Red Hat OpenStack Platform 13

Spine Leaf Networking

Configure routed spine-leaf networks using Red Hat OpenStack Platform director

Last Updated: 2018-06-29

Red Hat OpenStack Platform 13 Spine Leaf Networking

Configure routed spine-leaf networks using Red Hat OpenStack Platform director

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide provides a basic scenario on how to configure a routed spine-leaf network on the overcloud. This includes setting up the undercloud, writing the main configuration files, and creating roles for your nodes.

Table of Contents

CHAPTER 1. INTRODUCTION

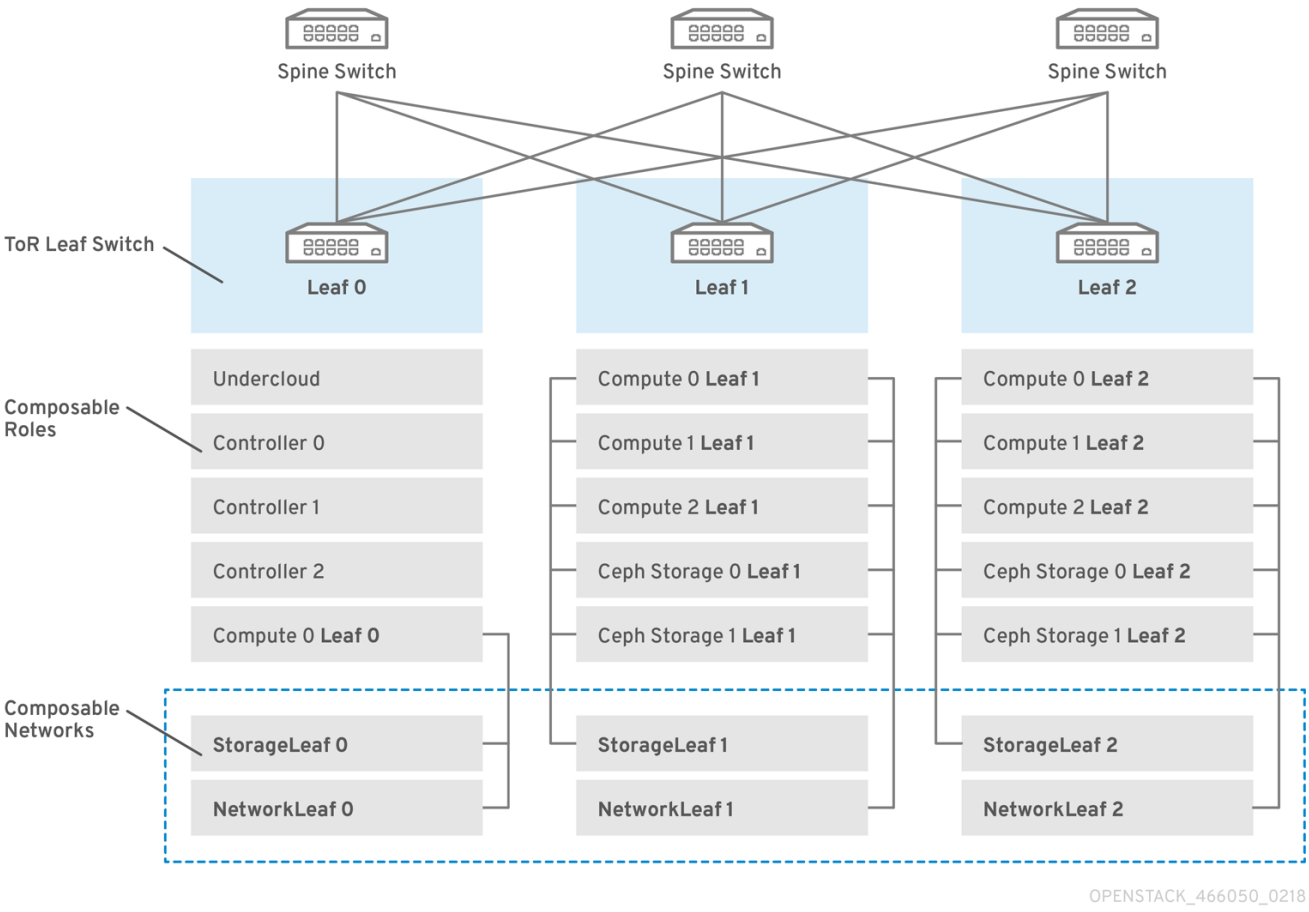
This guide provides information of how to construct a spine-leaf network topology for your Red Hat OpenStack Platform environment. This includes a full end-to-end scenario and example files to help replicate a more extensive network topology within your own environment.

1.1. SPINE-LEAF NETWORKING

Red Hat OpenStack Platform's composable network architecture allows you to adapt your networking to the popular routed spine-leaf data center topology. In a practical application of routed spine-leaf, a leaf is represented as a composable Compute or Storage role usually in a data center rack, as shown in Figure 1.1, "Routed spine-leaf example". The *Leaf 0* rack has an undercloud node, controllers, and compute nodes. The composable networks are presented to the nodes, which have been assigned to composable roles. In this diagram:

- The `StorageLeaf` networks are presented to the Ceph storage and Compute nodes.
- The `NetworkLeaf` represents an example of any network you might want to compose.

Figure 1.1. Routed spine-leaf example



1.2. NETWORK TOPOLOGY

The routed spine-leaf bare metal environment has one or more layer 3 capable switches, which route traffic between the isolated VLANs in the separate layer 2 broadcast domains.

The intention of this design is to isolate the traffic according to function. For example, if the controller

nodes host an API on the *Internal API* network, when a compute node accesses the API it should use its own version of the *Internal API* network. For this routing to work, you need routes that force traffic destined for the *Internal API* network to use the required interface. This can be configured using *supernet* routes. For example, if you use `172.18.0.0/24` as the *Internal API* network for the controller nodes, you can use `172.18.1.0/24` for the second *Internal API* network, and `172.18.2.0/24` for the third, and so on. As a result, you can have a route pointing to the larger `172.18.0.0/16` supernet that uses the gateway IP on the local *Internal API* network for each role in each layer 2 domain.

This scenario uses the following networks:

Table 1.1. Leaf 0 Networks

Network	Roles attached	Interface	Bridge	Subnet
Provisioning / Control Plane	All	nic1	br-ctlplane (undercloud)	192.168.10.0/24
Storage	Controller	nic2		172.16.0.0/24
Storage Mgmt	Controller	nic3		172.17.0.0/24
Internal API	Controller	nic4		172.18.0.0/24
Tenant	Controller	nic5		172.19.0.0/24
External	Controller	nic6	br-ex	10.1.1.0/24

Table 1.2. Leaf 1 Networks

Network	Roles attached	Interface	Bridge	Subnet
Provisioning / Control Plane	All	nic1	br-ctlplane (undercloud)	192.168.11.0/24
Storage1	Compute1, Ceph1	nic2		172.16.1.0/24
Storage Mgmt1	Ceph1	nic3		172.17.1.0/24
Internal API1	Compute1	nic4		172.18.1.0/24
Tenant1	Compute1	nic5		172.19.1.0/24

