Openstack – Mitaka
LBaaSv2, HEAT

UDF/Openstack Lab Guide

July 2016

Nicolas Ménant

Lab design and access

About the OpenStack installation

OpenStack has been installed via the RDO project: https://www.rdoproject.org/

For this lab, we will use the latest version of OpenStack: Mitaka

The openstack infrastructure contains the following components:

- 1 Openstack Controller
- 1 Openstack Network Node
- 2 Openstack compute nodes

Neutron is configured with the following setup:

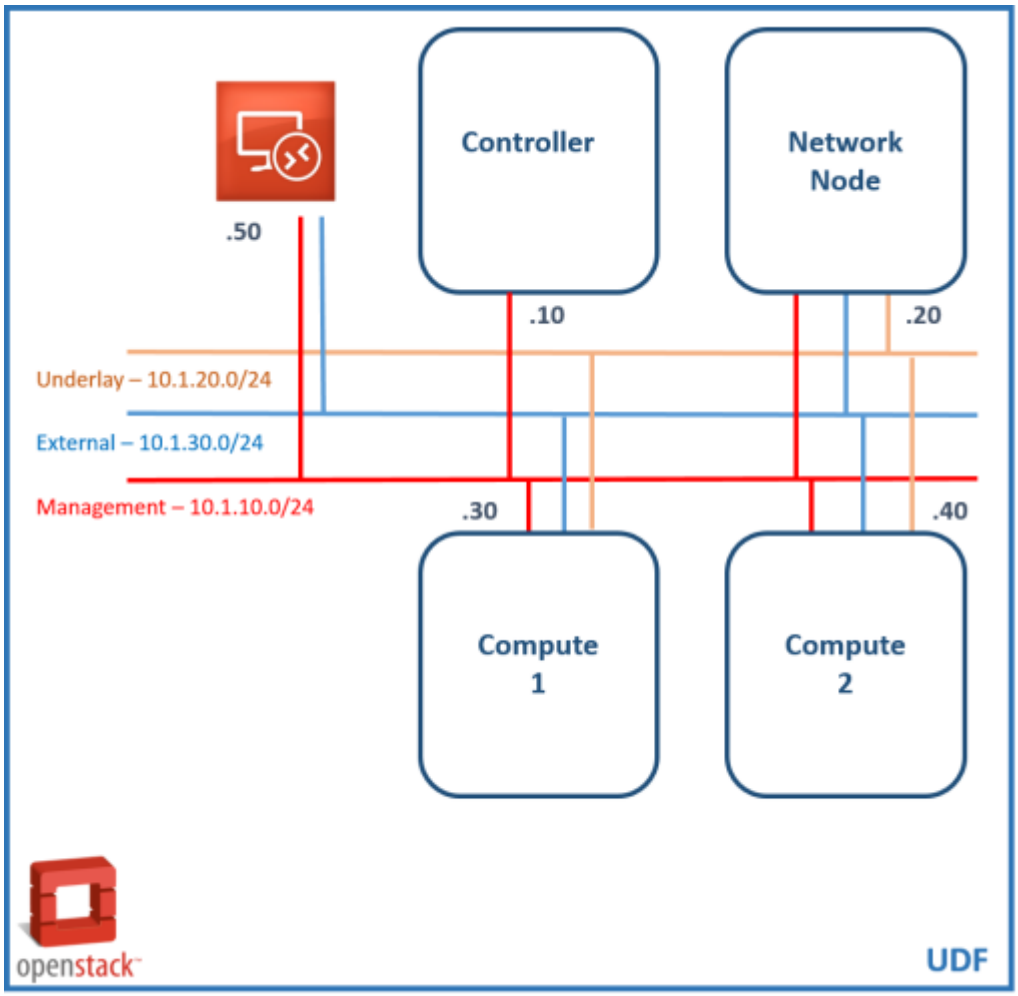
- DVR is enabled
- Gre tunnel is used for overlay networks running on top of the "Underlay Network" (defined in UDF). We could have used vxlan in a similar way

The only remaining tasks are:

- to setup and define your tenants' networks
- install the F5 LBaaS plugin
- use HEAT to do some automated deployments

Lab design

Here is a summary of the OpenStack lab design.

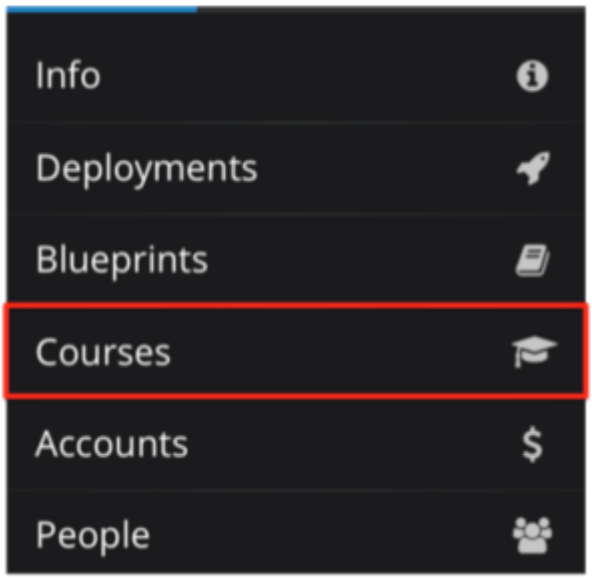


There is an additional network, which is created by default by UDF (Management – 10.1.1.X/24). It ensures proper remote access to each component's console from the UDF interface. This network is not shown here since it is irrelevant to the lab.

Deploy your environment

To have your own environment, you'll need to access F5's UDF at this url: https://itc.io

Once you're logged in, go to "courses":



Click on the Course called "[Cloud] OpenStack labs" showing the status "In progress". Once you're inside the course, click on "Join the session". Be aware that you can only join one course at a time.

Wait for your environment to be deployed.

To access the lab remotely, you have several options:

- Connect directly via SSH to your components (work for the Controller, Network node and compute nodes) – convenient for CLI manipulation. For more information, here, check this UDF link: https://f5udf.freshdesk.com/support/solutions/articles/5000664764-ssh-component-access
- Connect to the RDP desktop and do everything from here (convenient when latency can be an issue) – required to manipulate the Openstack Dashboard (Horizon) and potentially have access to VM consoles within Openstack

The RDP host has access to all the different components. The following tools are installed on it:

- Putty client with bookmarks to access the Controller, Network node, compute nodes.
- Chrome with a bookmark to access the OpenStack Dashboard.

Lab credentials

Here are the different credentials needed

- RDP
  - Login user
  - Password user

- Openstack Controller, Network node and Compute nodes (SSH)
  - Login: root
  - Password: default
  - In the /root/ directory, you have a keystone auth files to be able to authenticate and execute openstack cli commands (just type "source <keystone auth file>" to load it into your environment variables. You have an admin (keystonerc_admin) and a demo (keystonrc_demo).
    - The admin file is to be able to do admin task within your environment
    - The demo file is to be able to simulate the demo user assigned to the demo project

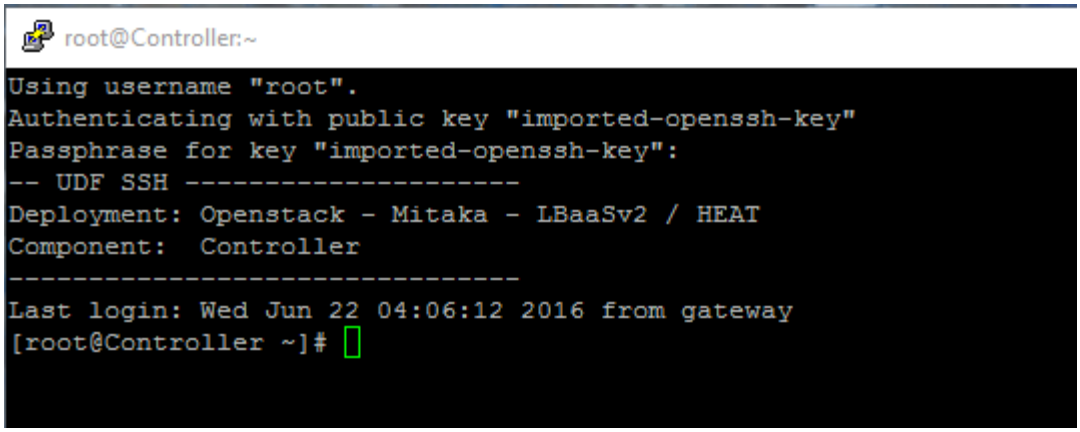
- Horizon dashboard/projects
  - Admin tenant:
    - Login: admin
    - Password: admin
  - Demo tenant:
    - Login: demo
    - Password: demo

The admin and demo password have been setup to something easy. You may consider to strengthen the passwords if you want to use the web proxy in UDF to access the Openstack Dashboard, you may do so (be warned that console for the VMs deployed in Openstack won't work) and still be (kind of) secure.

