

Contrail Insights Installation Guide

Published
2023-10-03

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Contrail Insights Installation Guide

Copyright © 2023 Juniper Networks, Inc. All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <https://support.juniper.net/support/eula/>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide | v

1

Contrail Insights Installation Requirements

Contrail Insights General Requirements | 2

Contrail Insights Agent Requirements | 5

Platform Dependencies | 9

2

Installing Contrail Insights

Contrail Insights Installation for Kubernetes | 11

Contrail Insights Installation for NorthStar | 25

Contrail Insights Installation and Configuration for OpenStack | 31

Contrail Insights Installation for Ubuntu Focal | 45

Contrail Insights Installation for Containerized OpenStack (OpenStack Kolla, Red Hat OpenStack Platform 13) | 46

Contrail Insights Installation for OpenStack in HA | 57

Contrail Insights Installation for OpenStack Helm | 62

Contrail Insights Installation for Standalone | 74

Requirements | 74

Installation | 75

Upgrade Contrail Insights for Standalone | 80

Uninstall Contrail Insights from Standalone | 80

Contrail Insights Settings | 80

3

Ansible Configuration

Bare Host | 86

Contrail Insights MultiCluster Mode | 87

Contrail Insights MultiCluster Proxy	89
Contrail Insights Port List	96
Contrail Insights Role-Based Access	102
Configure Network Device from JSON File	106
Contrail Insights Plug-Ins	115
Contrail Insights User-Defined Plug-Ins	117
Instance Scope Plug-Ins	121
Contrail Insights Object Plug-In	128
Custom SNMP Plug-Ins	134
Custom Sensors for JTI, gRPC, and NETCONF	137
Remote Hosts	141
Monitor NFX250 with Contrail Insights Agent	144
Contrail Insights SDKs	150
Contrail Insights with SSL (HTTPS) Enabled	153
Service Monitoring Ansible Variables	153
OpenStack Services Monitoring Using Service Group Profiles	160
Ansible Configuration Variables	164

About This Guide

Use this guide to install Contrail Insights (formerly known as AppFormix). Contrail Insights manages intent-driven operations, visibility, and reporting for Multicloud and Network Functions Virtualization (NFV).

1

CHAPTER

Contrail Insights Installation Requirements

[Contrail Insights General Requirements](#) | 2

[Contrail Insights Agent Requirements](#) | 5

[Platform Dependencies](#) | 9

Contrail Insights General Requirements

IN THIS SECTION

- [Hardware Requirements](#) | 2
- [Software Requirements](#) | 3
- [Network Requirements](#) | 3
- [User Requirements](#) | 4
- [Agent Requirements](#) | 5

As described in [Contrail Insights Architecture](#), the Contrail Insights software installs onto two types of hosts. A Platform host executes the control plane services. Compute nodes execute the Contrail Insights Agent that analyzes metrics and evaluates policies.

Contrail Insights provides Ansible playbooks to install and uninstall the software on compute hosts and the Platform host. The Ansible playbooks can be executed from a separate host other than the Platform host and compute hosts.

The following requirements apply to installations of Contrail Insights in all environments ("[Contrail Insights Installation and Configuration for OpenStack](#)" on page 31, "[Contrail Insights Installation for NorthStar](#)" on page 25, "[Contrail Insights Installation for Kubernetes](#)" on page 11, and so on). See specific installation topics for additional requirements.

Hardware Requirements

The Platform host, on which Contrail Insights Platform is installed has the following requirements.

- CPU: 16 cores (virtual or physical)
- Memory: 64 GB
- Storage: 300 GB (recommended)

Software Requirements

- docker 17.03.1-ce, installed on the Platform host(s).
- docker version 3.7.1, installed on the Platform host(s).
- Ansible versions 2.3.0 and earlier than 2.8.0, installed on a host that has SSH access to Platform hosts and compute hosts to which Contrail Insights will be deployed.

Starting with Contrail Insights Release 3.2.8 and later, Ansible version 2.4.0 through Ansible version 2.8.9 is supported.

Starting in Contrail Insights Release 3.3.11, if Python version 3 is installed, the minimum required Ansible version is 2.5.1.

- httplib2 must be installed on the host where Ansible is executed.

Both Docker packages are installed by the Contrail Insights playbooks and are installed on the Platform host only.

NOTE: Starting with Contrail Insights Release 3.3.0, Python 3.6.8 is the minimum Python version required on all hosts where Agent gets installed.

Ansible can be executed from any host that can reach the Platform host and compute hosts through SSH. A supported version of Ansible may already be available in your environment. If not, Ansible can be installed on the Platform host, or a separate deployment host from which the playbooks are executed. On the host on which Ansible is executed, httplib2 must be installed (required by Ansible modules used in the Contrail Insights playbooks).

Network Requirements

IP connectivity:

- One IP address for the Platform host.
- Platform host must have IP connectivity to reach compute nodes.
- Remote agents must have IP connectivity to reach management address of:
 - Network devices for SNMP polling.
 - Remote hosts for SNMP polling.

- Remote hosts for Intelligent Platform Management Interface (IPMI) polling.
- Platform host requires Internet connectivity during installation.
- Dashboard client (in browser) must have IP connectivity to Platform host.

DNS:

- Platform host must be able to resolve hostnames of compute nodes.
- Compute nodes must be able to resolve hostname of Platform host.
- Dashboard client (in browser) must be able to resolve hostname of Platform host.

TCP port requirements:

Users interact with the Dashboard client that runs in a Web browser. The Dashboard client communicates with control plane services over port 9000 on the Platform host.

Agent serves REST API on port 42595 on compute hosts. Controller must be able to open connections to the agent for configuration. Agent opens connections to controller and DataManager using port 9000. (A proxy routes API requests to the appropriate service.)

Within the backend management network, Contrail Insights components listen on the following ports:

- Contrail Insights Platform serves REST API on port 7000.
- OpenStack adapter serves REST API on port 7500.
- DataManager serves REST API on port 8090.
- Dashboard service listens on port 9000 for connections from browser.

The specific port numbers can be configured to suitable values to meet requirements of a given environment.

User Requirements

- Root or sudo access on Platform host.
- Root or sudo access on compute node(s) to install agent package.
- See also ["Contrail Insights Agent Requirements" on page 5](#).

Agent Requirements

Contrail Insights Agent executes on each compute node. See ["Contrail Insights Agent Requirements" on page 5](#) for details.

RELATED DOCUMENTATION

[Contrail Insights Installation for Standalone](#) | 74

Contrail Insights Agent Requirements

IN THIS SECTION

- [Contrail Insights Agent Supported Platforms](#) | 5
- [Software Requirements](#) | 6
- [System Capability Requirements](#) | 8
- [Network Requirements](#) | 8
- [Resource Requirements](#) | 8

Contrail Insights Agent runs on a host to monitor resource consumption of the host itself and the virtual machines and containers executing on that host.

Contrail Insights Agent Supported Platforms

- CentOS 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.9
Contrail Insights Release 3.3.6 supports CentOS 7.9.
- Debian 8
- Red Hat Enterprise Linux 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 8.4
Contrail Insights Release 3.3.10 supports Red Hat Enterprise Linux 8.4.

- Ubuntu 14.04, 16.04, 18.04, 20.04 (Focal)

Contrail Insights Release 3.3.5 supports Ubuntu 20.04 (Focal).

Software Requirements

System-Level Dependencies

The following software packages are system-level packages that must be installed on the host on which the agent runs. The minimum compatible version is listed.

- libvirt 1.2.2
- msr-tools 1.3 ** (also requires Linux kernel 'msr' module to be loaded)
- Python 2.7.5

Python 2 is not installed by default with Ubuntu 20.04 (Focal). Starting in Contrail Insights Release 3.3.11, if Python version 3 is installed, the minimum required Ansible version is 2.5.1. For more information, see ["Contrail Insights Installation for Ubuntu Focal" on page 45](#).

NOTE: Starting with Contrail Insights Release 3.3.0, Python 3.6.8 is the minimum Python version required on all hosts where Agent gets installed.

- smartmontools 6.2 **
- systemd 204 (for udevadm. package is 'udev' on Ubuntu.)
- util-linux 2.20.1

** Optional, some metrics will not be available without this dependency.

Python Dependencies

During installation of the agent, all Python dependencies are installed inside of a Python virtualenv. This isolates the agent's Python dependencies from the host system. The agent depends on the following Python packages.

NOTE: Python 2 is not installed by default with Ubuntu 20.04 (Focal). Starting in Contrail Insights Release 3.3.11, if Python version 3 is installed, the minimum required Ansible version is 2.5.1. For more information, see ["Contrail Insights Installation for Ubuntu Focal" on page 45](#).

- aniso8601 0.92
- backports.ssl-match-hostname 3.4.0.2
- docker-py 1.1.0
- Flask 0.10.1
- Flask-RESTful 0.3.5
- flask-restful-swagger 0.19
- IPy 0.81
- itsdangerous 0.24
- Releases earlier than Release 3.3.11: Jinja2 2.7.3
Release 3.3.11 and later: Jinja2 3.0.3
- jsonschema 2.4.0
- libvirt-python 1.2.14
- MarkupSafe 0.23
- protobuf 2.5.0
- protobuf-to-dict 0.1.0
- psutil 2.2.1
- python-iptables 0.11.0
- pytz 2014.10
- requests 2.6.0
- six 1.5.2
- urllib3 1.10.2
- websocket-client 0.26.0

- Werkzeug 0.9.4
- xmlltodict 0.9.2

System Capability Requirements

Contrail Insights Agent requires capability to read hardware and process-level metrics. Agent can run as a root or non-root user. Agent package will create an `appformix` Linux user account and group that has sufficient privileges for operation. Some metrics are not available when the agent is run as non-root user.

- `libvirtd` group permission to access `qemu:///system/` (via `python-libvirt`)
- Read access to disk devices (e.g., `/dev/sda`) for SMART counters (via `smartctl`). Requires root privilege.
- Read access to CPU MSR registers for CPU temperature (`/dev/cpu/*/msr`). Requires root privilege.

Network Requirements

Contrail Insights Agent communicates with controller components. The following ports are used for communication between the hosts.

- Agent serves an HTTP-based REST API on port 42595.
- Agent opens HTTP connections to Contrail Insights Platform REST API on port 9000.

Resource Requirements

The Contrail Insights Agent CPU and memory footprint is dependent on number of instances monitored on the host and number of alarms configured.

- CPU consumption is 0.1% per instance.
- Memory footprint is 40 MB + 25 KB per configured alarm.

Release History Table

Release	Description
3.3.5	Contrail Insights Release 3.3.5 supports Ubuntu 20.04 (Focal).

Platform Dependencies

When installing Contrail Insights control plane services on a Platform Host, the Ansible playbooks fetch third-party dependencies, such as Mongo and Redis, from repositories on the Internet. In some environments, the Contrail Insights Platform host might not be able to open connections to external hosts. The Contrail Insights software can be installed in those environments by loading the dependencies prior to executing the Ansible playbook.

Installation of Contrail Insights Agent does not require fetching any dependencies. The following steps apply only to an Contrail Insights Platform host.

The following steps require a host that has Docker installed and can pull images from the Internet (specifically, from Docker Hub).

1. On a host (for example, host1) that has Internet access, pull Mongo and Redis images, and save them to a file:

```
host1$ docker pull mongo:3.2.17
host1$ docker pull redis:3.0.4
host1$ docker save mongo:3.2.17 redis:3.0.4 | gzip > mongo_redis.tar.gz
```

2. Copy `mongo_redis.tar.gz` to the Contrail Insights Platform Host.
3. On the Contrail Insights Platform Host, load the Docker images.

```
appformix01$ docker load -i mongo_redis.tar.gz
```

With the dependencies loaded, proceed with the installation of Contrail Insights by executing the Ansible playbook appropriate for the environment.

RELATED DOCUMENTATION

[Contrail Insights General Requirements | 2](#)

[Contrail Insights Agent Requirements | 5](#)

[Contrail Insights Role-Based Access | 102](#)

[Contrail Insights Settings | 80](#)

[Service Monitoring Ansible Variables | 153](#)

2

CHAPTER

Installing Contrail Insights

[Contrail Insights Installation for Kubernetes | 11](#)

[Contrail Insights Installation for NorthStar | 25](#)

[Contrail Insights Installation and Configuration for OpenStack | 31](#)

[Contrail Insights Installation for Ubuntu Focal | 45](#)

[Contrail Insights Installation for Containerized OpenStack \(OpenStack Kolla, Red Hat OpenStack Platform 13\) | 46](#)

[Contrail Insights Installation for OpenStack in HA | 57](#)

[Contrail Insights Installation for OpenStack Helm | 62](#)

[Contrail Insights Installation for Standalone | 74](#)

[Contrail Insights Settings | 80](#)

Contrail Insights Installation for Kubernetes

IN THIS SECTION

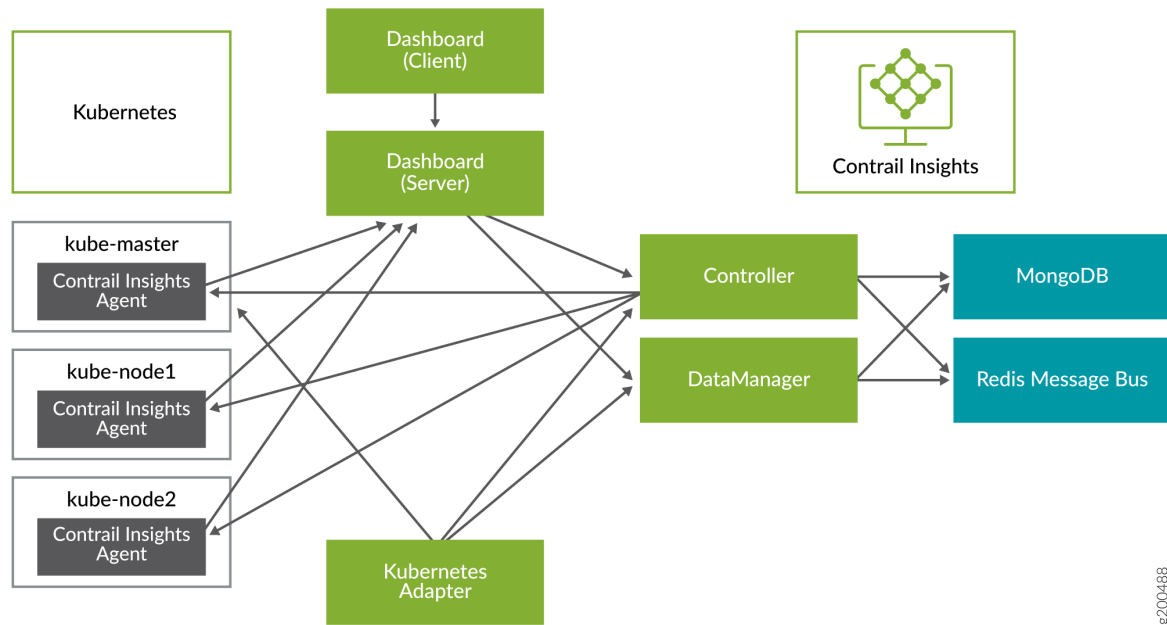
- [Architecture and Terminology | 11](#)
- [Requirements | 12](#)
- [Workflow in Four Steps | 13](#)
- [Initial Setup | 13](#)
- [Configure Kubernetes | 14](#)
- [Install Contrail Insights | 16](#)
- [Set up the Contrail Insights Scheduler Extender | 19](#)
- [Using the Contrail Insights SLA Profile for Scheduling | 21](#)

Architecture and Terminology

Kubernetes cluster nodes	Primary and worker nodes of the Kubernetes cluster being monitored by Contrail Insights. These nodes will run the Contrail Insights Agent.
Contrail Insights Platform node	Node on which Contrail Insights Platform components will be installed. Should be able to reach the Kubernetes cluster nodes.

[Figure 1 on page 12](#) shows the different components of Contrail Insights and how they interact with the Kubernetes cluster.

Figure 1: Contrail Insights and Kubernetes Workflow



Requirements

The following are the requirements for installing Contrail Insights for Kubernetes.

- Kubernetes versions 1.8 and later.
- See "[Contrail Insights General Requirements](#)" on page 2 for hardware and software requirements.
- API access to the Kubernetes API server. Contrail Insights reads information about the cluster from the API server. The token provided during configuration must provide sufficient permission for read-only API calls. Contrail Insights Platform must also be able to open a connection to the host and port on which the API server runs.
- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:
Target customers or use:
Cluster type: Kubernetes
Number of hosts:
Number of instances:
```

Workflow in Four Steps

The installation consists of the following steps:

1. Initial setup.
2. Configuring Kubernetes.
3. Installing Contrail Insights.
4. Setting up the Contrail Insights Scheduler Extender.

Initial Setup

Perform the following steps for the initial setup:

1. Install the following required files on the Contrail Insights Platform node.

```
#Ubuntu
apt-get update
apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
pip install ansible==2.1.2 markupsafe httplib2
```

```
#RHEL/CentOS
yum install epel-release                                #Enable EPEL repository
In case the above command does not work, manually download and install the epel-release
package with one of the below commands, depending on your system's version.
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

```

yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

yum groupinstall 'Development Tools'           #Install development tools
yum install openssl-devel libffi libffi-devel  #Dependencies
yum install python-pip python-devel           #Install Pip
pip install ansible==2.3.0                     #Install Ansible 2.3
pip install markupsafe httpplib2              #Dependencies

```

NOTE: For RHEL, the following iptables rule is needed to access port 9000:

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

2. Edit the `/etc/hosts/` file on the Contrail Insights Platform node and enter the IP addresses of the Kubernetes cluster nodes.

```

vi /etc/hosts
<kube-master-ip> k8s-master
<kube-worker1-ip> k8s-node1
<kube-worker2-ip> k8s-node2

```

3. Set up passwordless SSH between the Contrail Insights Platform node and the Kubernetes cluster nodes. Run the following commands to generate and copy the SSH public keys to all the nodes:

```

ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-master
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-node1
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-node2
ssh-copy-id -i ~/.ssh/id_rsa.pub root@<IP of Contrail Insights Platform node>

```

Configure Kubernetes

Contrail Insights reads information about resources in your Kubernetes clusters. The software requires the **cluster-admin** role or another role that provides read-only access to all objects in the cluster. We recommend that you create a new Service Account for Contrail Insights and assign it the **cluster-admin** role.

If you do not create a new Service Account, then you must provide the token from an existing Service Account that has the required access during the configuration of Contrail Insights.

To create a new Service Account with the required access for Contrail Insights, perform the following steps in the Kubernetes cluster:

1. Create a YAML file with the following:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: appformix
```

2. Create the appformix Service Account using the file you created in Step 1:

```
$ kubectl create -f <file>.yaml
```

3. Confirm that the Service Account has been created. Make a note of its namespace as you'll need this later.

```
$ kubectl describe serviceaccount appformix
Name:          appformix
Namespace:     default
Labels:        <none>
Annotations:   <none>
Image pull secrets: <none>
Mountable secrets:  appformix-token-pkljk
Tokens:         appformix-token-pkljk
Events:        <none>
```

4. Add the cluster-admin role to the appformix Service Account, substituting *<namespace>* for the namespace noted in Step 3:

```
$ kubectl create clusterrolebinding appformix-binding --clusterrole=cluster-admin --
serviceaccount=<namespace>:appformix
```

5. Run the following command to confirm that the `appformix` Service Account has the required access:

```
$ kubectl auth can-i get nodes --as=system:serviceaccount:<namespace>:appformix --all-namespaces
```

The output of the command should be **yes**.

6. Contrail Insights must be configured to communicate with the Kubernetes cluster. Get the following details from the Kubernetes cluster to use during the Contrail Insights installation.

kubernetes_cluster_url This is the URL of the Kubernetes API Server. To get this value, run the following command on the Kubernetes cluster:

```
$ kubectl cluster-info | grep 'Kubernetes master'
Kubernetes master is running at https://172.24.1.173:6443
```

kubernetes_auth_token This is the authentication token of the `appformix` Service Account. To get this value, run the following commands on the Kubernetes cluster:

```
$ kubectl describe serviceaccount appformix
Name:          appformix
Namespace:     default
Labels:        <none>
Annotations:   <none>
Tokens:        appformix-token-pkljk
[...]
$ kubectl describe secret appformix-token-pkljk
Name:          appformix-token-pkljk
Namespace:     default
[...]
token:         eyJhb[...]
```

Install Contrail Insights

To install Contrail Insights:

1. Download the Contrail Insights installation packages from [software downloads](#) to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-kubernetes-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Unzip `contrail-insights-<version>.tar.gz`. This package contains all the Ansible playbooks required to install Contrail Insights.

```
tar -xvzf contrail-insights-<version>.tar.gz
cd contrail-insights-<version>/
```

NOTE: The remaining steps should be executed from within the `contrail-insights-<version>/` directory. Although the product name changed from "AppFormix" to "Contrail Insights," the UI and internal command paths continue to show AppFormix and will reflect the new name at a later date.

3. Using `sample_inventory` as a template, create an inventory file for the installation. List the Kubernetes cluster nodes in the `compute` section and the Contrail Insights Platform node in the `appformix_controller` section.

```
cp sample_inventory inventory
vi inventory
[compute]
k8s-master
k8s-node1
k8s-node2
[appformix_controller]
<IP of the Contrail Insights Platform node>
```

4. Create a directory called `group_vars`. Create a file named `all` inside this directory with configuration variables required by Contrail Insights.

```
mkdir group_vars
vi group_vars/all
appformix_docker_images:
- /path/to/contrail-insights-platform-images-<version>.tar.gz
- /path/to/contrail-insights-dependencies-images-<version>.tar.gz
- /path/to/contrail-insights-kubernetes-images-<version>.tar.gz
- /path/to/contrail-insights-network_device-images-<version>.tar.gz
appformix_dns_version: 2
kubernetes_platform_enabled: True
openstack_platform_enabled: False
kubernetes_cluster_url: <URL from Configuring Kubernetes step 6 above>
kubernetes_auth_token: <token from Configuring Kubernetes step 6 above>
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

NOTE: Deprecation Notice: The `appformix_mongo_cache_size_gb` parameter previously available starting in Contrail Insights 2.19.5 is now deprecated and no longer supported from Contrail Insights 3.2.0 and going forward. Starting with Contrail Insights version 3.2.0, Mongo will be configured to use a maximum of 40 percent of the available memory on the Contrail Insights Platform nodes.

5. To enable network device monitoring in the cluster, include the following in the `group_vars/all` file:

```
# For enabling pre-requisites for packdge installation.
appformix_install_snmp_dependencies: true
appformix_install_jti_dependencies: true
# For running the appformix-network-device-adapter
network_device_discovery_enabled: true
appformix_plugins: '{{ appformix_network_device_factory_plugins }}'
# After 3.1, SNMP Traps can be enabled also so appformix_plugins can be specified as below:
# appformix_plugins: '{{ appformix_network_device_factory_plugins }} +
{{ appformix_snmp_trap_factory_plugins }}'
```

6. Run the Ansible playbook.

```
ansible-playbook -i inventory appformix_kubernetes.yml
```

Playbook should run to completion without any errors.

7. Log into the Contrail Insights Dashboard at:

```
http://<contrail-insights-node-ip>:9000
```

8. Log in using the tokenId from the following file on the Contrail Insights Platform node:

```
/opt/appformix/etc/appformix_token.rst
```

Set up the Contrail Insights Scheduler Extender

Contrail Insights comes with a Scheduler Extender module that can be added to the Kubernetes scheduler. With this module in place, the Kubernetes scheduler will use user-defined SLA policies in addition to its default policies to determine where to schedule a pod in the cluster.

To set up the Scheduler Extender:

1. Create a JSON file describing the Contrail Insights Scheduler Extender. Place this file inside `/etc/kubernetes` on the Kubernetes primary node.

```
vi /etc/kubernetes/appformix_scheduler_extender.json
{
  "kind" : "Policy",
  "apiVersion" : "v1",
  "predicates" : [
    {"name": "NoVolumeZoneConflict"},
    {"name": "MaxEBSVolumeCount"},
    {"name": "MaxGCEPDVolumeCount"},
    {"name": "MatchInterPodAffinity"},
    {"name": "NoDiskConflict"},
    {"name": "GeneralPredicates"},
    {"name": "PodToleratesNodeTaints"},
    {"name": "CheckNodeMemoryPressure"},
  ]
}
```



```

    {"name": "CheckNodeDiskPressure"}
  ],
  "priorities" : [
    {"name" : "SelectorSpreadPriority", "weight" : 1},
    {"name" : "InterPodAffinityPriority", "weight" : 1},
    {"name" : "LeastRequestedPriority", "weight" : 1},
    {"name" : "BalancedResourceAllocation", "weight" : 1},
    {"name" : "NodePreferAvoidPodsPriority", "weight" : 10000},
    {"name" : "NodeAffinityPriority", "weight" : 1},
    {"name" : "TaintTolerationPriority", "weight" : 1}
  ],
  "extenders": [
    {"urlPrefix" : "http://<contrail-insights-platform-node-ip>:9000/appformix/v1.0/
kubernetes_adapter",
      "filterVerb" : "kubernetes_schedule_request",
      "weight": 1,
      "enableHttps" : false}
  ]
}

```

2. Add the extender to the kube-scheduler on the primary node by adding the `--policy-config-file` option to the `spec.containers.command` block:

```

vi /etc/kubernetes/manifests/kube-scheduler.yaml
...
- command:
  - kube-scheduler
  - --address=127.0.0.1
  - --leader-elect=true
  - --kubeconfig=/etc/kubernetes/scheduler.conf
  - --policy-config-file=/etc/kubernetes/appformix_scheduler_extender.json
...

```

3. Update the kube-scheduler container by restarting the kubelet service on the primary node.

```
service kubelet restart
```

The kube-scheduler is now running with the Contrail Insights Scheduler Extender.

By default, Kubernetes does not allow any user pods to be scheduled on the primary node. To really see the Contrail Insights Scheduler Extender in action on a 3-node Kubernetes cluster, enable scheduling on the Kubernetes primary node with the following command:

```
kubectl taint nodes master node-role.kubernetes.io/master:NoSchedule-
```

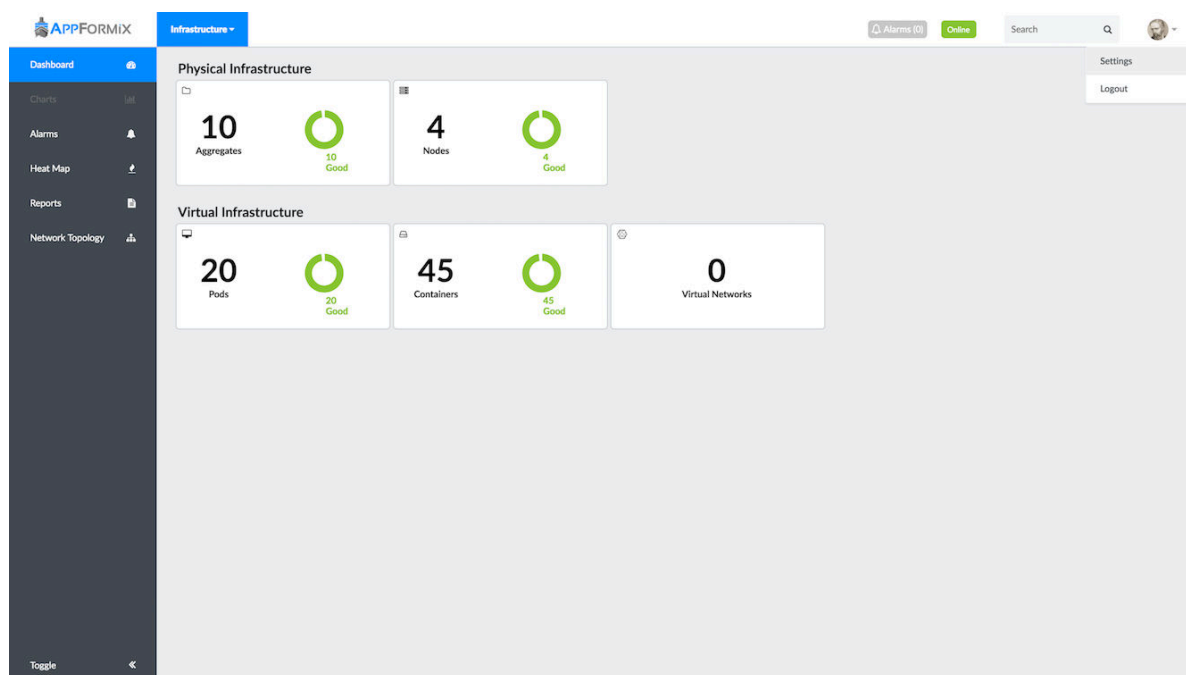
Using the Contrail Insights SLA Profile for Scheduling

Contrail Insights ships with a default Scheduling SLA that includes alarms for missed heartbeat, high CPU load, and high memory usage.

