Cisco ACI Installation Guide for Red Hat OpenStack Using the OpenStack Platform 13 Director

First Published: 2019-04-19

Last Modified: 2020-08-31

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2019-2020 Cisco Systems, Inc. All rights reserved.

CONTENTS

PREFACE

CHAPTER 1

CHAPTER 2

CHAPTER 3

Preface

This preface includes the following sections:

Audience

This guide is intended primarily for data center administrators with responsibilities and expertise in one or more of the following:

- Virtual machine installation and administration
- Server administration
- Switch and network administration
- Cloud administration

Document Conventions

Command descriptions use the following conventions:

Convention	Description
bold	Bold text indicates the commands and keywords that you enter literally as shown.
Italic	Italic text indicates arguments for which the user supplies the values.
[x]	Square brackets enclose an optional element (keyword or argument).
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical bar indicate an optional choice.

Convention	Description
{x | y}	Braces enclosing keywords or arguments separated by a vertical bar indicate a required choice.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
`variable`	Indicates a variable for which you supply values, in context where italics cannot be used.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.

Examples use the following conventions:

Convention	Description
`screen font`	Terminal sessions and information the switch displays are in screen font.
`boldface screen font`	Information you must enter is in boldface screen font.
italic screen font	Arguments for which you supply values are in italic screen font.
< >	Nonprinting characters, such as passwords, are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

This document uses the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Warning IMPORTANT SAFETY INSTRUCTIONS

This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS

Related Documentation

Cisco Cloud APIC Documentation

The Cisco Cloud APIC documentation is available at the following URL: https://www.cisco.com/c/en/us/support/cloud-systems-management/cloud-application-policy-infrastructure-controller/tsd-products-support-series-home.html

Cisco Application Policy Infrastructure Controller (APIC) Documentation

The following companion guides provide documentation for Cisco APIC:

- *Cisco APIC Getting Started Guide*
- *Cisco APIC Basic Configuration Guide*
- *Cisco ACI Fundamentals*
- *Cisco APIC Layer 2 Networking Configuration Guide*
- *Cisco APIC Layer 3 Networking Configuration Guide*
- *Cisco APIC NX-OS Style Command-Line Interface Configuration Guide*
- *Cisco APIC REST API Configuration Guide*
- *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*
- *Cisco ACI Virtualization Guide*
- *Cisco Application Centric Infrastructure Best Practices Guide*

All these documents are available at the following URL: http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html

Cisco Application Centric Infrastructure (ACI) Documentation

The broader Cisco ACI documentation is available at the following URL: http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html.

Cisco Application Centric Infrastructure (ACI) Simulator Documentation

The Cisco ACI Simulator documentation is available at http://www.cisco.com/c/en/us/support/cloud-systems-management/application-centric-infrastructure-simulator/tsd-products-support-series-home.html.

Cisco Nexus 9000 Series Switches Documentation

The Cisco Nexus 9000 Series Switches documentation is available at http://www.cisco.com/c/en/us/support/switches/nexus-9000-series-switches/tsd-products-support-series-home.html.

Cisco ACI Virtual Edge Documentation

The Cisco Application Virtual Edge documentation is available at https://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html.

Cisco ACI Virtual Pod Documentation

The Cisco Application Virtual Pod (vPod) documentation is available at https://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html.

Cisco Application Centric Infrastructure (ACI) Integration with OpenStack Documentation

Cisco ACI integration with OpenStack documentation is available at http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html.

Documentation Feedback

To provide technical feedback on this document, or to report an error or omission, please send your comments to apic-docfeedback@cisco.com. We appreciate your feedback.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation* at: http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html

Subscribe to *What’s New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation as an RSS feed and delivers content directly to your desktop using a reader application. The RSS feeds are a free service.

CHAPTER 1

New and Changed Information

New and Changed Information

The following table provides an overview of the significant changes to this guide up to this current release. The table does not provide an exhaustive list of all changes that are made to the guide or of the new features up to this release.

Table 1: New Features and Changed Behavior

Cisco APIC Release Version	Feature	Description	Where Documented
5.0(1)	Ironic bare metal support	The Cisco Application Centric Infrastructure (ACI) OpenStack plug-in now supports Ironic, the OpenStack feature that provisions bare metal machines.	OpenStack Configuring Ironic, on page 43
4.2(3.20191220) tripleo-ciscoaci-13.0-749.noarch.rpm version of the OpenStack plug-in	Support for the Transport Layer Security (TLS) Everywhere	This feature allows you to use the Cisco ACI OpFlex plug-in with Red Hat OpenStack 13 deployed with TLS on internal endpoints (EPs), external EPs, or both.	Deploying with TLS, on page 35
4.2(1)	Support for an upstream Docker registry.	The feature allows you to use an upstream- or satellite-based Docker Registry when deploying Red Hat OpenStack.	Cisco Application Policy Infrastructure Controller OpenStack Plug-in Release Notes, Release 4.2(1)

Cisco APIC Release Version	Feature	Description	Where Documented
4.2(1)	Run Bond Watch Service from Heat template	When integrating Cisco UCS B-Series to work with Cisco ACI and OpenStack orchestration, you can run the bond watch service by enabling a parameter to the Heat template. The ACIEnableBondWatchService parameter enables the `apic-bond-watch service` when it is added to the Cisco ACI resources template and set to true.	• Parameters for the Cisco ACI Environment, on page 26 • Run the Bond Watch Service, on page 38
4.2(1)	OpenStack Director deployments can now use certificate-based authentication with Cisco ACI.	Add the ACIApicCertName and ACIApicPrivateKey parameters to the Heat template to enable certificate-based authentication for Cisco ACI.	Parameters for the Cisco ACI Environment, on page 26
4.2(1)	Automatic purge of the tenant from Cisco ACI when a project is deleted from OpenStack	Setting the parameter AciKeystoneNotificationPurge to True enables Keystone notification purge.	Parameters for the Cisco ACI Environment, on page 26
N/A	Configuration of UCS B-Series for Cisco ACI and OpenStack Orchestration	Added appendix about configuring UCS B-Series for Cisco ACI and OpenStack orchestration.	Configuring UCS B-Series, on page 37
4.1(1)	--	This guide was released.	--

CHAPTER 2

Installation

This chapter contains the following sections:

Cisco ACI with OpenStack Using the OpenStack Platform 13 Director

Cisco Application Centric Infrastructure (ACI) is a comprehensive policy-based architecture that provides an intelligent, controller-based network switching fabric. This fabric is designed to be programmatically managed through an API interface that can be directly integrated into multiple orchestration, automation, and management tools, including OpenStack. Integrating Cisco ACI with OpenStack allows dynamic creation of networking constructs to be driven directly from OpenStack requirements, while providing additional visibility within the Cisco Application Policy Infrastructure Controller (APIC) down to the level of the individual virtual machine (VM) instance.

OpenStack defines a flexible software architecture for creating cloud-computing environments. The reference software-based implementation of OpenStack allows for multiple Layer 2 transports including VLAN, GRE, and VXLAN. The Neutron project within OpenStack can also provide software-based Layer 3 forwarding. When utilized with Cisco ACI, the Cisco ACI fabric provides an integrated Layer 2 and Layer 3 VXLAN-based overlay networking capability that can offload network encapsulation processing from the compute nodes onto the top-of-rack or Cisco ACI leaf switches. This architecture provides the flexibility of software overlay networking along with the performance and operational benefits of hardware-based networking.

The Cisco ACI OpenStack plug-in can be used in either ML2 or GBP mode. In Modular Layer 2 (ML2) mode, a standard Neutron API is used to create networks. This is the traditional way of deploying VMs and services in OpenStack. In Group Based Policy (GBP) mode, a new API is provided to describe, create, and deploy applications as policy groups without worrying about network-specific details. Keep in mind that mixing GBP and Neutron APIs in a single OpenStack project is not supported. and For more information, see the *OpenStack Group-Based Policy User Guide* at:

http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/openstack/b_OpenStack_Group-Based_Policy_User_Guide.html

Requirements and Prerequisites for Cisco ACI with OpenStack Using OSP Director

- Target audience: You must have working knowledge of Linux, Red Hat OpenStack distribution, the Cisco Application Centric Infrastructure (ACI) policy model and GUI-based Cisco Application Policy Infrastructure Controller (APIC) configuration. You must also be familiar with OpenStack architecture and deployment.

- Cisco ACI fabric: You must have a Cisco ACI fabric that is installed and initialized with the minimum supported version that is documented in the Cisco ACI Virtualization Compatibility Matrix. For basic guidelines on initializing a new Cisco ACI fabric, see the Cisco ACI Fabric Initialization Example, on page 13, on page.

  Note For communication between multiple leaf pairs, the fabric must have a BGP route reflector that is enabled to use an OpenStack external network.

- When using bonded fabric interface with a virtual port channel (vPC), adding the `ovs_bond` for the fabric interface is not supported. That is because it must be added as a single interface to the Open vSwitch (OVS) bridge. You must set the `type` to `linux_bond` for aggregating the fabric interfaces. Here is a rough example of how the fabric interface must be created in the `nic-config` templates:

```
type: ovs_bridge
              name: {get_input: bridge_name}
              mtu: 1500
              members:
                -
                  type: linux_bond
                  name: bond1
                  ovs_options: {get_param: BondInterfaceOvsOptions}
                  mtu: 1600
                  members:
                    -
                      type: interface
                      name: nic1
                      primary: true
                      mtu: 1600
                    -
                      type: interface
                      name: nic2
                      mtu: 1600
```

- When using bonding, only 802.3ad is supported.

- When deploying with UCS-B series, only dual vNICs with bonding is supported for the fabric interface for redundancy.

  Note Do not use a single vNIC with hardware failover.

- In the Cisco APIC GUI, disable the OpFlex authentication in the fabric. Make sure "To enforce Opflex client certificate authentication for GOLF and Linux." is not checked in **System** > **System Settings** > **Fabric Wide Setting** > **Fabric Wide Setting Policy** pane.
- When you delete the Overcloud Heat stack, the Overcloud nodes are freed, but the virtual machine manager (VMM) domain remains present in Cisco APIC. The VMM appears in Cisco APIC as a stale VMM domain along with the tenant unless you delete the VMM domain manually.

  Before you delete the VMM domain, verify that the stack has been deleted from the undercloud, and check that any hypervisors appearing under the VMM domain are no longer in the connected state. Once both of these conditions are met, then you can safely delete the VMM domain from Cisco APIC.

Related Documentation

For more information, see the *Director Installation and Usage, Red Hat OpenStack Platform 13* documentation on the Red Hat website.

Deploying OpFlex

This section describes how to install and configure the Cisco Application Centric Infrastructure (ACI) OpenStack Plug-in on a Red Hat OpenStack distribution.

These example steps were validated on OpenStack Platform 13 releases of Red Hat OpenStack. OpenStack systems can vary widely in how they are installed. Therefore, the examples provided may be used as a basis to be adapted to the specifics of your installation.

Follow the Red Hat OpenStack Platform Director installation document to prepare the OpenStack Platform Director and create the correct deployment and resource files.

For more information, see the Related Documentation, on page 5.

Preparing Cisco ACI for OpenStack Installation

Setting Up the Cisco APIC and the Network

This section describes how to set up the Cisco Application Policy Infrastructure Controller (APIC) and the network.

Refer to the Network Planning section of the OpenStack Platform Director documentation for network layout such as the one shown in the figure below.

For more information, see Related Documentation, on page 5.

Figure 1: Typical OpenStack platform topology

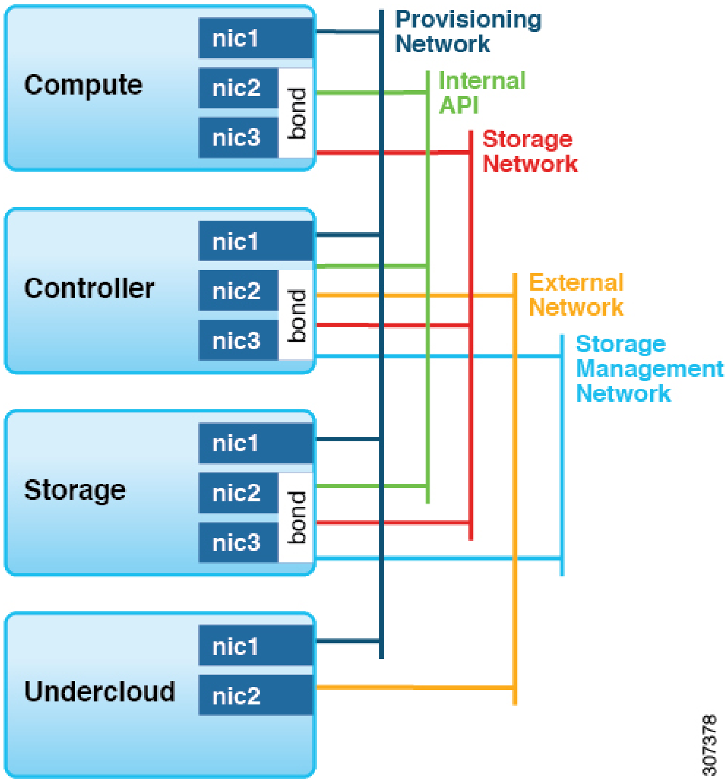
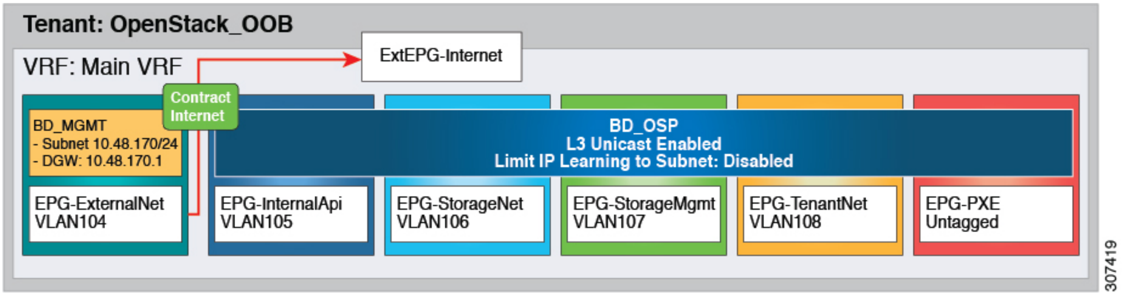


Figure 2: Typical topology for installation of Red Hat OpenStack Platform 13 with the Cisco ACI plug-in



- The PXE network must use a native VLAN. Because native VLAN typically is defined as a dedicated NIC on the OpenStack nodes, you can connect PXE network interfaces either to the Cisco ACI fabric or to a different switching fabric.
- All OpenStack Platform (OSP) networks except for PXE are in-band (IB) through Cisco Application Centric Infrastructure (ACI). The following VLANs are examples:
  - API: VLAN 10
  - Storage: VLAN 11
  - StorageMgmt: VLAN 12

  - Tenant: VLAN 13
  - External: VLAN 14
  - Cisco ACI Infra: VLAN 4093

- ExtEPG-Internet used in this example is the L3Out external EPG that allows connectivity to Internet for the OpenStack External Network. You also may need to provide external connectivity for the Internal API OpenStack network, according to your requirements.

To prepare Cisco ACI for in-band configuration you can use the physical domain and the static binding to the EPGs created for these networks. This involves creating the required physical domain and attachable access entity profile (AEP). Note that the infra VLAN should be enabled for the AEP. For more details, see the knowledge base article *Creating Domains, Attach Entity Profiles, and VLANs to Deploy an EPG on a Specific Port*.

Procedure

Step 1 Log in to the Cisco APIC GUI and create a VLAN pool for the VLANs required for OpenStack Platform installation.

a) On the menu bar, choose **Fabric** > **Access Policies** > **Pools** and right-click **VLAN** to create a VLAN pool.
b) In the **Name** field, enter the VLAN range namespace policy name. (For example, OSP13-infra.)
c) (Optional) In the **Description** field, enter the description of the VLAN range namespace policy.
d) In the **Encap Blocks** section, click on the + icon to enter the encap block range.
e) Click **SUBMIT**.

