Tetration. This may take a few minutes. The appliance status on Tetration should transition from *Pending Registration* to *Active*. Please refer to the screenshot for *Tetration Ingest appliance in Pending Registration state*

Note: We do not recommend vMotion to be enabled for Tetration external appliances.

Note: We recommend to use Tetration external appliance OVAs as-is to deploy VMs. Please do not change the reservations for CPU and memory when deploying the VM. If sufficient resources are not available, the VM will not

boot and vCenter will display an error message similar to the following message.

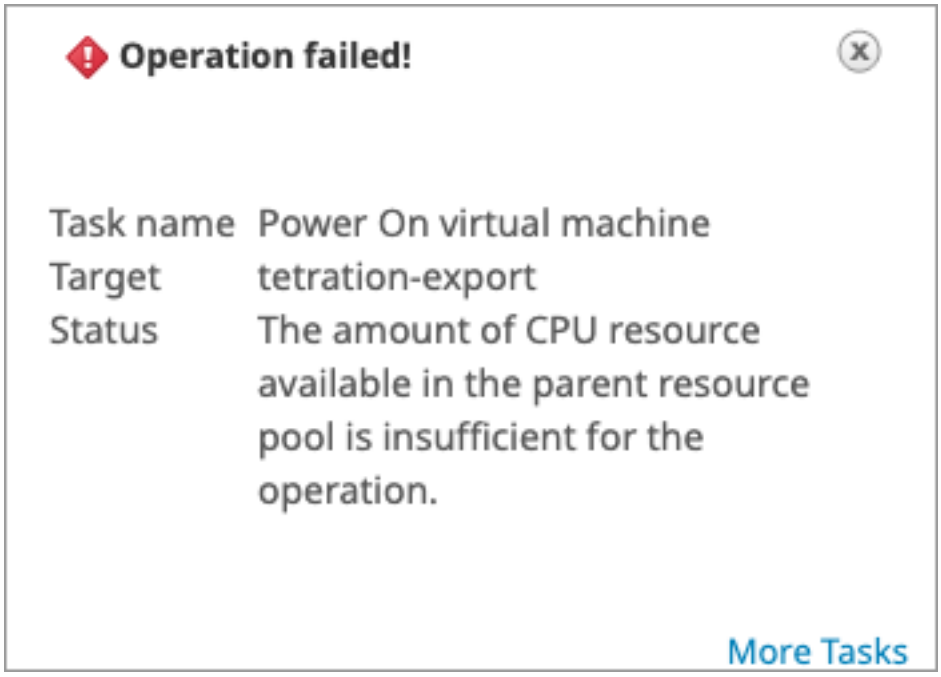


Fig. 18.2.2.1: Tetration export appliance failed to deploy for lack of resources

Note: For OVA versions 3.3.2.12 and earlier, please ensure that CPU and memory are reserved for the deployed VMs. The reservations should match the corresponding appliance specification.

Once the appliance is *Active*, connectors can be enabled and deployed on it.

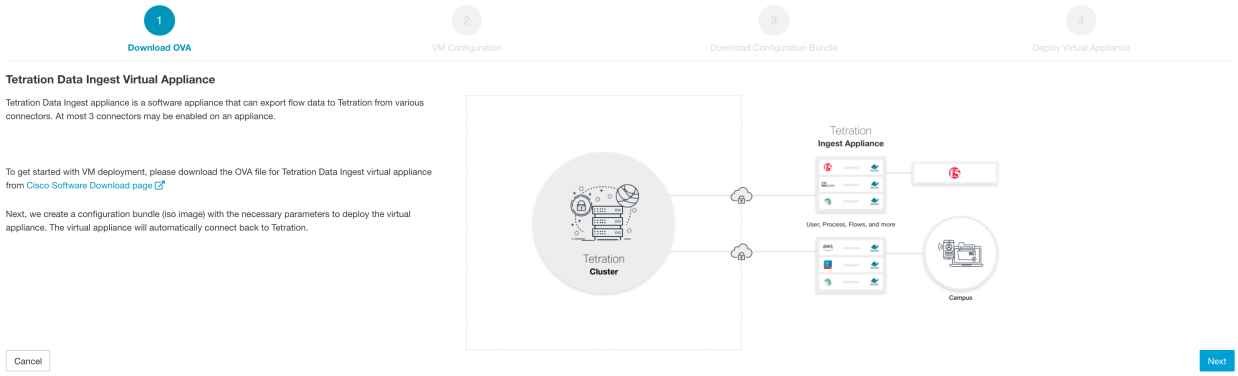


Fig. 18.2.2.2: Deploying a Tetration Ingest appliance

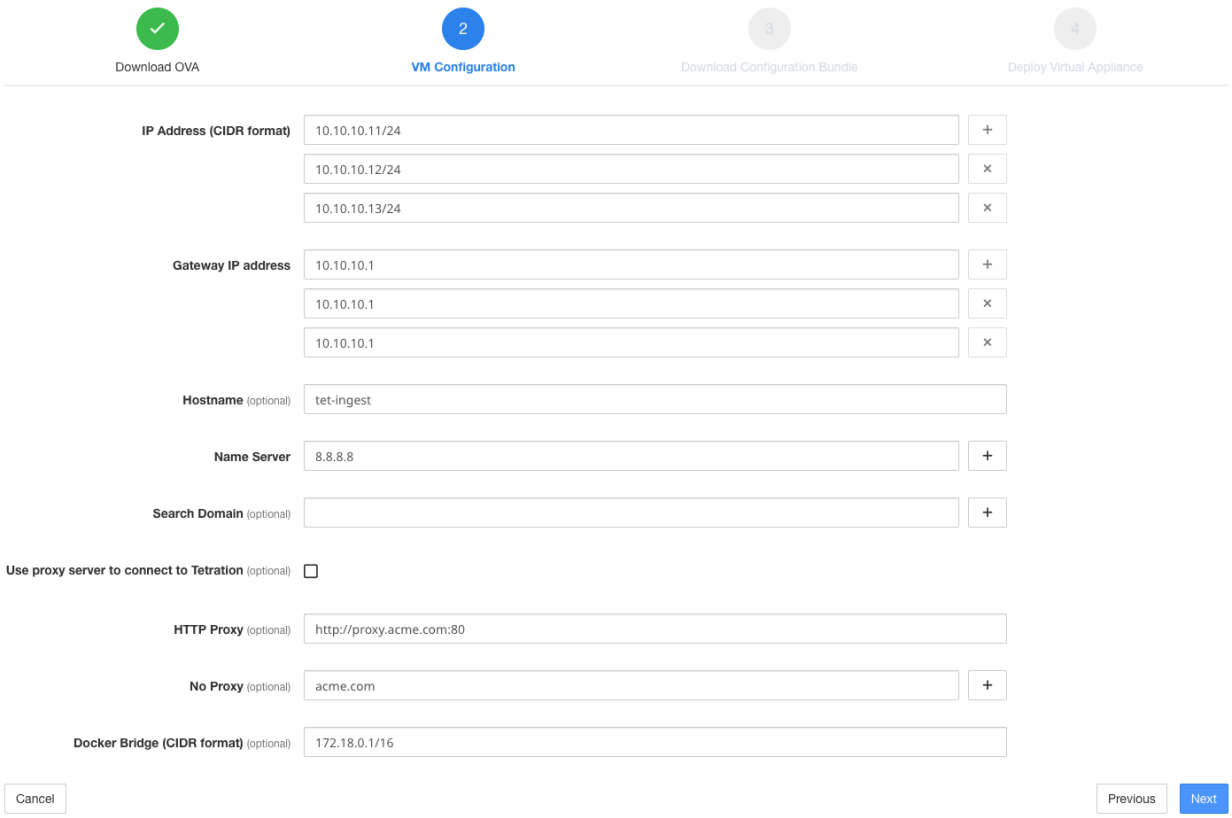


Fig. 18.2.2.3: Configuring the VM with network parameters

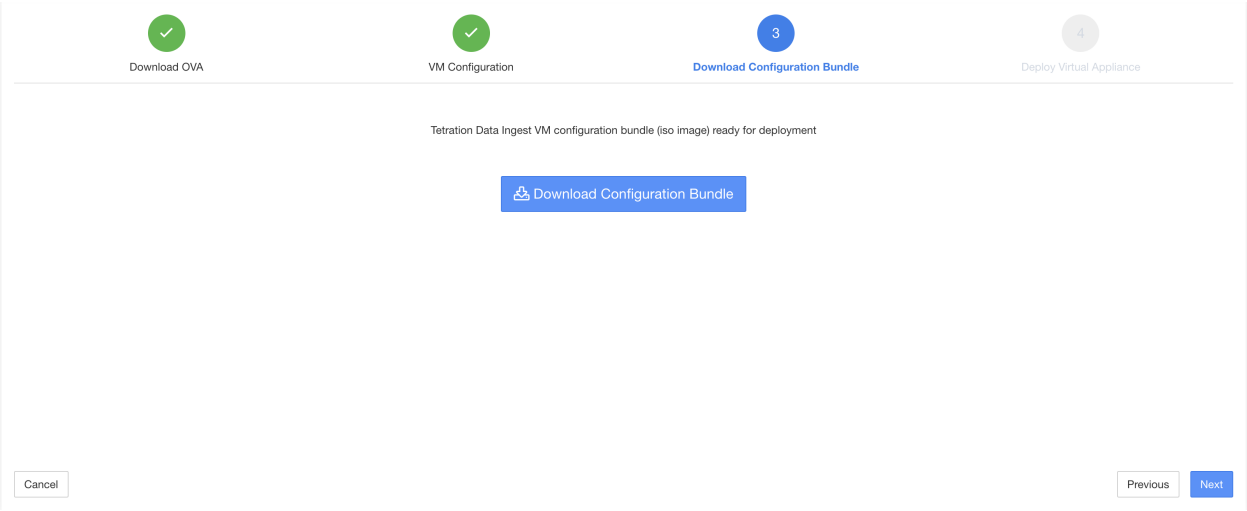


Fig. 18.2.2.4: Download the VM configuration bundle

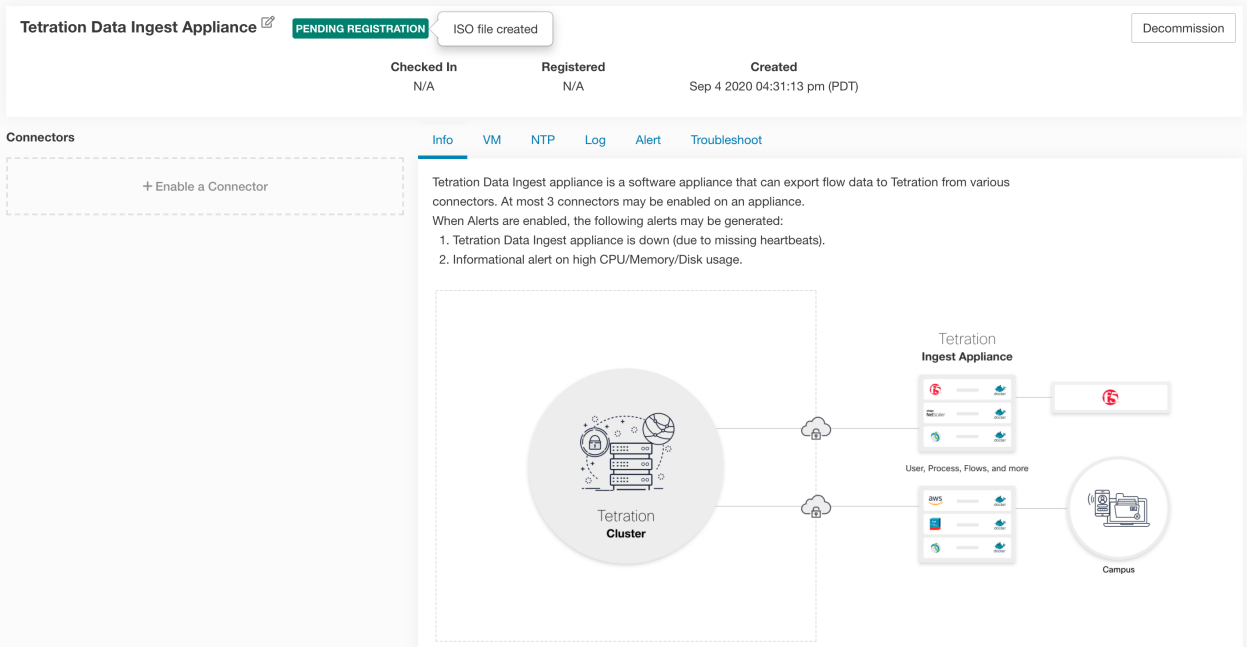


Fig. 18.2.2.5: Tetration Ingest appliance in *Pending Registration* state

When a virtual appliance is deployed and booted up for the first time, *tet-vm-setup* service executes and sets up the appliance. This service is responsible for the following tasks

1. **Validate the appliance**: validate the appliance for mandatory resource requirements for the type of the virtual appliance deployed.
2. **IP address assignment**: assign IP addresses to all the network interfaces provisioned on the appliance.
3. **Hostname assignment**: assign hostname for the appliance (if hostname is configured).
4. **DNS configuration**: update the DNS resolv.conf file (if nameserver and/or search-domain parameters are configured).
5. **Proxy server configuration**: update HTTPS_PROXY and NO_PROXY settings on the appliance (if provided).
6. **Prepare appliance**: copies cert bundle for the Kafka topic over which appliance management messages are sent and received.
7. **Install appliance controller**: install and bringup *Appliance Controller* which is managed by *supervisord* as *tet-controller* service.

Once *tet-controller* is instantiated, it takes over the management of the appliance. This service is responsible for the following functions:

1. **Registration**: registers the appliance with Tetration. Until the appliance is registered, no connectors can be enabled on the appliance. When Tetration receives a registration request for an appliance, it updates the state of the appliance to *Active*.
2. **Deploying a connector**: deploys a connector as a Docker service on the appliance. Please refer to *Enabling a Connector* for more information.
3. **Deleting a connector**: stops and removes the Docker service and the corresponding Docker image from the appliance. Please refer to *Deleting a Connector* for more information.
4. **Configuration updates on appliances**: tests and applies configuration updates on the appliance. Please refer to *Configuration Management on Connectors and Virtual Appliances* for more information.

5. **Troubleshooting commands on appliances**: executes allowed set of commands on the appliances for troubleshooting and debugging issues on the appliance. Please refer to the *Troubleshooting* for more information.
6. **Heartbeats**: periodically sends heartbeats and statistics to Tetration to report the health of the appliance. Please refer to *Monitoring a Virtual Appliance* for more information.
7. **Pruning**: periodically prune all Docker resources that are unused or dangling in order to recover storage space. This task is executed once every 24 hours.
8. **Decommissioning the appliance**: decommissions and deletes all Docker instances from the appliance. Please refer to *Decommissioning a Virtual Appliance* for more information.

The list of deployed virtual appliances can be found at: **Connectors > Virtual Appliances**.

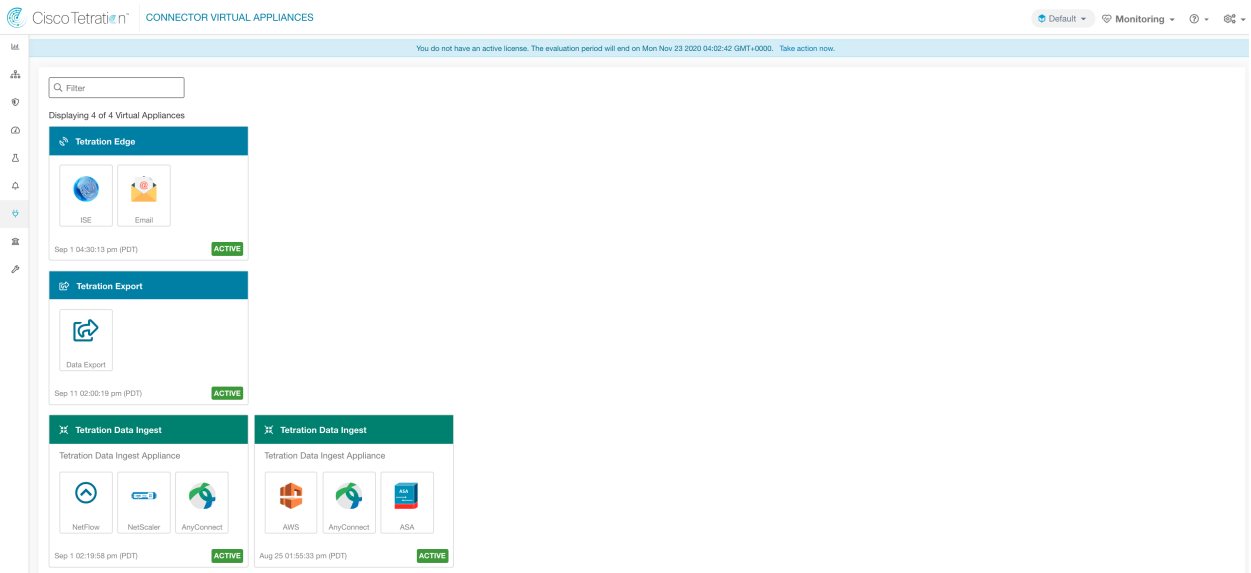


Fig. 18.2.2.6: List of deployed virtual appliances

18.2.3 Decommissioning a Virtual Appliance

A virtual appliance can be decommissioned from Tetration. When an appliance is decommissioned, the following actions are triggered.

1. All configurations on the appliance and the connectors enabled on the appliance are removed.
2. All the connectors enabled on the appliance are deleted.
3. The appliance is marked *Pending Delete*.
4. When the appliance replies back with a successful delete response, appliance Kafka topic and certs are deleted.

Note: Decommissioning an appliance cannot be undone. To restore the appliance and the connectors, a new appliance should be deployed and the connectors should be enabled on the new appliance.

18.2.4 Monitoring a Virtual Appliance

Tetration virtual appliances periodically send heartbeats and statistics to Tetration. The heartbeat interval is 5 minutes. The heartbeat messages include statistics about the health of the appliance include system statistics, process statistics,

and statistics about how many messages sent/received/error-ed over the Kafka topic that is used for the appliance management.

All metrics are available in *Digger* (OpenTSDB) and are labelled with appliance ID and root scope name. Additionally, Grafana dashboards for *Appliance Controller* are also available for important metrics from the appliance.

18.3 Life Cycle Management of Connectors

Connectors can be enabled, deployed, configured, troubleshooted, and deleted from Tetration directly.

18.3.1 Enabling a Connector

From Connectors page, a connector can be selected and enabled. The connector can be deployed on a new virtual appliance (which has to be provisioned first and become *Active* before a connector can be enabled on it) or an existing virtual appliance. Once the virtual appliance is chosen, Tetration sends the rpm package for the connector to the appliance.

When Appliance Controller on the chosen appliance receives the rpm, it does the following:

1. Construct a Docker image using the rpm package received from Tetration. This Docker image includes the configuration required to communicate with Kafka topic on which appliance management messages are sent. This enables the service instantiated from this image to be able to send and receive messages for managing the corresponding connector.
2. Create a Docker container from the Docker image.
3. On Tetration Ingest appliance, the following additional tasks are performed.
   - A free slot is identified and the corresponding IP address is determined.
   - Connector listening ports (for example, 4729 and 4739 ports on NetFlow connector to receive flow records from NetFlow V9 or IPFIX enabled switches and routers), are exposed to the host on IP corresponding to the chosen slot.
   - A Docker volume is created and added to the container.
4. The Docker container is started and it executes the connector as a *supervisord* managed service. The service starts *Service Controller* as *tet-controller* which registers with Tetration and spawns the actual connector service.

Note: On Tetration Export appliance, Tetration Export connector is enabled implicitly. And, *docker-compose* is used to bring up Logstash, Elasticsearch and Grafana stack. These Docker containers do not run *Service Controller*. Appliance Controller is responsible for managing the appliance as well as the containers.

```
[root@beretta-ingest-1 tetter]# docker images
REPOSITORY                                          TAG                     IMAGE ID        CREATED             SIZE
netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow  5d379fac6e37d85f2bdeff45  2635145b44c8    About a minute ago  650MB
tet-service-base                                    latest                  6be171bbe648    4 days ago          519MB
artifacts.tet.wtf:6555/centos                       7.3.1611                c5d48e81b986    4 months ago        192MB
[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.1: Docker Images

```
[root@beretta-ingest-1 tetter]# docker volume ls
DRIVER              VOLUME NAME
local               373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439
[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.2: Docker Volumes

```
[root@beretta-ingest-1 tetter]# docker ps
CONTAINER ID        IMAGE                                                                                    COMMAND                  CREATE
D              STATUS              PORTS                                                          NAMES
2c7a7ed4f853        netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45   "/usr/bin/supervisor…"   About
a minute ago   Up About a minute   172.29.142.26:4729->4729/udp, 172.29.142.26:4739->4739/udp   nf-5d379fac6e37d85f2bdeff45
[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.3: Docker containers

```
[root@beretta-ingest-1 tetter]# cat /local/tetration/appliance/appliance.conf
{
  "type": "TETRATION_DATA_INGEST",
  "slots": [
    {
      "available": false,
      "index": 0,
      "mapped_ip": "172.29.142.26",
      "share_volume": true,
      "count": 1,
      "service_containers": {
        "5d379fac6e37d85f2bdeff45": {
          "connector_id": "5d379fac6e37d85f2bdeff44",
          "service_id": "5d379fac6e37d85f2bdeff45",
          "container_id": "2c7a7ed4f853e85f3d620c663f1c7f5395b53b9dd6696276ac439d34fe142bf1",
          "image_name": "netflow_sensor-3.4.2.52222.maarumug.mrpm.build-netflow:5d379fac6e37d85f2bdeff45",
          "container_name": "nf-5d379fac6e37d85f2bdeff45",
          "service_type": "NETFLOW_SENSOR",
          "ip_bindings": [
            {
              "ip": "172.29.142.26",
              "port": "4729",
              "protocol": "udp"
            },
            {
              "ip": "172.29.142.26",
              "port": "4739",
              "label": 1,
              "protocol": "udp"
            }
          ],
          "volume_id": "373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439"
        }
      }
    },
    {
      "available": true,
      "index": 1,
      "mapped_ip": "172.29.142.27",
      "share_volume": true,
      "count": 0,
      "service_containers": null
    },
    {
      "available": true,
      "index": 2,
      "mapped_ip": "172.29.142.28",
      "share_volume": true,
      "count": 0,
      "service_containers": null
    }
  ]
}[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.4: Slot used by the Docker container and list of exposed ports

```
[root@beretta-ingest-1 tetter]# docker port 2c7a7ed4f853
4729/udp -> 172.29.142.26:4729
4739/udp -> 172.29.142.26:4739
[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.5: List of ports exposed by Docker container