To change the profiles in the Scheduling SLA, do the following:

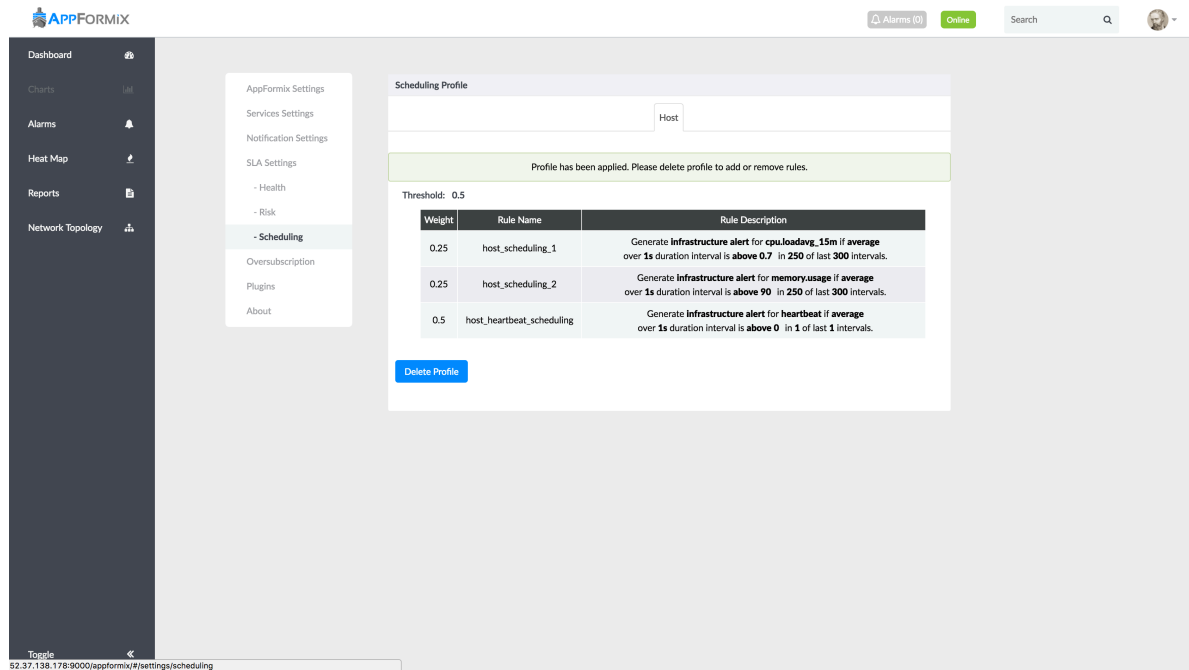
1. Select **Settings** from the list in the top right of the Dashboard, then select **SLA Settings > Scheduling**.

Figure 2: Contrail Insights Settings in Dashboard



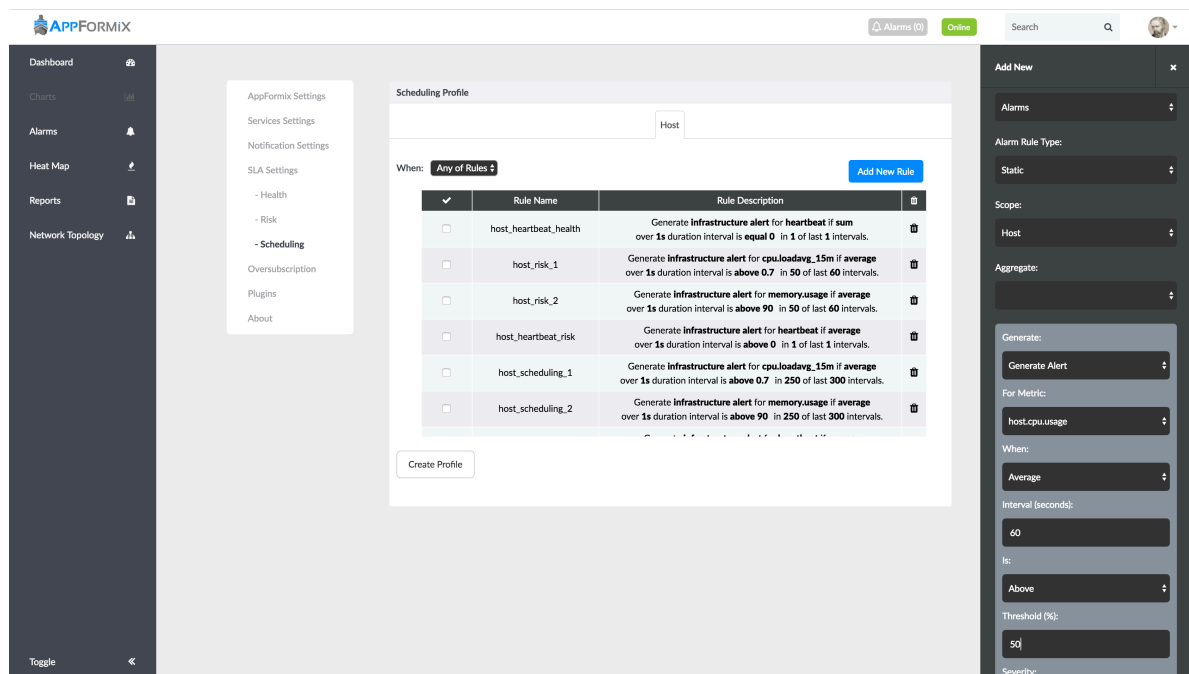
2. Click **Delete Profile** to delete the existing profile.

Figure 3: Delete Scheduling Profile



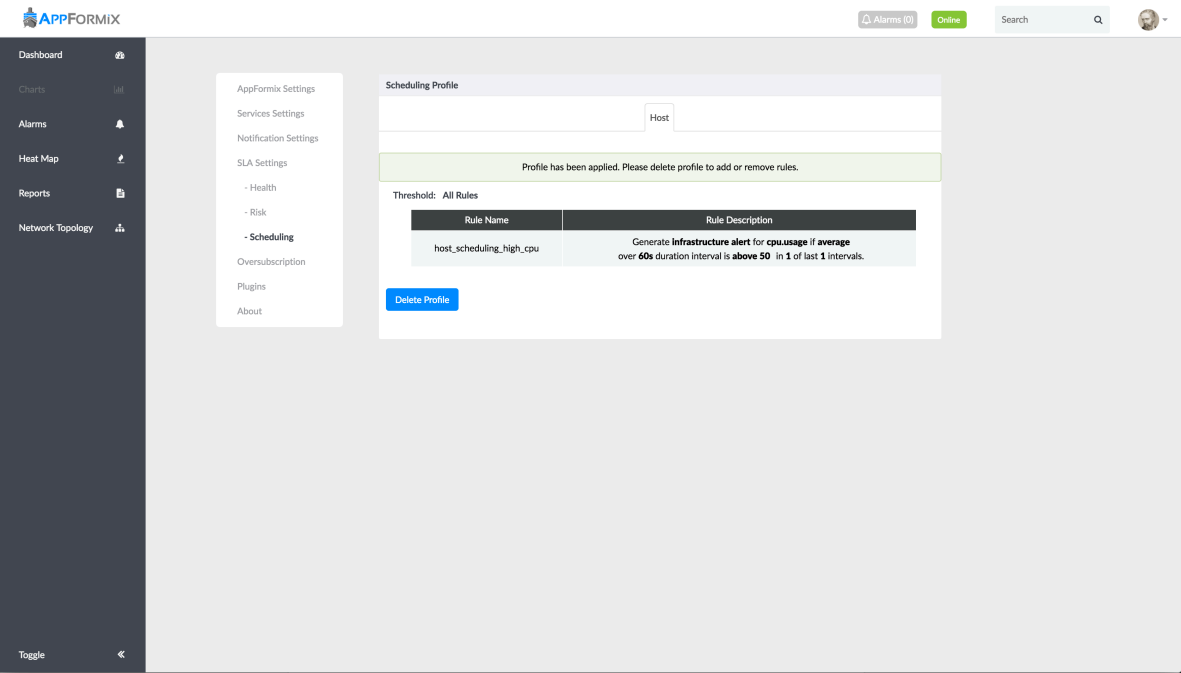
3. Click **Add New Rule** and define a new alarm.

Figure 4: Add New Rule in Scheduling Profile



- 4. Select the newly created alarm from the list of available alarms and click **Create Profile**. You can add several alarms with custom weights to the SLA profile.

Figure 5: Create Profile in Scheduling SLA



- 5. To see the Scheduler Extender in action, generate some load on one of the Kubernetes cluster nodes so that the Scheduling SLA is violated. Check the status of the SLA from the Alarms page.

Figure 6: Violated Scheduling SLA in Alarms page

The screenshot shows the AppFormix interface with the 'Alarms' tab selected. The 'Latest Alarm States' section displays a table of alarms. The first alarm, 'host_scheduling...', is in a 'triggered' state, indicating a violated Scheduling SLA. The details of this alarm are shown in a table below the main list.

Name	Time Ago	State	Details												
host_scheduling...	<1m	triggered	<table border="1"> <thead> <tr> <th>Severity</th> <th>Host</th> <th>analytics_type</th> <th>scheduling</th> </tr> </thead> <tbody> <tr> <td>critical</td> <td>node1</td> <td>new_state</td> <td>scheduling not ok</td> </tr> <tr> <td></td> <td></td> <td>old state</td> <td>scheduling ok</td> </tr> </tbody> </table>	Severity	Host	analytics_type	scheduling	critical	node1	new_state	scheduling not ok			old state	scheduling ok
Severity	Host	analytics_type	scheduling												
critical	node1	new_state	scheduling not ok												
		old state	scheduling ok												

On the right side, the 'Alarm Rules' section shows a list of rules, including 'host_scheduling_transitio...', which is currently disabled.

Then create some pods on the Kubernetes cluster and check which nodes they get scheduled on. The node that is violating the scheduling SLA will not get any new pods scheduled on it.

```
root@master:~# kubectl scale rc bpsloadgenrc --replicas 4; kubectl get pods -o wide
replicationcontroller "bpsloadgenrc" scaled
NAME                READY   STATUS    RESTARTS   AGE   IP            NODE
bpsloadgenrc-2lsbg  1/1     Running   0           3h    10.244.2.20   node1
bpsloadgenrc-4kddr  0/1     ContainerCreating  0           0s    <none>        master
bpsloadgenrc-fm16h  1/1     Running   0           3h    10.244.0.18   master
bpsloadgenrc-sz2r6  0/1     ContainerCreating  0           0s    <none>        master
fectrl-3278x        1/1     Running   0           3h    10.244.1.24   node1
redismaster          1/1     Running   0           3h    10.244.2.19   node1
redissc-ffsqd       1/1     Running   0           3h    10.244.0.17   master
root@master:~# kubectl scale rc bpsloadgenrc --replicas 6; kubectl get pods -o
wide
replicationcontroller "bpsloadgenrc" scaled
NAME                READY   STATUS    RESTARTS   AGE   IP            NODE
bpsloadgenrc-2lsbg  1/1     Running   0           3h    10.244.2.20   node1
bpsloadgenrc-4kddr  1/1     Running   0           17s   10.244.1.25   master
bpsloadgenrc-9zgx5  0/1     ContainerCreating  0           0s    <none>        master
bpsloadgenrc-fm16h  1/1     Running   0           3h    10.244.0.18   master
bpsloadgenrc-p7drd  0/1     ContainerCreating  0           0s    <none>        master
```

bpsloadgenrc-sz2r6	1/1	Running	0	17s	10.244.0.19	master
fectrl-3278x	1/1	Running	0	3h	10.244.1.24	node1
redismaster	1/1	Running	0	3h	10.244.2.19	node1
redissc-ffsqd	1/1	Running	0	3h	10.244.0.17	master

RELATED DOCUMENTATION

- [Contrail Insights Agent Requirements | 5](#)
- [Platform Dependencies | 9](#)

Contrail Insights Installation for NorthStar

IN THIS SECTION

- [Requirements | 26](#)
- [Install the NorthStar Controller | 26](#)
- [Install Contrail Insights | 26](#)

NorthStar controller for traffic optimization provides granular visibility into, and control over, IP/MPLS flows in large service provider and enterprise networks. NorthStar automates the creation of traffic-engineering paths across the network, increasing network utilization and enabling a customized and programmable networking experience.

Integrating NorthStar and Contrail Insights is a robust solution where Contrail Insights monitors nodes in the network topology by collecting telemetry data and notifies NorthStar using HTTP endpoints to take appropriate action when an alarm is triggered. Some actions that can happen with this integration are:

- Putting a node in the network into maintenance mode when node metrics such as CPU, memory, etc. are above a threshold.
- Changes in LSP when an interface on a node is affected.

Requirements

- NorthStar controller version 4.0.0
- See "[Contrail Insights General Requirements](#)" on page 2 for hardware and software requirements.

- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:  
Target customers or use:  
Cluster type: NorthStar  
Number of hosts:  
Number of instances:
```

Install the NorthStar Controller

For information on installing the NorthStar Controller, see the [NorthStar Documentation](#).

Install Contrail Insights

To integrate Contrail Insights with NorthStar:

1. Download the following packages from <https://www.juniper.net/support/downloads/?p=appformix> on to the host where you will run Ansible.

```
contrail-insights-<version>.tar.gz  
contrail-insights-dependencies-images-<version>.tar.gz  
contrail-insights-platform-images-<version>.tar.gz  
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Install Ansible on the installer node.

Ansible will install **docker** and **docker-py** on the `appformix_controller`.

```
# sudo apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
# sudo yum install python-pip python-devel gcc gcc-c++ make openssl-devel libffi-devel
# sudo easy_install pyOpenSSL
# sudo pip install ansible==2.7.6 markupsafe httpplib2
```

If you are using Ansible 2.3, use the following commands.

```
# sudo pip install ansible==2.3 markupsafe httpplib2 cryptography==1.5
```

3. Install **python** and **python-pip** on the `appformix_controller` so that Ansible can run between the installer node and the `appformix_controller` node.

```
# sudo apt-get install -y python python-pip
```

4. Install the python **pip** package on the hosts where the Contrail Insights Agents run.

```
# apt-get install -y python-pip
```

5. To enable passwordless login to the Contrail Insights Platform by Ansible, create a SSH public key on the node where Ansible playbooks are run, and then copy the key to the `appformix_controller` node.

```
$ ssh-keygen -t rsa                                #Creates keys
$ ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host>    #Copies key from the node to
appformix_controller node
```

Or, you can use `ansible_connection=local` in the inventory example.

```
[appformix_controller]
172.16.70.119 ansible_connection=local
```


6. Create an Ansible inventory. Create a directory named **inventory**, and in that directory edit the **inventory/hosts** file. For example:

```
#
Contrail Insights Network Agent hosts
[appformix_network_agents]
172.16.10.1
172.16.10.2
# Contrail Insights Platform host
[appformix_controller]
172.16.70.119
```

The host listed under `appformix_controller` is mandatory. This is the host onto which the Contrail Insights Platform is installed.

The Contrail Insights Agent is installed on the hosts listed under `appformix_network_agents`. This field is optional as the Agent is already installed on the `appformix_controller` host.

7. Within the inventory directory, create a subdirectory called **inventory/group_vars**. Then create an **inventory/groups_vars/all** file inside that subdirectory. For example:

```
$ mkdir inventory/group_vars
$ touch inventory/group_vars/all
```

8. Add the following configuration parameters to the **all** file.

NOTE: All the variables in `group_vars` are mandatory. If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`.

```
appformix_docker_images:
- path-to/contrail-insights-platform-images-<version>.tar.gz
- path-to/contrail-insights-dependencies-images-<version>.tar.gz
- path-to/contrail-insights-network_device-images-<version>.tar.gz

# For enabling pre-requisites for package installation.
appformix_network_device_monitoring_enabled: true

appformix_plugins:
```

```

- { plugin_info: 'certified_plugins/snmp_network_device_usage.json' }
- { plugin_info: 'certified_plugins/snmp_config_ifxtable_mib.json' }
- { plugin_info: 'certified_plugins/snmp_config_juniper_device_mib.json' }
- { plugin_info: 'certified_plugins/snmp_config_tcp_mib.json' }
- { plugin_info: 'certified_plugins/snmp_config_juniper_alarm_mib.json' }
- { plugin_info: 'certified_plugins/snmp_config_device_resources_mib.json' }
- { plugin_info: 'certified_plugins/jti_config_all_sensors.json' }
- { plugin_info: 'certified_plugins/grpc_config_all_sensors.json' }

appformix_northstar_monitoring_enabled: true
northstar_user_name: <NorthStar UI Login Id>
northstar_password: <NorthStar UI Login Password>
northstar_controller_url: https://<Northstar-controller-ip>:8443
northstar_tenant_id: 1
northstar_topology_id: 1

```

9. Unpack the Contrail Insights TAR file, and change directory to the directory that is created. For example:

```

tar -xvzf contrail-insights-<version>.tar.gz
cd contrail-insights-<version>/

```

NOTE: The remaining steps should be executed from within the `contrail-insights-<version>/` directory. Although the product name changed from "AppFormix" to "Contrail Insights," the UI and internal command paths continue to show AppFormix and will reflect the new name at a later date.

10. Run Ansible using the path to the inventory directory created in steps 6 and 7. For example:

```

$ ansible-playbook -i ../inventory appformix_standalone.yml

```

11. If you are running the playbooks as the root user, this step can be skipped. If you are running the playbooks as any other user, for example ubuntu, that user account must have access to the docker user group.

For example, the below command adds the user ubuntu to the docker group.

```
$ sudo usermod -aG docker ubuntu
```

- Obtain the token to log in to the Contrail Insights UI from the `appformix_token.rst` file located on the Contrail Insights Platform host. For example:

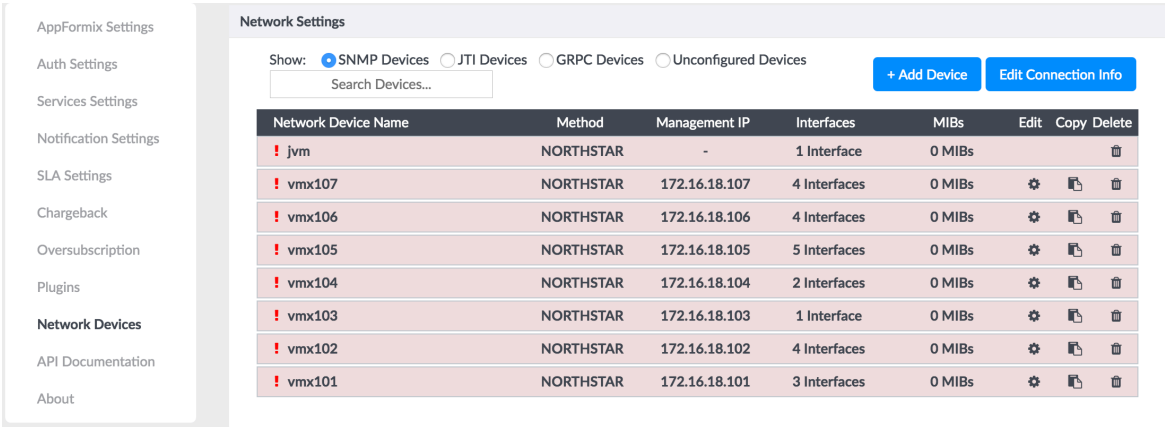
```
$ cat /opt/appformix/etc/appformix_token.rst
```

- Use the token from Step 12 to login to the Contrail Insights dashboard.
- Select **Settings > Network Devices**.

The network devices from NorthStar that are discovered by Contrail Insights appear in a list. Each device that has an SNMP/JTI/gRPC config is posted to the Contrail Insights Platform.

Figure 7 on page 30 shows an example of network devices of a NorthStar monitored network topology discovered by Contrail Insights.

Figure 7: Example Discovered Network Devices



RELATED DOCUMENTATION

Contrail Insights General Requirements	2
Contrail Insights Agent Requirements	5
Platform Dependencies	9

Contrail Insights Installation and Configuration for OpenStack

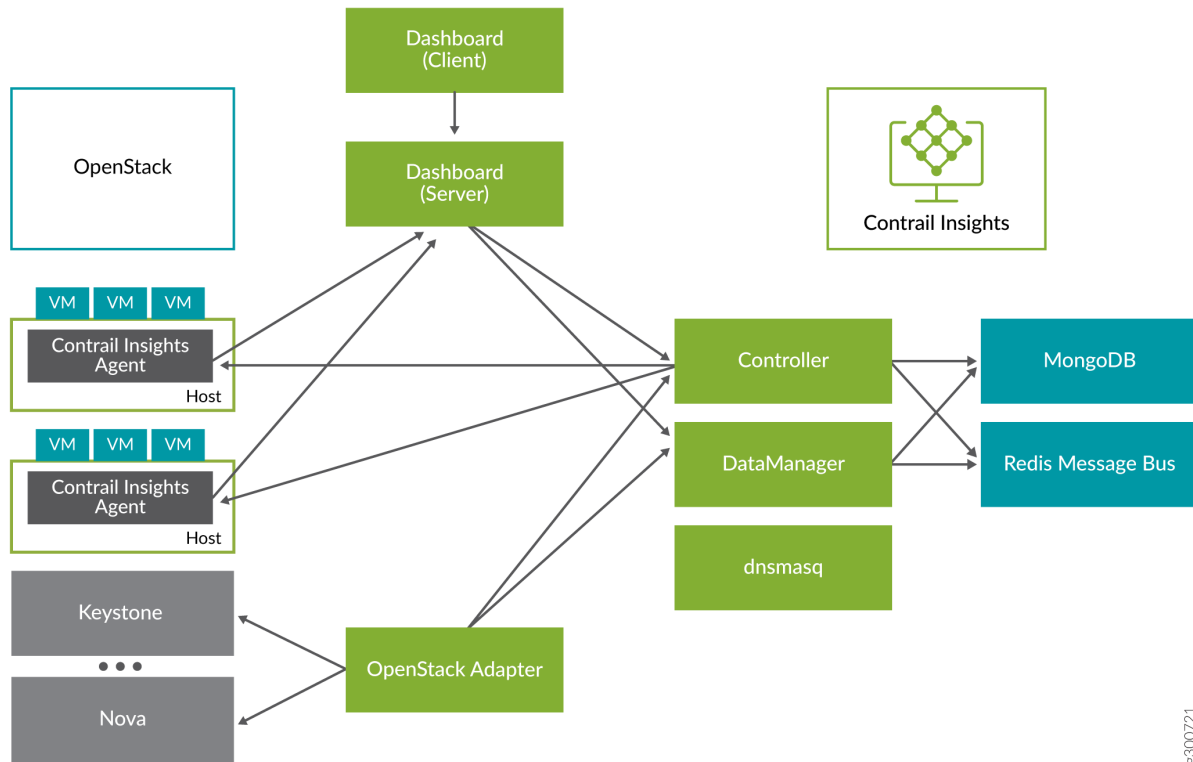
IN THIS SECTION

- [Getting Started with OpenStack | 31](#)
- [Requirements | 32](#)
- [Workflow in Three Steps | 33](#)
- [Configure OpenStack | 33](#)
- [Create Ansible Inventory | 34](#)
- [Install Contrail Insights | 36](#)
- [Additional Ansible Configuration | 39](#)
- [Upgrade Notices | 42](#)
- [Remove a Node from Contrail Insights | 44](#)
- [Upgrade Contrail Insights for an OpenStack Cluster | 44](#)
- [Uninstall Contrail Insights from an OpenStack Cluster | 45](#)

Getting Started with OpenStack

Contrail Insights provides resource control and visibility for hosts and virtual machines in an [OpenStack](#) cluster. This topic explains how to install Contrail Insights for an OpenStack cluster. Contrail Insights Agent runs on a host to monitor resource consumption of the host itself and the virtual machines executing on that host. [Figure 8 on page 32](#) shows the Contrail Insights architecture with OpenStack.

Figure 8: Contrail Insights Architecture with OpenStack



g300721

- Contrail Insights Agent monitors resource usage on the compute nodes.
- Contrail Insights Platform offers REST APIs to configure the system.
- Contrail Insights DataManager stores data from multiple Agents.
- Contrail Insights Dashboard provides a Web-based user interface.
- An adapter discovers platform-specific resources and configures the Contrail Insights Platform.
- Adapters exist for OpenStack and Kubernetes.

Requirements

The following are the requirements for installing Contrail Insights for OpenStack.

- Supported OpenStack versions: Icehouse, Juno, Kilo, Liberty, Mitaka, Newton, Ocata.
- See "[Contrail Insights General Requirements](#)" on page 2 for software and hardware requirements.

- An administrator account for OpenStack.
- API access to OpenStack services: Cinder, Glance, Heat, Keystone, Neutron, Nova, and Swift. Contrail Insights reads information from these services. The administrator account must provide sufficient permission for read-only API calls. Further, Contrail Insights Platform must be able to open connections to the host and port on which these services listen. Contrail Insights can be configured to use the `admin`, `internal`, or `public` service endpoints. See `OS_ENDPOINT_TYPE` in OpenStack environment variables in the section Installing Contrail Insights.
- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:
Target customers or use:
Cluster type: OpenStack
Number of hosts:
Number of instances:
```

Workflow in Three Steps

Installation consists of the following steps:

1. Configure OpenStack.
2. Create Ansible inventory.
3. Install Contrail Insights.

Configure OpenStack

To create an administrator account for Contrail Insights, perform the following steps in the OpenStack Horizon dashboard:

1. Create a user account and name it **appformix**.
2. Select a new project for the user account.
3. Select role as **admin**.

Create Ansible Inventory

Ansible is used to deploy the software to the compute node(s) and the Platform Host. An Ansible inventory file describes groups of hosts in your cluster. Define the inventory in a separate location than the release files, so that the inventory may be reused for an upgrade.

Contrail Insights requires two groups *compute* and *appformix_controller*. Each group lists the hosts in that group. Only the agent is installed on the *compute* hosts. The agent and the Contrail Insights Platform services are installed on the *appformix_controller* host.

Optionally, an *openstack_controller* group can be defined. The agent is installed on hosts in this group to monitor the hosts that execute OpenStack controller services. (New in v2.3.0)

Create a directory inventory (or name of your choice) that contains a hosts file and a group_vars/all file. For example:

```
inventory/  
  hosts          # inventory file  
  group_vars/  
    all          # configuration variables
```

The inventory/hosts file contains the list of hosts in each group. For example:

```
[appformix_controller]  
appformix01  
  
[compute]  
compute01  
compute02  
compute03  
  
[openstack_controller]
```

```
openstack_infra01
openstack_infra02
```

See [Ansible inventory documentation](#).

Ansible Configuration Variables

The Contrail Insights software includes a number of Ansible roles to perform the configuration of Contrail Insights settings. Define the values of variables in the `inventory/group_vars/all` file, in order to be able to use the settings and inventory for future upgrades.

In the `inventory/group_vars/all` file, configure the following variables for installation of Contrail Insights for OpenStack.

```
appformix_docker_images:
  - /path/to/contrail-insights-platform-images-<version>.tar.gz
  - /path/to/contrail-insights-dependencies-images<version>.tar.gz
  - /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

Refer to "[Platform Dependencies](#)" on [page 9](#) for steps to install dependencies on a Platform Host that cannot fetch files from the Internet.

Configure an HTTP Proxy for Fetching Prerequisites

The Ansible playbook will fetch files from the Internet to install prerequisites on the Platform Host. If the Platform Host requires an HTTP proxy to access the Internet, configure the following variables in `inventory/group_vars/all`:

```
http_proxy_url: 'http://proxy.example.com:1234'

prerequisites_env:

  http_proxy: '{{ http_proxy_url }}'
  https_proxy: '{{ http_proxy_url }}'
```

The `prerequisites_env` is a dictionary that defines environment variables that will be used when invoking commands to install prerequisites. In the above example, the same proxy URL (`http://proxy.example.com:1234`) is used for both the `http_proxy` and `https_proxy` environment variables because the single proxy can be used to access HTTP and HTTPS URLs. As a convenience, the proxy URL is defined once in the `http_proxy_url` variable. Adjust `prerequisites_env` as necessary for the proxy requirements of your network.

Install Contrail Insights

To install Contrail Insights:

1. Install Ansible on the Contrail Insights Platform node. Ansible will install docker and docker-py on the platform.

```
#Ubuntu
apt-get install python-pip python-dev                #Installs Pip
pip install ansible==2.3.0                          #Installs Ansible 2.3
sudo apt-get install build-essential libssl-dev libffi-dev #Dependencies
pip install markupsafe httpplib2 requests           #Dependencies

#RHEL/CentOS
yum install epel-release                            #Enable EPEL repository
In case the above command does not work, manually download and install the epel-release
package with one of the below commands, depending on your system's version.
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

yum groupinstall 'Development Tools'                 #Install development tools
yum install openssl-devel libffi libffi-devel        #Dependencies
yum install python-pip python-devel                 #Install Pip
pip install ansible==2.3.0                          #Install Ansible 2.3
pip install markupsafe httpplib2 requests           #Dependencies
```

NOTE: For RHEL, the following IPtables rule is needed to access port 9000.