Check here to learn about enabling the webproxy feature in UDF:
https://f5udf.freshdesk.com/support/solutions/articles/5000643979-webproxy-feature

If you want to change the admin / demo password to something more complex, here is the procedure (not recommended for this lab):

1. Connect to your Openstack Controller (via the SSH feature in UDF or from putty in your RDP desktop)



2. Type "source keystonerc_admin"
3. Use the following command to update the password:
   `keystone user-password-update --pass <password> <user id>`
   You may see python message related to the deprecation of this command but it will still work as expected

Example to change the admin password:

[root@Controller ~]# source keystonerc_admin
[root@Controller ~(keystone_admin)]# keystone user-password-update --pass admin 54fgdgere5g4g5erg
/usr/lib/python2.7/site-packages/keystoneclient/shell.py:64: DeprecationWarning: The keystone CLI is deprecated in favor of python-openstackclient. For a Python library, continue using python-keystoneclient.
'python-keystoneclient.', DeprecationWarning)
/usr/lib/python2.7/site-packages/keystoneclient/v2_0/client.py:145: DeprecationWarning: Constructing an instance of the keystoneclient.v2_0.client.Client class without a session is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
'the 2.0.0 release.', DeprecationWarning)
/usr/lib/python2.7/site-packages/keystoneclient/auth/identity/base.py:56: DeprecationWarning: keystoneclient auth plugins are deprecated as of the 2.1.0 release in favor of keystoneauth1 plugins. They will be removed in future releases.

4. Once you've updated the password, you'll need to also update the admin password in the keystone_adminrc file. This is the following line that needs to be updated: export OS_PASSWORD=admin
5. Once you updated the file, you need to do a "source /root/keystonerc_admin" again

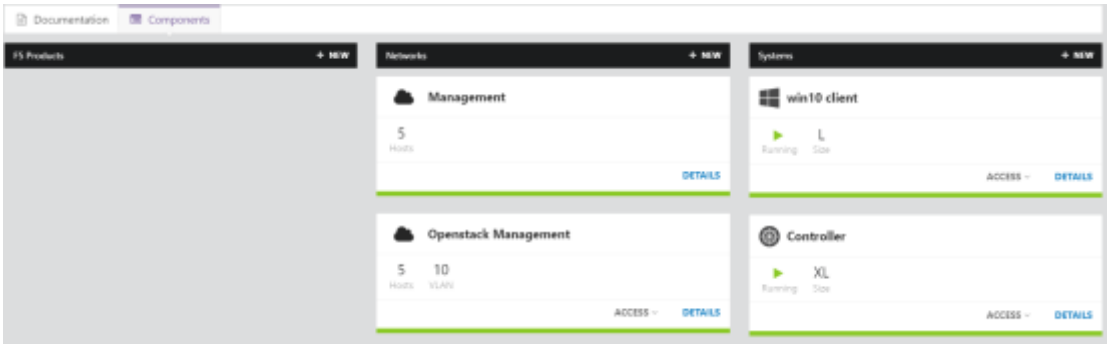
[root@Controller ~(keystone_admin)]# vi keystonerc_admin
[root@Controller ~(keystone_admin)]# source keystonerc_admin

If you want to do the same for demo:

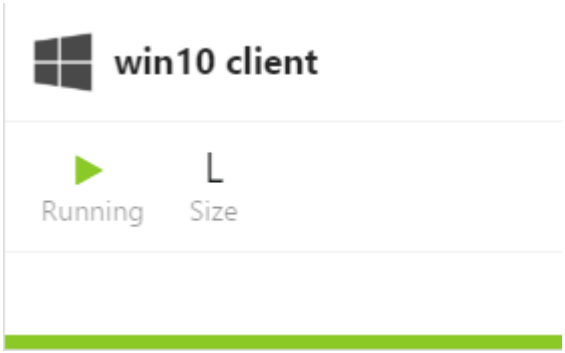
[root@Controller ~(keystone_admin)]# keystone user-password-update --pass demo <your demo password>

Access your lab via RDP

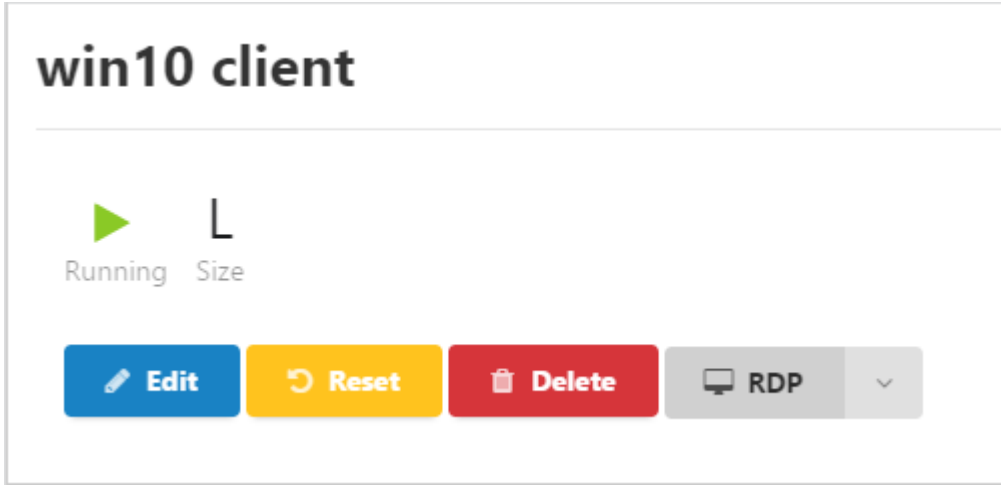
In the course, click on the “Components” tab



Once it’s done, click on “Details” in the “Win10 client” Box (Systems column)



Click on “RDP” to download a RDP shortcut that will launch your RDP session with all the required information.



For MAC user, it is recommended to use "Microsoft Remote Desktop". It is available in the App store (FREE).

Click on the shortcut you downloaded to have access to your environment.

For some reason, your Windows 10 client may crash after a few minutes on it. This will trigger a reboot of it (you can check this by going to UDF, Win10 Client and check its console). This happen once after the environment is setup and won't happen again.

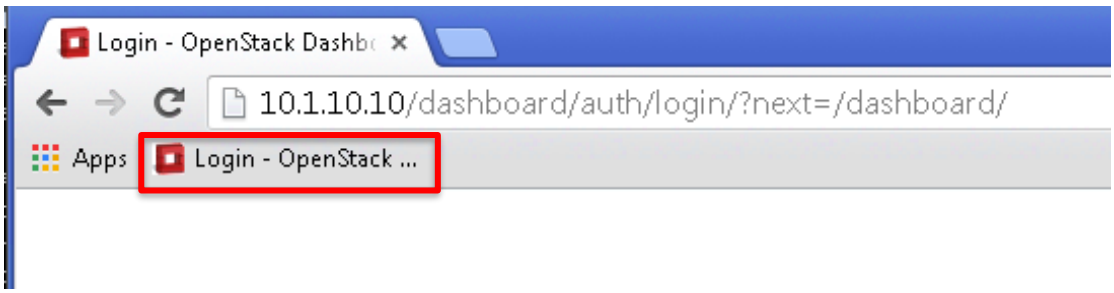
Before we create our networks for the demo project, we need to make sure that everything is running as expected:

1. We have access to our OpenStack environment in CLI / UI
2. Test we have access to all the nodes and they run as expected
3. Review our neutron agent's states

Test access to the OpenStack environment

We need to test GUI and CLI access to our OpenStack environment.

