Contents

Overview

The OpenStack Havana Release All-In-One (AIO) deployment builds off of the Cisco OpenStack Installer (COI) instructions. The Cisco OpenStack Installer provides support for a variety of deployment scenarios to include:

- All-in-One
- All-in-One plus additional Compute nodes
- 2 Node
- Full HA
- Compressed HA

This document will cover the deployment of two networking scenarios based on the All-in-One scenario:

- Model 1: All-in-One node using the Per-Tenant Router with Private Networks model for tenant network access (FlatDHCP + FloatingIPs using a Neutron Router)
- Model 2: All-in-One with an additional Compute node using Per-Tenant Router with Private Networks
- Model 3: All-in-One with an additional Compute node using Provider Network Extensions with VLANs (VLANs trunked into nodes from ToR switch)

Before you begin

Please read the release notes for the release you are installing. The release notes contain important information about limitations, features, and how to use snapshot repositories if you're installing an older release.

Diagrams

Figure 1 illustrates the topology used in Model 1

Figure 1: AIO Per-Tenant Router with Private Networks Diagram

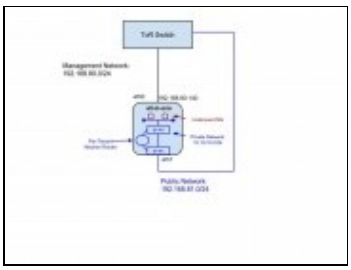


Figure 2 illustrates the topology used in Model 2

Figure 2: AIO & Additional Compute Node using Per-Tenant Router with Private Networks Diagram

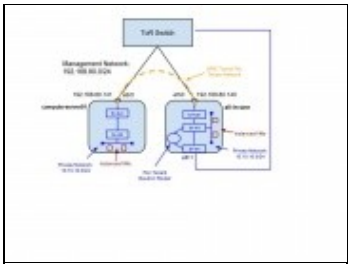
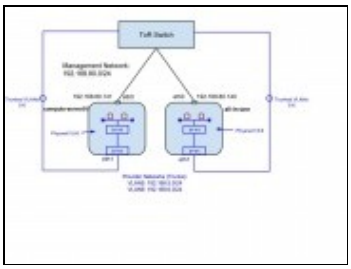


Figure 3' illustrates the topology used in Model 3

Figure 3: AIO & Additional Compute Node using Provider Network Extensions with VLANs Diagram



Model 1

This section describes the process for deploying OpenStack with the Cisco OpenStack Installer in an All-In-One node configuration with Per-Tenant Routers with Private Networks

Assumptions

- The Cisco OpenStack Installer requires that you have two physically or logically (VLAN) separated IP networks. One network is used to provide connectivity for OpenStack API endpoints, Open vSwitch (OVS) GRE endpoints (especially important if multiple compute nodes are added to the AIO deployment), and OpenStack/UCS management. The second network is used by OVS as the physical bridge interface and by Neutron as the public network. An example of the AIO node

/etc/network/interfaces file:

```
# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.80.140
        netmask 255.255.255.0
        network 192.168.80.0
        broadcast 192.168.80.255
        gateway 192.168.80.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 8.8.8.8 8.8.4.4
        dns-search example.com

auto eth1
iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ifconfig $IFACE 0.0.0.0 down
```

- The AIO node is running Ubuntu Linux 12.04 LTS, deployed on physical baremetal hardware (i.e. Cisco UCS) or as a Virtual Machine (i.e. VMware ESXi). For performance reasons, a baremetal install is recommended for any non-trivial application. See OpenStack: Installing Ubuntu Linux for instructions to complete an installation of Ubuntu from an ISO image onto a standalone server.

- You have followed the installation steps in the Cisco OpenStack Installer (COI) instructions. **Note:** A recap of the AIO-specific instructions are provided below.

- You are using hostnames for the various OpenStack roles that match those in the **/root/puppet_openstack_builder/data/role_mappings.yaml** file. If you are not using the default hostnames then you must add your custom hostname and role to the **/root/puppet_openstack_builder/data/role_mappings.yaml** file before running the installation script.
    - The default hostname for the AIO node is:

      ```
      all-in-one
      ```

Building the All-in-One OpenStack Node

The deployment of the AIO node in Model 1 will begin after a fresh install of Ubuntu 12.04 LTS and with the network configuration based on the example shown in Figure 1.