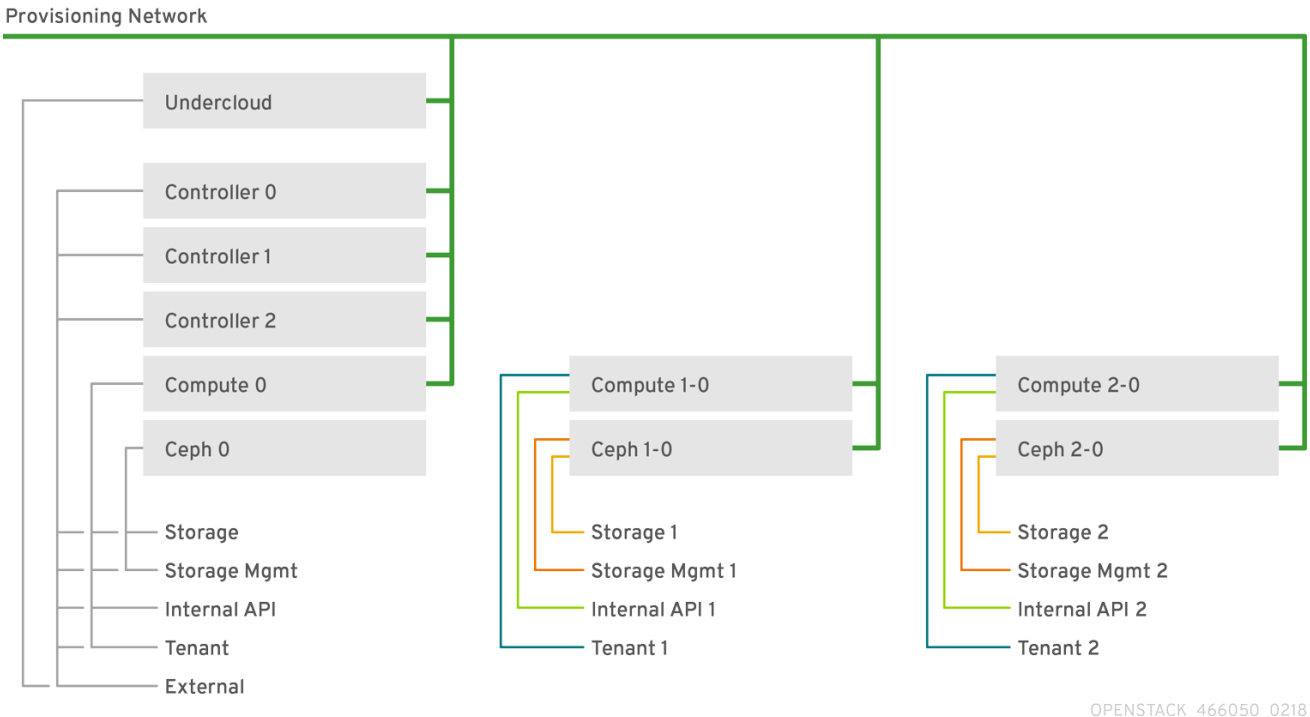
Table 1.3. Leaf 2 Networks

Network	Roles attached	Interface	Bridge	Subnet
Provisioning / Control Plane	All	nic1	br-ctlplane (undercloud)	192.168.12.0/24

Network	Roles attached	Interface	Bridge	Subnet
Storage2	Compute2, Ceph2	nic2		172.16.2.0/24
Storage Mgmt2	Ceph2	nic3		172.17.2.0/24
Internal API2	Compute2	nic4		172.18.2.0/24
Tenant2	Compute2	nic5		172.19.2.0/24

Table 1.4. Supernet Routes

Network	Subnet
Storage	172.16.0.0/16
Storage Mgmt	172.17.0.0/16
Internal API	172.18.0.0/16
Tenant	172.19.0.0/16



1.3. SPINE-LEAF REQUIREMENTS

To deploy the overcloud on a network with a layer-3 routed architecture, you must meet the following requirements:

Layer-3 routing

The network infrastructure must have routing configured to enable traffic between the different layer-2 segments. This can be statically or dynamically configured.

DHCP-Relay

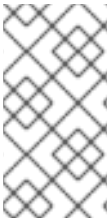
Each layer-2 segment not local to the undercloud must provide `dhcp-relay`. You must forward DHCP requests to the undercloud on the provisioning network segment where the undercloud is connected.

NOTE

The undercloud uses two DHCP servers. One for baremetal node introspection, and another for deploying overcloud nodes. Make sure to read DHCP relay configuration to understand the requirements when configuring `dhcp-relay`.

1.4. SPINE-LEAF LIMITATIONS

- Some roles, such as the Controller role, use virtual IP addresses and clustering. The mechanism behind this functionality requires layer-2 network connectivity between these nodes. These nodes are all be placed within the same leaf.
- Similar restrictions apply to Networker nodes. The network service implements highly-available default paths in the network using Virtual Router Redundancy Protocol (VRRP). Since VRRP uses a virtual router IP address, you must connect master and backup nodes to the same L2 network segment.
- When using tenant or provider networks with VLAN segmentation, you must share the particular VLANs between all Networker and Compute nodes.



NOTE

It is possible to configure the network service with multiple sets of Networker nodes. Each set share routes for their networks, and VRRP would provide highly-available default paths within each set of Networker nodes. In such configuration all Networker nodes sharing networks must be on the same L2 network segment.

CHAPTER 2. CONFIGURING THE UNDERCLOUD

This section describes a use case on how to configure the undercloud to accommodate routed spine-leaf with composable networks.

2.1. CONFIGURING THE SPINE LEAF PROVISIONING NETWORKS

To configure the provisioning networks for your spine leaf infrastructure, edit the **undercloud.conf** file and set the relevant parameters as defined in the following procedure.

Procedure

1. Log into the undercloud as the **stack** user.
2. If you do not already have an **undercloud.conf**, copy the sample template file:

```
[stack@director ~]$ cp /usr/share/instack-
undercloud/undercloud.conf.sample ~/undercloud.conf
```

3. Edit your **undercloud.conf**.
4. In the **[DEFAULT]** section:
   a. Set **enable_routed_networks** to **true**:

```
enable_routed_networks = true
```

   b. Define your list of subnets using the **subnets** parameter. Define one subnet for each layer 2 segment in the routed spine and leaf:

```
subnets = leaf0,leaf1,leaf2
```

   c. Specify the subnet associated with the physical layer 2 segment local to the undercloud using the **local_subnet** parameter:

```
local_subnet = leaf0
```

5. Create a new section per each subnet defined with the **subnets** parameter:

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
```

```
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False
```

6. Save the **undercloud.conf** file.

7. Run the undercloud installation command:

```
[stack@director ~]$ openstack undercloud install
```

This creates three subnets on the provisioning network / control plane. The overcloud uses each network to provision systems within each respective leaf.

To ensure proper relay of DHCP requests to the undercloud, you might need to configure a DHCP relay. The next section provides some information on how to configure a DHCP relay.

2.2. CONFIGURING A DHCP RELAY

The undercloud uses two DHCP servers on the provisioning network:

- one for introspection.
- one for provisioning.

When configuring a DHCP relay make sure to forward DHCP requests to both DHCP servers on the undercloud.

You can use UDP broadcast with devices that support it to relay DHCP requests to the L2 network segment where the undercloud provisioning network is connected. Alternatively you can use UDP unicast which relays DHCP requests to specific IP addresses.

NOTE

Configuration of DHCP relay on specific devices types is beyond the scope of this document. As a reference, this document provides a DHCP relay configuration example using the implementation in ISC DHCP software is available below. Please refer to manual page dhcrelay(8) for further details on how to use this implementation.

Broadcast DHCP relay

This method relays DHCP requests using UDP broadcast traffic onto the L2 network segment where the DHCP server(s) resides. All devices on the network segment receive the broadcast traffic. When using UDP broadcast, both DHCP servers on the undercloud receive the relayed DHCP request. Depending on implementation this is typically configured by specifying either the interface or IP network address:

Interface

Specifying an interface connected to the L2 network segment where the DHCP requests are relayed.

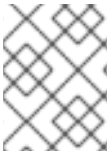
IP network address

Specifying the network address of the IP network where the DHCP request are relayed.

Unicast DHCP relay

This method relays DHCP requests using UDP unicast traffic to specific DHCP servers. When using UDP unicast, you must configure the device providing DHCP relay to relay DHCP requests to both the IP address assigned to the interface used for introspection on the undercloud and the IP address of the network namespace created by the OpenStack Networking (neutron) service to host the DHCP service for the **ctlplane** network.

The interface used for introspection is the one defined as inspection_interface in **undercloud.conf**.



NOTE

It is common to use the **br-ctlplane** interface for introspection. The IP address defined as **local_ip** in **undercloud.conf** is on the **br-ctlplane** interface.

The IP address allocated to the Neutron DHCP namespace is the first address available in the IP range configured for the local_subnet in undercloud.conf. The first address in the IP range is the one defined as **dhcp_start** in the configuration. For example: **172.20.0.10** would be the IP address when the following configuration is used:

```
[DEFAULT]
local_subnet = leaf0
subnets = leaf0,leaf1,leaf2

[leaf0]
cidr = 172.20.0.0/26
dhcp_start = 172.20.0.10
dhcp_end = 172.20.0.19
inspection_iprange = 172.20.0.20,172.20.0.29
gateway = 172.20.0.62
masquerade = False
```

WARNING

The IP address for the DHCP namespace is automatically allocated. In most cases, it will be the first address in the IP range. Ensure sure to verify this is the case by running the following commands on the undercloud:

```
$ openstack port list --device-owner network:dhcp -c "Fixed
IP Addresses"
+-----------------------------------------------------------
--------------------+
| Fixed IP Addresses
|
+-----------------------------------------------------------
--------------------+
| ip_address='172.20.0.10', subnet_id='7526fbe3-f52a-4b39-
a828-ec59f4ed12b2' |
+-----------------------------------------------------------
--------------------+
$ openstack subnet show 7526fbe3-f52a-4b39-a828-
ec59f4ed12b2 -c name
+-------+--------+
| Field | Value  |
+-------+--------+
| name  | leaf0  |
+-------+--------+
```

Example DHCP relay configuration

In the following example, the **dhcrelay** command from ISC DHCP software uses the following configuration:

- Interfaces to relay incoming DHCP request: **eth1**, **eth2**, and **eth3**.
- Interface the undercloud DHCP servers on the network segment are connected to: **eth0**.
- The DHCP server used for introspection is listening on IP address: **172.20.0.1**.
- The DHCP server used for provisioning is listening on IP address **172.20.0.10**.

This results in the following **dhcrelay** command:

```
$ sudo dhcrelay -d --no-pid 172.20.0.10 172.20.0.1 \
  -iu eth0 -id eth1 -id eth2 -id eth3
```

Now that you have configured the provisioning network, you can configure the remaining overcloud leaf networks. You accomplish this with a series of configuration files.

2.3. CREATING FLAVORS AND TAGGING NODES FOR LEAF NETWORKS

Each role in each leaf network requires a flavor and role assignment so you can tag nodes into their respective leaf. This procedure shows how to create each flavor and assign them to a role.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Create flavors for each custom role:

```
$ ROLES="control0 compute0 compute1 compute2 ceph-storage0 ceph-
storage1 ceph-storage2"
$ for ROLE in $ROLES; do openstack flavor create --id auto --ram
4096 --disk 40 --vcpus 1 $ROLE ; done
$ for ROLE in $ROLES; do openstack flavor set --property
"cpu_arch"="x86_64" --property "capabilities:boot_option"="local" --
property "capabilities:profile"="$ROLE" $ROLE ; done
```

3. Tag nodes to their respective leaf networks. For example, run the following to tag a node with UUID **58c3d07e-24f2-48a7-bbb6-6843f0e8ee13** to the compute role on Leaf2:

```
$ openstack baremetal node set --property
capabilities='profile:compute2,boot_option:local' 58c3d07e-24f2-
48a7-bbb6-6843f0e8ee13
```

4. Create an environment file (**~/templates/node-data.yaml**) that contains the mapping of flavors to roles:

```
parameter_defauls:
  OvercloudController0Flavor: control0
  OvercloudController0Count: 3
  OvercloudCompute0Flavor: compute0
  OvercloudCompute0Count: 3
  OvercloudCompute1Flavor: compute1
  OvercloudCompute1Count: 3
  OvercloudCompute2Flavor: compute2
  OvercloudCompute2Count: 3
  OvercloudCephStorage0Flavor: ceph-storage0
  OvercloudCephStorage0Count: 3
  OvercloudCephStorage1Flavor: ceph-storage1
  OvercloudCephStorage1Count: 3
  OvercloudCephStorage2Flavor: ceph-storage2
  OvercloudCephStorage2Count: 3
```

You can also set the number of nodes to deploy in the overcloud using each respectve *Count` parameter.

2.4. MAPPING BARE METAL NODE PORTS TO CONTROL PLANE NETWORK SEGMENTS

To enable deployment onto a L3 routed network the bare metal ports must have its **physical_network** field configured. Each baremetal port is associated with a bare metal node in the

OpenStack Bare Metal (ironic) service. The physical network names are the ones used in the **subnets** option in the undercloud configuration.

NOTE

The physical network name of the subnet specified as **local_subnet** in **undercloud.conf** is special. It is always named **ctlplane**.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Check the bare metal nodes:

```
$ openstack baremetal node list
```

3. Ensure the bare metal nodes are either in **enroll** or **manageable** state. If the bare metal node is not in one of these states, the command used to set the physical_network property on the baremetal port will fail. To set all nodes to **manageable** state run the following command:

```
$ for node in $(openstack baremetal node list -f value -c Name); do
openstack baremetal node manage $node --wait; done
```

4. Check which baremetal ports are associated with which baremetal node. For example:

```
$ openstack baremetal port list --node <node-uuid>
```

5. Set the **physical-network** parameter for the ports. In the example below, three subnets are defined in the configuration: **leaf0**, **leaf1**, and **leaf2**. The local_subnet is **leaf0**. Since the physical network for the **local_subnet** is always **ctlplane**, the baremetal port connected to **leaf0** uses ctlplane. The remaining ports use the other leaf names:

```
$ openstack baremetal port set --physical-network ctlplane <port-
uuid>
$ openstack baremetal port set --physical-network leaf1 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
```

6. Make sure the nodes are in available state before deploying the overcloud:

```
$ openstack overcloud node provide --all-manageable
```

CHAPTER 3. CONFIGURING THE OVERCLOUD

Now that you have configured the undercloud, you can configure the remaining overcloud leaf networks. You accomplish this with a series of configuration files. Afterwards, you deploy the overcloud and the resulting deployment has multiple sets of networks with routing available.

3.1. CREATING A NETWORK DATA FILE

To define the leaf networks, you create a network data file, which contain a YAML formatted list of each composable network and its attributes. The default network data is located on the undercloud at **`/usr/share/openstack-tripleo-heat-templates/network_data.yaml`**.

Procedure

1. Create a new **`network_data_spine_leaf.yaml`** file in your **`stack`** user's local directory.
2. In the **`network_data_spine_leaf.yaml`** file, create a YAML list that define each network and leaf network as an composable network item. For example, the Internal API network and its leaf networks are defined using the following syntax:

```
# Internal API
- name: InternalApi0
  name_lower: internal_api0
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: true
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: true
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]
```

NOTE

You do not define the Control Plane networks in the network data file since the undercloud has already created these networks. However, you need to manually set the parameters so that the overcloud can configure its NICs accordingly.

See Appendix A, *Example network_data file* for a full example with all composable networks.

Each role has its own NIC configuration. Before configuring the spine-leaf configuration, you need to create a base set of NIC templates to suit your current NIC configuration.

3.2. CREATING A CUSTOM NIC CONFIGURATION

Each role requires its own NIC configuration. Create a copy of the base set of NIC templates and modify them to suit your current NIC configuration.

Procedure

1. Create a new directory to store your NIC templates. For example:

```
$ mkdir ~/templates/spine-leaf-nics/
$ cd ~/templates/spine-leaf-nics/
```

2. Create a base template called **base.yaml**. Use the boilerplate content from Appendix B, *Custom NIC template*. We use this template as a basis for copying our templates for individual roles.

Resources

- See "Creating Custom Interface Templates" in the *Advanced Overcloud Customization* guide for more information on customizing your NIC templates.

3.3. CREATING A CUSTOM CONTROLLER NIC CONFIGURATION

This procedure creates a YAML structure for Controller nodes on Leaf0 only.

Procedure

1. Change to your custom NIC directory:

```
$ cd ~/templates/spine-leaf-nics/
```

2. Copy the base template (**base.yaml**) for Leaf0. For example:

```
$ cp base.yaml controller0.yaml
```

3. Edit the template for **controller0.yaml** and scroll to the network configuration section, which looks like the following:

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
```

4. In the **network_config** section, define the control plane / provisioning interface. For example

```
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
```

```
    dns_servers:
      get_param: DnsServers
    addresses:
    - ip_netmask:
        list_join:
        - /
        - - get_param: ControlPlaneIp
          - get_param: ControlPlane0SubnetCidr
    routes:
    - ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: Leaf0EC2MetadataIp
    - ip_netmask: 192.168.10.0/24
      next_hop:
        get_param: ControlPlane0DefaultRoute
```

Note that the parameters used in this case are specific to Leaf0: **`ControlPlane0SubnetCidr`**, **`Leaf0EC2MetadataIp`**, and **`ControlPlane0DefaultRoute`**. Also note the use of the CIDR for Leaf0 on the provisioning network (192.168.10.0/24), which is used as a route.

5. Define a new interface for our external bridge:

```
  - type: ovs_bridge
    name: br-ex
    use_dhcp: false
    addresses:
    - ip_netmask:
        get_param: ExternalIpSubnet
    routes:
    - default: true
      next_hop:
        get_param: ExternalInterfaceDefaultRoute
    members:
    - type: interface
      name: nic2
      primary: true
```

The **`members`** section will also contain the VLAN configuration for our networks.

6. Add each VLAN to the **`members`** section. For example, to add the Storage network:

```
    members:
    - type: interface
      name: nic2
      primary: true
    - type: vlan
      vlan_id:
        get_param: Storage0NetworkVlanID
      addresses:
      - ip_netmask:
          get_param: Storage0IpSubnet
      routes:
      - ip_netmask:
```

```
                      get_param: StorageSupernet
                    next_hop:
                      get_param: Storage0InterfaceDefaultRoute
```

Note that each interface structure uses parameters specific to Leaf0 (**`Storage0NetworkVlanID`**, **`Storage0IpSubnet`**, **`Storage0InterfaceDefaultRoute`**) as well as the supernet route (**`StorageSupernet`**).

Add a VLAN structure for the following Controller networks: **`Storage`**, **`StorageMgmt`**, **`InternalApi`**, and **`Tenant`**.

7. Save this file.

3.4. CREATING CUSTOM COMPUTE NIC CONFIGURATIONS

This procedure creates a YAML structure for Compute nodes on Leaf0, Leaf1, and Leaf2.

Procedure

1. Change to your custom NIC directory:

```
$ cd ~/templates/spine-leaf-nics/
```

2. Copy the base template (**`base.yaml`**) for Leaf0. For example:

```
$ cp base.yaml compute0.yaml
```

3. Edit the template for **`compute0.yaml`** and scroll to the network configuration section, which looks like the following:

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
```

4. In the **`network_config`** section, define the control plane / provisioning interface. For example

```
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
                dns_servers:
                  get_param: DnsServers
                addresses:
```

```
      - ip_netmask:
          list_join:
          - /
          - - get_param: ControlPlaneIp
            - get_param: ControlPlane0SubnetCidr
      routes:
      - ip_netmask: 169.254.169.254/32
        next_hop:
          get_param: Leaf0EC2MetadataIp
      - ip_netmask: 192.168.10.0/24
        next_hop:
          get_param: ControlPlane0DefaultRoute
```

Note that the parameters used in this case are specific to Leaf0: **`ControlPlane0SubnetCidr`**, **`Leaf0EC2MetadataIp`**, and **`ControlPlane0DefaultRoute`**. Also note the use of the CIDR for Leaf0 on the provisioning network (192.168.10.0/24), which is used as a route.

5. Define a new interface for our external bridge:

```
    - type: ovs_bridge
      name: br-ex
      use_dhcp: false
      members:
      - type: interface
        name: nic2
        primary: true
```

The **`members`** section will also contain the VLAN configuration for our networks.

6. Add each VLAN to the **`members`** section. For example, to add the Storage network:

```
      members:
      - type: interface
        name: nic2
        primary: true
      - type: vlan
        vlan_id:
          get_param: Storage0NetworkVlanID
        addresses:
        - ip_netmask:
            get_param: Storage0IpSubnet
        routes:
        - ip_netmask:
            get_param: StorageSupernet
          next_hop:
            get_param: Storage0InterfaceDefaultRoute
```

Note that each interface structure uses parameters specific to Leaf0 (**`Storage0NetworkVlanID`**, **`Storage0IpSubnet`**, **`Storage0InterfaceDefaultRoute`**) as well as the supernet route (**`StorageSupernet`**).

Add a VLAN structure for the following Controller networks: **`Storage`**, **`InternalApi`**, and **`Tenant`**.

7. Save this file.

8. Copy this file for use with Leaf1 and Leaf2.

```
$ cp compute0.yaml compute1.yaml
$ cp compute0.yaml compute2.yaml
```

9. Edit **compute1.yaml** and scroll to the **network_config** section. Replace the Leaf0 parameters with the Leaf1 parameters. This includes parameters for the following networks: **Control Plane**, **Storage**, **InternalApi**, and **Tenant**. Save this file when complete.

10. Edit **compute2.yaml** and scroll to the **network_config** section. Replace the Leaf0 parameters with the Leaf2 parameters. This includes parameters for the following networks: **Control Plane**, **Storage**, **InternalApi**, and **Tenant**. Save this file when complete.

3.5. CREATING CUSTOM CEPH STORAGE NIC CONFIGURATIONS

This procedure creates a YAML structure for Ceph Storage nodes on Leaf0, Leaf1, and Leaf2.

Procedure

1. Change to your custom NIC directory:

```
$ cd ~/templates/spine-leaf-nics/
```

2. Copy the base template (**base.yaml**) for Leaf0. For example:

```
$ cp base.yaml compute0.yaml
```

3. Edit the template for **ceph-storage0.yaml** and scroll to the network configuration section, which looks like the following:

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
```

4. In the **network_config** section, define the control plane / provisioning interface. For example

```
              network_config:
              - type: interface
                name: nic1
                use_dhcp: false
```

```
    dns_servers:
      get_param: DnsServers
    addresses:
    - ip_netmask:
        list_join:
        - /
        - - get_param: ControlPlaneIp
          - get_param: ControlPlane0SubnetCidr
    routes:
    - ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: Leaf0EC2MetadataIp
    - ip_netmask: 192.168.10.0/24
      next_hop:
        get_param: ControlPlane0DefaultRoute
```

Note that the parameters used in this case are specific to Leaf0: **`ControlPlane0SubnetCidr`**, **`Leaf0EC2MetadataIp`**, and **`ControlPlane0DefaultRoute`**. Also note the use of the CIDR for Leaf0 on the provisioning network (192.168.10.0/24), which is used as a route.