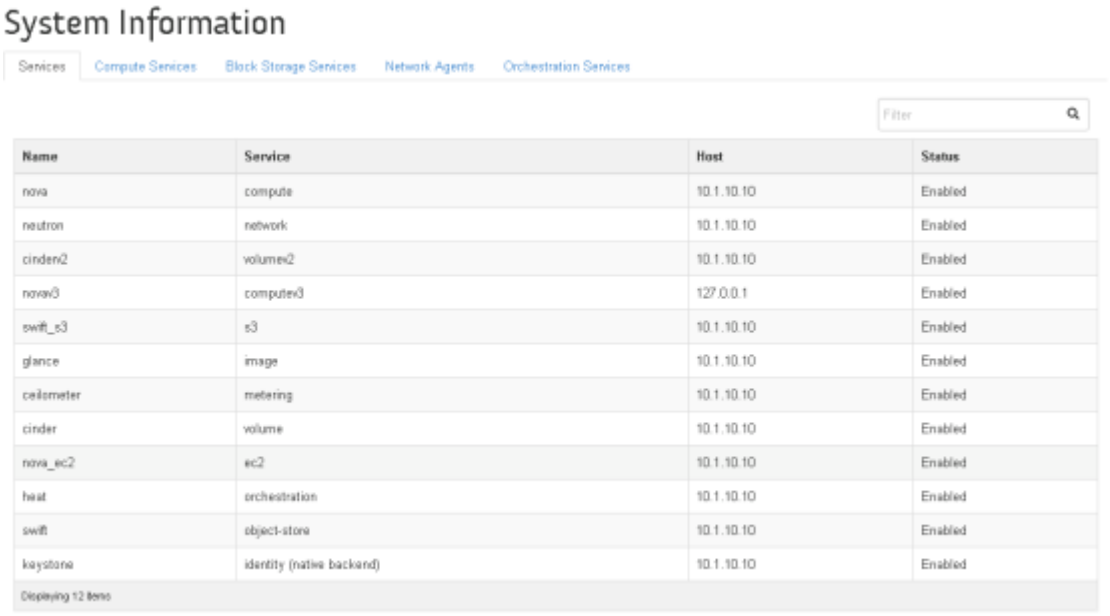
To test the GUI access, just open your Chrome browser (inside the RDP session – user/user) and you should have a bookmark for the Horizon dashboard:



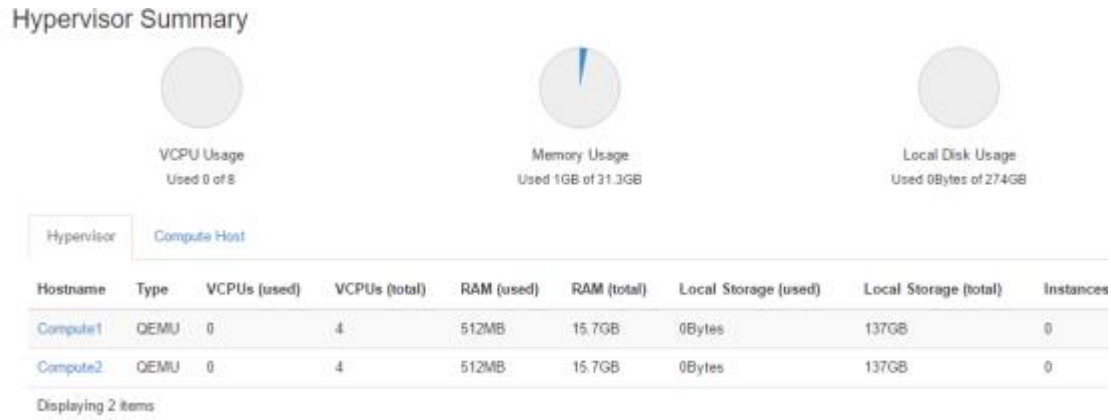
After clicking on the bookmark, you should have the following page:

Login as admin / admin (or with the new password you staged)

Go to Admin > System > System information and make sure that everything is enabled

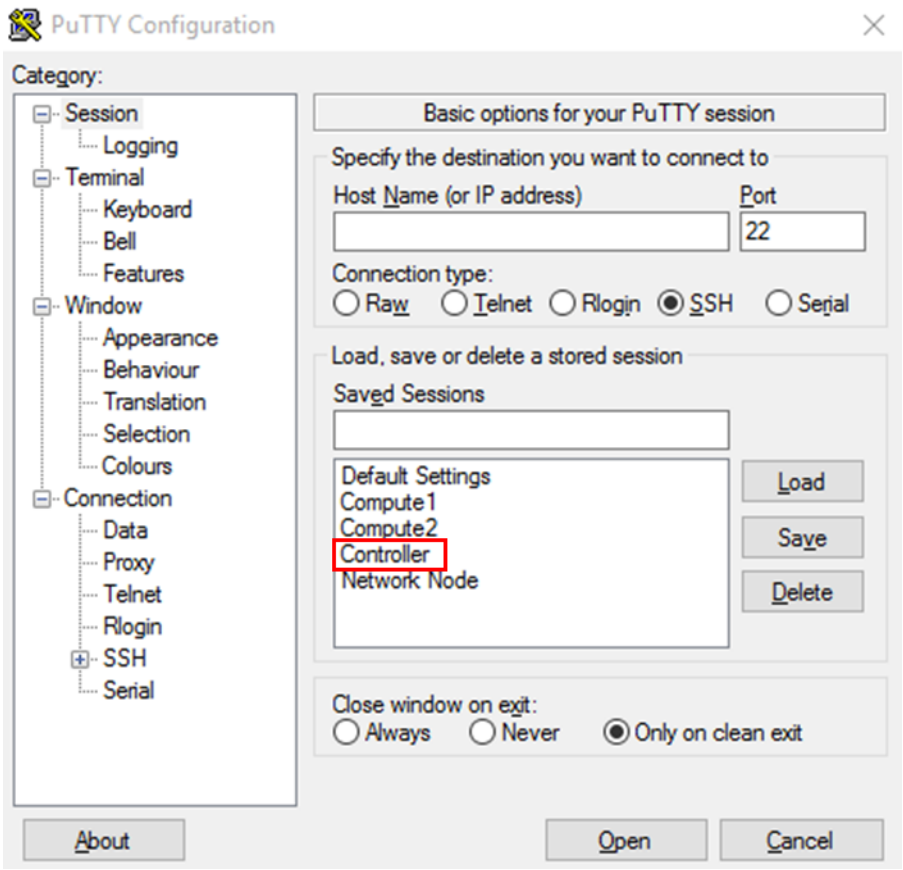


Go to Admin > System > Hypervisors and make sure you have two hypervisors defined:



Next we need to review our Neutron configuration and the states of the different agents. To do so, we will need to use CLI.

Launch Putty and connect to the session called “Controller” (or use the SSH feature in UDF):



Login: Root, password default

To get access to the OpenStack CLI client, you’ll need to authenticate yourself as an admin or a project owner. To make it easier, we can simply load some environment variables. This is the purpose of the keystonerc* files in the root home directory (/root/).