```
[root@beretta-ingest-1 tetter]# docker inspect --format='{{json .Mounts}}' 2c7a7ed4f853
[{"Type":"volume","Name":"373b5b682a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439","Source":"/var/lib/docker/volumes/373b5b6
82a96547bf2526784a5943c2f110593b88485b996e7259fa4e314c439/_data","Destination":"/local/tetration","Driver":"local","Mode":"z","RW":tr
ue,"Propagation":""}]
[root@beretta-ingest-1 tetter]#
```

Fig. 18.3.1.6: Docker Volume mounted to a container

Service Controller is responsible for the following functions:

1. **Registration**: registers the connector with Tetration. Until the connector is registered and marked *Enabled*, no configuration updates can be pushed to the connector. When Tetration receives a registration request for a connector, it updates the state of the connector to *Enabled*.
2. **Configuration updates on connector**: tests and applies configuration updates on the connector. Please refer to *Configuration Management on Connectors and Virtual Appliances* for more information.
3. **Troubleshooting commands on connector**: executes allowed commands on the connector service for troubleshooting and debugging issues on the connector service. Please refer to *Troubleshooting* for more information.
4. **Heartbeats**: periodically sends heartbeats and statistics to Tetration to report the health of the connector. Please refer to *Monitoring a Virtual Appliance* for more information.

List of all enabled connectors can be found at: **Connectors > Connectors** and select *Manage* button on the right side of the page.

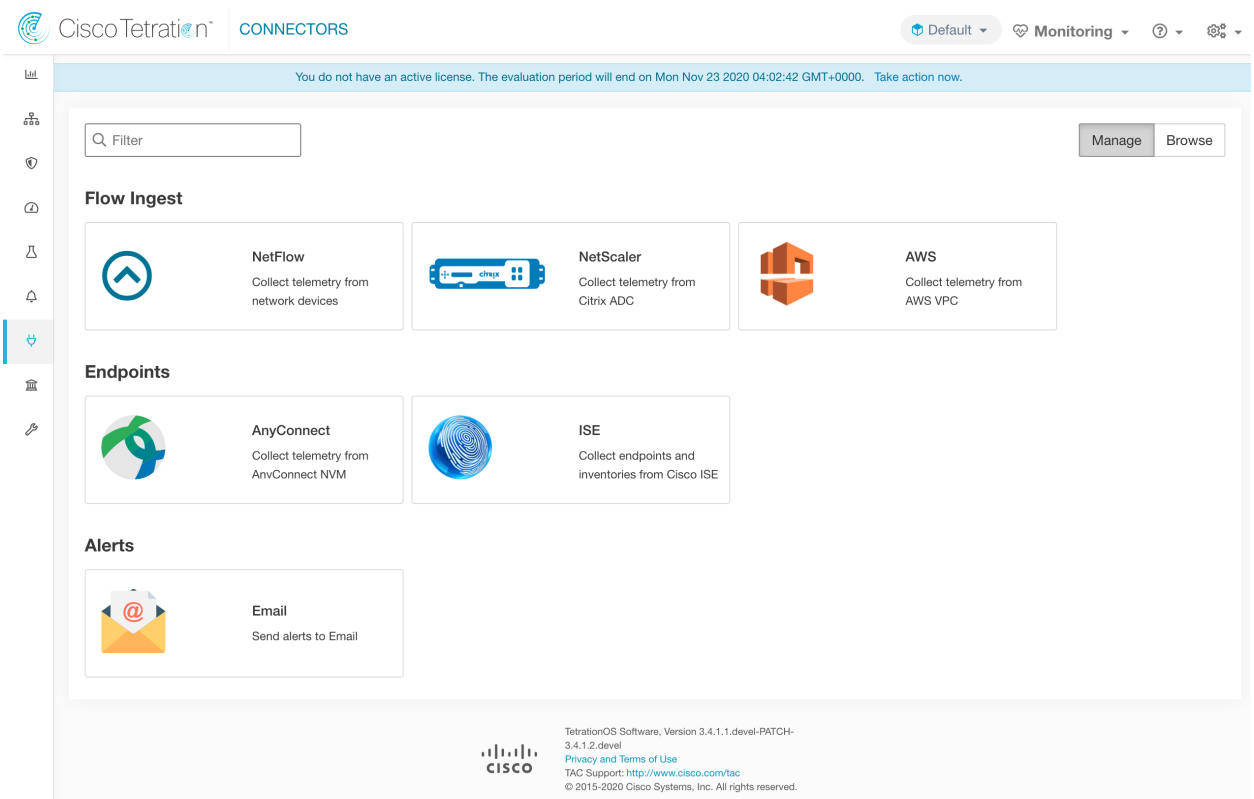


Fig. 18.3.1.7: List of enabled connectors

And, the list of deployed virtual appliances can be found at: **Connectors > Virtual Appliances**.

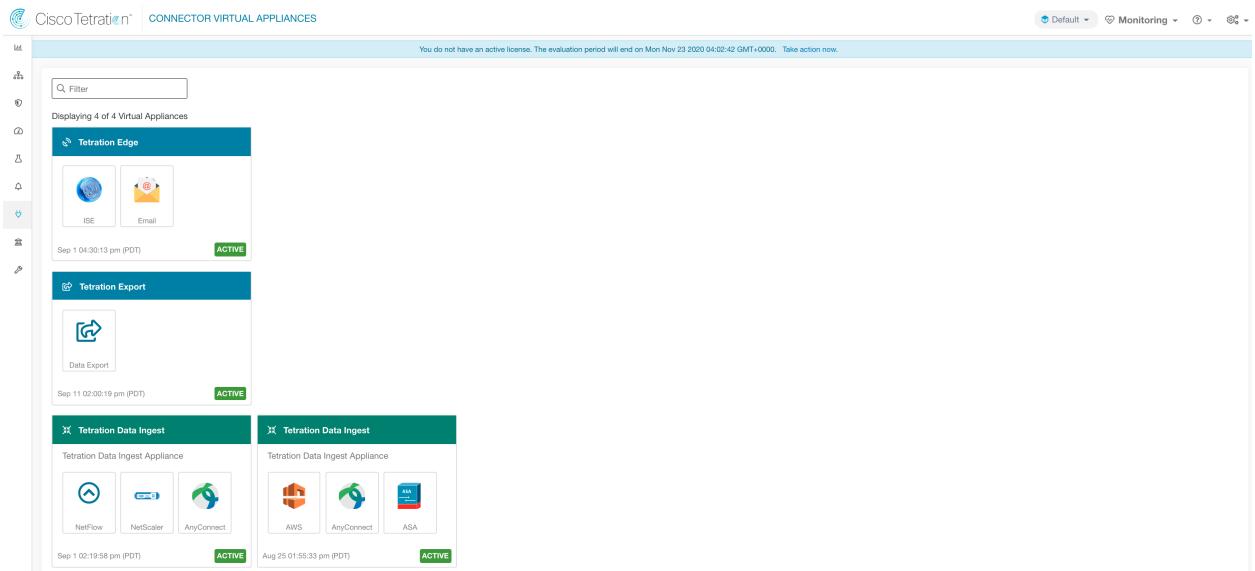


Fig. 18.3.1.8: List of deployed virtual appliances

A detailed view of an appliance can be fetched by clicking on the appliance directly from *List of deployed virtual appliances*.

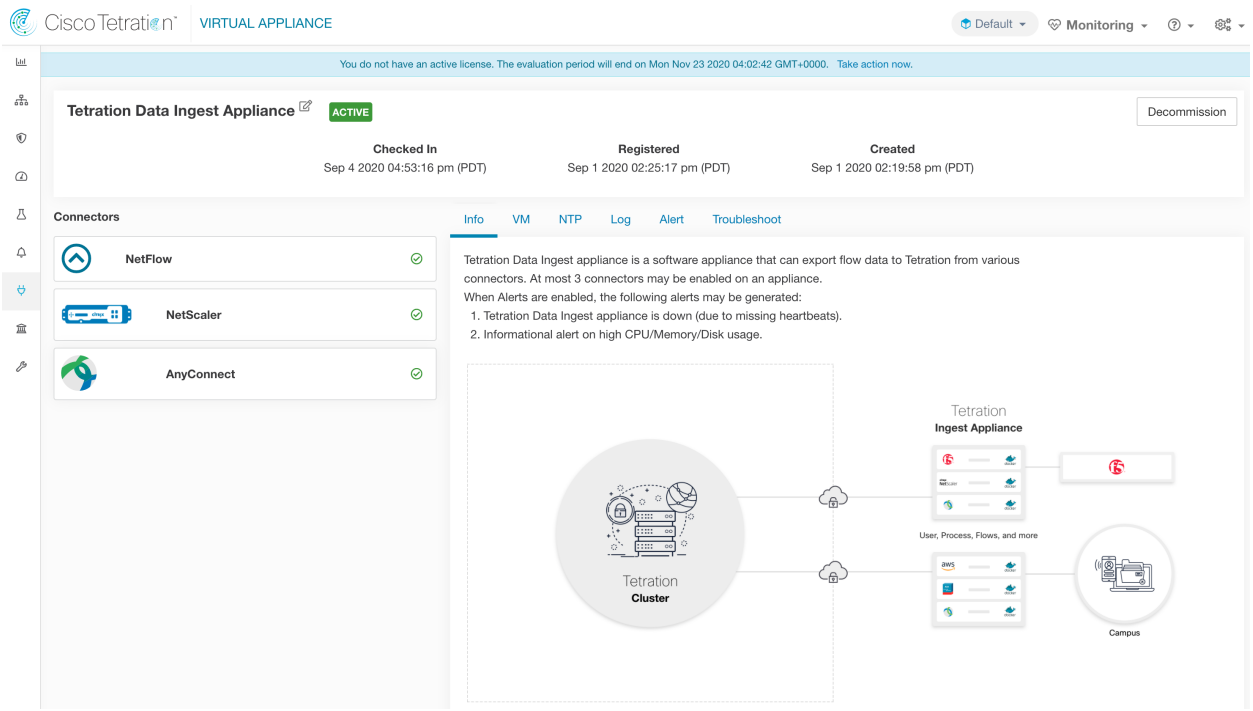


Fig. 18.3.1.9: Appliance details and the connectors

And, finally, details about the connector can be fetched by clicking on the connector. This page shows the port bindings -if any- that can be used to configure upstream network elements to send telemetry data to the correct IP and port.

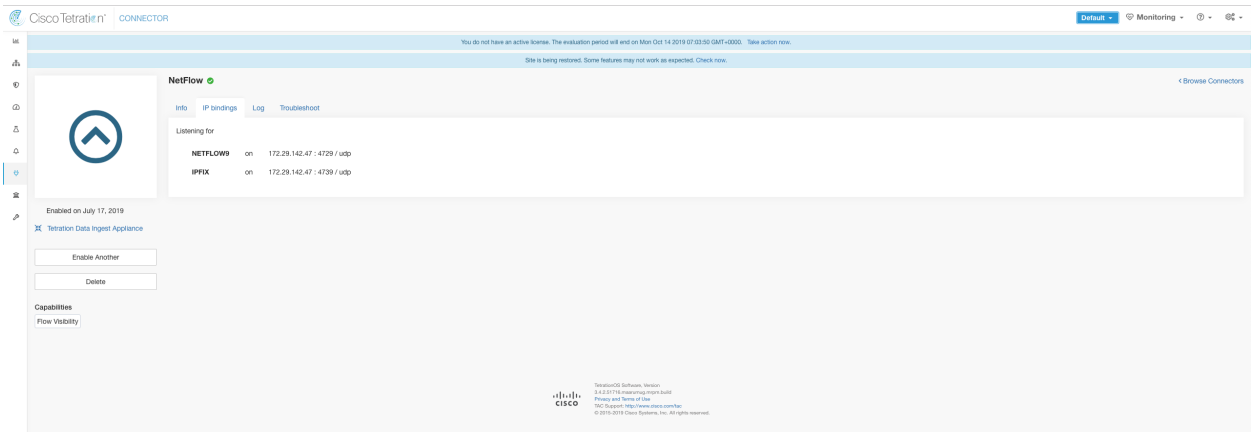


Fig. 18.3.1.10: Connector details

18.3.2 Deleting a Connector

When a connector is deleted, Appliance Controller on the appliance where the connector is enabled will receive a message to remove the services created for the connector. Appliance Controller does the following:

1. Stop the Docker container corresponding to the connector.
2. Remove the Docker container.

3. If the connector is deployed on a Tetration Ingest appliance and it exposes ports, then remove the Docker volume that was mounted to the container.
4. Remove the Docker image that was created for the connector.
5. Finally, send a message back to Tetration indicating the status of the delete request.

18.3.3 Monitoring a Connector

Connector services periodically send heartbeats and statistics to Tetration. The heartbeat interval is 5 minutes. The heartbeat messages include statistics about the health of the service include system statistics, process statistics, and statistics about how many messages sent/received/error-ed over the Kafka topic that is used for the appliance management. In addition, it includes statistics exported by the connector service itself.

All metrics are available in *Digger* (OpenTSDB) and are annotated with appliance ID, connector ID, and root scope name. Additionally, Grafana dashboards for connector services are also available for important metrics from the service.

18.4 Configuration Management on Connectors and Virtual Appliances

Configuration updates can be pushed to appliances and connectors from Tetration. The appliance should have registered successfully with Tetration and be *Active* before configuration updates can be initiated. Similarly, the connectors should have registered with Tetration before configuration updates can be initiated on the connector services.

There are 3 modes of configuration updates possible in appliances and connectors.

1. **Test and Apply**: Test the configuration and on successful test, commit the configuration.
2. **Discovery**: Test the configuration, and on successful test, discovery additional properties that can be enabled for the configuration.
3. **Remove**: Remove the configuration.

18.4.1 Test and Apply

Configurations that support *Test and Apply* mode verify the configuration before applying (committing) the configuration on the desired appliance and/or connector.

18.4.1.1 NTP Configuration

NTP configuration allows the appliance to synchronize the clock with the specified NTP server(s).

Parameter Name	Type	Description
Enable NTP	checkbox	Should NTP sync be enabled?
NTP Servers	list of strings	List of NTP servers. At least one server should be given and at most 5 servers may be provided.

Test: Test if a UDP connection can be made to the given NTP servers on port 123. If an error occurs for any of the NTP servers, do not accept the configuration.

Apply: Update `/etc/ntp.conf` and restart ntpd service using `systemctl restart ntpd.service`. Here is the template for generating the *ntp.conf*.

```
# --- GENERAL CONFIGURATION ---
server <ntp-server>
...
server 127.127.1.0
fudge 127.127.1.0 stratum 10