Step 2 Create an attachable entity profile and assign the above PhysDom to it. Also make sure **Enable Infra VLAN** is selected:

a) On the menu bar, choose **Fabric** > **Access Policies** > **Global Policies** and right-click **Attachable Access Entity Profile** to create an attachable access entity profile.
b) In the **Name** field, enter the name of the attachable access entity profile. (For example, OSP13-AEP.)
c) (Optional) In the **Description** field, enter the description of the attachable access entity profile.
d) Check the **Enable Infrastructure VLAN** check box to enable the infrastructure VLAN.
e) In the **Domains (VMM, Physical or External) To Be Associated To Interfaces:** section, click on the + icon, from the drop-down list, choose the domain profile and click **Update**.
f) Click **Next**.
g) Click **Finish**.

Step 3 Create a Physical Domain (PhysDom) and assign the VLAN pool to it.

a) On the menu bar, choose **Fabric** > **Access Policies** > **Physical and External Domains** and right-click **Physical Domains** to create a Physical Domain.
b) In the **Name** field, enter the name of the physical domain. (OSP13-phys).
c) In the **Associated Attachable Entity Profile** field, choose an associated attachable entity profile.
d) In the **VLAN Pool** field, choose a VLAN pool ([OSP13-infra-dynamic]).

   If VLAN is used as the encapsulation method between the OpenStack nodes and the Cisco ACI leaf switches, you need to choose a VLAN pool range according to the pool used from OpenStack Neutron networks.

e) Click **Submit**.

Step 4 In a separate tenant, you can also use Common to create an application profile. (For example, OSP-13.) Create the EPGs, bridge domains, and a VRF for the OSP Networks. If the PXE network is also going through Cisco ACI then also create EPG and BD for PXE (This is not shown in this example).

Step 5 Add static bindings (Paths) for the required VLANs. You have to expand the EPG to see the ":Static Binding Paths".

a) Make sure the physical domain you created is attached to this EPG. You can add the physical domain using **Application Profiles** > **EPG** > **EPG_name** > **Domains**.

b) On the menu bar, choose **Tenants** > **Tenant common** > **Application Profiles** > ***ACI-OSP13*** > **Application EPGs** > ***EPG API*** > **Static Binding Paths**.

Step 6 Make sure the PhysDom is attached to the EPG.

Note Cisco ACI needs to be provisioned for networks mentioned above except for Tenant, External and Floating IP network. This involves creating the required phys-doms and attached entity profile. Important thing to note is that Infra VLAN should be enabled for the attached entity profile.

Cisco ACI should now be ready for OpenStack deployment.

Setting Up Overcloud

You must follow the *Director Installation and Usage, Red Hat OpenStack Platform 13* document to prepare the OpenStack Platform 13 Director and create the correct deployment and resource files.

For more information, see the document on the Red Hat website. When following Chapter 5—"Configuring a Container Image Source"—note the registry address. You might need to prepare the custom NIC templates as required following the Red Hat documentation.

After you set up the OpenStack Plaform Director, you must install the Cisco Application Centric Infrastructure (ACI) TripleO orchestration before proceeding with deployment.

Preparing for Cisco ACI with OpFlex Agent

The following is a summary of steps required to install and enable Cisco Application Centric Infrastructure (ACI) OpFlex agent on the Overcloud nodes. The following sections explain the steps in detail.

- Modify the undercloud to include the necessary software packages.
- Add to the Neutron puppet manifests, which are part of Overcloud image.
- Add the OpFlex puppet manifests.
- Modify some files on the undercloud tripleO infrastructure.
- Create a HEAT environment file to provide Cisco ACI-related parameter values.
- After making the preceding modifications, you can provision Overcloud using the **openstack overcloud deploy** command and add the new environment file to the **openstack overcloud deploy** command.

Preparing Undercloud for Cisco ACI with OpFlex Orchestration

This section describes how to install the integration package for Cisco Application Centric Infrastructure (ACI) with OpFlex Orchestration.

Note In previous versions, the plug-in consisted of only an RPM file, and installing the RPM file created a yum repository with plug-in packages. Users were required to run `/opt/ciscoaci-tripleo-heat-templates/tools/build_openstack_aci_containers.py` to build the containers, but did not need to add the option `-z`.

Procedure

Step 1 Log in to undercloud as user `stack`.

Step 2 Download the latest Cisco ACI OSP (tripleo-ciscoaci) RPM and the corresponding plug-in tarball (openstack-ciscorpms-repo) from Cisco.com and place them on the OpenStack Platform Director.

Step 3 Install the RPM. This action installs the dependencies. If the RPM is installed using the `rpm` command, some dependency may need to be manually installed.

Example:

```
$ sudo yum --nogpgcheck localinstall <rpm file>
```

Step 4 Create the Cisco ACI containers by running the following command: **`/opt/ciscoaci-tripleo-heat-templates/tools/build_openstack_aci_containers.py`** and pointing it to the downloaded plug-in tarball. For example:

```
/opt/ciscoaci-tripleo-heat-templates/tools/build_openstack_aci_containers.py -z
/home/stack/openstack-ciscorpms-repo-13.0-848.tar.gz
```

The command uses the upstream Docker images as a base to build the required containers and pushes them to the local Docker repository. It creates a `/home/stack/templates/ciscoaci_containers.yaml` environment file, which you should include as a template during Overcloud deployment. You can use the **`-o`** option to override the output filename. You then verify that the output file was created as you specified.

Note

- You might need to log out of stack user and log back in because the Docker groups membership for stack user might not have taken effect. To check this, make sure that you can run Docker commands like **`docker ps`** as stack user without permission problems.
- During execution of the container-creation command, you may see an error that is generated by the command **`/bin/gbp-db-manage`**. You can safely ignore this error, which should not cause the execution of the script to fail.

Note Starting with the 4.2(1) release, OpenStack Director 13 (Queens) deployments support configuration of a Docker registry. Users have the following choices for the registry:

- Upstream registry (allows for using a local satellite server – currently the Red Hat registry)
- Downstream registry address/port/URI (currently the underlay controller, 8787, /rhosp13)

This is configured using the `build_openstack_aci_containers.py` script:

```
usage: build_openstack_aci_containers.py [-h] [-u UCLOUD_IP] [-o OUTPUT_FILE]
                                         [-c CONTAINERS_TB]
                                         [-s UPSTREAM_REGISTRY]
                                         [-d DESTINATION_REGISTRY]
                                         [-r REGSEPARATOR] [-t TAG] [--force]
                                         (-f FILE | -z FILE)

Build containers for ACI Plugin

optional arguments:
  -h, --help            show this help message and exit
  -u UCLOUD_IP, --ucloud_ip UCLOUD_IP
                        Undercloud ip address
  -o OUTPUT_FILE, --output_file OUTPUT_FILE
                        Environment file to create, default is
                        /home/stack/templates/ciscoaci_containers.yaml
  -c CONTAINERS_TB, --container CONTAINERS_TB
                        Containers to build, comma separated, default is all
  -s UPSTREAM_REGISTRY, --upstream UPSTREAM_REGISTRY
                        Upstream registry to pull base images from, eg.
                        registry.access.redhat.com/rhosp13, defaults to
                        registry.access.redhat.com/rhosp13
  -d DESTINATION_REGISTRY, --destregistry DESTINATION_REGISTRY
                        Destination registry to push to, eg:
                        1.100.1.1:8787/rhosp13
  -r REGSEPARATOR, --regseparator REGSEPARATOR
                        Upstream registry separator for images, eg. '/' for
                        normal upstream registrys (default). Will be added
                        between upstream registry name and container name. Use
                        '_' for satellite based registries.
  -t TAG, --tag TAG     tag for images, defaults to current timestamp
  --force               Override check for md5sum mismatch
  -f FILE, --aci_repo_file FILE
                        Path to yum repoistory file, which describes the
                        repository which provides ACI plugin rpm files. If you
                        want this script to create a repository on undercloud,
                        please use the -z option to provide path to openstack-
                        aci-rpms-repo tar file downloaded from cisco website
  -z FILE, --aci_rpm_repo_tar_file FILE
                        Path to openstack-aci-rpms-repo tar file. This will be
                        use to create a local yum repository on undercloud
```

Installing Overcloud

This section describes how to install Overcloud.

Procedure

Step 1 Copy the /usr/share/openstack-tripleo-heat-templates/roles_data.yaml file to a private location.

Example:

```
cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml
/home/stack/templates/aci_roles_data.yaml
```

Step 2 Edit the local copy of roles_data.yaml(aci_roles_data.yaml) to add CiscoAciAIM and CiscoAciLldp service to controller role and CiscoAciLldp service to compute role.

From Cisco ACI Release 5.2(1), CiscoAciOpflexAgent service is supported. If you are deploying a release prior to 5.2(1), you must remove the CiscoAciOpflexAgent service.

a) Under the controller role, add the following lines:

```
- OS::TripleO::Services::CiscoAciAIM
- OS::TripleO::Services::CiscoAciLldp
- OS::TripleO::Services::CiscoAciOpflexAgent
```

b) Under the compute role, add the following line:

```
- OS::TripleO::Services::CiscoAciLldp
- OS::TripleO::Services::CiscoAciOpflexAgent
```

Step 3 Declare resources for Cisco Application Centric Infrastructure (ACI) enviroment.

You must define Cisco ACI resources in a .yaml template file to include with deployment. For example, /home/stack/templates/aci_cs.yaml. This step describes the resource declaration for an OpFlex agent use case.

The following example shows resources for deploying OSP with OpFlex.

Note

- For an example of a full resources declaration, see the section Example of Resources Declaration, on page 32 in the appendix of this guide.
- For an example of a resources declaration for non-OpFlex use cases (neutron-openvswitch-agent), see the section Example of Resources Declaration When Using the Neutron OVS Agent, on page 33 in the appendix of this guide.
- For a list of parameters that are required for the Cisco ACI environment, see the section Parameters for the Cisco ACI Environment, on page 26 in the appendix of this guide.

Example:

(for Cisco ACI releases prior to 5.2(1))

```
resource_registry:
#controller
  OS::TripleO::ControllerExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates/nodepre.yaml
  OS::TripleO::Services::NeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::NeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_opflex.yaml
  OS::TripleO::Docker::NeutronMl2PluginBase:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci-ml2.yaml
  OS::TripleO::Services::CiscoAciAIM:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_aciaim.yaml

#compute
```

```
  OS::TripleO::ComputeExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates/nodepre.yaml
  OS::TripleO::Services::ComputeNeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_opflex.yaml
  OS::TripleO::Services::ComputeNeutronCorePlugin:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci_compute.yaml
  OS::TripleO::Services::ComputeNeutronMetadataAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/compute-neutron-metadata.yaml

#if using LLDP to discover switch connections
  OS::TripleO::Services::CiscoAciLldp:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_lldp.yaml

#if not using LLDP
  OS::TripleO::Services::CiscoAciLldp: OS::Heat::None
```

Example:

(for Cisco ACI release 5.2(1))

```
resource_registry:
#controller
  OS::TripleO::ControllerExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates/nodepre.yaml
  OS::TripleO::Services::NeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::NeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_neutron_opflex.yaml
  OS::TripleO::Docker::NeutronMl2PluginBase:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci-ml2.yaml
  OS::TripleO::Services::CiscoAciAIM:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_aciaim.yaml

#compute
  OS::TripleO::ComputeExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates/nodepre.yaml
  OS::TripleO::Services::ComputeNeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_neutron_opflex.yaml
  OS::TripleO::Services::ComputeNeutronCorePlugin:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci_compute.yaml
  OS::TripleO::Services::ComputeNeutronMetadataAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/compute-neutron-metadata.yaml

#if using LLDP to discover switch connections
  OS::TripleO::Services::CiscoAciLldp:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_lldp.yaml
  OS::TripleO::Services::CiscoAciOpflexAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_opflex.yaml

#if not using LLDP
  OS::TripleO::Services::CiscoAciLldp: OS::Heat::None”
```

Step 4 In order to use Cisco ACI certificate-based authentication, create a local user with a X.509 certificate and specify the certificate and key in the Cisco ACI resources file using the parameters ACIApicPrivateKey and ACIApicCertName. .

See the .section "Creating a Local User and Adding a User Certificate" in *Cisco APIC Security Configuration Guide, Release 4.2(x)*

Note When you use certificate-based authentication, make sure that you do not specify the parameter `ACIApicPassword`.

Step 5 Deploy Overcloud.

When deploying Overcloud, include the custom roles data file created using the `-r` option. Also include the Cisco ACI environment file and Cisco ACI containers YAML file in the environment list in addition to site-specific environment files.

Example:

```
openstack overcloud deploy --templates /home/stack/tripleo-heat-templates -r
/home/stack/templates/aci_roles_data.yaml  -e
/home/stack/tripleo-heat-templates/environments/network-isolation.yaml -e
/home/stack/templates/overcloud_images.yaml -e /home/stack/templates/network-environment.yaml
 -e /home/stack/templates/ciscoaci_containers.yaml -e /home/stack/templates/ciscoaci-env.yaml
 -e /home/stack/templates/rhel-registration-resource-registry.yaml -e
/home/stack/templates/environment-rhel-registration.
```

Note The preceding example illustrates the use of Cisco ACI templates and roles. Other templates may differ depending on your installation configuration. Follow the Red Hat guidelines for the creation of custom templates and autogeneration of the network environment template.

References

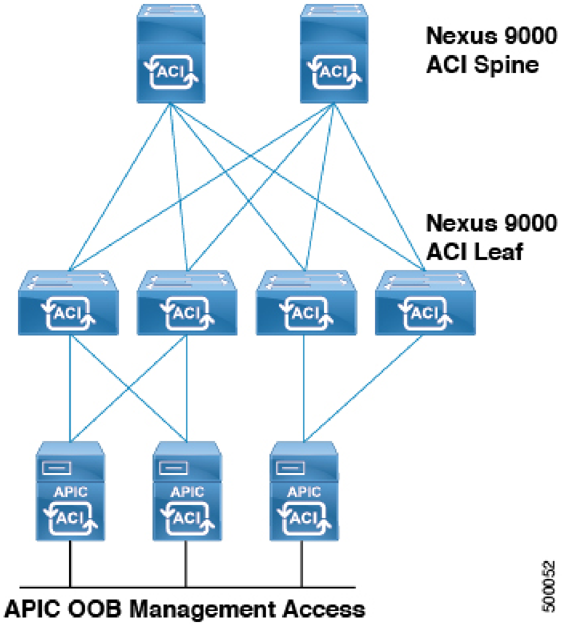
- Director installation and usage for the Red Hat OpenStack platform on the Red Hat website.
- The knowledge base article *Creating Domains, Attach Entity Profiles, and VLANs to Deploy an EPG on a Specific Port* on Cisco.com.

Cisco ACI Fabric Initialization Example

This example solution is based on a basic Spine/Leaf switching fabric, installed with all defaults on the Cisco Application Policy Infrastructure Controller (APIC) configuration other than Fabric Name and controller IP addressing. Three Cisco APICs are used to form a highly available cluster. Each Cisco APIC is connected to one or more of the leaf switches in the fabric. It is best to use diverse leaf switches for connecting multiple Cisco APICs to provide higher availability of the controller services.

The switching system continues to forward traffic regardless of the presence of the Cisco APIC cluster. All configuration of the fabric is driven through the cluster so no configuration could be added, changed, or deleted without Cisco APIC connectivity in place. To ensure that administrative control of the fabric is not dependent on the fabric itself, an Out-Of-Band (OOB) network connection is needed on each of the Cisco APICs as shown in the following figure:

Figure 3: Cisco APIC Cluster Connectivity



Before you begin

It is a good practice to make a note of the serial number of each of the switches in the Cisco Application Centric Infrastructure (ACI) fabric before discovering the fabric. Ideally, the console port of each of the switches is also connected to a terminal server so there is always administrative control regardless of the state of the Cisco ACI fabric.

To recover the serial number when logged into a switch running a Cisco ACI software image, enter the **show inventory** command at the Cisco ACI switch CLI, noting the primary system serial number. This is the number that displays in the Cisco APIC during fabric discovery, allowing you to assign the correct name and node numbering in your scheme to the devices.

Procedure

Step 1 To allow the Cisco APIC to discover and register the switches in the fabric, log in to the Cisco APIC GUI and then complete the following steps:

a) On the menu bar, choose **Fabric** > **Inventory**.
b) In the **Navigation** pane, choose **FabricMembership**.
c) In the **Work** pane, you should see an entry for the first switch discovered by the Cisco APIC.
d) Verify this is the expected first switch for the first Cisco APIC in the cluster based on serial number.
e) In the **Work** pane, choose the switch, right-click and choose **Register Switch**.