On the node that you just built, become root:

```
sudo su -
```

Install git:

```
apt-get install -y git
```

Clone the Cisco OpenStack Installer repository:

```
cd /root && git clone -b havana https://github.com/CiscoSystems/puppet_openstack_builder && cd pup
```

Note: Before running the installation script for COI it is important to make any modifications to the baseline AIO configuration if you have non-standard interface definitions, hostnames (can be viewed in

/root/puppet_openstack_builder/data/role_mappings.yaml file), proxies, etc... Details on setting some of these custom values can be found in the Cisco OpenStack Installer (COI) instructions.

Here are three examples that include a way to set custom interface definitions and custom hostnames for the AIO Model 1 setup:

- If you are using an interface other than 'eth0' on your node for SSH/Management access then export the default_interface value to the correct interface definition. In the example below, eth1 is used:
  - `export default_interface=eth1 # This is the interface you logged into via ssh`
- If you are using an interface other than 'eth1' on your node for external instance (public) access then export the external_interface value. In the example below, eth2 is used:
  - `export external_interface=eth2`
- If you are using a hostname other than "all-in-one" for the AIO node then you must update the **/root/puppet_openstack_builder/data/role_mappings.yaml** file to include your hostname and its role. For example, if your hostname is "all-in-one-test1" then the **role_mappings.yaml** file should have an entry that looks like this:
  - `all-in-one-test1: all_in_one`

Export 'cisco' as the vendor:

```
export vendor=cisco
```

Export the AIO scenario:

```
export scenario=all_in_one
```

Change directory to where the install script is located and start the installation (this will take awhile depending on your Internet connection):

```
cd ~/puppet_openstack_builder/install-scripts
./install.sh 2>&1 | tee install.log
```

After the install script and Puppet run are completed, you should be at the prompt again with a "Finished catalog run". You can verify that all of the OpenStack Nova services were installed and running correctly by checking the Nova service list:

```
root@all-in-one:~# nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-consoleauth all-in-one                           internal         enabled    :-)   2014-03-11
nova-scheduler   all-in-one                           internal         enabled    :-)   2014-03-11
nova-conductor   all-in-one                           internal         enabled    :-)   2014-03-11
nova-compute     all-in-one                           nova             enabled    :-)   2014-03-11
nova-cert        all-in-one                           internal         enabled    :-)   2014-03-11
```

You can connect into the OpenStack Dashboard by entering:

```
http://ip-of-your-aio
```

using username ***admin*** and password ***Cisco123***.

Neutron Networking for Models 1 & 2

This section will walk through buiding a Per-Tenant Router with Private Networks Neutron setup. You can opt to perform all of the steps below in the OpenStack Dashboard or via CLI. The CLI steps are shown below. Also, please consult the Figure 1 diagram so that you can easily understand the network layout used by Neutron in our example.

Before running OpenStack client commands, you need to source the installed openrc file located in the **/root/** directory:

```
source openrc
```

Create a public network to be used for instances (VMs) to gain external (public) connectivity:

```
neutron net-create Public_Network --router:external=True
```

Create a subnet that is associated with the previously created public network. **Note:** If you have existing hosts on the same subnet that you are about to use for the public subnet then you must use an allocation pool that starts in a range that will not conflict with other network nodes. One example of this is if you have HSRP/VRRP/GLPB upstream and they are using addresses in the public subnet ranges (i.e. 192.168.81.1, 192.168.81.2, 192.168.81.3) then your allocation range must start in a non-overlapping range.

```
neutron subnet-create --name Public_Subnet --allocation-pool start=192.168.81.10,end=192.168.81.25
```

Create a private network and subnet that will be used to attach instances to:

```
neutron net-create Private_Net10
neutron subnet-create --name Private_Net10_Subnet Private_Net10 10.10.10.0/24 --dns_nameservers li
```