# Drift file
driftfile /etc/ntp/drift
```

Allowed Tetration virtual appliances: All

Allowed connectors: None

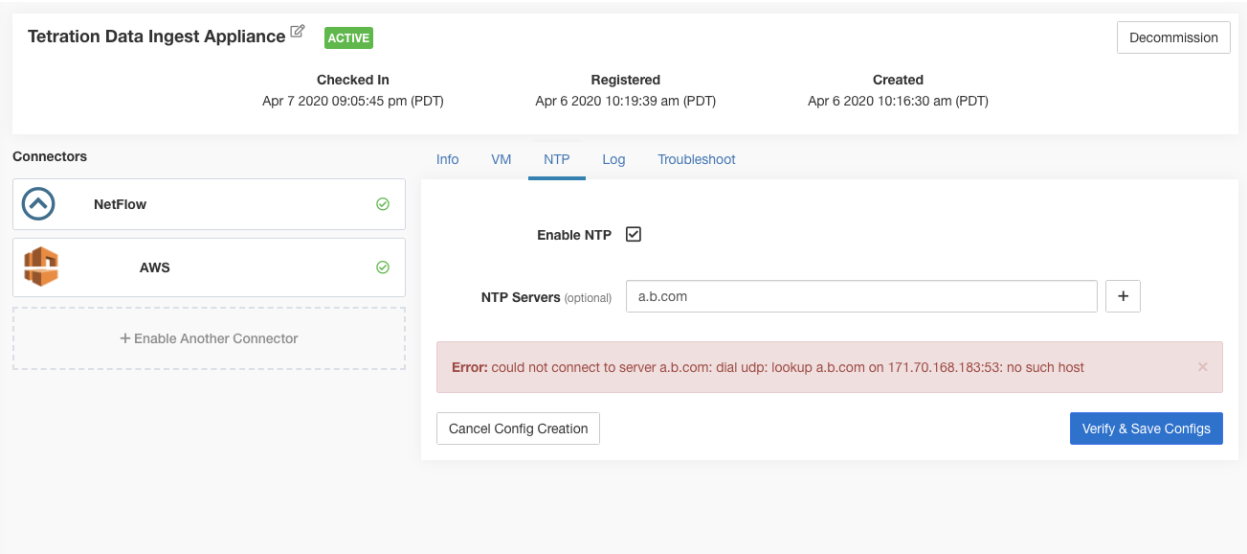


Fig. 18.4.1.1.1: Error while testing NTP configuration

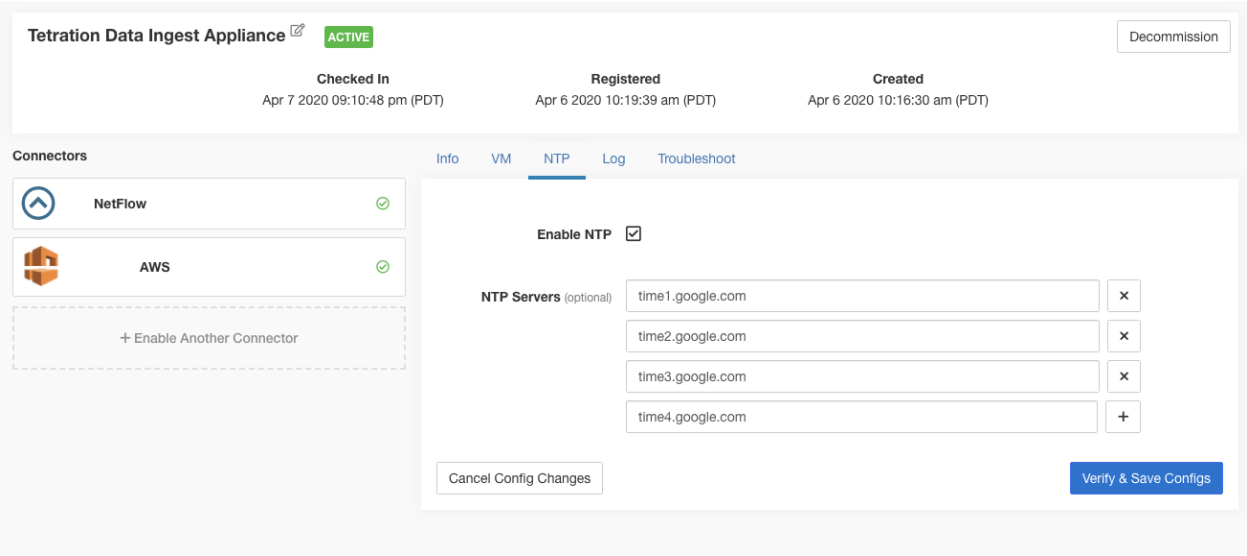


Fig. 18.4.1.1.2: NTP configuration with valid NTP servers

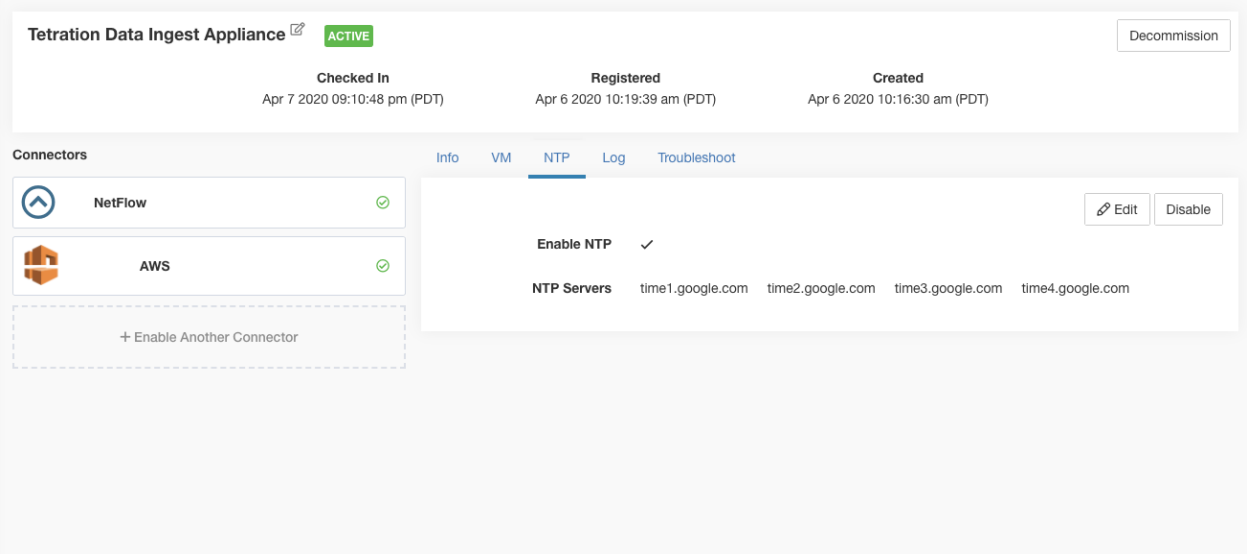


Fig. 18.4.1.1.3: NTP configuration verified and applied

18.4.1.2 Log Configuration

Log configuration updates the log levels, maximum size of the log files, and log rotation parameters on the appliance and/or connector. If the configuration update is triggered on the appliance, appliance controller log settings are updated. On the other hand, if the configuration update is triggered on a connector, service controller and service log settings are updated.

Parameter Name	Type	Description
Logging level	dropdown	Logging level to be set
	• *debug*	Debug log level
	• *info*	Informational log level
	• *warn*	Warning log level
	• *error*	Error log level
Max log file size (in MB)	number	Maximum size of a log file before log rotation kicks in
Log rotation (in days)	number	Maximum age of a log file before log rotation kicks in
Log rotation (in instances)	number	Maximum instances of log files kept

Test: No op.

Apply: If the configuration is trigged on an appliance, update the configuration file of *tet-controller* on the appliance. If the configuration is triggered on a connector, update the configuration files of *tet-controller* and the service managed by the controller on the Docker container responsible for the connector.

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, ISE, ASA, and Meraki.

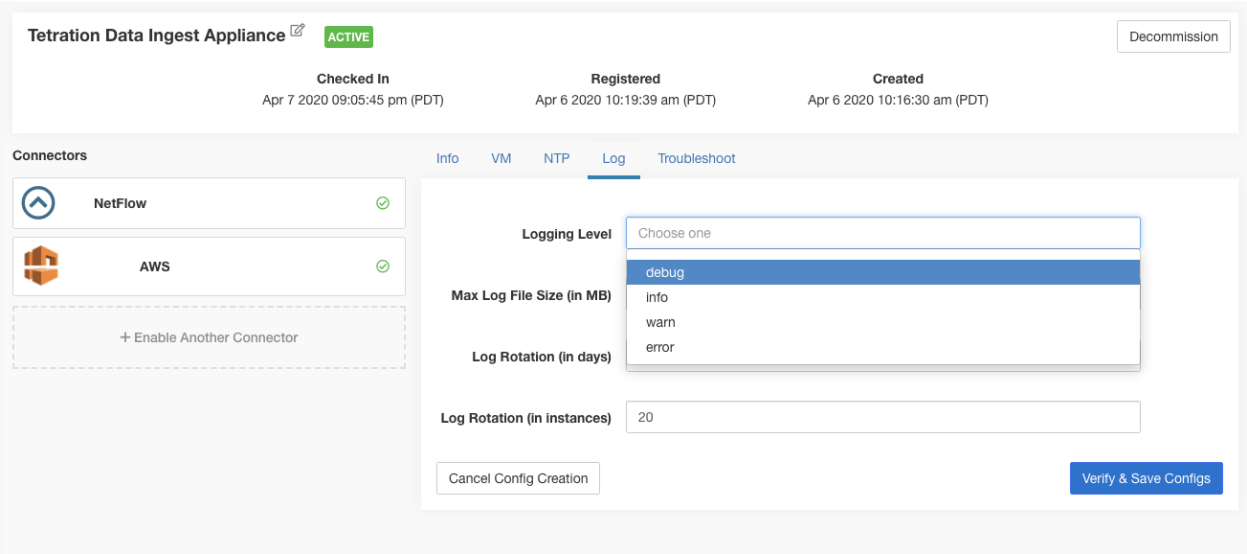


Fig. 18.4.1.2.1: Log configuration on the appliance

Note: Since all alert notifier Connectors (Syslog, Email, Slack, PagerDuty, and Kinesis) run on a single Docker service (Tetration Alert Notifier) on Tetration Edge, it is not possible to update the log config of a connector without impacting the config of another alert notifier connector. The log configurations of Tetration Alert Notifier (TAN) Docker service on Tetration Edge appliance can be updated using an allowed command. See *Update Alert Notifier*

Connector Log Configuration for more details.

18.4.1.3 AWS Configuration

AWS configuration specifies the AWS credentials and the S3 buckets from where AWS VPC flow logs should be downloaded and processed by the AWS connector.

Parameter Name	Type	Description
AWS Access Key ID	string	AWS access key ID to communicate with AWS.
AWS Secret Access Key	string	AWS secret access key to communicate with AWS.
AWS Region	dropdown of AWS regions	Name of the AWS region that hosts the S3 buckets from where VPC flow logs should be downloaded.
List of AWS S3 Buckets	list of strings	List of S3 buckets that has the VPC flow logs exported via Cloud Watch or VPC flow log.

Test: Create a new session to AWS and get a listing of all AWS S3 buckets for the configured region. If the list of all AWS S3 buckets includes the list of buckets to fetch VPC flow logs then the test is successful.

Apply: Update configuration files for the downloader service in AWS connector to pull the VPC flow log files from S3 buckets.

Allowed Tetration virtual appliances: None

Allowed connectors: AWS

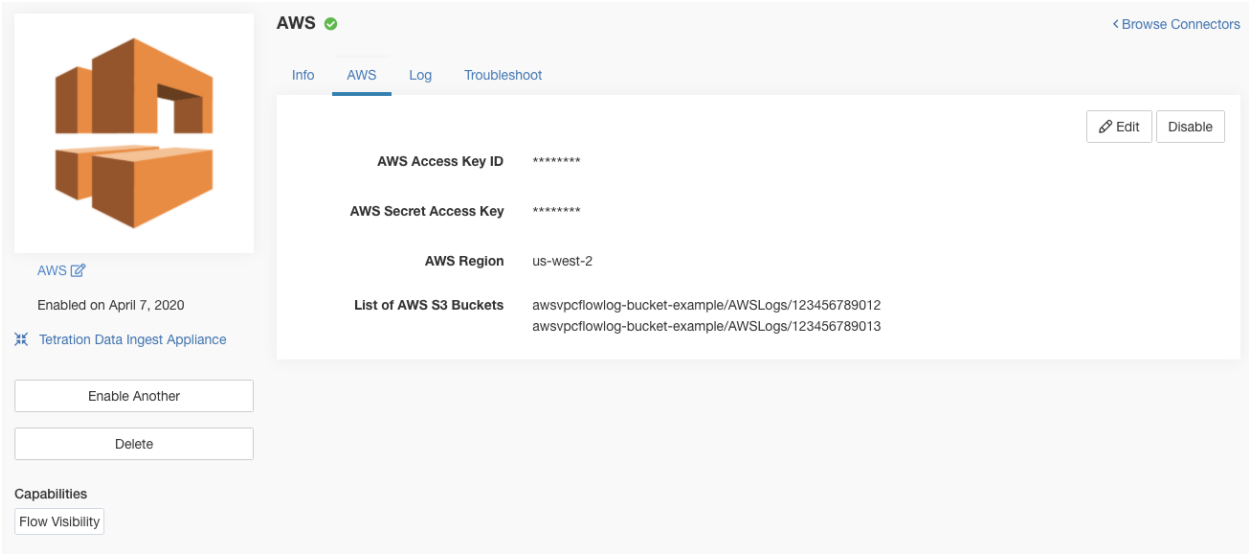


Fig. 18.4.1.3.1: AWS config on AWS connector

Note: The value of *List of AWS S3 Buckets* should be a valid S3 path. For example *awsvpcflowlog/AWSLogs/123456789012/* refers to bucket name *awsvpcflowlog* and all sub-folder in *AWSLogs/123456789012/*.

18.4.1.4 Endpoint Configuration

Endpoint configuration specifies the inactivity timeout for endpoints on AnyConnect and ISE connectors. When an endpoint times out, the connector stops checking in with Tetration and purges the local state for the endpoint on the connector.

Parameter Name	Type	Description
Inactivity Timeout for Endpoints (in minutes)	number	Inactivity timeout for endpoints published by AnyConnect / ISE connectors. On timeout, the endpoint will not longer checkin Tetration. Default is 30 minutes.

Test: No op.

Apply: Update the configuration file of the connector with the new value

Allowed Tetration virtual appliances: None

Allowed connectors: AnyConnect and ISE

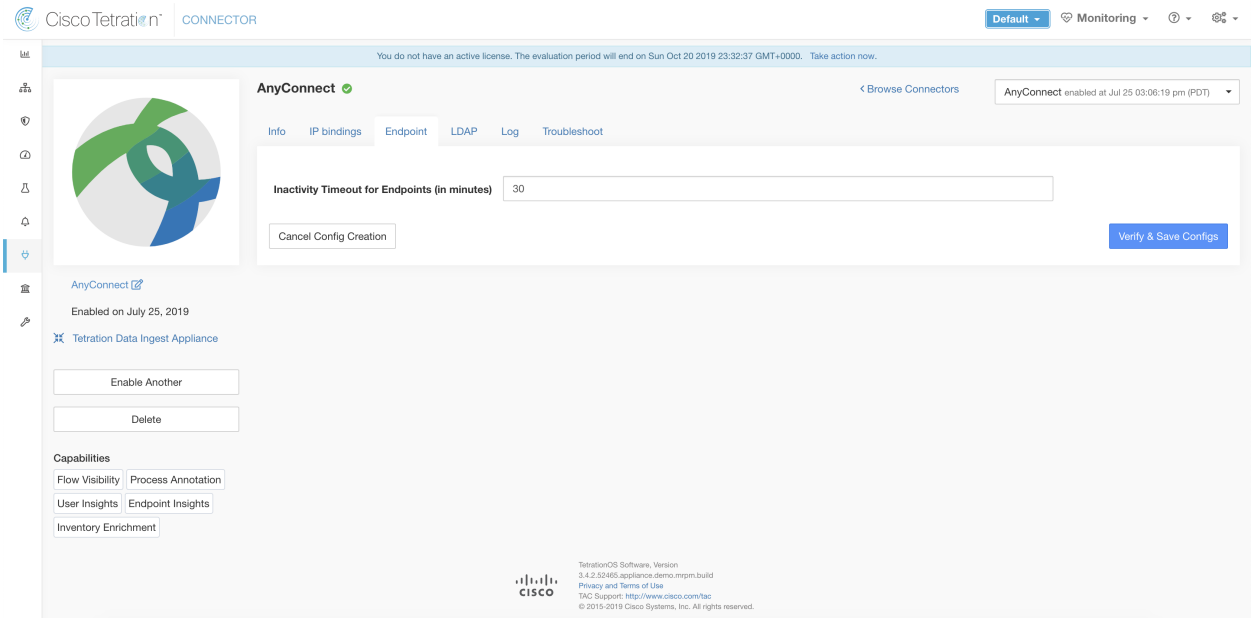


Fig. 18.4.1.4.1: Endpoint inactivity timeout configuration on AnyConnect connector

18.4.1.5 Slack Notifier Configuration

Default configuration for publishing Tetration alerts on Slack.

Parameter Name	Type	Description
Slack Webhook URL	string	Slack webhook on which Tetration alerts should be published

Test: Send a test alert to Slack using the webhook. If the alert is posted successfully, the test passes.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: Slack

18.4.1.6 PagerDuty Notifier Configuration

Default configuration for publishing Tetration alerts on PagerDuty.

Parameter Name	Type	Description
PagerDuty Service Key	string	PagerDuty service key for pushing Tetration alerts on PagerDuty

Test: Send a test alert to PagerDuty using the service key. If the alert is published successfully, the test passes.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: PagerDuty

18.4.1.7 Kinesis Notifier Configuration

Default configuration for publishing Tetration alerts on Amazon Kinesis.

Parameter Name	Type	Description
AWS Access Key ID	string	AWS access key ID to communicate with AWS
AWS Secret Access Key	string	AWS secret access key to communicate with AWS
AWS Region	dropdown of AWS regions	Name of the AWS region where Kinesis stream is configured
Kinesis Stream	string	Name of the Kinesis stream
Stream Partition	string	Partition Name of the stream

Test: Send a test alert to Kinesis stream using the given configuration. If the alert is published successfully, the test passes.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: Kinesis

18.4.1.8 Email Notifier Configuration

Default configuration for publishing Tetration alerts on Email.

Parameter Name	Type	Description
SMTP Username	string	SMTP server username. This parameter is optional.
SMTP Password	string	SMTP server password for the user (if given). This parameter is optional.
SMTP Server	string	IP address or hostname of the SMTP server
SMTP Port	number	Listening port of SMTP server. Default value is 587.
Secure Connection	checkbox	Should SSL be used for SMTP server connection?
From Email Address	string	Email address to use for sending alerts
Default Recipients	string	Comma separated list of recipient email addresses

Test: Send a test email using the given configuration. If the alert is published successfully, the test passes.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: Email

18.4.1.9 Syslog Notifier Configuration

Default configuration for publishing Tetration alerts on Syslog.

Parameter Name	Type	Description
Protocol	dropdown • *UDP* • *TCP*	Protocol to use to connect to server
Server Address	string	IP address or hostname of the Syslog server
Port	number	Listening port of Syslog server. Default port value is 514.

Test: Send a test alert to Syslog server using the given configuration. If the alert is published successfully, the test passes.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: Syslog

18.4.1.10 Syslog Severity Mapping Configuration

The following table shows the default severity mapping for Tetration alerts on Syslog

Tetration Alerts Severity	Syslog Severity
LOW	LOG_DEBUG
MEDIUM	LOG_WARNING
HIGH	LOG_ERR
CRITICAL	LOG_CRIT
IMMEDIATE ACTION	LOG_EMERG

This setting can be modified using this configuration.

Parameter Name	Dropdown of mappings
IMMEDIATE_ACTION	• *Emergency* • *Alert* • *Critical* • *Error* • *Warning* • *Notice* • *Informational* • *Debug*
CRITICAL	
HIGH	
MEDIUM	
LOW	

Test: No op.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: Syslog

18.4.1.11 ISE Instance Configuration

This configuration provides the parameters required to connect to Cisco Identity Services Engine (ISE). By providing multiple instances of this configuration, the ISE connector can connect and pull metadata about endpoints from multiple ISE appliances. Up to 20 instances of ISE configuration may be provided.

Parameter Name	Type	Description
ISE Client Certificate	string	ISE client certificate to connect to ISE using pxGrid
ISE Client Key	string	ISE client key to connect to ISE
ISE Server CA Certificate	string	CA certificate of ISE
ISE Hostname	string	FQDN of ISE pxGrid
ISE Nodename	string	Node name of ISE pxGrid

Test: Connect to ISE using the given parameters. On successful connection, accept the configuration.

Apply: Update configuration file of the connector with the specified parameters.

Allowed Tetration virtual appliances: None

Allowed connectors: ISE

18.4.2 Discovery

Configurations that support *Discovery* mode do the following.

1. Collect a basic configuration from the user.
2. Verify the basic configuration.
3. Discovery additional properties about the configuration and present them to the user.
4. Let the user enhance the configuration using the discovered properties.
5. Verify and apply the enhanced configuration.

In 3.3.1.x release, LDAP configuration supports discovery mode.

18.4.2.1 LDAP Configuration

LDAP configuration specifies how to connect to LDAP, what is the base Distinguished Name (DN) to use, what is the attribute that corresponds to username, and what attributes to fetch for each username. LDAP attributes are properties of LDAP that are specific to that environment.

Given the configuration of how to connect to LDAP and the base DN, it is possible to discover the attributes of users in LDAP. These discovered attributes can then be presented to the user in the UI. From these discovered attributes, the user selects the attribute that corresponds to the username and a list of up to 6 attributes to collect for each username from LDAP. As a result, this eliminates the manual configuration of the LDAP attributes and reduces errors.

Here are the detailed steps for creating LDAP configuration through discovery.

Step 1: Start the LDAP Configuration

Initiate an LDAP configuration for the connector.

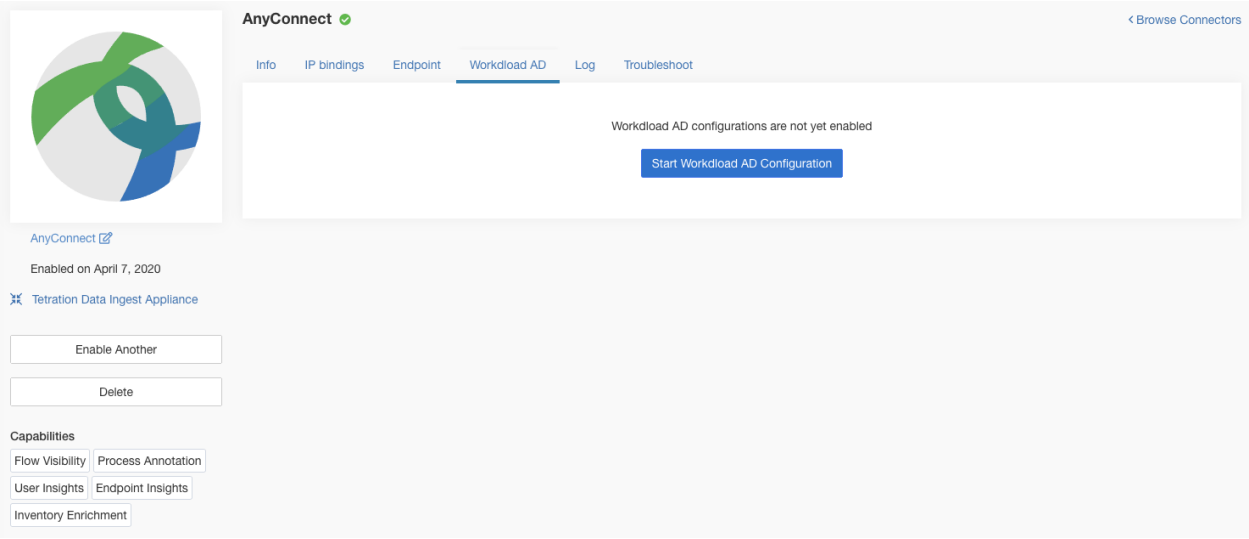


Fig. 18.4.2.1.1: Start the LDAP configuration discovery

Step 2: Provide Basic LDAP Configuration

Specify the basic configuration for connecting to LDAP. In this configuration, the users provide the LDAP Bind DN or username to connect to LDAP server, LDAP password to use to connect to LDAP server, LDAP server address, LDAP server port, Base DN to connect to, and a filter string to fetch users that match this filer.

Parameter Name	Type	Description
LDAP Username	string	LDAP username or bind DN to access LDAP server
LDAP Password	string	LDAP password for the username to access LDAP server
LDAP Server	string	LDAP server address
LDAP Port	number	LDAP server port
Use SSL	checkbox	Should the connector connect to LDAP securely? Optional. Default is false.
Verify SSL	checkbox	Should the connector verify LDAP cert? Optional. Default is false.
LDAP Server CA Cert	string	Server CA certificate. Optional.
LDAP Server Name	string	Servername for which the LDAP cert is issued (mandatory if *Verify SSL* is checked.
LDAP Base DN	string	LDAP base DN, the starting point for directory searches in LDAP
LDAP Filter String	string	LDAP filter prefix string. Filter the search result that match only this condition.

Continued on next page

Table 18.4.2.1.1 – continued from previous page

Parameter Name	Type	Description
Snapshot Sync Interval (in hours)	number	Specify the time interval in hours to (re)create LDAP snapshot. Optional. Default is 24 hours.
Use Proxy to reach LDAP	checkbox	Should the connector use proxy server to access LDAP server?
Proxy Server to reach LDAP	string	Proxy server to access LDAP

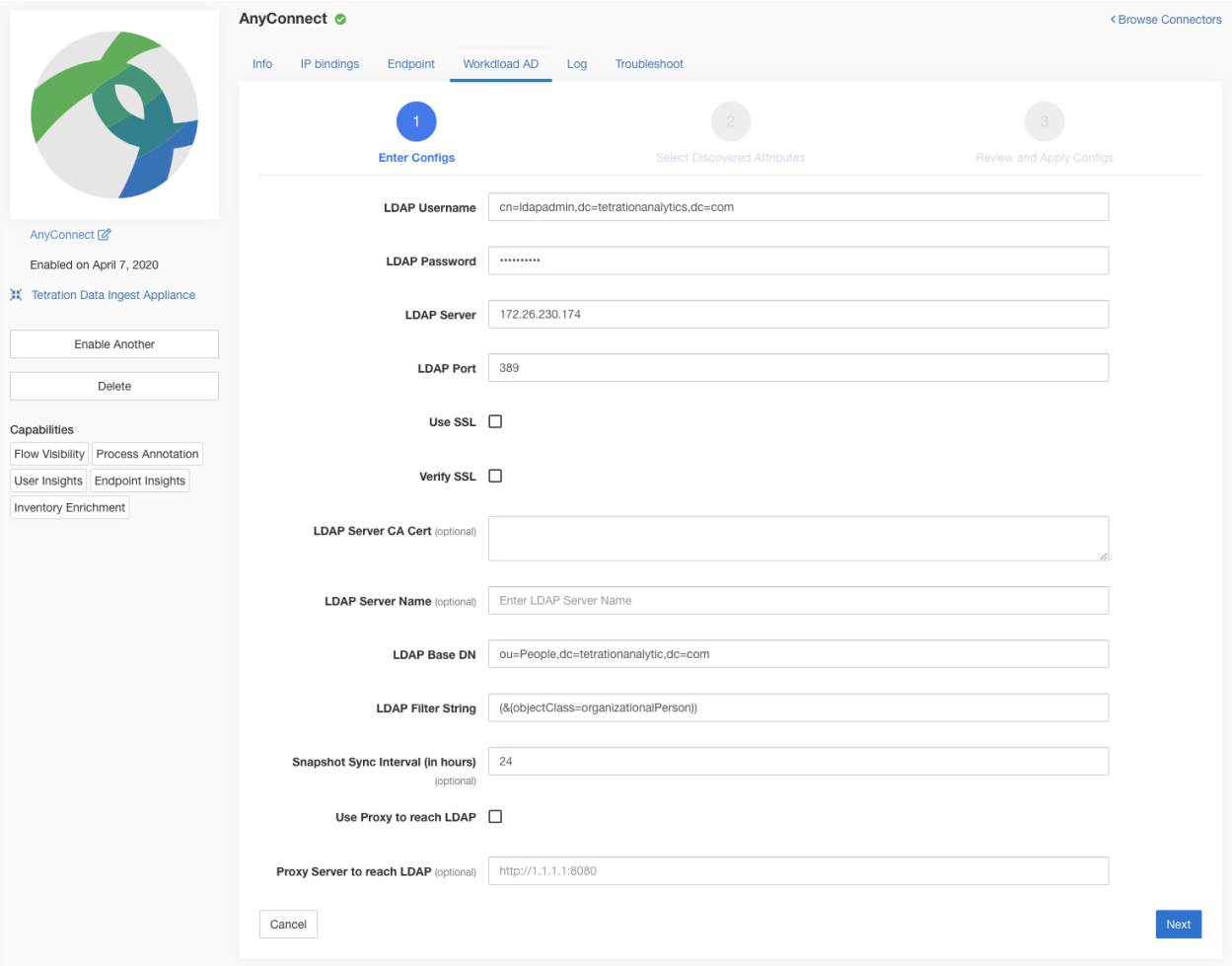


Fig. 18.4.2.1.2: Initial LDAP configuration

Step 3: Discovery in Progress

Once the user clicks *Next*, this configuration is sent to the connector. The connector establishes a connection with LDAP server using the given configuration. It fetches up to 1000 users from LDAP server and identifies all the attributes. Furthermore, it computes a list of all the single-valued attributes are common across all 1000 users. The connector returns this result back to Tetration.

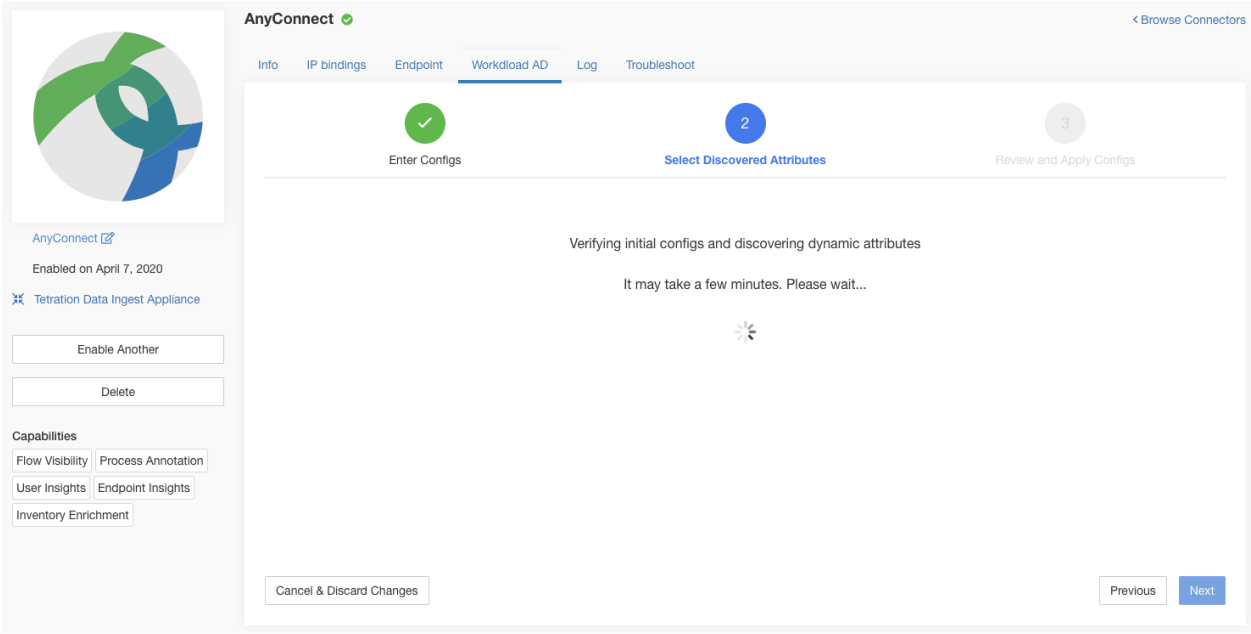


Fig. 18.4.2.1.3: Discovery in progress

Step 4: Enhance the Configuration with Discovered Attributes

The user has to pick which attribute corresponds to username and select up to 6 attributes that the connector has to fetch and snapshot for each user in the organization (i.e., users matching the filter string). This action is performed using a dropdown of list of discovered attributes. Thus, eliminating manual errors and misconfiguration.

Parameter Name	Type	Description
LDAP Username Attribute	string	LDAP attribute that contains the username
LDAP Attributes to Fetch	list of strings	List of LDAP attributes that should be fetched for a user

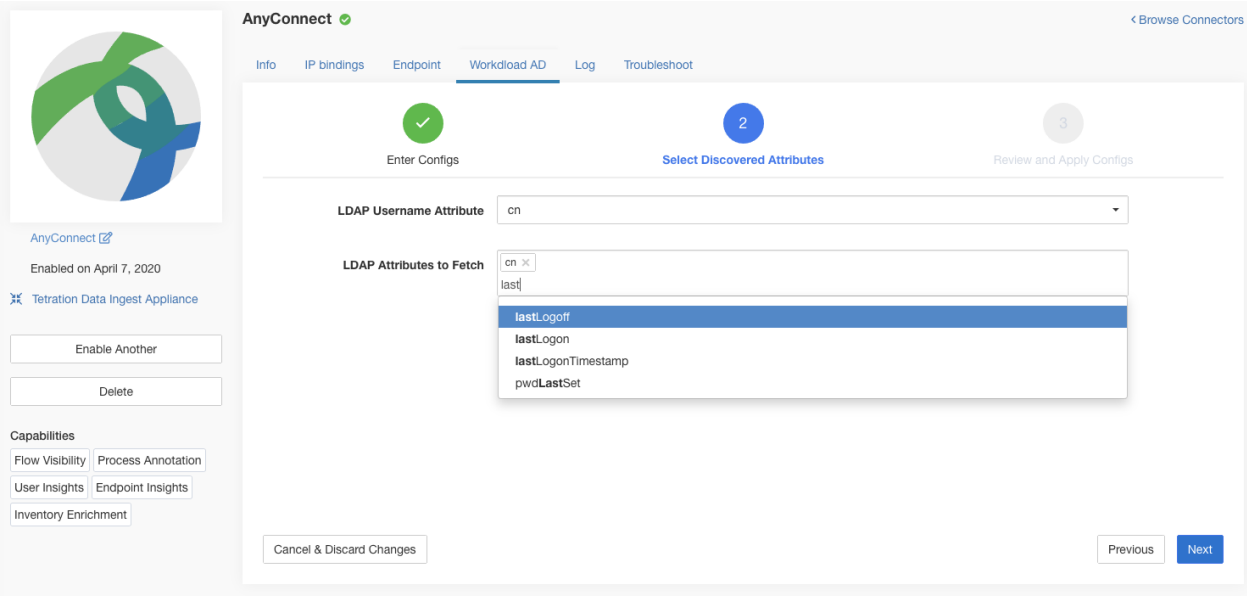


Fig. 18.4.2.1.4: Discovered LDAP attributes

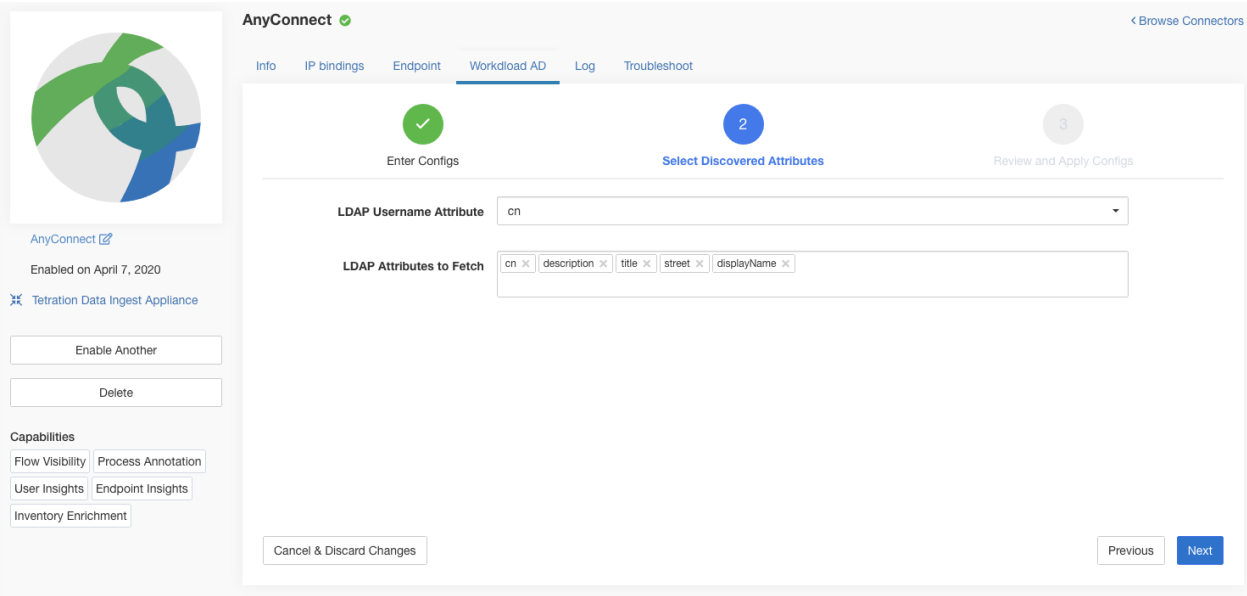


Fig. 18.4.2.1.5: Identify username attribute and attributes to collect for each username

Step 5: Finalize, Save, and Apply the Configuration

Finally, the configuration is completed by clicking *Save and Apply Changes*.

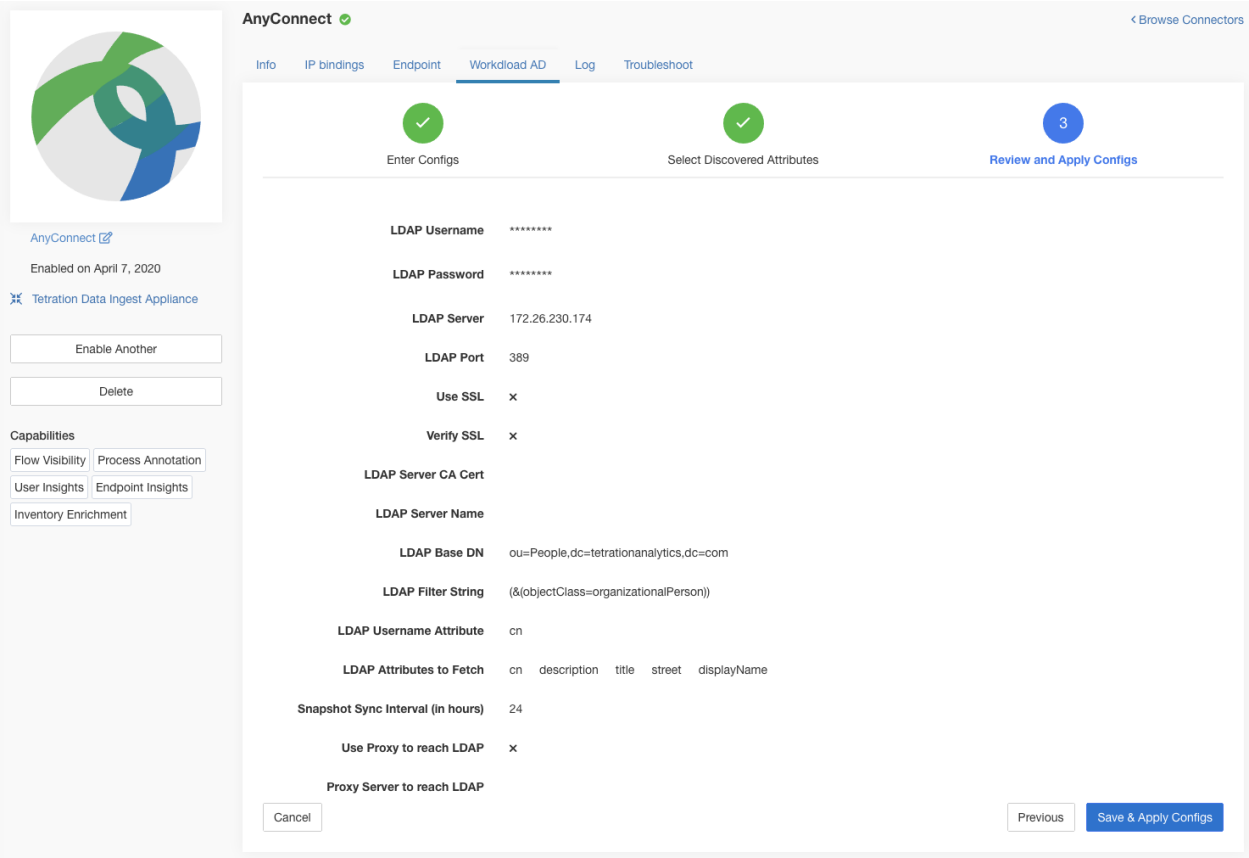


Fig. 18.4.2.1.6: Complete LDAP configuration discovery and commit

The connector receives the completed configuration. It creates a local snapshot of all users matching the filter string and fetches only the selected attributes. Once the snapshot is completed, the connector services can start using the snapshot for annotating users and their LDAP attributes in inventories.

Allowed Tetration virtual appliances: None

Allowed connectors: AnyConnect, ISE, and F5.

18.4.3 Remove

All the configuration that are added can be removed from the connectors and/or appliances. There is a *Delete* button in each configuration that allows the user to remove the configuration.

18.5 Troubleshooting

Connectors and virtual appliances supports various troubleshooting mechanisms to debug possible issues.

18.5.1 Allowed set of commands

Allowed set of commands provide the ability to run some debug commands on the appliances and Docker containers (for connectors). These commands include from retrieving logs, current running configuration, testing network Connectivity and capturing packets matching a specified port.

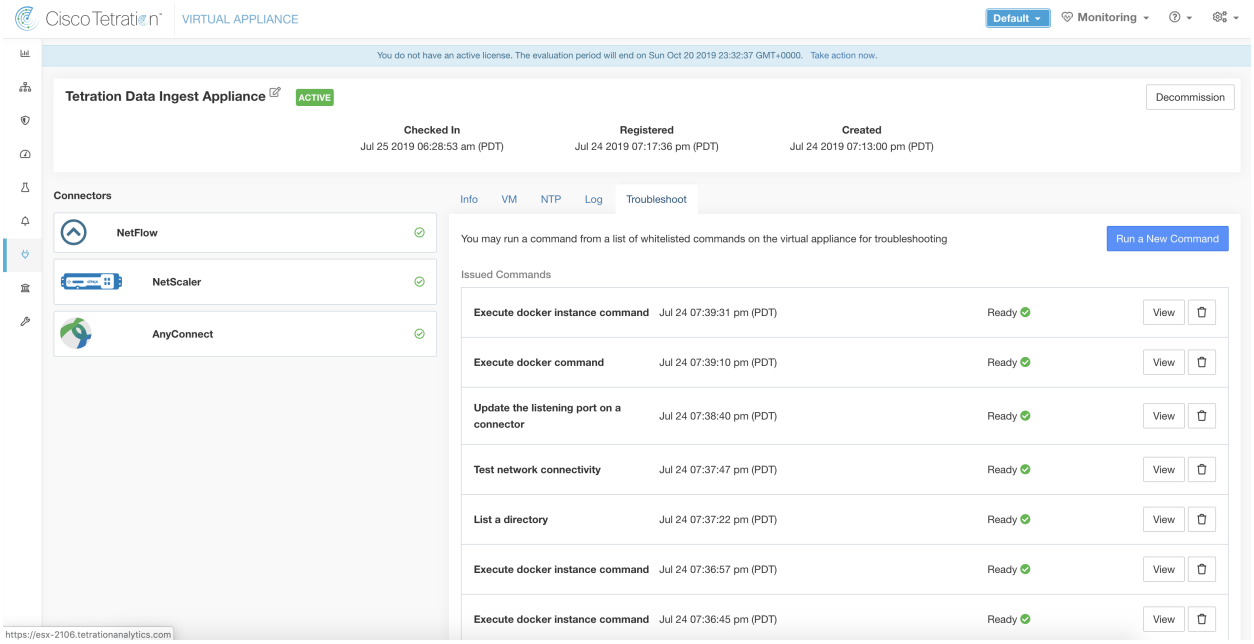


Fig. 18.5.1.1: Troubleshoot page on Tetration virtual appliance