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

2. Install libvirt on the compute nodes using the following command:

```
yum -y install libvirt
```

This installs the KVM (Kernel-based Virtual Machine) and associated packages for collecting data from virtual machines running on the compute nodes.

3. On the compute nodes where Contrail Insights Agent runs, verify that python virtualenv is installed.

```
#Ubuntu
apt-get install -y python-pip
pip install virtualenv

#RHEL/CentOS
yum install -y python-pip
pip install virtualenv
```

4. Enable passwordless login to facilitate Contrail Insights Platform node with Ansible to install agents on the nodes. Create a SSH public key on the node where Ansible playbooks are run, and then copy the key to the appformix_controller node.

```
ssh-keygen -t rsa #Creates Keys

ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host> #Copies key from the node to other hosts
```

5. Use the Sample_Inventory file as a template to create a host file.

```
# Example naming schemes are as below:
# hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
172.16.70.5
172.16.70.17

# appformix_controller host
#
# Host variables can be defined to control Contrail Insights configuration parameters
# for particular host. For example, to specify the directory in which MongoDB
# data is stored on hostname1 (the default is /opt/appformix/mongo/data):
#
# hostname1 appformix_mongo_data_dir=/var/lib/appformix/mongo
#
# For variables with same value for all appformix_controller hosts, set group
# variables below.
#
[appformix_controller]
```

```
172.16.70.119

[openstack_controller]
172.16.70.120
```

6. Verify that all the hosts listed in the inventory file are reachable from the Contrail Insights Platform.

```
export ANSIBLE_HOST_KEY_CHECKING=False # Eliminates interactive experience prompting for
Known_Hosts

ansible -i inventory -m ping all          # Pings all the hosts in the inventory file
```

7. At the top-level of the distribution, create a directory named `group_vars`.

```
mkdir group_vars
```

8. Download the Contrail Insights installation packages from [software downloads](#) to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-openstack-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

9. In `group_vars` directory, create a file named `all`. Add the following:

```
openstack_platform_enabled: true

appformix_manager_version: <version>
appformix_docker_images:
  - /path/to/contrail-insights-platform-images-<version>.tar.gz
  - /path/to/contrail-insights-dependencies-images-<version>.tar.gz
  - /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

10. Contrail Insights must be configured to communicate with the OpenStack cluster. The Ansible playbooks use OpenStack environment variables to configure Contrail Insights with details of the OpenStack environment.

OS_AUTH_URL	Keystone URL (e.g., <code>https://host:5000/</code>)
OS_ENDPOINT_TYPE	endpoint type to communicate with service (default: <code>publicURL</code>)
OS_USERNAME	admin account to be used by Contrail Insights
OS_PASSWORD	password for admin account
OS_PROJECT_NAME	admin project created for Contrail Insights account
OS_PROJECT_DOMAIN_NAME	domain for admin project
OS_USER_DOMAIN_NAME	domain for admin user
OS_DOMAIN_NAME	(optional) use domain-scoped token for admin account

11. Source the `openrc` file that contains the (step 10) environment variables and ensure the variables are in the environment of the shell from which the Ansible-playbooks are going to be executed. Then, install Contrail Insights by executing the `appformix_openstack.yml` playbook. Specify the path to the inventory directory that you created earlier.
12. Open the Contrail Insights Dashboard in a Web browser. For example:

```
http://<contrail-insights-platform-node-ip>:9000
```

Select **Skip Installation** because the initial configuration was performed by Ansible using the OpenStack environment variables in step 10. Log in to Contrail Insights Dashboard using OpenStack Keystone credentials.

Additional Ansible Configuration

To set up additional Ansible configurations:

1. To install in a Keystone SSL-enabled cluster, include the following variables in the `group_vars/all` file:

```
appformix_keystone_ssl_ca: '/path/to/keystone_ca.crt'
```

Contrail Insights Ansible will distribute this Keystone CA to all of the Contrail Insights Platform nodes and ask Contrail Insights components to talk to Keystone using this CA file with SSL enabled.

NOTE: Deprecation Notice: The `appformix_mongo_cache_size_gb` parameter previously available starting in Contrail Insights 2.19.5 is now deprecated and no longer supported from Contrail Insights 3.2.0 and going forward. Starting with Contrail Insights version 3.2.0, Mongo will be configured to use a maximum of 40 percent of the available memory on the Contrail Insights Platform nodes.

2. To enable network device monitoring in the cluster, include the following in the `group_vars/all` file:

```
# For enabling pre-requisites for packdge installation.
appformix_install_snmp_dependencies: true
appformix_install_jti_dependencies: true
# For running the appformix-network-device-adapter
network_device_discovery_enabled: true
appformix_plugins: '{{ appformix_network_device_factory_plugins }}'
# After 3.1, SNMP Traps can be enabled also so appformix_plugins can be specified as below:
# appformix_plugins: '{{ appformix_network_device_factory_plugins }} +
{{ appformix_snmp_trap_factory_plugins }}'
```

3. To install Contrail Insights certified plug-ins on the cluster, include the following variables in the `group_vars/all` file:

```
appformix_plugins: <list of certified plugins to be installed>
appformix_openstack_log_plugins: <list of OpenStack log plugins to be installed>
```

For example:

```
appformix_plugins:
- { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
- { plugin_info: 'certified_plugins/contrail_vrouter.json' }
- { plugin_info: 'certified_plugins/zookeeper_usage.json' }
- { plugin_info: 'certified_plugins/heavy_hitters.json' }
```

```

appformix_openstack_log_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json',
      log_file_path: '/var/log/cinder/cinder-api.log' }
  - { plugin_info: 'certified_plugins/glance_logparser.json',
      log_file_path: '/var/log/glance/glance-api.log' }
  - { plugin_info: 'certified_plugins/keystone_logparser.json',
      log_file_path: '/var/log/apache2/keystone_access.log,/var/log/httpd/
keystone_wsgi_admin_access.log,/var/log/keystone/keystone.log' }

```

For a list of all Contrail Insights certified plug-ins that can be installed, look for the entries starting with `plugin_info` in the file `roles/appformix_defaults/defaults/main.yml`.

The OpenStack log parser plug-ins parse the API log files of each OpenStack service to collect metrics about API calls and response status codes. To install these plug-ins, add them to the variable `appformix_openstack_log_plugins` in `group_vars/all`, as shown above. Each plug-in entry in this list requires a parameter called `log_file_path` to be specified. This parameter should be set to the complete path to the service's API log file on the OpenStack Controller node(s). Multiple comma-separated paths may be specified.

To identify the right log file to be specified in `log_file_path`, look for entries like the following, containing a client IP address, REST call type, and response status code:

```

2019-04-02 06:50:13.103 3465 INFO nova.osapi_compute.wsgi.server [req-d07e953a-6921-4224-a056-
afb6ff69adde 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -]
172.18.0.6 "GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/os-hypervisors/detail HTTP/1.1" status:
200 len: 1427 time: 0.0208740
2019-04-02 06:50:13.183 3465 INFO nova.osapi_compute.wsgi.server [req-34b2f686-9eb5-4112-b3fc-
e0b37798a302 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -]
172.18.0.6 "GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/servers/detail?
all_tenants=1&status=SHELVED_OFFLOADED HTTP/1.1" status: 200 len: 211 time: 0.0754580

```

Default locations for these files are listed in the variable `appformix_openstack_log_factory_plugins` in `roles/appformix_defaults/defaults/main.yml`.

4. In Contrail Insights version 2.19.8, a timeout value can be configured for connecting to OpenStack services. The default value of this timeout is 10 seconds and can be changed to a value between 5 and 20 seconds (both inclusive). To change the value, add the following variable to `group_vars/all`:

```

appformix_openstack_session_timeout: <number of seconds>

```

To modify the timeout value after the Contrail Insights Platform has been installed, add the variable to the `group_vars/all` file as described above and re-run the Contrail Insights installation playbook. Restart the `appformix-openstack-adapter` Docker container after the playbook has completed:

```
docker restart appformix-openstack-adapter
```

Upgrade Notices

NOTE: In Contrail Insights version 3.2.0, support for discovering OpenStack Octavia Load Balancer services is added. Contrail Insights only collects load balancer state information, such as `provisioning_status` and `operating_status`, as well as flavor information. To enable this service discovery, provide Octavia service's endpoint as variable `appformix_octavia_endpoint_url` in the `group_vars/all` file. For example:

```
appformix_octavia_endpoint_url: http://10.1.1.1:9876
```

Chargeback costs can also be configured for the Octavia Load Balancer services. See [Configure Load Balancer Costs](#).

Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

NOTE: In Contrail Insights version 3.0, the variable `appformix_openstack_factory_plugins` is deprecated. All OpenStack log parser plug-ins have to be specified in the variable `appformix_openstack_log_plugins`. When upgrading from an older version to version 3.0, make sure to move all OpenStack log parser plug-ins defined in `appformix_openstack_factory_plugins` or `appformix_plugins` to `appformix_openstack_log_plugins`. Also, in Contrail Insights version 3.0, all entries in this list have to be specified with a `log_file_path` value, as described in step 3 above.

When upgrading from version 2.18.x to version 3.0, make the following changes in the `group_vars/all` file:

In version 2.18.x:

```

appformix_openstack_factory_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json', log_file_path: '/var/log/
cinder/cinder-api.log'}
  - { plugin_info: 'certified_plugins/glance_logparser.json', log_file_path: '/var/log/glance/
api.log'}
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }
  - { plugin_info: 'certified_plugins/keystone_logparser.json', log_file_path: '/var/log/
keystone/keystone.log'}
  - { plugin_info: 'certified_plugins/neutron_logparser.json', log_file_path: '/var/log/neutron/
server.log'}
  - { plugin_info: 'certified_plugins/nova_logparser.json', log_file_path: '/var/log/nova/nova-
api.log'}

appformix_plugins: [{ appformix_openstack_factory_plugins }] + ...

```

In version 3.0.x:

```

appformix_plugins:
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }

appformix_openstack_log_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json', log_file_path: '/var/log/
cinder/cinder-api.log'}
  - { plugin_info: 'certified_plugins/glance_logparser.json', log_file_path: '/var/log/glance/
api.log'}
  - { plugin_info: 'certified_plugins/keystone_logparser.json', log_file_path: '/var/log/
keystone/keystone.log'}
  - { plugin_info: 'certified_plugins/neutron_logparser.json', log_file_path: '/var/log/neutron/
server.log'}
  - { plugin_info: 'certified_plugins/nova_logparser.json', log_file_path: '/var/log/nova/nova-
api.log'}

```

Source the `openrc` file from the OpenStack controller node (`/etc/contrail/openstackrc`) to the Contrail Insights Platform node to authenticate the adapter to access administrator privileges over the controller services. The file should look like the following:

```

export OS_USERNAME=admin
export OS_PASSWORD=<admin-password>
export OS_AUTH_URL=http://172.16.80.2:5000/v2.0/

```



```
export OS_NO_CACHE=1
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

Remove a Node from Contrail Insights

To remove a node from Contrail Insights:

1. Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from Contrail Insights.

```
# Example naming schemes are as below:
#  hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
172.16.70.5 appformix_state=absent
172.16.70.17
```

2. Run Ansible with the edited inventory file. This will remove the node and all its resources from Contrail Insights.

```
ansible-playbook -i inventory appformix_openstack.yml
```

Upgrade Contrail Insights for an OpenStack Cluster

Contrail Insights can be easily upgraded by running the `appformix_openstack.yml` playbook of the new release. Follow the same procedure as the installation.

Uninstall Contrail Insights from an OpenStack Cluster

To uninstall Contrail Insights and destroy all data, execute the following command:

```
ansible-playbook -i <inventory_file> clean_appformix_openstack.yml
```

RELATED DOCUMENTATION

[Contrail Insights Agent Requirements](#) | 5

Contrail Insights Installation for Ubuntu Focal

Contrail Insights Release 3.3.5 supports Ubuntu 20.04 (Focal).

Software Requirements

- `docker-ce : 5:19.03.9~3-0~ubuntu-focal`
- Python 2 is not installed by default with Ubuntu 20.04 (Focal).

Starting in Contrail Insights Release 3.3.11, if Python version 3 is installed, the minimum required Ansible version is 2.5.1.

Follow these steps before you install Contrail Insights.

1. Install `python` and `python-pip` on the Contrail Insights Controller nodes, and on the host(s) that the Contrail Insights Agent runs on.

```
sudo apt-get install -y python python3-pip  
sudo apt install python-is-python3
```

2. For Contrail Insights Release 3.3.11 and later, install Jinja2 version 3.0.3.

```
pip3 install jinja2==3.0.3
```

3. In the `groups_vars/all` file, set the following to true.

```
appformix_ansible_python3_interpreter_enabled: true
```

4. Run the iptables rule to access port 9000.

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

NOTE: Ignore any errors that may arise if `IN_public_allow` does not exist.

After you have completed these steps, you can install Contrail Insights.

- For more information on Contrail Insights Agent, see ["Contrail Insights Agent Requirements" on page 5](#)
- For more information on Contrail Insights Installation for OpenStack, see ["Contrail Insights Installation and Configuration for OpenStack" on page 31](#).

Release History Table

Release	Description
3.3.5	Contrail Insights Release 3.3.5 supports Ubuntu 20.04 (Focal).

Contrail Insights Installation for Containerized OpenStack (OpenStack Kolla, Red Hat OpenStack Platform 13)

IN THIS SECTION

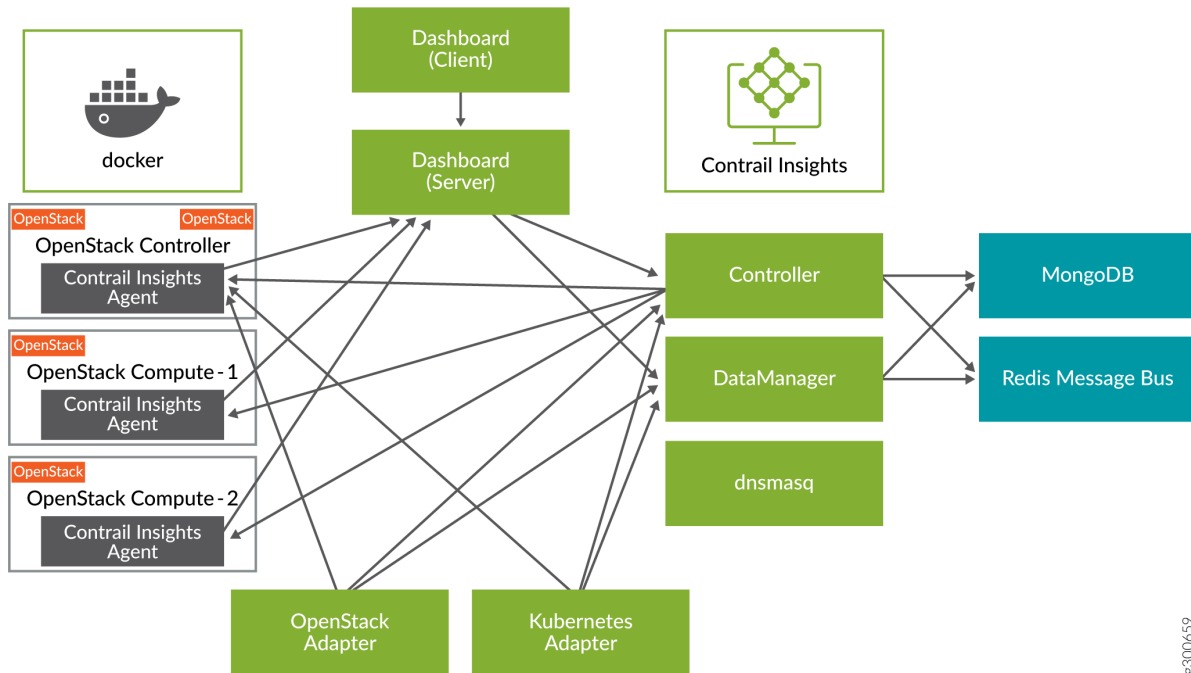
- [Architecture | 48](#)
- [Requirements | 48](#)
- [Connectivity | 49](#)

- Workflow | 49
- Initial Setup | 50
- Prerequisites for Agent Installation | 51
- Configure OpenStack | 52
- Install Contrail Insights | 52
- Optional Configuration | 55
- Remove a Node from Contrail Insights | 57

On a containerized OpenStack setup, Contrail Insights monitors OpenStack resources as well as the containers in which the OpenStack services are deployed. This topic provides the steps to deploy Contrail Insights when OpenStack services are running inside Docker containers, without a container orchestration engine. For a setup in which OpenStack is running in containers orchestrated by Kubernetes (OpenStack Helm), see "[Contrail Insights Installation for OpenStack Helm](#)" on page 62.

Architecture

Figure 9: Contrail Insights and Containerized OpenStack (OpenStack Kolla, Red Hat OpenStack Platform 13)



- Contrail Insights Agent monitors resource usage on OpenStack compute nodes.
- Contrail Insights Platform offers REST APIs to configure the system.
- Contrail Insights DataManager stores data from multiple Agents.
- Contrail Insights Dashboard provides a Web-based user interface.
- OpenStack Adapter discovers virtual machines (VMs) running on the OpenStack cluster.
- Contrail Insights Agent discovers Docker containers on which OpenStack services are running.

Requirements

- For each host, on which Contrail Insights Platform is installed, see "[Contrail Insights General Requirements](#)" on page 2 for hardware and software requirements. For a list of Contrail Insights Agent supported platforms, see "[Contrail Insights Agent Requirements](#)" on page 5.

- API access to OpenStack services. Contrail Insights reads information about the OpenStack cluster through the APIs exposed by the various OpenStack services. The user credentials provided during configuration must provide sufficient permission for read-only API calls. Further, Contrail Insights Platform must be able to open a connection to the host and port on which the API server runs.
- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:
Target customers or use:
Cluster type: Containerized OpenStack
Number of hosts:
Number of instances:
```

Connectivity

- One IP address for the Platform host.
- Platform host must have IP connectivity to reach compute nodes.
- Platform host requires Internet connectivity during installation.
- Control plane services are distributed as Docker images that are retrieved from a registry hosted by Contrail Insights. Optionally, the Docker images may be downloaded and served from a local registry.
- Dashboard client (in browser) must have IP connectivity to Platform host.

Workflow

The installation includes the following workflow:

1. Initial setup.
2. Prerequisites for Agent installation.

3. Configuring OpenStack.
4. Contrail Insights Installation.
5. Optional Configuration.

Initial Setup

To complete the initial setup:

1. Install Ansible and other required packages on the `appformix_controller` node:

```
#Ubuntu
apt-get update
apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
pip install ansible==2.3.0.0 markupsafe httpplib2
```

```
#RHEL/CentOS
yum install epel-release                                #Enable EPEL repository
In case the above command does not work, manually download and install the epel-release
package with one of the below commands, depending on your system's version.
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

yum groupinstall 'Development Tools'                    #Install development tools
yum install openssl-devel libffi libffi-devel           #Dependencies
yum install python-pip python-devel                     #Install Pip
pip install ansible==2.3.0                              #Install Ansible 2.3
pip install markupsafe httpplib2                        #Dependencies
```

NOTE: For RHEL, the following iptables rule is needed to access port 9000:

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

2. Install libvirt on the compute nodes using the following command:

```
yum -y install libvirt
```

This installs the KVM (Kernel-based Virtual Machine) and associated packages for collecting data from virtual machines running on the compute nodes.

3. Install Python `virtualenv` on the vRouter compute nodes where Contrail Insights Agent runs:

```
#Ubuntu
apt-get install -y python-pip
pip install virtualenv
```

```
#RHEL/CentOS
yum install -y python-pip
pip install virtualenv
```

4. Set up passwordless SSH between the Contrail Insights Platform node and the OpenStack cluster nodes. Run the following commands to generate and copy the SSH public keys to all the nodes:

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub <user>@<openstack-node-1>
...
ssh-copy-id -i ~/.ssh/id_rsa.pub <user>@<appformix-platform-node>
```

Prerequisites for Agent Installation

Contrail Insights uses the `libvirt` library to collect data from virtual machines running on an OpenStack compute node. Run the following command on all OpenStack computes to install the library:

```
On Ubuntu hosts:
sudo apt-get install libvirt0:amd64
On Centos hosts:
sudo yum install libvirt-client
```


Configure OpenStack

Contrail Insights reads information about all projects configured in your OpenStack cluster. The software requires administrator privilege. We recommend you create a new administrator account for Contrail Insights. If you do not create a new administrator account, then you must provide the username and password of an existing administrator account during the configuration of Contrail Insights.

To create an administrator account for Contrail Insights, perform the following steps in the OpenStack Horizon dashboard:

1. Create a user account and name it `appformix`.
2. Select a new project for the user account.
3. Select role as `admin`.

Install Contrail Insights

To install Contrail Insights:

1. Download the Contrail Insights installation packages from [software downloads](#) to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz  
contrail-insights-dependencies-images-<version>.tar.gz  
contrail-insights-openstack-images-<version>.tar.gz  
contrail-insights-platform-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Unzip `contrail-insights-<version>.tar.gz`. This package contains all the Ansible playbooks required to install Contrail Insights.

```
tar -xvzf contrail-insights-<version>.tar.gz  
cd contrail-insights-<version>/
```

NOTE: The remaining steps should be executed from within the `contrail-insights-<version>/` directory. Although the product name changed from "AppFormix" to "Contrail Insights," the UI and internal command paths continue to show AppFormix and will reflect the new name at a later date.

3. Using `sample_inventory` as a template, create an inventory file for the installation. List the OpenStack compute nodes in the `compute` section and the Contrail Insights Platform node in the `appformix_controller` section. List the node running the OpenStack controller services in the `openstack_controller` section.

```
# List all Compute Nodes
[compute]
<openstack-compute-node1-ip-or-hostname>
<openstack-compute-node2-ip-or-hostname>
...
# appformix_controller host
[appformix_controller]
<appformix-controller-node-ip-or-hostname>
[openstack_controller]
<openstack-controller-node-ip-or-hostname>
```

4. Ensure that all of the hosts mentioned in the inventory file are reachable from the Contrail Insights Platform.

```
export ANSIBLE_HOST_KEY_CHECKING=False
ansible -i inventory -m ping all
```

5. Create a directory called `group_vars`. Create a file named `all` inside this directory with configuration variables required by Contrail Insights.

```
mkdir group_vars
vi group_vars/all
appformix_docker_images:
- /path/to/contrail-insights-platform-images-<version>.tar.gz
- /path/to/contrail-insights-dependencies-images-<version>.tar.gz
- /path/to/contrail-insights-openstack-images-<version>.tar.gz
- /path/to/contrail-insights-network_device-images-<version>.tar.gz
appformix_dns_version: 2
```

```

openstack_platform_enabled: true
appformix_docker_container_discovery: true
openstack_deployment_mode: containerized_openstack

```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```

appformix_license: path/to/<contrail-insights-license-file>.sig

```

To enable network device monitoring in the cluster, include the following in the `group_vars/all` file:

```

# For enabling prerequisites for package installation.
appformix_install_snmp_dependencies: true
appformix_install_jti_dependencies: true
# For running the appformix-network-device-adapter
network_device_discovery_enabled: true
appformix_plugins: '{{ appformix_network_device_factory_plugins }}'
# After 3.1, SNMP Traps can be enabled also so appformix_plugins can be specified as below:
# appformix_plugins: '{{ appformix_network_device_factory_plugins }} +
{{ appformix_snmp_trap_factory_plugins }}'

```

6. Source the `openrc` file from the OpenStack controller node (`/etc/contrail/openstackrc`) to the Contrail Insights Platform node where the playbooks are going to be executed. The file should look like the following:

```

$ cat openrc
export OS_USERNAME=admin
export OS_PASSWORD=<admin-password>
export OS_AUTH_URL=http://172.16.80.2:5000/v2.0/
export OS_NO_CACHE=1
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_NAME=admin
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
$ source openrc

```

7. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

The playbook should run to completion without any errors.

8. Log in to the Contrail Insights Dashboard:

```
http://<contrail-insights-platform-node-ip>:9000
```

In the Auth Service dialog box, two options are provided:

OpenStack Log in with OpenStack credentials.

AppFormix Log in with Contrail Insights credentials available at `/opt/appformix/etc/appformix_token.rst` on the Contrail Insights Platform node.

Optional Configuration

To install Contrail Insights certified plug-ins on the cluster, include the following variables in the `group_vars/all` file:

```
appformix_plugins: <list of certified plugins to be installed>
appformix_openstack_log_plugins: <list of OpenStack log plugins to be installed>
```

Example:

```
appformix_plugins:
  - { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
  - { plugin_info: 'certified_plugins/contrail_vrouter.json' }
  - { plugin_info: 'certified_plugins/zookeeper_usage.json' }
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }
appformix_openstack_log_plugins:
  - { plugin_info: 'certified_plugins/cinder_api_logparser.json',
      log_file_path: '/var/log/cinder/cinder-api.log' }
  - { plugin_info: 'certified_plugins/glance_logparser.json',
      log_file_path: '/var/log/glance/glance-api.log' }
  - { plugin_info: 'certified_plugins/keystone_logparser.json',
```

```
log_file_path: '/var/log/apache2/keystone_access.log,/var/log/httpd/
keystone_wsgi_admin_access.log,/var/log/keystone/keystone.log' }
```

For a list of all Contrail Insights certified plug-ins that can be installed, look for the entries starting with `plugin_info` in the file `roles/appformix_defaults/defaults/main.yml`.

The OpenStack log parser plug-ins parse the API log files of each OpenStack service to collect metrics about API calls and response status codes. To install these plug-ins, add them to the variable `appformix_openstack_log_plugins` in `group_vars/all`, as shown in the example above. Each plug-in entry in this list requires a parameter called `log_file_path` to be specified. This parameter should be set to the complete path to the service's API log file on the OpenStack Controller node(s). Multiple comma-separated paths may be specified.

To identify the right log file to be specified in `log_file_path`, look for entries like the following, containing a client IP address, REST call type, and response status code:

```
2019-04-02 06:50:13.103 3465 INFO nova.osapi_compute.wsgi.server [req-d07e953a-6921-4224-a056-
afb6ff69adde 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/os-hypervisors/detail HTTP/1.1" status: 200 len: 1427
time: 0.0208740
2019-04-02 06:50:13.183 3465 INFO nova.osapi_compute.wsgi.server [req-34b2f686-9eb5-4112-b3fc-
e0b37798a302 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/servers/detail?all_tenants=1&status=SHELVED_OFFLOADED
HTTP/1.1" status: 200 len: 211 time: 0.0754580
```

Default locations for these files are listed in the variable `appformix_openstack_log_factory_plugins` in `roles/appformix_defaults/defaults/main.yml`.

In containerized OpenStack environments, log files are generated inside the containers running the OpenStack services. However, they have to be available on the OpenStack controller host for the Contrail Insights plug-ins to be able to read them. The path specified in `log_file_path` should be the location of the file on the OpenStack Controller host.

NOTE: In Contrail Insights version 3.0, all OpenStack log parser plug-ins have to be specified in the variable `appformix_openstack_log_plugins`. When upgrading from an older version to 3.0, make sure to move all OpenStack log parser plug-ins defined under `appformix_plugins` to `appformix_openstack_log_plugins`. Also, in Contrail Insights 3.0, all entries in this list have to be specified with a `log_file_path` value, as described above.

Remove a Node from Contrail Insights

To remove a node from Contrail Insights:

1. Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from Contrail Insights.

```
# Example naming schemes are as below:
#  hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
172.16.70.5 appformix_state=absent
172.16.70.17
```

2. Run Ansible with the edited inventory file.

```
ansible-playbook -i inventory appformix_openstack.yml
```

This removes the node and all its resources from Contrail Insights.

RELATED DOCUMENTATION

[Contrail Insights Installation and Configuration for OpenStack | 31](#)

[Contrail Insights Installation for OpenStack in HA | 57](#)

[Contrail Insights Installation for OpenStack Helm | 62](#)

Contrail Insights Installation for OpenStack in HA

IN THIS SECTION

- [HA Design Overview | 58](#)
- [Requirements | 58](#)

HA Design Overview

Contrail Insights Platform can be deployed to multiple hosts for high availability (HA). Platform services continue to communicate using an API proxy that listens on a virtual IP address. Only one host will have the virtual IP at a time, and so only one API proxy will be the “active” API proxy at a time.

The API proxy is implemented by HAProxy. HAProxy is configured to use services in active-standby or load-balanced active-active mode, depending on the service.

At most, one host will be assigned the virtual IP at any given time. This host is considered the “active” HAProxy. The virtual IP address is assigned to a host by keepalived, which uses VRRP protocol for election.

Services are replicated in different modes of operation. In the “active-passive” mode, HAProxy sends all requests to a single “active” instance of a service. If the service fails, then HAProxy will select a new “active” from the other hosts, and begin to send requests to the new “active” service. In the “active-active” mode, HAProxy load balances requests across hosts on which a service is operational.

Contrail Insights Platform can be deployed in a 3-node, 5-node, or 7-node configuration for high availability.

Requirements

- For each host, on which Contrail Insights Platform is installed, see ["Contrail Insights General Requirements" on page 2](#) for hardware and software requirements. For a list of Contrail Insights Agent supported platforms, see ["Contrail Insights Agent Requirements" on page 5](#).

- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:
Target customers or use:
Cluster type: OpenStack
Number of hosts:
Number of instances:
```

Connectivity

- One virtual IP address to be shared among all the Platform Hosts. This IP address should not be used by any host before installation. It should have reachability from all the Platform Hosts after installation.
- Dashboard client (in browser) must have IP connectivity to the virtual IP.
- IP addresses for each Platform Host for installation and for services running on these hosts to communicate.
- `keepalived_vrrp_interface` for each Platform Host which would be used for assigning virtual IP address. Details on how to configure this interface is described in the `sample_inventory` section.