Create a Neutron router:

```
neutron router-create os-router-1
```

Associate a Neutron router interface with the previously created private subnet:

```
neutron router-interface-add os-router-1 Private_Net10_Subnet
```

Set the default gateway (previously created public network) for the Neutron router:

```
neutron router-gateway-set os-router-1 Public_Network
```

Modify the default Neutron security group to allow for ICMP and SSH (for access to the instances):

```
neutron security-group-rule-create --protocol icmp --direction ingress default
neutron security-group-rule-create --protocol tcp --port-range-min 22 --port-range-max 22 --direct
```

SSH Keys for Models 1, 2 and 3

Create SSH keys from the node that will be used to SSH into the OpenStack instances (example keypair name is "aio-key"):

```
ssh-keygen
cd /root/.ssh/
nova keypair-add --pub_key id_rsa.pub aio-key
```

Upload Images into Glance for Models 1, 2 and 3

Download the image of your choice. Below there are examples for downloading Cirros, Ubuntu 12.04 and Fedora:

- Cirros:

```
wget http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
```

- Fedora:

```
wget http://download.fedoraproject.org/pub/fedora/linux/releases/19/Images/x86_64/Fedora-x8
```

- Ubuntu:

```
wget http://cloud-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img
```

Upload the images into Glance:

- Cirros:

```
glance image-create --name cirros-x86_64 --is-public True --disk-format qcow2 --container-f
```

- Fedora:

```
glance image-create --name Fedora --is-public True --disk-format qcow2 --container-format b
```

- Ubuntu:

```
glance image-create --name precise-x86_64 --is-public True --disk-format qcow2 --container-
```

Boot an Instance - for Models 1 and 2

1. Boot an Instance (Cirros image example shown below). Run the **neutron net-list** command to get a list of networks. Use the ID for the Private_Net10 network from the net-list output in the **--nic net-id=** field:

```
root@all-in-one:~# neutron net-list
+--------------------------------------+----------------+--------------------------------------
| id                                   | name           | subnets
+--------------------------------------+----------------+--------------------------------------
| 42823c88-bb86-4e9a-9f7b-ef1c0631ee5e | Private_Net10  | f48bca75-7fe4-4510-b9fd-c0323e416376 10.
| 85650115-093b-49be-9fe1-ba2d34b4d3e2 | Public_Network | 2d89ac21-3611-44ef-b5d7-924fd7854e0d 192
+--------------------------------------+----------------+--------------------------------------

nova boot --image cirros-x86_64 --flavor m1.tiny --key_name aio-key --nic net-id=42823c88-bb86-4e9
```

Verify that your instance has spawned successfully. **Note:** The first time an instance is launched on the system it can take a bit longer to boot than subsequent launches of instances:

```
nova show test-vm1
```

2. Verify connectivity to the instance from the AIO node. Since namespaces are being used in this model, you will need to run the commands from the context of the qrouter using the **ip netns exec qrouter** syntax. List the qrouter to get its router-id, connect to the qrouter and get a list of its addresses, ping the instance from the qrouter and then SSH into the instance from the qrouter:

```
root@all-in-one:~# neutron router-list
+--------------------------------------+-------------+--------------------------------------
| id                                   | name        | external_gateway_info
+--------------------------------------+-------------+--------------------------------------
```

```
| 58d8840a-74ca-48a2-a6f2-7853eef9a36e | os-router-1 | {"network_id": "85650115-093b-49be-9fe1-ba2
+--------------------------------------+-------------+-----------------------------------------
```

Alternatively, you can get the qrouter ID via:

```
root@all-in-one:~# ip netns
qdhcp-d7c039ae-6f11-429b-aaf9-10a2d659608a
qrouter-58d8840a-74ca-48a2-a6f2-7853eef9a36e

ip netns exec qrouter-<neutron-router-id> ip addr list
ip netns exec qrouter-<neutron-router-id> ping <fixed-ip-of-instance>
ip netns exec qrouter-<neutron-router-id> ssh cirros@<fixed-ip-of-instance>
```