To authenticate as an admin, you can simply execute the command source and specify the keystonerc_admin file:

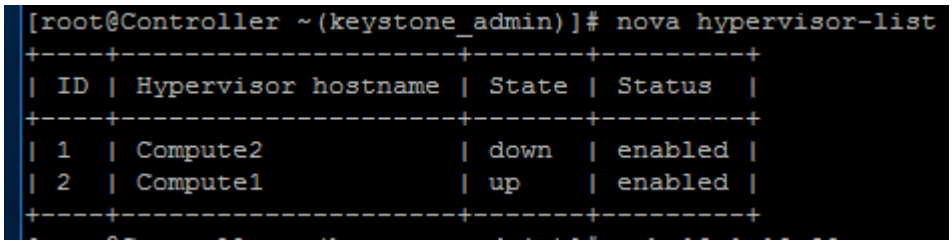
[root@Openstack-Controller ~]$ source keystonerc_admin
[root@Openstack-Controller ~(keystone_admin)]$

Now that we are authenticated, we can communicate with OpenStack via different commands:

- keystone – to manage information about tenants and users
- neutron – to manage the networking configuration , lbaas agents, etc…
- nova – to manage image, instances, flavors, …

Try to execute those commands and make sure you don't get any error messages:

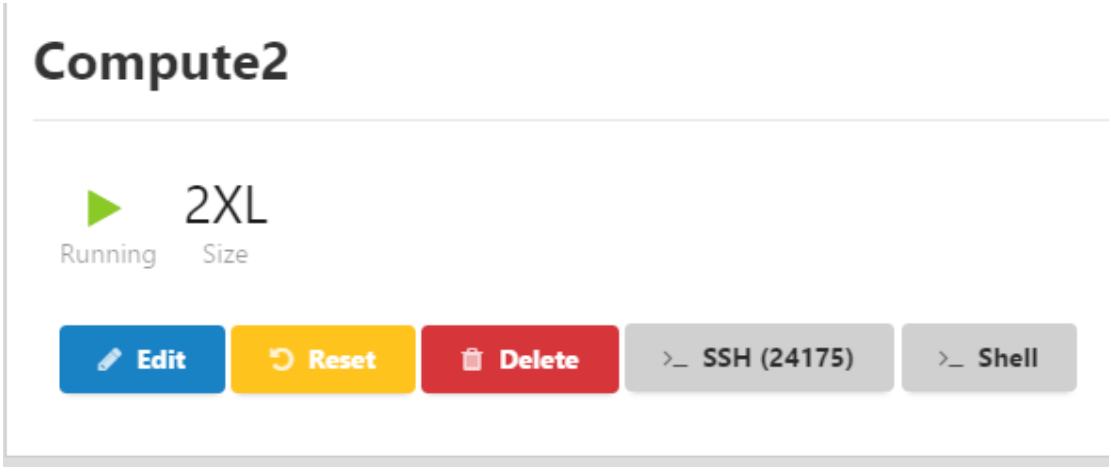
- nova hypervisor-list



If the state/status is not "up/enabled", try to ping Compute1 and Compute2 from your controller ssh session (ping Compute1 / ping Compute2). If it does not work, you'll need to use the UDF interface to restart your compute node. To do so you'll need to:

- click on the Compute1 (or 2) node. In our case, we seem to have an issue with Compute2
- click on the reset yellow button

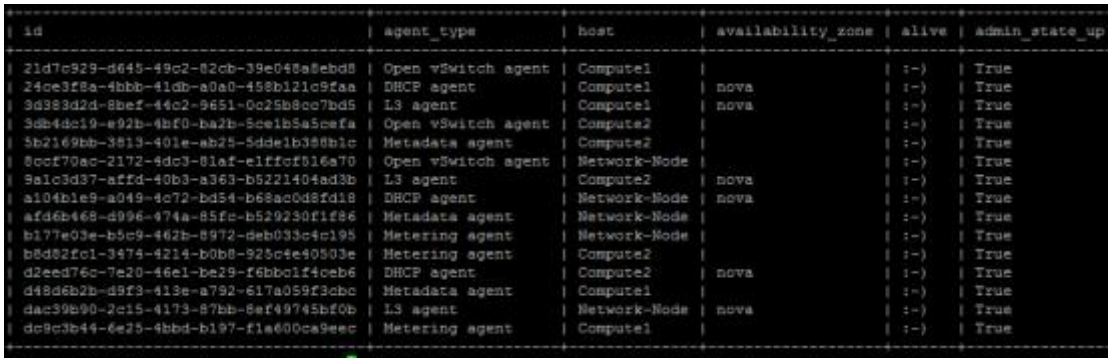
Use the console tab on the same page to make sure it started properly.



- Once your compute node restarted, check if it came back alive with the same command

```
[root@Controller ~(keystone_admin)]# nova hypervisor-list
+----+---------------------+-------+---------+
| ID | Hypervisor hostname | State | Status  |
+----+---------------------+-------+---------+
| 1  | Compute2            | up    | enabled |
| 2  | Compute1            | up    | enabled |
+----+---------------------+-------+---------+
```

- neutron help
- neutron router-list
- neutron agent-list



id	agent_type	host	availability_zone	alive	admin_state_up
21d7c929-d645-49c2-82cb-39e048a8ebd8	Open vSwitch agent	Compute1		:-)	True
24ce3f8a-4bbb-41db-a0a0-458b121c9faa	DHCP agent	Compute1	nova	:-)	True
3d383d2d-8bef-44c2-9651-0c25b8cc7bd5	L3 agent	Compute1	nova	:-)	True
3db4dc19-e92b-4bf0-ba2b-5ce1b5a5cefa	Open vSwitch agent	Compute2		:-)	True
5b2169bb-3813-401e-ab25-5dde1b388b1c	Metadata agent	Compute2		:-)	True
8ccf79ac-2172-4dc3-81af-e1ffcf516a70	Open vSwitch agent	Network-Node		:-)	True
9a1c3d37-affd-40b3-a363-b5221404ad3b	L3 agent	Compute2	nova	:-)	True
a104b1e9-a049-4c72-bd54-b68ac0d8fd18	DHCP agent	Network-Node	nova	:-)	True
afd6b468-d996-474a-85fc-b529230f1f86	Metadata agent	Network-Node		:-)	True
b177e03e-b5c9-462b-8972-deb033c4c195	Metering agent	Network-Node		:-)	True
b8d82fc1-3474-4214-b0b8-925c4e90503e	Metering agent	Compute2		:-)	True
d2eed76c-7e20-46e1-be29-f6bbc1f4ceb6	DHCP agent	Compute2	nova	:-)	True
d48d6b2b-d9f3-413e-a792-617a059f3cbc	Metadata agent	Compute1		:-)	True
dac39b90-2c15-4173-87bb-8ef49745bf0b	L3 agent	Network-Node	nova	:-)	True
dc9c3b44-6e25-4bbd-b197-f1a600ca9eec	Metering agent	Compute1		:-)	True

Use one of the neutron agent ids with the command neutron agent-show

- neutron agent-show 3db4dc19-e92b-4bf0-ba2b-5ce1b5a5cefa

Try this command with different neutron agents (DHCP, L3 agent, Metadata agent)

If any of your agents are not alive (the xxx description in the “alive column), you’ll need to access the relevant host and check if anything is wrong in the log files.

LAB0 – Licenses requirement

For our lab, you'll need BIG-IP licenses. Retrieve the following evaluation (i.e. not internal) licenses:

- 3 x F5-BIG-VE-BT-3G-V13-LIC (you need to make sure they have the SDN add-on license!)

Generate those licenses, we will use it later

LAB 1 – Understanding OpenStack Networks

Overview of the networking configuration

Here are some docs explaining how Openstack works from a networking perspective:

- https://devcentral.f5.com/d/f5-and-openstack-adc-integration-guide - this pdf explains the main networking concepts to be aware of with OpenStack (Neutron, ML2 plugin, transport network/overlay networks, how to deploy an ADC in OpenStack). If you want to understand more deeply how Neutron works, make sure to review this doc.
- https://devcentral.f5.com/d/deploying-f5-virtual-editions-on-openstack - this doc will help you setup OpenStack Neutron/ML2 plugin, your networks for setting up a F5 virtual edition on top.

Provider vs. Tenant Networks

When logged in as a non-admin user, whether through the CLI or through Horizon, when one creates a network it is inherently private and single tenant, owned by the particular project in which it was created. This provides the tenant with a wholly-owned network, instances, etc., that can be seen only by him/her. When networks are created, their network segmentation ID, be it a 802.1Q VLAN ID, VxLAN VNI, or transparent Ethernet GRE key id, is decided by Neutron and is allocated dynamically.

When logged in as admin, tenant networks can be created which are only visible to the admin tenant, but, via the CLI or the API, there is the additional option to instead create a provider network. Provider network segmentation IDs are not dynamically allocated by Neutron, but must be specified along with the type of network to be created. Provider networks can be marked as shared (or public), in which case they are available for use, and will appear to all tenants in the cloud.