Install Contrail Insights for High Availability

To install Contrail Insights to multiple hosts for high availability:

1. Download the Contrail Insights installation packages from [software downloads](#) to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-openstack-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
contrail-insights-network_device-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Install Ansible on the installer node. Ansible will install docker and the docker Python package on the appformix_controller.

```
# sudo apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
# sudo pip install ansible==2.7.6 markupsafe httpplib2
```

For Ansible 2.3:

```
# sudo pip install ansible==2.3 markupsafe httpplib2 cryptography==1.5
```

3. Install Python and python-pip on all the Platform hosts so that Ansible can run between the installer node and the appformix_controller node.

```
# sudo apt-get install -y python python-pip
```

4. Install python pip package on the hosts where Contrail Insights Agents run.

```
# apt-get install -y python-pip
```

5. To enable passwordless login to all Platform hosts by Ansible, create an SSH public key on the node where Ansible playbooks are run and then copy the key to all the Platform hosts.

```
# ssh-keygen -t rsa                                #Creates Keys
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_1>.....#Copies key from the node to
all platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_2>.....#Copies key from the node to
all platform hosts
# ssh-copy-id -i ~/.ssh/id_rsa.pub <platform_host_3>.....#Copies key from the node to
all platform hosts
```

6. Use the sample_inventory file as a template to create a host file. Add all the Platform hosts and compute hosts details.

```
# List all compute hosts which needs to be monitored by Contrail Insights
[compute]
203.0.113.5
203.0.113.17
```

```
# Contrail Insights controller hosts
[appformix_controller]
203.0.113.119 keepalived_vrrp_interface=eth0
203.0.113.120 keepalived_vrrp_interface=eth0
203.0.113.121 keepalived_vrrp_interface=eth0
```

NOTE: Note: In the case of 5-node or 7-node deployment, list all the nodes under `appformix_controller`.

7. At top-level of the distribution, create a directory named `group_vars` and then create a file named `all` inside this directory.

```
# mkdir group_vars
# touch group_vars/all
```

Add the following entries to the newly created `all` file:

```
appformix_vip: <ip-address>
appformix_docker_images:
- /path/to/contrail-insights-platform-images-<version>.tar.gz
- /path/to/contrail-insights-dependencies-images-<version>.tar.gz
- /path/to/contrail-insights-openstack-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

8. Copy and source the `openrc` file from the OpenStack controller node (`/etc/contrail/openrc`) to the `appformix_controller` to authenticate the adapter to access admin privileges over the controller services.

```
root@installer_node:~# cat /etc/contrail/openrc
export OS_USERNAME=<admin user>
export OS_PASSWORD=<password>
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://<openstack-auth-URL>/v2.0/
```

```
export OS_NO_CACHE=1
root@installer_node:~# source /etc/contrail/openrc
```

NOTE: In Contrail Insights version 3.2.0, support for discovering OpenStack Octavia Load Balancer services is added. Contrail Insights only collects load balancer state information, such as provisioning_status and operating_status, as well as flavor information. To enable this service discovery, provide Octavia service's endpoint as variable `appformix_octavia_endpoint_url` in the `group_vars/all` file. For example:

```
appformix_octavia_endpoint_url: http://10.1.1.1:9876
```

Chargeback costs can also be configured for the Octavia Load Balancer services. See [Configure Load Balancer Costs](#).

9. Run Ansible with the created inventory file.

```
ansible-playbook -i inventory appformix_openstack_ha.yml
```

10. If running the playbooks as root user then this step can be skipped. As a non-root user (for example, "ubuntu"), the user "ubuntu" needs access to the docker user group. The following command adds the user to the docker group.

```
sudo usermod -aG docker ubuntu
```

Contrail Insights Installation for OpenStack Helm

IN THIS SECTION

- [Architecture and Terminology | 63](#)
- [Requirements | 64](#)
- [Workflow | 65](#)
- [Initial Setup | 65](#)

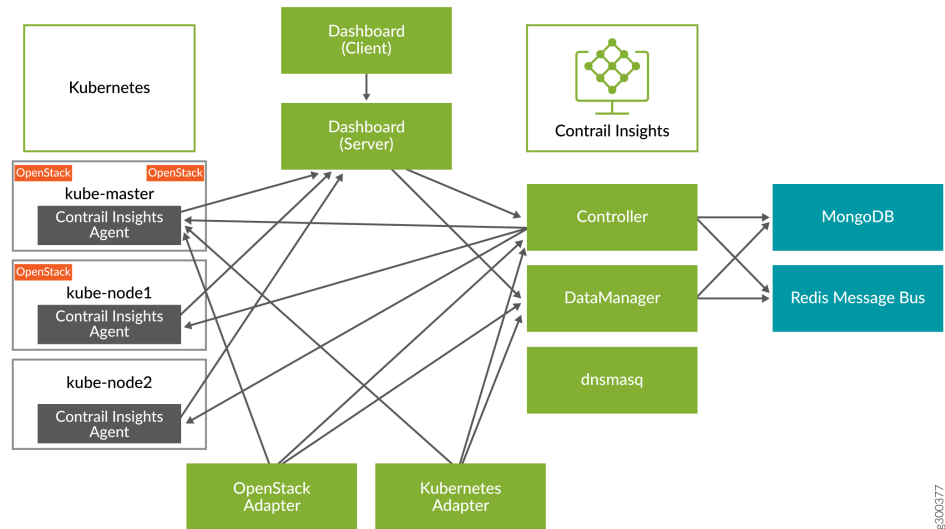
- [Prerequisites for Agent Installation | 67](#)
- [Configure Kubernetes | 67](#)
- [Configure OpenStack Administrator Account for Contrail Insights | 69](#)
- [Install Contrail Insights | 69](#)
- [Optional Configuration | 72](#)
- [Remove a Node from Contrail Insights | 73](#)

OpenStack Helm is a project that provides Helm charts for deploying OpenStack services on a Kubernetes cluster. Contrail Insights can be deployed to monitor both the Kubernetes resources as well as the OpenStack resources from a single Dashboard.

Architecture and Terminology

Kubernetes cluster nodes	Primary and worker nodes of the Kubernetes cluster being monitored by Contrail Insights. These nodes will run the Contrail Insights Agent.
OpenStack cluster nodes	Kubernetes nodes from the cluster above that are running OpenStack services (Keystone, Nova, Neutron, and so on) in containers.
Contrail Insights Platform node	Node on which Contrail Insights Platform components will be installed. Should be able to reach the Kubernetes cluster nodes.

Figure 10: Contrail Insights and OpenStack Helm Workflow



g300377

Requirements

The following are the requirements for installing Contrail Insights for OpenStack Helm.

- Supported Kubernetes versions: 1.8.x and later
- See "[Contrail Insights General Requirements](#)" on page 2 for hardware and software requirements.
- API access to Kubernetes API server. Contrail Insights reads information about the Kubernetes cluster from the API server. The token provided during configuration must provide sufficient permission for read-only API calls. In addition, Contrail Insights Platform must be able to open a connection to the host and port on which the API server runs.
- API access to OpenStack services. Contrail Insights reads information about the OpenStack cluster through the APIs exposed by the various OpenStack services. The user credentials provided during configuration must provide sufficient permission for read-only API calls. In addition, Contrail Insights Platform must be able to open a connection to the host and port on which the API server runs.

- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

You can obtain a license key from <mailto:APPFORMIX-KEY-REQUEST@juniper.net>. Provide the following information in your request:

```
Group name:  
Target customers or use:  
Cluster type: OpenStack on Kubernetes  
Number of hosts:  
Number of instances:
```

Workflow

The installation consists of the following steps:

1. Initial setup.
2. Prerequisites for Agent installation.
3. Configuring Kubernetes.
4. Configuring OpenStack.
5. Installing Contrail Insights.
6. Optional configuration.

Initial Setup

Run the following commands for initial setup:

1. Install these required packages on the Contrail Insights Platform node.

```
#Ubuntu  
apt-get update
```

```
apt-get install python-pip python-dev build-essential libssl-dev libffi-dev
pip install ansible==2.3.0.0 markupsafe httpplib2
```

```
#RHEL/CentOS
yum install epel-release                                #Enable EPEL repository
In case the above command does not work, manually download and install the epel-release
package with one of the below commands, depending on your system's version.
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

yum groupinstall 'Development Tools'                    #Install development tools
yum install openssl-devel libffi libffi-devel           #Dependencies
yum install python-pip python-devel                    #Install Pip
pip install ansible==2.3.0                             #Install Ansible 2.3
pip install markupsafe httpplib2                       #Dependencies
```

NOTE: For RHEL, the following iptables rule is needed to access port 9000.

```
iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

2. Edit the `/etc/hosts` file on the Contrail Insights Platform node and enter the IP addresses of the OpenStack on Kubernetes cluster nodes.

```
vi /etc/hosts
<kube-master-ip> k8s-master
<kube-worker1-ip> k8s-node1
<kube-worker2-ip> k8s-node2
```

3. Set up passwordless SSH between the Contrail Insights Platform node and the OpenStack on Kubernetes cluster nodes. Run the following commands to generate and copy the SSH public keys to all the nodes.

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-master
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-node1
ssh-copy-id -i ~/.ssh/id_rsa.pub root@k8s-node2
ssh-copy-id -i ~/.ssh/id_rsa.pub root@<IP of Contrail Insights Platform node>
```

Prerequisites for Agent Installation

Contrail Insights uses the `libvirt` library to collect data from virtual machines running on an OpenStack compute node. Run the following command on all OpenStack computes to install the library:

```
On Ubuntu hosts:
sudo apt-get install libvirt0:amd64
On Centos hosts:
sudo yum install libvirt-client
```

Configure Kubernetes

Contrail Insights reads information about resources in your Kubernetes clusters. The software requires the `cluster-admin` role or another role that gives it read-only access to all objects in the cluster. We recommend that you create a new Service Account for Contrail Insights and assign it the `cluster-admin` role. If you do not create a new Service Account, then you must provide the token from an existing Service Account that has the required access during the configuration of Contrail Insights.

To create a new Service Account with the required access for Contrail Insights, perform the following steps in the OpenStack on Kubernetes cluster primary node:

1. Create a YAML file with the following:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: appformix
```

2. Create the `appformix` Service Account using the file created in Step 1.

```
$ kubectl create -f <file>.yaml
```

3. Confirm that the Service Account has been created. Make a note of its namespace.

```
$ kubectl describe serviceaccount appformix
Name:          appformix
Namespace:    default
```



```
Labels:      <none>
Annotations: <none>
Image pull secrets: <none>
Mountable secrets: appformix-token-pkljk
Tokens:      appformix-token-pkljk
Events:      <none>
```

4. Add the cluster-admin role to the appformix Service Account as follows, substituting *namespace* for the namespace noted in Step 3.

```
$ kubectl create clusterrolebinding appformix-binding --clusterrole=cluster-admin --
serviceaccount=<namespace>:appformix
```

5. Run the following command to confirm that the appformix Service Account has the required access:

```
$ kubectl auth can-i get nodes --as=system:serviceaccount:<namespace>:appformix --all-
namespaces
```

The output of the command should be yes.

6. Contrail Insights must be configured to communicate with the Kubernetes cluster. Get the following details from the Kubernetes cluster to use during the Contrail Insights installation.

kubernetes_cluster_url This is the URL of the Kubernetes API Server. To get this value, run the following command on the OpenStack on Kubernetes cluster:

```
$ kubectl cluster-info | grep 'Kubernetes master'
Kubernetes master is running at https://172.24.1.173:6443
```

kubernetes_auth_token This is the authentication token of the appformix Service Account. To get this value, run the following commands on the OpenStack on Kubernetes cluster:

```
$ kubectl describe serviceaccount appformix
Name:      appformix
Namespace: default
Labels:    <none>
Annotations: <none>
Tokens:    appformix-token-pkljk
```

```
[...]
$ kubectl describe secret appformix-token-pkljk
Name:          appformix-token-pkljk
Namespace:     default
[...]
token:         eyJwb[...]
```

Configure OpenStack Administrator Account for Contrail Insights

Contrail Insights reads information about all projects configured in your OpenStack cluster. The software requires administrator privilege. We recommend you create a new administrator account for Contrail Insights. If you do not create a new administrator account, then you must provide the username and password of an existing administrator account during the configuration of Contrail Insights.

To create an administrator account for Contrail Insights, perform the following steps in the OpenStack Horizon dashboard:

1. Create a user account and name it **appformix**.
2. Select a new project for the user account.
3. Select role as **admin**.

Install Contrail Insights

To install Contrail Insights:

1. Download the Contrail Insights installation packages from [software downloads](#) to the Contrail Insights Platform node. Get the following files:

```
contrail-insights-<version>.tar.gz
contrail-insights-dependencies-images-<version>.tar.gz
contrail-insights-kubernetes-images-<version>.tar.gz
contrail-insights-openstack-images-<version>.tar.gz
contrail-insights-platform-images-<version>.tar.gz
```

If you are installing a version earlier than 3.2.6, copy the Contrail Insights license file to the Contrail Insights Platform node.

2. Unzip `contrail-insights-<version>.tar.gz`. This package contains all the Ansible playbooks required to install Contrail Insights.

```
tar -xvzf contrail-insights-<version>.tar.gz
cd contrail-insights-<version>/
```

NOTE: The remaining steps should be executed from within the `contrail-insights-<version>/` directory. Although the product name changed from "AppFormix" to "Contrail Insights," the UI and internal command paths continue to show AppFormix and will reflect the new name at a later date.

3. Using `sample_inventory` as a template, create an inventory file for the installation. List the Kubernetes cluster nodes in the `compute` section and the Contrail Insights Platform node in the `appformix_controller` section. List the node running the OpenStack controller services in the `openstack_controller` section.

```
cp sample_inventory inventory
vi inventory
[compute]
k8s-master
k8s-node1
k8s-node2
[appformix_controller]
<IP of the Contrail Insights Platform node>
[openstack_controller]
<IP of the Kubernetes node(s) running OpenStack controller services>
```

4. Create a directory called `group_vars`. Create a file named `all` inside this directory with configuration variables required by Contrail Insights.

```
mkdir group_vars
vi group_vars/all
appformix_docker_images:
- /path/to/contrail-insights-platform-images-<version>.tar.gz
- /path/to/contrail-insights-dependencies-images-<version>.tar.gz
- /path/to/contrail-insights-kubernetes-images-<version>.tar.gz
- /path/to/contrail-insights-openstack-images-<version>.tar.gz
appformix_dns_version: 2
kubernetes_platform_enabled: True
```

```

openstack_platform_enabled: True
kubernetes_cluster_url: <URL from Configuring Kubernetes step 4 above>
kubernetes_auth_token: <token from Configuring Kubernetes step 4 above>

```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```

appformix_license: path/to/<contrail-insights-license-file>.sig

```

5. Source the `openrc` file from the OpenStack Controller node in the environment in which the Ansible playbooks are going to be executed. The file should look like the following:

```

$ cat openrc
export OS_CLOUD=openstack_helm
export OS_USERNAME='admin'
export OS_PASSWORD='password'
export OS_PROJECT_NAME='admin'
export OS_PROJECT_DOMAIN_NAME='default'
export OS_USER_DOMAIN_NAME='default'
export OS_AUTH_URL='http://keystone.openstack.svc.cluster.local/v3'
$ source openrc

```

6. Run the Ansible playbook.

```

ansible-playbook -i inventory appformix_openstack_on_kubernetes.yml

```

Playbook should run to completion without any errors.

7. Log into the Contrail Insights Dashboard at:

```

http://<contrail-insights-platform-node-ip>:9000

```

In the Auth Service dropdown list, three options are provided:

OpenStack Log in with OpenStack credentials and view only OpenStack resources.

Kubernetes Log in with Contrail Insights credentials and view only Kubernetes resources.

AppFormix Log in with Contrail Insights credentials and view both OpenStack and Kubernetes resources.

For the last two options, the token to use for login is available at `/opt/appformix/etc/appformix_token.rst` on the Contrail Insights Platform node.

Optional Configuration

To install Contrail Insights certified plug-ins on the cluster, include the following variables in the `group_vars/all` file:

```
appformix_plugins: <list of certified plugins to be installed>
appformix_openstack_log_plugins: <list of OpenStack log plugins to be installed>
```

For example:

```
appformix_plugins:
- { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
- { plugin_info: 'certified_plugins/contrail_vrouter.json' }
- { plugin_info: 'certified_plugins/zookeeper_usage.json' }
- { plugin_info: 'certified_plugins/heavy_hitters.json' }
appformix_openstack_log_plugins:
- { plugin_info: 'certified_plugins/cinder_api_logparser.json',
  log_file_path: '/var/log/cinder/cinder-api.log' }
- { plugin_info: 'certified_plugins/glance_logparser.json',
  log_file_path: '/var/log/glance/glance-api.log' }
- { plugin_info: 'certified_plugins/keystone_logparser.json',
  log_file_path: '/var/log/apache2/keystone_access.log,/var/log/httpd/
keystone_wsgi_admin_access.log,/var/log/keystone/keystone.log' }
```

For a list of all Contrail Insights certified plug-ins that can be installed, look for the entries starting with `plugin_info` in the file `roles/appformix_defaults/defaults/main.yml`.

The OpenStack log parser plug-ins parse the API log files of each OpenStack service to collect metrics about API calls and response status codes. To install these plug-ins, add them to the variable `appformix_openstack_log_plugins` in `group_vars/all`, as shown above. Each plug-in entry in this list requires a parameter called `log_file_path` to be specified. This parameter should be set to the complete path to the service's API log file on the OpenStack Controller node(s). Multiple comma-separated paths can be specified.

To identify the correct log file to be specified in `log_file_path`, look for entries like the following, containing a client IP address, REST call type, and response status code:

```
2019-04-02 06:50:13.103 3465 INFO nova.osapi_compute.wsgi.server [req-d07e953a-6921-4224-a056-afb6ff69adde 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - ] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/os-hypervisors/detail HTTP/1.1" status: 200 len: 1427
time: 0.0208740
2019-04-02 06:50:13.183 3465 INFO nova.osapi_compute.wsgi.server [req-34b2f686-9eb5-4112-b3fc-e0b37798a302 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - ] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/servers/detail?all_tenants=1&status=SHELVED_OFFLOADED
HTTP/1.1" status: 200 len: 211 time: 0.0754580
```

Default locations for these files are listed in the variable `appformix_openstack_log_factory_plugins` in `roles/appformix_defaults/defaults/main.yml`.

On containerized OpenStack environments, log files are generated inside the containers running the OpenStack services. However, they have to be available on the OpenStack controller host for the Contrail Insights plug-ins to be able to read them. The path specified in `log_file_path` should be the location of the file on the OpenStack Controller host.

NOTE: In Contrail Insights 3.0, all OpenStack log parser plug-ins have to be specified in the variable `appformix_openstack_log_plugins`. When upgrading from an earlier version to 3.0, make sure to move all OpenStack log parser plug-ins defined in `appformix_plugins` to `appformix_openstack_log_plugins`. Also, in Contrail Insights 3.0, all entries in this list have to be specified with a `log_file_path` value, as described in example above.

Remove a Node from Contrail Insights

Edit the inventory file and add `appformix_state=absent` to each node that you want to remove from Contrail Insights.

```
# Example naming schemes are as below:
#   hostname ansible_ssh_user='username' ansible_sudo_pass='password'

# List all Compute Nodes
[compute]
```

```
172.16.70.5 appformix_state=absent
172.16.70.17
```

Run Ansible with the edited inventory file.

```
ansible-playbook -i inventory appformix_openstack_on_kubernetes.yml
```

This removes the node and all its resources from Contrail Insights.

Contrail Insights Installation for Standalone

IN THIS SECTION

- [Requirements | 74](#)
- [Installation | 75](#)
- [Upgrade Contrail Insights for Standalone | 80](#)
- [Uninstall Contrail Insights from Standalone | 80](#)

Requirements

Contrail Insights can monitor hosts and network devices in a standalone cluster without a management platform (for example, OpenStack) from which to discover entities. This topic explains how to install a standalone version of Contrail Insights.

- For hardware and software requirements, see ["Contrail Insights General Requirements" on page 2](#).

- **NOTE:** Upgrade notice: Starting with Contrail Insights 3.2.6, the requirement for a license file is removed. If you are installing a version earlier than 3.2.6, a license is required prior to installation.

In case the above command does not work, manually download and install the epel-release package with one of the below commands, depending on your system's version.

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm
```

```
# sudo yum groupinstall 'Development Tools'           #Install development tools
# sudo yum install python-pip python-devel gcc gcc-c++ make openssl-devel libffi-devel
# sudo yum install openssl-devel libffi libffi-devel    #Dependencies
# sudo yum install python-pip python-devel             #Install Pip
# sudo pip install ansible==2.8.9                     #Install Ansible 2.8.9
# sudo pip install markupsafe httpplib2 requests       #Dependencies
```

NOTE: For RHEL, the following IPtables rule is needed to access port 9000:

```
# sudo iptables -t filter -A IN_public_allow -p tcp --dport 9000 -j ACCEPT
```

3. Install Python and python-pip on the `appformix_controller` so that Ansible can run between the installer node and `appformix_controller` node.

Ubuntu

```
# sudo apt-get install -y python python-pip
```

RHEL/CentOS

```
# sudo yum install -y python python-pip
```

4. Install the python-pip package on the hosts where Contrail Insights Agents run.

Ubuntu

```
# apt-get install -y python-pip
```

RHEL/CentOS

```
# yum install -y python-pip
```

5. To enable passwordless login to the Contrail Insights Platform by Ansible, create an SSH public key on the node where Ansible playbooks are run and then copy the key to `appformix_controller`.

```
$ ssh-keygen -t rsa                                #Creates Keys
$ ssh-copy-id -i ~/.ssh/id_rsa.pub <target_host>    #Copies key from the node to
appformix_controller node
```

Alternately, you can use `ansible_connection=local` as shown in the following inventory example:

```
[appformix_controller]
172.xx.xx.119 ansible_connection=local
```

6. Create an Ansible inventory by creating a directory named `inventory` and in that directory edit the `inventory/hosts` file. The following is an example:

```
# List all hosts which needs to be monitored by Contrail Insights
[bare_host]
172.xx.xx.5
172.xx.xx.17
# Contrail Insights Platform host
[appformix_controller]
172.xx.xx.119
# List of all hosts which will do network telemetry collection. You can use same hosts from
other tags such [bare_host], [appformix_controller]
[appformix_network_agents]
172.xx.xx.119
172.xx.xx.5
172.xx.xx.6
```

The host listed under group `appformix_controller` is mandatory. This is the host onto which Contrail Insights Platform will be installed. Optionally, you can monitor additional hosts by listing them in the `bare_host` section. All the hosts listed under `appformix_network_agents` are used as collectors to collect network device telemetry. Workload is distributed evenly between hosts listed under `appformix_network_agents`. As you add more network devices to Contrail Insights, you might need to add more hosts to do the telemetry collecting, which can be done by adding hosts to `appformix_network_agents` aggregate from the UI after installation.

7. In the inventory directory, create a subdirectory named `inventory/group_vars` and then create a file `inventory/groups_vars/all` inside this subdirectory:

```
$ mkdir inventory/group_vars
$ touch inventory/group_vars/all
```

Add the following configuration parameters to the `all` file:

```
appformix_docker_images:
  - path/to/contrail-insights-platform-images-<version>.tar.gz
  - path/to/contrail-insights-dependencies-images-<version>.tar.gz
  - path/to/contrail-insights-network_device-images-<version>.tar.gz
appformix_kvm_instance_discovery: true
# For enabling prerequisites for package installation.
appformix_network_device_monitoring_enabled: true
# For running the appformix-network-device-adapter
network_device_discovery_enabled: true
appformix_plugins: '{{ appformix_network_device_factory_plugins }}'
# After 3.1, SNMP Traps can be enabled also so appformix_plugins can be specified as
below:
# appformix_plugins: '{{ appformix_network_device_factory_plugins }} +
{{ appformix_snmp_trap_factory_plugins }}'
```

If you are installing a version earlier than 3.2.6, include the path to the Contrail Insights license file in `group_vars/all`:

```
appformix_license: path/to/<contrail-insights-license-file>.sig
```

NOTE: `appformix_docker_images` is mandatory. Also, be sure to add all required Contrail Insights SNMP, JTI Native, and gRPC network device monitoring plug-ins to the `groups_vars/all` file under variable `appformix_plugins`. There are more than 20 built-in plug-ins available in the `certified_plugins/` directory. The example above shows how you can enable all built-in network device plug-ins.

There are two ways to add network devices. One method should be chosen to add devices for monitoring.

- ["Configure Network Device from JSON File" on page 106](#)

- [Configure Network Devices from the UI](#)

8. Unpack the Contrail Insights TAR file and change to the created directory:

```
$ tar xzf contrail-insights-<version>.tar.gz
$ cd contrail-insights-<version>
```

NOTE: The remaining steps should be executed from within the `contrail-insights-<version>/` directory. Although the product name changed from "AppFormix" to "Contrail Insights," the UI and internal command paths continue to show AppFormix and will reflect the new name at a later date.

9. Run Ansible using the path to the `inventory` directory created in Step 6 and Step 7. For example:

```
$ ansible-playbook -i ../inventory appformix_standalone.yml
```

10. If running the playbooks as root user, you can skip this step. As a non-root user, for example "ubuntu" then user "ubuntu" needs access to the `docker` user group. Run the following command to add the user to the `docker` group:

```
sudo usermod -aG docker ubuntu
```

11. To locate the Contrail Insights token for Auth Service, look in the following file on the Contrail Insights Platform host.

```
$ cat /opt/appformix/etc/appformix_token.rst
```

NOTE: Deprecation Notice: The `appformix_mongo_cache_size_gb` parameter previously available starting in Contrail Insights 2.19.5 is now deprecated and no longer supported from Contrail Insights 3.2.0 and going forward. Starting with Contrail Insights Release 3.2.0, Mongo will be configured to use a maximum of 40 percent of the available memory on the Contrail Insights Platform nodes.

Upgrade Contrail Insights for Standalone

Contrail Insights can be easily upgraded by running the `appformix_standalone.yml` playbook of the new release. Follow the same procedure as for the installation.

Uninstall Contrail Insights from Standalone

If you want to uninstall Contrail Insights and destroy all data, execute the following command:

```
ansible-playbook -i <inventory_file> clean_appformix_standalone.yml
```

SEE ALSO

[Custom SNMP Plug-Ins | 134](#)

No Link Title

Contrail Insights Settings

IN THIS SECTION

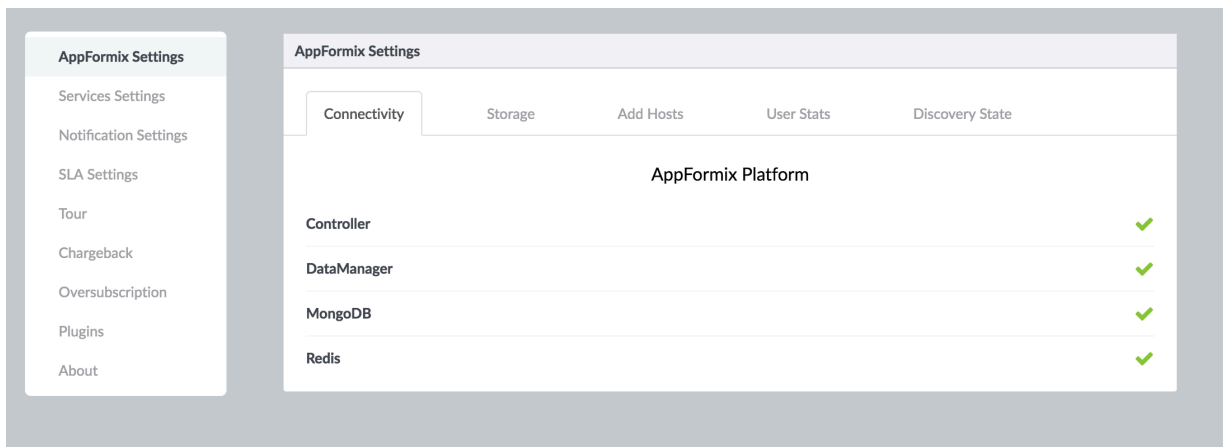
- [Contrail Insights Settings | 81](#)
- [Services Settings | 81](#)
- [Notifications Settings | 81](#)
- [SLA Settings | 82](#)
- [Chargeback Settings | 82](#)
- [Oversubscription Settings | 82](#)
- [Plug-In Settings | 83](#)

Settings for Contrail Insights can be accessed from the top right corner. From the Settings page, you can set any of the following configurations.