NOTE: You can get the internal fixed IP of your instance with the following command: nova show <your_instance_name>

3. Create and associate a Floating IP. You will need to get a list of the networks and copy the correct IDs:

```
neutron net-list
neutron port-list
neutron floatingip-create --port_id <internal VM port-id> <public net-id>
```

- Example:

```
root@all-in-one:~# neutron floatingip-create --port_id 5510471e-2b48-4736-9112-aee22f3c6ecb e1a318
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    | 10.10.10.2                           |
| floating_ip_address | 192.168.81.12                        |
| floating_network_id | e1a31822-26f1-461a-85b9-7d1e084e619c |
| id                  | 48ffe524-43bc-44fc-85b5-9c568ed64af1 |
| port_id             | 5510471e-2b48-4736-9112-aee22f3c6ecb |
| router_id           | 526f262b-225e-4e1d-9a5b-1619f806960a |
| tenant_id           | 5ed6e50345bb49cfa0090746fdb68533     |
+---------------------+--------------------------------------+
```

4. Ping and SSH to your instance using the "floating_ip_address" from an external host.

Model 2

This section describes the process for deploying OpenStack with the COI All-in-One node plus a 2nd node acting as a dedicated OpenStack compute role. This model will use Per-Tenant Routers with Private Networks and will build upon the AIO install from Model 1. This section will not include a Neutron networking, Image upload, SSH Keys or Instance boot section as everything in the Model 1 example directly applies to Model 2. The focus of this section is only for adding the 2nd compute node.

Reference Figure 2 to align the topology with the configuration steps below.

Assumptions

- In the Model 2 example, the AIO node will act as the 'networking node' and all traffic into and out of the instances running on the 2nd compute node will traverse a GRE tunnel between the br-tun interfaces on the AIO node and the compute node.

- The compute node is built on Ubuntu 12.04 LTS which can be installed via manual ISO/DVD or PXE setup and can be deployed on physical baremetal hardware (i.e. Cisco UCS) or as Virtual Machines (i.e. VMware ESXi).

- You have followed the installation steps in the Cisco OpenStack Installer (COI) instructions or the steps in Model 1 above to install the AIO node.

- You have added a local host entry or DNS entry for the AIO node. In our example there should be a hosts entry on the compute node that looks like this:
    - `192.168.80.140  all-in-one.example.com  all-in-one`

- You are using hostnames for the various OpenStack roles that match those in the **/root/puppet_openstack_builder/data/role_mappings.yaml** file. If you are not using the default hostnames then you must add your custom hostname and role to the **/root/puppet_openstack_builder/data/role_mappings.yaml** before running the installation script. For example, if your compute node hostname is "compute-server01-test1" then the **role_mappings.yaml** file should have an entry that looks like this:
    - `compute-server01-test1: compute`

Building the OpenStack Compute Node

The deployment of the compute node in Model 2 will begin after a fresh install of Ubuntu 12.04 LTS and with the network configuration based on the example shown in Figure 2 and after you have completed the full Model 1 walk-thru (the setup of the compute node requires that the AIO node is already deployed).

On the node that you just built, become root:

```
sudo su -
```

Install git:

```
apt-get install -y git
```

Clone the Cisco OpenStack Installer repository:

```
cd /root && git clone -b havana https://github.com/CiscoSystems/puppet_openstack_builder && cd pup
```

Change to the install_scripts directory:

```
cd install_scripts/
```

Export the IP address of your AIO node (which is also acting as the Puppet master/Build Server):

```
export build_server_ip=192.168.80.140
```

Run the setup.sh file to prep the node for the Puppet agent run:

```
bash setup.sh
```

Begin the OpenStack Compute node build by starting the Puppet agent (Note the "all-in-one.example.com" hostname used. Modify for your environment):

```
puppet agent -td --server=all-in-one.example.com --pluginsync
```

After the Puppet agent runs, you will end up at the prompt after a "Finished catalog run" message.

