Configuring Ironic for OpenStack

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.

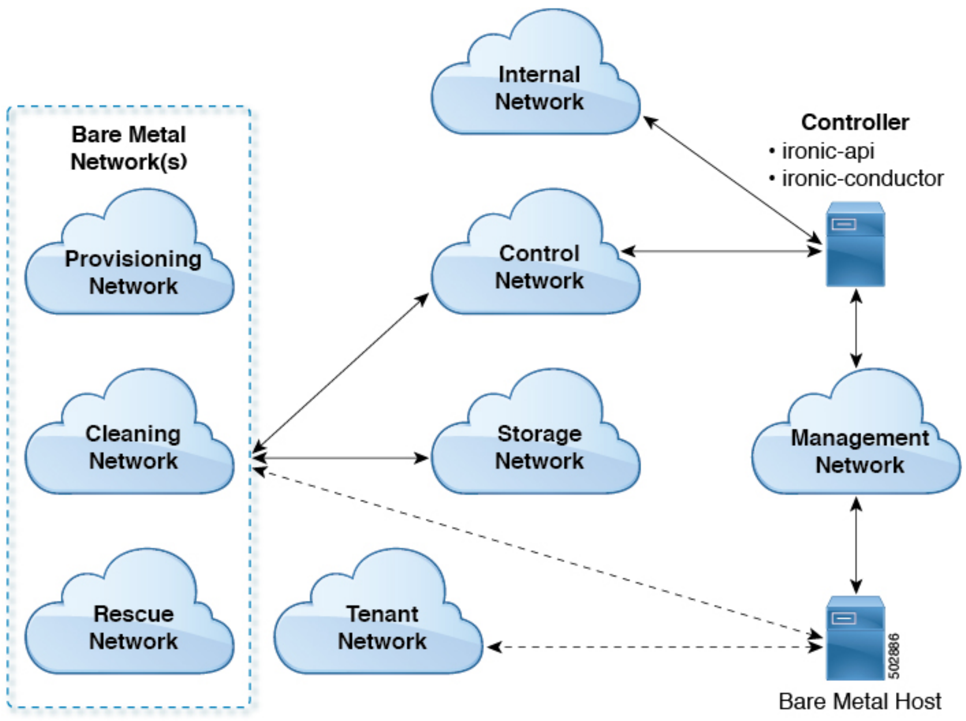
Ironic, which is deployed on the OpenStack overcloud, integrates with OpenStack services, such as Compute (Nova) and Network (Neutron). You can use Ironic on its own or as part of an OpenStack cloud. Using Ironic with OpenStack enables you to provision both virtual machines and bare metal servers on 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 3 in this guide.

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

See the section Predeployment Configuration, on page 4 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 13 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 14 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 15 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 OSP16.1.

- 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 OSP16.1 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 OSP16.1 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

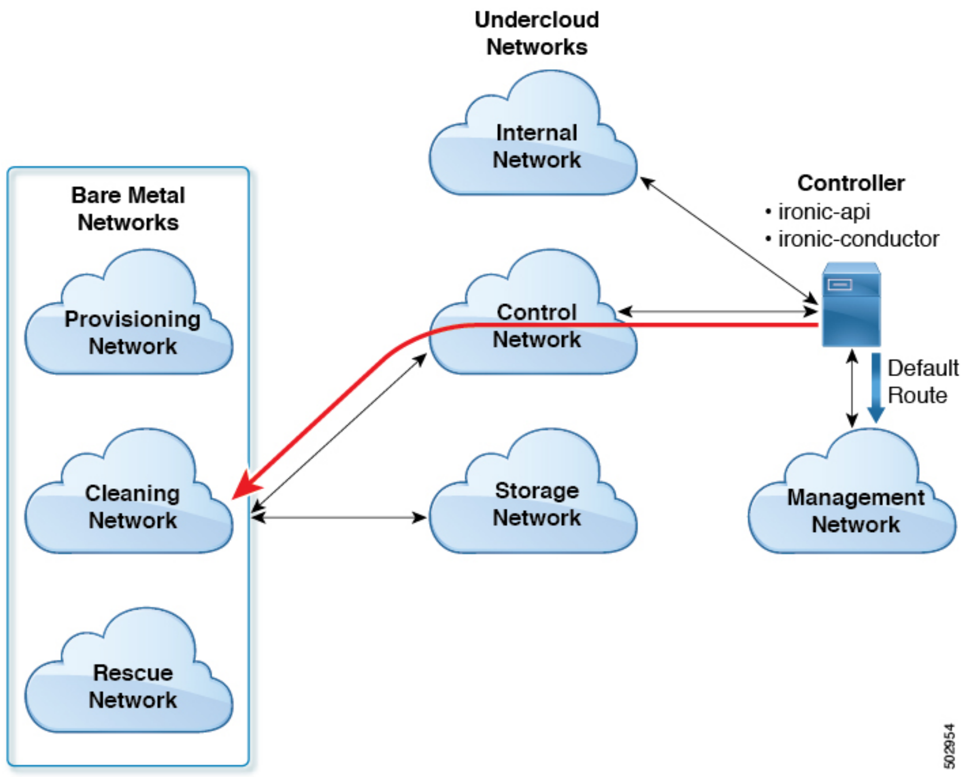
The Port Channel Policy should have the mode configured for “LACP Active.” This configuration is needed because bare metal instances only use a single NIC when Unified Extensible Firmware Interface (UEFI) boots and attaches to bare metal networks. Both ports are used when connecting to tenant networks. 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 3.

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 3.
- Deploy the overcloud.

Procedure

Create and upload images to the overcloud, following instructions in Red Hat OpenStack Platform 16.1 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 OSP16.1 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 OSP16.1 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 OSP16.1 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 Centric Infrastructure (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 ACI 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 3 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 3 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 3 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 3 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 3 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 3.

Procedure

Launch the bare metal instances.

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