Contrail Insights Settings

The Settings page shows the connectivity of the Contrail Insights components. You can also view the current storage size of their cluster. You can add more hosts from this page, check user statistics about their cluster, and look at the Contrail Insights discovery status of your cluster. [Figure 11 on page 81](#) shows the Contrail Insights settings and component connectivity.

Figure 11: AppFormix Settings and Component Connectivity



Services Settings

The Services Settings page is where you can configure the different services that you want Contrail Insights to monitor. For example, you can configure RabbitMQ or MySQL. See [Service Monitoring from the UI](#) for more details about configuring these services.

Notifications Settings

The Notification Settings page is where you can configure your notification services. For more information on configuring notifications, see [Notifications](#).

SLA Settings

The SLA settings page is where you can set up your SLA policies. For more information on configuring these policies, see [Health Monitor](#).

Chargeback Settings

The Chargeback settings page is where you can configure the cost for usage of different resources. See [Chargeback](#).

Oversubscription Settings

The Oversubscription settings page is where you can set the oversubscription ratios for disk, memory, and vCPUs. Oversubscription ratios are used by the Capacity Planning feature. See [Capacity Planning](#). [Figure 12 on page 82](#) shows the oversubscription resource and ratio settings.

Figure 12: Oversubscription Resource and Ratio Settings

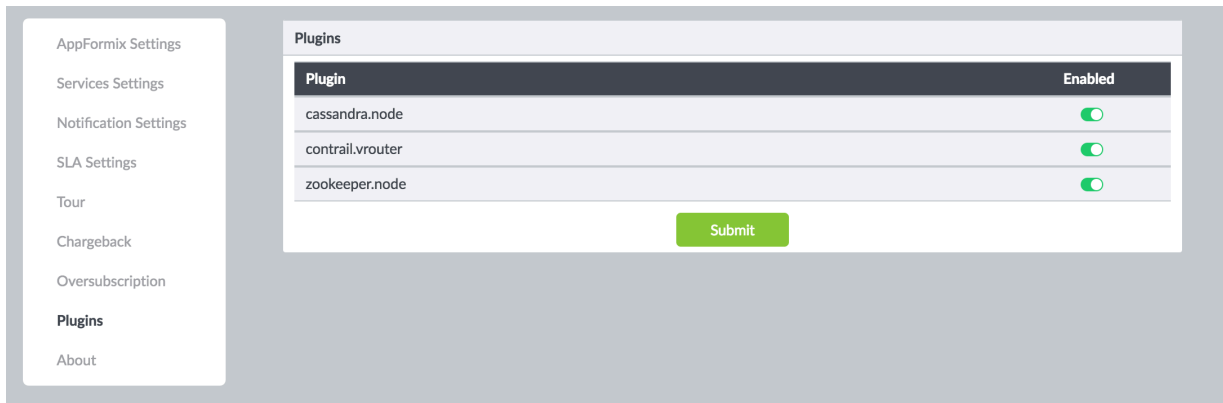
Resource	Oversubscription Ratio
Disk	1
Memory	1
Vcpus	1

Apply

Plug-In Settings

From the Plugins page, you can enable or disable plug-ins, as shown in [Figure 13 on page 83](#).

Figure 13: Plug-In Settings Page



3

CHAPTER

Ansible Configuration

Bare Host	86
Contrail Insights MultiCluster Mode	87
Contrail Insights MultiCluster Proxy	89
Contrail Insights Port List	96
Contrail Insights Role-Based Access	102
Configure Network Device from JSON File	106
Contrail Insights Plug-Ins	115
Contrail Insights User-Defined Plug-Ins	117
Instance Scope Plug-Ins	121
Contrail Insights Object Plug-In	128
Custom SNMP Plug-Ins	134
Custom Sensors for JTI, gRPC, and NETCONF	137
Remote Hosts	141
Monitor NFX250 with Contrail Insights Agent	144
Contrail Insights SDKs	150
Contrail Insights with SSL (HTTPS) Enabled	153
Service Monitoring Ansible Variables	153
OpenStack Services Monitoring Using Service Group Profiles	160
Ansible Configuration Variables	164

Bare Host

IN THIS SECTION

- [Configure a Bare Host | 86](#)

A *bare host* is a host that is not discovered from a cloud management system, such as OpenStack. Any Linux host can be monitored by Contrail Insights (see "[Contrail Insights Agent Requirements](#)" on page 5). A bare host can be configured using the Contrail Insights Ansible playbooks.

The Ansible playbook will install Contrail Insights Agent on the host and configure the host in the Contrail Insights Platform. The bare host will be displayed in the Contrail Insights Dashboard, where charts and alarms can be used to monitor metrics from the host.

Configure a Bare Host

To configure a bare host, add an entry for the bare host in the inventory file following the [bare_host] group tag. For example, the Ansible inventory will look something like this:

```
[bare_host]
hostname1
hostname2 ansible_ssh_user='user' ansible_ssh_pass='password'
```

RELATED DOCUMENTATION

[Contrail Insights Role-Based Access | 102](#)

[Remote Hosts | 141](#)

Contrail Insights MultiCluster Mode

IN THIS SECTION

- [Add Contrail Insights Clusters to Contrail Insights Primary Cluster](#) | 87

Contrail Insights supports a MultiCluster View, meaning that multiple Contrail Insights clusters can be added to one Contrail Insights cluster so all clusters are accessible through that single Contrail Insights cluster.

NOTE: For MultiCluster Mode to be available, the Contrail Insights Dashboard client (Web browser) needs to be able to reach all of the Contrail Insights Clusters. If the other clusters are not reachable from the Dashboard, then the Contrail Insights MultiCluster Proxy mode is needed. Refer to "[Contrail Insights MultiCluster Proxy](#)" on page 89.

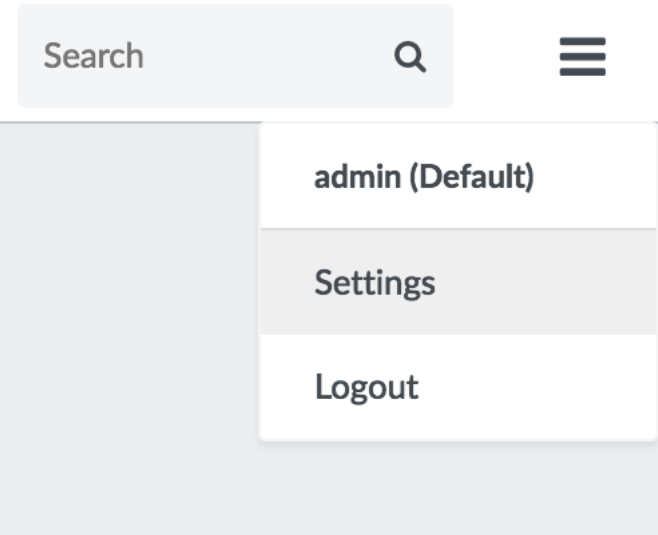
Add Contrail Insights Clusters to Contrail Insights Primary Cluster

For the following procedure, Cluster1, Cluster2, and Cluster3 are used as examples. Cluster1 is the primary cluster and therefore has MultiCluster View enabled.

To add clusters to the primary cluster:

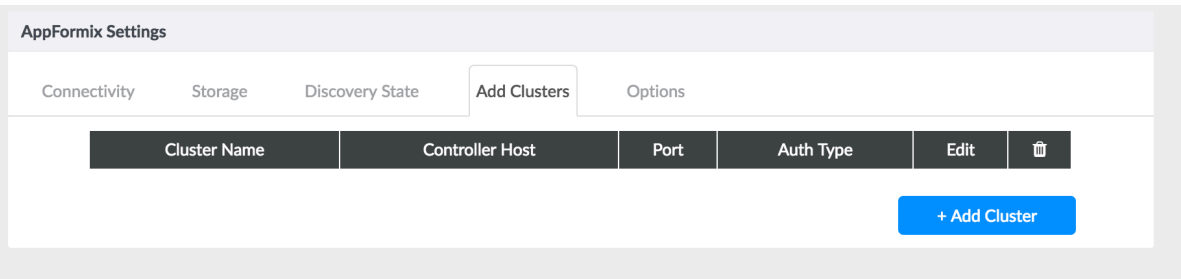
1. Log in to the Contrail Insights Dashboard using your administrator credentials.
2. Select **Settings** from the list in the top right of the Dashboard.

Figure 14: Select Settings



3. Select the **Add Clusters** tab, then click **Add Cluster**.

Figure 15: Add Cluster



4. Complete the following fields:

- | | |
|------------------------------|--|
| Cluster Name | Name for the cluster. |
| Controller Host | The IP/DNS name of the cluster being added. This IP must be reachable from the primary cluster and the browser. |
| Controller Port | The port on which the cluster's dashboard is running. Default is 9000. |
| Username and Password | Credentials for the user that you want the cluster to login with. When you click this added cluster in the future, you will be viewing the cluster with this user's credentials. |

AuthType This specifies the type of the cluster. OpenStack, vCenter, Standalone/Token are currently supported.

5. After completing these fields, click **Setup** and then the cluster is added. You can see the cluster by clicking the **Clusters** tab in the side panel.
6. Complete the Add Cluster steps for Cluster1, Cluster2, and Cluster3. You only have to add clusters to the primary cluster itself (Cluster1).

Contrail Insights MultiCluster Proxy

IN THIS SECTION

- [Overview | 89](#)
- [Requirements | 90](#)
- [Ansible | 90](#)
- [Overview of v2 API URLs | 93](#)
- [Make GET Query with Contrail Insights MultiCluster Proxy | 93](#)
- [Testing the MultiCluster Proxy | 94](#)

Overview

With the MultiCluster Proxy feature, the primary Contrail Insights Platform is capable of redirecting requests through a proxy container so that requests to different clusters in the MultiCluster environment can be made through the primary cluster.

The use case for this MultiCluster Proxy mode is when not all clusters in the MultiCluster environment can be accessed by the user directly. This mode can be enabled so that all requests go through a proxy container exposed on the primary `appformix_controller` node. This proxy container will then reroute the request to the appropriate cluster and deliver back the response to the client.

The default MultiCluster View does not support this proxy behavior and can only be used in situations where the client/browser have access to all clusters in the MultiCluster environment. MultiCluster Proxy is a second mode of MultiCluster that allows this limited connectivity to work where the client only needs access to the primary cluster.

Requirements

- Contrail Insights installed on all of the clusters.
- Deploy Primary Contrail Insights Platform node(s) with a MultiClusterProxy enabled license.
- Primary Contrail Insights Platform node(s) must have connectivity to all of the clusters that are a part of the MultiCluster environment.
- If the primary cluster is a high availability (HA) setup then port 27000 must be open on all of the Platform nodes.

Ansible

NOTE: Configuring MultiClusterProxy on an HA setup through Ansible is deprecated starting with AppFormix version 3.2. Use the Contrail Insights Dashboard to configure clusters by selecting **Settings > AppFormix Settings > Add Clusters**.

Ansible Playbooks

`appformix_proxy_add_cluster.yml`

Playbook to add cluster to the MultiCluster Proxy enabled Contrail Insights.

- Gets the `appformix_master_auth` token to POST to `/cluster_definition`. This role takes the setup from the `inventory/group_vars/all` for the new cluster that is being added. See “MultiCluster Proxy Deployment Workflow.”
- Executes `appformix_docker_images` role to load haproxy container for the first time.
- Runs scripts to add required entries into `haproxy.cfg` for each cluster that is added.

NOTE: If the primary cluster is an HA setup then use the playbook `appformix_proxy_add_cluster_ha.yml`.

`appformix_proxy_delete_cluster.yml`

Playbook to delete cluster from the MultiCluster Proxy enabled Contrail Insights.

- Does DELETE to `/cluster_definition` using `master_auth_token`.
- Runs scripts to remove entries from `haproxy.cfg` for each cluster that is removed.
- When primary cluster is removed, `haproxy` container is also deleted.

NOTE: If the primary cluster is an HA setup then use the playbook `appformix_proxy_delete_cluster_ha.yml`.

Ansible Roles

`appformix_multiclustert_config`

Role to add or remove clusters from `appformix_controller`.

- Does POST/DELETE to `/cluster_definition` end point depending on whether the action is add or delete.
- The `cluster_username` and `cluster_password` can not be set concurrently with `cluster_token`. If the license for a cluster supports authentication with username and password, then `cluster_username` and `cluster_password` can be set.
- If `cluster_username` and `cluster_password` are set, set `cluster_token` to empty string (`""`).
- If `cluster_token` is set, set both `cluster_username` and `cluster_password` to empty string (`""`).

The `appformix_multiclustert_config` requires the following variables be defined:

- `cluster_name`
- `cluster_controller_host`
- `cluster_controller_port`
- `cluster_auth_type`
- `cluster_username`

- cluster_password
- cluster_token
- is_master_cluster

Example:

```
cluster_name: <name of cluster>
cluster_controller_host: <hostname or ip>
cluster_controller_port: 9000 #(unless configured)
cluster_auth_type: appformix
cluster_token: <appformix_token> #(can be found in controller_node: #/opt/appformix/etc/
appformix_token.rst)
is_master_cluster: <True/False> #(True for adding master node only)
```

MultiCluster Proxy Deployment Workflow

Deploying the MultiCluster Proxy feature consists of the following steps:

1. Install Contrail Insights on all the clusters.
 - a. We recommend installing the same Contrail Insights version on all clusters.
 - b. Primary cluster is required to be deployed with a MultiClusterProxy enabled software license.
2. In inventory/group_vars/all, provide the information for primary node first. Refer to ["Ansible Roles" on page 91](#) for the sample inventory entry. Confirm that is_master_cluster is set to **True**.
3. Run the playbook appformix_proxy_add_cluster.yml to add this cluster as primary cluster. Refer to ["Ansible Playbooks" on page 90](#).
4. For adding other clusters, make an inventory entry as shown in ["Ansible Roles" on page 91](#) (with is_master_cluster set to **False**). After each entry, run the playbook appformix_proxy_add_cluster.yml to add this cluster. Refer to ["Ansible Playbooks" on page 90](#).
5. Open the Dashboard for the MultiCluster setup in a Web browser. For example:

http://<master-cluster-IP>:18081

NOTE: Note that after AppFormix release 3.2.0, in HA setups, the Dashboard should be accessed on port 9000 and clusters should be added through the UI.

To delete a cluster:

1. Keep the inventory entry of the cluster that needs to be deleted, having the other clusters' inventory entry commented out.
2. Run the playbook `appformix_proxy_delete_cluster.yml` to delete the cluster. See ["Ansible Playbooks" on page 90](#).

NOTE: Removing the primary cluster, removes the proxy container as well, if the primary cluster is a single node installation.

Use the playbooks `appformix_proxy_add_cluster_ha.yml` or `appformix_proxy_delete_cluster_ha.yml` if the primary cluster is an HA setup.

Overview of v2 API URLs

GET <code>/appformix/analytics/v2.0/cluster_stats</code>	Returns cluster statistics for the cluster.
GET <code>/appformix/controller/v2.0/clusters</code>	Returns a list of all clusters added to the Contrail Insights primary cluster.
GET <code>/appformix/controller/v2.0/clusters/<cluster_id></code>	Returns information of a cluster added to Contrail Insights primary cluster with <code>cluster_id</code> .

Make GET Query with Contrail Insights MultiCluster Proxy

Call `GET /appformix/controller/v2.0/clusters` to identify which cluster you would like to query. Take note of the `ClusterId`.

Make the GET queries by entering the following commands:

URL format:

```
http://<appformix_controller_primary_proxy_node>:18081/clusters/cluster_Id/
<any_appformix_v2_endpoint>
```

Then run the following command to get the response with appropriate headers:

```
curl -X GET -H 'Content-Type: application/json' -H 'X-Auth-Type: appformix' -H 'X-Auth-Token: appformix_master_auth_token_for_cluster1' http://<master-cluster-IP>:18081/cluster/<cluster_Id>/appformix/controller/v2.0/<any_v2_end_point>
```

NOTE: Above port 18081 should be replaced with 9000 in HA deployments starting with AppFormix release 3.2.

Example:

To get hosts from a cluster with ID b50adf04-4c07-11e9-b016-0242ac120005, whose Contrail Insights master_auth_token is d90f9d18-4c07-11e9-9545-0242ac120005, with a MultiCluster primary IP as 10.1.1.2, run the following:

```
curl -X GET -H 'Content-Type: application/json' -H 'X-Auth-Type: appformix' -H 'X-Auth-Token: d90f9d18-4c07-11e9-9545-0242ac120005' http://10.1.1.2:18081/cluster/b50adf04-4c07-11e9-b016-0242ac120005/appformix/controller/v2.0/hosts
```

NOTE: Verify that you use the correct Contrail Insights token for corresponding clusters. The Contrail Insights token can be found at /opt/appformix/etc/appformix_token.rst in the Contrail Insights Platform node of that cluster.

Testing the MultiCluster Proxy

To test the MultiClusterProxy feature:

1. Select **Settings > API Documentation > Link to AppFormix Documentation**.
2. Use the following API:

```
GET /appformix/controller/v2.0/cluster_haproxy_output/{cluster_id}
```

3. Provide the following arguments:

Argument	Description
id	Id of the cluster.
X-Auth-Type	Authentication Type for Contrail Insights of cluster with id.
X-Auth-Token	Authentication Token for Contrail Insights of cluster with id.
component	API component.
endpoint	GET Endpoint to forward.

id	<p>The cluster id can be obtained by these commands:</p> <ul style="list-style-type: none"> GET /appformix/controller/v2.0/clusters Returns a list of all clusters added to the Contrail Insights primary cluster. GET /appformix/controller/v2.0/clusters/<cluster_id> Returns information of a cluster added to Contrail Insights primary cluster with cluster_id.
X-Auth-Type, X-Auth-Token	Authentication type and token of Contrail Insights Platform on the cluster to forward to.
component	Either controller or analytics, depending on the endpoint you want to forward to.
endpoint	Any available v2 GET endpoint.

NOTE: For endpoints that require query or string arguments, for example controller/data/metrics, you need to construct the argument in the endpoint with standard format:

<endpoint>?arg1=val1&arg2=val2 ...

RELATED DOCUMENTATION

Contrail Insights MultiCluster Mode 87
No Link Title

Contrail Insights Port List

IN THIS SECTION

- [Platform Port List for Single Node | 96](#)
- [Platform and Agent Common Ports | 98](#)
- [Platform Port List for HA Setup | 98](#)
- [Platform Port List for Proxy in HA Setup | 100](#)
- [Communication between Platform Host and Services | 100](#)

Platform Port List for Single Node

Table 1: Platform Port List for Single Node

Entity	Port
Controller	7000
Datamanager	8090
Dashboard	9000
OpenstackAdapter	7500

Table 1: Platform Port List for Single Node (Continued)

Entity	Port
KubernetesAdapter	7700
NetworkDeviceAdapter	7650
Redis	6379
Mongo	27017

Ports with SSL Enabled**Table 2: Platform Port List for Single Node with SSL Enabled**

Entity	Port
Controller	7001
Datamanager	8091
Dashboard	9001
OpenstackAdapter	7501
KubernetesAdapter	7701
NetworkDeviceAdapter	7651
Redis	6379
Mongo	27017

Platform and Agent Common Ports

Communication from Outside (Browser Client) to Platform

IN port 9000

IN port 9001 (When SSL is enabled.)

Communication between Platform Nodes and Agents on Compute Nodes

OUT port 42595

IN port 9000

Communication between Agents and Platform Nodes on Compute Nodes

IN port 42595

OUT port 9000

Communication between Network Devices and Agents on Platform Aggregate

IN port 42596

OUT port 830 (for JTI), 161 (for SNMP)

IN port 42597 (SNMP Traps)

Platform Port List for HA Setup

HA Setup

Table 3: Communication between Platform Nodes for HA Setup

Entity	Port(s)
Controller	27000
Datamanager	28090

Table 3: Communication between Platform Nodes for HA Setup (Continued)

Entity	Port(s)
Dashboard	29000
OpenstackAdapter	27500
KubernetesAdapter	27700
NetworkDeviceAdapter	27650
Redis	26379
Mongo	27017
Haproxy	9000, 6379, 7777, 1936

With SSL Enabled**Table 4: Platform Nodes for HA Setup with SSL Enabled**

Entity	Port(s)
Controller	27001
Datamanager	28091
Dashboard	29001
OpenstackAdapter	27501
KubernetesAdapter	27701

Table 4: Platform Nodes for HA Setup with SSL Enabled *(Continued)*

Entity	Port(s)
NetworkDeviceAdapter	27651
Redis	26379
Mongo	27017
Haproxy	9001, 6379, 7777, 1936

Platform Port List for Proxy in HA Setup

Table 5: Port List for Proxy in HA Setup

Entity	Port(s)
AppFormixMultiClusterHaproxy	18081
Haproxy	9000 (9001 when SSL is enabled), 6379, 7777, 1936, 1935, 1937

Communication between Platform Host and Services

Table 6: Communication Port between Platform Host and Services

OUT port 9696 (neutron)
OUT port 35357 (keystone)
OUT port 5000 (keystone)

OUT port 8888 (swift)

OUT port 8004 (heat)

OUT port 8776 (cinderv2)

OUT port 8774 (nova)

OUT port 9292 (glance)

OUT port 8081 (contrail-analytics)

OUT port 8082 (contrail-config)

OUT port 15672 (rabbitmq)

OUT port 3306 (mysql)

OUT port 9042 (cassandra)

OUT port 80 (scaleio)

OUT port 6789 (ceph-monitor)

OUT port 6800-6804 (ceph-osd)

OUT port 9042, 7000, 7199 (cassandra plug-in, jmx ports)

RELATED DOCUMENTATION

| [Contrail Insights with SSL \(HTTPS\) Enabled](#) | 153

Contrail Insights Role-Based Access

IN THIS SECTION

- [Configure Roles | 103](#)
- [Granting Contrail Insights Permissions to Read-Only OpenStack Users | 104](#)

Contrail Insights has role-based access for system configuration and data visibility. Contrail Insights integrates with Keystone for identity. The roles by which a user is a member of projects determine that user's capabilities in Contrail Insights.

Contrail Insights has three categories of user permissions. Each category is configured with a set of Keystone roles that are permitted to access the functionality in that category. If a user is a member of a project with a role that matches one of the roles configured for a category, then the user will gain access to the functionality associated with that user category.

Table 7: Contrail Insights User Permissions Categories

Category	Description
Contrail Insights Administrator	<p>Users in this category can configure system settings in Contrail Insights, such as:</p> <ul style="list-style-type: none">• Data retention policy.• Host and instance SLA policies.• Chargeback rate cards and departments. <p>Users can also view and operate all entities in an environment, as described in the following Infrastructure View category.</p>

Table 7: Contrail Insights User Permissions Categories *(Continued)*

Category	Description
Infrastructure View	<p>Users in this category can view and operate all entities in an environment: hosts, projects, instances, aggregates, and infrastructure services. Users can perform the following actions:</p> <ul style="list-style-type: none"> • Configure alarms. • Configure composite alarms. • Configure notification services. • Start, stop, and migrate instances. This capability is available for OpenStack. Nova API is invoked using a user token and OpenStack authorizes the user request. • Generate reports.
Read-Only Infrastructure View	<p>Users in this category can view the same entities as the Infrastructure View category but they cannot perform any of the actions listed in Infrastructure View. For example, a user in this category cannot configure an alarm. User can view an report generated by another Administrator or Infrastructure View user but a user in this category cannot generate a report.</p>
Tenant View	<p>A user that is not in any of the above categories will by default have access in the Tenant View category (if enabled). Users in this category can view all project and instances for which the user is a member. User can create alarms and generate reports for instances that are visible.</p>

Configure Roles

During installation, you can configure the set of roles for each user category. The following Ansible variables configure the roles. Define these variables in an inventory file for the Platform Host or in group variables for the `appformix_controller` group. (For example, in the file `group_vars/appformix_controller`).

<code>appformix_administrator_roles</code>	List of Keystone roles that comprise the Contrail Insights Administrator user category.
<code>appformix_infrastructure_view_roles</code>	List of Keystone roles that comprise the Infrastructure View user category.

`appformix_read_only_infrastructure_view_roles` List of Keystone roles that comprise the Read-Only Infrastructure View user category.

Granting Contrail Insights Permissions to Read-Only OpenStack Users

It is possible to give users access to Contrail Insights without granting OpenStack privileges to those users. This is achieved by creating a Keystone role and project in OpenStack such that users can interact with OpenStack APIs only in a read-only manner.

As an example, the following steps create an *appformix-admin* user that has a Contrail Insights Administrator role and an *appformix-infra* user that has Infrastructure View role. Both accounts do not have any quota by which to create resources in OpenStack. Other accounts, roles, and projects can be created in a similar manner.

To create an *appformix-admin* user that has a Contrail Insights Administrator role and an *appformix-infra* user that has Infrastructure View role:

1. Create a new Keystone role for users that will have an administrator role in Contrail Insights. For example, *AppFormixAdmin*.

```
$ openstack role create AppFormixAdmin
```

2. Create a new Keystone role for users that will have an infrastructure view role in Contrail Insights. For example, *AppFormixInfra*.

```
$ openstack role create AppFormixInfra
```

3. Create a new project in OpenStack. For example, *ReadOnly*.

```
$ openstack project create ReadOnly
```

4. Set all quotas for the *ReadOnly* project to 0. This is most easily accomplished using the OpenStack Horizon dashboard.

5. For users that should have administrator privilege in Contrail Insights, create a user in the ReadOnly project with the *AppFormixAdmin* role.

```
$ openstack user create --password-prompt \
  --description "Read-only OpenStack user for Contrail Insights administrator" \
  appformix-admin
$ openstack role add --project ReadOnly --user appformix-admin AppFormixAdmin
```

For users that should have infrastructure view privilege in Contrail Insights, create a user, and add the user to the ReadOnly project with the *AppFormixInfra* role.

```
$ openstack user create --password-prompt \
  --description "Read-only OpenStack user for AppFormix infrastructure view" \
  appformix-infra
$ openstack role add --project ReadOnly --user appformix-infra AppFormixInfra
```

6. Configure the mapping from Keystone roles to Contrail Insights roles.

Define the `appformix_administrator_roles` Ansible variable to include the Keystone roles that will have administrator privilege in Contrail Insights.

NOTE: The `admin` role is required for Contrail Insights Platform to access OpenStack.

Define the `appformix_infrastructure_view_roles` Ansible variable to include the Keystone roles that will have administrator privilege in Contrail Insights.

These variables can be defined in an inventory file for the Platform Host or in group variables for the `appformix_controller` group. For example, in the file `group_vars/appformix_controller`.

```
appformix_administrator_roles:
  - 'admin'
  - 'AppFormixAdmin'

appformix_infrastructure_view_roles:
  - 'AppFormixInfra'
```

RELATED DOCUMENTATION

[Bare Host | 86](#)[Remote Hosts | 141](#)[Contrail Insights User-Defined Plug-Ins | 117](#)[Service Monitoring Ansible Variables | 153](#)

Configure Network Device from JSON File

IN THIS SECTION

- [Overview | 106](#)
- [Network Device JSON Example for SNMP Device | 107](#)
- [Network Device JSON Example for JTI Device | 110](#)
- [Network Device JSON Example for gRPC Device | 111](#)
- [Network Device JSON Example for NetConf Device | 114](#)

Overview

To add network devices to Contrail Insights, you can draft a network device JSON file and set the following variable in `group_vars/all`:

```
network_device_file_name: /path/to/network_device_file_name.json
```

Multiple network devices can be added to the JSON file. This file can be written manually from the network inventory. Alternatively, tools like LLDP can be used to generate the device names, interface connections, and topology information.

The network device topology file should adhere to the following example schema(s). Note that a device can also be configured with multiple sources (SNMP, JTI, gRPC and NETCONF). Following is one

example for each source. Note that only `user.snmp`, `user.jti`, `user.grpc`, and `user.netconf` are accepted in the `Source` field and a period (".") cannot be specified in the `NetworkDeviceId` field.