Verify that the OpenStack Nova services are running on the all-in-one node and computer-server01 node:

```
root@all-in-one:~# nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-consoleauth all-in-one                           internal         enabled    :-)   2014-03-11
nova-scheduler   all-in-one                           internal         enabled    :-)   2014-03-11
nova-conductor   all-in-one                           internal         enabled    :-)   2014-03-11
nova-compute     all-in-one                           nova             enabled    :-)   2014-03-11
nova-cert        all-in-one                           internal         enabled    :-)   2014-03-11
nova-compute     compute-server01                     nova             enabled    :-)   2014-03-11
```

You can connect into the OpenStack Dashboard by entering:

```
http://ip-of-your-aio
```

using username ***admin*** and password ***Cisco123***.

You can also verify that the new compute node appears in the OpenStack Nova Hypervisor-list:

```
root@all-in-one:~# nova hypervisor-list
+----+-------------------------------+
| ID | Hypervisor hostname           |
+----+-------------------------------+
| 1  | all-in-one.example.com        |
| 2  | compute-server01.example.com  |
+----+-------------------------------+
```

Launch an Instance

You can follow all of the steps in Model 1 to setup your Neutron network, Glance images, SSH keys and Instance launch as they will directly apply to Model 2. However, if you want to test that an instance successfully launches against your new compute node direclty then you can boot an instance and identify the compute node by name:

```
nova boot --image cirros-x86_64 --flavor m1.tiny --key_name aio-key --nic net-id=42823c88-bb86-4e9
```

Check to see if the instance has launched against the compute node:

```
root@all-in-one:~# nova hypervisor-servers compute-server01
+--------------------------------------+-------------------+---------------+----------------------
| ID                                   | Name              | Hypervisor ID | Hypervisor Hostname
+--------------------------------------+-------------------+---------------+----------------------
| ba995773-bb1c-419c-9aa1-be67d6967345 | instance-00000006 | 2             | compute-server01.exam
+--------------------------------------+-------------------+---------------+----------------------
```

Model 3

This section describes the process for deploying OpenStack with the COI All-in-One node plus a 2nd node acting as a dedicated OpenStack compute role. This model will use Provider Network Extensions with VLANs networking model.

You will build upon the installation of Ubuntu and the deployment of OpenStack from Models 1 and 2. You will modify the appropriate configuration files to account for the change between the Per-Tenant Router with

Private Networks model and this Provider Network Extensions with VLANs model.

Assumptions

- The Provider Network Extensions with VLANs networking model requires that you have two physically or logically (VLAN) separated IP networks on both the AIO and the compute node. One network is used to provide connectivity for OpenStack API endpoints, Open vSwitch (OVS) GRE endpoints (especially important if multiple compute nodes are added to the AIO deployment), and OpenStack/UCS management. The second network is used by OVS as the physical bridge interface and by Neutron as the public network. An example of both the AIO and the Compute node **/etc/network/interfaces** files is shown below:

```
# The primary network interface for all-in-one
auto eth0
iface eth0 inet static
        address 192.168.80.140
        netmask 255.255.255.0
        network 192.168.80.0
        broadcast 192.168.80.255
        gateway 192.168.80.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 8.8.8.8 8.8.4.4
        dns-search example.com

auto eth1
iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ifconfig $IFACE 0.0.0.0 down

# The primary network interface for compute-server01
auto eth0
iface eth0 inet static
        address 192.168.80.141
        netmask 255.255.255.0
        network 192.168.80.0
        broadcast 192.168.80.255
        gateway 192.168.80.1
        # dns-* options are implemented by the resolvconf package, if installed
        dns-nameservers 8.8.8.8 8.8.4.4
        dns-search example.com

auto eth1
iface eth1 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ifconfig $IFACE 0.0.0.0 down
```

- The AIO node and compute nodes are built on Ubuntu 12.04 LTS which can be installed via manual ISO/DVD or PXE setup and can be deployed on physical baremetal hardware (i.e. Cisco UCS) or as Virtual Machines (i.e. VMware ESXi).

- You have followed the previously shown installation steps for setting up OpenStack on each node.

- You are using hostnames for the various OpenStack roles that match those in the **/root/puppet_openstack_builder/data/role_mappings.yaml** file. If you are not using the default hostnames then you must add your custom hostname and role to the **/root/puppet_openstack_builder/data/role_mappings.yaml** before running the installation script.