5. Define a new interface for our external bridge:

```
  - type: ovs_bridge
    name: br-ex
    use_dhcp: false
    members:
    - type: interface
      name: nic2
      primary: true
```

The **`members`** section will also contain the VLAN configuration for our networks.

6. Add each VLAN to the **`members`** section. For example, to add the Storage network:

```
    members:
    - type: interface
      name: nic2
      primary: true
    - type: vlan
      vlan_id:
        get_param: Storage0NetworkVlanID
      addresses:
      - ip_netmask:
          get_param: Storage0IpSubnet
      routes:
      - ip_netmask:
          get_param: StorageSupernet
        next_hop:
          get_param: Storage0InterfaceDefaultRoute
```

Note that each interface structure uses parameters specific to Leaf0 (**`Storage0NetworkVlanID`**, **`Storage0IpSubnet`**, **`Storage0InterfaceDefaultRoute`**) as well as the supernet route (**`StorageSupernet`**).

Add a VLAN structure for the following Controller networks: **Storage**, **StorageMgmt**.

7. Save this file.
8. Copy this file for use with Leaf1 and Leaf2.

```
$ cp ceph-storage0.yaml ceph-storage1.yaml
$ cp ceph-storage0.yaml ceph-storage2.yaml
```

9. Edit **ceph-storage1.yaml** and scroll to the **network_config** section. Replace the Leaf0 parameters with the Leaf1 parameters. This includes parameters for the following networks: **Control Plane**, **Storage**, **InternalApi**, and **Tenant**. Save this file when complete.
10. Edit **ceph-storage2.yaml** and scroll to the **network_config** section. Replace the Leaf0 parameters with the Leaf2 parameters. This includes parameters for the following networks: **Control Plane**, **Storage**, **InternalApi**, and **Tenant**. Save this file when complete.

3.6. CREATING A NETWORK ENVIRONMENT FILE

This procedure creates a basic network environment file for use later.

Procedure

1. Create a **network-environment.yaml** file in your stack user's **templates** directory.
2. Add the following sections to the environment file:

```
resource_registry:

parameter_defaults:
```

Note the following:

- The **resource_registry** will map networking resources to their respective NIC templates.
- The **parameter_defaults** will define additional networking parameters relevant to your configuration.

The next couple of sections add details to your network environment file to configure certain aspects of the spine leaf architecture. Once complete, you include this file with your **openstack overcloud deploy** command.

3.7. MAPPING NETWORK RESOURCES TO NIC TEMPLATES

This procedure maps the relevant resources for network configurations to their respective NIC templates.

Procedure

1. Edit your **network-environment.yaml** file.
2. Add the resource mappings to your **resource_registry**. The resource names take the following format:

```
OS::TripleO::[ROLE]::Net::SoftwareConfig: [NIC TEMPLATE]
```

For this guide's scenario, the **resource_registry** includes the following resource mappings:

```
resource_registry:
  OS::TripleO::Controller0::Net::SoftwareConfig: ./spine-leaf-
nics/controller0.yaml
  OS::TripleO::Compute0::Net::SoftwareConfig: ./spine-leaf-
nics/compute0.yaml
  OS::TripleO::Compute1::Net::SoftwareConfig: ./spine-leaf-
nics/compute1.yaml
  OS::TripleO::Compute2::Net::SoftwareConfig: ./spine-leaf-
nics/compute2.yaml
  OS::TripleO::CephStorage0::Net::SoftwareConfig: ./spine-leaf-
nics/CephStorage0.yaml
  OS::TripleO::CephStorage1::Net::SoftwareConfig: ./spine-leaf-
nics/CephStorage1.yaml
  OS::TripleO::CephStorage2::Net::SoftwareConfig: ./spine-leaf-
nics/CephStorage2.yaml
```

3. Save the **network-environment.yaml** file.

3.8. SPINE LEAF ROUTING

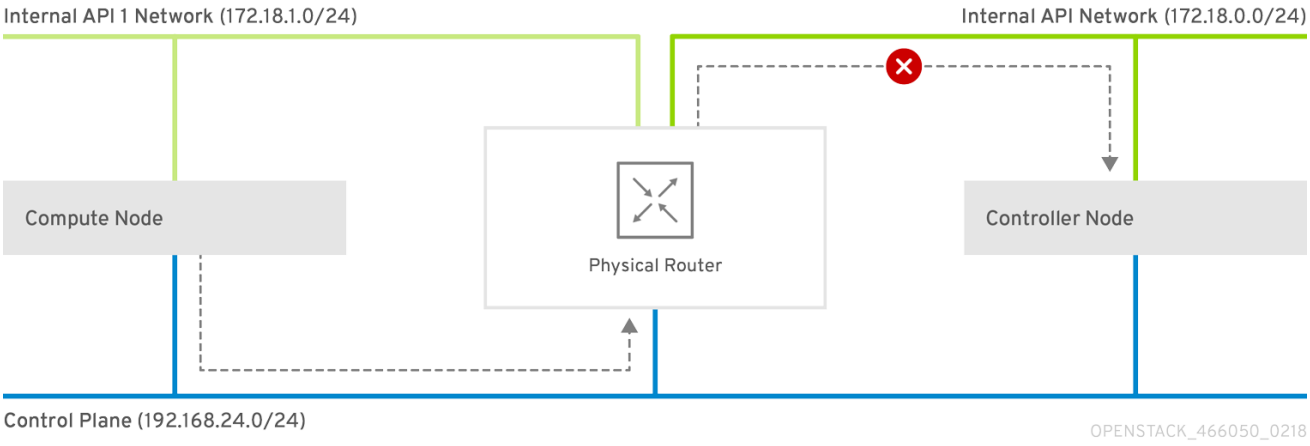
Each role requires routes on each isolated network, pointing to the other subnets used for the same function. So when a *Compute1* node contacts a controller on the **InternalApi** VIP, the traffic should target the **InternalApi1** interface through the **InternalApi1** gateway. As a result, the return traffic from the controller to the **InternalApi1** network should go through the **InternalApi** network gateway.

The supernet routes apply to all isolated networks on each role to avoid sending traffic through the default gateway, which by default is the *Control Plane* network on non-controllers, and the *External* network on the controllers.

You need to configure these routes on the isolated networks because Red Hat Enterprise Linux by default implements strict reverse path filtering on inbound traffic. If an API is listening on the *Internal API* interface and a request comes in to that API, it only accepts the request if the return path route is on the *Internal API* interface. If the server is listening on the *Internal API* network but the return path to the client is through the *Control Plane*, then the server drops the requests due to the reverse path filter.