Note Assign logical numeric node IDs and node names that make sense for future troubleshooting, and Virtual Port Channel (vPC) pairing plans. For example, Node IDs 101/102 for the first two leaf switches, to be named leaf1/leaf2.

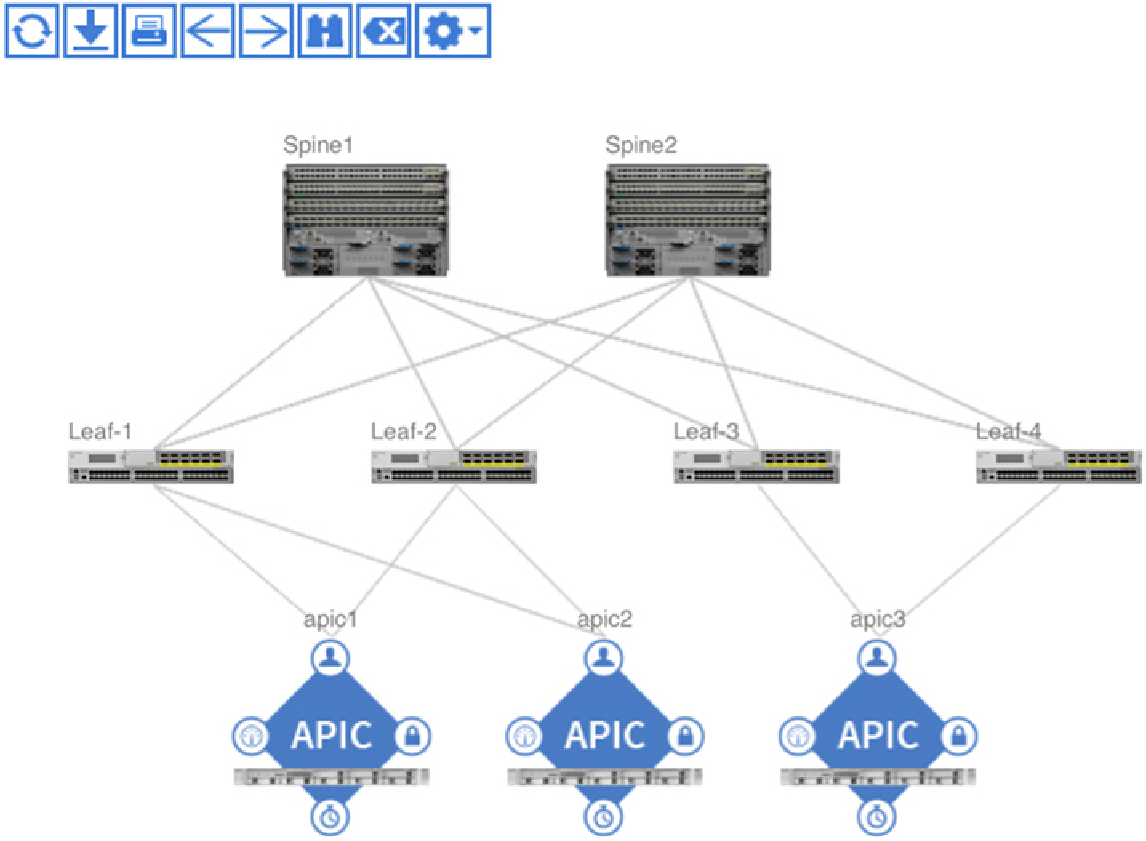
Step 2 Once the first leaf is discovered, the system will pass through that leaf to discover the spine switches, and then use the spine switches to discover remaining leaf switches. Register the additional nodes assigning logical node ID numbers and names according to the spine/leaf fabric layout.

Step 3 Confirm visually that the topology is discovered and physically connected as expected, perform the following actions:

a) On the menu bar, choose **Fabric** > **Inventory**.

b) In the **Navigation** pane, choose **Topology**.

Figure 4: Discovered Spine/Leaf Topology



c) Once the fabric is discovered, choose **Admin** > **Firmware** and validate the firmware versions running on all Cisco APICs, and fabric nodes (switches). If needed, upgrade to current or consistent versions before beginning initial configuration.

CHAPTER 3

Upgrading Cisco ACI and OSP

Guidelines for Upgrading Cisco APIC and OSP

The OpenStack plug-in is released with Cisco Application Policy Infrastructure ControllerAPIC releases, and therefore uses the same semantic version as Cisco APIC. For example, the 4.2(1) plug-in is provided with the Cisco APIC 4.2(1) release. Generally, the OpenStack plug-in releases are tested against the matching Cisco APIC release, as well as the previous Long Term Support (LTS) Cisco APIC release. However, a given plug-in release may be compatible with additional Cisco APIC releases. See the *Cisco ACI Virtualization Compatibility Matrix* to verify that the version of the plug-in used is compatible with the version of Cisco APIC.

See the *Cisco ACI Virtualization Compatibility Matrix* for information about compatible Cisco APIC and Red Hat OSP releases.

Pre-upgrade Guidelines

You need to first upgrade the Cisco Application Centric Infrastructure (ACI) plug-in.

For more information about the compatibility of the plug-in with various OpenStack versions, see the *Cisco ACI Virtualization Compatibility Matrix* at:

https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/aci/virtualization/matrix/virtmatrix.html

Upgrade Guidelines

The Cisco ACI fabric can be upgraded following the information on Cisco Application Policy Infrastructure Controller (APIC) Management, Installation, Upgrade, and Downgrade Guide at:

https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/2-x/managing_ACI_fabric_upgrades_and_downgrades/b_Managing_ACI_Fabric_Upgrades_and_Downgrades/b_Managing_ACI_Fabric_Upgrades_and_Downgrades_chapter_01.html

Optionally, you can upgrade the Cisco ACI fabric without upgrading the plug-in, as long as the Cisco ACI plug-in and Cisco ACI fabric release combination is supported. For more information, see the *Cisco ACI Virtualization Compatibility Matrix* at:

https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/aci/virtualization/matrix/virtmatrix.html

Post-upgrade Guidelines

After you upgrade the Cisco Application Centric Infrastructure (ACI) fabric, you can optionally upgrade the OpenStack Cisco ACI packages. Refer to the OpenStack Cisco ACI plug-in Release Notes for specific information.

For more information on how to upgrade the OpenStack Cisco ACI plug-in, see Upgrading the Cisco ACI Packages, on page 18 in this guide.

Upgrading the Cisco ACI Packages

The following procedure updates fully deployed Overcloud with the new version of the Cisco Application Centric Infrastructure (ACI) plug-in. The upgrade can be live.

Note Follow the Red Hat Director documentation to upgrade the plug-in in step 4.

Procedure

Step 1 Copy the updated version of the `tripeo-ciscoaci-`*`version`* RPM and corresponding plug-in tarball (`openstack-ciscorpms-repo`) from Cisco.com to the OSP Director.

Step 2 Update the `tripeo-ciscoaci-`*`version`* package using `yum`: **`yum update tripleo-ciscoaci-`**

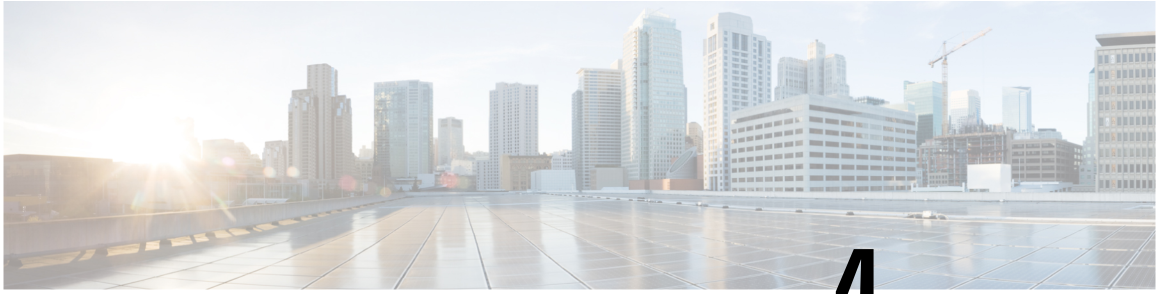
Step 3 Create the Cisco ACI containers by running the command `/opt/ciscoaci-tripleo-heat-templates/tools/build_openstack_aci_containers.py` and then pointing it to the downloaded plug-in tarball. For example:

```
/opt/ciscoaci-tripleo-heat-templates/tools/build_openstack_aci_containers.py -z
/home/stack/openstack-ciscorpms-repo-13.0-848.tar.gz
```

The command updates the Docker images and updates the `/home/stack/templates/cisco_containers.yaml` file.

Step 4 Follow the Red Hat upgrade procedure on the Red Hat Customer Portal.

See the Chapter 4, "Updating the Overcloud" in the article *Keeping Red Hat OpenStack Platform Updated*. Go to **Products & Services** > **Product Documentation** > **Red Hat OpenStack Platform13** > **13**. The article is in the **Installing & Upgrading** section.



CHAPTER 4

OpenStack External Network

This chapter contains the following sections:

Add an OpenStack External Network

This section describes how to add an OpenStack external network.

Note Execute the commands in this procedure sourcing the keystone file for the project where you want to create the network constructs and the instance.

Before you begin

You must have done the following before adding an OpenStack external network:

- Created a Layer 3 outside connection (L3Out) in Cisco Application Policy Infrastructure Controller (ACI).

  The L3Out can be in the OpenStack-created tenant (dedicated L3out for the OpenStack tenant) or in the Common tenant (Shared L3out across multiple OpenStack tenants). This procedure assumes that a dedicated L3out called *l3out1* is configured in the OpenStack tenant.

- Specified the following in the L3Out:
  - Interfaces and their IP information.
  - Dynamic routing, if used.
  - An external endpoint group (EPG).

    This procedure uses an external EPG named *extEpg*.

Note Do not add any contracts; the plug-in adds them automatically.

Important If you require Source Network Address Translation (SNAT) or a floating IP (FIP) address, you must define the L3Out in a different VRF from the one created by OpenStack.

Procedure

Step 1 Create the Neutron external network and provide the distinguished name of the L3Out.

Example:

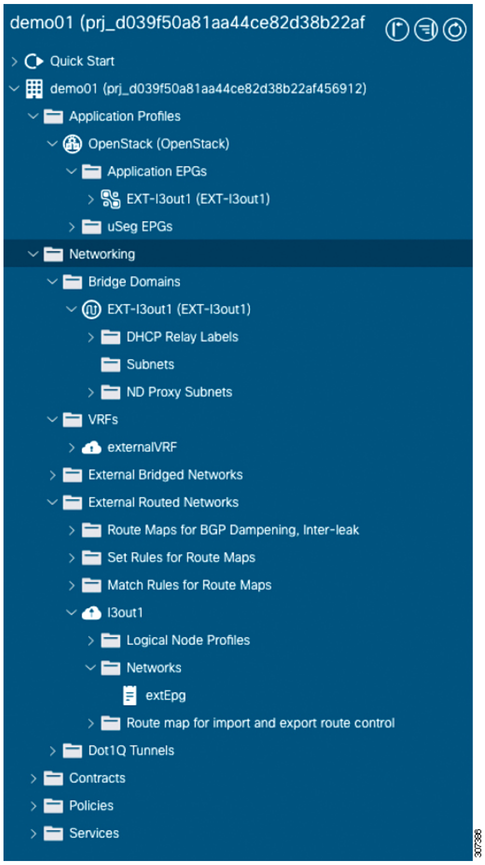
```
neutron net-create network_name --router:external --apic:distinguished_names type=dict
ExternalNetwork=uni/tn-ACI_tenant_name/out-ACI_L3out_name/instP-ACI_externalEPG_name
(--apic:nat_type "")
```

`--apic:nat_type ""` is optional. Use it only if you do not use NAT for the specific external Neutron network.

The following shows an example of the creation of the external network with NAT enabled:

```
neutron net-create external-net-dedicated --router:external --apic:distinguished_names
type=dict ExternalNetwork=uni/tn-prj_$demo01/out-l3out1/instP-extEpg
Created a new network:
+--------------------------------------+----------------------------------------------------+
| Field                                 | Value
    |
+--------------------------------------+----------------------------------------------------+
| admin_state_up                        | True
    |
| apic:bgp_asn                          | 0
    |
| apic:bgp_enable                       | False
    |
| apic:bgp_type                         | default_export
    |
| apic:distinguished_names              | {"EndpointGroup":
"uni/tn-prj_cdeda9c674a94394a09e86a2fea498c2/ap-OpenStack/epg-EXT-l3out1",
"ExternalNetwork": "uni/tn-prj_cdeda9c674a94394a09e86a2fea498c2/out-l3out1/instP-extEpg",
"VRF": "uni/tn-prj_cdeda9c674a94394a09e86a2fea498c2/ctx-externalVRF", "BridgeDomain":
"uni/tn-prj_cdeda9c674a94394a09e86a2fea498c2/BD-EXT-l3out1"}
    |
| apic:external_cidrs                   | 0.0.0.0/0
    |
| apic:nat_type                         | distributed
    |
| apic:nested_domain_allowed_vlans      |
    |
| apic:nested_domain_infra_vlan         |
    |
| apic:nested_domain_name               |
    |
| apic:nested_domain_node_network_vlan  |
    |
| apic:nested_domain_service_vlan       |
    |
| apic:nested_domain_type               |
    |
| apic:svi                              | False
    |
| apic:synchronization_state            | build
    |
| availability_zone_hints               |
    |
| availability_zones                   |
    |
| created_at                           | 2019-05-22T13:38:32Z
    |
| description                          |
    |
| id                                   | 635623ed-5dba-42ec-b3f8-3cff18f925c6
    |
| ipv4_address_scope                   |
    |
| ipv6_address_scope                   |
    |
| is_default                           | False
    |
| mtu                                  | 9000
    |
| name                                 | external-net-dedicated
    |
| port_security_enabled                | True
    |
| project_id                           | cdeda9c674a94394a09e86a2fea498c2
    |
| provider:network_type                | opflex
    |
| provider:physical_network            | physnet1
    |
| provider:segmentation_id             |
    |
| revision_number                      | 6
    |
| router:external                      | True
    |
| shared                               | False
    |
| status                               | ACTIVE
    |
| subnets                              |
    |
| tags                                 |
    |
| tenant_id                            | cdeda9c674a94394a09e86a2fea498c2
    |
| updated_at                           | 2019-05-22T13:38:33Z
    |
+------------------------------------+----------------------------------------------------+
```

In Cisco ACI, the command creates a new EPG—*EXT-l3out1*— and a new bridge domain—*EXT-l3out1*, as shown in the following screen capture of the Cisco Application Policy Infrastructure Controller (APIC) GUI:



Step 2 Create a Neutron subnet that will be used for SNAT and the floating IP address.

This step is not required if you used `--apic:nat_type ""` when you created the Neutron external network (because NAT is disabled).

Example:

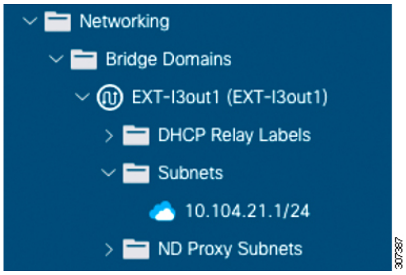
```
neutron subnet-create net_name subnet/mask --name subnet_name --disable-dhcp --gateway
gateway_ip --apic:snat_host_pool True
```

The OpFlex agent automatically assigns one IP address for every compute node from the subnet. Virtual machines (VMs) connecting to the external use this IP address (one-to-many NAT) unless they have been assigned with a floating IP address.