See the notes on custom interfaces and hostnames in the Model 1 and 2 sections above.

- You have configured the ToR switch and any upstream network devices for VLAN trunking for the range of VLANs you will use in the setup. The examples below will use VLANs 5 and 6 and they are being trunked from the Data Center Aggregation layer switches into the ToR and then directly into the 'eth1' interface on both the AIO and the compute nodes. Please see Figure 3 for the topology layout.

Modifying Models 1 and 2 for Provider Network Extensions with VLANs

As was mentioned before, you are going to modify the setup you had in Models 1 and 2 (AIO node built first [Model 1] and then add the compute node [Model 2]).

There are a few configuration variables that have to be modified in various COI Puppet yaml files in order for the Provider Network Extensions with VLANs networking design to work.

Set the bridge_uplinks, VLAN ranges and bridge mappings in the **/etc/puppet/data/hiera_data/user.yaml** file. The settings below instructs Puppet to associate the "br-ex" bridge with the physical "eth1" interface on both the AIO node and the compute node and identify the pair as the OVS bridge uplinks to be used for trunks to the ToR. The next line identifies the user-defined name of the OVS network name "physnet1" and the VLAN ranges (5-6). Finally, the last entry creates an OVS mapping between the "physnet1" network name and the external bridge "br-ex" :

```
root@all-in-one:~# vim /etc/puppet/data/hiera_data/user.yaml
neutron::agents::ovs::bridge_uplinks: br-ex:eth1
neutron::plugins::ovs::network_vlan_ranges: physnet1:5:6
neutron::agents::ovs::bridge_mappings:
  - "physnet1:br-ex"
```

Set the network_type and tenant_network_type in the **/etc/puppet/data/global_hiera_params/common.yaml** file. This changes the network_type from the COI default of "per-tenant router" to the "provider-router" type (indicating that there is an external upstream router - i.e. the Data Center Aggregation layer switches). Finally, the tenant_network_type is changed from the default of "gre" to "vlan":

```
root@all-in-one:~# vim /etc/puppet/data/global_hiera_params/common.yaml
network_type: provider-router
tenant_network_type: vlan
```

In our example below you are going to disable IP namespaces because you are going to segregate tenant networks via VLANs and upstream networking policies. Modify the **/etc/puppet/data/hiera_data/network_type/provider-router.yaml** file:

```
root@all-in-one:~# vim /etc/puppet/data/hiera_data/network_type/provider-router.yaml
neutron::agents::dhcp::use_namespaces: false
```

Note: If you are using custom interface mappings on either/both of your nodes then you need to modify the default interface definitions for the OVS settings there were defined above. For example, if the compute-server01 node is using 'eth2' as its external interface (the one VLANs are trunked to) then you can create a custom hostname-based yaml file and modify the OVS settings like this:

```
root@all-in-one:~# vim /etc/puppet/data/hiera_data/hostname/compute-server01.yaml
external_interface: eth2
```

```
neutron::agents::ovs::bridge_uplinks: br-ex:eth2
neutron::plugins::ovs::network_vlan_ranges: physnet1:5:6
neutron::agents::ovs::bridge_mappings:
  - "physnet1:br-ex"
```

Now that the configuration files have been updated, run Puppet on the AIO node:

```
root@all-in-one:~# puppet apply -v /etc/puppet/manifests/site.pp
```

Restart the Neutron and OVS services:

```
root@all-in-one:~# cd /etc/init.d/; for i in $( ls neutron-* ); do sudo service $i restart; done
root@all-in-one:~# cd /etc/init.d/; for i in $( ls openvswitch-* ); do sudo service $i restart; do
```

Re-run the Puppet agent on the compute node:

```
root@compute-server01:~# puppet agent -td --server=all-in-one.example.com --pluginsync
```

You can connect into the OpenStack Dashboard by entering:

```
http://ip-of-your-aio
```

using username ***admin*** and password ***Cisco123***.