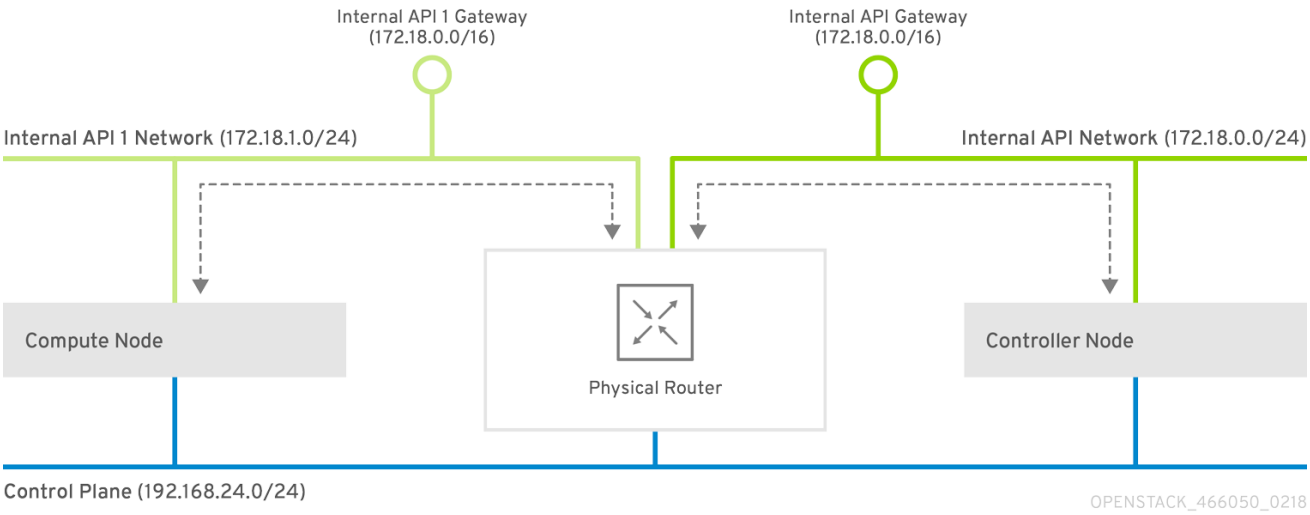
This following diagram shows an attempt to route traffic through the control plane, which will not succeed. The return route from the router to the controller node does not match the interface where the VIP is listening, so the packet is dropped. **192.168.24.0/24** is directly connected to the controller, so it is considered local to the *Control Plane* network.

Figure 3.1. Routed traffic through Control Plane



For comparison, this diagram shows routing running through the *Internal API* networks:

Figure 3.2. Routed traffic through Internal API



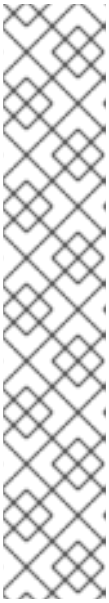
3.9. ASSIGNING ROUTES FOR COMPOSABLE NETWORKS

This procedure defines the routing for the leaf networks.

Procedure

1. Edit your `network-environment.yaml` file.
2. Add the supernet route parameters to the `parameter_defaults` section. Each isolated network should have a supernet route applied. For example:

```
parameter_defaults:
  StorageSupernet: 172.16.0.0/16
  StorageMgmtSupernet: 172.17.0.0/16
  InternalApiSupernet: 172.18.0.0/16
  TenantSupernet: 172.19.0.0/16
```



NOTE

The network interface templates should contain the supernet parameters for each network. For example:

```
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
      get_param: Storage0IpSubnet
  routes:
  - ip_netmask:
      get_param: StorageSupernet
    next_hop:
      get_param: Storage0InterfaceDefaultRoute
```

3. Add the following **ExtraConfig** settings to the **parameter_defaults** section to address routing for specific components on Compute and Ceph Storage nodes.

```
parameter_defaults:
  ...
  Compute1ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api1')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant1')}"
  Compute2ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api2')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant2')}"
  Compute3ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api3')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant3')}"
  CephAnsibleExtraConfig:
    public_network: '172.16.0.0/24,172.16.1.0/24,172.16.2.0/24'
    cluster_network: '172.17.0.0/24,172.17.1.0/24,172.17.2.0/24'
```

- For the Compute **ExtraConfig** parameters:
  - Defines the IP address to use for the VNC proxy.
  - Defines the IP address to use for the ML2 agent.
- For **CephAnsibleExtraConfig**:
  - The **public_network** setting lists all the storage networks (one per leaf).
  - The **cluster_network** entries lists the storage management networks (one per leaf).

3.10. SETTING CONTROL PLANE PARAMETERS

You usually define networking details for isolated spine-leaf networks using a **network_data** file. The exception is the control plane network, which the undercloud created. However, the overcloud requires access to the control plane for each leaf. This requires some additional parameters, which you define in your **network-environment.yaml** file. For example, the following snippet is from an example NIC template for the Controller role on Leaf0

```
- type: interface
  name: nic1
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
  - ip_netmask:
      list_join:
      - /
      - - get_param: ControlPlaneIp
        - get_param: ControlPlane0SubnetCidr
  routes:
  - ip_netmask: 169.254.169.254/32
    next_hop:
      get_param: Leaf0EC2MetadataIp
  - ip_netmask: 192.168.10.0/24
    next_hop:
      get_param: ControlPlane0DefaultRoute
```

In this instance, we need to define the IP, subnet, metadata IP, and default route for the respective Control Plane network on Leaf 0.

Procedure

1. Edit your **network-environment.yaml** file.
2. In the **parameter_defaults** section:
   a. Add the mapping to the main control plane subnet:

```
parameter_defaults:
  ...
  ControlPlaneSubnet: leaf0
```

   b. Add the control plane subnet mapping for each spine-leaf network:

```
parameter_defaults:
  ...
  Controller0ControlPlaneSubnet: leaf0
  Compute0ControlPlaneSubnet: leaf0
  Compute1ControlPlaneSubnet: leaf1
  Compute2ControlPlaneSubnet: leaf2
  CephStorage0ControlPlaneSubnet: leaf0
  CephStorage1ControlPlaneSubnet: leaf1
  CephStorage2ControlPlaneSubnet: leaf2
```

   c. Add the control plane routes for each leaf:

```
parameter_defaults:
  ...
  ControlPlane0DefaultRoute: 192.168.10.1
  ControlPlane0SubnetCidr: '24'
  ControlPlane1DefaultRoute: 192.168.11.1
```

```
  ControlPlane1SubnetCidr: '24'
  ControlPlane2DefaultRoute: 192.168.12.1
  ControlPlane2SubnetCidr: '24'
```

The default route parameters are typically the IP address set for the **gateway** of each provisioning subnet. Refer to your **undercloud.conf** file for this information.

d. Add the parameters for the EC2 metadata IPs:

```
parameter_defaults:
  ...
  Leaf0EC2MetadataIp: 192.168.10.1
  Leaf1EC2MetadataIp: 192.168.11.1
  Leaf2EC2MetadataIp: 192.168.12.1
```

These act as routes through the control plane for the EC2 metadata service (169.254.169.254/32) and you should typically set these to the respective **gateway** for each leaf on the provisioning network.

3. Save the **network-environment.yaml** file.

3.11. CREATING A ROLES DATA FILE

This section demonstrates how to define each composable role for each leaf and attach the composable networks to each respective role.

Procedure

1. Create a custom **roles** director in your **stack** user's local directory:

```
$ mkdir ~/roles
```

2. Copy the default Controller, Compute, and Ceph Storage roles from the director's core template collection to the roles director. Rename the files for Leaf 0:

```
$ cp /usr/share/openstack-tripleo-heat-
templates/roles/Controller.yaml ~/roles/Controller0.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Compute.yaml
~/roles/Compute0.yaml
$ cp /usr/share/openstack-tripleo-heat-
templates/roles/CephStorage.yaml ~/roles/CephStorage0.yaml
```

3. Edit the **Controller0.yaml** file:

```
$ vi ~/roles/Controller0.yaml
```

4. Edit the **name**, **networks**, and **HostnameFormatDefault** parameters in this file so that they align with the Leaf 0 specific parameters. For example:

```
- name: Controller0
  ...
  networks:
    - External
```

```
    - InternalApi0
    - Storage0
    - StorageMgmt0
    - Tenant0
  ...
  HostnameFormatDefault: '%stackname%-controller0-%index%'
```

Save this file.