If a provider network is shared (or public), any tenant can start Nova guest instances (VM or container) interfaces on these networks. It is not uncommon to

find OpenStack clouds where all networking addresses, both MAC and IP layer, are managed by the provider. In such a situation, all guest tenant connectivity would be provided by shared provider networks. Provider only network designs are often either closely coupled with the underlying physical network topology (i.e. fixed VLANs), or else used where the network virtualization technology expects to control the networking outside of Neutron control.

NOTE: network virtualization technologies which are not under the control of Neutron would not be suitable for any multi-tenant Neutron advanced services. Instead any advanced services would be integrated with the proprietary network virtualization mechanism (i.e. direct integration with a contrail controller).

If a provider network is not shared, it will only be visible to the admin tenant. Non-shared provider networks are used to allow provider service ports to be under the control of Neutron. An example where a non-shared provider network might be useful would be the use of Nova guest instances for infrastructure services, like multi-tenant virtual ADCs or caching proxies.

When Neutron Routers, a standard Neutron logical element which provides communications to Nova guest instances off their local subnet, are supported, a special type of provider network can be created. This network is marked as external (or router:external). This indicates that a Neutron router can use this provider network's subnets to SNAT outbound traffic and allocate Neutron Floating IPs for inbound NATed access from and to Nova guest instances. The obvious use of an external provider network would permit access to and from a public network which is not managed by Neutron (i.e. the Internet). External provider networks are not typically shared, thus tenants cannot typically create Nova guest interfaces on them, but require the use of a Neutron Router.

Single-tenant vs. Multi-tenant services

Single-tenant services from F5 can be thought of as ADCaaS. In this mode, the device, as well as all configuration (including networking) is the responsibility of a single OpenStack tenant. No other tenant can use the F5 services provided on that tenant's ADC. This is also known as "over the cloud" as it is typically implemented with virtual ADCs with network interfaces on tenant networks.

LBaaS, based on the community defined API is a multi-tenant service, and BIG-IP can participate as an LBaaS service provider through our LBaaS plug-in. Because of our LBaaS plug-in, all configuration of the BIG-IPs participating in this service (again, can be virtual editions or hardware) is automated. The tenant requesting LBaaS interacts with Neutron, not with the BIG-IP implementing the service. When a device implements a multi-tenant service as part of the cloud networking infrastructure, it is typically designated as "under the cloud". F5 hardware or virtual appliances can provide multi-tenant Neutron service when attached to provider networks.

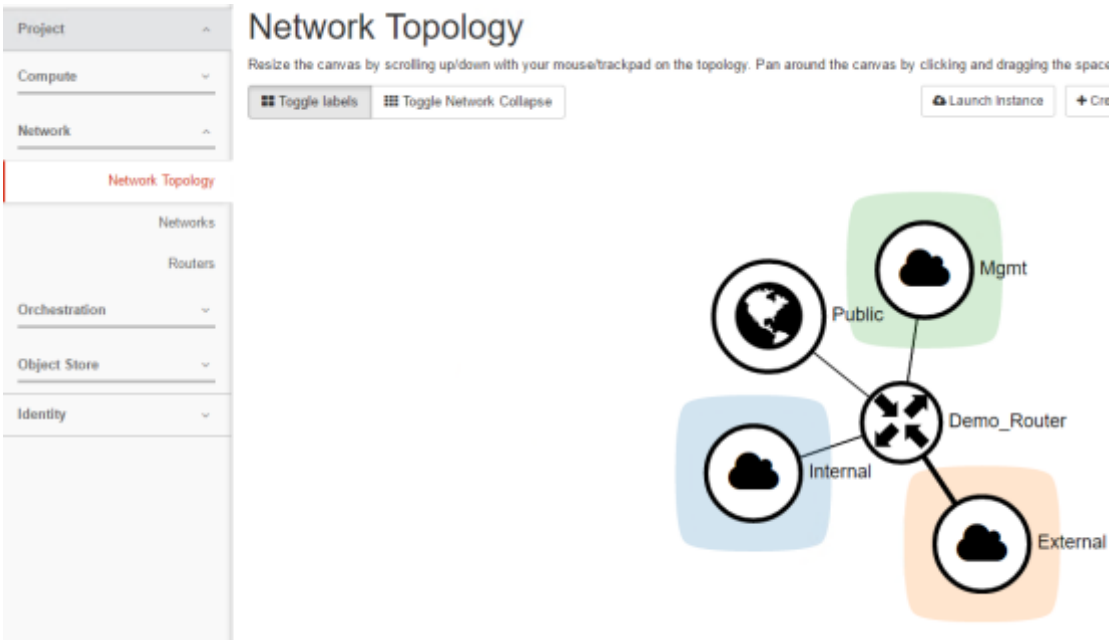
In your OpenStack environment, Neutron and ML2 plugins are already configured. If you want to review how they are setup, you should review the following files on the controller, network node, compute nodes:

- /etc/neutron/neutron.conf – define that we rely on ML2 plugin, which "service plugins" should be loaded (router, lbaas) and some other things about our network topologies
- /etc/neutron/plugin.ini (symbolic link to /etc/neutron/plugins/ml2/ml2_conf.ini since we use ML2) – define our tenant network types, tunnel ID ranges for network overlays, our local ip that will act as a vtep,..)

Here you have the lab design. However, this does not reflect what will be deployed for our OpenStack tenant (called "Project" within Horizon). For this lab, we will design specific topologies for each project: admin and demo.

Demo project – network topology

Here is the network topology that we will create through this lab:



Lab – create admin provider network

For each project other than admin, we want to make them use overlay networks (GRE tunnels). To enforce this, the following needs to be done

On the Controller:

- Setup the ML2 plugin to use GRE tunnels for tenant – this is already done in the /etc/neutron/plugin.ini file (symbolic link to /etc/neutron/plugins/ml2/ml2_conf.ini). you can review the configuration
- Define a provider network: a provider network is a network that will be used for our GRE tunnels. This is done by creating a service provider network in an admin project.

Here are the pertinent information regarding the ml2_conf.ini file

- **type_drivers = gre, local, flat** – it defines what kind of network we want to be able to create
- **tenant_network_types = gre** – This means that OpenStack projects (tenant in OpenStack) will use GRE tunnels on top of a provider network
- **mechanism_drivers = openvswitch,l2population** – specify some drivers to be loaded
- **tunnel_id_ranges = 100:1000** – GRE tunnel IDs will be between 100 and 1000
-

ML2 plugin and l2population are requirements if we want to deploy the F5 LBAaS plugin

The ML2 plugin is already setup on each OpenStack component that needs to manage VM traffic. So it's installed on:

- the network node
- the compute nodes

Each of those node has its own specific ml2 configuration. This is defined in /etc/neutron/plugins/ml2/openvswitch_agent.ini

Pertinent information in this file:

- **tunnel_types = gre**
- **l2_population = True**
- **enable_distributed_routing = true** – explain that we want to use neutron DVR (Distributed Virtual Router)
- **local_ip = 10.1.20.30**- this define the local IP that will be used as a VTEP for GRE tunnels. This is the only information that should differ on all your OpenStack nodes
- **bridge_mappings = physnet:br-data** – this defines a LABEL called physnet. This LABEL specifies which interface should be used when using

it. This is needed to define our provider network. br-data is an interface defined on your component. The IP assigned to this interface should be the IP assigned to the variable local_ip.

- **enable_tunneling = True**

Here is how we will define our provider network:

1. Identify the LABEL used to identify the interface that will be use to connect the VM interface to. Based on our /etc/neutron/plugin.ini file (on the controller), the LABEL is physnet

2. Identify our admin project ID. In CLI (on the controller), after you source the /root/keystonerc_admin file, type the command "***keystone tenant-list***" and retrieve the admin project ID.