Note: Troubleshooting using allowed set of commands is available on appliances and connectors only for users with *Customer Support* role.

18.5.1.1 Show Logs

Show the contents of a controller log file and optionally grep the file for a specified pattern. Tetration sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). When the result is available at Tetration, a download button is presented to download the file.

Argument Name	Type	Description
Grep Pattern	string	Pattern string to grep from the logfile

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

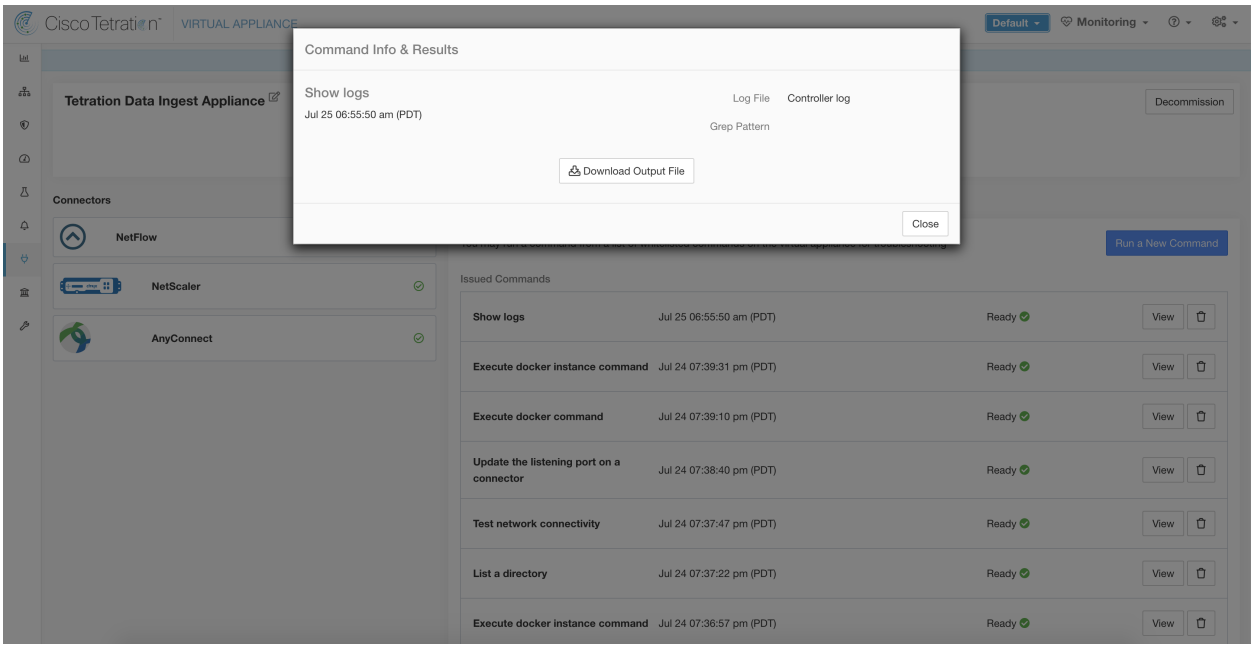


Fig. 18.5.1.1.1: Download *Show Logs* output from Tetration Ingest appliance

18.5.1.2 Show Service Logs

Show the contents of service log files and optionally grep the file for a specified pattern. Tetration sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). When the result is available at Tetration, a download button is presented to download the file.

Argument Name	Type	Description
Log File	dropdown	The name of the logfile to collect
	• *Service log*	Logs of the connector service
	• *Upgrade log*	Upgrade logs of the service
	• *LDAP loader log*	Logs of the LDAP snapshot for connectors that have LDAP enabled
Grep Pattern	string	Pattern string to grep from the logfile

Allowed Tetration virtual appliances: None (only available on valid connector services)

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

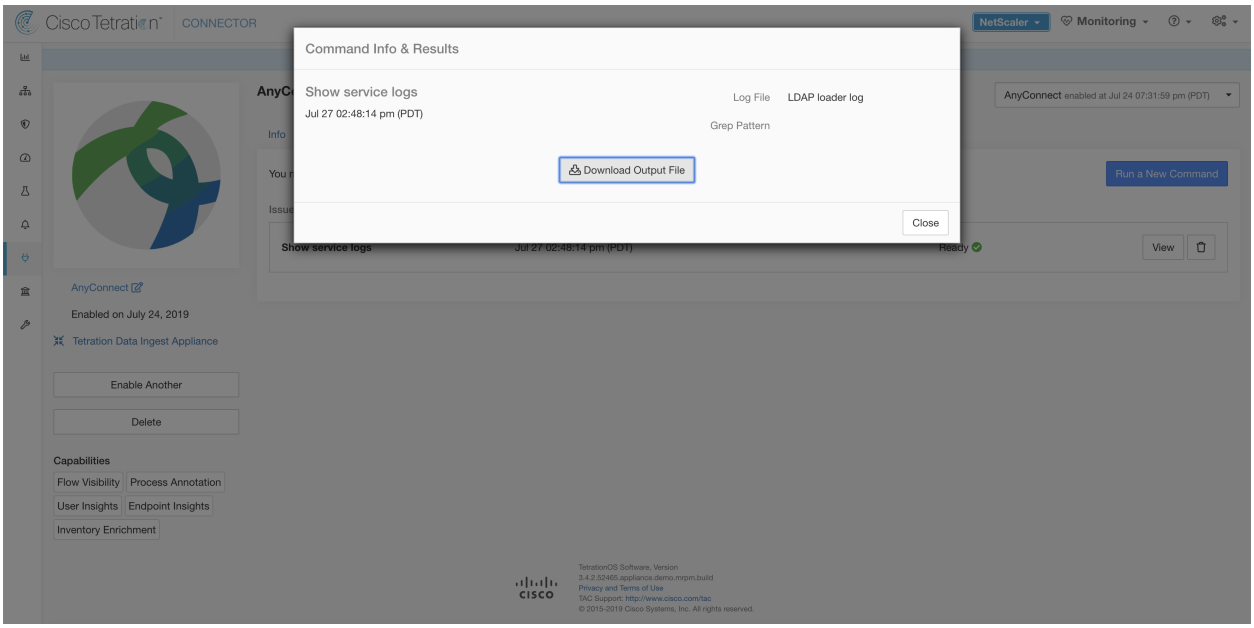


Fig. 18.5.1.2.1: Download *Show Service Logs* output from AnyConnect connector for *LDAP loader log* log file

18.5.1.3 Show AWS VPC FlowLogs Downloader logs

Show the contents of AWS downloader log file. Tetration sends the command to the AWS connector where the command was issued. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). When the result is available at Tetration, a download button is presented to download the file.

Argument Name	Type	Description
Log File	dropdown	The name of the logfile to collect
	• *S3 Downloader log*	Logs of the connector service
	• *Downloader buffer log*	Logs of the connector service
	• *List of skipped files*	Upgrade logs of the service
	• *API Stats*	Logs of the LDAP snapshot for

Allowed connectors: AWS

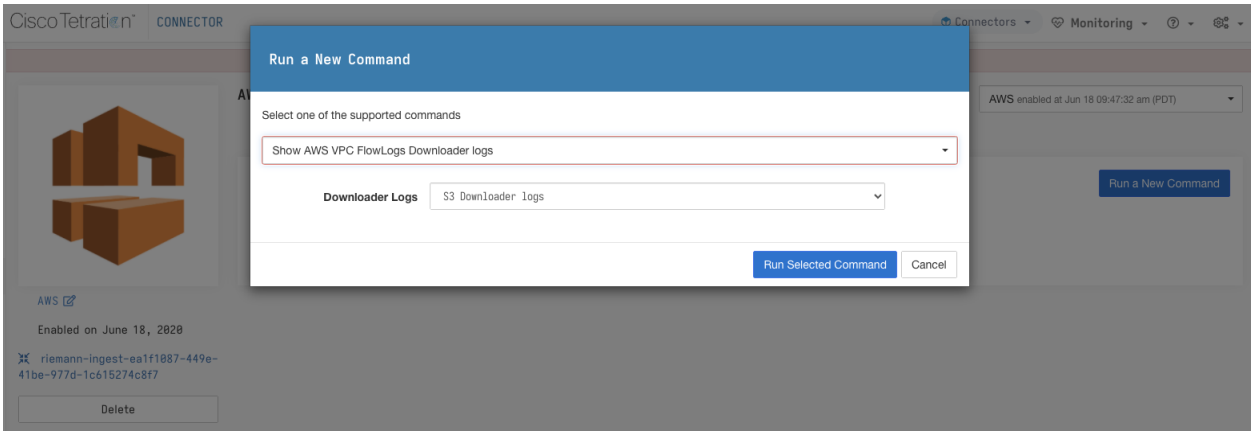


Fig. 18.5.1.3.1: Download *Show AWS VPC FlowLogs Downloader Logs* output from Tetration Ingest appliance

18.5.1.4 Show Running Configuration

Show running configuration of an appliance/connector controllers. The controller on appliance/connector retrieves the configuration corresponding to the requested argument and returns the result. When the result is available at Tetration, the contents of the configuration are shown in a text box.

Argument Name	Type	Description
Configuration Type	dropdown	Configuration file to collect
	• *Controller conf*	Configuration file of the appliance controller
	• *Supervisor conf*	Configuration file of the supervisor that runs the controller
	• *NTP conf*	NTP configuration file

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

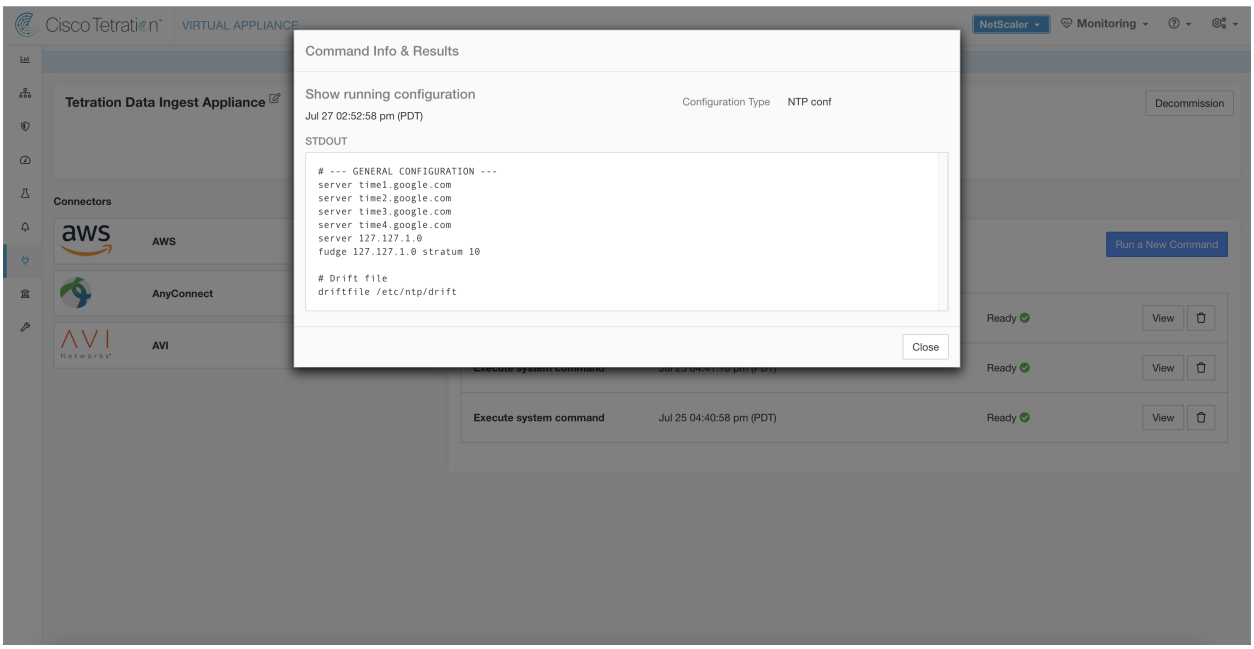


Fig. 18.5.1.4.1: Show running configuration for NTP conf on a Tetration Ingest Appliance

18.5.1.5 Show Service Running Configuration

Show running configuration of an services instantiated for connectors on the appliances. The controller on the service retrieves the configuration corresponding to the requested argument and returns the result. When the result is available at Tetration, the contents of the configuration are shown in a text box.

Argument Name	Type	Description
Configuration Type	dropdown	Configuration file to collect
	• *Controller conf*	Configuration file of the service controller
	• *Supervisor conf*	Configuration file of the supervisor that runs the controller
	• *Service conf*	Service configuration file
	• *LDAP conf*	LDAP configuration for connectors that have LDAP enabled.

Allowed Tetration virtual appliances: None (only available on valid connector services)

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

18.5.1.6 Show System Commands

Execute a system command and optionally grep for a specified pattern. The controller on the appliance/connector service returns the result (tailed for the last 5000 lines). Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
System Command	dropdown	System command to execute
	• *IP configuration*	`ifconfig`
	• *IP route configuration*	`ip route`
	• *IP packet filtering rules*	`iptables -L`
	• *Network status*	`netstat`
	• *Process status*	`ps -aux`
	• *List of top processes*	`top -b -n 1`
	• *NTP status*	`ntpstat`
	• *NTP query*	`ntpq -pn`
	• *CPU info*	`lscpu`
	• *Memory info*	`lsmem`
	• *Disk free*	`df -H`
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

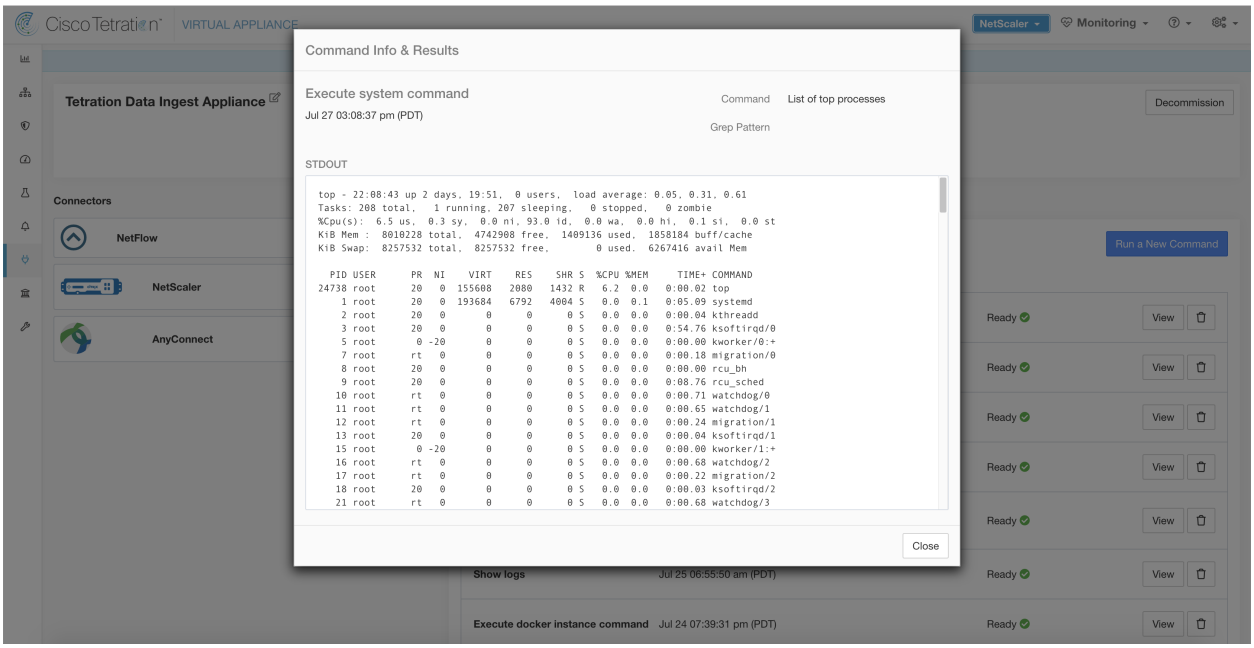


Fig. 18.5.1.6.1: Show system command on Tetration Ingest appliance to retrieve list of top processes

18.5.1.7 Show Docker Commands

Execute a Docker command and optionally grep for a specified pattern. The command is executed on the appliance by the appliance controller. The result tailed for the last 5000 lines. Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Docker Command	dropdown	Docker command to execute
	• *Docker info*	`docker info`
	• *List images*	`docker images --no-trunc`
	• *List containers*	`docker ps --no-trunc`
	• *List networks*	`docker network ls --no-trunc`
	• *List volumes*	`docker volume ls`
	• *Container stats*	`docker stats --no-trunc --no-stream`
	• *Docker disk usage*	`docker system df -v`
	• *Docker system events*	`docker system events --since '10m'`
	• *Version*	`docker version`
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: All

Allowed connectors: None

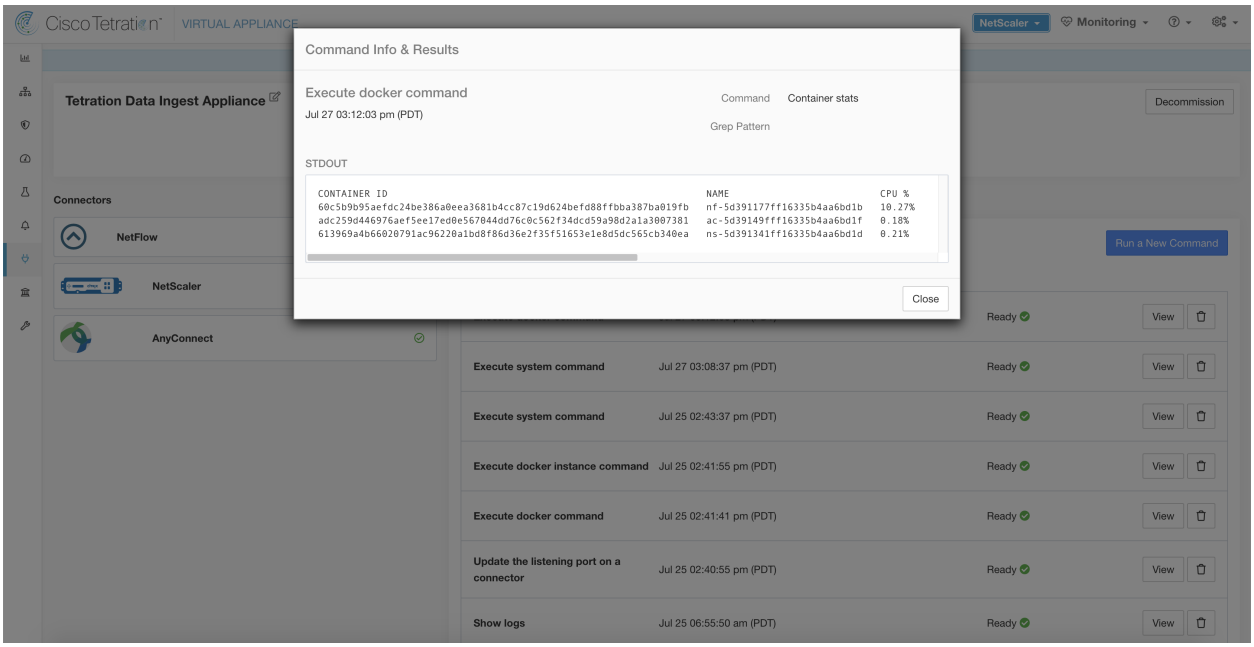


Fig. 18.5.1.7.1: Execute a docker command on Tetration Ingest appliance to show container stats

18.5.1.8 Show Docker Instance Commands

Execute a docker command on a specific instance of a Docker resource. The instance ID can be fetched using *Show Docker Commands*. The command is executed on the appliance by the appliance controller. The result tailed for the last 5000 lines. Optionally, a grep pattern can be provided as argument and the output is filtered accordingly. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Docker Command	dropdown	Docker command to execute
	• *Image info*	`docker images --no-trunc <instance>`
	• *Network info*	`docker network inspect <instance>`
	• *Volume info*	`docker volume inspect <instance>`
	• *Container info*	`docker container inspect --size <instance>`
	• *Container logs*	`docker logs --tail 5000 <instance>`
	• *Container port mappings*	`docker port <instance>`
	• *Container resource usage stats*	`docker stats --no-trunc --no-stream <instance>`
	• *Container running processes*	`docker top <instance>`
Instance	string	Docker resource (image, network, volume, container) ID (See *Show Docker Commands*)
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: All

Allowed connectors: None

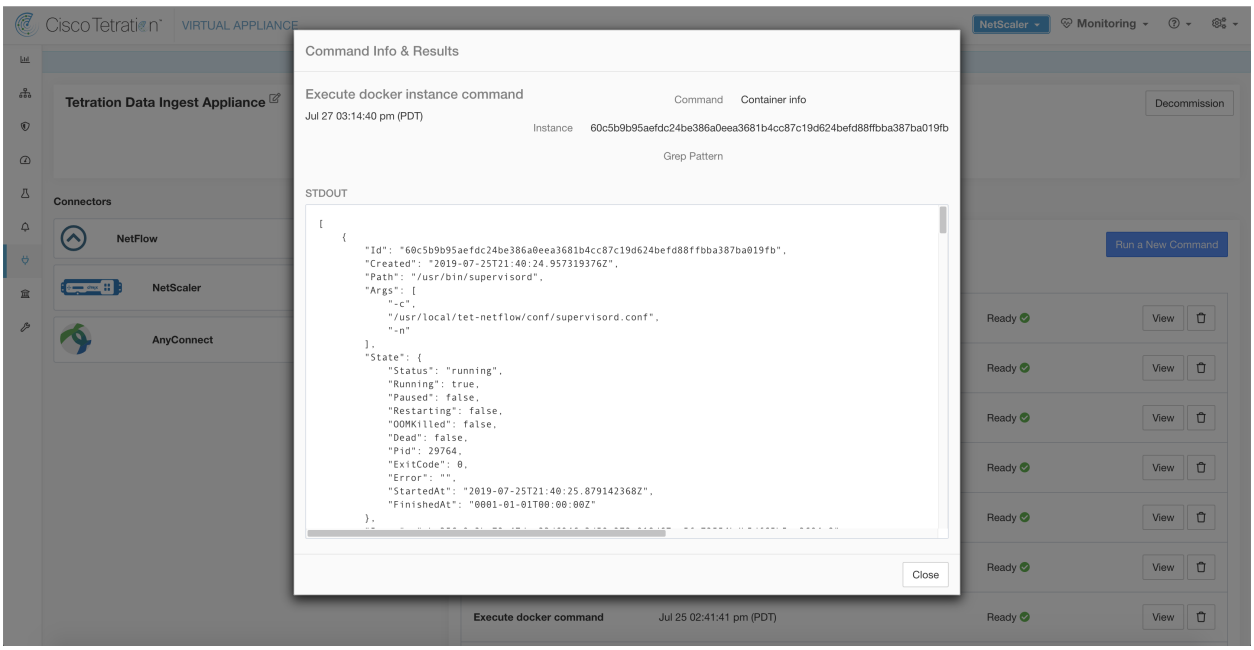


Fig. 18.5.1.8.1: Execute a docker instance command on Tetration Ingest appliance to retrieve container info

18.5.1.9 Show Supervisor Commands

Execute a supervisorctl command and return the result. Tetration sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the result When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
SupervisorCtl Command	dropdown	*supervisorctl* command to execute
	• *Status of all services*	`supervisorctl status`
	• *PID of supervisor*	`supervisorctl pid`
	• *PID of all services*	`supervisorctl pid all`

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

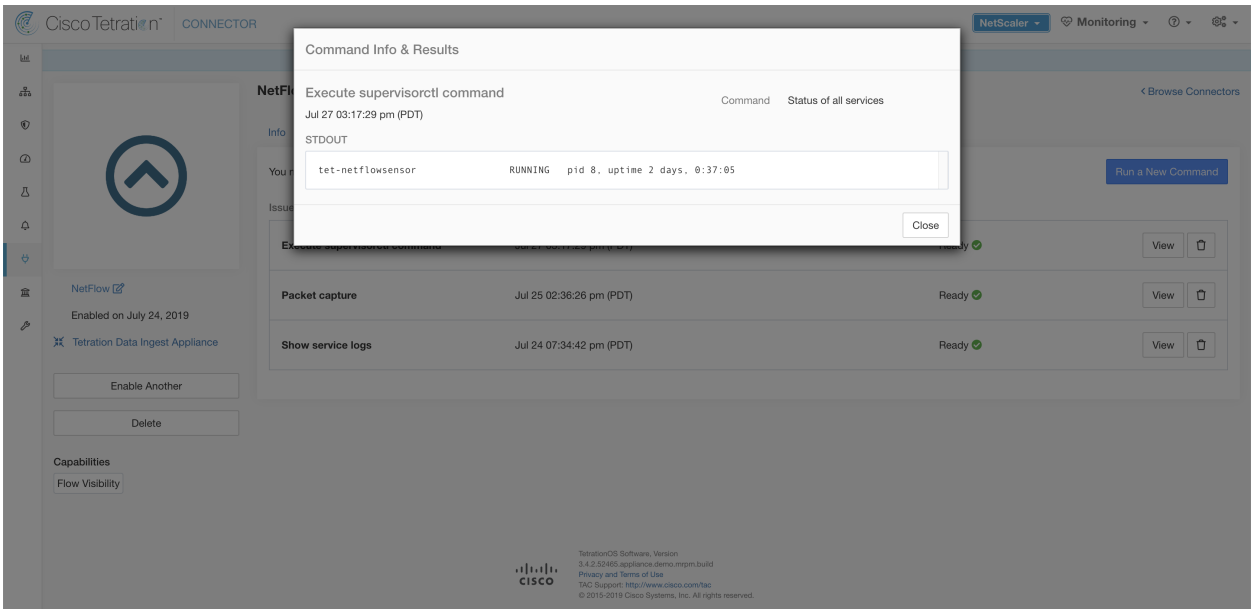


Fig. 18.5.1.9.1: Execute supervisorctl command on NetFlow connector to get the status of all services

18.5.1.10 Show Supervisor Service Commands

Execute a supervisorctl command on a specific service. The service name can be fetched using *Show Supervisor Commands*. Tetration sends the command to appliance/connector where the command was issued. The controller on the appliance/connector service returns the resultWhen the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
SupervisorCtl Command	dropdown	*supervisorctl* command to execute
	• *Status of a service*	`supervisorctl status <service name>`
	• *PID of a service*	`supervisorctl pid <service name>`
Service name	string	Name of the supervisor controlled service (see *Show Supervisor Commands*)

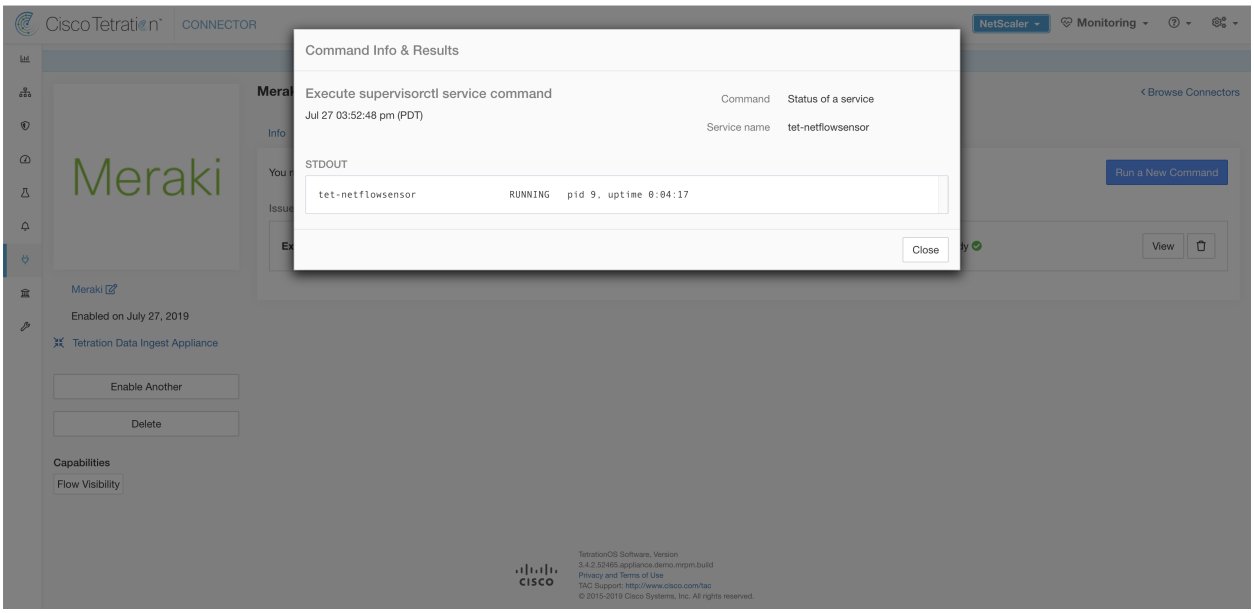


Fig. 18.5.1.10.1: Execute supervisorctl command on NetFlow connector to get the status of specified service name

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

18.5.1.11 Network Connectivity Commands

Test network connectivity from the appliance/connector. The command is executed on the appliance by the appliance controller. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Network Command	dropdown	Network connectivity command to execute
	• *ping*	`ping -c 5 <destination>`
	• *curl*	`curl -I <destination>`
Destination	string	Destination to use for the test

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

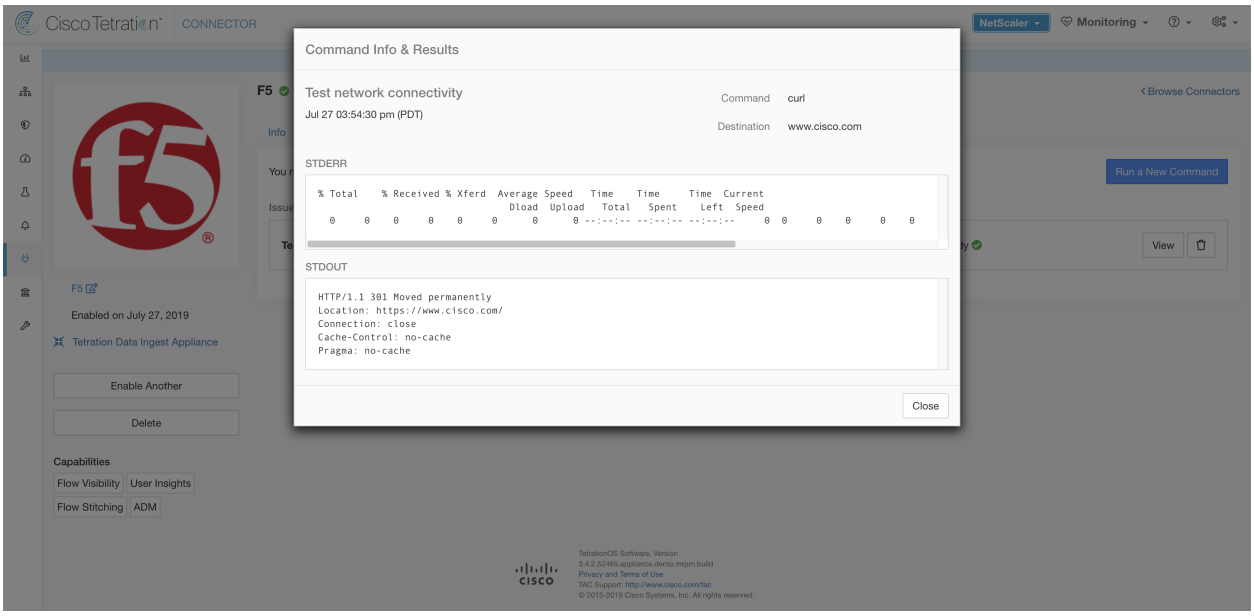


Fig. 18.5.1.11.1: Test network connectivity on F5 connector by running a curl

18.5.1.12 List Files

List the files in well known locations of the appliance. Optionally, grep for a specified pattern. Tetration sends the command to appliance where the command was issued. The controller on the appliance returns the result. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Location	dropdown	List files in a target location
	• *Controller configuration folder*	List the contents in the folder where controller configuration files are kept.
	• *Controller cert folder*	List the contents in the folder where controller certs are kept.
	• *Log folder*	List the contents in the folder where log files are present.
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: All

Allowed connectors: None

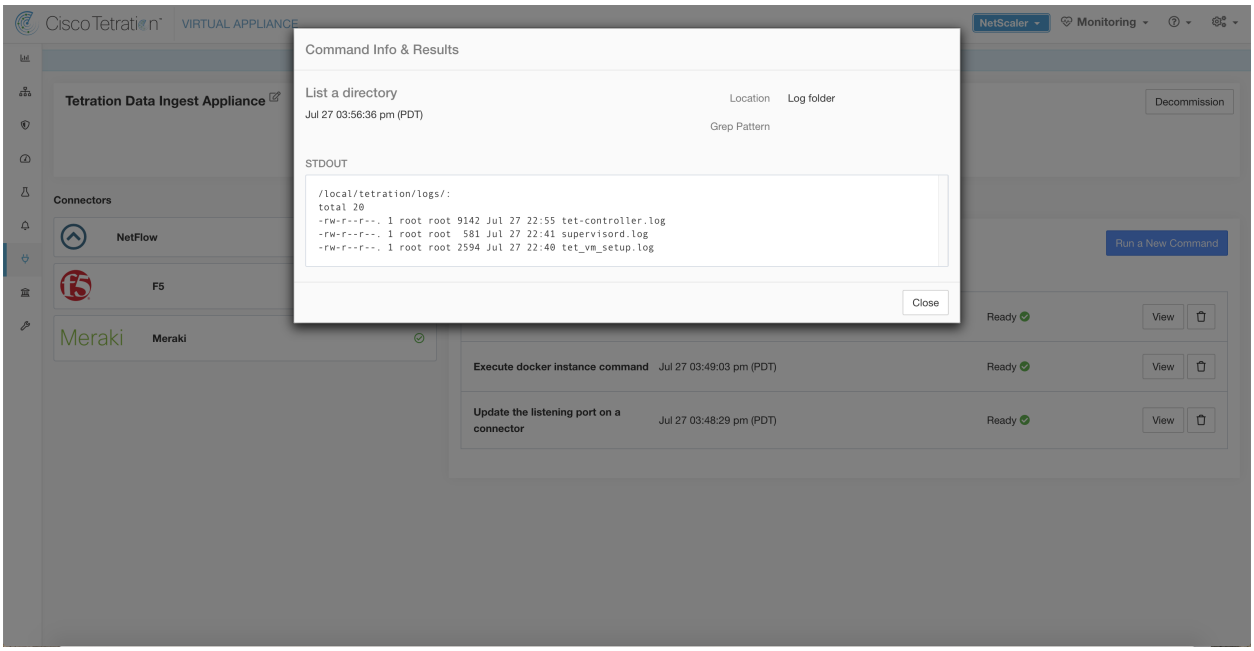


Fig. 18.5.1.12.1: List the files in log folder in Tetration Ingest appliance

18.5.1.13 List Service Files

List the files in well known locations of the connector service. Optionally, grep for a specified pattern. Tetration sends the command to connector where the command was issued. The controller on the connector service returns the result. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Location	dropdown	List files in a target location
	• *Service configuration folder*	List the contents in the folder where service configuration files are kept.
	• *Service cert folder*	List the contents in the folder where service certs are kept.
	• *Log folder*	List the contents in the folder where log files are present.
	• *DB folder*	List the contents in the folder where state of endpoints (esp. for AnyConnect and ISE connectors) are kept.
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: None

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

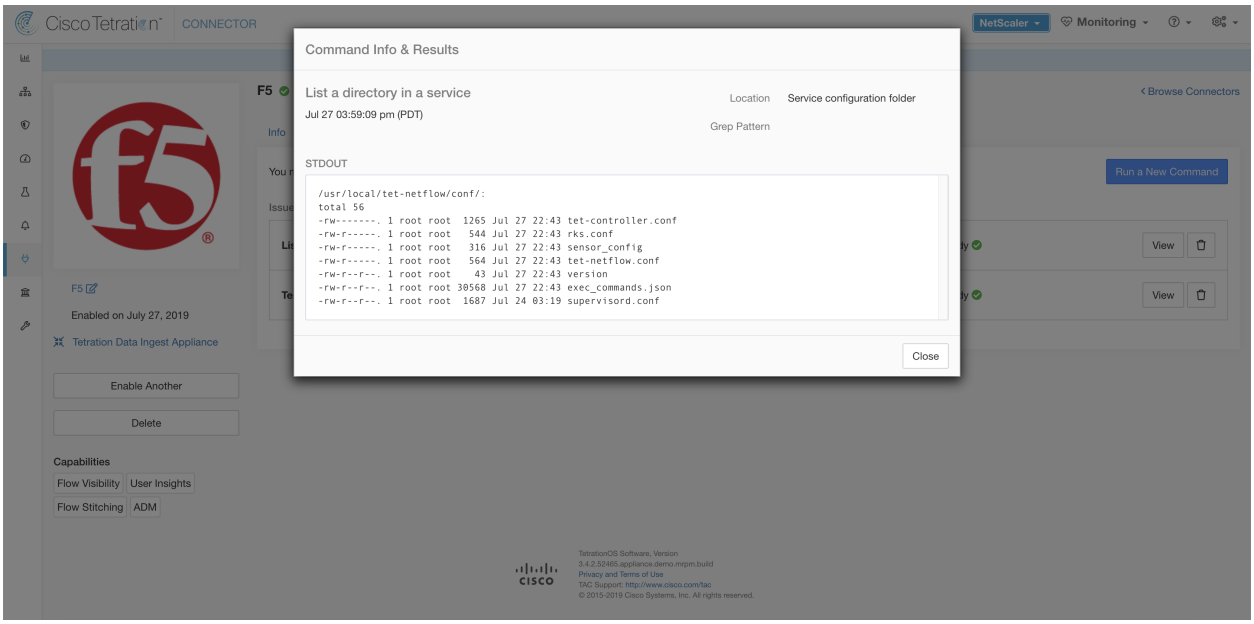


Fig. 18.5.1.13.1: List the files in configuration folder of F5 connector in Tetration Ingest appliance

18.5.1.14 Packet Capture

Capture incoming packets on an appliance/connector. Tetration sends the command to the appliance/connector where the command was issued. The controller on the appliance/connector service captures packets, encodes them and returns the result to Tetration. When the result is available at Tetration, a download button is presented to download the file in `.pcap` format.

Argument Name	Type	Description
Listening port	number	Capture packets that are sent/received on this port
Max packets to collect	number	Maximum packets to collect before returning the result. Should be < 1000
Max collection duration in seconds	number	Maximum duration to collect before return the result. Should be < 600 seconds.

Allowed Tetration virtual appliances: All

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, and Meraki.

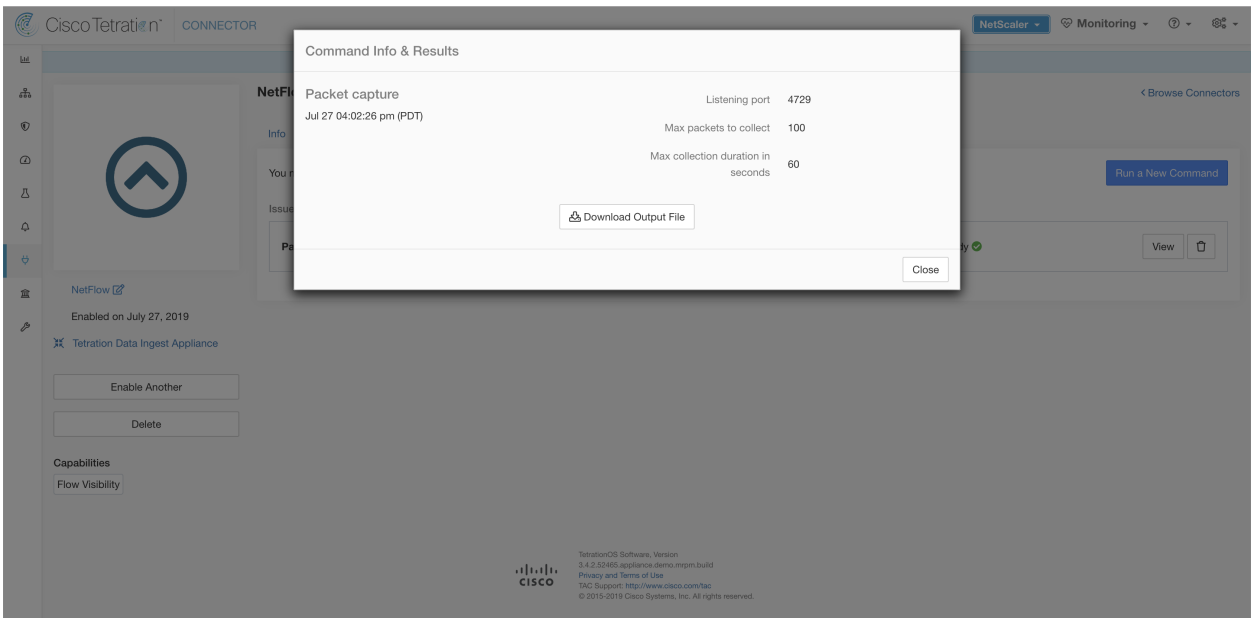


Fig. 18.5.1.14.1: Capture packets on a given port on NetFlow connector

18.5.1.15 Show Data Export Logs

Show the contents of Docker logs for various Docker containers running in a Tetration Export appliance. Optionally grep the file for a specified pattern. Tetration sends the command to appliance controller on Tetration Export appliance. The result is tailed for the last 5000 lines. When the result is available at Tetration, a download button is presented to download the file.

Argument Name	Type	Description
Service name	dropdown	Name of the Docker container
	• *logstash1*	`docker logs --tail 5000 logstash1`
	• *logstash2*	`docker logs --tail 5000 logstash2`
	• *elasticsearch*	`docker logs --tail 5000 elasticsearch`
	• *grafana*	`docker logs --tail 5000 grafana`
	• *curator*	`docker logs --tail 5000 curator`
Grep Pattern	string	Pattern string to grep from logs

Allowed Tetration virtual appliances: Tetration Export

Allowed connectors: None

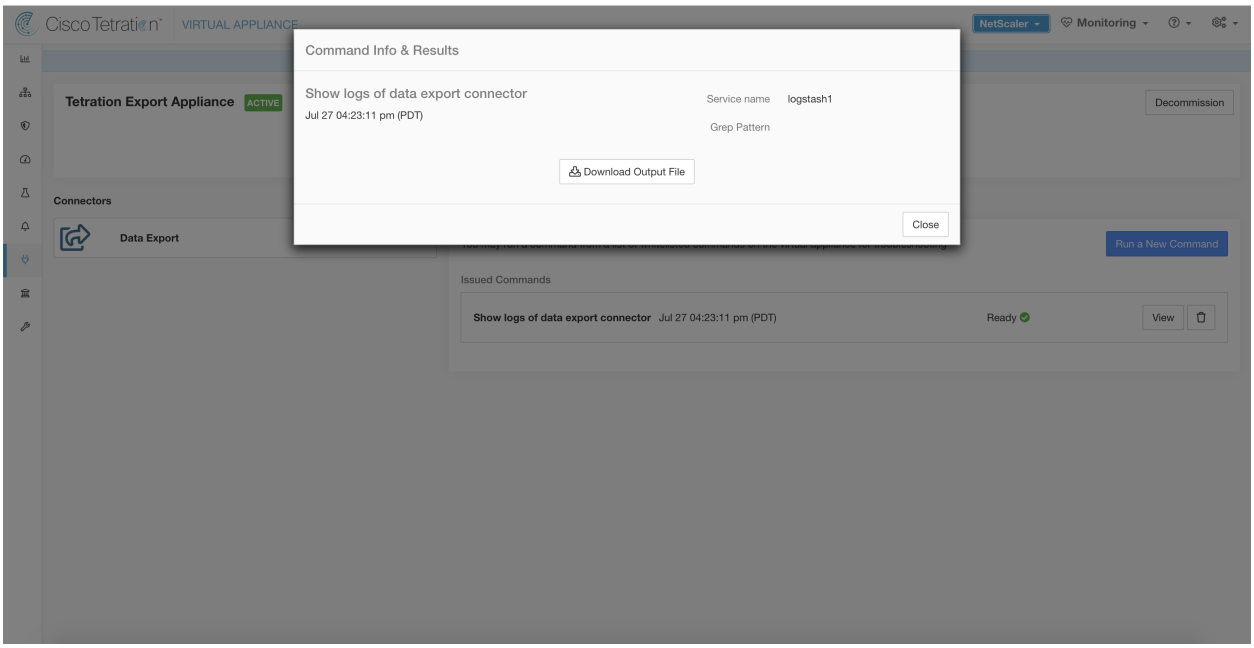


Fig. 18.5.1.15.1: Download logs from logstash1 Docker in Tetration Export appliance

18.5.1.16 Show Data Export Running Configuration

Show running configuration of in logstash services on Tetration Export appliance. The controller on Tetration Export appliance retrieves the configuration corresponding to the requested argument and returns the result. When the result is available at Tetration, the contents of the configuration are shown in a text box.

Argument Name	Type	Description
Service name	dropdown	Logstash instance
	• *logstash1*	Logstash instance #1
	• *logstash2*	Logstash instance #2
Config file Type	dropdown	Config to fetch
	• *logstash*	Logstash configuration file
	• *pipeline*	Logstash pipeline configuration file
	• *kafka*	Kafka configuration for Logstash

Allowed Tetration virtual appliances: Tetration Export

Allowed connectors: None

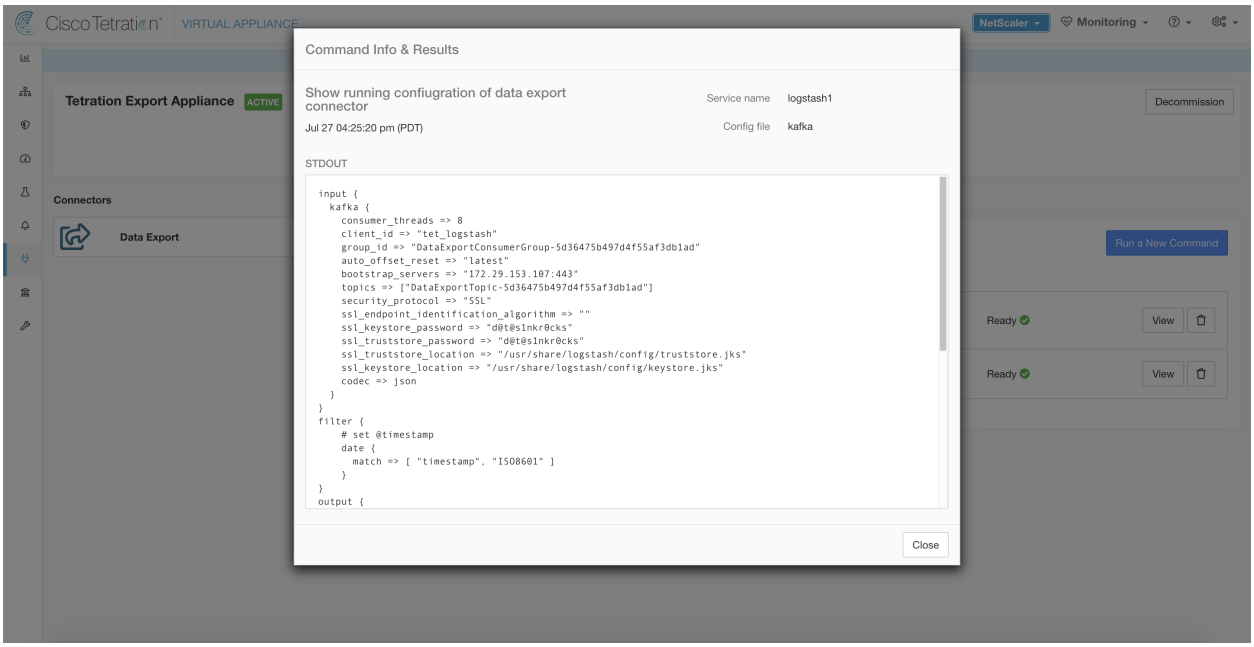


Fig. 18.5.1.16.1: Show running conf for Kafka on logstash1 Docker in Tetration Export appliance

18.5.1.17 Show Data Export System Commands

Execute a system command on Tetration Export appliance and optionally grep for a specified pattern. The controller on Tetration Export appliance executes the system command and returns the result. When the result is available at Tetration, the contents of the configuration are shown in a text box.

Argument Name	Type	Description
Service name	dropdown	Name of the Docker container
	• *logstash1*	Logstash instance #1
	• *logstash2*	Logstash instance #2
	• *elasticsearch*	Elasticsearch instance
	• *grafana*	Grafana instance
	• *curator*	Elasticsearch Retention service instance
System Command	dropdown	System command to execute
	• *Process status*	`ps -aux` on the target container