5. Edit the **Compute0.yaml** file:

```
$ vi ~/roles/Compute0.yaml
```

6. Edit the **name**, **networks**, and **HostnameFormatDefault** parameters in this file so that they align with the Leaf 0 specific parameters. For example:

```
- name: Compute0
  ...
  networks:
    - InternalApi0
    - Tenant0
    - Storage0
  HostnameFormatDefault: '%stackname%-compute0-%index%'
```

Save this file.

7. Edit the **CephStorage0.yaml** file:

```
$ vi ~/roles/CephStorage0.yaml
```

8. Edit the **name`and `networks** parameters in this file so that they align with the Leaf 0 specific parameters. In addition, add the **HostnameFormatDefault** parameter and define the Leaf 0 hostname for our Ceph Storage nodes. For example:

```
- name: CephStorage0
  ...
  networks:
    - Storage0
    - StorageMgmt0
  HostnameFormatDefault: '%stackname%-cephstorage0-%index%'
```

Save this file.

9. Copy the Leaf 0 Compute and Ceph Storage files as a basis for your Leaf 1 and Leaf 2 files:

```
$ cp ~/roles/Compute0.yaml ~/roles/Compute1.yaml
$ cp ~/roles/Compute0.yaml ~/roles/Compute2.yaml
$ cp ~/roles/CephStorage0.yaml ~/roles/CephStorage1.yaml
$ cp ~/roles/CephStorage0.yaml ~/roles/CephStorage2.yaml
```

10. Edit the **name**, **networks**, and **HostnameFormatDefault** parameters in the Leaf 1 and Leaf 2 files so that they align with the respective Leaf network parameters. For example, the parameters in the Leaf 1 Compute file have the following values:

```
- name: Compute1
  ...
  networks:
    - InternalApi1
    - Tenant1
    - Storage1
  HostnameFormatDefault: '%stackname%-compute1-%index%'
```

The Leaf 1 Ceph Storage parameters have the following values:

```
- name: CephStorage1
  ...
  networks:
    - Storage1
    - StorageMgmt1
  HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
```

11. When you roles are ready, generate the full roles data file using the following command:

```
$ openstack overcloud roles generate --roles-path ~/roles -o
roles_data_spine_leaf.yaml Controller0 Compute0 Compute1 Compute2
CephStorage0 CephStorage1 CephStorage2
```

This creates a full **roles_data_spine_leaf.yaml** file that includes all the custom roles for each respective leaf network.

See Appendix C, *Example roles_data file* for a full example of this file.

3.12. DEPLOYING A SPINE-LEAF ENABLED OVERCLOUD

All our files are now ready for our deployment. This section provides a review of each file and the deployment command:

Procedure

1. Review the **/home/stack/template/network_data_spine_leaf.yaml** file and ensure it contains each network for each leaf.

NOTE

There is currently no validation performed for the network subnet and **allocation_pools** values. Be certain you have defined these consistently and there is no conflict with existing networks.

2. Review the NIC templates contained in **~/templates/spine-leaf-nics/** and ensure the interfaces for each role on each leaf are correctly defined.

3. Review the **network-environment.yaml** environment file and ensure it contains all custom parameters that fall outside control of the network data file. This includes routes, control plane parameters, and a **resource_registry** section that references the custom NIC templates for each role.

4. Review the **/home/stack/templates/roles_data_spine_leaf.yaml** values and ensure you have defined a role for each leaf.

5. Check the `/home/stack/templates/nodes_data.yaml file and ensure all roles have an assigned flavor and a node count. Check also that all nodes for each leaf are correctly tagged.

6. Run the **openstack overcloud deploy** command to apply the spine leaf configuration. For example:

```
openstack overcloud deploy --templates \
-n /home/stack/template/network_data_spine_leaf.yaml \
-r /home/stack/templates/roles_data_spine_leaf.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
-e /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/nodes_data.yaml \
-e [OTHER ENVIRONMENT FILES]
```

   Add any additional environment files. For example, an environment file with your container image locations or Ceph cluster configuration.

7. Wait until the spine-leaf enabled overcloud deploys.

APPENDIX A. EXAMPLE NETWORK_DATA FILE

```
# Storage
- name: Storage0
  vip: true
  name_lower: storage0
  ip_subnet: '172.16.0.0/24'
  allocation_pools: [{'start': '172.16.0.4', 'end': '172.16.0.250'}]
- name: Storage1
  vip: true
  name_lower: storage1
  ip_subnet: '172.16.1.0/24'
  allocation_pools: [{'start': '172.16.1.4', 'end': '172.16.1.250'}]
- name: Storage2
  vip: true
  name_lower: storage2
  ip_subnet: '172.16.2.0/24'
  allocation_pools: [{'start': '172.16.2.4', 'end': '172.16.2.250'}]

# StorageMgmt
- name: StorageMgmt0
  name_lower: storage_mgmt0
  vip: true
  ip_subnet: '172.17.0.0/24'
  allocation_pools: [{'start': '172.17.0.0', 'end': '172.17.0.250'}]
- name: StorageMgmt1
  name_lower: storage_mgmt1
  vip: true
  ip_subnet: '172.17.1.0/24'
  allocation_pools: [{'start': '172.17.1.4', 'end': '172.17.1.250'}]
- name: StorageMgmt2
  name_lower: storage_mgmt2
  vip: true
  ip_subnet: '172.17.2.0/24'
  allocation_pools: [{'start': '172.17.2.4', 'end': '172.17.2.250'}]

# Internal API
- name: InternalApi0
  name_lower: internal_api0
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: true
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: true
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]

# Tenant
- name: Tenant0
  vip: false  # Tenant network does not use VIPs
```

```
  name_lower: tenant0
  ip_subnet: '172.19.0.0/24'
  allocation_pools: [{'start': '172.19.0.4', 'end': '172.19.0.250'}]
- name: Tenant1
  vip: false  # Tenant network does not use VIPs
  name_lower: tenant1
  ip_subnet: '172.19.1.0/24'
  allocation_pools: [{'start': '172.19.1.4', 'end': '172.19.1.250'}]
- name: Tenant2
  vip: false  # Tenant network does not use VIPs
  name_lower: tenant2
  ip_subnet: '172.19.2.0/24'
  allocation_pools: [{'start': '172.19.2.4', 'end': '172.19.2.250'}]

- name: External
  vip: true
  name_lower: external
  ip_subnet: '10.0.0.0/24'
  allocation_pools: [{'start': '10.0.0.4', 'end': '10.0.0.250'}]
  gateway_ip: '10.0.0.1'
```