Example:

```
[root@Openstack-Controller neutron(keystone_admin)]# keystone
tenant-list
 /usr/lib/python2.7/site-packages/keystoneclient/shell.py:65:
DeprecationWarning: The keystone CLI is deprecated in favor of
python-openstackclient. For a Python library, continue using
python-keystoneclient.
   'python-keystoneclient.', DeprecationWarning)
+----------------------------------+----------+---------+
|                id                |   name   | enabled |
+----------------------------------+----------+---------+
| 0b02c277107a4c318c72ed7f60227413 |  admin   |   True  |
| c6b1e874e0f04b8e9904311bb74d1221 |   demo   |   True  |
| 1fe175c56c304bccb19b2c8c0b91abd2 | services |   True  |
+----------------------------------+----------+---------+
[root@Openstack-Controller neutron(keystone_admin)]#
```

3. Type the following commands:

```
neutron net-create 'Underlay Network' --tenant_id <TENANT_ID> --
provider:network_type flat --provider:physical_network
<PHYS_NET_NAME LABEL>
```

Example:

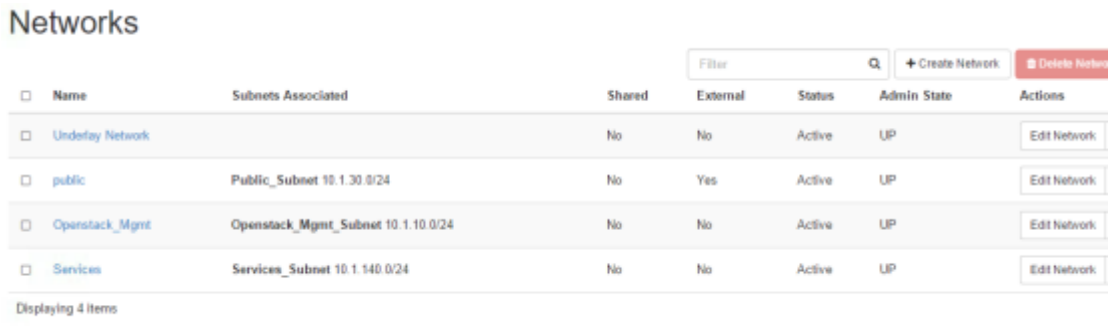
```
neutron net-create 'Underlay Network' --tenant_id
0b02c277107a4c318c72ed7f60227413 --provider:network_type flat --
provider:physical_network physnet
```

You should see something like this:

```
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        |                                      |
| created_at                | 2016-07-07T07:30:35                  |
| description               |                                      |
| id                        | 307eb92f-294b-44d8-ad8c-c0c04b39d065 |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1500                                 |
| name                      | Underlay Network                     |
| provider:network_type     | flat                                 |
| provider:physical_network | physnet                              |
| provider:segmentation_id  |                                      |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tags                      |                                      |
| tenant_id                 | 0b02c277107a4c318c72ed7f60227413     |
| updated_at                | 2016-07-07T07:30:35                  |
+---------------------------+--------------------------------------+
```

Note that it is also possible to do the same via the GUI.

Access the horizon dashboard, log in as admin and go to System > Networks, you should see the following:



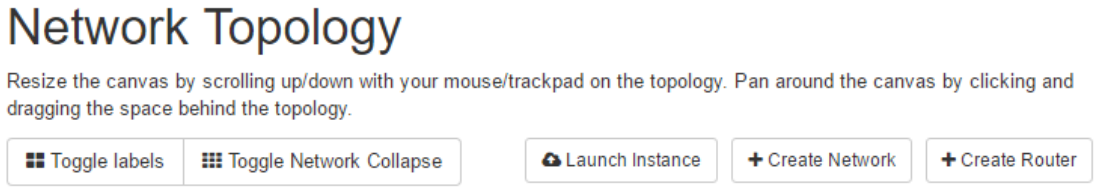
Public, Services and Openstack_Mgmt have been already built. Public is to give our Openstack infrastructure access to the external network and the others will be used for the HEAT Lab.

Lab – create demo project networks

The demo project is empty from a networking perspective and networks will have to be created to deploy instances.

For this exercise we will create our networks via the GUI. **Access the OpenStack dashboard and login as demo (password demo - except if you changed it).** If you are still logged as admin, you have to sign out first (top right corner, click on admin and sign out).

Go to Project > Network > Network Topology. You should see something like this:



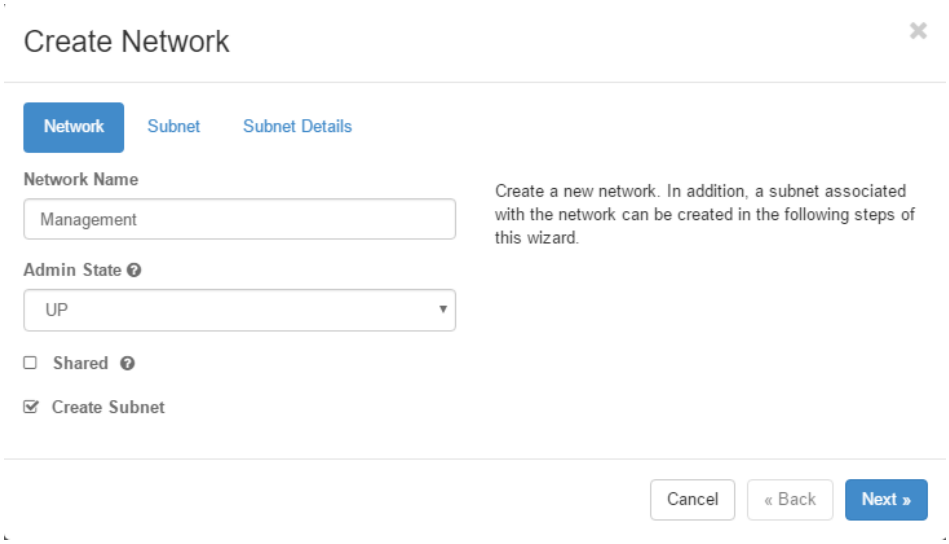
The public network is a network created in the admin project to give access to the outside world. The demo project will be able to connect to this network to have access to internet or external networks to the OpenStack environment.

For this project, we will define a few networks:

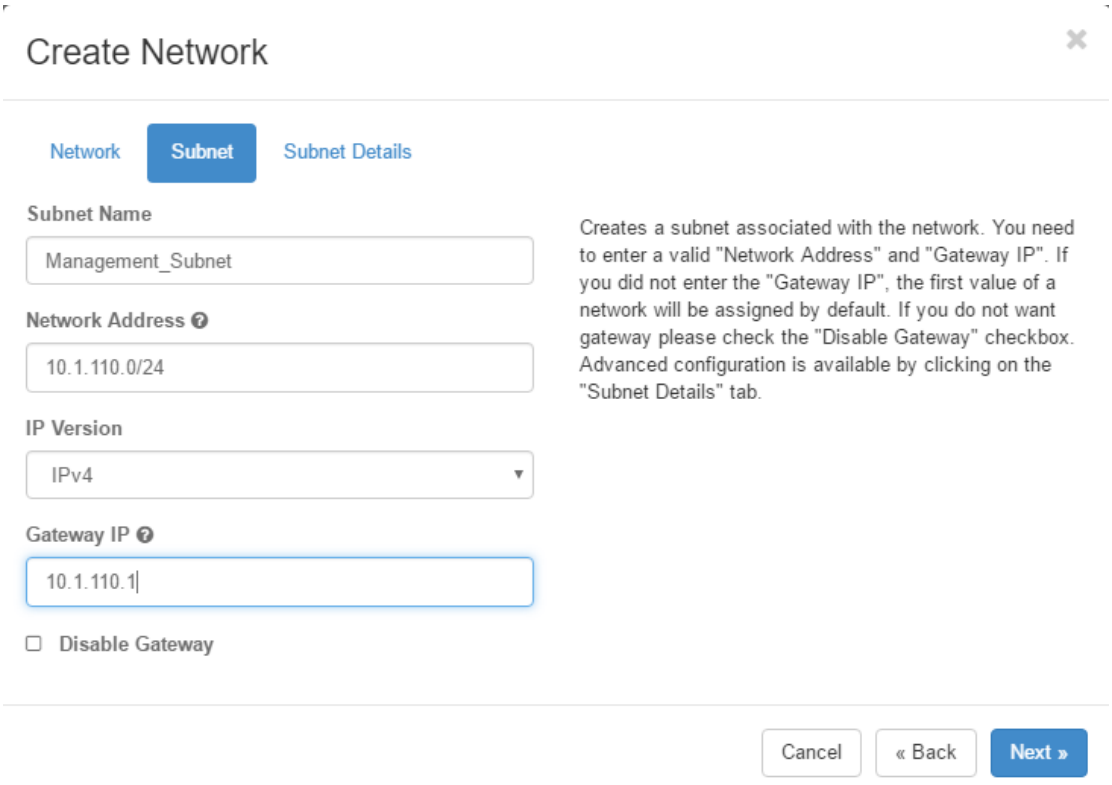
- Management: 10.1.110.0/24
- External: 10.1.120.0/24

On this page, click on the "+Create Network" (top right of the page).