Included in the release bundle is a sample configuration file: `sample_network_device_config.json`

Network Device JSON Example for SNMP Device

```
{
  "NetworkDeviceList": [
    {
      "NetworkDevice": {
        "MetaData": {
          "SnmpConfig": {
            "Version": "2c",
            "Community": "public",
            "PollInterval": 60,
            "OIDList": ["IF-MIB::ifTable",
                       "IF-MIB::ifXTable"]
          }
        },
        "ManagementIp": "10.1.1.2",
        "Name": "qfx1",
        "Source": ["user.snmp"],
        "NetworkDeviceId": "qfx1",
        "ChassisType": "tor",
        "ConnectionInfo": [
          {
            "remote_interface_name": "et-0/0/1",
            "local_interface_name": "et-0/1/0",
            "remote_system_id": "qfx2"
          },
          {
            "local_interface_name": "et-0/0/2",
            "remote_interface_name": "et-0/1/0",
            "remote_system_id": "qfx3"
          }
        ]
      }
    },
    {

```



```

    "NetworkDevice": {
      "MetaData": {
        "SnmpConfig": {
          "Version": "3",
          "Username": "user1",
          "Password": "password1",
          "Level": "authPriv",
          "Protocol": "SHA",
          "PrivProtocol": "DES",
          "PrivKey": "privacy protocol pass phrase",
          "SnmpEngineId": "800000000000",
          "OIDList": ["IF-MIB::ifTable",
                     "TCP-MIB::tcp"]
        }
      },
      "ManagementIp": "10.1.1.3",
      "Name": "qfx2",
      "NetworkDeviceId": "qfx2",
      "Source": ["user.snmp"],
      "ChassisType": "spine",
      "ConnectionInfo": [
        {
          "local_interface_name": "et-0/0/1",
          "remote_interface_name": "et-0/1/0",
          "remote_system_id": "qfx1"
        }
      ]
    }
  ]
}

```

Some fields have a fixed list of valid choices:

- NetworkDevice.MetaData.SnmpConfig.Version: "2c", "3"
- NetworkDevice.MetaData.SnmpConfig.Protocol: "MD5", "SHA"
- NetworkDevice.MetaData.SnmpConfig.Level: "noAuthNoPriv", "authNoPriv", "authPriv"
- NetworkDevice.MetaData.SnmpConfig.PrivProtocol: "AES", "DES"

The field `MetaData.SnmpConfig.PollInterval` is optional. You can input a value which is a multiple of 60. It determines the period that Contrail Insights will poll for SNMP data from the device (units is in seconds). By default, this value is set to 60 seconds.

Following are the possible values for the field `MetaData.SnmpConfig.OIDList`:

- "TCP-MIB::tcp"
- "IF-MIB::ifXTable"
- "IF-MIB::ifTable"
- "UDP-MIB::udp"
- "UCD-SNMP-MIB::memory"
- "IP-MIB::ipInReceives"
- "IPV6-MIB::ipv6IfTable"
- "JUNIPER-FIREWALL-MIB::jnxFWCounterDisplayFilterName"
- "NETWORK_DEVICE_SYSTEM_METRICS_MIB"
- "JUNIPER-PFE-MIB::jnxPfeMemoryUkernFreePercent"
- "JUNIPER-SUBSCRIBER-MIB::jnxSubscriberTotalCount"
- "JUNIPER-IPv6-MIB::jnxIpv6StatsReceives"
- "1.3.6.1.4.1.2636.3.4.2"
- "enterprises.2636.3.1.13.1"
- "enterprises.2636.3.39.1.12"
- "1.3.6.1.2.1.25.2"

These are all of the MIBs that Contrail Insights supports out of the box. To enable additional SNMP MIB monitoring, see ["Custom SNMP Plug-Ins" on page 134](#) for more information.

NOTE: SNMP Traps are automatically enabled on SNMP devices if the SNMP Trap plug-in is enabled. The `SnmpEngineId` field is needed for SNMP Trap v3.

Network Device JSON Example for JTI Device

```
{
  "NetworkDeviceList": [
    {
      "NetworkDevice": {
        "MetaData": {
          "Version": "17.2R1",
          "Username": "user2",
          "Password": "password2",
          "NetConfPort": 830,
          "JtiConfig": {
            "ReportRate": 60,
            "Dscp": 20,
            "DeviceLocalPort": 21112,
            "LocalAddress": "172.16.0.1",
            "ForwardingClass": "best-effort",
            "SensorList": [
              {
                "Name": "Interface_Sensor",
                "Resource": "/junos/system/linecard/interface/"
              }
            ]
          }
        },
        "ManagementIp": "10.1.1.4",
        "Name": "qfx3",
        "Source": ["user.jti"],
        "NetworkDeviceId": "qfx3",
        "ChassisType": "tor",
        "ConnectionInfo": [
          {
            "remote_interface_name": "et-0/0/2",
            "local_interface_name": "et-0/1/0",
            "remote_system_id": "qfx2"
          }
        ]
      }
    }
  ]
}
```

Following are the possible values for the field `MetaData.JtiConfig.SensorList`:

- `"/junos/system/linecard/cpu/memory/"`
- `"/junos/system/linecard/fabric/"`
- `"/junos/system/linecard/firewall/"`
- `"/junos/system/linecard/interface/"`
- `"/junos/system/linecard/interface/logical/usage/"`
- `"/junos/system/linecard/npv/memory/"`
- `"/junos/system/linecard/npv/utilization/"`
- `"/junos/system/linecard/optics/"`
- `"/junos/system/linecard/packet/usage/"`
- `"/junos/system/linecard/qmon/"`

These are all of the JTI UDP sensors Contrail Insights supports out of the box. To enable additional JTI sensor monitoring, see ["Custom Sensors for JTI, gRPC, and NETCONF" on page 137](#) for more information.

Note that the variables `ForwardingClass`, `DeviceLocalPort`, `Dscp`, `ReportRate`, `PayloadSize`, and `LocalAddress` are optional variables. `DeviceLocalPort` will default to 21112, `Dscp` will default to 20, and `ReportRate` will default to 60 if you do not supply these values. `LocalAddress` is used to set the local-address of export-profile in the device configuration. If you do not supply `LocalAddress`, then Contrail Insights will use the device's `ManagementIp` field to set local-address of export-profile on the device.

Network Device JSON Example for gRPC Device

```
{
  "NetworkDeviceList": [
    {
      "NetworkDevice": {
        "MetaData": {
          "Version": "17.2R1",
          "Username": "root",
          "Password": "pass",
          "NetConfPort": 830,
          "GrpcConfig": {
```

```

        "DeviceLocalPort": 50051,
        "SSLEnabled": false,
        "SensorList": [
            {
                "Resource": "/interfaces/",
                "ReportRate": 60
            },
            {
                "Resource": "/junos/system/subscriber-management/infra/sdb/
statistics/",
                "ReportRate": 60
            }
        ],
        "ManagementIp": "10.1.1.4",
        "Name": "qfx4",
        "Source": ["user.grpc"],
        "NetworkDeviceId": "qfx4",
        "ChassisType": "tor",
        "ConnectionInfo": [
            {
                "remote_interface_name": "et-0/0/2",
                "local_interface_name": "et-0/1/0",
                "remote_system_id": "qfx5"
            }
        ]
    }
}
]
}

```

Following are the possible values for the field `MetaData.GrpcConfig.SensorList`:

- `"/arp-information/ipv4/neighbors/neighbor/"`
- `"/bgp-rib/afi-safis/afi-safi/ipv4-unicast/loc-rib/"`
- `"/bgp-rib/afi-safis/afi-safi/ipv4-unicast/neighbors/"`
- `"/bgp-rib/afi-safis/afi-safi/ipv6-unicast/loc-rib/"`
- `"/bgp-rib/afi-safis/afi-safi/ipv6-unicast/neighbors/"`
- `"/components/"`

- "/interfaces/"
- "/junos/npu-memory/"
- "/junos/rsvp-interface-information/"
- "/junos/services/label-switched-path/usage/"
- "/junos/services/segment-routing/interface/egress/usage/"
- "/junos/services/segment-routing/interface/ingress/usage/"
- "/junos/services/segment-routing/sid/usage/"
- "/junos/system/linecard/cpu/memory/"
- "/junos/system/linecard/firewall/"
- "/junos/system/linecard/intf-exp/"
- "/junos/system/linecard/npu/memory/"
- "/junos/system/linecard/npu/utilization/"
- "/junos/system/linecard/optics/"
- "/junos/system/linecard/packet/usage/"
- "/junos/system/linecard/qmon-sw/"
- "/junos/system/subscriber-management/aaa/accounting-statistics/"
- "/junos/system/subscriber-management/aaa/dynamic-request-statistics/"
- "/junos/system/subscriber-management/client-protocols/dhcp/v6/"
- "/junos/system/subscriber-management/client-protocols/dhcp/v4/"
- "/junos/system/subscriber-management/client-protocols/l2tp/summary/"
- "/junos/system/subscriber-management/client-protocols/ppp/statistics/"
- "/junos/system/subscriber-management/client-protocols/pppoe/statistics/"
- "/junos/system/subscriber-management/infra/network/ppp/"
- "/junos/system/subscriber-management/infra/network/pppoe/"
- "/junos/system/subscriber-management/infra/sdb/statistics/"

- `"/junos/task-memory-information/task-memory-overall-report/task-memory-stats-list/task-memory-stats/"`
- `"/junos/task-memory-information/task-memory-overall-report/task-size-block-list/task-size-block/"`
- `"/lacp/"`
- `"/lldp/interfaces/interface/neighbors/"`
- `"/lldp/interfaces/interface/state/"`
- `"/nd6-information/"`

These are all of the gRPC sensors Contrail Insights supports out of the box. To enable additional gRPC sensor monitoring, see ["Custom Sensors for JTI, gRPC, and NETCONF" on page 137](#) for more information.

Note that the variable `SSLEnabled` determines whether you want to set up a secure connection to the device. See [Contrail Insights JTI \(gRPC\) Monitoring](#) for more information.

`DeviceLocalPort` is an optional variable which defaults to 50051.

Network Device JSON Example for NetConf Device

```
{
  "NetworkDeviceList": [
    {
      "NetworkDevice": {
        "MetaData": {
          "Version": "17.2R1",
          "Username": "root",
          "Password": "pass",
          "NetConfPort": 830,
          "NetConfConfig": {
            "CommandList": [
              {
                "CommandLine": "show interfaces terse",
                "Interval": 60
              }
            ]
          }
        },
        "ManagementIp": "10.1.1.5",
        "Name": "qfx5",

```

```

    "Source": ["user.netconf"],
    "NetworkDeviceId": "qfx5",
    "ChassisType": "tor",
    "ConnectionInfo": [
      {
        "remote_interface_name": "et-0/1/0",
        "local_interface_name": "et-0/0/2",
        "remote_system_id": "qfx4"
      }
    ]
  }
}
]
}

```

RELATED DOCUMENTATION

No Link Title

No Link Title

Contrail Insights Plug-Ins

To install Contrail Insights certified plug-ins on the cluster, include the following variables in the `group_vars/all` file:

```

appformix_plugins: <list of certified plugins to be installed>
appformix_openstack_log_plugins: <list of OpenStack log plugins to be installed>

```

For example:

```

appformix_plugins:
  - { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
  - { plugin_info: 'certified_plugins/contrail_vrouter.json' }
  - { plugin_info: 'certified_plugins/zookeeper_usage.json' }
  - { plugin_info: 'certified_plugins/heavy_hitters.json' }
appformix_openstack_log_plugins:

```



```
- { plugin_info: 'certified_plugins/cinder_api_logparser.json',
  log_file_path: '/var/log/cinder/cinder-api.log' }
- { plugin_info: 'certified_plugins/glance_logparser.json',
  log_file_path: '/var/log/glance/glance-api.log' }
- { plugin_info: 'certified_plugins/keystone_logparser.json',
  log_file_path: '/var/log/apache2/keystone_access.log,/var/log/httpd/
keystone_wsgi_admin_access.log,/var/log/keystone/keystone.log' }
```

For a list of all Contrail Insights certified plug-ins that can be installed, look for the entries starting with `plugin_info` in the file `roles/appformix_defaults/defaults/main.yml`.

The OpenStack log parser plug-ins parse the API log files of each OpenStack service to collect metrics about API calls and response status codes. To install these plug-ins, add them to the variable `appformix_openstack_log_plugins` in `group_vars/all`, as shown above. Each plug-in entry in this list requires a parameter called `log_file_path` to be specified. This parameter should be set to the complete path to the service's API log file on the OpenStack Controller node(s). Multiple comma-separated paths can be specified.

To identify the correct log file to be specified in `log_file_path`, look for entries like the following, containing a client IP address, REST call type, and response status code:

```
2019-04-02 06:50:13.103 3465 INFO nova.osapi_compute.wsgi.server [req-d07e953a-6921-4224-a056-
afb6ff69adde 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/os-hypervisors/detail HTTP/1.1" status: 200 len: 1427
time: 0.0208740
2019-04-02 06:50:13.183 3465 INFO nova.osapi_compute.wsgi.server [req-34b2f686-9eb5-4112-b3fc-
e0b37798a302 953ea56a96b944b3b170a299af9e87bd 10c9e8809feb4bd1b55955d9c2ed5aba - - -] 172.18.0.6
"GET /v2/10c9e8809feb4bd1b55955d9c2ed5aba/servers/detail?all_tenants=1&status=SHELVED_OFFLOADED
HTTP/1.1" status: 200 len: 211 time: 0.0754580
```

Default locations for these files are listed in the variable `appformix_openstack_log_factory_plugins` in `roles/appformix_defaults/defaults/main.yml`.

On containerized OpenStack environments, log files are generated inside the containers running the OpenStack services. However, they have to be available on the OpenStack controller host for the Contrail Insights plug-ins to be able to read them. The path specified in `log_file_path` should be the location of the file on the OpenStack Controller host.

NOTE: In AppFormix 3.0, all OpenStack log parser plug-ins have to be specified in the variable `appformix_openstack_log_plugins`. When upgrading from an earlier version to 3.0, make sure to move all OpenStack log parser plug-ins defined in `appformix_plugins` to `appformix_openstack_log_plugins`. Also,

in AppFormix 3.0, all entries in this list have to be specified with a `log_file_path` value, as described in example above.

Contrail Insights User-Defined Plug-Ins

IN THIS SECTION

- [Add a User-Defined Command Plug-In | 117](#)
- [Configure a Supported Plug-In | 121](#)

Contrail Insights plug-ins can be used to extend the set of metrics monitored and analyzed by Contrail Insights. Plug-ins are executed by a Contrail Insights Agent. The metrics produced by the plug-in are associated with the host on which the Agent executed.

Contrail Insights Ansible playbooks install a plug-in executable file on a host(s) and configure the plug-in in the Contrail Insights Platform.

Add a User-Defined Command Plug-In

To create a new plug-in in Contrail Insights, provide the executable files with the valid output format. For this example, the `check_nginx.py` file is used, which has the following output:

```
$ python ./check_nginx.py
OK - nginx.node: 4qps nginx_concurrent_connections
```

```
$ cat /opt/appformix/manager/tailwind_manager/check_nginx.py
import sys
import traceback
import commands
```

```

def nginx_metrics():
    result = {'connections': 0}
    try:
        command = "netstat -lanp | grep 'nginx: worker' | wc -l"
        out = commands.getoutput(command)
        result['connections'] = int(out)
    except Exception as e:
        traceback.print_exc()
    return result

def main(argv):
    result = nginx_metrics()
    try:
        msg = ('nginx.node: {0}qps nginx_concurrent_connections').\
            format(
                result['connections']
            )
    except Exception as e:
        traceback.print_exc()
    test_state = 0
    logger_helper(
        msg,
        test_state
    )

def logger_helper(message, state):
    if state == 2:
        print "CRITICAL - " + message
        sys.exit(2)
    elif state == 1:
        print "WARNING - " + message
        sys.exit(1)
    elif state == 0:
        print "OK - " + message
        sys.exit(0)
    else:
        print "CRITICAL - Unexpected value : %d" % state + "; " + message
        sys.exit(2)

```

```
if __name__ == "__main__":
    sys.exit(main(sys.argv[1:]))
```

In the Contrail Insights installation directory, there is a directory named `user_defined_plugins`. You need to create a `<PLUGIN_FILE>.json` with the configuration for the plug-in and the metrics it exposes. In the following example, we create a `nginx_connections.json` for the `check_nginx.py` plug-in referenced above.

```
$ cd appformix-<VERSION>/
$ touch appformix-<VERSION>/user_defined_plugins/<PLUGIN_NAME>.json

{
  "AggregateId": "process_monitoring_A",
  "Collection": "host_pythonchecknginxpy_collection",
  "Config": {
    "CommandLine": "python check_nginx.py"
  },
  "MetricMap": [
    {
      "Name": "nginx_concurrent_connections",
      "Units": "qps"
    }
  ],
  "PluginId": "pythonchecknginxpy",
  "PluginName": "nginx.node",
  "PluginType": "command",
  "Scope": "host",
  "State": "enabled",
  "status": "success"
}
```

The field `MetricMap` is a list of metrics and its unit that is collected by this plug-in and field `AggregateId` indicates in what host aggregate you want to run this plug-in. If the `AggregateId` is specified, then the plug-in will only apply to the hosts in that aggregate. If `AggregateId` is not specified, then the plug-in will run on all of the hosts monitored by Contrail Insights.

[Table 8 on page 120](#) describes the fields in the sample configuration.

Table 8: Explanation of Fields in the Plug-In Configuration JSON File

Field	Description
AggregateId	Specifies a host aggregate. Plug-in is installed and configured to run on hosts in this aggregate. If AggregateId is not specified, then plug-in is configured to run on all hosts on which Contrail Insights Agent is installed.
Collection	A label applied to data produced by this plug-in.
Config.CommandLine	The command to execute.
MetricMap	A list of metrics produced by the plug-in and the units of each metric (for display in the Dashboard).
PluginId	Plug-in identifier in Contrail Insights. This must be unique among all configured plug-ins.
PluginName	Name of the plug-in. Name is used as a prefix for the name of all metrics produced by the plug-in.
PluginType	Type of the plug-in. Valid option: command.
Scope	Scope of this plug-in. Valid option: host.
State	Specifies if plug-in is active. Valid options: enabled, disabled.

The user- defined plug-ins need to be specified in the `group_vars` as follows:

```
appformix_user_defined_plugins:
  - { plugin_info: 'user_defined_plugins/nginx_connections.json',
      plugin_file: 'user_defined_plugins/check_nginx.py' }
```

The Contrail Insights Ansible playbooks copies these two types of files to all of the appropriate agents and then configures the plug-in in the `appformix_controller`.

Configure a Supported Plug-In

Contrail Insights includes a set of supported plug-ins in the **certified_plugins** directory of the release bundle. You can configure a supported plug-in to be installed by setting the `appformix_plugins` variable in Ansible and running the installation playbook. For example:

```
appformix_plugins:
  - { plugin_info: 'certified_plugins/contrail_vrouter.json' }
  - { plugin_info: 'certified_plugins/cassandra_node_usage.json' }
  - { plugin_info: 'certified_plugins/zookeeper_usage.json' }
```

RELATED DOCUMENTATION

[Contrail Insights Plug-Ins | 115](#)

[Instance Scope Plug-Ins | 121](#)

[Contrail Insights Object Plug-In | 128](#)

[Custom SNMP Plug-Ins | 134](#)

Instance Scope Plug-Ins

IN THIS SECTION

- [Create Instance Scope Plug-In | 122](#)
- [Configure Instance Scope Plug-In | 126](#)
- [Post Instance Scope Plug-In to Contrail Insights | 128](#)

Instance scope plug-in is provided as an extension to host scope plug-in. Its purpose is to monitor user-defined metrics and associate them with instances. The instances being monitored are expected to be members of an instance aggregate. Currently, the plug-in type REST is supported. REST plug-ins allow you to receive structured metrics data from one or multiple endpoints. This data can then be monitored and analyzed on the Dashboard.

Create Instance Scope Plug-In

To create an instance scope plug-in, provide an executable script that produces output in a valid Nagios-Style string format. The script should accept two arguments: instance ID and input parameters in a JSON string. The input parameters argument contains whatever information is needed to get metrics data, such as one or multiple endpoints, authentication, and so on.

The following is an example plug-in script called `demo_check_instance_plugin.py`:

```
$ cat /Appformix/controller/tools/sdk/samples/demo_check_instance_plugin.py
```

```
import json
import sys
import traceback
import requests
import argparse

# Step 1 - Get metrics data from input parameters.
# The suggested output format for each instance is as follows.
# If the output is not in this format, please process your data
# in this function to follow this format.
# {
#   <metric.name>: {
#     "type": <metric.type>,
#     "unit": <metric.units>,
#     "value": <metric.value>
#   },
#   <metric.name>: {
#     "type": <metric.type>,
#     "unit": <metric.units>,
#     "value": <metric.value>
#   },
#   "roomKey": {
#     "type": "value",
#     "unit": "roomKey",
#     "value": <instance_id>
#   }
# }
# The parameters argument is a JSON string which is entirely user defined.
# Thus please modify the code to get specific parameter,
# construct and rest get endpoint url according to your parameters format.
```

```

# And also add and process auth info as needed.
def get_plugin_metrics_data(argv):
    instance_id = argv.instanceId
    parameters = argv.parameters
    endpoint = parameters.get('Endpoint', '')
    parameters.pop('Endpoint')
    url = endpoint + '/' + instance_id
    try:
        resp = requests.get(url=url, params=parameters)
        output = json.loads(resp.text)
    except Exception as e:
        output = {}
    return output

# Step 2 - Output metrics data as a Nagios-Style string.
# The output format is:
#     OK - <plugin_name_suffix>: {"roomKey": {"type":
#     "value", "value": <instance_id>, "unit": "roomKey"},
#     "metric1": {"type": <metric1.type>, "value": <metric1.value>,
#     "unit": <metric1.units>}, "metric2": {"type": <metric2.type>,
#     "value": <metric2.value>, "unit": <metric2.units>}}
def main(argv):
    result = get_plugin_metrics_data(argv)
    try:
        result = json.dumps(result)
        msg = u"instance_plugin: {}".format(result)
    except Exception as e:
        traceback.print_exc()
        msg = u'instance_plugin: {}'
    print "OK - " + msg
    sys.exit(0)

# Step 3 - Two arguments are taken for the command:
# instance id and input parameters in a JSON string.
# The JSON string contains whatever information
# that is needed to get metrics data,
# such as one or multiple endpoints, authentication, etc.
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '-i', '--instanceId', help='Add an instanceId',
        required=True, default='', type=unicode)

```



```

parser.add_argument(
    '-p', '--parameters', help='Add input parameters',
    required=True, default='{}', type=json.loads)
args = parser.parse_args()
sys.exit(main(args))

```

This script expects the data received for each instance to have the following JSON format.

```

{
  <metric.name>: {
    "type": <metric.type>,
    "unit": <metric.units>,
    "value": <metric.value>
  },
  <metric.name>: {
    "type": <metric.type>,
    "unit": <metric.units>,
    "value": <metric.value>
  },
  "roomKey": {
    "type": "value",
    "unit": "roomKey",
    "value": <instance_id>
  }
}

```

For each instance, there is a `roomKey` field whose value is the instance ID. This ID is used by the Dashboard to associate an instance with its data.

[Table 9 on page 124](#) describes the metric fields.

Table 9: Explanation of Metrics Fields in Instance Plug-In

Field	Description
<code>metric.value</code>	Must contain only digits and optional decimal point: <code>[0-9]+.[0-9]</code> .
<code>metric.units</code>	Must be a valid string that starts with a letter.
<code>metric.name</code>	Must be a valid string that starts with a letter.

Table 9: Explanation of Metrics Fields in Instance Plug-In (Continued)

Field	Description
metric.type	Must be a valid string that starts with a letter. Valid option: value .

The sample script `demo_check_instance_plugin.py` produces output in the following format:

```
OK - <instance_plugin>: {"roomKey": {"type": "value", "value": <instance_id>, "unit":
    "roomKey"}, "metric1": {"type": <metric1.type>, "value": <metric1.value>, "unit":
404
    <metric1.units>}, "metric2": {"type": <metric2.type>, "value": <metric2.value>,
    "unit": <metric2.units>}}
```

Example: Metrics data for instance.

The example uses:

- `instanceid—3a0b1cf8-037a-4005-88e6-929c9c0e44f4`
- `parameters—{'Endpoint': 'http://172.24.82.192:5000/metrics_data'}`

```
{
  "metric1": {
    "type": "value",
    "unit": "Count",
    "value": 700
  },
  "metric2": {
    "type": "value",
    "unit": "Count",
    "value": 700
  },
  "roomKey": {
    "type": "value",
    "unit": "roomKey",
    "value": "3a0b1cf8-037a-4005-88e6-929c9c0e44f4"
  }
}
```

After running the command, output is as follows:

```
$ python demo_check_instance_plugin.py -i 3a0b1cf8-037a-4005-88e6-929c9c0e44f4 -p '{"Endpoint":
"http://172.24.82.192:5000/metrics_data"}'

OK - instance_plugin: {"roomKey": {"type": "value", "value":
"3a0b1cf8-037a-4005-88e6-929c9c0e44f4", "unit": "roomKey"}, "metric1": {"type": "value",
"value": 700, "unit": "Count"}, "metric2": {"type": "value", "value": 700, "unit": "Count"}}
```

Configure Instance Scope Plug-In

To add the instance scope plug-in, define a JSON configuration file which specifies how to execute the plug-in and the metrics that are produced by the plug-in. The following is a sample configuration file `demo_instance_plugin.json`. You can customize this sample file according to your needs.

Example: Instance plug-in configuration JSON file.

```
$ cat /Appformix/controller/tools/sdk/samples/demo_instance_plugin.json

{
  "AggregateId": "instance_plugin_rest",
  "Collection": "instance_plugin_collection",
  "Config": {
    "CommandLine": "python demo_check_instance_plugin.py"
  },
  "MetricMap": [
    {
      "Name": "metric1",
      "Units": "Count"
    },
    {
      "Name": "metric2",
      "Units": "Count"
    }
  ],
  "PluginId": "instance_plugin_id",
  "PluginName": "instance_plugin_name",
  "PluginType": "rest",
  "Scope": "instance",
```

```

    "State": "enabled",
    "MetaData": {
        "Endpoint": "http://172.24.82.192:5000/metrics_data",
        "Username": "username",
        "Password": "password"
    },
}

```

Table 10 on page 127 describes the fields in the sample configuration.

Table 10: Explanation of Fields in Instance Plug-In Configuration JSON File

Field	Description
AggregateId	Specifies an instance type aggregate. Plug-in is installed and configured to run on instances in this aggregate.
Collection	A label applied to data produced by this plug-in.
Config.CommandLine	The command to execute, For example, Python demo_check_instance_plugin.py. Exclude the arguments list here.
MetricMap	A list of metrics produced by the plug-in and the units of each metric (for display in the Dashboard).
PluginId	Plug-in identifier in Contrail Insights. This must be unique among all configured plug-ins.
PluginName	Name of the plug-in. Name is used as a prefix for the name of all metrics produced by the plug-in.
PluginType	Type of plug-in. Valid option: rest .
Scope	Scope of this plug-in. Valid option: instance .
State	Specifies if plug-in is active. Valid options: enabled , disabled .

Table 10: Explanation of Fields in Instance Plug-In Configuration JSON File *(Continued)*

Field	Description
MetaData	Specifies whatever information that is needed to get metrics data, such as endpoints, authentication, and so on. This will be passed as input parameters argument for the executable command.

Post Instance Scope Plug-In to Contrail Insights

After creating the configuration file for the instance scope plug-in, the next step is to post your plug-in to Contrail Insights, which is done by using the Ansible playbooks, as described in ["Contrail Insights User-Defined Plug-Ins" on page 117](#). In the Contrail Insights installation directory, there is a directory named `user_defined_plugins`. Copy both the python and JSON files to this directory and include the plug-in descriptor in the `appformix_user_defined_plugins` variable in `group_vars/all`. This variable must be initialized at the time of Contrail Insights Platform installation.

The user-defined instance plug-in needs to be specified in the `group_vars` file as follows:

```
appformix_user_defined_plugins:
  - { plugin_info: 'user_defined_plugins/demo_instance_plugin.json',
      plugin_file: 'user_defined_plugins/demo_check_instance_plugin.py' }
```

The Contrail Insights Ansible playbooks will copy these two types of files to all of the appropriate Agents and then configure the plug-in in the `appFormix_controller`.

Contrail Insights Object Plug-In

IN THIS SECTION

- [Create an Object Scope Plug-In | 129](#)
- [Add Object Scope Plug-In to Contrail Insights | 133](#)

Object scope plug-in is supported to extend the instance and host scope plug-in. It will assign an aggregate of hosts running a plug-in script to collect data for an aggregate of objects (such as instances). Currently, only an aggregate of instances can be monitored and only the plug-in type `object.rest` is supported.

The collector failover High Availability (HA) feature is supported in object scope plug-ins. If one of the collectors in the aggregate is down, Contrail Insights will automatically assign another host in the same aggregate as the collector. As a result, you can set up an aggregate of several collectors which will make sure downtime of some of the collectors will not affect the object monitoring.