The following shows an example of the creation of the external network with NAT enabled:

```
neutron subnet-create external-net-dedicated 10.104.21.0/24 --name ext-subnet --disable-dhcp
 --gateway 10.104.21.1 --apic:snat_host_pool True
Created a new subnet:
+---------------------------+---------------------------------------------------+
| Field                     | Value                                             |
+---------------------------+---------------------------------------------------+
| allocation_pools          | {"start": "10.104.21.2", "end": "10.104.21.254"} |
| apic:distinguished_names  | {}                                                |
| apic:snat_host_pool       | True                                              |
| apic:synchronization_state | N/A                                              |
| cidr                      | 10.104.21.0/24                                    |
| created_at                | 2019-05-22T13:38:35Z                              |
| description               |                                                   |
| dns_nameservers            |                                                  |
| enable_dhcp                | False                                            |
| gateway_ip                 | 10.104.21.1                                      |
| host_routes                |                                                  |
| id                         | 238aa55d-1537-4f01-86c9-5f6fc4bde625             |
| ip_version                 | 4                                                |
| ipv6_address_mode          |                                                  |
| ipv6_ra_mode               |                                                  |
| name                       | ext-subnet                                       |
| network_id                 | 635623ed-5dba-42ec-b3f8-3cff18f925c6             |
| project_id                 | cdeda9c674a94394a09e86a2fea498c2                 |
| revision_number            | 0                                                |
| service_types              |                                                  |
| subnetpool_id              |                                                  |
| tags                       |                                                  |
| tenant_id                  | cdeda9c674a94394a09e86a2fea498c2                 |
| updated_at                 | 2019-05-22T13:38:35Z                             |
+----------------------------+--------------------------------------------------+
```

Creating a SNAT subnet generates a new subnet under the bridge domain, as shown in the following screen capture of the Cisco APIC GUI:



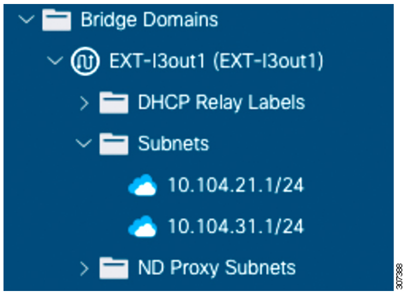
Step 3 (Optional) Assign one or more floating subnets to the external Neutron network:

Example:

```
neutron subnet-create net_name fip_subnet/mask --name subnet_name --allocation-pool
start=start_ip,end=end_ip --disable-dhcp --gateway gateway_ip
```

The following output and screen capture in Cisco APIC GUI show an example of the creation of a floating IP subnet:

```
neutron subnet-create external-net-dedicated 10.104.31.0/24 --name ext-subnet-FIP
--allocation-pool start=10.104.31.10,end=10.104.31.100 --disable-dhcp --gateway 10.104.31.1
Created a new subnet:
+----------------------------+---------------------------------------------------+
| Field                      | Value                                             |
+----------------------------+---------------------------------------------------+
| allocation_pools           | {"start": "10.104.31.10", "end": "10.104.31.100"} |
| apic:distinguished_names   | {}                                                |
| apic:snat_host_pool        | False                                             |
| apic:synchronization_state | N/A                                               |
| cidr                       | 10.104.31.0/24                                    |
| created_at                 | 2019-05-22T13:38:38Z                              |
| description                |                                                   |
| dns_nameservers            |                                                   |
| enable_dhcp                | False                                             |
| gateway_ip                 | 10.104.31.1                                       |
| host_routes                |                                                   |
| id                         | 107c2714-2ace-44a7-9cb0-1a7f40ba2833              |
| ip_version                 | 4                                                 |
| ipv6_address_mode          |                                                   |
| ipv6_ra_mode               |                                                   |
| name                        | ext-subnet-FIP                                    |
| network_id                  | 635623ed-5dba-42ec-b3f8-3cff18f925c6              |
| project_id                  | cdeda9c674a94394a09e86a2fea498c2                  |
| revision_number             | 0                                                 |
| service_types               |                                                   |
| subnetpool_id               |                                                   |
| tags                        |                                                   |
| tenant_id                   | cdeda9c674a94394a09e86a2fea498c2                  |
| updated_at                  | 2019-05-22T13:38:38Z                              |
+-----------------------------+---------------------------------------------------+
```



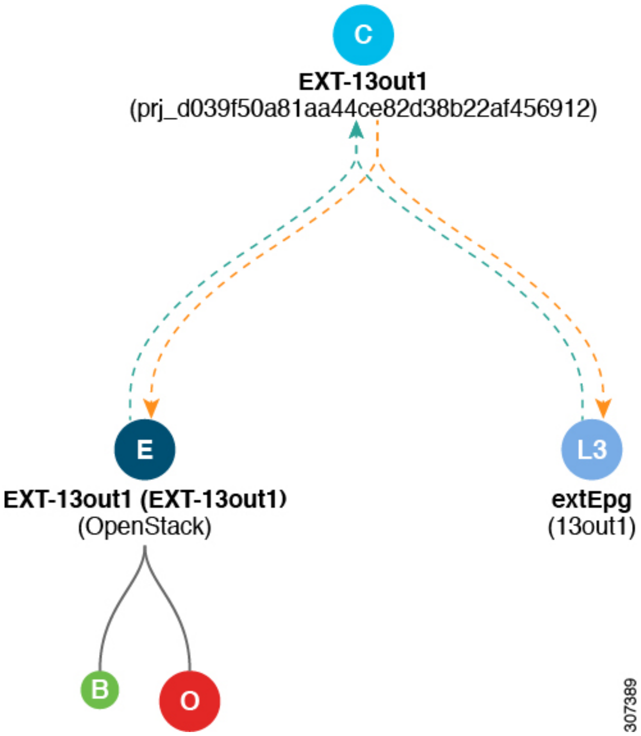
- Bridge Domains
  - EXT-l3out1 (EXT-l3out1)
    - DHCP Relay Labels
    - Subnets
      - 10.104.21.1/24
      - 10.104.31.1/24
    - ND Proxy Subnets

Step 4 Attach the Neutron external network to one OpenStack router as a gateway.

Example:

```
openstack router set --external-gateway external_net_name router_name
```

The command creates a contract that allows external connectivity for tenant networks attached to the OpenStack router of the external Neutron network, as shown in the following image:



APPENDIX A

Reference Information

This chapter contains the following sections:

Configuring Hierarchical Port Binding

This section describes configuring the Single Root I/O Virtualization (SR-IOV) and other VLAN-based ml2 mechanism agents to work with OpFlex plug-in. This is accomplished by using Hierarchical Port Binding (HPB) and should work without any special modification to the configuration. Here are the basic steps needed to configure OpFlex with SR-IOV.

When using HPB, datapath connectivity in Cisco Application Centric Infrastructure (ACI) is accomplished by creating static VLAN bindings to the EPGs for networks created by OpenStack. There maybe other configuration required for datapath, for example, setting up VLAN on SR-IOV NIC or configuring OVS (or a load balancer in case of LBaaS). This is done by the third party agent or mechanism driver (for example, sriovnicswitch).

How to create these assets:

Before you begin

In order to configure the datapath using static VLAN bindings. The plug-in requires following assets:

- A physical domain (physdom) with the correct VLAN pool.
- Host-link information (which compute node fabric Ethernet interface is connected to which leaf switch port)
- Host-link-network-label information (describing which fabric Ethernet interface on compute node is used to serve which physnet)

  This information is only needed if the deployment uses multiple physnets.

Procedure

Step 1 Before deploying OpenStack Platform Overcloud, make sure you have one Physical Domain (physdom) created per each physnet required. Add pdom_ prefix to the name of physical domain created. For example for physnet1 create pdom_physnet1, and attach the right VLAN pool.

Step 2 The host-link information usually comes from LLDP auto discovery. It can also be provided statically using ACIHostLinks parameter in the Cisco ACI deployment template as shown in example below.

You also need to set NeutronNetworkVLANRanges and enable the third party mechanism drivers using ACIMechanismDrivers parameter, make sure that the apic_aim is the last mechanism in the list.

Example:

```
NeutronPhysicalDevMappings: physnet1:ens11,physnet2:ens7,physnet3:ens9
NeutronNetworkVLANRanges:physnet1:1200:1250,physnet2:1251:1300,physnet3:1301:1350
ACIMechanismDrivers: 'sriovnicswitch,apic_aim'
ACIHostLinks: '{"101": [{"host01|ens11": "1/14"}], "102": [{"host02|ens9": "1/14"}]}'
```

Parameters for the Cisco ACI Environment

The following table provides information about parameters that are required to configure the Cisco Application Centric Infrastructure (ACI) environment.

Parameter	Details
NeutronCorePlugin	• **Value:** 'ml2plus' • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
NeutronServicePlugins	• **Value:** 'group_policy,ncp,apic_aim_l3' • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
NeutronEnableIsolatedMetadata	• **Value:** true • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** Must be set to true

Parameter	Details
NeutronEnableForceMetadata	• **Value:** true • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** Must be set to true
ACIYumRepo	• **Value:**: http://*undercloud pxe network ipaddress* • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
ACIApicHosts	• **Value:** Cisco APIC name and addresses • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
ACIApicUsername	• **Value:** Username with administrative privileges • **Default:** admin • **Mandatory or Optional:** Optional • **Comments:** None
ACIApicPassword	• **Value:** Password • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None **Note** Do not provide this parameter if certificate-based authentication is used.
ACIMechanismDrivers	• **Value:** 'apic_aim' • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** Add extra drivers—for example, for Open vSwitch when using neutron ovs agent or for sriovnicswitch when using sriov

Parameter	Details
ACIApicEntityProfile	• **Value:** The Cisco ACI entity profile that has been preprovisioned on Cisco ACI • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
ACIApicInfraVlan	• **Value:** The Cisco ACI fabric infra VLAN • **Default:** 4093 • **Mandatory or Optional:** Optional • **Comments:** Contact the Cisco ACI administrator for the correct value
ACIApicInfraSubnetGateway	• **Value:** The Cisco ACI infra subnet gateway • **Default:** 10.0.0.30 • **Mandatory or Optional:** Optional • **Comments:** Contact the Cisco ACI administrator for the correct value
ACIApicInfraAnycastAddr	• **Value:** The Cisco ACI anycast address • **Default:** 10.0.0.32 • **Mandatory or Optional:** Optional • **Comments:** Contact the Cisco ACI administrator for the correct value
ACIUseLldp	• **Value:** true or false • **Default:** true • **Mandatory or Optional:** Optional • **Comments:** If set to false, set CiscoAciLldp service to `OS::Hat:None`
ACIOpflexUplinkInterface	• **Value:** Interface name that is connected to the Cisco ACI leaf switch • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** Actual interface name—for example, enp8s0

Parameter	Details
ACIOpflexEncapMode	• **Value:** vxlan or vlan • **Default:** vxlan • **Mandatory or Optional:** Optional • **Comments:** None
ACIOpflexVlanRange	• **Value:** *starting_vlan*:*ending_vlan* • **Default:** None • **Mandatory or Optional:** Mandatory if ACIOpflexEncapMode is set to vlan • **Comments:** None
ACIOpflexInterfaceType	• **Value:** linux or ovs • **Default:** linux • **Mandatory or Optional:** Optional • **Comments:** Set this value to 'ovs' when planning to deploy an "OpenShift on OpenStack" nested installation. The setting causes the OpFlex interface to be created on the ovs switch.
ACIOpflexInterfaceMTU	• **Value:** Intended MTU size • **Default:** 1500 • **Mandatory or Optional:** Optional • **Comments:** Use this parameter to set the MTU for the OpFlex interface. This must be set to 8000 for installing OpenShift on OpenStack.
NeutronPluginMl2PuppetTags	• **Value:** 'neutron_plugin_ml2,neutron_plugin_cisco_aci,neutron_sfc_service_config' • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** None
NeutronNetworkVLANRanges	• **Value:** *physnet*:*starting vlan*:*ending vlan* (For example, physnet1:1100:1150,physnet2:1201:1211) • **Default:** None • **Mandatory or Optional:** Mandatory when using neutron ovs agent • **Comments:** None

Parameter	Details
NeutronBridgeMappings	• **Value:** For example: 'physnet1:br-ex,physnet2:br-ex' • **Default:** None • **Mandatory or Optional:** Mandatory when using neutron ovs agent • **Comments:** Physnets should match as provided in NeutronBridgeMappings
AciTenantNetworkType	• **Value:** vlan • **Default:** None • **Mandatory or Optional:** Mandatory when using neutron ovs agent • **Comments:** None
AciOpenvswitch	• **Value:** true or false • **Default:** false • **Mandatory or Optional:** Set to true when using neutron ovs agent • **Comments:** None
NeutronOVSFirewallDriver	• **Value:** 'neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver' • **Default:** None • **Mandatory or Optional:** Mandatory • **Comments:** Set to the value shown when using neutron ovs agent
ACIHostLinks	• **Value:** For example: '{"101":{"ha.dom":"1/1", "hb.dom":"1/2"}, "102":{"hc.dom":"1/1"} }' • **Default:** None • **Mandatory or Optional:** Mandatory when using neutron ovs agent and not using lldp agent • **Comments:** Describes the host connections to switches in JSON format. In the example, The ha.dom host is connected to port 1/1 of switch ID 101, the hb.dom host is connected to port 1/2 of switch ID 101, and the hc.dom is connected to port 1/1 of switch ID 102.
NeutronPhysicalDevMappings	• **Value:** • **Default:** None • **Mandatory or Optional:** Optional • **Comments:**

Parameter	Details
NeutronPhysicalDevMappings	- **Value:** For example: physnet1:eth1,physnet2:eth2 - **Default:** None - **Mandatory or Optional:** Optional - **Comments:** You must set this parameter when you want to map a particular interface to a specific physnet
ACIApicCertName	- **Value:** Name of the Cisco APIC cert User (used for certificate-based authentication) - **Type:** String - **Default:** None - **Mandatory or Optional:** Optional
ACIApicPrivateKey	- **Value:** Private key for the cert User - **Type:** String - **Default:** None - **Mandatory or Optional:** Optional
ACIEnableBondWatchService	- **Value:** True or False - **Type:** Boolean - **Default:** False - **Comment:** Set this parameter to True if you use Cisco Unified Computing System (UCS) blade servers for OpenStack nodes.
AciKeystoneNotificationPurge	- **Value:** True or False - **Type:** Boolean - **Default:** False - **Comment:** Enables the automatic purge of Cisco APIC tenants when the project is deleted in OpenStack.
NeutronPluginExtensions	- **Value:** Comma-separated list of enabled extension plugins. - **Default:** apic_aim,port_security - **Mandatory or Optional:** Optional - **Comment:**Recommended values when parameter is explicitly configured are - apic_aim, port_security, qos.

Example of Resources Declaration

The following is a full example of the Cisco Application Centric Infrastructure (ACI) resources declaration (`ciscoaci-env.yaml`):

You declare resources for Cisco ACI environment when you install Overcloud. See the procedure Installing Overcloud, on page 10 in this guide.

The following example is applicable for Cisco ACI Release 5.2(1) and later:

```
# A Heat environment file which can be used to enable a
# a Neutron Cisco Aci backend on the controller, configured via puppet
resource_registry:

  #controller
  OS::TripleO::ControllerExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates//nodepre.yaml

  OS::TripleO::Services::NeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::NeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_nuetron_opflex.yaml
  OS::TripleO::Docker::NeutronMl2PluginBase:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci-ml2.yaml
  OS::TripleO::Services::CiscoAciAIM:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_aciaim.yaml

  #compute
  OS::TripleO::ComputeExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates//nodepre.yaml
  OS::TripleO::Services::ComputeNeutronOvsAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_neutron_opflex.yaml
  OS::TripleO::Services::ComputeNeutronCorePlugin:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci_compute.yaml
  OS::TripleO::Services::ComputeNeutronMetadataAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/compute-neutron-metadata.yaml


  OS::TripleO::Services::CiscoAciLldp:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_lldp.yaml
  OS::TripleO::Services::CiscoAciOpflexAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_opflex.yaml


parameter_defaults:
  NeutronSfcDriver: 'aim'
  NeutronFcDriver: 'aim'
  AdminPassword: 'admin0123'
  EC2MetadataIp: 1.100.1.1
  ControlPlaneDefaultRoute: 1.100.1.1
  DockerInsecureRegistryAddress: 1.100.1.1:8787
  OvercloudControllerFlavor: control
  OvercloudComputeFlavor: compute
  ExternalNetworkVlanID: 1
  InternalApiNetworkVlanID: 301
  StorageNetworkVlanID: 302
  StorageMgmtNetworkVlanID: 303
  TenantNetworkVlanID: 304
  NeutronCorePlugin: 'ml2plus'
  NeutronPluginMl2PuppetTags: 'neutron_plugin_ml2,neutron_plugin_cisco_aci'
  NeutronServicePlugins: 'group_policy,ncp,apic_aim_l3'
  NeutronEnableIsolatedMetadata: true
  NeutronEnableForceMetadata: true
  EnablePackageInstall: true
  ACIScopeNames: true
  ACIYumRepo: http://1.100.1.1/acirepo
  ACIApicHosts: 172.171.170.169
  ACIApicUsername: admin
  ACIApicPassword: mypass
  ACIApicSystemId: osp13
  ACIMechanismDrivers: 'apic_aim'
  ACIApicEntityProfile: osp13-aep
  ACIApicInfraVlan: 4093
  ACIApicInfraSubnetGateway: 10.0.0.30
  ACIApicInfraAnycastAddr: 10.0.0.32
  ACIOpflexUplinkInterface: enp8s0
  ACIOpflexEncapMode: vxlan
  HeatEnginePluginDirs:
/usr/lib64/heat,/usr/lib/heat,/usr/local/lib/heat,/usr/local/lib64/heat,/usr/lib/python2.7/site-packages/gbpautomation/heat

  NeutronGlobalPhysnetMtu: 8950
  ACIOpflexInterfaceMTU: 9000
  NtpServer: ['172.28.184.8']
  NeutronPluginMl2PuppetTags:
'neutron_plugin_ml2,neutron_plugin_cisco_aci,neutron_sfc_service_config'
```

Note If you are deploying a release prior to Cisco ACI Release 5.2(1), you need to make the following changes in the above example:

- Remove the definition for the `OS::TripleO::Services::CiscoAciOpflexAgent` service.
- Change the `OS::TripleO::Services::NeutronOvsAgent` and `OS::TripleO::Services::ComputeNeutronOvsAgent` to reference the `/opt/ciscoaci-tripleo-heat-templates/deployment/opflex/copflex-agent-container-puppet.yaml` template.