	• *List of top processes*	`top -b -n 1` on the target container
	• *CPU info*	`lscpu` on the target container
	• *Memory info*	`lsmem` on the target container
	• *Disk free*	`df -H` on the target container
Grep Pattern	string	Pattern string to grep from the output

Allowed Tetration virtual appliances: Tetration Export

Allowed connectors: None

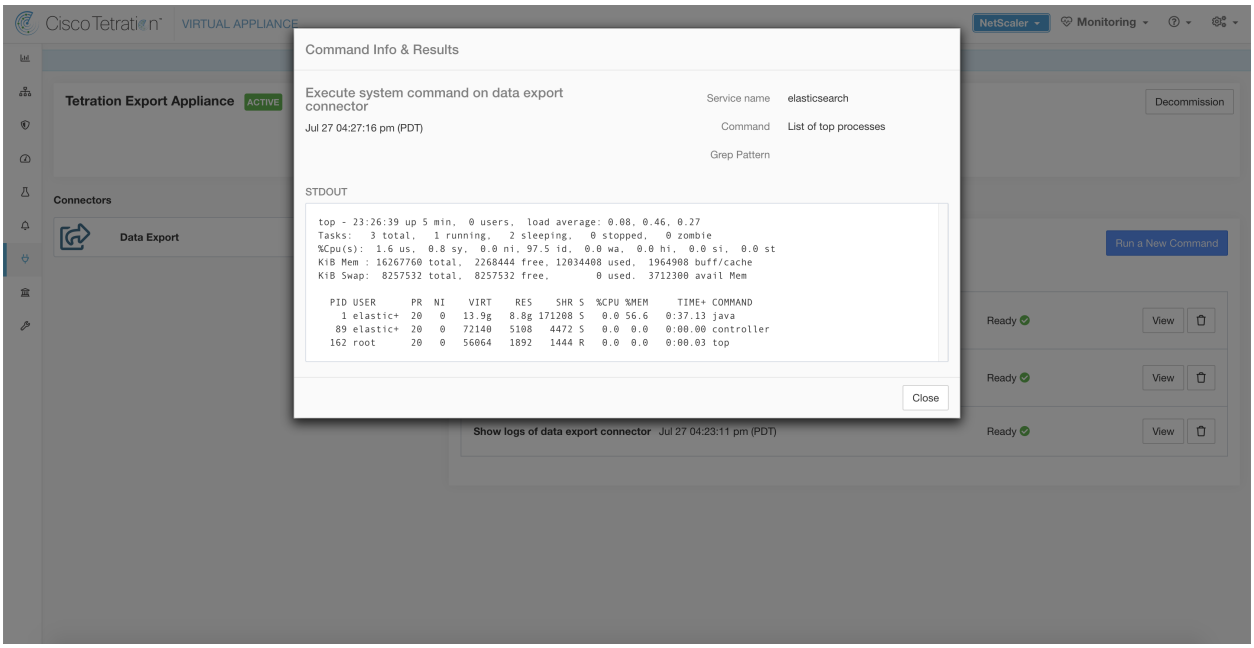


Fig. 18.5.1.17.1: Execute a system command on elasticsearch Docker to retrieve the list of top processes

18.5.1.18 Update Listening Ports of Connectors

Update the listening port on a connector in Tetration Ingest appliance. Tetration sends the command to the appliance controller on the appliance where the command is issued. The controller does the following actions:

- Stops the Docker service corresponding to the connector.
- Collect the current running configuration of the service.
- Remove the Docker service.
- Update the running configuration of the service to use the new ports.
- Start a new container from the same Docker image that was used in the removed container with new exposed ports. Also, if a Docker volume was mounted to the removed container earlier, the same volume is mounted to the new container.
- Return the new IP bindings of the connector to Tetration.
- Tetration shows the result in a text box.

Argument Name	Type	Description
Connector ID	string	Connector ID of the connector for which listening ports need to be updated
Listening port label	dropdown	The type of port that is updated.
	NET-FLOW9	NetFlow v9 listening port
	IPFIX	IPFIX listening port
Listening port	string	New port for the connector

Allowed Tetration virtual appliances: Tetration Ingest

Allowed connectors: None

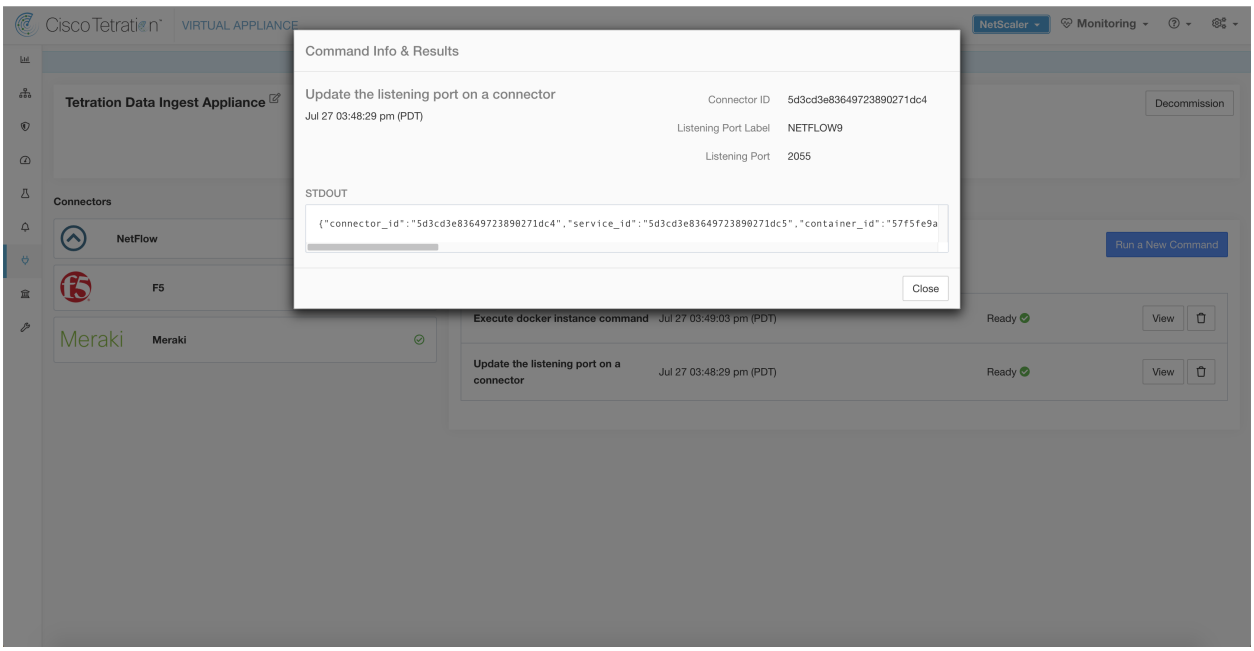


Fig. 18.5.1.18.1: Update listening port on Meraki connector to 2055 in Tetration Ingest appliance

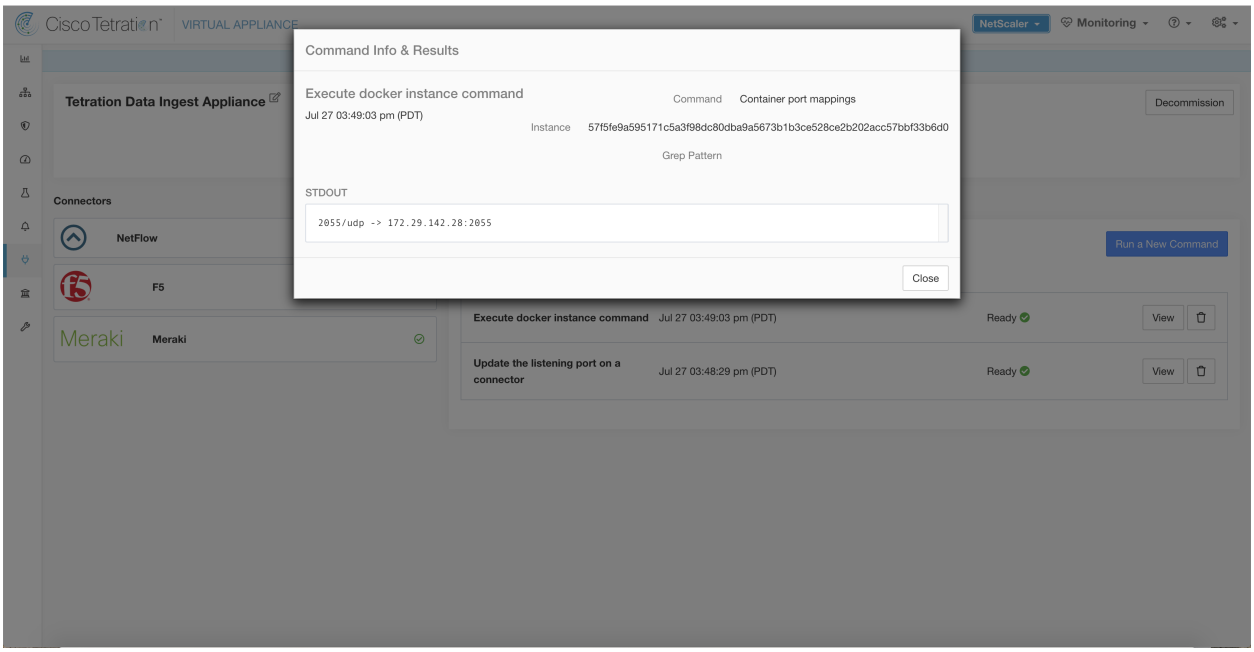


Fig. 18.5.1.18.2: Retrieve the port mappings on Meraki connector in Tetration Ingest appliance

18.5.1.19 Update Alert Notifier Connector Log Configuration

Update log configuration for Tetration Alert Notifier (TAN) service that hosts Syslog, Email, Slack, PagerDuty, and Kinesis alert notifier connectors. Since TAN hosts multiple connectors, log configuration cannot be updated from connector page directly. This allowed command allows the user to update the log configuration.

Tetration sends the command to the service controller on TAN Docker service of Tetration Edge appliance. The controller applies the configuration on the service and returns the status of the configuration update.

Argument Name	Type	Description
Logging level	dropdown	Logging level to be used by the service
	• *debug*	Debug log level
	• *info*	Informational log level
	• *warn*	Warning log level
	• *error*	Error log level
Max log file size (in MB)	number	Maximum size of a log file before log rotation kicks in
Log rotation (in days)	number	Maximum age of a log file before log rotation kicks in
Log rotation (in instances)	number	Maximum instances of log files kept

Allowed Tetration virtual appliances: Tetration Edge

Allowed connectors: None

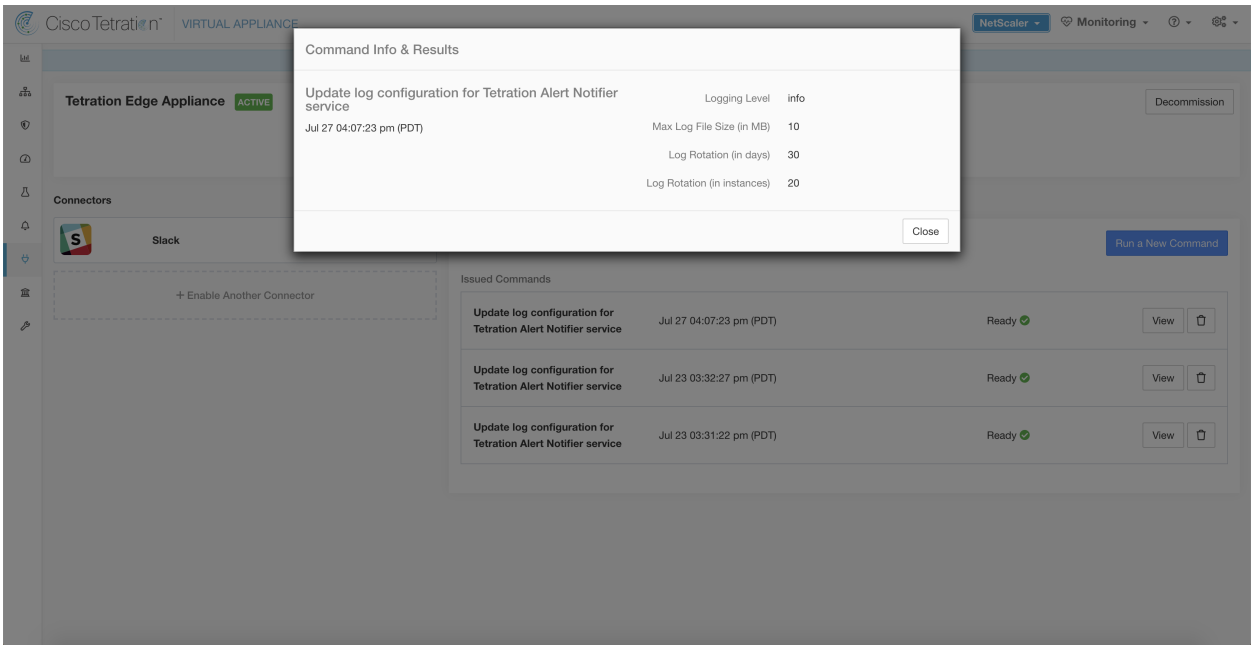


Fig. 18.5.1.19.1: Update the log configuration on Tetration Alert Notifier Docker service in Tetration Edge appliance

18.5.1.20 Collect Snapshot From Appliance

Tetration sends the command to the appliance where the command was issued. When the controller on the appliance receives this command from Tetration, it collects appliance snapshot, encodes them and returns the result to Tetration.

When the result is available at Tetration, a download button is presented to download the file in `.tar.gz` format.

Files included in the snapshot:

- /local/tetration/appliance/appliance.conf
- /local/tetration/{logs, sqlite, user.cfg}
- /opt/tetration/tet_vm_setup/conf/tet-vm-setup.conf
- /opt/tetration/tet_vm_setup/docker/Dockerfile
- /opt/tetration/ova/version
- /usr/local/tet-controller/conf
- /usr/local/tet-controller/cert/{topic.txt, kafkaBrokerIps.txt}
- /var/run/supervisord.pid

Command outputs included in the snapshot:

- ps aux
- iptables -L
- netstat {-nat, -rn, -suna, -stna, -tunlp}
- /usr/local/tet-controller/tet-controller -version
- supervisorctl status
- rpm -qi tet-nic-driver tet-controller
- du -shc /local/tetration/logs
- ls {/usr/local/tet-controller/cert/, -l /local/tetration/sqlite/, -l /opt/tetration/tet_vm_setup/.tet_vm.done, -l /opt/tetration/tet_vm_setup/templates/}
- docker {images, ps -a}
- blkid/ifconfig/lscpu/uptime
- free -m
- df -h

Argument Name	Type	Description
Max time for collection in minutes	number	Maximum duration to collect before returning the results. Should be < 20 minutes.

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

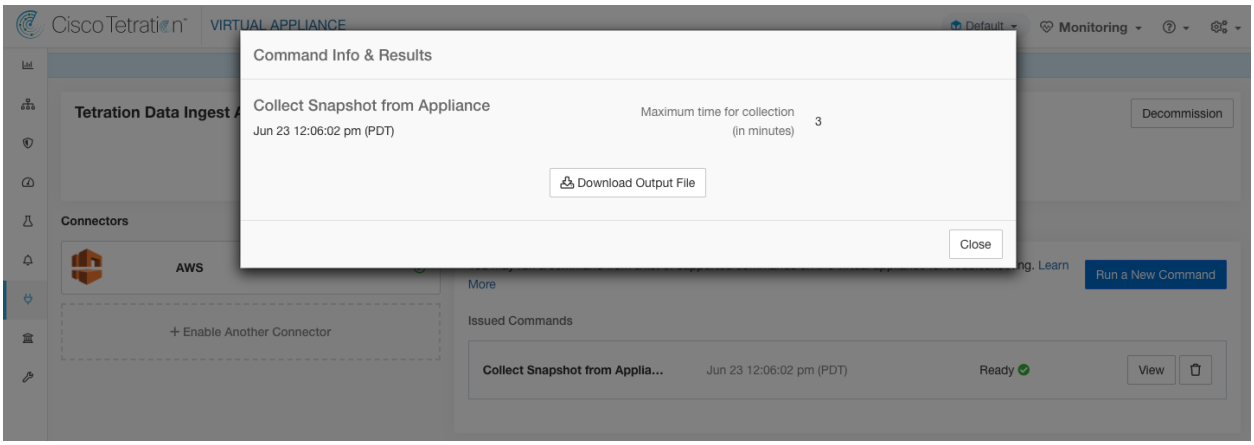


Fig. 18.5.1.20.1: Collect snapshot from Tetration appliance

18.5.1.21 Collect Snapshot From Connector

Tetration sends the command to the appliance where the connector is deployed. According to connector ID, the controller collects connector snapshot, encodes them and returns the result to Tetration. When the result is available at Tetration, a download button is presented to download the file in `.tar.gz` format.

Files included in the snapshot:

- /usr/local/tet-netflow/conf
- /local/tetration/{logs, sqlite}
- /var/run/{supervisord.pid, tet-netflow.pid}

Command outputs included in the snapshot:

- ps aux
- netstat {-nat, -rn, -suna, -stna, -tunlp}

Argument Name	Type	Description
Connector ID	string	Connector ID of the connector for which the snapshot command is run.
Capture packets	check-box	Should packets be captured?
Max time for collection in minutes	number	Maximum duration to collect before returning the results. Should be < 20 minutes.

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

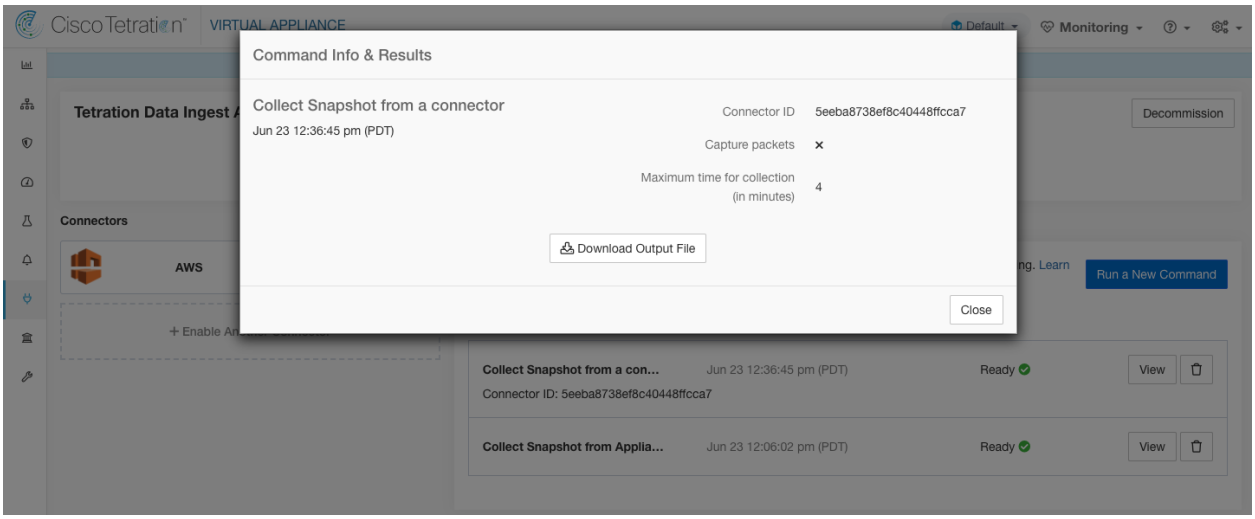


Fig. 18.5.1.21.1: Collect snapshot from Tetration connector on designated connector ID

18.5.1.22 Collect Controller Profile

Collect controller process profiling result on appliance or connectors. Tetration sends the command to the connector where the command was issued. The service controller restarts the connector service in the specified profiling mode. After collecting the profiling result, service controller restarts the service in normal mode and send the result to Tetration. When the result is available at Tetration, a download button is presented to download the file in `.tar.gz` format.

Argument Name	Type	Description
Profile Mode	dropdown	Profiling mode.
	• *memory*	Memory profiling mode.
	• *cpu*	CPU profiling mode.
	• *block*	Block profiling mode.
	• *mutex*	Mutex profiling mode.
	• *goroutine*	Goroutine profiling mode.
Maximum time for collection (in minutes)	number	Maximum duration to collect before returning the result.
Memory profile rate (only valid when choosing "memory" mode)	number	Memory profiling rate. This field is optional. If not provided, default value in Golang will be used.

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, and Meraki.

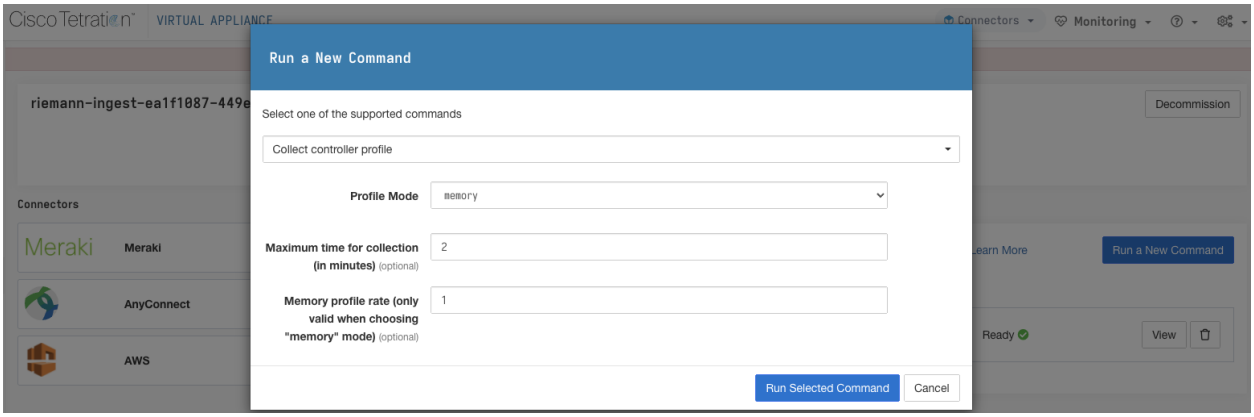


Fig. 18.5.1.22.1: Collect controller profile from Tetration appliance

18.5.1.23 Collect Connector Profile

Collect connector process profiling result on connectors. Tetration sends the command to the connector where the command was issued. The service controller restart the connector service in the specified profiling mode. After collecting the profiling result, service controller restart the service in normal mode and send the result to Tetration. When the result is available at Tetration, a download button is presented to download the file in `.tar.gz` format.

Argument Name	Type	Description
Profile Mode	dropdown	Profiling mode.
	• *memory*	Memory profiling mode.
	• *cpu*	CPU profiling mode.
	• *block*	Block profiling mode.
	• *mutex*	Mutex profiling mode.
	• *goroutine*	Goroutine profiling mode.
Maximum time for collection (in minutes)	number	Maximum duration to collect before returning the result.
Memory profile rate (only valid when choosing "memory" mode)	number	Memory profiling rate. This field is optional. If not provided, default value in Golang will be used.

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, and Meraki.

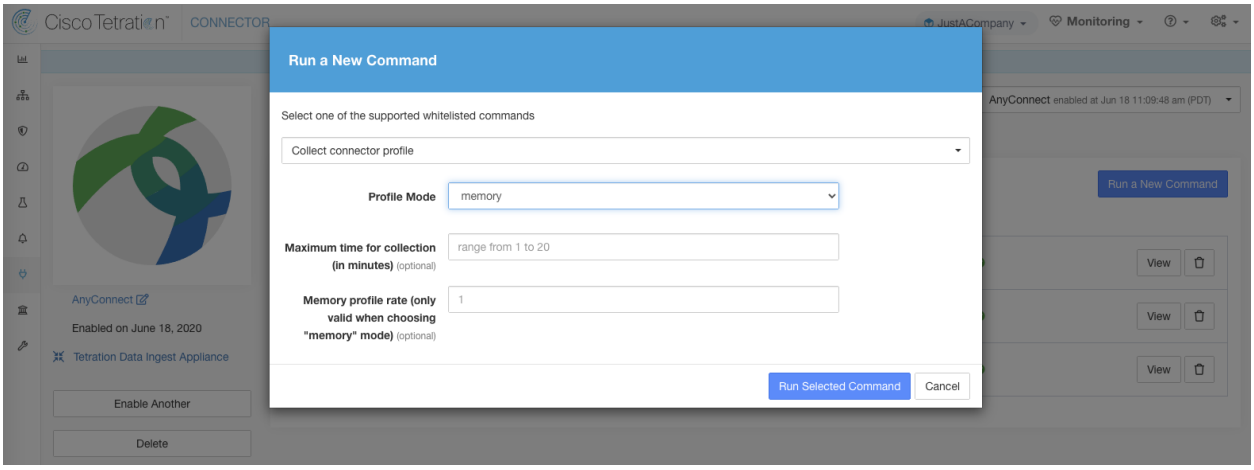


Fig. 18.5.1.23.1: Collect connector profile from Tetration connector

18.5.1.24 Override connector alert interval for Appliance

Override default connector alert interval for appliance. Tetration restricts same connector alert to send only once a day in default. This command is for administrator to override interval when they think once a day is too long. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Alert Type	dropdown	The connector alert type to override.
	• *Check-in missed*	Miss appliance's check-in.
	• *CPU usage*	High CPU usage.
	• *Memory usage*	High memory usage.
	• *Disk usage*	High disk usage.
Interval (in minutes)	number	Duration to override interval in minutes.

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: None

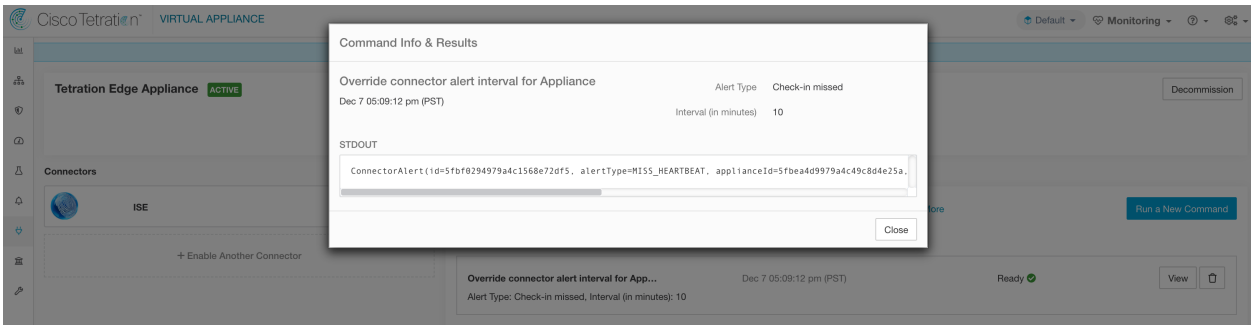


Fig. 18.5.1.24.1: Override connector alert interval for Tetration appliance

18.5.1.25 Override connector alert interval for Connector

Override default connector alert interval for connector. Tetration restricts same connector alert to send only once a day in default. This command is for administrator to override interval when they think once a day is too long. When the result is available at Tetration, the result is shown in a text box.

Argument Name	Type	Description
Alert Type	dropdown	The connector alert type to override.
	• *Check-in missed*	Miss connector's check-in.
Interval (in minutes)	number	Duration to override interval in minutes.

Allowed Tetration virtual appliances: None

Allowed connectors: NetFlow, NetScaler, F5, AWS, AnyConnect, Syslog, Email, Slack, PagerDuty, Kinesis, ISE, ASA, Meraki, ServiceNow, WAD.

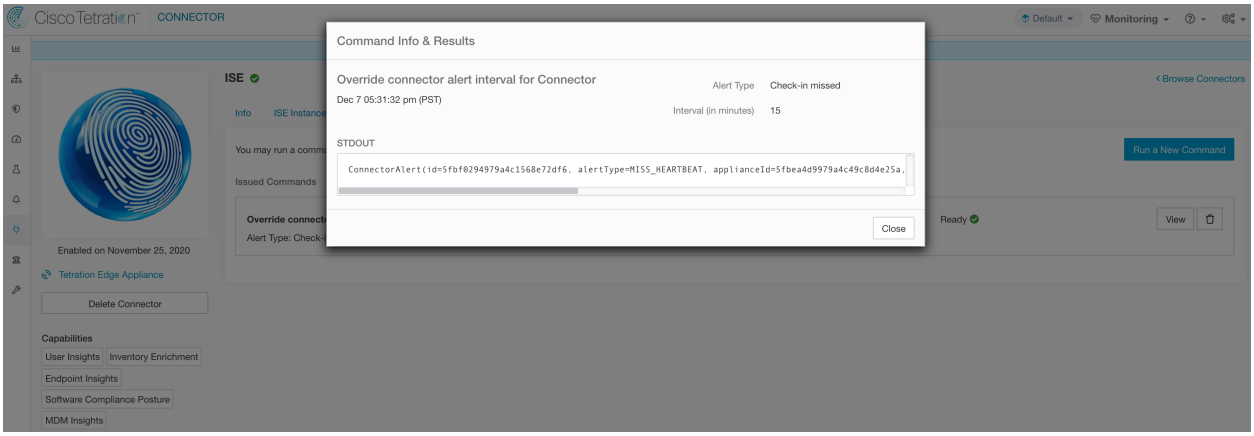


Fig. 18.5.1.25.1: Override connector alert interval for Tetration connector

18.5.2 Hawkeye Dashboards

Hawkeye dashboards provide insights about health of the connectors and virtual appliances where the connectors are enabled.

18.5.2.1 Appliance Controller Dashboard

Appliance controller dashboard provides information about network statistics, system metrics such as CPU usage percentage, memory usage percentage, disk usage percentage, and number of open file descriptors.

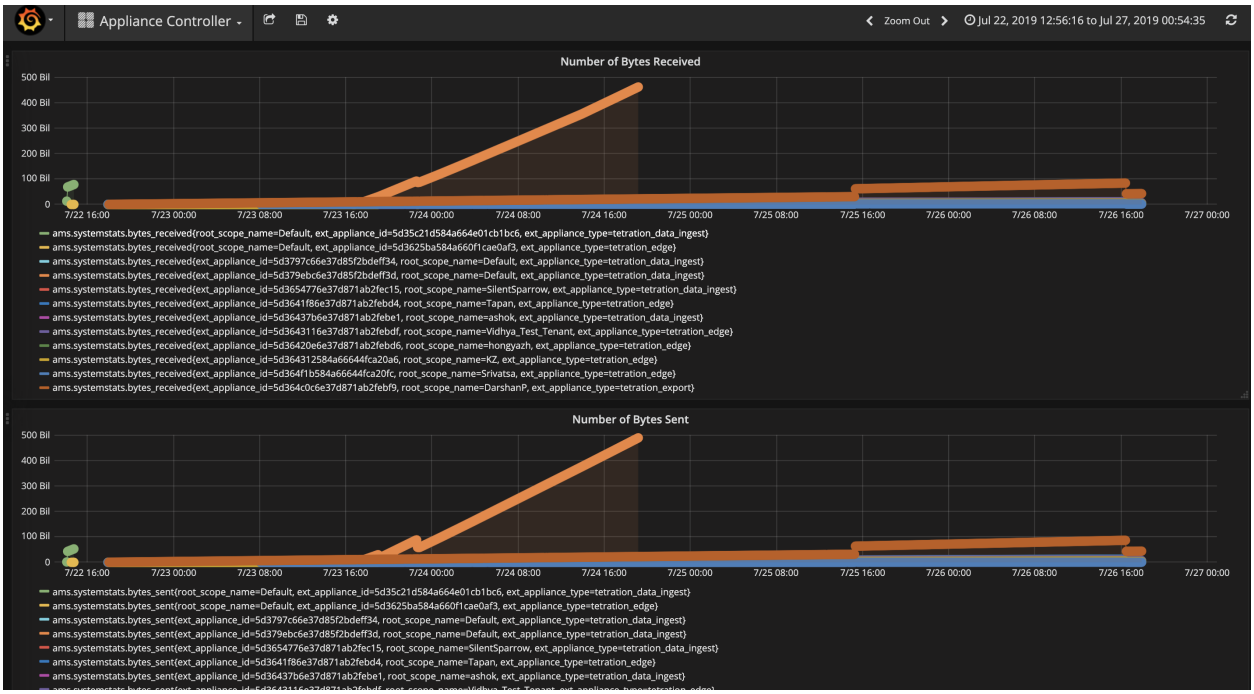


Fig. 18.5.2.1.1: Appliance controller dashboard

18.5.2.2 Service Dashboard

Service dashboard provides information about export metrics -if applicable- including number of flow observations exported to Tetration, number of packets exported to Tetration, and number of bytes exported to Tetration. In addition, this dashboard also provides information about protocol processing and decoding (for example, services that process NetFlow v9, IPFIX, and AWS VPC flow logs). Metrics such as decoded count, decoded error count, flow count, packet count, and byte count are available in this dashboard. Furthermore, system metrics for the Docker container where the service is running are also included in this dashboard. Metrics such as CPU usage percentage, memory usage percentage, disk usage percentage, and number of open file descriptors are part of this dashboard.

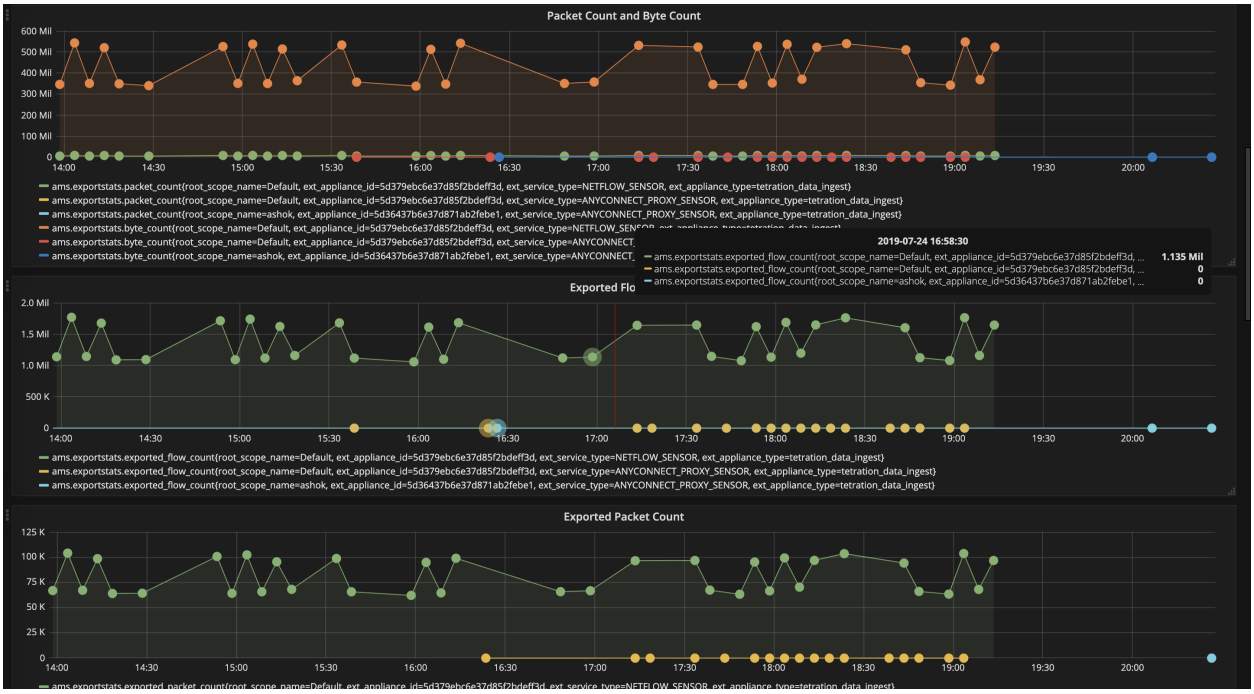


Fig. 18.5.2.2.1: Service dashboard

18.5.2.3 AnyConnect Service Dashboard

AnyConnect service dashboard provides information about AnyConnect specific service information. Metrics such as number of endpoints, number of inventories, number of users reported by AnyConnect connector to Tetration are available in this dashboard. In addition, this dashboard also provides information about IPFIX protocol processing and decoding. Metrics such as decoded count, decoded error count, flow count, packet count, and byte count are available in this dashboard.

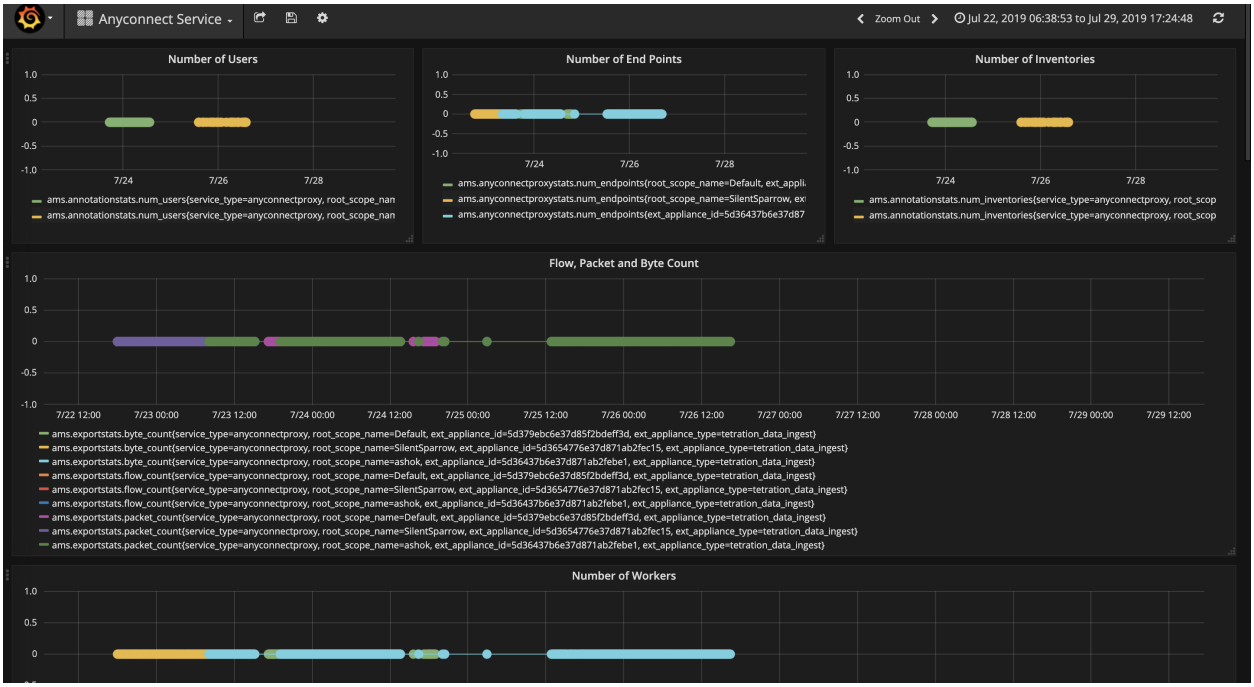


Fig. 18.5.2.3.1: AnyConnect dashboard

18.5.2.4 Appliance and Service DIO Dashboard

Appliance and service DIO dashboard provides information about number of messages exchanged in the Kafka topic on which the appliance manager and appliance/service controllers communicate. Metrics such as number of messages received, number of messages sent, number of messages failed are included in this dashboard. In addition, the last offset read by the controllers are also provided to understand whether the controller is lagging behind in processing the control messages from the manager.

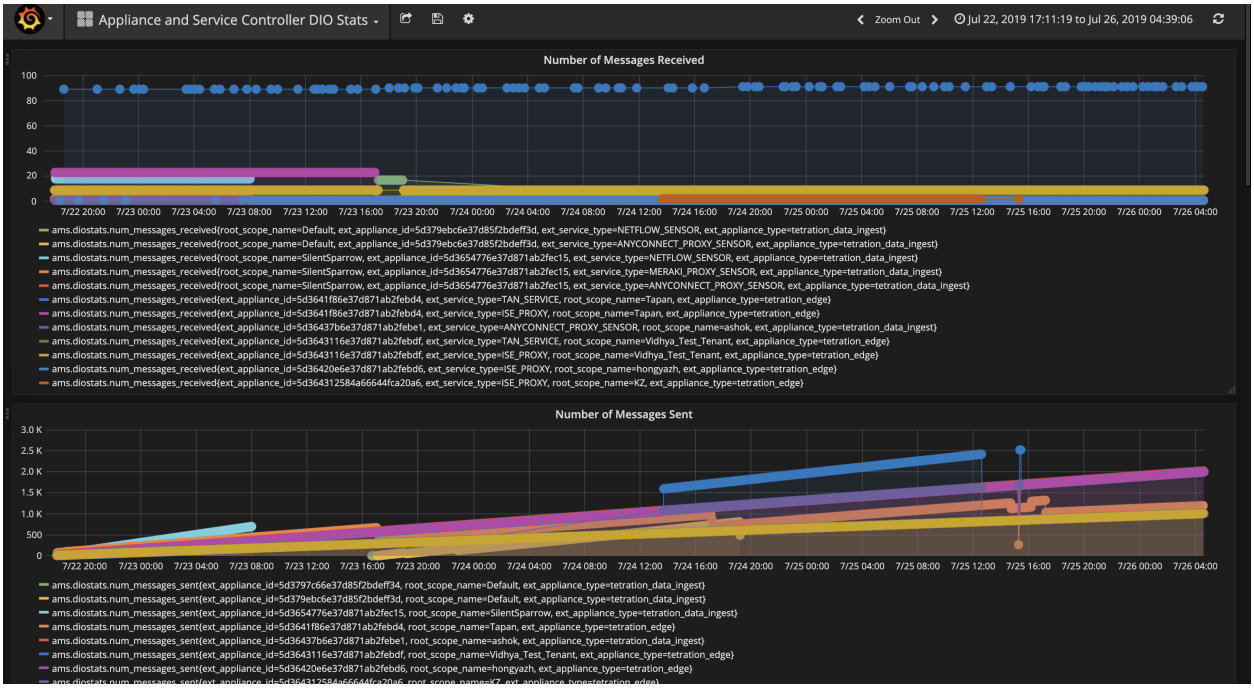


Fig. 18.5.2.4.1: Appliance and service DIO dashboard

18.5.3 General Troubleshooting Guidelines

Once a connector show in active state in connectors page in Tetration, no action is needed on the appliance where the connector is enabled; user does not need to log into it. If that is not happening, following information will help troubleshoot such problems.

In normal conditions, on the appliance:

- `systemctl status tet_vm_setup.service` reports an *inactive* service with *SUCCESS* exit satus.
- `systemctl status tet-nic-driver` reports an *active* service.
- `supervisorctl status tet-controller` reports *RUNNING* service. This indicates that the appliance controller is up and running.
- `docker network ls` reports 3 networks: `bridge`, `host`, and `none`.
- `docker ps` reports the containers that are running on the appliance. Typically, when a connector is enabled successfully on an appliance, a Docker container is instantiated on the appliance. For Syslog, Email, Slack, PagerDuty and Kinesis connectors, a Tetration alert notifier service is instantiated as a Docker container on Tetration edge appliance. And, for Tetration export connector, no special container is instantiated on the Tetration Export appliance. Tetration Export container comes up with `docker-compose` and instantiates Logstash, Elasticsearch, and Grafana stack by default.
- `docker logs <cid>` for each container should report that `tet-netflowsensor entered RUNNING state`.
- `docker exec <cid> ifconfig` reports only one interface, besides the loopback.
- `docker exec <cid> netstat -rn` reports the default gateway.
- `cat /local/tetration/appliance/appliance.conf` on the appliance to see the list of Docker services running on the appliance. It includes details about service ID, connector ID, container, image ID and

port mappings (if applicable). On a Tetration Ingest appliance, at most 3 services be running on the appliance. The port mappings and Docker volumes that are mounted on the containers are available in this file.