1. We define the network name: Management. Click Next.

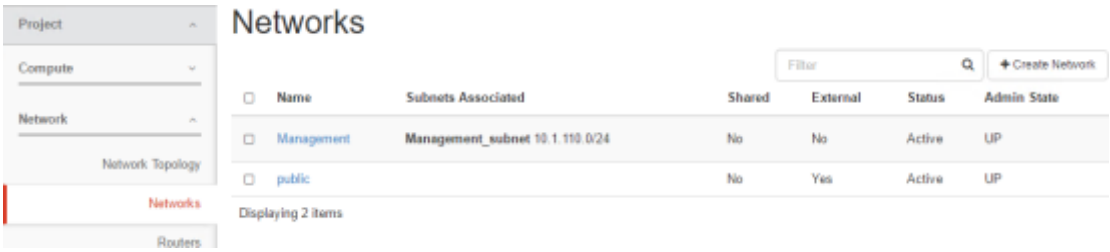


2. Define the subnet related to this network: 10.1.110.0/24 and define the GW to be 10.1.110.1

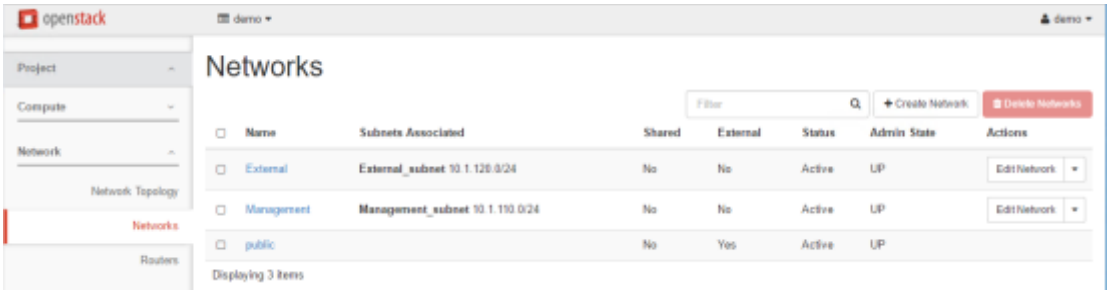


3. We define the IP range used by the subnet to provide IP to instances via DHCP.
   - Allocation Pools: 10.1.110.100,10.1.110.200
   - DNS: 8.8.4.4

4. Once you're done, click on Create. You should see something like in Network > Networks:



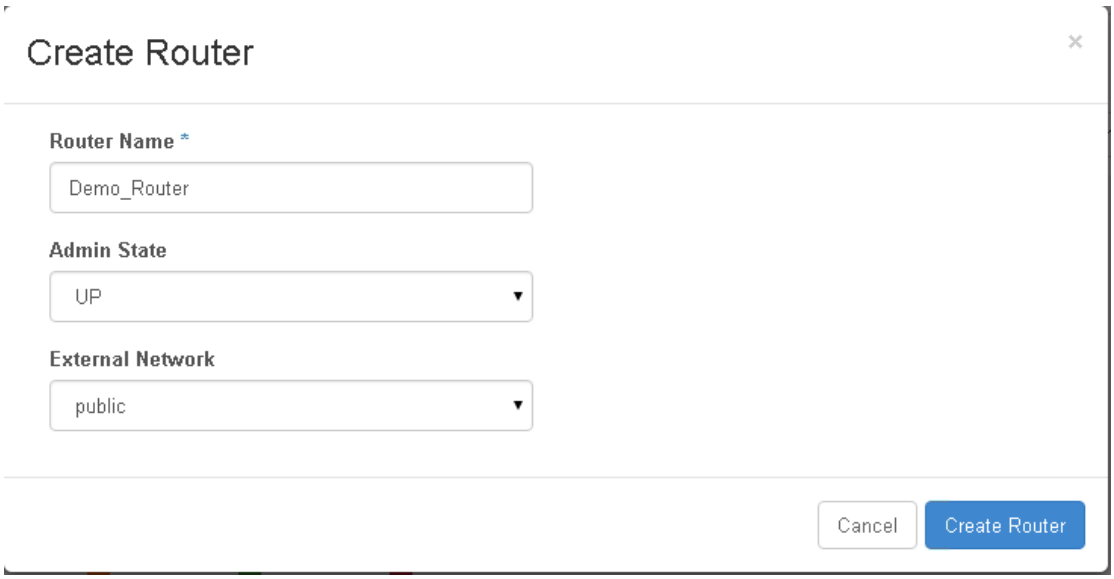
5. Repeat the same process to create the external network:
   - External: 10.1.120.0/24 with allocation pool 10.1.120.100,10.1.120.200, a gateway set to 10.1.120.1
6. You should have something like this:



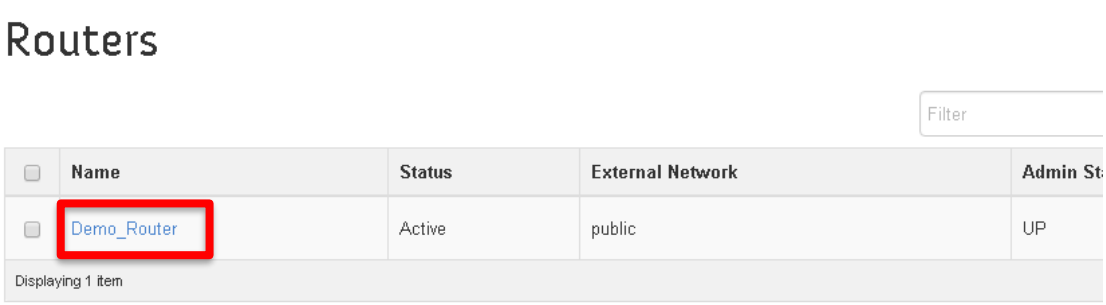
Now that we have defined our networks, we need to create a router to provide external access to this environment. We will create a router that will have access to the public network, the external network and the management network (to facilitate things, not a best practice). Go to Network > Routers.

On this page click on "+Create Router":

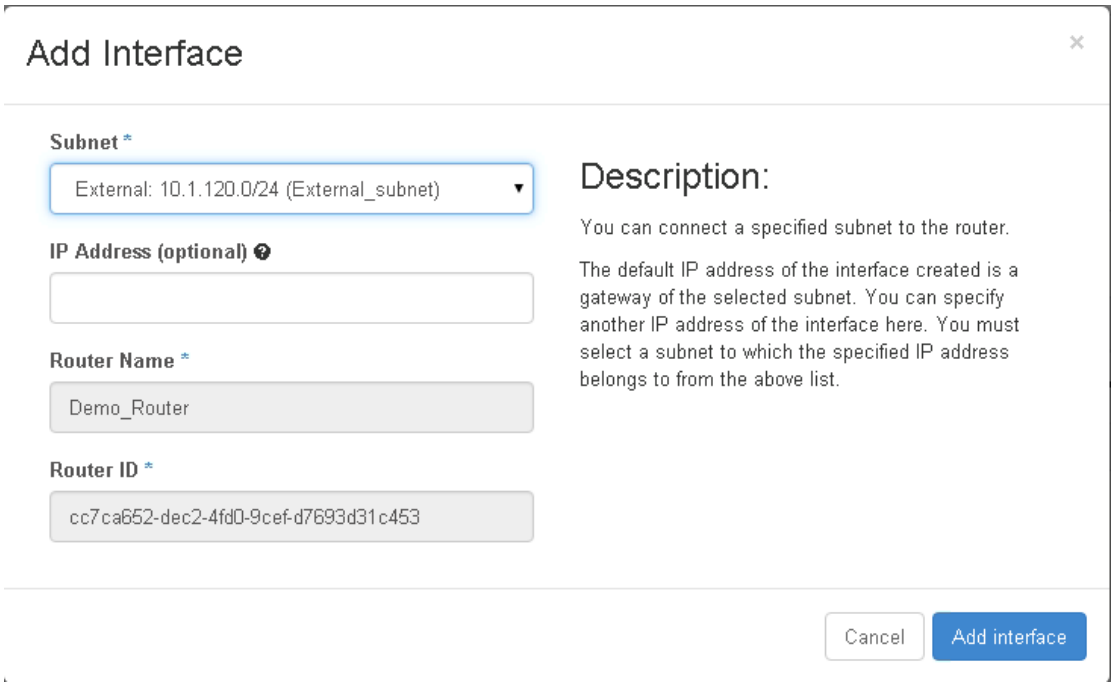
7. Specify your router's name: "Demo_Router" and select "public" as the external network