Example of Resources Declaration When Using the Neutron OVS Agent

When you use the Neutron OVS agent, Cisco-specific containers for Horizon and Heat may be optional. If that is the case, remove them from the `ciscoaci_containers.yaml` file that was generated in step 5 in the section Preparing Undercloud for Cisco ACI with OpFlex Orchestration, on page 9 in this guide.

You can use the following commands to remove Horizon and Heat from the `ciscoaci_containers.yaml` file:

```
resource_registry:
#controller
  OS::TripleO::ControllerExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates//nodepre.yaml

  OS::TripleO::Services::NeutronL3Agent: OS::Heat::None
  OS::TripleO::Services::NeutronOvsAgent:
/usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-agent.yaml
  OS::TripleO::Docker::NeutronMl2PluginBase:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci-ml2.yaml
  OS::TripleO::Services::CiscoAciAIM:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_aciaim.yaml
  OS::TripleO::Services::NeutronSfcApi:
/usr/share/openstack-tripleo-heat-templates/puppet/services/neutron-sfc-api.yaml
```

```
  #compute
  OS::TripleO::ComputeExtraConfigPre: /opt/ciscoaci-tripleo-heat-templates//nodepre.yaml
  OS::TripleO::Services::ComputeNeutronOvsAgent:
/usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-agent.yaml
  OS::TripleO::Services::ComputeNeutronCorePlugin:
/opt/ciscoaci-tripleo-heat-templates/puppet/services/ciscoaci_compute.yaml
  OS::TripleO::Services::ComputeNeutronMetadataAgent:
/opt/ciscoaci-tripleo-heat-templates/docker/services/compute-neutron-metadata.yaml

#if using LLDP, else set it to OS::Heat::None
  OS::TripleO::Services::CiscoAciLldp:
/opt/ciscoaci-tripleo-heat-templates/docker/services/cisco_lldp.yaml

parameter_defaults:
  NeutronSfcDriver: 'aim'
  NeutronFcDriver: 'aim'
  AdminPassword: 'myPass123'
  EC2MetadataIp: 10.10.253.10
  ControlPlaneDefaultRoute: 10.10.253.10
  DockerInsecureRegistryAddress: 10.10.253.10:8787
  OvercloudControllerFlavor: control
  OvercloudComputeFlavor: compute
  NeutronCorePlugin: 'ml2plus'
  NeutronPluginMl2PuppetTags: 'neutron_plugin_ml2,neutron_plugin_cisco_aci'
  NeutronServicePlugins: 'group_policy,ncp,apic_aim_l3,flow_classifier,sfc,trunk'
  NeutronEnableIsolatedMetadata: true
  NeutronEnableForceMetadata: true
  EnablePackageInstall: true
  ACIScopeNames: true
  ACIYumRepo: http://10.10.253.10/acirepo
  ACIApicHosts: 171.36.17.106
  ACIApicUsername: admin
  ACIApicPassword: apicPass123
  ACIApicSystemId: osp13-ovs
  ACIApicEntityProfile: OSP-AEP
  ACIApicInfraVlan: 4093
  ACIApicInfraSubnetGateway: 10.0.0.30
  ACIApicInfraAnycastAddr: 10.0.0.32
  ACIOpflexUplinkInterface: ens9
  HeatEnginePluginDirs:
/usr/lib64/heat,/usr/lib/heat,/usr/local/lib/heat,/usr/local/lib64/heat,/usr/lib/python2.7/site-packages/gbpautomation/heat

  NeutronGlobalPhysnetMtu: 8950
  NtpServer: ['172.28.184.8']
  NeutronPluginMl2PuppetTags:
'neutron_plugin_ml2,neutron_plugin_cisco_aci,neutron_sfc_service_config'
  AciExternalRoutedDomain: 'myERD'
  ACIMechanismDrivers: openvswitch,apic_aim
  NeutronNetworkVLANRanges: physnet1:1200:1250
  AciOpenvswitch: true
  NeutronBridgeMappings: 'physnet1:br-ex'
  NeutronTunnelTypes: ''
  NeutronOVSFirewallDriver:
'neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver'
  NeutronEnableDVR: false
```

Examples of Creating Host Reports

During troubleshooting, you might need to collect host reports from the OpenStack cluster. You do it using provided playbook `/opt/ciscoaci-tripleo-heat-templates/tools/report.yaml`. This section provides example of using the host report playbook.

- `ansible-playbook /opt/ciscoaci-tripleo-heat-templates/tools/report.yaml`

  This example collects data from all nodes and creates the file `/home/stack/overcloud_aci_report.tgz`.

- `ansible-playbook /opt/ciscoaci-tripleo-heat-templates/tools/report.yaml -e '{"limit_flavors":['control'], "dest_file":/tmp/abc}`

  This example limits the repot to controllers and changes the default output file.

- `ansible-playbook /opt/ciscoaci-tripleo-heat-templates/tools/report.yaml -e '{"limit_hosts":[overcloud-controller-0, overcloud-controller-2]}'`

  This example limits the report collection to the hosts specified. You can club `"limit_flavors"` and `"limit_hosts"` to further filter the nodes from which to collect data.

Deploying with TLS

Deploying Red Hat OpenStack 13 with Transport Layer Security (TLS) is a supported configuration. To enable TLS on OpenStack endpoints, follow the instructions in *Advanced Overcloud Customization* on the Red Hat website.

To enable TLS between AIM and Cisco Application Policy Infrastructure Controller (APIC), follow the cert base authentication procedure described in step 4 of Installing Overcloud, on page 10 in this guide.

APPENDIX B

Configuring UCS B-Series

Configuring UCS B-Series for Cisco ACI and OpenStack Orchestration

You need three levels on configuration for Cisco Unified Computing System (UCS) B-Series to work with Cisco Application Centric Infrastructure (ACI) and OpenStack orchestration. The first layer is on Cisco UCS, the second on the host, and the third on the leaf switches.

Note This document applies to the Cisco UCS B-Series and C-Series servers connected to Fabric Interconnects in UCS mode and provides additional configuration required to install OpenStack on Cisco UCS.

Configuration on Linux Hosts

Configuration on Linux hosts includes binding the NICs in Active Backup mode, running the BondWatch service, and setting the NIC maximum transmission unit (MTU).

Bind the NICs

Bind the NICs in Active Backup mode, which you can do by setting the appropriate configuration in your OSP network environment NIC templates.

Procedure

Set the appropriate configuration.

Example:

```
type: linux_bond
bonding_options: "mode=active-backup miimon=10"
name: bond0
mtu: 1600
members:
 -
  type: interface
  name: nic2
  mtu: 1600
 -
  type: interface
  name: nic3
  primary: true
  mtu: 1600
```

Run the Bond Watch Service

The bond watch service (`apic-bond-watch`) detects failure of a NIC in the bond and sends gratuitous ARP requests to inform the fabric of the currently active NIC. We recommend that you run the bond watch service on the undercloud.

There are two ways you run the `apic-bond-watch` service, depending on which version of Cisco Application Policy Infrastructure Controller (APIC) that you use:

- **Cisco APIC Release 4.1(x) and earlier:** You perform a short series of steps.
- **Cisco APIC Release 4.2(1) and later:** You set a single parameter, and the `apic-bond-watch` is enabled and started. There are no manual steps that are required to set up, enable, or start the `apic-bond-watch` service.

The following is list of guidelines and recommendations for running the bond watch service:

- Verify that you have installed `/usr/bin/apic-bond-watch`.

  The file is part of the `apicapi` package.

- Add the OpFlex uplink device to `/etc/environments` (opflex_bondif=bond1)

  You must perform this step if the interface is other than default (bond0).

- Enable the bond watch service: `systemctl enable apic-bond-watch`.
- Start the bond watch service: `systemctl start apic-bond-watch`.

Note In releases earlier than Cisco Cisco Application Policy Infrastructure Controller (APIC) 4.1(2), you may need to manually run `apic-bond-watch` because the service file may be missing. To manually start the binary, you can use `nohup /usr/bin/apic-bond-watch <interface name>&` as the root user. The default interface name is `bond0`. For example:

```
nohup /usr/bin/apic-bond-watch &          //To use bond0
nohup /usr/bin/apic-bond-watch bond1 &    //To use bond1
```

Procedure

Step 1 Complete one of the following actions, depending on your version of Cisco Application Policy Infrastructure Controller (APIC).

- **Cisco APIC 4.2(1) or later:** Set the parameter ACIEnableBondWatchService to True. See the section "Installing Overcloud" in *Cisco ACI Installation Guide for Red Hat OpenStack Using OpenStack Platform 10 Director* or the section "Parameters for the Cisco ACI Environment" in *Cisco ACI Installation Guide for Red Hat OpenStack Using the OpenStack Platform 13 Director*. Do not complete the remaining steps of this procedure.
- **Cisco APIC 4.1(x) or earlier:** Complete step 2 through step 4 in this procedure.

Step 2 Create an inventory of all compute node IP addresses.

Example:

```
source ~/stackrc
openstack server list --flavor compute -f value -c Networks|cut -d=  -f2 >~/compute-nodes
```

If necessary, you can create an inventory of all controllers:

```
openstack server list --flavor control -f value -c Networks|cut -d= -f2  >>~/compute-nodes
```

Step 3 Install and enable the service.

Example:

```
ansible --become --inventory=compute-nodes all -m shell -u heat-admin -a "yum -y install
apicapi"
```

Step 4 Start the bond watch service.

Example:

```
ansible --become --inventory=compute-nodes all -m shell -u heat-admin -a "systemctl start
apic-bond-watch"
ansible --become --inventory=compute-nodes all -m shell -u heat-admin -a "systemctl enable
 apic-bond-watch"
```

For versions that do not define bond watch, you can start the service manually:

```
ansible --become --inventory=compute-nodes all -m shell -u heat-admin -a 'nohup
/usr/bin/apic-bond-watch&'
```

Identify Which NIC Is Active in the Bond

The `bond0` file in the `/proc/net/bonding` directory shows which of the two NICs is active.

Procedure

Examine the `bond0` to see which NIC is active.

Example:

```
[root@overcloud-compute-0 heat-admin]# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)
```

```
Bonding Mode: fault-tolerance (active-backup)
Primary Slave: None
Currently Active Slave: enp13s0
MII Status: up
MII Polling Interval (ms): 1
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: enp13s0
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 3
Permanent HW addr: 00:25:b5:00:00:0f
Slave queue ID: 0

Slave Interface: enp14s0
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 3
Permanent HW addr: 00:25:b5:00:00:10
Slave queue ID: 0
[root@overcloud-compute-0 heat-admin]#
```

Set the NIC MTU

Set and verify the maximum transmission unit (MTU) of the NICs. The MTU is based on settings that you specify in the Cisco Unified Computing System (UCS) Manager.

Procedure

Step 1 Set the MTU of the NICs to either 1600 or 9000.

Step 2 Verify the MTU setting by navigating to the UCS B-Series server, choosing the NIC, and then checking the value in the MTU field.

Verify MTU Settings for the NICs

Check the maximum transmission unit (MTU) setting on a NIC.

Procedure

Enter the following command:

```
ip link
```

You should see output similar to the following:

```
5: enp13s0: <BROADCAST,MULTICAST> mtu 9000 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 00:25:b5:00:00:03 brd ff:ff:ff:ff:ff:ff
```

```
6: enp14s0: <BROADCAST,MULTICAST> mtu 9000 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 00:25:b5:00:00:04 brd ff:ff:ff:ff:ff:ff
```

Configuration on Cisco UCS

Configure Cisco Unified Computing System (UCS) B-Series properly to integrate it with the Cisco Application Centric Infrastructure (ACI) and OpenStack. A supportable configuration must include the following:

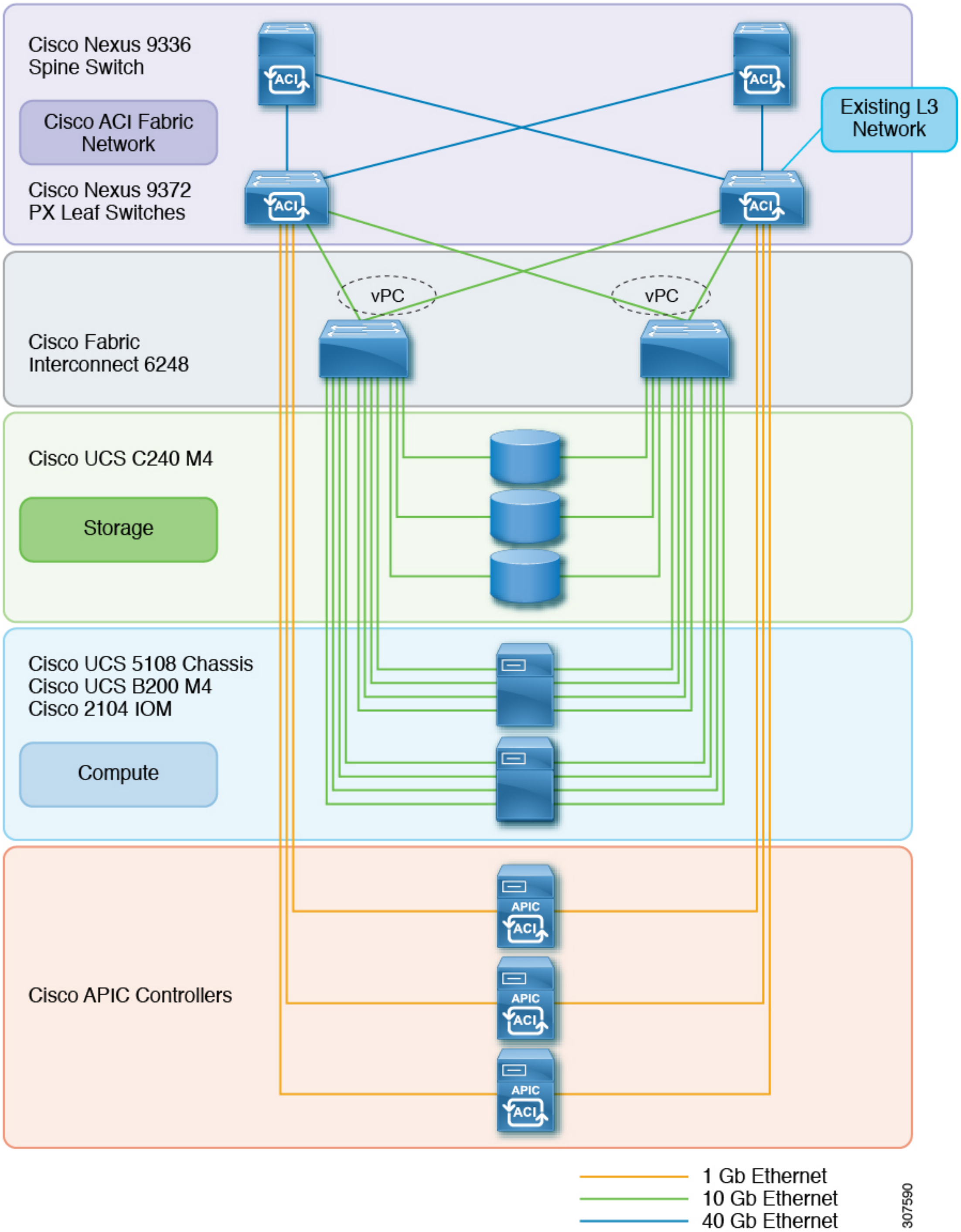
- Configuring the Cisco UCS service profile with two NICs. These NICs are used for OpFlex communication and VM data path.
- Disabling fabric failover on these NICs.
- Connecting the virtual NICs (vNICs) to different fabric interconnects.
- Setting the maximum transmission unit (MTU) on the vNICs.
- Ensuring that the VNICs allow the desired VLANs.
- Turning multicast on for the fabric interconnects.
- Configuring a port channel interface policy on the fabric interconnects.