```
[root@esx-2106-ingest tetter]# systemctl status tet_vm_setup.service
● tet_vm_setup.service - Tetration Appliance Setup
   Loaded: loaded (/etc/systemd/system/tet_vm_setup.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Sat 2019-07-27 23:51:29 UTC; 21h ago
 Main PID: 1249 (code=exited, status=0/SUCCESS)

Jul 27 23:51:12 localhost.localdomain python[1249]: mount: /dev/sr0 is write-protected, mounting read-only
Jul 27 23:51:29 esx-2106-ingest python[1249]: Docker version 18.09.8, build 0dd43dd87f
Jul 27 23:51:29 esx-2106-ingest python[1249]: REPOSITORY          TAG                 IMAGE ID            CREATE... SIZE
Jul 27 23:51:29 esx-2106-ingest python[1249]: userPrivateKey.key
Jul 27 23:51:29 esx-2106-ingest python[1249]: intermediateCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: kafkaBrokerIps.txt
Jul 27 23:51:29 esx-2106-ingest python[1249]: userCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: kafkaCA.cert
Jul 27 23:51:29 esx-2106-ingest python[1249]: topic.txt
Jul 27 23:51:29 esx-2106-ingest python[1249]: Created symlink from /etc/systemd/system/multi-user.target.wants/s...vice.
Hint: Some lines were ellipsized, use -l to show in full.
[root@esx-2106-ingest tetter]#
```

Fig. 18.5.3.1: Tetration appliance deployment service and status