Create an Object Scope Plug-In

To add an object scope plug-in in Contrail Insights, you need to supply both a plug-in script and plug-in JSON file. Contrail Insights Agent will pass a list of `object_ids` separated by commas (-o) and a dictionary to `MetaData` (-p) to the plug-in script. The Contrail Insights Agent expects the plug-in script to return data with the following format:

```
OK - {{plugin_name}}: [{"roomKey": {"value": {{instance_id_1}}, "unit": "roomKey"}, "metric_1": {"value": {{value_1}}, "unit": "m1_unit"}, "metric_2": {"value": {{value_2}}, "unit": "m2_unit"}}, {"roomKey": {"value": {{instance_id_2}}, "unit": "roomKey"}, "metric_1": {"value": {{value_3}}, "unit": "m1_unit"}, "metric_2": {"value": {{value_4}}, "unit": "m2_unit"}}]
```

The main body of the data should be a list of dictionaries and each entry should represent a data point with all of the metrics for one instance.

Following is an example of a plug-in script which will get instance data from some endpoint, passed in from the `MetaData` variable.

```
$ cat check_object_plugin.py

import sys
import traceback
import requests
import json
import argparse

def main(argv):
    instance_list = argv.instances.split(',')

```

```

url = json.loads(argv.metadata)['url']
data = json.dumps({'InstanceList': instance_list})
resp = requests.get(url=url, data=data,
                    headers={'content-type': 'application/json'})
result = json.loads(resp.text)
return_value = []
for entry in result['data']:
    instance_data = \
        {'roomKey': {'unit': 'roomKey', 'value': entry['id']},
         'm1': {'unit': 'm1_unit', 'value': entry['m1']},
         'm2': {'unit': 'm2_unit', 'value': entry['m2']}}
    return_value.append(instance_data)
try:
    return_value = json.dumps(return_value)
    msg = 'instance_object_plugin: {}'.format(return_value)
except Exception as e:
    traceback.print_exc()
    msg = 'instance_object_plugin: []'
test_state = 0
logger_helper(
    msg,
    test_state
)

def logger_helper(message, state):
    if state == 2:
        print "CRITICAL - " + message
        sys.exit(2)
    elif state == 1:
        print "WARNING - " + message
        sys.exit(1)
    elif state == 0:
        print "OK - " + message
        sys.exit(0)
    else:
        print "CRITICAL - Unexpected value : %d" % state + "; " + message
        sys.exit(2)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(

```

```

    '-o', '--instances', help='instance list', required=True,
    default='', type=unicode)
parser.add_argument(
    '-p', '--metadata', help='metadata', required=False,
    default='{}', type=unicode)
args = parser.parse_args()
sys.exit(main(args))

```

Following is the corresponding flask server running on localhost:9999 for the above example:

```

from flask import Flask, request
import json
import copy

data = {'a53a441b-fb1f-4e08-9e05-d94fd4d5a0ba': {'m1': 1, 'm2': 2},
        '82c20111-32b2-4a47-a668-cd1d6c69270d': {'m1': 3, 'm2': 4},
        'fd5977d4-150d-4af4-bfaf-96663902447e': {'m1': 5, 'm2': 6},
        'a35414ab-e779-48a8-93d9-1ae50647fdc5': {'m1': 7, 'm2': 8}}

app = Flask(__name__)

@app.route('/', methods=['GET'])
def app_message_get():
    if request.headers['Content-Type'] != 'application/json':
        print "invalid"
        return json.dumps({'result': "invalid"})
    args = request.json
    result = []
    for instance_id in args.get('InstanceList', []):
        if instance_id in data:
            entry = copy.deepcopy(data[instance_id])
            entry['id'] = instance_id
            result.append(entry)
    return json.dumps({'data': result})

if __name__ == '__main__':
    app.run(
        host="0.0.0.0",

```



```
port=int("9999")
)
```

For example, if you want to get data for instances fd5977d4-150d-4af4-bfaf-96663902447e and a53a441b-fb1f-4e08-9e05-d94fd4d5a0ba with URL <http://localhost:9999> (because your server is running on localhost port 9999), you can run the following command:

```
python check_object_plugin.py -o fd5977d4-150d-4af4-bfaf-96663902447e,a53a441b-fb1f-4e08-9e05-d94fd4d5a0ba -p '{"url": "http://localhost:9999"}'
```

The output of the command is:

```
OK - instance_object_plugin: [{"roomKey": {"value": "fd5977d4-150d-4af4-bfaf-96663902447e",
"unit": "roomKey"}, "m1": {"value": 5, "unit": "m1_unit"}, "m2": {"value": 6, "unit":
"m2_unit"}}, {"roomKey": {"value": "a53a441b-fb1f-4e08-9e05-d94fd4d5a0ba", "unit": "roomKey"},
"m1": {"value": 1, "unit": "m1_unit"}, "m2": {"value": 2, "unit": "m2_unit"}}]
```

The corresponding plug-in JSON file will be:

```
$ cat instance_usage.json
{
  "Collection": "instance_plugin_usage_collection",
  "Config": {
    "CommandLine": "python check_object_plugin.py",
    "ObjectAggregateId": "7ae841ce-ead4-11e9-b283-0242ac130005"
  },
  "MetaData": {"url": "http://localhost:9999"},
  "AggregateId": "appformix_platform",
  "MetricMap": [
    {
      "Name": "m1",
      "Units": "m1_unit"
    },
    {
      "Name": "m2",
      "Units": "m2_unit"
    }
  ],
  "PluginId": "instance_object_plugin",
  "PluginName": "instance_object_plugin",
```

```

    "PluginType": "object.rest",
    "Scope": "object",
    "State": "enabled",
    "status": "success"
  }

```

Where `ObjectAggregateId` should be the aggregate identifier of the aggregate of instances you want to monitor and `AggregateId` should be the aggregate identifier of the host aggregate which is assigned to collect the data for the instances.

Note that you can put additional information in the `MetaData` field in the plug-in as needed. The above example is just a simple example.

Add Object Scope Plug-In to Contrail Insights

To add the object scope plug-in to Contrail Insights, you need to post your plug-in to Contrail Insights through Ansible. Add the following lines in your Ansible `group_vars/all` file and re-run the Ansible playbook.

```

appformix_user_defined_plugins:
  - { plugin_info: 'user_defined_plugins/instance_usage.json',
      plugin_file: 'user_defined_plugins/check_object_plugin.py' }

```

The Contrail Insights Ansible playbooks will then copy these two types of files to all of the appropriate Agents and then configure the plug-in in the Contrail Insights Platform.

RELATED DOCUMENTATION

[Instance Scope Plug-Ins | 121](#)

[Contrail Insights User-Defined Plug-Ins | 117](#)

Custom SNMP Plug-Ins

IN THIS SECTION

- [Sample Custom SNMP Plug-In Configuration File | 134](#)
- [Configure Custom SNMP Plug-Ins in Contrail Insights | 136](#)
- [Install MIBs in Contrail Insights Agents from Ansible | 136](#)

Contrail Insights can monitor SNMP metrics of network devices in your data center. Contrail Insights includes several supported SNMP plug-ins shipped with the product including ifTable, ifXTable, and TCP plug-ins for network devices for all vendors and three more plug-ins for Juniper devices. In addition to the Contrail Insights built-in SNMP plug-ins, Contrail Insights allows you to create your own custom SNMP plug-ins. The custom SNMP plug-ins allow you to define the SNMP MIBs and metrics that you want to monitor. The only thing you need is to define a configuration file for the SNMP MIB you want to monitor.

NOTE: Confirm that you installed the related MIBs on the `appformix_network_agent` nodes before you generate the plug-ins.

Sample Custom SNMP Plug-In Configuration File

The sample configuration file is listed. You can also find this sample file in the Contrail Insights release bundle `sample_snmp_config.py`:

```
SNMP_OID = ['enterprises.2636.3.1.13.1']
SNMP_MIB_KEYS = [
    {'Name': 'jnxOperatingTemp', 'Units': 'C',
     'CalculationMethod': 'value'},
    {'Name': 'jnxOperatingCPU', 'Units': '%',
     'CalculationMethod': 'value'}
]
PLUGIN_TYPE = 'snmp_network_device'
PLUGIN_NAME = 'snmp_network_device_usage'
```

```

SNMP_TAG_KEY = 'jnxOperatingDescr'
AGGREGATE_ID = 'appformix_network_agents'
PLUGIN_PREFIX = 'device'
LOAD_ALL = True
ENTITY_TYPE = ["fpc", "routing_engine", "pic", "fan", "power"]

```

Table 11 on page 135 describes the fields.

Table 11: SNMP Plug-In Configuration File Field Descriptions

Field	Description
SNMP_OID	The OID of SNMP MIB you want to monitor.
SNMP_MIB_KEYS	The metrics in this MIB are metrics that you want to monitor. The key needs to match the key received from the SNMP command.
PLUGIN_TYPE	<p>There are three types of SNMP plug-ins:</p> <p>snmp_network_device Refers to tabular MIBs which define multiple related object instances grouped in MIB tables. For example, ifTable defined the metrics for a list of interfaces on this device.</p> <p>snmp_device_info Refers to singular MIBs which define a single object instance. For example, 1.3.6.1.2.1.1.7 (OID for sysServices). You can specify by OID and name it.</p> <p>snmp_device_attribute Refers to the case where there is not a field to determine the tag for each entry in tabular MIBs. Instead, the sample SNMP output of these MIBs are metric_name .x.x.x.x = value where x.x.x.x is the tag name. For instance, OID 1.3.6.1.4.1.9.9.42.1.1.8.1 (OID for CISCO-RTTMON-MIB).</p>
SNMP_TAG_KEY	This field is required for tabular (plugin_type=snmp_network_device) MIBs to determine the tag for each entry of the SNMP output. This field is not needed for singular (plugin_type=device_info) MIBs .
AGGREGATE_ID	Must be set to "appformix_network_agents".

Table 11: SNMP Plug-In Configuration File Field Descriptions (*Continued*)

Field	Description
PLUGIN_PREFIX	(Optional) This field is the prefix of the metrics shown in the Contrail Insights Dashboard. For example, for OID IF-MIB::ifTable, the prefix could be <i>interface</i> .
CalculationMethod	Prior to release 2.16, this field was called CalculationType. This denotes how Contrail Insights calculates this metric (rate/value).
ENTITY_TYPE	A list of strings that indicates what this MIBs data represents. If it reports interface data, ENTITY_TYPE=['interface'].

Configure Custom SNMP Plug-Ins in Contrail Insights

After you create the configuration file for your custom SNMP plug-in, the next step is to post your plug-in to Contrail Insights. Copy your configuration file to your Contrail Insights Ansible directory and include the plug-in descriptor in the `appformix_custom_snmp_plugins` variable in `group_vars/all`. This variable must be initialized at the time of the Contrail Insights Platform installation. Save the configuration file as **sample_snmp_config.py** in your installation directory.

```
appformix_custom_snmp_plugins:
  - { config_file_name: 'sample_snmp_config.py' }
```

Then Contrail Insights will take your configuration file, generate the plug-in configuration JSON file and install this plug-in to Contrail Insights during the Contrail Insights Platform installation.

Install MIBs in Contrail Insights Agents from Ansible

When adding or enabling a new plug-in in Contrail Insights, you need to install the corresponding MIBs in your Contrail Insights Agents. The Contrail Insights Ansible script helps you install the MIBs in both `appformix_controller` nodes and Contrail Insights Agents.

Set the following variable in `group_vars/all`:

```
installer_mib_location: 'snmp_mibs/*'
```

Add the needed MIB files (.txt) in a folder and Contrail Insights will install all the MIBs from that folder in your `appformix_controller` nodes and Contrail Insights Agents by copying those MIB files to the location set in the `platform_mib_location` variable. You can change this variable in `group_vars/all`. The default value is `/usr/share/snmp/mibs/`.

RELATED DOCUMENTATION

| No Link Title

Custom Sensors for JTI, gRPC, and NETCONF

IN THIS SECTION

- [Overview | 137](#)
- [Sensor Configuration File | 138](#)
- [Enable Sensors Through Contrail Insights Ansible Playbook | 140](#)
- [Add More Metrics to Existing JTI, gRPC, and NETCONF Sensors | 140](#)

Overview

Contrail Insights includes several built-in JTI, gRPC, and NETCONF sensors. To enable these sensors, you can add the sensor information to the network device configuration file or from UI.

Contrail Insights also allows you to add new JTI, gRPC, and NETCONF sensors into the system. You need to supply a configuration file, set the variables in Ansible `group_vars`, and then re-run the playbook. The Contrail Insights Ansible playbook will deploy the new sensor configurations and start parsing data for the new sensor.

Starting with Contrail Insights version 3.3.0, NETCONF custom command support is added.

Sensor Configuration File

The sensor configuration file you provide for Contrail Insights needs to have certain fields.

For example: `sample_jti_grpc_netconf_config.py`

```
# Sensor path of JTI/GRPC/NETCONF sensor, for example, '/interface/' for GRPC
# openconfig sensor path, '/junos/system/linecard/interface/' for JTI
# interface sensor, 'show interface terse' for NETCONF interface sensor.
SENSOR_NAME = ''

# We support two sensor types: device_entity and device.
# device_entity means that the data output for this sensor is for entity of the
# device, e.g. device interface data, device FAN data.
# device means that the data output for this sensor is for the device itself,
# such as device's system uptime or device's tcp stats
SENSOR_TYPE = ''

# This refers to which collection name user wants this data to be saved in
# mongodb
COLLECTION_NAME = ''

# Entity Type is a list of strings. It indicates what is this sensor's data
# for. If it reports interface data, ENTITY_TYPE=['interface'], if it reports
# both FAN and FPC data, ENTITY_TYPE=['FAN', 'FPC'], if it reports the device
# data, ENTITY_TYPE=['network_device']
ENTITY_TYPE = []

# METRIC_LIST is a list of metrics you want to monitor, Name will be the
# metric_name shown in AppFormix, CalculationMethod can be either value or
# rate. If you want AppFormix to calculate the rate of this metric, you should
# put 'rate', otherwise, put value(and we will not do any calculation)
METRIC_LIST = [
    {
        'Name': 'active-state-count',
        'Units': 'Count',
        'CalculationMethod': 'value'
    }
]
```

```

# We need a parse_data function, which takes in raw json_data that we get from the device
# and device OS version, that then returns the data AppFormix needs. The version
# variable is optional and we use it for parsing sensor data for different
# device OS versions. (Different device OS versions can result in different data
# formats for the same sensor).
# The return value of this function should be a dictionary, key will be the
# entity_types this sensor reports data for which matches with the ENTITY_TYPE
# specify above. The value should also be a dictionary, the keys is post_data.
# 'post_data' is a list of data we post to AppFormix, each entry is a
# dictionary, key is the metric name and value is the corresponding value. In
# each entry, there is a mandatory key called 'entity_name' which should be
# this data's corresponding entity_name(cannot include comma(',') in it) such as
# 'xe-0/0/0'(interface_name), 'Fan0'(fan name) or
# 'Routing Engine 0'(routing engine name). If this is a device data, entity_name
# should be 'network_device'
def parse_data(json_data, version=None):
    return {ENTITY_TYPE: {'post_data': []}}

# After you finish the config file, we need to add this sensor from AppFormix
# ansible playbook.
# Put your sensor config file in AppFormix ansible directory and add following
# variables in group_vars/all:
# appformix_control_plane_plugin_config_files:
#   - { config_file_name: 'config_file.py', type: 'type_of_sensor' }

# config_file.py should be the name of the sensor config file and
# type_of_sensor should be either 'GRPC' if it is a gRPC sensor, 'JTI' if it
# is a JTI sensor or 'NETCONF' if it is a NETCONF sensor.

# Then re-run AppFormix ansible playbook will add the new sensors into
# AppFormix. It will copy your config file to all the agents and then update
# certified_plugins/grpc_network_device_usage.json,
# certified_plugins/jti_network_device_usage.json and
# certified_plugins/netconf_commandline.json for you.

```

This sample configuration file indicates and describes all the required fields for a JTI, gRPC, and NETCONF configuration file. Contrail Insights Agent uses the `parse_data` function supplied in the configuration file to parse out the data received from the data. Then Contrail Insights pushes the output of the parser function to both `event_stream` and Contrail Insights DataManager.

Note that the input variable `version` is needed in some scenarios because data format might be different across different device OS versions. You can implement the `parse_data` function with multiple parsers based on the `version` variable input.

Enable Sensors Through Contrail Insights Ansible Playbook

After completing the configuration file, you need to add the sensor from the Contrail Insights Ansible playbook.

Copy your sensor configuration file to the Contrail Insights Ansible directory and add the following variables in `group_vars/all`:

```
appformix_control_plane_plugin_config_files:
  - { config_file_name: 'config_file.py', type: 'type_of_sensor' }
```

`config_file.py` should be the name of the sensor configuration file and `type_of_sensor` should be either "GRPC" if it is a gRPC sensor, "JTI" if it is a JTI sensor, or "NETCONF" if it is a NETCONF sensor.

Then re-run the Contrail Insights Ansible playbook to add the new sensors into Contrail Insights. Contrail Insights copies your configuration file to all the Agents and then updates `certified_plugins/grpc_network_device_usage.json` and `certified_plugins/jti_network_device_usage.json` and `certified_plugins/netconf_commandline.json` for you.

Add More Metrics to Existing JTI, gPRC, and NETCONF Sensors

Contrail Insights has several built-in JTI, gRPC, and NETCONF sensors. To add more metrics for these sensors, instead of creating a new `sensor_config` file, you modify the existing sensor configuration file for these sensors.

To add metrics to existing JTI and gRPC sensors:

1. Starting with AppFormix version 3.1, the built-in `sensor_config` file and the dependent files can be found in the `/opt/appformix/manager/tailwind_manager/` folder in every Agent. Copy these files from the Agent.
2. Update the `sensor_config` file by adding more metrics, then modifying the `parse_data` function to parse data for these new metrics.
3. Place the updated files in your Ansible directory and re-run the playbook with `appformix_control_plane_plugin_config_files` defined correctly in `group_vars/all`.

NOTE: If your AppFormix version is earlier than 3.1, send an email to <mailto:appformix-support> and request the sensor configuration file for built-in JTI and gRPC sensors.

Release History Table

Release	Description
3.3.0	Starting with Contrail Insights version 3.3.0, NETCONF custom command support is added.

RELATED DOCUMENTATION

No Link Title
No Link Title
No Link Title

Remote Hosts

IN THIS SECTION

- [Configure a Remote Host in Contrail Insights for Monitoring | 142](#)
- [Packages Needed for Remote Host Monitoring | 143](#)

Contrail Insights can monitor hosts remotely without installing Contrail Insights Agent on the host. Such hosts can be configured using Ansible playbooks. Metric collection is supported using SNMP (version 2c or 3) and Intelligent Platform Management Interface (IPMI).

Configure a Remote Host in Contrail Insights for Monitoring

Set the variable `appformix_plugins` with an enabled plug-in for remote host monitoring. Some certified plug-ins already ship out of box and are defined in the **certified_plugins** directory. This variable must be run at the time of Contrail Insights Platform installation. For example:

```
appformix_plugins:
  - { plugin_info: 'certified_plugins/snmp_host_usage.json' }
  - { plugin_info: 'certified_plugins/ipmi_host_usage.json' }
```

During installation a `remote_host` tag needs to be defined in the inventory. For each host in this group, configuration variables need to be specified in the `inventory/hosts`. Refer to the [Create Ansible Inventory](#) topic for instructions on which step of the installation these variables need to be set. See ["Contrail Insights Installation and Configuration for OpenStack" on page 31](#).

The following configurations can be enabled for remote host monitoring.

- Host with only SNMP version 2c.
- Host with only SNMP version 3 and its required parameters.
- Host with only IPMI metrics.
- Host with IPMI metrics and SNMP version 2c metrics.
- Host with IPMI metrics and SNMP version 3 metrics.

The following sample inventory illustrates how to configure the above cases.

```
[remote_host]
  host1 snmp_ip='host1_snmp_ip' snmp_version='2c' snmp_community='public'
  host2 snmp_ip='host2_snmp_ip' snmp_version='3' snmp_level='authnoPriv' snmp_user='user1'
snmp_protocol='MD5' snmp_pwd='pass1'
  host3 ipmi_ip='host3_ipmi_ip' ipmi_user='user1' ipmi_pwd='pass1'
  host4 ipmi_ip='host4_ipmi_ip' ipmi_user='user1' ipmi_pwd='pass1' snmp_ip='host4_snmp_ip'
snmp_version='2c' snmp_community='public'
  host5 ipmi_ip='host5_ipmi_ip' ipmi_user='user1' ipmi_pwd='pass1' snmp_ip='host5_snmp_ip'
snmp_version='3' snmp_level='authnoPriv' snmp_user='user1' snmp_protocol='MD5' snmp_pwd='pass1'
```

The following sample inventory illustrates how to configure the devices with variables that share common values for all hosts. Refer to [Ansible inventory documentation](#) for further details.

```
[remote_host]
  host1 snmp_ip='host1_snmp_ip'
  host2 snmp_ip='host2_snmp_ip'

[remote_host:vars]
  snmp_version='2c'
  snmp_community='public'
```

Packages Needed for Remote Host Monitoring

The SNMP and IPMI plug-ins are executed by Contrail Insights Agent running on a host in the `appformix_platform` aggregate. Some packages must be present on such hosts to ensure the plug-in execution succeeds. The Contrail Insights playbooks will handle the installation of the packages if the following flag is set in the `group_vars/all` at the time the Contrail Insights Platform is installed.

```
appformix_remote_host_monitoring_enabled: true
```

The following are the packages that are required by the SNMP and IPMI plug-ins running on the `appformix_platform` hosts.

For Ubuntu:

- `snmp`
- `snmp-mibs-downloader`
- `ipmitool`

For CentOS or Red Hat Enterprise Linux:

- `net-snmp`
- `net-snmp-utils`
- `ipmitool`

RELATED DOCUMENTATION

[Bare Host | 86](#)

Monitor NFX250 with Contrail Insights Agent

IN THIS SECTION

- [Prerequisites | 144](#)
- [Install Contrail Insights Agent on NFX250 | 145](#)
- [Verify Installation of Contrail Insights Agent on NFX250 | 146](#)
- [Configure Contrail Insights Manually for NFX250 | 146](#)
- [Add NFX250 as a Host | 147](#)
- [Add a Project for VNFs on the NFX250 | 148](#)
- [Add a VM on the NFX250 | 149](#)

This topic describes how to install Contrail Insights Agent on the Juniper Networks NFX250 Network Services Platform.

- In the Contrail Insights architecture, an *adapter* is used to discover and correlate virtual machines and containers in cloud environments, and configure elements in the Contrail Insights Platform.
- Contrail Insights is typically installed using Ansible. Currently, Ansible is not able to gather facts when connecting to a NFX host.
- Contrail Insights Agent currently supports `systemd` and `upstart` scripts to manage the daemon. In the following installation instructions, Agent is started manually.

Prerequisites

- Contrail Insights Platform running OpenStack Adapter for identity and authentication.
 - In the following instructions, commands executed on the Contrail Insights Platform use the prefix `appformix1$`.

- To install Contrail Insights Platform, see "[Contrail Insights Installation and Configuration for OpenStack](#)" on page 31.
- NFX250
 - NFX250 has multiple IP addresses and shells that can be accessed, such as the host and Junos Device Manager (JDM).
 - Commands executed on JDM shell use prefix `root@nfx1-jdm#`.
 - Commands executed on host (Linux KVM hypervisor) shell use `root@nfx1-host#`.
- Time must be correct on NFX250. Use Network Time Protocol (NTP) to synchronize Junos device time. Verify the result of `show system uptime` command is the same as the time of AppFormix-VM.

Install Contrail Insights Agent on NFX250

To install Contrail Insights Agent on NFX250:

1. Download and extract the Contrail Insights installation package from [software downloads](#).

```
contrail-insights-<version>.tar.gz
$ tar -xvf contrail-insights-<version>.tar.gz
```

2. Copy the installer script to the NFX250:

```
$ scp appformix-<version>/appformix_installer_wrl-7.0_<version>.sh
root@nfx1:/tmp/
```

3. Log in to the NFX250 and execute the script. When prompted, provide the log in credentials for `appformix_controller` to enable a secure tunnel connection between `appformix_platform_host` and NFX250.

```
$ssh root@nfx1
root@nfx1-jdm:~# chmod +x /tmp/appformix_installer_wrl-7.0_<version>.sh
root@nfx1-jdm:~# /tmp/appformix_installer_wrl-7.0_<version>.sh
<appformix_platform_host_ip_address> <appformix_platform_host_user>
[ssh_key_path port:default=9000 agent_port:default=42595]
```

Verify Installation of Contrail Insights Agent on NFX250

Run the following commands to verify Contrail Insights Agent is installed on the NFX250.

On Contrail Insights Platform node:

```
appformix1$ curl http://<nfx-ip>:42595/appformix/v1.0/status
```

On the NFX Hypervisor:

```
appformix1$ curl http://<appformix-ip>:9000/appformix/controller/v2.0/status
```

Configure Contrail Insights Manually for NFX250

Currently, there is not an adapter to discover the NFX250 and the virtual machines that execute on the device. See the following workflow and procedures to configure Contrail Insights manually.

Get an Authentication Token

To get an authentication token:

1. Substitute the values in the example file in Step 2 for an OpenStack user (UserName), project (ProjectName), and password (Password).
2. Set APPFORMIX_PLATFORM_HOST to the hostname or IP address of the Contrail Insights Platform host.

```
$ APPFORMIX_PLATFORM_HOST=myhost1
$ curl -sS -H "Content-type: application/json" -d@- http://${APPFORMIX_PLATFORM_HOST}:9000/
appformix/controller/v2.0/auth_credentials <<EOF
{
  "UserName": "user1",
  "ProjectName": "project1",
  "UserDomainName": "Default",
  "Password": "password",
  "AuthType": "openstack",
  "ProjectDomainName": "Default"
```

```
}
EOF
```

Output will look similar to the following:

```
{
  "Token": {
    "expiresAt": "2017-04-13T00:05:05.000000Z",
    "isAdmin": true,
    "isConfigAdmin": true,
    "issuedAt": "2017-04-12T23:05:05.000000Z",
    "tokenId":
    "gAAAAABY7rKh7PUBI3HUczx1P097qt05R3IsLBF1qztz4YPRMoXepYZGUKFwrChx9frjhixjafoMzphughHuQfz4
    EurAKGasAuI0e-Za01tDAWmBTbyNvBVxVipNAQGpJsFIE1in15Zux3oCs0KEHI2nyYWMrgI-
    m27BoxnUCG_ZnLWAVEvVDAM"
  }
}
```

3. Set `TOKEN` environment variable for subsequent commands:

```
$
TOKEN="gAAAAABY7rKh7PUBI3HUczx1P097qt05R3IsLBF1qztz4YPRMoXepYZGUKFwrChx9frjhixjafoMzphughHuQfz
4Eur
AKGasAuI0e-Za01tDAWmBTbyNvBVxVipNAQGpJsFIE1in15Zux3oCs0KEHI2nyYWMrgI-m27BoxnUCG_ZnLWAVEvVDAM"
```

Add NFX250 as a Host

To add the NFX250 as a host:

1. Enter the following values as shown in the example file in Step 2 of **Add NFX250 as a Host** section to add NFX250 as a host:

- **Name**—Name is displayed in the Dashboard and is the user-facing identifier.
- **HostName**—Hostname or IP address used to reach NFX250 host address.
- **ServerId**—Unique identifier for your NFX250 server.
- **Source**—Enter "bare-metal" as Source value to indicate this host was not configured by an adapter that discovered the host from a cloud management system (such as OpenStack or Kubernetes).

2. POST your file to /appformix/controller/v2.0/hosts.

```
$ curl -sS -H "Content-type: application/json" -H "X-Auth-Token: ${TOKEN}" -H "X-Auth-Type:
openstack" -d@- http://${APPFORMIX_PLATFORM_HOST}:9000/appformix/controller/v2.0/hosts <<EOF
{
  "Name": "nfx1",
  "HostName": "nfx1",
  "HostType": "kvm",
  "LinkCapacity": "10G",
  "Source": "bare-metal",
  "ServerId": "nfx1",
  "AgentBaseUrl": "http://nfx-ip:42595"
}
EOF
```

Enter the following values in the example file to add NFX250 as a host:

Add a Project for VNFs on the NFX250

All instances in Contrail Insights must belong to a project. To create a project in which VNF VMs are added, see the following steps:

1. Enter "bare-metal" as Source value as shown in Step 2 of **Add a Project for VNFs on the NFX250** section to indicate this project was not configured by an adapter that discovered the host from a cloud management system (such as OpenStack or Kubernetes).
2. POST to /appformix/controller/v2.0/projects.

```
$ curl -sS -H "Content-type: application/json" -H "X-Auth-Type: openstack" -H "X-Auth-Token: $
${TOKEN}" -d@- http://${APPFORMIX_PLATFORM_HOST}:9000/appformix/controller/v2.0/projects <<EOF
{
  "ProjectId": "nfx",
  "VirtualMachines": [],
  "Name": "nfx",
  "ProjectDomain": "default",
  "Source": "user"
}
EOF
```

Add a VM on the NFX250

The virtual machine (VM) has to be added manually if the `automatic_kvm_instance_discovery` flag is not set in Ansible during playbook installation.