Neutron Networking for Model 3

The steps for setting up the Neutron Networking for Model 3 is similar to Models 1 and 2 only there is no concept of a 'private' network and, in this example, a Neutron router is not being used. The instance will map directly to the network that is logically representing the VLAN networks that were previously trunked into OVS.

Source the openrc file:

```
source openrc
```

Create a Neutron Provider Network. In the example below, you will create a network named "vlan5", identify the network type as "vlan", associate that network with a physical network named "physnet1" and the segmentation ID is "5" (mapping to VLAN 5 being trunked):

```
neutron net-create vlan5 --provider:network_type vlan --provider:physical_network physnet1 --provi
```

Create a Neutron subnet that is associated with the network that was defined in the previous step. Again, be aware of existing hosts on this network that may already be using IP address out of that subnet range. Create an allocation pool that is in a 'free' range of IPs:

```
neutron subnet-create --name subnet5 --allocation-pool start=192.168.5.10,end=192.168.5.254 vlan5
```

Repeat the same steps for VLAN 6:

```
neutron net-create vlan6 --provider:network_type vlan --provider:physical_network physnet1 --provi
neutron subnet-create --name subnet6 --allocation-pool start=192.168.6.10,end=192.168.6.254 vlan6
```

Modify the default Neutron security group to allows for ICMP (for pings) and SSH (for access to the instances):

```
neutron security-group-rule-create --protocol icmp --direction ingress default
neutron security-group-rule-create --protocol tcp --port-range-min 22 --port-range-max 22 --direct
```

Launch an Instance

Follow the steps in Model 1 to ensure that you have images uploaded into Glance and that you have SSH keys uploaded into Nova.

Get the list of Neutron networks:

```
root@all-in-one:~# neutron net-list
+--------------------------------------+-------+-----------------------------------------------
| id                                   | name  | subnets
+--------------------------------------+-------+-----------------------------------------------
| 7c79fddf-3e51-473d-96f3-2af822e05dbf | vlan6 | a4fdb5dc-8319-4f47-8991-00186e0d622d 192.168.6.0/
| f7d7ff38-a23a-463c-bdd0-145abc7f82c0 | vlan5 | 6c55a72f-7b8c-4445-91b6-18c7c1c28d5b 192.168.5.0/
+--------------------------------------+-------+-----------------------------------------------
```

Boot the test instance using the ID from the previous step:

```
root@all-in-one:~# nova boot --image cirros-x86_64 --flavor m1.tiny --key_name aio-key --nic net-i
```

You can now ping/SSH directly to the instance as it has an IP address within the subnet associated with the trunked VLAN.

Known Issues

For a complete list of bugs, please visit: https://bugs.launchpad.net/openstack-cisco

In an AIO deployment, you will see several warnings about collecting exported resources without storeconfigs being enabled during the initial puppet run. These should go away if you do further puppet catalog runs. See also Bug #1282281

In an AIO deployment, you may see error messages about three swift services not starting: swift-container-replicator, swift-container-sync, and swift-account-replicator. This is due to a race condition that may cause these services to be started before the Swift ring sync has completed. If you run into this problem, you can simply start the services manually ("service swift-container-replicator start", etc) or simply perform a second puppet catalog run. See also Bug #1274358

In an AIO deployment or other installation which includes Swift, you may see warnings like 'swift storage server $service must specify $service-server'. These are harmless and can be ignored--they're caused by an upstream bug that issues spurious warnings. Refer to Bug #1289187

You may see warnings like 'keystone_host, keystone_port and keystone_scheme are deprecated. Use keystone_url instead'. These are harmless and can be ignored. They will go away when Cisco moves to Icehouse.

You may see warnings about nagios service restarting. These are harmless and can be ignored; they simply mean that the example configuration of nagios monitoring will not function, but they do not impact OpenStack functioning in any way.

Users deploying Neutron's Firewall as a Service should note that they may need to include "--config-file /etc/neutron/fwaas_driver.ini" when starting up the neutron l3 agent.

Associated Documents

- Deploying LBaaS on COI H.2

Authors

Shannon McFarland (@eyepv6) - Principal Engineer