```
[root@esx-2106-ingest tetter]# systemctl status tet-nic-driver.service
● tet-nic-driver.service - NIC network driver plugin for Docker
   Loaded: loaded (/etc/systemd/system/tet-nic-driver.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2019-07-27 23:51:12 UTC; 21h ago
 Main PID: 733 (nic)
   Memory: 4.4M
   CGroup: /system.slice/tet-nic-driver.service
           └─733 /usr/local/tet/nic-driver/nic -log-level debug

Jul 27 23:51:12 localhost.localdomain systemd[1]: Started NIC network driver plugin for Docker.
Jul 27 23:51:12 localhost.localdomain systemd[1]: Starting NIC network driver plugin for Docker...
Jul 27 23:51:12 localhost.localdomain nic[733]: time="2019-07-27T23:51:12Z" level=info msg="NIC network driver started"
Hint: Some lines were ellipsized, use -l to show in full.
[root@esx-2106-ingest tetter]#
```

Fig. 18.5.3.2: Tetration network driver service status

```
[root@esx-2106-ingest tetter]# supervisorctl status tet-controller
tet-controller                   RUNNING   pid 1971, uptime 21:43:29
[root@esx-2106-ingest tetter]#
```

Fig. 18.5.3.3: Appliance controller status

If any of the above does not hold true, please check the deployment script logs in `/local/tetration/logs` for the reason why the appliance and/or the connector deployment failed.

Any other connector registration/connectivity issue can be troubleshooted as follows.

- `docker exec <cid> ps -ef` reports `tet-netflowsensor-engine`, `/usr/local/tet/tet-netflowsensor -config /usr/local/tet-netflow/conf/tet-netflow.conf` instances, along with the process manager `/usr/bin/supervisord -c /usr/local/tet-netflow/conf/supervisord.conf -n` instance.

```
[root@esx-2106-ingest tetter]# docker ps
CONTAINER ID        IMAGE                                                                                    COMMAND
     CREATED             STATUS              PORTS                           NAMES
c82decfaa877        asa_sensor-3.4.2.52465.appliance.demo.mrpm.build-asa:5d3ce5e43649723890271dd3   "/usr/bin/supervisor
…"   22 hours ago        Up 22 hours         172.29.142.27:4729->4729/udp   asa-5d3ce5e43649723890271dd3
eddd5cd59839        aws_sensor-3.4.2.52465.appliance.demo.mrpm.build-aws:5d3ce3b73649723890271dce   "/usr/bin/supervisor
…"   22 hours ago        Up 22 hours                                         aws-5d3ce3b73649723890271dce
[root@esx-2106-ingest tetter]# docker exec c8 ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 00:01 ?        00:00:15 /usr/bin/python /usr/bin/supervisord -c /usr/local/tet-netflow/conf/supe
rvisord.conf -n
root         8     1  0 00:01 ?        00:02:24 /usr/local/tet-netflow/tet-netflowsensor-engine -ctrl-config /usr/local/
tet-netflow/conf/tet-controller.conf -upgrade-script /usr/local/tet-netflow/scripts/check_config_update.sh -service /usr
/local/tet-netflow/tet-netflowsensor -config /usr/local/tet-netflow/conf/tet-netflow.conf
root     27002     8  0 21:31 ?        00:00:00 /usr/local/tet-netflow/tet-netflowsensor -config /usr/local/tet-netflow/
conf/tet-netflow.conf
root     27024     0  0 21:32 ?        00:00:00 ps -ef
[root@esx-2106-ingest tetter]#
```

Fig. 18.5.3.4: Running processes on ASA connector in Tetration Ingest appliance

18.5.3.1 Log Files

The following commands can be used to view the logs from various services on the appliance.

- `/local/tetration/logs/tet-controller.log` shows the logs of the appliance controller.
- `docker exec <cid> cat /local/tetration/logs/tet-controller.log` shows the logs of the service controller on the connector.
- `docker exec <cid> cat /local/tetration/logs/tet-netflow.log` shows the logs of the connector service.
- `docker exec <cid> cat /local/tetration/logs/tet-ldap-loader.log` shows the logs of LDAP snapshot creation (if LDAP config is applicable for the connector).
- `docker exec <cid> cat /local/tetration/logs/check_conf_update.log` shows the configuration update polling logs (for connectors on Tetration Ingest appliance).

Note: There are allowed set of commands on Tetration that can pull these logs from the appliance and/or connectors directly. Please see *Allowed set of commands* for more details.

Debug Mode

The default logging level for the appliance/service controller and connector service is set to *info* level. For troubleshooting issues, we may need to set the agent in *debug* mode. To do this, please update the log configuration on the appliance/connector on Tetration directly for the desired appliance/connector. The log levels for both the controller and services are updated if the configuration is updated on the connector. Please see *Log Configuration* for more details.

18.6 Connector Alerts

Connector alert would be created when an appliance/service has abnormal behavior.

18.6.1 Alert Configuration

Alert configuration for appliances and connectors allow users to enable alerts to be generated for various events. In 3.4 release, this configuration enables all types of alerts that are potentially possible for the configured appliance/connector.

Parameter Name	Type	Description
Enable Alert	checkbox	Should alert be enabled?

Note: The default value for *Enable Alert* is *true*.

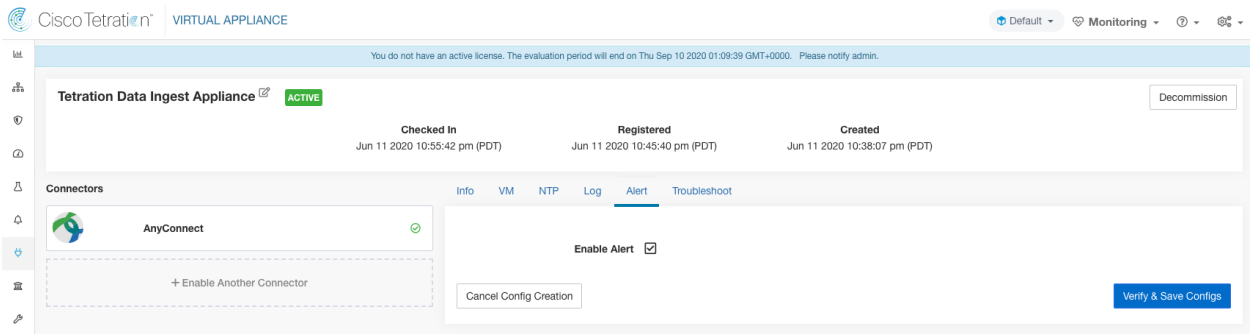


Fig. 18.6.1.1: Show alert configuration on a Tetration Data Ingest Appliance

18.6.2 Alert Type

Each appliance and connector would have different alert types. It could be found on Info Tab on the appliance and connector pages.

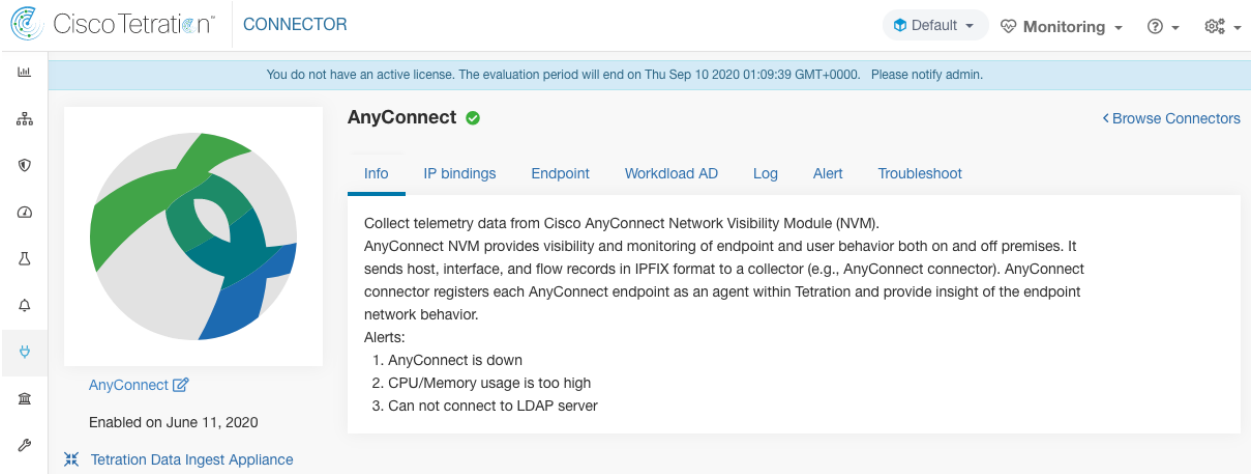


Fig. 18.6.2.1: Alert list info

18.6.2.1 Appliance/Connector down

This alert is generated when an appliance (or a connector) is potentially down due to missing heartbeats from the appliance/connector respectively at Tetration.

Alert text: Missing <Appliance/Connector> heartbeats, it might be down.

Severity: High

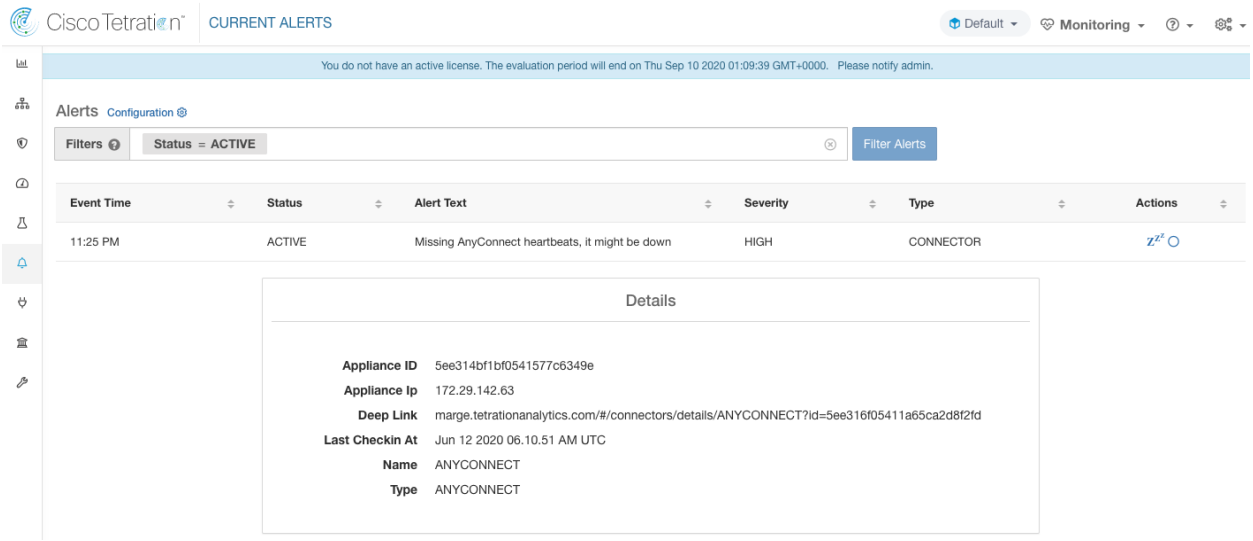


Fig. 18.6.2.1.1: Alert for connector down

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: All

18.6.2.2 Appliance/Connector system usage

When system usage (CPU, memory, and disk) is more than 90% on an appliance (and a connector), this informational alert is generated to indicate that the appliance (and/or connector) is currently handling an increased system load. It is normal for appliances and connectors to consume more than 90% of system resources during heavy processing activity.

Alert text: <Number> of CPU/Memory/Disk usage on <Appliance/Connector> is too high.

Severity: High

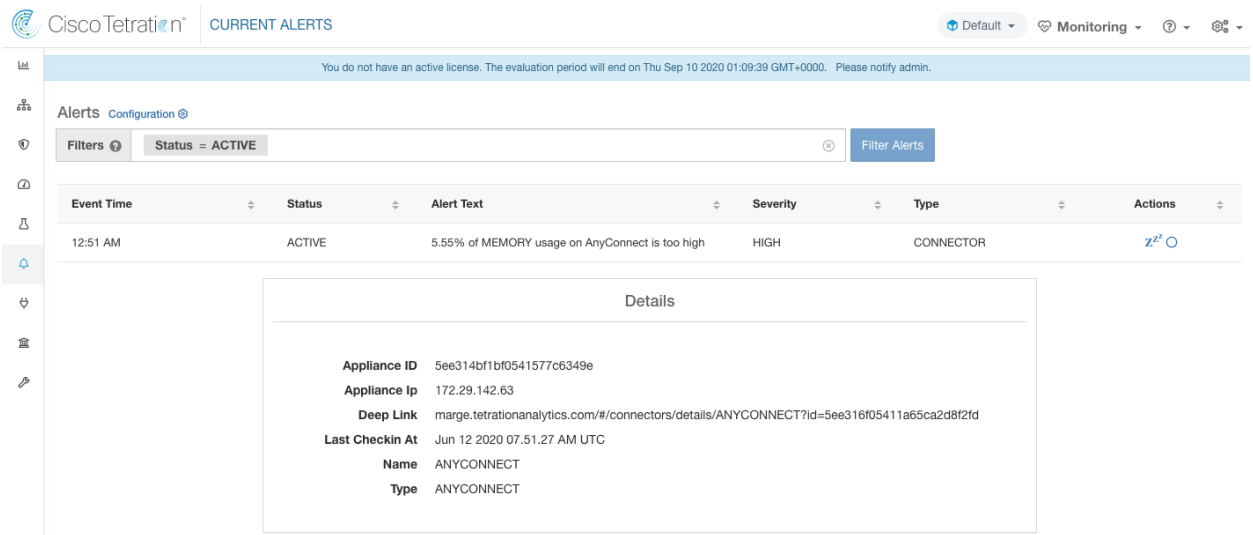


Fig. 18.6.2.2.1: Alert for connector system usage too high

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: All

18.6.2.3 Connector config error

When a configuration for a connector cannot connect to configured server, this alert is generated to indicate a potential issue with the configuration after it was accepted and deployed. For example, AnyConnect connector can take LDAP configuration, validate and accept the configuration. However, during the normal operation, it is possible that the configuration is no longer valid. This alert captures this scenario and indicates that the user has to take corrective action to update the configuration.

Alert text: Cannot connect to <Appliance/Connector> server, please check <Appliance/Connector> config.

Severity: High, Low (It is particular for AWS cannot find object in designated bucket)

Server	Connector
Ldap server	AnyConnect, F5, ISE, WDC
AWS server	AWS
ISE server	ISE
ServiceNow server	ServiceNow

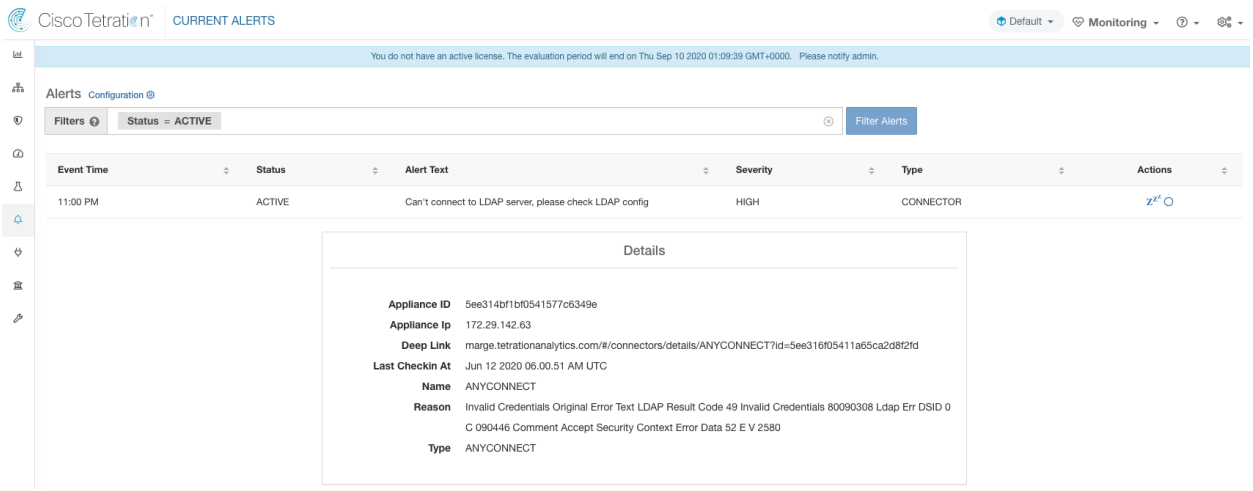


Fig. 18.6.2.3.1: Alert for config status error

Allowed Tetration virtual appliances: Tetration Ingest and Tetration Edge

Allowed connectors: AnyConnect, F5, AWS, ISE, WDC and ServiceNow

18.6.3 Connector UI Alert Details

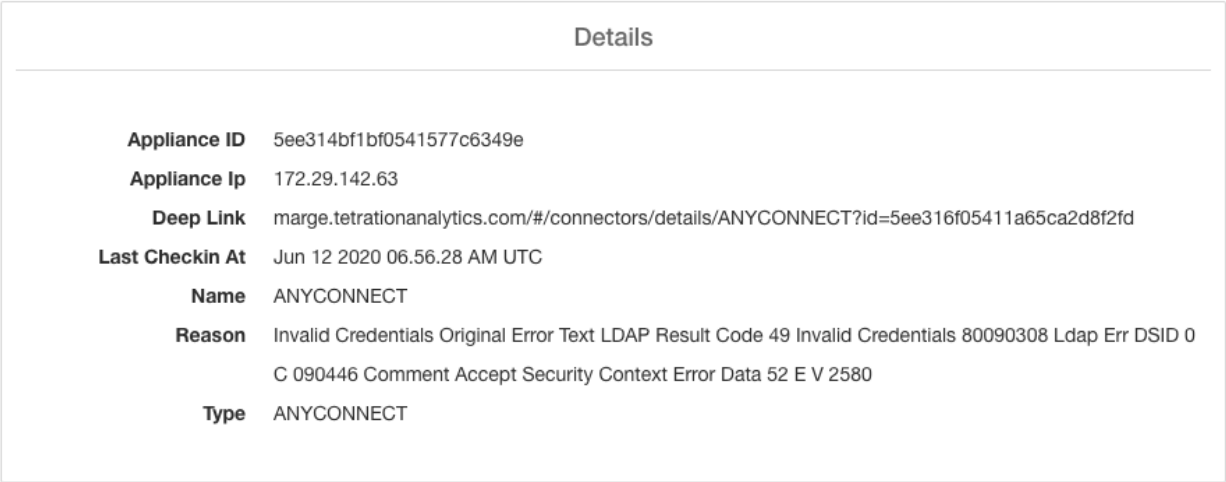


Fig. 18.6.3.1: Connector UI Alert details

18.6.4 Alert Details

See *Common Alert Structure* for general alert structure and information about fields. The *alert_details* fields are structured and will contain the following subfields for connector alerts

Field	Type	Description
Appliance ID	String	Appliance ID
Appliance IP	String	Appliance IP
Connector ID	String	Connector ID
Connector IP	String	Connector IP
Deep Link	Hyperlink	Redirect to appliance/connector page
Last CheckIn At	String	Last checkin time
Name	String	Appliance/Connector name
Reason	String	The reason that Appliance/Connector cannot connect to Tetration
Type	String	Appliance/Connector type

18.6.5 Example of Alert Details

After alert_details is parsed as json (unstringified), then it would look like following

```
{
  "Appliance ID": "5f1f3d26d674b01832c6792a",
  "Connector ID": "5f1f3e47baba512a70abee43",
  "Connector IP": "172.29.142.22",
  "Deep Link": "bingo.tetrationanalytics.com/#/connectors/details/F5?
↪id=5f1f3e47baba512a70abee43",
  "Last checkin at": "Aug 04 2020 20.37.33 PM UTC",
  "Name": "F5",
  "Reason": "Invalid Credentials (Original error text: LDAP Result Code 49 \"Invalid␣
↪Credentials\": )",
  "Type": "F5"
}
```

CHAPTER

NINETEEN

VIRTUAL APPLIANCES

Cisco Tetration Virtual Appliances.

19.1 Cisco Tetration ERSPAN Virtual Appliance

The Cisco Tetration ERSPAN is a software appliance for remotely monitoring hosts' traffic via dedicated Cisco Tetration SPAN agents. Using this solution, the hosts do not need to run software agents, because the Cisco switches will relay the hosts' traffic to the ERSPAN appliance for processing.

19.1.1 What is ERSPAN

Encapsulated Remote Switch Port Analyzer (ERSPAN) is a feature present in most of Cisco switches. It mirrors frames seen by a network device, encapsulates them in a IP packet and sends them to a remote analyzer. Users can select a list of interfaces and/or VLANS on the switch to be monitored.

Commonly, the setup involves configuring source ERSPAN monitoring session(s) on one or more network devices and configuring the destination ERSPAN monitoring session(s) on the remote network device(s) directly connected to a traffic analyzer.

The Tetration ERSPAN VM Appliance provides both the destination ERSPAN session and traffic analyzer functionalities; therefore there is no need to configure any destination sessions on the switches with the Tetration solution.

19.1.2 What are the SPAN Agents

The Tetration SPAN agents are regular Tetration agents configured to only process ERSPAN packets: Like Cisco destination ERSPAN sessions, they decapsulate the mirrored frames; then they process and report the flows like a regular Tetration agent. Unlike Deep Visiblity Agents, they do not report any process or interface information.

They can be downloaded from the Software Agent Download Page:

Version	Agent Type	Platform	Created At	Actions
2.1.1.16.devel-1-span ALPHA	SPAN	CentOS-7.0	3:17 PM	Download
2.1.1.16.devel-1-span ALPHA	SPAN	CentOS-7.1	3:17 PM	Download
2.1.1.16.devel-1-span ALPHA	SPAN	CentOS-7.2	3:17 PM	Download
2.1.1.16.devel-1-span ALPHA	SPAN	CentOS-7.3	3:17 PM	Download

Fig. 19.1.2.1: Software Agent Download Page

19.1.3 What is the ERSPAN Virtual Appliance

The Cisco Tetration ERSPAN Virtual Appliance is a Virtual Machine that internally runs three SPAN Tetration agents.

Each agent runs inside a dedicated Docker container to which one vNIC and two vCPU cores with no limiting quota are exclusively assigned.

The SPAN Agents register with the cluster with the container hostname: <VM hostname>-<interface IP address>.

The agents are preserved/restored upon VM, Docker daemon or Docker container crash/reboot.

19.1.4 How to deploy the appliance

1. Download the Tetration ERSPAN Virtual Machine Appliance OVA file from the Cisco CCO page.
2. Create a Virtual Machine from the OVA file and provision it with eigth vCPU cores, four gigabytes of RAM and three virtual interfaces in bridged networking mode.
3. Download the CentOS-7.3 **-span** Agent image bundle into a directory **<cfg dir>**.
4. Add a **<cfg dir>/ip_config** file containing the IP and gateway address for each interface. It must contain three rows in the format: <CIDR> <gateway IP> (Example: 172.33.9.8/24 172.33.9.1). All interfaces must belong to the same subnet. Optionally, add a **<cfg dir>/host_name** file containing the hostname string. Default is **erspan-vm**.
5. Optionally, add a **<cfg dir>/resolv.conf** file containing the nameserver address and search domain. You would need this in case your cluster specifies the agent configuration entry **config_server_url** as a FQDN instead of an IP address. This is the case when deploying Tetration SaaS solution.
6. Optionally, add a **<cfg_dir>/user.cfg** file containing the proxy server settings relevant to the network where the VM is deployed. We accept only HTTPS_PROXY setting and that should point to a http proxy server in the network where the VM is deployed. In addition, this file should contain value for ACTIVATION_KEY if the sensor is deployed against a TaaS cluster. The value for this key can be fetched from TaaS UI in Software Agent Download page. See Example user.cfg for reference.
7. Create a ISO file via: **mkisofs -r -o <name>.iso <cfg dir>** and add it to the VM as CDROM/DVD disc image.
8. Boot the VM.

Note: **ip_config** file should not contain any blank lines including at the end of the file.

Note: **Configuration ISO disk creation is now fully automated**. User has to enable the ERSPAN under the Connectors page on cluster UI. It will then be guided through the ISO creation steps. User will be able to download the ISO containing the configurations and the CentOS 7.3 span Agent image bundle. ISO will then need to be attached to the VM as CDROM/DVD disc image. This process automates and replaces the manual steps 3 through 7 from the list above.

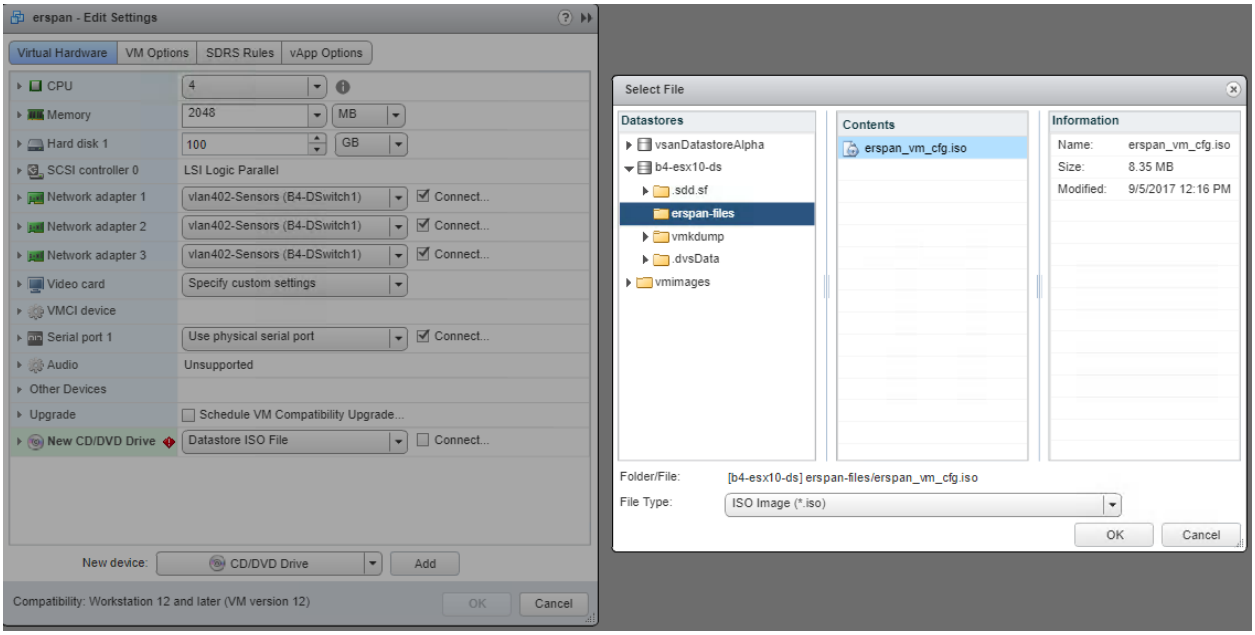


Fig. 19.1.4.1: VM provisioning and config ISO mounting

An example of the config ISO preparation is presented below:

```
$ ll vm-cfg
total 8208
-rw-r--r-- 1 user user      12 Sep  7 05:10 host_name
-rw-r--r-- 1 user user      93 Sep  5 20:53 ip_config
-rw-r--r-- 1 user user 8393441 Sep  5 18:22 tet-sensor-2.1.1.15-1.el7-rosen.span.x86_64.rpm
$
$ cat vm-cfg/ip_config
172.28.126.236/24 172.28.126.1
172.28.126.237/24 172.28.126.1
172.28.126.242/24 172.28.126.1
$
$ cat vm-cfg/host_name
erspan-vm-1
$
$ mkisofs -r -o erspan_vm_cfg.iso vm-cfg
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 450
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
4276 extents written (8 MB)
$
$ ll erspan_vm_cfg.iso
-rw-r--r-- 1 user user 8757248 Sep  7 05:11 erspan_vm_cfg.iso
$
```

Fig. 19.1.4.2: Preparing the configuration ISO for the Virtual Machine

19.1.5 How to configure the source ERSPAN session

The following steps are for a Nexus 9000 switch. The configurations may slightly differ for other Cisco platforms. In any case, please also refer to the official Cisco configuration guide for the Cisco platform you are configuring.