Configuration on Leaf Switches

For path redundancy, configure a virtual port channel (vPC) interface policy across the two leaf switches. There are different ways to configure a virtual port channel (vPC), see the *Cisco APIC Layer 2 Networking Configuration Guide* for details.

Regardless of the method you choose to configure the UCS and vPCs, the configuration should resemble the following illustration:

Figure 5: Configuration on Leaf Switches



APPENDIX C

OpenStack Configuring Ironic

Ironic for OpenStack with Cisco ACI

Beginning in the 5.0(1) OpenStack plug-in release for Cisco Application Policy Infrastructure Controller (APIC), the use of Ironic provisioning of bare metal servers is supported.

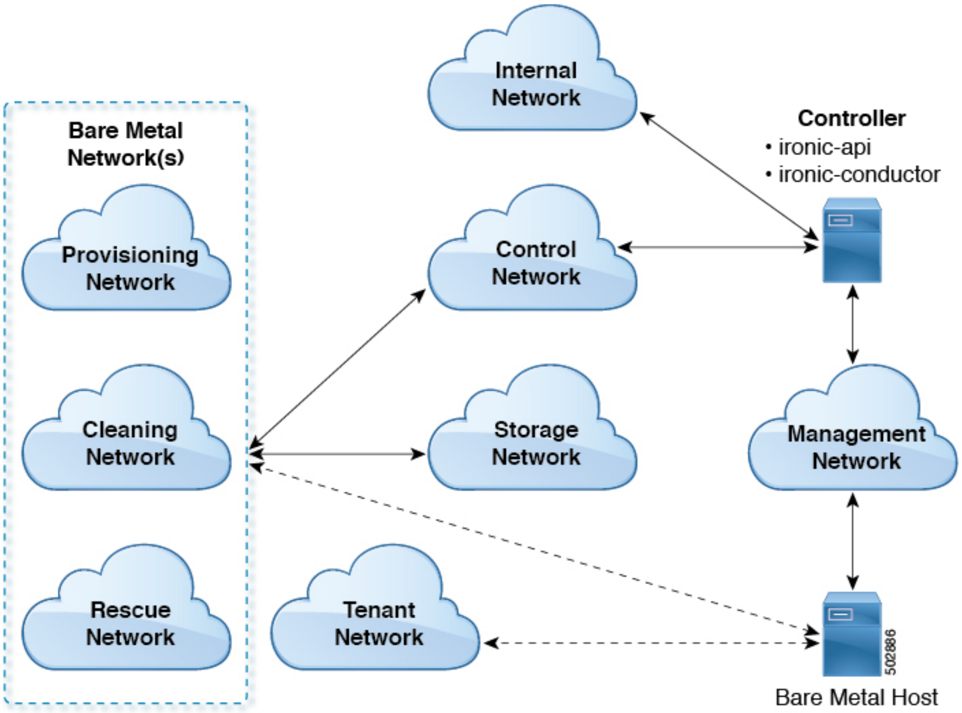
Ironic, which is deployed on the OpenStack overcloud, integrates with OpenStack services, such as Compute (Nova) and Network (Neutron). You can use Ironic on its own or as part of an OpenStack cloud. Using Ironic with OpenStack enables you to provision both virtual machines and bare metal servers on the same OpenStack cluster.

This appendix provides information about changes for Ironic in the 5.0(1) release when using the Cisco Application Centric Infrastructure (ACI) plug-in for OpenStack.

Ironic In Your Network

Neutron manages connectivity between bare metal hosts and switches. Ironic maintains an inventory of bare metal hosts, and connections between the hosts's NICs and ports on switches. Ironic provides this inventory information to Neutron when bare metal host ports must be connected to Neutron networks. Mechanism drivers use this information to configure switch ports with the appropriate VLANs for the Neutron networks.

The following illustration shows networks when you use Ironic.



Ironic defines three logical networks, collectively called bare metal networks, which are named for their purpose in a bare metal instance's lifecycle:

- Provisioning: Configure instances for bringup
- Cleaning: Erase hard disk and other security issues
- Rescue: Attach instances for recovery

Bare metal networks are regular networks in Neutron created by administrators. However, they should be created so that they are only visible to the admin project in OpenStack. (Create them using the `--no-share` option). All three logical networks can be implemented on the same Neutron network, separate Neutron networks, or any combination of networks. Using the provision network provides more security than using a single provider network for both provisioning and tenant network connectivity.

Workflow for Configuring and Deploying Ironic

This section provides a high-level overview of the tasks that you need to complete to configure and deploy OpenStack Ironic bare metal instances.

1. Fulfill all the prereqisites.

   See the section Prerequisites for Deploying OpenStack to Support Ironic, on page 45 in this guide.

2. Create the template required to deploy Ironic services in the overcloud.

   See the section Predeployment Configuration, on page 46 in this guide.

3. Deploy the overcloud, following instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

4. Create and upload images to the overcloud, following instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

5. Create the Ironic cleaning, provisioning, and rescue networks, which are collectively known as the bare metal networks.

   See the section Create the Bare Metal Networks, on page 55 in this guide.

6. Connect bare metal networks to Ironic services. The connection is required for contracts to be applied and for the addition of Cisco Application Centric Infrastructure (ACI) policies for extra protection of OpenStack services.

   See the section Connect the Bare Metal Networks to Ironic Services, on page 56 in this guide.

7. Enroll bare metal nodes, following instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

8. Create bare metal ports and port groups; although the procedure is almost the same as the one in Red Hat OpenStack Platform 13 documentation, you must specify additional topology information.

   See the section Create the Bare Metal Ports and Port Groups, on page 57 in this guide.

9. Create the bare metal flavors and availability zones, following instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

10. Launch the bare metal instances, following instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

Prerequisites for Deploying OpenStack to Support Ironic

You must complete the following tasks before you deploy OpenStack to support Ironic:

- Ensure that Red Hat OpenStack Platform (OSP)13 is installed with Cisco Application Centric Infrastructure (ACI) ML2 plug-in version 5.0(1) or above.

  Note Ironic bare metal provisioning is supported only for Red Hat OSP13.

- Ensure that all the OpenStack servers are connected only to Nexus 9000 EX, FX or GX switches.

- Decide the location of the Ironic services.

  Ironic adds the following services to the deployment:

  - **ironic-api**: Is the application programming interface for Ironic deployment.
  - **ironic-conductor**: Creates the conductor manager, which performs all actions on bare metal resources.
  - **Preboot Execution (PXE) server**: Allows networked computers that are not yet loaded with an operating system to be configured and booted remotely by an administrator. For HTTP, TFTP or both.

The recommended approach is to use a custom composable network to host the Ironic services. Follow the guidelines in the OSP13 documentation for creating custom composable networks, as well as the Ironic section that describes how to target Ironic services on this network.

- Define interfaces on the controller and their associated bridge mappings.

  The OpenStack Platform (OSP) ensures that each controller host has an interface on the undercloud control network. That means that if the default configuration is used, Ironic services will use it for their network interface. However, if you use the ServiceNetMap parameter to specify a different network for Ironic services, you will need to configure additional network interfaces. See the page "OpenStack Provisioning" for OSP13 documentation on the Red Hat website.

Ironic services are typically configured to run on a controller node. Like other services, they need IP addresses, service authentication in OpenStack Identity (Keystone), and configuration files with information on how to reach the message bus, database, and other services in OpenStack.

Predeployment Configuration

The Red Hat OpenStack Platform13 Director documentation explains the additional template configuration needed to deploy Ironic services in the overcloud. There is no new template configuration that is specific to the Cisco Application Centric Infrastructure (ACI) ML2 plug-in integration for OpenStack, when used with Ironic services.

If you use virtual port channels (vPCs) to connect the bare metal servers, then the Port Channel Policy assigned to the Leaf Interface Policy Group must have only the following control parameters:

- Fast Select Hot Standby Ports

- Graceful Convergence

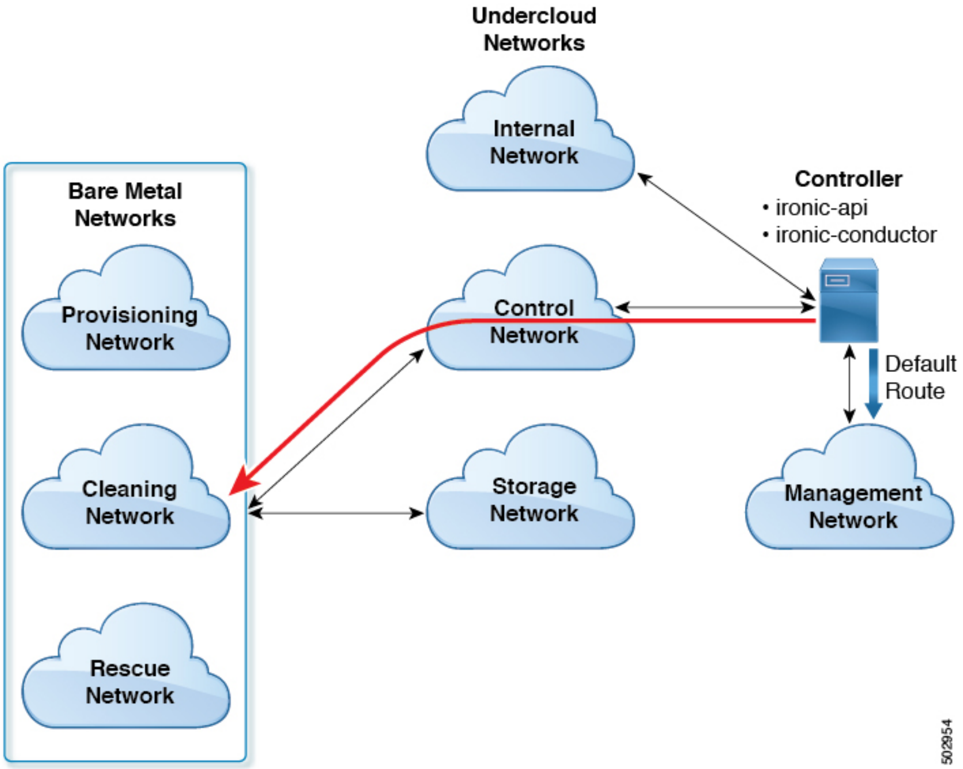
The Port Channel Policy should have the mode configured for “LACP Active.” This configuration is needed because bare metal instances only use a single NIC when Unified Extensible Firmware Interface (UEFI) boots and attaches to bare metal networks. Both ports are used when connecting to tenant networks. Also, the "Suspend Individual Port" option should be disabled.

If a custom composable network is used for the Ironic services, then an endpoint group (EPG) and bridge domain must be created in Cisco ACI to implement that network. This is the same for any undercloud network implemented in Cisco ACI, as described in the section “Setting Up the Cisco APIC and the Network” section in the Cisco ACI Installation Guide for Red Hat OpenStack Using the OpenStack Platform 13 Director guide.

Regardless of where the Ironic services are created—for example, on the undercloud control network or on a custom composable network—a subnet must be configured in the bridge domain that implements the undercloud network hosting the Ironic services. This subnet must have a CIDR IP address that matches the subnet used for the undercloud network.

For example, if the control plane network is used for the service (default), the default value for the subnet is 1.100.1.0/24. Therefore, some IP address on this subnet should be used as the CIDR configured in the subnet in this bridge domain.

The following diagram shows the different paths that the controller should use to reach the bare metal networks. The red arrow going from the Controller to the Bare Metal networks shows the path that needs to be established (as opposed to the default route). The two-headed black arrows show connections: the controller is connected to the internal, control, and management networks, and the control and storage networks are connected to the bare metal networks.



Note When using the control network for Ironic services, the IP address should be something other than 1.100.1.1, because the OpenStack Platform allocates this IP address for the undercloud virtual machine (VM).

This subnet configured in the bridge domain is used as the next-hop IP address for routes that are added after deployment so that the Ironic services can reach the bare metal networks.

Deploy the Overcloud

Before you begin

Perform the tasks in the section Prerequisites for Deploying OpenStack to Support Ironic, on page 45.

Procedure

Deploy the Red Hat OpenStack Platform overcloud.

Follow the instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

Create and Upload Bare Metal Images to the Overcloud

Before you begin

- Perform the tasks in the section Prerequisites for Deploying OpenStack to Support Ironic, on page 45.
- Deploy the overcloud.

Procedure

Create and upload images to the overcloud, following instructions in Red Hat OpenStack Platform 13 documentation on the Red Hat website.

Creating Contracts to Allow Traffic Between EPGs

Policy is needed to allow traffic between the endpoint group (EPG) for the undercloud network that hosts Ironic services and any EPGs for overcloud bare metal networks. Because Ironic services may be on the same network as other services, the policy should be limited to only the traffic needed between Ironic services and bare metal instances. The contracts, subjects, filters, and filter entries for these policies need to cover the protocols in this section.

Required Contracts

- Contracts that must be provided by Ironic services, consumed by bare metal networks:
  - PXE-TFTP (UDP destination port 69)
  - PXE-HTTP (TCP, default destination port 8088)

    Confirm the port in use by running the following script on the controller for OSP13 installations:

    ```
    # egrep ^http_url /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf
      \
       | awk -F":" '{print $3}'
    ```

  - Swift API for direct-deploy interfaces (TCP, default destination port 8080)

    Check which port is used, and run the following script on the controller for OSP13 installations:

    ```
    # egrep -A 1 'listen swift' \
       /var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg \
       | awk -F":" '{print $2}' | awk '{print $1}'
    ```

  - Ironic API (TCP, default destination port 6385)

    Check which port is used, and run the following script on the controller for OSP13 installations:

    ```
    # egrep -A 1 'listen ironic' \
       /var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg \
       | awk -F":" '{print $2}' | awk '{print $1}'
    ```

- Contracts that must be provided by bare metal networks, consumed by Ironic services:

- RAMDISK Server (TCP destination port 9999)
- iSCSI for iSCSI deploy interfaces (TCP destination port 3260)

Note Contracts for on-network traffic, such as DHCP and ARP aren't needed, as the on-network traffic is intra-EPG traffic.