To add a VM on the NFX250:

1. Run `virsh list` to determine the list of VMs on the NFX250. For example:

```
root@nfx1-host:~# virsh list
Id      Name                               State
-----
 3      vjunos0                           running
 5      vsrx                              running
 6      riverbed                          running
```

2. Configure the VM (instance) in Contrail Insights using the following values:

- InternalName
- Name
- VirtualMachineId
- ProjectId
- POST `/appformix/controller/v2.0/instances`

Example:

```
$ curl -sS -H "Content-type: application/json" -H "X-Auth-Type: openstack" -H "X-Auth-Token: ${TOKEN}" -d@- http://${APPFORMIX_PLATFORM_HOST}:9000/appformix/controller/v2.0/instances <
{
  "InternalName": "vjunos0",
  "Name": "vjunos0",
  "VirtualMachineId": "vjunos0",
  "ProjectId": "nfx",
  "ServerId": "nfx1",
  "InstanceType": "kvm",
  "MetaData": {
    "VolumeList": [],
    "FlavorId": ""
  }
}
```

```
}
}
```

Contrail Insights SDKs

The Contrail Insights software development kit (SDK) is a Python library that can be used to interact with the Contrail Insights Platform using the Contrail Insights REST API. The following steps describe how to install and use the Contrail Insights SDK.

1. Unpack the SDK TAR file.

```
$ tar -xvzf appformix-sdk.tar.gz
$ ls sdk/
appformix  README  requirements.txt  samples
```

2. Set HTTP proxy environment variables, if required for access to the Internet. Dependencies will be downloaded from the Internet in Step 4.

```
$ export http_proxy=<proxy URL>
$ export https_proxy=<proxy UR>
```

3. Create a virtual environment (venv) for running the SDK.

```
$ virtualenv venv
New python executable in /home/user/sdk/venv/bin/python
Installing setuptools, pip, wheel...done.
$ source venv/bin/activate
(venv) $
```

4. Download the Python packages required for the SDK.

```
(venv) $ cd sdk
(venv) $ pip install -r requirements.txt
Collecting requests[security]==2.8.1 (from -r requirements.txt (line 1))
...
```

```
Successfully installed Babel-2.4.0 Jinja2-2.9.6 MarkupSafe-1.0
PrettyTable-0.7.2 Werkzeug-0.12.1 aniso8601-1.2.0 asn1crypto-0.22.0 cffi-1.10.0
cryptography-1.8.1 debtcollector-1.13.0 enum34-1.1.6 flask-0.10.1 flask-restful-0.3.5
funcsigs-1.0.2 idna-2.5 ipaddress-1.0.18 iso8601-0.1.11 itsdangerous-0.24
keystoneauth1-2.19.0 monotonic-1.3 msgpack-python-0.4.8 ndg-httpsclient-0.4.2 netaddr-0.7.19
netifaces-0.10.5 oslo.config-3.24.0 oslo.i18n-3.15.0 oslo.serialization-2.18.0
oslo.utils-3.25.0 pbr-3.0.0 positional-1.1.1 pyOpenSSL-17.0.0 pyasn1-0.2.3 pycparser-2.17
python-dateutil-2.6.0 python-keystoneclient-2.0.0 python-novaclient-3.1.0 pytz-2017.2
requests-2.8.1 rfc3986-0.4.1 simplejson-3.10.0 stevedore-1.21.0 wrapt-1.10.10
```

5. Unset the HTTP proxy environment variables (if set in Step 2).

```
(venv) $ unset http_proxy
(venv) $ unset https_proxy
```

6. Set OpenStack variables required for Contrail Insights.

```
(venv) $ cd samples/
(venv) $ vi appformix_config.py
# OpenStack Credentials of user
OPENSTACK_USERNAME = <username of an OpenStack user>
OPENSTACK_PASSWORD = <password of OpenStack user>
OPENSTACK_PROJECT_NAME = <project that OpenStack user is a member of>
OPENSTACK_USER_DOMAIN_NAME = <OpenStack user domain or 'default'>
OPENSTACK_PROJECT_DOMAIN_NAME = <OpenStack project domain or 'default'>
# Host where Contrail Insights Controller is running
APPFORMIX_HOST = <IP or hostname of Contrail Insights Controller host>
# Port for Contrail Insights Controller
APPFORMIX_PORT = <Contrail Insights Controller port, default: 9000>
# Authentication Type
AUTH_TYPE = 'openstack'
```

The PagerDuty Configuration, Custom Notification Configuration and ServiceNow Configuration sections are optional.

7. Set the PYTHONPATH environment variable to include appformix directory.

```
(venv) $ export PYTHONPATH=$PYTHONPATH:<SDK_ROOT_DIR>/sdk/appformix
```

8. Edit and run the sample scripts in the samples directory.

Example: Script to get metrics for a host for a specific period.

```
(venv) $ cd samples
(venv) $ vi get_data.py - Edit 'start_time_str' and 'end_time_str' to the desired values
(venv) $ python get_data.py
Data for server ace96
{'entity_id': 'ace96', 'entity_type': 'host', 'api_version':
'u'v1.0', 'entries': [{u'metrics': {u'host': {u'memory': {u'usage':
12.8, u'page_fault': {u'rate': 18449.49}, u'page_in_out': {u'rate': 0},
u'swap': {u'usage': 0}, u'dirty': {u'rate': 9}}, u'disk':
{u'read_response_time': 0, u'write_response_time': 0, u'io': {u'read':
0, u'write': 0}, u'response_time': 0, u'usage': {u'bytes': 15.3,
u'percent': 1}}, u'network': {u'ingress': {u'bit_rate': 0.03, u'drops':
...

```

Example: Script to configure an alarm for CPU usage of hosts in an aggregate.

```
(venv) $ python configure_host_alarm.py
Configured alarm host_alarm_rule1 for aggregate appformix_platform

```

Example: Script to generate usage reports for all hosts and all projects.

```
(venv) $ cd samples
(venv) $ vi generate_reports.py - Edit 'start_time_str' and 'end_time_str' to the desired
values
(venv) $ python generate_reports.py
Usage report for all_projects written to file all_projects_2016-07-13_2016-07-14.json
Usage report for all_hosts written to file all_hosts_2016-07-13_2016-07-14.json

```

Example: Script to create a new host aggregate.

```
(venv)$ python create_host_aggregate.py
Created aggregate aggregate1 with ID 10a0853e-2636-11e7-b1a9-0242ac120002

```

RELATED DOCUMENTATION

No Link Title

Contrail Insights with SSL (HTTPS) Enabled

You can configure Contrail Insights to use HTTPS protocol for REST APIs. To do this, an SSL certificate and private key are provided as arguments.

Configure the following arguments in `inventory/group_vars/all`:

```
appformix_ssl_cert : path to SSL certificate file
appformix_ssl_key  : path to SSL private key file
```

Alternatively, an SSL certificate and private key can be provided as arguments when running the installation playbook as shown:

```
ansible-playbook -i inventory appformix_openstack.yml -e 'appformix_ssl_cert=<path/to/cert>' -e
'appformix_ssl_key=<path/to/ssl_key>'
```

Contrail Insights services uses this certificate to verify the server identity to clients accessing the HTTPS endpoints.

When HTTPS is enabled, the default port used is 9001. So the Contrail Insights Dashboard is accessible at `https://appformix_vip:9001`. See the ["Contrail Insights Port List" on page 96](#) for more details about the ports.

Service Monitoring Ansible Variables

IN THIS SECTION

- [Configure Cassandra Monitoring | 154](#)
- [Configure Ceph Monitoring | 154](#)
- [Configure Contrail Monitoring | 155](#)
- [Configure MySQL Monitoring | 156](#)
- [Configure RabbitMQ Monitoring | 158](#)
- [Configure ScaleIO Monitoring | 159](#)

To monitor services, you must configure where a service is running and credentials (if any) to connect to the service. You can provide the configuration in the Dashboard settings but you can also provide the configuration by defining Ansible variables. The Ansible playbook will configure Contrail Insights Platform with the information to monitor a service.

Configure Cassandra Monitoring

Cassandra monitoring is configured by the Ansible role `appformix_cassandra_config`. This Ansible role is applied to the `appformix_controller` group of hosts. Ansible will perform the configuration if the following variables are set as extra vars, group vars, and so on.

- **cassandra_username:** Cassandra username to access API.
- **cassandra_password:** Cassandra password to access API.
- **cassandra_host:** Hostname or IP address of Cassandra API server.

There are two optional variables that can be specified as well:

- **cassandra_cluster_name:** A name by which the Cassandra instance is displayed in the Dashboard. If not specified, this variable has a default value of `default_cassandra_cluster`.
- **cassandra_cluster_port:** Port used by the Cassandra service. The default value is 9042.

NOTE: The variable `cassandra_cluster_port` is available starting in release 3.1.0.

Configure Ceph Monitoring

Enable Ceph service monitoring by specifying hosts in the Ansible groups `ceph_monitor` and `ceph_osd`. As of Contrail Insights v2.12, the Ansible playbooks install Contrail Insights Agent on hosts in those groups to monitor both the host and the Ceph service running on that host.

To access the Ceph service, Contrail Insights Agent uses information from the Ceph configuration files located in standard directories on the Ceph monitor host: `/etc/ceph/ceph.conf`, `/etc/ceph/ceph.client.admin.keyring`. If the files are located in a non-default location, the location of these files can be specified by setting the `CEPH_CONF` and `CEPH_KEY` environment variables for the agent. Refer to the Contrail Insights Agent startup scripts.

An example portion of the hosts file to specify Ceph nodes in Ansible inventory:

```
[ceph_monitor]
ceph_monitor01
ceph_monitor02
ceph_monitor03

[ceph_osd]
ceph_osd01
ceph_osd02
ceph_osd03
ceph_osd04
```

Ceph Configuration Prior to Contrail Insights v2.12

For Contrail Insights v2.11 and earlier, Ceph service monitoring is performed by remotely accessing Ceph service from the Contrail Insights Platform host. Ansible configures Contrail Insights Platform to monitor Ceph if the following variables are set as extra vars, group vars, and so on.

Configure Contrail Monitoring

Contrail service monitoring is configured by the Ansible role `appformix_contrail_config`. This Ansible role is applied to the `appformix_controller` group of hosts. Ansible performs the configuration if the following variables are set as extra vars, group vars, and so on.

Configuration of Contrail uses the same OpenStack credentials as provided for Contrail Insights to access OpenStack services. The Ansible role reads the credentials from environment variables (for example, `OS_USERNAME`, `OS_PASSWORD`). Administrator credentials to the OpenStack cluster are a requirement. Contrail Insights connects to the analytics and configuration nodes of Contrail.

Since Contrail Insights v2.15, connections to Contrail are configured by providing complete URLs by which to access the analytics and configuration API services.

- **contrail_analytics_url:** URL for the Contrail analytics API. The URL should only specify the protocol, address, and optionally port.

For example:

```
http://contrail.example.com:8081
```


- **contrail_config_url:** URL for the Contrail configuration API. The URL should only specify the protocol, address, and optionally port.

For example:

```
http://contrail.example.com:8082
```

The following optional variable can be specified:

- **contrail_cluster_name:** A name by which the Contrail instance will be displayed in the Dashboard. If not specified, this variable has a default value of `default_contrail_cluster`.

Contrail Configuration Prior to Contrail Insights v2.15

In versions prior to Contrail Insights v2.15, configuration is specified as a single hostname on which both the analytics and configuration APIs can be accessed. Contrail Insights connects to port 8081 for the analytics API and port 8082 for the configuration API.

- **contrail_host:** Hostname or IP address of Contrail API server.

Configure MySQL Monitoring

As of Contrail Insights v2.12, MySQL service monitoring is performed by Contrail Insights Agent, preferably running on the MySQL host itself. Agent can monitor the MySQL database service running on a host, and the host's compute, network, storage, and operating system metrics. Contrail Insights Ansible playbooks install Contrail Insights Agent on hosts in host groups such as `appformix_controller`, `openstack_controller`, and `bare_host`. If MySQL is running on a host in one such group, then `AgentUrl` (see below) can be left blank, and the agent on the host will be used to monitor the MySQL database running locally on the same host. If MySQL is running on a host that is not already in the Ansible inventory, then add the host to the `bare_host` group.

Alternately, you can configure the MySQL cluster to be monitored by an Contrail Insights Agent running on a different host than the host that is running MySQL. Agent will collect metrics from MySQL remotely. For this option, configure the `AgentUrl` (see below) to a host in the `bare_host` host group in the Ansible inventory.

MySQL monitoring is configured by the Ansible role `appformix_mysql_config`. Contrail Insights Ansible playbooks configure Contrail Insights to monitor a MySQL cluster only if the following variables are defined:

- **appformix_mysql_username:** MySQL username to access database metrics.

- **appformix_mysql_password:** MySQL password to access database metrics.
- **appformix_mysql_hosts:** A list of MySQL host configuration dictionaries. Each entry in the list specifies to monitor MySQL running on a host in a MySQL cluster. Each dictionary must contain the following fields:
 - **MySQLHost:** Hostname or IP address of MySQL host.
 - **AgentUrl:** URL to Agent that monitors MySQL on this host. Contrail Insights Agent can monitor more than one MySQL instance. If an empty string is specified, then Contrail Insights will use an agent running locally on the MySQL host. Agent listens on port 42595. An example URL is `http://hostname:42595/`.

There are two optional variables that can be specified as well:

- **appformix_mysql_cluster_name:** A name by which the MySQL cluster is displayed in the Dashboard. If not specified, this variable has a default value of `default_mysql_cluster`.
- **appformix_mysql_port:** MySQL port. If not specified, this variable has a default value of 3306.

Example configuration for MySQL in `group_vars/all`:

```
appformix_mysql_hosts:
  - { "MySQLHost": "10.87.68.94", "AgentUrl": "" }
  - { "MySQLHost": "10.87.68.89", "AgentUrl": "" }
appformix_mysql_username: "mysql_user1"
appformix_mysql_password: "secret"
```

MySQL Configuration Prior to Contrail Insights v2.12

For v2.11 and earlier, MySQL service monitoring is performed by remotely accessing MySQL service from the Contrail Insights Platform host. Ansible configures Contrail Insights Platform to monitor MySQL if the following variables are set as extra vars, group vars, and so on.

- **mysql_username:** MySQL username to access database metrics.
- **mysql_password:** MySQL password to access database metrics.
- **mysql_hosts:** A list of hosts separated by commas. For example, `1.1.1.1,2.2.2.2`. If MySQL is running in clustered mode, then Contrail Insights monitors each host in the cluster.

There are two optional variables that can be specified as well:

- **mysql_cluster_name:** A name by which the MySQL cluster is displayed in the Dashboard. If not specified, this variable has a default value of `default_mysql_cluster`.

- **mysql_port:** MySQL port. If not specified, this variable has a default value of 3306.

Configure RabbitMQ Monitoring

As of Contrail Insights v2.12, RabbitMQ service monitoring is performed by Contrail Insights Agent, preferably running on the RabbitMQ host itself. Agent can monitor the RabbitMQ service running on a host, and the host's compute, network, storage, and operating system metrics. Contrail Insights Ansible playbooks install Contrail Insights Agent on hosts in host groups such as `appformix_controller`, `openstack_controller`, and `bare_host`. If RabbitMQ is running on a host in one such group, then `AgentUrl` (see below) can be left blank, and the agent on the host will be used to monitor the RabbitMQ service running locally on the same host. If RabbitMQ is running on a host that is not already in the Ansible inventory, then add the host to the `bare_host` group.

Alternately, you can configure the RabbitMQ service to be monitored by an Contrail Insights Agent running on a different host than the host that is running RabbitMQ. Agent will collect metrics from RabbitMQ remotely. For this option, configure the `AgentUrl` (see below) to a host in the `bare_host` host group in the Ansible inventory.

RabbitMQ monitoring is configured by the Ansible role `appformix_rabbit_config`.

Contrail Insights Ansible playbooks configure Contrail Insights to monitor RabbitMQ only if the following variables are defined:

- **appformix_rabbitmq_username:** RabbitMQ username to access monitoring API.
- **appformix_rabbitmq_password:** RabbitMQ password to access monitoring API.
- **appformix_rabbitmq_nodes:** A list of RabbitMQ node configuration dictionaries. Each dictionary must contain the following fields:
 - **RabbitNode:** RabbitMQ node name. Node name must match the configuration in the RabbitMQ cluster, as seen in the output of `rabbitmqctl status`.
 - **RabbitUrl:** URL to RabbitMQ monitoring API. For example, http://rabbit_node1:15672.
 - **AgentUrl:** URL to Contrail Insights Agent that monitors this RabbitMQ node. Contrail Insights Agent can monitor more than one RabbitMQ node. If an empty string is specified, then Contrail Insights uses an Agent running locally on the RabbitMQ node. Agent listens on port 42595. An example URL is `http://hostname:42595/`.

There is one optional variable that can be specified as well:

- **appformix_rabbitmq_cluster_name:** A name by which the RabbitMQ cluster is displayed in the Dashboard. If not specified, this variable has a default value of `default_rabbit_cluster`.

Example configuration for RabbitMQ in `group_vars/all`:

```
appformix_rabbitmq_username: "rabbit_user1"
appformix_rabbitmq_password: "secret"
appformix_rabbitmq_nodes:
- { "RabbitUrl": "http://rabbit_node1.example.com:15672",
    "RabbitNode": "rabbit@node1",
    "AgentUrl": "" }
- { "RabbitUrl": "http://rabbit_node1.example.com:15672",
    "RabbitNode": "rabbit@node2",
    "AgentUrl": "" }
```

RabbitMQ Configuration Prior to Contrail Insights v2.12

For Contrail Insights v2.11 and earlier, RabbitMQ service monitoring is performed by remotely accessing RabbitMQ service from the Contrail Insights Platform host. Ansible configures Contrail Insights Platform to monitor RabbitMQ if the following variables are set as extra vars, group vars, and so on.

- **rabbit_username:** RabbitMQ username to access monitoring API.
- **rabbit_password:** RabbitMQ password to access monitoring API.

Configure ScaleIO Monitoring

ScaleIO monitor is configured by the Ansible role `appformix_scaleio` when the variable `scaleio_cluster` is defined. This Ansible role is applied to the `appformix_controller` group of hosts. Ansible will perform the configuration if the `scaleio_cluster` variable is set as extra vars, group vars, and so on. The `scaleio_cluster` variable is a dictionary with the following required keys:

- **host:** ScaleIO API gateway hostname or IP address.
- **username:** ScaleIO username to access API.
- **password:** ScaleIO password to access API.

Optionally, the following keys can also be defined in the `scaleio_cluster` dictionary.

- **name:** Name of this ScaleIO cluster. If not specified, this variable has a default value of `default_scaleio_cluster`.
- **port:** ScaleIO API gateway port number. If not specified, this variable has a default value of 80.

For example, in `group_vars/all`, the `scaleio_cluster` dictionary can be minimally defined as follows:

```
scaleio_cluster:
  host: my_scaleio_host.example.com
  username: my_scaleio_username
  password: my_scaleio_password
```

RELATED DOCUMENTATION

| No Link Title

OpenStack Services Monitoring Using Service Group Profiles

IN THIS SECTION

- [Profile Overview | 161](#)
- [Setting Up a Profile | 161](#)
- [Adding a Profile to Ansible | 163](#)
- [Adding Multiple Profiles to Ansible | 164](#)

A service group tests the connectivity and latency between the Contrail Insights Agent and one or more endpoints. A service group consists of a list of endpoints with certain specifications. There are two ways to add service groups: either by using Ansible or the Contrail Insights Dashboard. To add service groups from the Dashboard, see [Endpoint Monitoring with Service Groups](#).

Profile Overview

In the directory **agent/tools/ansible/profiles/**, there are five profiles, each pertaining to an OpenStack service. The prefix of each file is the name of the OpenStack service; either `cinder`, `glance`, `keystone`, `neutron`, or `nova`. The suffix is `*_default_service_profile.json.j2`. For example, the profile for the OpenStack service "Glance" is named `glance_default_service_profile.json.j2`.

The default layout of the "Glance" profile is shown. The other profiles have an identical layout, just with the corresponding OpenStack service listed:

```
{
  "ServiceGroupName": "AppformixGlanceServiceGroup",
  "Endpoints": [
    {
      "Url": "{{ glance_url }}",
      "EndpointName": "glanceEndpoint",
      "Method": "GET",
      "Interval": 2
    }
  ],
  "ServiceGroupId": "GlanceServiceGroupId",
  "RefreshTokenData": {
    "RefreshToken": "False",
    "Username": "admin",
    "AuthType": "openstack",
    "Password": "",
    "AuthUrl": "",
    "Project": ""
  }
}
```

Setting Up a Profile

Profiles support unauthenticated and authenticated endpoints.

Adding an Unauthenticated Endpoint

To add an unauthenticated endpoint, simply add the variable that the `Url` key is mapped to in your `group_vars`. Verify this variable is mapped to a working endpoint. For example, in the `group_vars`, if you are using the "Glance" profile, add the `glance_url` variable as shown:

```
glance_url: "http://0.0.0.0:9292"
```

Adding an Authenticated Endpoint

To add an endpoint that needs authentication, a `RefreshToken` is required. A refresh token enables access to endpoints that require authentication and retains that access by getting a new token when the current one is about to expire.

To procure a refresh token, set the `RefreshToken` field in the `RefreshTokenData` dictionary to be `True`. Then provide `Username`, `Password`, and `AuthUrl` in the same `RefreshTokenData`.

GET Example:

```
{
  "ServiceGroupName": "AppformixGlanceServiceGroup",
  "Endpoints": [
    {
      "Url": "{{ glance_url }}",
      "EndpointName": "glanceEndpoint",
      "Method": "GET",
      "Interval": 2
    }
  ],
  "ServiceGroupId": "GlanceServiceGroupId",
  "RefreshTokenData": {
    "RefreshToken": "True",
    "Username": "admin",
    "AuthType": "openstack",
    "Password": "password",
    "AuthUrl": "auth_url",
    "Project": ""
  }
}
```

POST Example:

```
{
  "ServiceGroupName": "AppformixGlanceServiceGroup",
  "Endpoints": [
    {
      "Url": "{{ glance_url }}",
      "EndpointName": "glanceEndpoint",
      "Method": "POST",
      "Interval": 2,
      "Data": "{ \"AuthType\": \"openstack\", \"UserName\": \"admin\", \"Password\": \"password\" }"
    }
  ],
  "ServiceGroupId": "GlanceServiceGroupId",
  "RefreshTokenData": {
    "RefreshToken": "True",
    "Username": "admin",
    "AuthType": "openstack",
    "Password": "password",
    "AuthUrl": "auth_url",
    "Project": ""
  }
}
```

Adding a Profile to Ansible

Using Ansible, a profile corresponding to an OpenStack service can be added to the Contrail Insights Dashboard during installation.

To do this, add the variable `appformix_service_connectivity_profiles` to your `group_vars`. The variable should be mapped to a list of dictionaries. Each dictionary in the list should only contain one key and one value. The key should always be `connectivity_profiles`. The value should be the path of the profile you want to be added to the Contrail Insights Dashboard during installation. Following is an example:

```
appformix_service_connectivity_profiles: [{ connectivity_profiles: '../../../profiles/
glance_default_service_profile.json.j2' }]
```


NOTE: In the example, the profile path is prefaced by `../../../../`. This is a necessary addition to the path of any profiles in the `profiles/` directory. It is due to how the Contrail Insights Ansible structure is setup.

After the `url` and `appformix_service_connectivity_profiles` variables are added to `group_vars`, the specified profile will be added to the Contrail Insights Dashboard during installation.

Adding Multiple Profiles to Ansible

To add multiple profiles, simply repeat the previous steps for as many profiles as needed:

1. Add `url` to `group_vars`.
2. Add dictionary to `appformix_service_connectivity_profiles` variable in `group_vars`.

RELATED DOCUMENTATION

No Link Title

[Service Monitoring Ansible Variables | 153](#)

Ansible Configuration Variables

IN THIS SECTION

- [Network Configuration | 165](#)
- [Proxy Configuration | 166](#)
- [Log File Configuration | 167](#)
- [Timeout Configuration | 167](#)

The following variables configure aspects of the Contrail Insights installation. We recommend these variables be defined in the `inventory/group_vars/all` file of your inventory. Refer to [Ansible inventory documentation](#) for more information about Ansible variables.

Network Configuration

[Table 12 on page 165](#) describes Ansible configuration variables for network configuration.

Table 12: Ansible Configuration Variables

Variable	Description
<code>appformix_network_ipv4_subnet</code>	Contrail Insights services on the Contrail Insights Platform host run in Docker containers. The Docker containers communicate over a private Docker network named AppFormixNetwork. The <code>appformix_network_ipv4_subnet</code> variable specifies the subnet in CIDR notation. For example, <code>172.18.0.0/16</code> . Use this variable to specify a subnet in case the default subnet conflicts with an existing network subnet.
<code>appformix_controller_port_http</code>	Port on which appformix-controller container listens for HTTP requests.
<code>appformix_controller_port_https</code>	Port on which appformix-controller container listens for HTTPS requests.
<code>appformix_dashboard_port_http</code>	Port on which appformix-dashboard container listens for HTTP requests.
<code>appformix_dashboard_port_https</code>	Port on which appformix-dashboard container listens for HTTPS requests.
<code>appformix_datamanager_port_http</code>	Port on which appformix-datamanager container listens for HTTP requests.

Table 12: Ansible Configuration Variables (Continued)

Variable	Description
<code>appformix_datamanager_port_https</code>	Port on which appformix-datamanager container listens for HTTPS requests.
<code>appformix_openstack_adapter_port_http</code>	Port on which appformix-openstack-adapter container listens for HTTP requests.
<code>appformix_openstack_adapter_port_https</code>	Port on which appformix-openstack-adapter container listens for HTTPS requests.

Proxy Configuration

Contrail Insights Platform services execute inside of Docker containers. Proxy environment variables on the host will not be set inside of the container. If a proxy is required (for example, to contact external notification services, such as PagerDuty or Service Now), then proxy configuration can be defined in the following variables to configure the environment inside of the container.

The `appformix_controller` uses HTTP/HTTPS for communication with other Contrail Insights components (for example, Contrail Insights Agent) and platform integrations (for example, OpenStack, Kubernetes). If `http_proxy` or `https_proxy` is configured, then an appropriate value for `no_proxy` should be set. The `no_proxy` variable can contain a comma-separated list of IP addresses, hostnames, and subnets in CIDR notation (for example, `192.0.2.0/24`). [Table 13 on page 166](#) describes Ansible configuration variables for proxy configuration.

Table 13: Ansible Configuration Variables for Proxy Configuration

Variable	Description
<code>appformix_controller_env_http_proxy</code>	Value for <code>http_proxy</code> environment variable inside of <code>appformix_controller</code> container.
<code>appformix_controller_env_https_proxy</code>	Value for <code>https_proxy</code> environment variable inside of <code>appformix_controller</code> container.

Table 13: Ansible Configuration Variables for Proxy Configuration (Continued)

Variable	Description
appformix_controller_env_no_proxy	Value for no_proxy environment variable inside of appformix_controller container.

Log File Configuration

Contrail Insights maintains multiple log files for services on the Contrail Insights Platform. Each log file is rotated independently according to size and time. Rotated files have a numeric suffix appended to the filename, starting at 1. A configurable number of rotated files are stored for each log file.

Log files are rotated when the size of the files exceeds a configurable limit. Optionally, log files can be rotated at the start of each day. When daily rotation is configured, a log file can still be rotated more than once per day if the size limit is exceeded. [Table 14 on page 167](#) describes Ansible configuration variables for log file configuration.

Table 14: Ansible Configuration Variables for Log File Configuration

Variable	Description
appformix_log_rotate_daily	If set to 1, log files are rotated daily. Default is 0.
appformix_log_rotate_max_size_in_mb	Size in megabytes (MB) at which a log file is rotated. Default is 10 MB.
appformix_log_rotate_max_count	Number of times a log file is rotated before being removed. Default is 5.

Timeout Configuration

[Table 15 on page 168](#) describes the Ansible configuration variable to specify timeouts for connections between Contrail Insights services.

Table 15: Ansible Variables for Timeout Configuration

Variable	Description
appformix_openstack_adapter_timeout	Timeout for connections between the appformix_controller and openstack_adapter.

RELATED DOCUMENTATION

Contrail Insights General Requirements 2
Contrail Insights Port List 96