```
Enter the configuration mode
# config terminal

Configure the erspan source IP address
(config)# monitor erspan origin ip-address 172.28.126.1 global

Create and configure the source erspan session
(config)# monitor session 10 type erspan-source
(config-erspan-src)# source interface ethernet 1/23 both
(config-erspan-src)# source vlan 315, 512
(config-erspan-src)# destination ip 172.28.126.194

Turn on the monitor session
(config-erspan-src)# no shut

Persist the configuration
# copy runnin-config startup-config
```

Fig. 19.1.5.1: Configuring ERSPAN source on Cisco Nexus 9000

The above steps created a source ERSPAN session with id 10. The switch will mirror the frames ingressing and egressing (both) the interface eth1/23 and the ones on VLANS 315 and 512. The outer GRE packet carrying the mirrored frame will have source IP 172.28.126.1 (must be the address of a L3 interface on this switch) and destination IP 172.28.126.194. This is one of the IP addresses configured on the ERSPAN VM.

19.1.6 Supported ERSPAN formats

The Tetration SPAN Agents can process ERSPAN type I, II and III packets described in the proposed ERSPAN RFC. Therefore they can process ERSPAN packets generated by Cisco devices. Among the non RFC compliant formats, they can process the ERSPAN packets generated by VMWare vSphere Distributed Switch (VDS).

19.1.7 Performance considerations when configuring ERSPAN source

Carefully choose the ERSPAN source's port/VLAN list. Although the SPAN agent has two dedicated vCPUs, the session may generate cosiderable amount of packets which could saturate the processing power of the agent. If an agent is receiving more packets than it can process, it will be shown in the Agent Packet Misses graph on the cluster's Deep Visibility Agent page.

More fine grained tuning on which frames the ERSPAN source will mirror can be achieved with ACL policies, usually via the `filter` configuration keyword.

If the switch supports it, the ERSPAN source session can be configured to modify the maximum transport unit (MTU) of the ERSPAN packet (commonly the default value 1500 bytes), usually via a `mtu` keyword. Deacreasing it will limit the ERSPAN bandwidth usage in your network infrastructure, but it will have no effect on the SPAN Agent load, given the agent's workload is on a per-packet basis. When reducing this value, please allow room for 160 bytes for the mirrored frame. Please refer to the proposed ERSPAN RFC for the ERSPAN header overhead details.

There are three versions of ERSPAN. The smaller the version, the lower the ERSPAN header overhead. Version II and III allow for applying QOS policies to the ERSPAN packets, and provide some VLAN info. Version III carries even more settings. Version II is usually the default one on Cisco switches. While Tetration SPAN Agents support all three versions, at the moment they do not make use of any extra information the ERSPAN version II and III packets carry.

19.1.8 Security considerations

The ERSPAN Virtual Machine guest Operating System is CentOS 7.3, from which OpenSSL server/clients packages were removed.

Once the VM is booted and the SPAN agent containers are deployed (this takes a couple of minutes on first time boot only), no network interfaces, besides the loopback, will be present in the Virtual Machine. Therefore the only way to access the appliance is via its console.

The VM network interface are now moved inside the Docker containers. The containers run a Centos:7 based Docker image with no TCP/UDP port open.

Also, the containers are run with the base privileges (no –privileged option) plus the NET_ADMIN capability.

In the unlikely case a container is compromised, the VM guest OS should not be compromisable from iside the container.

All the other security consideration valid for Tetration Agents running inside a host do also apply to the Tetration SPAN Agents running inside the Docker containers.

19.1.9 Troubleshooting

Once SPAN Agents show in active state in the cluster Monitoring/Agent Overview page, no action is needed on the ERSPAN Virtual Machine, user does not need to log into it. If that is not happening or if the flows are not reported to the cluster, following information will help pinpoint deployment problems.

In normal conditions, on the VM:

- the directory `/mnt/sensor-rpm/` contains `tet-sensor-<...>.span-x86_64.rpm` and the `ip_config` files;
- `systemctl status tet-span-sensors` reports an *inactive* service with *SUCCESS* exit satus;
- `systemctl status tet-nic-driver` reports an *active* service;
- `docker network ls` reports five networks: `host`, `none` and three `erspan-<iface name>`;
- `ip link` only reports the loopback interface;
- `docker ps` reports three running containers;
- `docker logs <cid>` for each container contains the message: `INFO success: tet-sensor entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)`
- `docker exec <cid> ifconfig` reports only one interface, besides the loopback;
- `docker exec <cid> route -n` reports the default gateway;
- `docker exec <cid> iptables -t raw -S PREROUTING` reports the rule `-A PREROUTING -p gre -j DROP`;

If any of the above does not hold true, please check the deployment script logs in `/usr/loca/tet/log/sensor_container_setup.log` for the reason why the SPAN agent containers deployment failed.

Any other agent registration/connectivity issue can be troubleshooted the same way it is done for agents running on a host via the *docker exec* command:

- `docker exec <cid> ps -ef` reports the two `tet-engine`, `tet-engine check_conf` instances and two `/usr/local/tet/tet-sensor -f /usr/local/tet/conf/.sensor_config` instances, one with `root` user and one with `tet-sensor` user, along with the process manager `/usr/bin/python /usr/bin/supervisord -c /etc/supervisord.conf -n` instance.
- `docker exec <cid> cat /usr/local/tet/log/tet-sensor.log` shows the agent's logs;
- `docker exec <cid> cat /usr/local/tet/log/fetch_sensor_id.log` shows the agent's registration logs;
- `docker exec <cid> cat /usr/local/tet/log/check_conf_update.log` shows the configuration update polling logs;

If necessary, traffic to/from the container can be monitored with tcpdump after setting into the container's network namespace:

1. Retrieve the container's network namespace (SanboxKey) via `docker inspect <cid> | grep SanboxKey`;
2. Set into the container's network namespace `nsenter --net=/var/run/docker/netns/...`;
3. Monitor `eth0` traffic `tcpdump -i eth0 -n`.

19.2 Performance numbers

Performance numbers for Cisco Tetration Virtual Appliances.

19.2.1 Performance numbers for ERSPAN Virtual Appliance

19.2.1.1 Summary

This section provides throughput numbers for the ERSPAN Virtual Appliance. It contains values actually measured in our lab and approximate extrapolated values for other packet sizes. For the valuation of the estimated entries, CPU, flow table size, and bandwidth limitations were taken into account.

The captured traffic was not under our control. Most of the traffic pertained to physical and virtual clusters forming our continuos integration, demo and production infrastructure which comprises of thousand of Virtual Machines.

The CPU usage per SPAN sensor was in the [150%-170%] range. CPU fluctuates especially during the flow export phase when the flow records are being scanned from the table, serialized in binary format then queued to be sent to collectors.

In general, the effective throughput numbers may vary by a great degree based on the type of traffic being monitored and by whether or not the ERSPAN source session is configured to truncate the ERSPAN packets. If the user configures the source to only send the minimum amount of bytes necessary to the SPAN sensor, which is 160+ERSPAN expansion, this will change the throughput numbers considerably. The truncation at source, though, has no effect on the number of packets that can be processed by the SPAN sensors: The Tetration sensors workload is per-packet based.

Looking at the average packet size measured in our lab, which is in the 270-300 bytes range, it is likely that a good amount of the captured packets are control packets (ICMP, ARP, keep-alive msgs...), small packets which bring the average packet size down.

In conclusion, assuming the number of flows is a constant of the number of received packets, the true ERSPAN Virtual Appliance performance metric is the number of incoming packets per second that can be processed within the CPU and flow table limits. From the measurement, this value is around **1.6 Mpps** (or an estimated **2.1Mpps** with full CPU usage) starting from 2.2.1.x release.

19.2.1.2 Environment

ERSPAN Source:

- Version II (it adds 50 bytes to the original packet size)
- No packet truncation
- Cisco Nexus9000 switch

Hypervisor:

- VMware ESXi, 6.5.0
- Model TA-BNODE-G1
- Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz

ERSPAN Virtual Appliance:

- 8 vCPU cores @ 2.30GHz
- 4GB RAM
- Three 10 Gbps vNICs, VMXNET driver
- It can process 1.6 million ERSPAN packets per second (estimated 2.1 million with full CPU usage)

19.2.1.3 Scale Limits

Packet size [bytes]	**Processed Packets** [pps]	**Throughput** [bps]	**Limit**
270	1.6M	3.5G	CPU
626	2.1M*	10.5G*	CPU/Flow table
9000	416K*	30G*	Bandwidth

Measured and estimated (*) throughput for ERSPAN Virtual Appliance

CHAPTER

TWENTY

TETRATION-V

20.1 What is Tetration-V

Tetration-V is a software solution for deploying Tetration on top of VMware vSphere Virtualized Infrastructure. It is suited for small scale deployments or environments where virtualization is the only available compute option.

20.2 Preparation

To make sure the installation process goes as smoothly as possible, it is important to first prepare all of the necessary information and infrastructure.

Note: Despite hardware recommendations being satisfied, no performance guarantee or SLA can be made since other factors may impact the overall performance of the Tetration Software. To ensure the highest performance, it is recommended to use dedicated hardware for the cluster.

The following hardware and software dependencies must be available to proceed with installation:

20.2.1 Software

- VMware cluster running vSphere 6.5 or 6.7
  - This includes all components, such as the hypervisors, core management, authentication and services, and upgrade and patch management
  - Must be running in a VMware supported configuration
- Tetration-V Orchestrator Appliance Open Virtualization Archive (OVA) file
- Required Tetration RPMs

20.2.2 Hardware

- Infrastructure capable of hosting:
  - 128 Central Processing Unit (CPU) cores
  - 2TB Random Access Memory (RAM)
    - Virtual Machines (VM) will be as large as 128GB
  - 18.1TB Storage

    * 5,000 Input/Output Operations Per Second (IOPS) capable
    * Storage must be accessible from all nodes in cluster
    * Must be in a single datastore
    * Storage must be durable
  - A resource check is performed at deployment, however the infrastructure must be managed to stay within requirements.
- Network infrastructure
  - All hosts in the cluster should be connected with at least 10G interfaces
  - All hosts must have three virtual networks available for Tetration purposes
    * Public Network: Dedicated or shared public network for external cluster traffic. Must be reachable from sensors and clients, and also have access to the vCenter
    * Private Network: Dedicated private network for internal cluster communication. Must not be routable
    * Configuration Network: Temporary network for bootstrapping cluster. Must be reachable from user performing deployment, and also have access to the vCenter. This should be a separate subnet from the Public subnet, and should be shut down after the deployment has completed
  - If a Distributed Virtual Switch is used, be certain that it includes all hosts in the cluster

20.2.3 Network Connectivity / Firewall Requirements during deployment

During deployment, the bootstrap orchestrator will have two public facing IP addresses assigned:

- Configuration Network
- Public Network

Orchestrator will automatically assign the last address in the Public network, and use that for performing Site Checker validation. This includes (but is not limited to): SMTP, DNS, NTP, Ping, vCenter connection tests. After Site Checker validation has completed, orchestrator will remove the IP address, and continue the rest of deployment using the Configuration network. The following connectivity will be established:

Network	Destination	Protocol	Port
Config & Public	DNS Server	UDP	53
Config & Public	NTP Server	UDP	123
Public	SMTP Server	TCP	SMTP Port
Config & Public	vCenter Host	TCP	443
Config	ESX Hosts[1]	TCP	443

- 1: OVF tool establishes connectivity with vCenter, which redirects to ESX for large file transfers

After the deployment has completed, the Configuration network will remove its IP address and shut down the interface. Upgrades will be done entirely using the Public network.

20.2.4 VMware Configuration

Note: Each vSphere deployment may have different requirements, constraints and best practices that the Administrator has implemented. No configuration changes are made by the Tetration Installer. The below recommendations should not replace the advice and careful planning of a VMware expert.

Note: If the following recommendations are not followed, availability of data may suffer and the product may not function completely or as intended.

- Datastores must be highly available and durable, meaning that data is stored redundantly and resistent to hardware failure.
- The hypervisor hosts and vCenter server must have their clocks set correctly and synchronized using Network Time Protocol (NTP).
- It is also recommended to configure anti affinity rules for certain roles that provide redundancy within the Tetration infrastructure. Anti affinity rules be put in place for instances of the following base types:
  - orchestrator
  - adhoc
  - appServer
  - collectorDatamover
  - datanode
  - druidCoordinator
  - druidHistoricalBroker
  - elasticsearch
  - enforcementCoordinator
  - enforcementPolicyStore
  - happobat
  - hbaseMaster
  - hbaseRegionServer
  - launcherHost
  - mongodb
  - namenode & secondaryNamenode
  - redis
  - tsdbBosunGrafana
  - zookeeper

20.2.5 VMware Permissions

The Tetration Installer requires credentials to access vSphere and create Virtual Machines, Folders, Files in the Datastore, and connect to Virtual Switches and Datastores.

It is strongly recommended to use a separate user account for the Tetration Installer, which has bare minimum permissions restricted to only the scope and privileges necessary to perform installation.

The following permissions can be used as a starting point for creating the VMware user account role:

- Content Library
- Datastore
- Folder\Create Folder

- Network\Assign network
- Resource
- Tasks
- Virtual Machine
- dvPort Group
- vApp
- vSphere Labeling

The open-source utility Terraform is used for managing some resources in vSphere. For more information on required permissions from Terraform, please refer to https://www.terraform.io/docs/providers/vsphere/index.html#notes-on-required-privileges

20.2.6 Limitations

- Snapshots of VMs are not supported

20.3 Site Info

The following items of site info will be required in addition to the standard deployment details.

20.3.1 Network Tab

- External Network

  Please make sure the External Network has at least 8 free Internet Protocol (IP) addresses present, and does not include the first three IP addresses of the subnet

20.3.2 ESX Tab

- vSphere Host

  IP address or hostname of vSphere Server

- vSphere Username

  Username for vSphere account which has necessary roles to upload files and create virtual machines

- vSphere Password

  Password for vSphere account

- vSphere Datacenter

  Name of target vSphere Datacenter. Spaces are allowed

- Cluster

  Cluster into which virtual machines will be placed. Spaces are allowed

- VM Folder Name

  Name of folder which Tetration Virtual Machines will be placed into. Note: nested folders are not supported

- Datastore

  Datastore into which attached storage will be placed

- Private Network Port Group

  Name of virtual switch port group to use for private networking

- Public Network

  Name of virtual switch port group to use for public networking

- Cloud Init Folder

  Folder name on datastore which may be used for storing deployment configuration files

20.3.3 Advanced Tab

- External IPs

  If a shared subnet will be used for the public network, the list of IP addresses that may be used by the deployment should be specified here.

  - 8 IP addresses are required
  - The first three IP addresses in a subnet cannot be specified
  - If no addresses are entered, automatic assignment will be used

20.4 Automatic IP Address Assignment

During initial configuration the orchestrator must self-assign IP addresses on the public and private networks. The IP address on the public network will be used for communication with vCenter, and other external services (DNS, NTP, SMTP).

Automatic IP Address Assignment operates by the following rules:

- For Public Network IP assignment, a list of available IPs will be calculated, first using the user specified "External IPs". If none are provided, the IP addresses from the "External Network" subnet will be used.

  Note: If a list of "External IPs" is provided, the order of that list will be maintained during the linear assignment process.

  Note: The "External IPs" may not contain the first three IP addresses of the subnet, and if they are specified, they will not be consumed.

- If no "External IPs" are specified, the available IPs list will be populated with all IP addresses from the "External Network" except for the first 3.
- For Private Network IP assignment, the first 3 IP addresses will be skipped and the last 3 IP addresses will be automatically assigned to Orchestrator IP addresses.
- Orchestrator Public IP will use the last IP in the available IP in the list of available IPs

Warning: Carefully plan any infrastructure IP addresses that are assigned outside of Tetration with an understanding of the Automatic IP Address Assignment rules. Please avoid placing any infrastructure IP addresses on ranges that will be used by Tetration.

20.5 Deployment

20.5.1 Orchestrator OVA Deployment

1. Log into the VMware User Interface (Flash Player Web Interface is recommended for vSphere 6.5)
2. Create a new Folder with the intended site name of the cluster
3. Right click on the target cluster and click "Deploy OVF Template"
4. Enter the location of the OVF Template. Click Next

   Note: It is recommended to host the orchestrator OVA on a webserver in close proximity to the hypervisor cluster, since the orchestrator ova is over 5GB and may take a long time to transfer on a slow link

5. Enter "orchestrator-1" for the VM name and make sure it is deploying in the intended data center, and in the Tetration deployment folder named with the cluster site name. Click Next
6. Confirm the selected cluster is the intended target. Click Next
7. Review the licensing agreement, and if you agree to the terms click Agree. Click Next
8. Leave the default configuration profile selected (2CPU-8GB). Click Next
9. Select the appropriate datastore that should be used for the deployment. All other options can be left at their default settings, unless the environment requires other settings. Click Next
10. Select the appropriate network mapping and Click Next
    - Configuration: Routable network where the orchestrator can be reached during the deployment phase of cluster bring up. This should be different from the Public network, and should be disconnected from orchestrator-1 after deployment has complete
    - Private: Non-routed internal network that Tetration will use for internal communication
    - Public: Routable network where UI, Collectors and VIPs will be reachable
11. Enter the orchestrator reachability details for the Configuration network. Click Next.
    - IP address: Enter the dotted quad notation of the IP address for the orchestrator
    - Netmask: Enter the dotted quad notation of the netmask for the network
    - Gateway: Enter the dotted quad notation gateway IP for the orchestrator on the configuration network
12. Confirm all of your configuration parameters, and click Finish

20.5.2 Tetration Setup

1. **After a few minutes the Open Virtualization Format (OVF) file will be deployed. After the OVA upload is complete, it may** "Refresh" the vSphere UI session to be able to power on and access the orchestrator VM. Click the "Refresh" button next to the logged in user name at the top right of the vSphere console

2. Power on the "orchestrator-1" VM
3. **Within a few minutes the IP address entered in step 10 should begin replying to ping requests. Once it is** up, navigate your web browser to http://orchestrator-ip:9000/
4. Upload the required RPMs/packages
5. **Enter the site info following the usual installation procedure, and reference the *Site Info* section** for hypervisor specific guidelines. For more details on site info, please see ./settings/maintenance/upgrade
6. Click Continue and follow the usual site installation steps
7. **After the deployment starts, you will begin to see virtual machines created in vSphere, first orchestrator-2** and orchestrator-3, then the rest of the Tetration stack. If you don't see any VMs created after 15 minutes, please check the deployment logs available in **Tetration Setup** under the "Details" button
8. **Monitor the Tetration Setup process, which on hardware matching the recommended specifications will** typically take approximately 1.5 hours to complete. Once deployment has reached 100%, make note of the VIP address that is shown in the status line. If you accidentally close the installer, note the IP address for the VIP, will be the first available IP address that the installer was provided.
9. **Open a web page in your browser pointing to the UI FQDN entered in the *Site Info*, and click the** "Forgot Password?" link. Enter the email address that was entered for the Site Administrator, and click "Send password reset link". Check your inbox (make sure you check your Spam folder) for the email, and follow the instructions included
10. In the VMware VM configuration screen, select "orchestrator-1", edit the Hardware and select "Network Adapter 3" and uncheck the "Connected" box. Click "OK" to apply the changes. Failure to follow this step may leave the cluster exposed to configuration after the installation process has completed.

Installation is complete

20.6 ESX Licensing

Clusters on ESX platform are restricted to a 30-day trial license when deployed. After 30 days the cluster will stop processing new data, however the user interface & data collected/processed when the cluster was active will still be accessible. If a cluster will be used for more than 30 days, it is strongly recommended to immediately request and apply a license extension, to avoid any interruption in data gathering.

If a cluster needs to be extended, send an email to tetration-esx-extensions@external.cisco.com with the following information:

1. Cluster Name
2. Cluster UUID
3. Customer Name
4. Cisco Account Manager

Cluster Name and Cluster UUID can be collected from the Company page in the Cluster Dashboard (Click on the Gear Icon on the top right and the company). Cluster UUID will be called cluster_uuid and Cluster Name will be called site_name:

cluster_state	Enabled till 2018-10-20 23:30:41.090000 UTC
cluster_uuid	fcf7519e-7c72-428d-eec9-e75915cb13c1
Sentinel Alert Email	esx-1006-support_cs+bosun@tetrationanalytics.com
site_cluster_type	ESX
DNS Domain	cisco.com
DNS Resolver	171.70.168.183 173.36.131.10
Strong SSL Ciphers for Agent Connections	False
External IPs	172.29.136.74 172.29.136.75 172.29.136.76 172.29.136.77 172.29.136.78 172.29.136.79 172.29.136.80 172.29.136.81
Internal Network	1.1.1.0/24
Site Name	esx-1006

Fig. 20.6.1: ESX Licensing

Once the purchase is validated a signed script will be sent that will work only for this cluster. Run this signed script in the Explore Page under Maintenance. This will extend the license for the cluster. If the cluster is already disabled, this will enable the cluster and extend the validity of the license.

For other platforms, licensing is disabled and there is no action required.

In the same company page, refer to cluster_state field to get the current state and until when the cluster's license is valid. If the cluster is in disabled state, it will display when the cluster license expired.

CHAPTER

TWENTYONE

LIMITS

21.1 Flows and Endpoints

Metric	Limit	ESXi/8RU/39RU/TaaS/-
Number of concurrent servers (virtual machine or bare metal) from which telemetry data can be analyzed by Tetration	up to 1000	ESXi
	up to 5000	8RU
	up to 25000	39RU
Number of flow events that can be processed by Tetration per second	up to 70000 per second	ESXi
	up to 500000 per second	8RU
	up to 2 million per second	39RU

21.2 Tenants, Child Scopes, Inventory Filters, and Roles

Metric	Limit	ESXi/8RU/39RU
Number of Tenants	7	ESXi
	7	8RU
	35	39RU
Number of Child Scopes per Tenant	140	ESXi
	999	8RU
	999	39RU
Number of Inventory Filters per Tenant	140	ESXi
	999	8RU
	999	39RU
Number of Roles per Child Scope	6	ESXi
	6	8RU
	6	39RU

21.3 Connectors[8]

[8] Please refer to *What are Connectors* for limits applicable to individual connectors.

Connector	Metric	Limit
AnyConnect Connector	Total number of AnyConnect endpoints supported by one AnyConnect connector	5000 endpoints[1]
AnyConnect Connector	Number of LDAP attributes that could be labelled on inventories of AnyConnect endpoints	6 attributes
AWS Connector	Total number of flows exported by AWS connector	15000 flows per second
F5 Connector	Total number of flows exported by F5 connector	15000 flows per second
NetFlow Connector	Total number of flows exported by one NetFlow connector	15000 flows per second
NetScaler Connector	Total number of flows exported by NetScaler connector	15000 flows per second

21.4 Tetration Virtual Appliances for Connectors

Appliance	Metric	Limit
Tetration Ingest Appliance	Number of connectors on one appliance	3
	Number of appliances per root scope	100
	Number of appliances per cluster	500
Tetration Edge Appliance	Number of connectors on one appliance	6
	Number of appliances per root scope	1
	Number of appliances per cluster	Number of root scopes
Tetration Export Appliance	Number of connectors on one appliance	1
	Number of appliances per root scope	1
	Number of appliances per cluster	Number of root scopes

21.5 Features

[1] The number of AnyConnect endpoints across all AnyConnect Proxy sensors is limited by the number of sensors supported by the Tetration appliance.

Feature	Metric	Limit	ESXi/8RU/39RU/TaaS/-
ADM	Maximum number of member endpoints allowed for ADM run	5000	-
	Maximum number of conversations allowed for ADM run	10,000,000	-
	Maximum number of member endpoints allowed for ADM run with deep policy generation option selected	25000	-
	Maximum number of conversations allowed for ADM run with deep policy generation option selected	20,000,000	-
	Maximum number of total unique endpoints allowed for ADM run	15,000,000	-
Alerts	Number of instances supported within a root scope	256	-
	Number of instances supported across root scopes	1024	-
	Number of latest alerts that are displayed on UI per root scope	5000	-
	Maximum alert rate to preview in UI	60 per minute[2]	-
	Number of alerts configured per root scope (via modal)	1000	-
	Maximum number of alerts processed by Alerts App per minute batch	20000	-
Compliance App	Number of application workspaces supported	128	-

[2] If more than 60 alerts are sent per minute then UI will show a summary message indicating that alerts were sent to the DataTap but are suppressed in UI. Note that the 60 alerts per minute applies to the rate at which alerts are sent to datataps, and does not apply to the alert time nor event time and is unrelated to any specific batch of data.

Feature	Metric	Limit	ESXi/8RU/39RU/TaaS/-
Datasink Dumper App	Number of instances supported	10	-
Lookout Annotation	Number of instances supported	256	-
	Number of root scopes on which Lookout Annotation can be enabled	256	-
	Number of Tetration tags limit	100000[7]	-
Neighborhood App	Number of root scopes on which Neighborhood app can be enabled	256	-
	Maximum number of alert configurations per type per root scope[6]	30	-
	Maximum number of live analysis filters and clusters per scope	500	

[7] Subnet limits defined under User Uploaded Annotations will also jointly apply.

[6] Please make sure that the number of alert configurations that you have currently for each type under Neighborhood app per root scope is within 30.

Feature	Metric	Limit	ESXi/8RU/39RU/TaaS/-
User Apps[3]	Number of sessions supported	10	-
	Number of concurrent jobs/user apps that can be run across all users	4	8RU
		15	39RU
	Number of scheduled jobs across all root scopes	10	-
	Number of instances per job that can view scheduler details/logs	100	-
	Maximum number of running instances across root scopes	5	-
Number of tracked inventory items .	Maximum number of IP Addresses that can be tracked across all root scopes	1,500,000	39RU
		500,000	8RU
		70,000	ESX
	Maximum number of subnets that can be tracked across all root scopes	200,000	39RU
		50,000	8RU
		7,000	ESX
	Maximum number of IP Addresses that can be tracked per tenant	6,000 / 100 licenses	
	Maximum number of subnets that can be tracked per tenant	120 / 100 licenses	
Visualization Data Source (VDS)	Maximum number of VDS supported	100	-
	Space for all VDS across Tetration cluster	500G	-
	Namespace limit (file/dir inodes) for all VDS	500000	-

21.6 Data-In / Data-Out

[3] User Apps are not supported in TaaS.

Feature	Metric	Limit	ESXi/8RU/39RU/TaaS/-
Data Lake	Upload limit per dataset	10G	-
	Download limit per dataset	10G	-
	Default instances of data source per user	3[4]	-
	Overall size of user data across all user apps, datasets	5TB	-
	Overall size of shared data across all user apps, datasets	5TB	-
	Namespace limit (file/dir inodes) for all user data	5M	-
	Namespace limit (file/dir inodes) for all shared data	5M	-
Data Sink	Number of data sinks supported per appliance	10	-
	Ingestion limit	80G per hour[5]	-
Data Taps	Number of data taps supported per appliance	10	-
External APIs	Tetration Open API instances per user	3	-

[4] Default number of instances of data source per user can be changed using *ImposeRetention* API.

[5] Ingestion limit of data sinks are calculated based on 39RU to be around 80G/hour. This limit is shared across different data sinks. Smaller Tetration clusters would have a reduced ingestion limit.