Commands for Creating Contracts

Contracts can be created using **aimctl** commands, or directly with the Cisco Application Policy Infrastructure Controller (APIC) GUI or CLI. See the following examples of XML commands that you can run on OpenStack Overcloud controller node in order to create the contracts in Cisco Application Centric Infrastructure (ACI):

```
<polUni>
  <fvTenant name="common">

      <!-- Here are the filters ironic-to-instances -->

      <vzFilter childAction="" descr="" dn="uni/tn-common/flt-ironic-to-instances"
extMngdBy="" fwdId="51" id="implicit" lcOwn="local" modTs="2020-05-14T16:47:26.264+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-to-instances" nameAlias="" ownerKey=""
 ownerTag="" revId="52" status="" txId="0" uid="15374" unsupportedEntries="no"
unsupportedMgmtEntries="no" userdom="" usesIds="yes">
          <vzRsFwdRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.264+00:00" rType="mo" rn="rsFwdRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-51" tType="mo"/>
          <vzRsRevRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.264+00:00" rType="mo" rn="rsRevRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-52" tType="mo"/>
         <vzRtSubjFiltAtt childAction="" lcOwn="local" modTs="2020-05-14T16:47:26.138+00:00"
 rn="rtsubjFiltAtt-[uni/tn-common/brc-ironic-services/subj-ironic-services]" status=""
tCl="vzSubj" tDn="uni/tn-common/brc-ironic-services/subj-ironic-services"/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="8088"
dToPort="8088" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:26.221+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-pxe-http" nameAlias="" prot="tcp"
rn="e-ironic-pxe-http" sFromPort="unspecified" sToPort="unspecified" stateful="yes" status=""
 tcpRules="" uid="15374" userdom=""/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="6385"
dToPort="6385" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:26.179+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-api" nameAlias="" prot="tcp"
rn="e-ironic-api" sFromPort="unspecified" sToPort="unspecified" stateful="yes" status=""
tcpRules="" uid="15374" userdom=""/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="69"
dToPort="69" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:26.264+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp" nameAlias="" prot="udp"
rn="e-ironic-tftp" sFromPort="unspecified" sToPort="unspecified" stateful="yes" status=""
tcpRules="" uid="15374" userdom=""/>
      </vzFilter>


      <!-- Here are the filters instances-to-ironic -->

      <vzFilter  childAction="" descr="" dn="uni/tn-common/flt-instances-to-ironic"
extMngdBy="" fwdId="35" id="implicit" lcOwn="local" modTs="2020-05-14T16:47:25.921+00:00"
monPolDn="uni/tn-common/monepg-default" name="instances-to-ironic" nameAlias="" ownerKey=""
```

```
 ownerTag="" revId="46" status="" txId="0" uid="15374" unsupportedEntries="no"
unsupportedMgmtEntries="no" userdom="" usesIds="yes">
          <vzRsFwdRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:25.921+00:00" rType="mo" rn="rsFwdRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-35" tType="mo"/>
          <vzRsRevRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:25.921+00:00" rType="mo" rn="rsRevRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-46" tType="mo"/>
         <vzRtSubjFiltAtt childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.762+00:00"
 rn="rtsubjFiltAtt-[uni/tn-common/brc-ironic-agents/subj-ironic-agents]" status=""
tCl="vzSubj" tDn="uni/tn-common/brc-ironic-agents/subj-ironic-agents"/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="3260"
dToPort="3260" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:25.809+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-iscsi" nameAlias="" prot="tcp"
rn="e-ironic-iscsi" sFromPort="unspecified" sToPort="unspecified" stateful="yes" status=""
 tcpRules="" uid="15374" userdom=""/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="9999"
dToPort="9999" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:25.921+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-ramdisk" nameAlias="" prot="tcp"
rn="e-ironic-ramdisk" sFromPort="unspecified" sToPort="unspecified" stateful="yes" status=""
 tcpRules="" uid="15374" userdom=""/>
      </vzFilter>

      <!-- Here are the filters for ironic-tftp-client -->

      <vzFilter  childAction="" descr="" dn="uni/tn-common/flt-ironic-tftp-client"
extMngdBy="" fwdId="29" id="implicit" lcOwn="local" modTs="2020-05-14T16:47:26.018+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp-client" nameAlias="" ownerKey=""
 ownerTag="" revId="29" status="" txId="0" uid="15374" unsupportedEntries="no"
unsupportedMgmtEntries="no" userdom="" usesIds="yes">
          <vzRsFwdRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.018+00:00" rType="mo" rn="rsFwdRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-29" tType="mo"/>
          <vzRsRevRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.018+00:00" rType="mo" rn="rsRevRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-29" tType="mo"/>
         <vzRtSubjFiltAtt childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.973+00:00"
 rn="rtsubjFiltAtt-[uni/tn-common/brc-ironic-tftp-client/subj-ironic-tftp-client]" status=""
 tCl="vzSubj" tDn="uni/tn-common/brc-ironic-tftp-client/subj-ironic-tftp-client"/>
          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction=""
dFromPort="unspecified" dToPort="unspecified" descr="" etherT="ip" extMngdBy=""
icmpv4T="unspecified" icmpv6T="unspecified" lcOwn="local" matchDscp="unspecified"
modTs="2020-05-14T16:47:26.018+00:00" monPolDn="uni/tn-common/monepg-default"
name="ironic-tftp-client" nameAlias="" prot="udp" rn="e-ironic-tftp-client"
sFromPort="unspecified" sToPort="unspecified" stateful="no" status="" tcpRules="" uid="15374"
 userdom=""/>
      </vzFilter>

      <!-- Here are the filters for ironic-tftp-server -->

      <vzFilter  childAction="" descr="" dn="uni/tn-common/flt-ironic-tftp-server"
extMngdBy="" fwdId="47" id="implicit" lcOwn="local" modTs="2020-05-14T16:47:26.096+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp-server" nameAlias="" ownerKey=""
 ownerTag="" revId="50" status="" txId="0" uid="15374" unsupportedEntries="no"
unsupportedMgmtEntries="no" userdom="" usesIds="yes">
          <vzRsRevRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.096+00:00" rType="mo" rn="rsRevRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-50" tType="mo"/>
          <vzRsFwdRFltPAtt childAction="deleteNonPresent" forceResolve="yes" lcOwn="local"
 modTs="2020-05-14T16:47:26.096+00:00" rType="mo" rn="rsFwdRFltPAtt" state="formed"
stateQual="none" status="" tCl="vzAFilterableUnit" tDn="uni/tn-common/fp-47" tType="mo"/>
          <vzRtFiltAtt childAction="" lcOwn="local" modTs="2020-05-14T16:47:26.057+00:00"
```

```
rn="rtfiltAtt-[uni/tn-common/brc-ironic-tftp-server/subj-ironic-tftp-server/intmnl]" status=""
 tCl="vzInTerm" tDn="uni/tn-common/brc-ironic-tftp-server/subj-ironic-tftp-server/intmnl"/>

          <vzEntry  applyToFrag="no" arpOpc="unspecified" childAction="" dFromPort="69"
dToPort="69" descr="" etherT="ip" extMngdBy="" icmpv4T="unspecified" icmpv6T="unspecified"
 lcOwn="local" matchDscp="unspecified" modTs="2020-05-14T16:47:26.096+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp-server" nameAlias="" prot="udp"
rn="e-ironic-tftp-server" sFromPort="unspecified" sToPort="unspecified" stateful="no"
status="" tcpRules="" uid="15374" userdom=""/>
      </vzFilter>

      <!-- Here are the contracts for ironic-agents -->

     <vzBrCP  childAction="" configIssues="" descr="" dn="uni/tn-common/brc-ironic-agents"
 extMngdBy="" intent="install" lcOwn="local" modTs="2020-05-14T16:47:25.921+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-agents" nameAlias="" ownerKey=""
ownerTag="" prio="unspecified" reevaluateAll="no" scope="context" status=""
targetDscp="unspecified" uid="15374" userdom="">
          <vzDirAssDef childAction="deleteNonPresent" descr="" lcOwn="local"
modTs="2020-05-14T16:47:24.933+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" ownerKey="" ownerTag="" rn="dirass" status="">
              <vzConsDef anyDn=""
bdDefDn="uni/bd-[uni/tn-common/BD-sauto-osd-bd-vlan105]-isSvc-no" bdDefStQual="none"
childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105" exceptionTag=""
intent="install" isAny="no" isCifLblIntersEmpty="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" modTs="2020-05-14T16:47:24.933+00:00"
monPolDn="uni/tn-common/monepg-default" name="sauto-osd-epg-vlan105" nameAlias="" ownerKey=""
 ownerTag="" pcEnfPref="enforced" pcTag="49155" prefGrMemb="exclude" prio="unspecified"
rn="cons-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]-any-no"
scopeId="2555905" status="" targetDscp="unspecified" txId="1729382256910271363"
useAnyDef="no"/>
              <vzProvDef anyDn=""
bdDefDn="uni/bd-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/BD-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-isSvc-no"
 bdDefStQual="none" childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
 exceptionTag="" hasTriggeredAlloc="no" intent="install" isAny="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" matchT="AtleastOne"
modTs="2020-05-14T16:47:24.956+00:00" monPolDn="uni/tn-common/monepg-default"
name="net_1005e02b-2ed2-44eb-8ac5-6d989176ebef" nameAlias="" ownerKey="" ownerTag=""
pcEnfPref="enforced" pcTag="49159" prefGrMemb="exclude" prio="unspecified"
rn="prov-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-any-no"
 scopeId="2555905" status="" targetDscp="unspecified" txId="7493989779947781613"
useAnyDef="no"/>
          </vzDirAssDef>
          <vzSubj  childAction="" configIssues="" consMatchT="AtleastOne" descr=""
extMngdBy="" lcOwn="local" modTs="2020-05-14T16:47:25.010+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-agents" nameAlias="" prio="unspecified"
 provMatchT="AtleastOne" revFltPorts="yes" rn="subj-ironic-agents" status=""
targetDscp="unspecified" uid="15374" userdom="">
              <vzRsSubjFiltAtt action="permit"  childAction="" directives="" extMngdBy=""
forceResolve="yes" lcOwn="local" modTs="2020-05-14T16:47:25.762+00:00"
monPolDn="uni/tn-common/monepg-default" priorityOverride="default" rType="mo"
rn="rssubjFiltAtt-instances-to-ironic" state="formed" stateQual="none" status=""
tCl="vzFilter" tContextDn="" tDn="uni/tn-common/flt-instances-to-ironic"
tRn="flt-instances-to-ironic" tType="name" tnVzFilterName="instances-to-ironic" uid="15374"
 userdom=""/>
          </vzSubj>
          <vzRtProv childAction="" lcOwn="local" modTs="2020-05-14T16:47:24.956+00:00"
rn="rtfvProv-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]"
```

```
 status="" tCl="fvAEPg"
tDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"/>

          <vzRtCons childAction="" lcOwn="local" modTs="2020-05-14T16:47:24.933+00:00"
rn="rtfvCons-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]" status=""
tCl="fvAEPg" tDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105"/>
      </vzBrCP>

      <!-- Here are the contracts for ironic-services -->

     <vzBrCP  childAction="" configIssues="" descr="" dn="uni/tn-common/brc-ironic-services"
 extMngdBy="" intent="install" lcOwn="local" modTs="2020-05-14T16:47:26.264+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-services" nameAlias="" ownerKey=""
ownerTag="" prio="unspecified" reevaluateAll="no" scope="context" status=""
targetDscp="unspecified" uid="15374" userdom="">
          <vzDirAssDef childAction="deleteNonPresent" descr="" lcOwn="local"
modTs="2020-05-14T16:47:25.067+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" ownerKey="" ownerTag="" rn="dirass" status="">
              <vzConsDef anyDn=""
bdDefDn="uni/bd-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/BD-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-isSvc-no"
 bdDefStQual="none" childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
 exceptionTag="" intent="install" isAny="no" isCifLblIntersEmpty="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" modTs="2020-05-14T16:47:25.081+00:00"
monPolDn="uni/tn-common/monepg-default" name="net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
nameAlias="" ownerKey="" ownerTag="" pcEnfPref="enforced" pcTag="49159" prefGrMemb="exclude"
 prio="unspecified"
rn="cons-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-any-no"
 scopeId="2555905" status="" targetDscp="unspecified" txId="7493989779947781613"
useAnyDef="no"/>
              <vzProvDef anyDn=""
bdDefDn="uni/bd-[uni/tn-common/BD-sauto-osd-bd-vlan105]-isSvc-no" bdDefStQual="none"
childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105" exceptionTag=""
hasTriggeredAlloc="no" intent="install" isAny="no" isMsiteEPg="no" l3CtxEncap="vxlan-2555905"
 lcOwn="local" matchT="AtleastOne" modTs="2020-05-14T16:47:25.067+00:00"
monPolDn="uni/tn-common/monepg-default" name="sauto-osd-epg-vlan105" nameAlias="" ownerKey=""
 ownerTag="" pcEnfPref="enforced" pcTag="49155" prefGrMemb="exclude" prio="unspecified"
rn="prov-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]-any-no"
scopeId="2555905" status="" targetDscp="unspecified" txId="1729382256910271363"
useAnyDef="no"/>
          </vzDirAssDef>
          <vzSubj  childAction="" configIssues="" consMatchT="AtleastOne" descr=""
extMngdBy="" lcOwn="local" modTs="2020-05-14T16:47:25.117+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-services" nameAlias="" prio="unspecified"
 provMatchT="AtleastOne" revFltPorts="yes" rn="subj-ironic-services" status=""
targetDscp="unspecified" uid="15374" userdom="">
              <vzRsSubjFiltAtt action="permit"  childAction="" directives="" extMngdBy=""
forceResolve="yes" lcOwn="local" modTs="2020-05-14T16:47:26.138+00:00"
monPolDn="uni/tn-common/monepg-default" priorityOverride="default" rType="mo"
rn="rssubjFiltAtt-ironic-to-instances" state="formed" stateQual="none" status=""
tCl="vzFilter" tContextDn="" tDn="uni/tn-common/flt-ironic-to-instances"
tRn="flt-ironic-to-instances" tType="name" tnVzFilterName="ironic-to-instances" uid="15374"
 userdom=""/>
          </vzSubj>
          <vzRtProv childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.067+00:00"
rn="rtfvProv-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]" status=""
tCl="fvAEPg" tDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105"/>
          <vzRtCons childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.081+00:00"
rn="rtfvCons-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]"
```

```
 status="" tCl="fvAEPg"
tDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"/>

      </vzBrCP>

      <!-- Here are the contracts for ironic-tftp-client -->

      <vzBrCP  childAction="" configIssues="" descr=""
dn="uni/tn-common/brc-ironic-tftp-client" extMngdBy="" intent="install" lcOwn="local"
modTs="2020-05-14T16:47:26.018+00:00" monPolDn="uni/tn-common/monepg-default"
name="ironic-tftp-client" nameAlias="" ownerKey="" ownerTag="" prio="unspecified"
reevaluateAll="no" scope="context" status="" targetDscp="unspecified" uid="15374" userdom="">

          <vzDirAssDef childAction="deleteNonPresent" descr="" lcOwn="local"
modTs="2020-05-14T16:47:25.168+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" ownerKey="" ownerTag="" rn="dirass" status="">
              <vzConsDef anyDn=""
bdDefDn="uni/bd-[uni/tn-common/BD-sauto-osd-bd-vlan105]-isSvc-no" bdDefStQual="none"
childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105" exceptionTag=""
intent="install" isAny="no" isCifLblIntersEmpty="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" modTs="2020-05-14T16:47:25.168+00:00"
monPolDn="uni/tn-common/monepg-default" name="sauto-osd-epg-vlan105" nameAlias="" ownerKey=""
 ownerTag="" pcEnfPref="enforced" pcTag="49155" prefGrMemb="exclude" prio="unspecified"
rn="cons-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]-any-no"
scopeId="2555905" status="" targetDscp="unspecified" txId="1729382256910271363"
useAnyDef="no"/>
              <vzProvDef anyDn=""
bdDefDn="uni/bd-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/BD-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-isSvc-no"
 bdDefStQual="none" childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
 exceptionTag="" hasTriggeredAlloc="no" intent="install" isAny="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" matchT="AtleastOne"
modTs="2020-05-14T16:47:25.189+00:00" monPolDn="uni/tn-common/monepg-default"
name="net_1005e02b-2ed2-44eb-8ac5-6d989176ebef" nameAlias="" ownerKey="" ownerTag=""
pcEnfPref="enforced" pcTag="49159" prefGrMemb="exclude" prio="unspecified"
rn="prov-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-any-no"
 scopeId="2555905" status="" targetDscp="unspecified" txId="7493989779947781613"
useAnyDef="no"/>
          </vzDirAssDef>
          <vzSubj  childAction="" configIssues="" consMatchT="AtleastOne" descr=""
extMngdBy="" lcOwn="local" modTs="2020-05-14T16:47:25.252+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp-client" nameAlias=""
prio="unspecified" provMatchT="AtleastOne" revFltPorts="yes" rn="subj-ironic-tftp-client"
status="" targetDscp="unspecified" uid="15374" userdom="">
              <vzRsSubjFiltAtt action="permit"  childAction="" directives="" extMngdBy=""
forceResolve="yes" lcOwn="local" modTs="2020-05-14T16:47:25.973+00:00"
monPolDn="uni/tn-common/monepg-default" priorityOverride="default" rType="mo"
rn="rssubjFiltAtt-ironic-tftp-client" state="formed" stateQual="none" status="" tCl="vzFilter"
 tContextDn="" tDn="uni/tn-common/flt-ironic-tftp-client" tRn="flt-ironic-tftp-client"
tType="name" tnVzFilterName="ironic-tftp-client" uid="15374" userdom=""/>
          </vzSubj>
          <vzRtProv childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.189+00:00"
rn="rtfvProv-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]"
 status="" tCl="fvAEPg"
tDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"/>

          <vzRtCons childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.168+00:00"
rn="rtfvCons-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]" status=""
tCl="fvAEPg" tDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105"/>
```

```
      </vzBrCP>

      <!-- Here are the contracts for ironic-tftp-server -->
      <vzBrCP  childAction="" configIssues="" descr=""
dn="uni/tn-common/brc-ironic-tftp-server" extMngdBy="" intent="install" lcOwn="local"
modTs="2020-05-14T16:47:26.096+00:00" monPolDn="uni/tn-common/monepg-default"
name="ironic-tftp-server" nameAlias="" ownerKey="" ownerTag="" prio="unspecified"
reevaluateAll="no" scope="context" status="" targetDscp="unspecified" uid="15374" userdom="">

          <vzDirAssDef childAction="deleteNonPresent" descr="" lcOwn="local"
modTs="2020-05-14T16:47:25.310+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" ownerKey="" ownerTag="" rn="dirass" status="">
              <vzConsDef anyDn=""
bdDefDn="uni/bd-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/BD-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-isSvc-no"
 bdDefStQual="none" childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
 exceptionTag="" intent="install" isAny="no" isCifLblIntersEmpty="no" isMsiteEPg="no"
l3CtxEncap="vxlan-2555905" lcOwn="local" modTs="2020-05-14T16:47:25.324+00:00"
monPolDn="uni/tn-common/monepg-default" name="net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"
nameAlias="" ownerKey="" ownerTag="" pcEnfPref="enforced" pcTag="49159" prefGrMemb="exclude"
 prio="unspecified"
rn="cons-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]-any-no"
 scopeId="2555905" status="" targetDscp="unspecified" txId="7493989779947781613"
useAnyDef="no"/>
              <vzProvDef anyDn=""
bdDefDn="uni/bd-[uni/tn-common/BD-sauto-osd-bd-vlan105]-isSvc-no" bdDefStQual="none"
childAction="deleteNonPresent" ctrctUpd="ctrct"
ctxDefDn="uni/ctx-[uni/tn-common/ctx-sauto_l3out-1_vrf]" ctxDefStQual="none" ctxPcTag="49153"
 ctxSeg="2555905" descr=""
epgDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105" exceptionTag=""
hasTriggeredAlloc="no" intent="install" isAny="no" isMsiteEPg="no" l3CtxEncap="vxlan-2555905"
 lcOwn="local" matchT="AtleastOne" modTs="2020-05-14T16:47:25.310+00:00"
monPolDn="uni/tn-common/monepg-default" name="sauto-osd-epg-vlan105" nameAlias="" ownerKey=""
 ownerTag="" pcEnfPref="enforced" pcTag="49155" prefGrMemb="exclude" prio="unspecified"
rn="prov-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]-any-no"
scopeId="2555905" status="" targetDscp="unspecified" txId="1729382256910271363"
useAnyDef="no"/>
          </vzDirAssDef>
          <vzSubj  childAction="" configIssues="" consMatchT="AtleastOne" descr=""
extMngdBy="" lcOwn="local" modTs="2020-05-14T16:47:25.350+00:00"
monPolDn="uni/tn-common/monepg-default" name="ironic-tftp-server" nameAlias=""
prio="unspecified" provMatchT="AtleastOne" revFltPorts="yes" rn="subj-ironic-tftp-server"
status="" targetDscp="unspecified" uid="15374" userdom="">
              <vzInTerm  childAction="" descr="" extMngdBy="" lcOwn="local"
modTs="2020-05-14T16:47:25.350+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" prio="unspecified" rn="intmnl" status="" targetDscp="unspecified" uid="15374"
 userdom="">
              <vzInTerm  childAction="" descr="" extMngdBy="" lcOwn="local"
modTs="2020-05-14T16:47:25.350+00:00" monPolDn="uni/tn-common/monepg-default" name=""
nameAlias="" prio="unspecified" rn="intmnl" status="" targetDscp="unspecified" uid="15374"
 userdom="">
                  <vzRsFiltAtt action="permit"  childAction="" directives="" extMngdBy=""
forceResolve="yes" lcOwn="local" modTs="2020-05-14T16:47:26.057+00:00"
monPolDn="uni/tn-common/monepg-default" priorityOverride="default" rType="mo"
rn="rsfiltAtt-ironic-tftp-server" state="formed" stateQual="none" status="" tCl="vzFilter"
 tContextDn="" tDn="uni/tn-common/flt-ironic-tftp-server" tRn="flt-ironic-tftp-server"
tType="name" tnVzFilterName="ironic-tftp-server" uid="15374" userdom=""/>
              </vzInTerm>
          </vzSubj>
          <vzRtProv childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.310+00:00"
rn="rtfvProv-[uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105]" status=""
tCl="fvAEPg" tDn="uni/tn-common/ap-sauto_osd_infra_app/epg-sauto-osd-epg-vlan105"/>
```

```
          <vzRtCons childAction="" lcOwn="local" modTs="2020-05-14T16:47:25.324+00:00"
rn="rtfvCons-[uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef]"
 status="" tCl="fvAEPg"
tDn="uni/tn-prj_4660546efda44d6e9d9dba6670920894/ap-OpenStack/epg-net_1005e02b-2ed2-44eb-8ac5-6d989176ebef"/>

      </vzBrCP>
```