APPENDIX B. CUSTOM NIC TEMPLATE

The following is a template to get you started with the configuring the network interface templates for spine leaf networking. Note that the **resources** section is incomplete and requires your interface definitions.

```
heat_template_version: queens

parameters:
  # Supernets
  StorageSupernet:
    type: string
  StorageMgmtSupernet:
    type: string
  InternalApiSupernet:
    type: string
  TenantSupernet:
    type: string
  ExternalSupernet:
    type: string

  # Default Routes
  ControlPlane0DefaultRoute:
    type: string
  ControlPlane1DefaultRoute:
    type: string
  ControlPlane2DefaultRoute:
    type: string
  Storage0InterfaceDefaultRoute:
    type: string
  Storage1InterfaceDefaultRoute:
    type: string
  Storage2InterfaceDefaultRoute:
    type: string
  StorageMgmt0InterfaceDefaultRoute:
    type: string
  StorageMgmt1InterfaceDefaultRoute:
    type: string
  StorageMgmt2InterfaceDefaultRoute:
    type: string
  InternalApi0InterfaceDefaultRoute:
    type: string
  InternalApi1InterfaceDefaultRoute:
    type: string
  InternalApi2InterfaceDefaultRoute:
    type: string
  Tenant0InterfaceDefaultRoute:
    type: string
  Tenant1InterfaceDefaultRoute:
    type: string
  Tenant2InterfaceDefaultRoute:
    type: string
  ExternalInterfaceDefaultRoute:
    type: string

  # IP subnets
```

```
  Storage0IpSubnet:
    type: string
  Storage1IpSubnet:
    type: string
  Storage2IpSubnet:
    type: string
  StorageMgmt0IpSubnet:
    type: string
  StorageMgmt1IpSubnet:
    type: string
  StorageMgmt2IpSubnet:
    type: string
  InternalApi0IpSubnet:
    type: string
  InternalApi1IpSubnet:
    type: string
  InternalApi2IpSubnet:
    type: string
  Tenant0IpSubnet:
    type: string
  Tenant1IpSubnet:
    type: string
  Tenant2IpSubnet:
    type: string
  ExternalIpSubnet:
    type: string
  ManagementIpSubnet:
    type: string

  # VLAN IDs
  Storage0NetworkVlanID:
    type: number
  Storage1NetworkVlanID:
    type: number
  Storage2NetworkVlanID:
    type: number
  StorageMgmt0NetworkVlanID:
    type: number
  StorageMgmt1NetworkVlanID:
    type: number
  StorageMgmt2NetworkVlanID:
    type: number
  InternalApi0NetworkVlanID:
    type: number
  InternalApi1NetworkVlanID:
    type: number
  InternalApi1NetworkVlanID:
    type: number
  Tenant0NetworkVlanID:
    type: number
  Tenant1NetworkVlanID:
    type: number
  Tenant2NetworkVlanID:
    type: number
  ExternalNetworkVlanID:
    type: number
```

```
  ManagementNetworkVlanID:
    type: number

  # Subnet CIDR
  ControlPlane0SubnetCidr:
    type: string
  ControlPlane1SubnetCidr:
    type: string
  ControlPlane1SubnetCidr:
    type: string

  ControlPlaneIp:
    type: string
  DnsServers:
    type: comma_delimited_list

  # EC2 metadata server IPs
  Leaf0EC2MetadataIp:
    type: string
  Leaf1EC2MetadataIp:
    type: string
  Leaf2EC2MetadataIp:
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                [NETWORK CONFIG HERE]

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

APPENDIX C. EXAMPLE ROLES_DATA FILE

```
###############################################################################
#####
# Role: Controller0
#
###############################################################################
#####
- name: Controller0
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi0
    - Storage0
    - StorageMgmt0
    - Tenant0
  default_route_networks: ['External']
  HostnameFormatDefault: '%stackname%-controller0-%index%'
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::BarbicanBackendSimpleCrypto
    - OS::TripleO::Services::BarbicanBackendDogtag
    - OS::TripleO::Services::BarbicanBackendKmip
    - OS::TripleO::Services::BarbicanBackendPkcs11Crypto
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMgr
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackendDellPs
```

```
    - OS::TripleO::Services::CinderBackendDellSc
    - OS::TripleO::Services::CinderBackendDellEMCUnity
    - OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
    - OS::TripleO::Services::CinderBackendDellEMCVNX
    - OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
    - OS::TripleO::Services::CinderBackendNetApp
    - OS::TripleO::Services::CinderBackendScaleIO
    - OS::TripleO::Services::CinderBackendVRTSHyperScale
    - OS::TripleO::Services::CinderBackup
    - OS::TripleO::Services::CinderHPELeftHandISCSI
    - OS::TripleO::Services::CinderScheduler
    - OS::TripleO::Services::CinderVolume
    - OS::TripleO::Services::Clustercheck
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Congress
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Ec2Api
    - OS::TripleO::Services::Etcd
    - OS::TripleO::Services::ExternalSwiftProxy
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::GlanceApi
    - OS::TripleO::Services::GlanceRegistry
    - OS::TripleO::Services::GnocchiApi
    - OS::TripleO::Services::GnocchiMetricd
    - OS::TripleO::Services::GnocchiStatsd
    - OS::TripleO::Services::HAproxy
    - OS::TripleO::Services::HeatApi
    - OS::TripleO::Services::HeatApiCloudwatch
    - OS::TripleO::Services::HeatApiCfn
    - OS::TripleO::Services::HeatEngine
    - OS::TripleO::Services::Horizon
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::IronicApi
    - OS::TripleO::Services::IronicConductor
    - OS::TripleO::Services::IronicPxe
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Keepalived
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Keystone
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::ManilaApi
    - OS::TripleO::Services::ManilaBackendCephFs
    - OS::TripleO::Services::ManilaBackendIsilon
    - OS::TripleO::Services::ManilaBackendNetapp
    - OS::TripleO::Services::ManilaBackendUnity
    - OS::TripleO::Services::ManilaBackendVNX
    - OS::TripleO::Services::ManilaBackendVMAX
    - OS::TripleO::Services::ManilaScheduler
    - OS::TripleO::Services::ManilaShare
    - OS::TripleO::Services::Memcached
    - OS::TripleO::Services::MistralApi
    - OS::TripleO::Services::MistralEngine
    - OS::TripleO::Services::MistralExecutor
    - OS::TripleO::Services::MistralEventEngine
    - OS::TripleO::Services::MongoDb
    - OS::TripleO::Services::MySQL
```

```
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronApi
    - OS::TripleO::Services::NeutronBgpVpnApi
    - OS::TripleO::Services::NeutronSfcApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL2gwAgent
    - OS::TripleO::Services::NeutronL2gwApi
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronLbaasv2Agent
    - OS::TripleO::Services::NeutronLbaasv2Api
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaDeploymentConfig
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::SkydiveAnalyzer
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftDispersion
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Timezone
```

```
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::Ptp
##############################################################################
#####
# Role: Compute0
#
##############################################################################
#####
- name: Compute0
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi0
    - Tenant0
    - Storage0
  HostnameFormatDefault: '%stackname%-compute0-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
```

```
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##############################################################################
#####
# Role: Compute1
#
##############################################################################
#####
- name: Compute1
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi1
    - Tenant1
    - Storage1
  HostnameFormatDefault: '%stackname%-compute1-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker
```

```
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##############################################################################
#####
# Role: Compute2
#
##############################################################################
#####
- name: Compute2
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi2
    - Tenant2
    - Storage2
  HostnameFormatDefault: '%stackname%-compute2-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
```

```
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##########################################################################
#####
# Role: CephStorage0
#
##########################################################################
#####
- name: CephStorage0
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage0
    - StorageMgmt0
  HostnameFormatDefault: '%stackname%-cephstorage0-%index%'
  uses_deprecated_params: False
```

```
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp
################################################################################
#####
# Role: CephStorage1
#
################################################################################
#####
- name: CephStorage1
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage1
    - StorageMgmt1
  HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
  uses_deprecated_params: False
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Ntp
```

```
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp
##############################################################################
#####
# Role: CephStorage2
#
##############################################################################
#####
- name: CephStorage2
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage2
    - StorageMgmt2
  HostnameFormatDefault: '%stackname%-cephstorage2-%index%'
  uses_deprecated_params: False
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp
```