The newly created contracts must be applied to the EPG used for the undercloud network hosting the Ironic services. This can be done with the Cisco APIC GUI, or any other supported means.

Create the Bare Metal Networks

The Ironic cleaning, provisioning, and rescue networks are collectively known as the bare metal networks. You can use one Neutron network for all three purposes, or you can use separate networks for each. Because the Ironic services must communicate directly with the instances, you must take additional steps when creating these bare metal networks.

Before you begin

You must have completed the tasks in the Prerequisites for Deploying OpenStack to Support Ironic, on page 45 and the other sections preceding this one.

Procedure

Step 1 Create the bare metal networks using the **apic:extra_consumed_contracts** and **apic:extra_provided_contracts** extensions.

These extensions create lists of additional provided or consumed contracts that are applied once a subnet on the Neutron network is connected to a Neutron router. The following example shows how the extensions are used when creating the network, using a `baremetal-contract` that was previously created in the common tenant in Cisco Application Policy Infrastructure Controller (APIC):

Example:

```
# neutron net-create baremetal_network --shared \
--apic:extra_consumed_contracts list=true ironic-services \
--apic:extra_provided_contracts list=true ironic-agents
```

The contract should only allow the traffic types needed between the bare metal host and the Ironic services. The contract should also already be provided and consumed by the endpoint group (EPG) used for the undercloud network that hosts the Ironic services.

Step 2 Ensure that the networks belong to the same VRF in Cisco Application Centric Infrastructure (ACI) as the EPG that is used by the Ironic services.

For example, if the Ironic services are placed on the undercloud control network (default), then the VRF used for the bridge domain for the undercloud control network must also be used for the overcloud bare metal networks.

You can control the VRF that a network belongs to by using address scopes in OpenStack. The `apic:distinguished_names` extension allows you to create an address scope that maps to an existing VRF in Cisco ACI. The following example shows how to create in OpenStack an address scope that maps to the control VRF under the common tenant in Cisco ACI.

Example:

```
# neutron address-scope-create baremetal_as_v4 4 \
    --apic:distinguished_names type=dict VRF=uni/tn-common/ctx-control
```

Step 3 Associate a subnet pool with the VRF in Cisco ACI, and then use the subnet pool to allocate a subnet for the bare metal network:

Example:

```
# neutron subnetpool-create --pool-prefix 40.40.40.0/24 \
    --address-scope baremetal_as_v4 baremetal_v4_sp

# openstack subnet create --network baremetal_network \
    --subnet-pool baremetal_v4_sp --dhcp  baremetal_subnet \
    --prefix-length 24 baremetal-subnet
```

Step 4 Use the following commands to update the Ironic configuration file and use the newly created bare metal networks:

```
# crudini --set /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf neutron
 provisioning_network <provisioning network UUID>
# crudini --set /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf neutron
 cleaning_network <cleaning network UUID>
# crudini --set /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf neutron
 rescuing_network <rescuing network UUID>
```

Step 5 Restart the Ironic containers:

Example:

```
# docker restart ironic_pxe_http \
                 ironic_pxe_tftp \
                 ironic_conductor \
                 ironic_api
```

Connect the Bare Metal Networks to Ironic Services

Before you begin

You must have completed the tasks in the Prerequisites for Deploying OpenStack to Support Ironic, on page 45 and the other sections preceding this one.

Procedure

Step 1 Attach the Neutron network's subnet to Neutron router:

The contracts specified in the bare metal network extension are applied after you make the attachment.

Example:

```
# openstack router create baremetal-router
# openstack router add subnet baremetal-router baremetal-subnet
```

Step 2 Add routing table entries to the host running Ironic services.

The default route on a host is typically not through the interface used for Ironic services. Add subnet routes to ensure that the Ironic services interface is used to reach the bare metal networks. A subnet must be configured in the bridge domain used for the undercloud network that hosts the Ironic services, so that Cisco Application Policy Infrastructure Controller (APIC) can provide a subnet gateway for the services. (If the control plane network is used for Ironic services, the .1 address can’t be used for the CIDR, as that is allocated for use by the undercloud VM.)

The following is an example of a static route to reach a bare metal network with a 40.40.40.0/24 subnet, through the subnet gateway 1.100.1.254 (CIDR used for the undercloud control plane network):

```
# route add -net 40.40.40.0 netmask 255.255.255.0 \
   gateway 1.100.1.254 dev br-baremetal
```

To ensure that the route persists across reboots, add it to network initialization scripts. For example, /etc/sysconfig/network-scripts/route-br-baremetal.

Enroll the Bare Metal Nodes

Before you begin

You must have completed the tasks in the Prerequisites for Deploying OpenStack to Support Ironic, on page 45 and the other sections preceding this one.

Procedure

Enroll the bare metal nodes.

Follow the instructions in the section "Adding Physical Machines as Bare Metal Nodes" in *Red Hat OpenStack Platform 13 Bare Metal Provisioning* on the Red Hat website.

Create the Bare Metal Ports and Port Groups

This step is almost identical to the steps in the OpenStack Platform (OSP) 13 documentation. However, you must specify additional topology information for the Ironic ports, so that the fabric can be reconfigured correctly whenever the bare metal instances are connected to Neutron networks.

You specify the topology information using the `switch_id` parameter in the local-link-connection portion of the Ironic port. This parameter is a string, which consists of a set of key:value pairs, separated by commas. Three parameters are currently supported:

- `apic_dn`:

  This parameter provides the distinguished name (DN) in Cisco Application Policy Infrastructure Controller (APIC) that is used for the static path binding. The following is the format for the DN when a single interface is used:

  ```
  topology/pod-pod number/paths-node/pathep-port or policy group name
  ```

For example:

`topology/pod-1/paths-101/pathep-[eth1/2]`

The following is the format for the DN when a virtual port channel (vPC) is used:

`topology/pod-`*`pod number`*`/protpaths-`*`nodes`*`/pathep-`*`vpc`*

- `physical_network`:

  This parameter is the physical network used if a VLAN must be allocated for the static path.

- `physical_domain`:

  This parameter is the name of the PhysDom that will be associated with the endpoint group (EPG) that the static port is created on. This parameter is optional.

This section shows how to set the topology information in the local_link_connection field in the bare metal ports. It is assumed that the bare metal node ID and its associated bare metal port IDs are defined in their respective environment variables.

Before you begin

You must have completed the tasks in the Prerequisites for Deploying OpenStack to Support Ironic, on page 45 and the other sections preceding this one.

Procedure

Step 1 Update the bare metal ports with the topology information:

Example:

```
# export VPC=”topology/pod-1/protpaths-501-502/pathep-[ironic-vpc]”
# openstack baremetal port set --node ${NODE_ID} \
    --local-link-connection port_id="Eth1/1" \
    --local-link-connection switch_id="00:be:75:8f:25:a1"  \
    --local-link-connection \
      switch_info="apic_dn:${VPC},physical_network:physnet1 \
    ${BAREMETAL_PORT1_ID}
# openstack baremetal port set --node ${NODE_ID} \
    --local-link-connection port_id="Eth1/1" \
    --local-link-connection switch_id="00:be:75:8f:25:a2"  \
    --local-link-connection \
      switch_info="apic_dn:${VPC},physical_network:physnet1 \
${BAREMETAL_PORT2_ID}
```

Note that although the `port_id` and `switch_id` are not currently used by the plug-in, they still must be populated.

Step 2 Create the bare metal port group:

Example:

```
# PORT_GROUP_ID=$(openstack baremetal port group create
    --node ${NODE_ID} --name bond1 --address 00:fc:ba:e8:86:4c \
    --mode 802.3ad --property miimon=100 \
    --property xmit_hash_policy='layer2' --support-standalone-ports
```

Step 3 Add the Ironic ports to the Ironic port group:

Example:

```
# openstack baremetal port set --node ${NODE_ID} \
     --port-group ${PORT_GROUP_ID} ${BAREMETAL_PORT1_ID}

# openstack baremetal port set --node ${NODE_ID} \
     --port-group ${PORT_GROUP_ID} ${BAREMETAL_PORT2_ID}
```

Create the Bare Metal Flavors and Availability Zones

Before you begin

You must have completed the tasks in the Prerequisites for Deploying OpenStack to Support Ironic, on page 45 and the other sections preceding this one.

Procedure

Create the bare metal ports and port groups.

Follow the instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

Launch the Bare Metal Instances

Before you begin

Perform the tasks in the section Prerequisites for Deploying OpenStack to Support Ironic, on page 45.

Procedure

Launch the bare metal instances.

Follow the instructions in the Red Hat OpenStack Platform 13 documentation on the Red Hat website.

APPENDIX D

Advanced Configuration

Advanced Configuration

For large installations using VXLAN encapsulation for Virtual Machine Manager (VMM) domains, you may need to make extra configurations.

You can configure the number of multicast groups to match the maximum number of endpoint groups (EPGs) for the host. You can also increase the maximum auxiliary memory for sockets and add extra kernel boot parameters to the compute or controller nodes.

Configure Multicast Groups and Increase Memory for Sockets

Procedure

Configure multicast groups and increase memory for sockets adding the following parameters to parameter_defaults in the deployment template:

Example:

```
parameter_defaults:
  ControllerParameters:
      ExtraSysctlSettings:
         net.ipv4.igmp_max_memberships:
             value: 4096
         net.core.optmem_max:
             value: 1310720
   ComputeParameters:
      ExtraSysctlSettings:
         net.ipv4.igmp_max_memberships:
             value: 1024
```

The IGMP maximum memberships value should be greater than or equal to the number of Neutron networks that the host has Neutron ports on. For example, if a compute host has 100 instances, and each instance is on a different Neutron network, then you must set this number to at least 100. Controller hosts running the `neutron-dhcp-agent` must set this value to match the number of Neutron networks managed by that agent. This means that this number probably must be higher on controller hosts than compute hosts.

Add Extra Kernel Boot Parameters

You add extra kernel boot parameters to the compute or controller nodes by modifying the `resource_registry` and the `parameter_defaults` files.

Procedure

Step 1 Modify the `resource_registry` file.

Example:

```
resource_registry:
     OS::TripleO::Compute::PreNetworkConfig:
/usr/share/openstack-tripleo-heat-templates/extraconfig/pre_network/host_config_and_reboot.yaml
```

Step 2 Modify the `parameter_defaults` file.

Example:

```
parameter_defaults:
    ComputeParameters:
        KernelArgs: "intel_iommu=on iommu=pt default_hugepagesz=1GB hugepagesz=1G
hugepages=60"
```