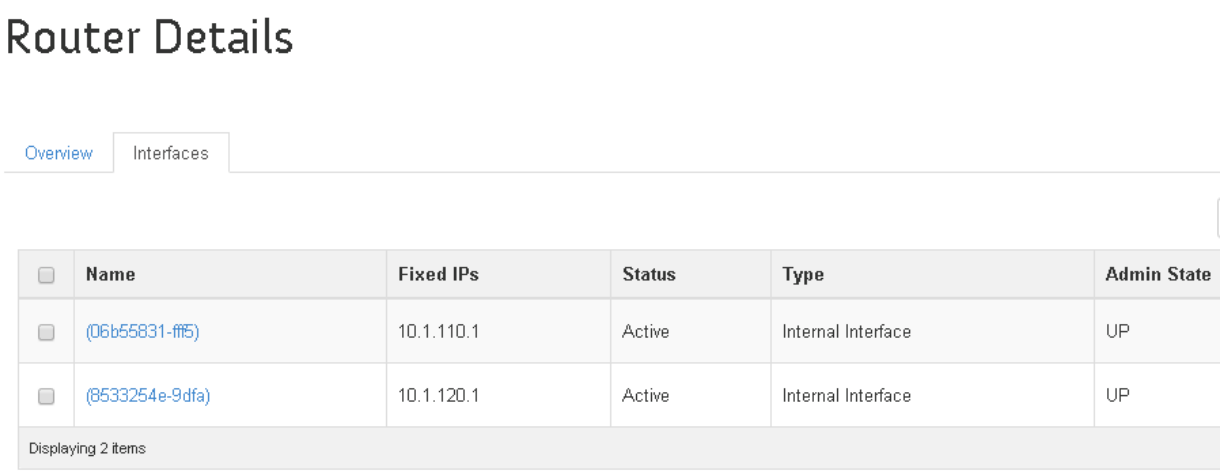
8. Go to Routers > click on your router "Demo_Router"



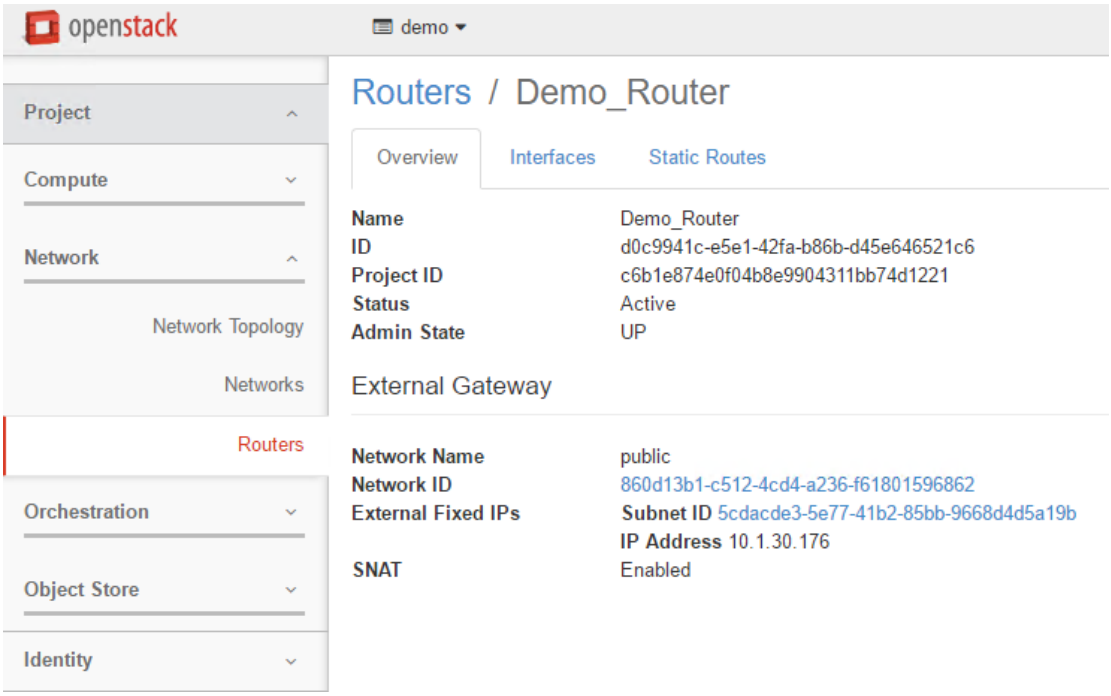
9. Click on the "Interfaces" tab and "+Add Interface"
10. Subnet: Select the external network and click on Add interface. It's not mandatory to specify the Gateway IP, if you specify it, it has to be 10.1.120.1 (the GW we specified in the subnet)



11. Repeat the previous step to add the management network to the router.



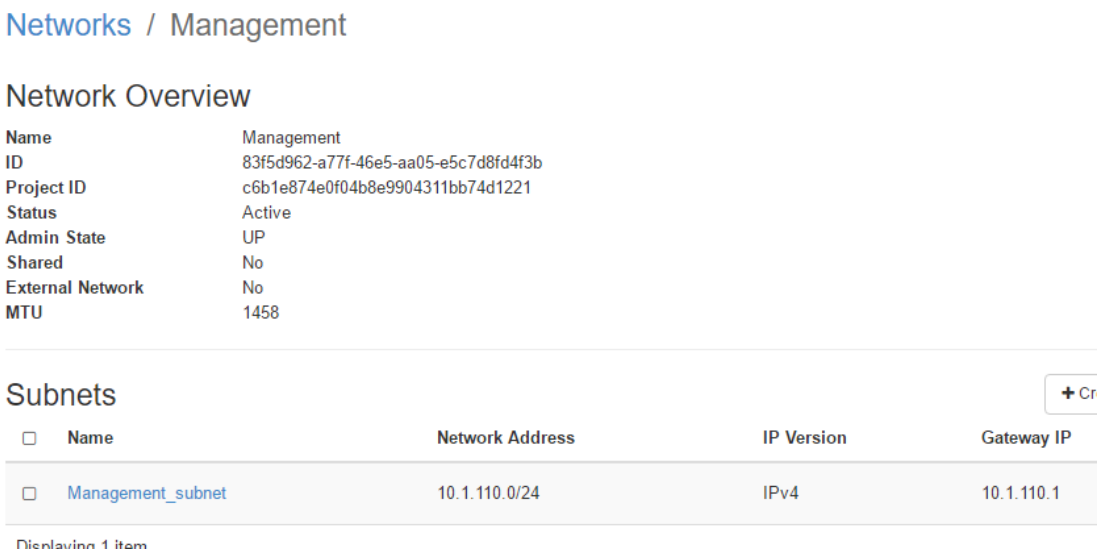
Click on the Overview tab. You should see an IP address in the 10.1.20.X subnet.



You can ping this IP address from your windows client. It should work.

Go to Project > Network > Networks and click on the "Management" network.

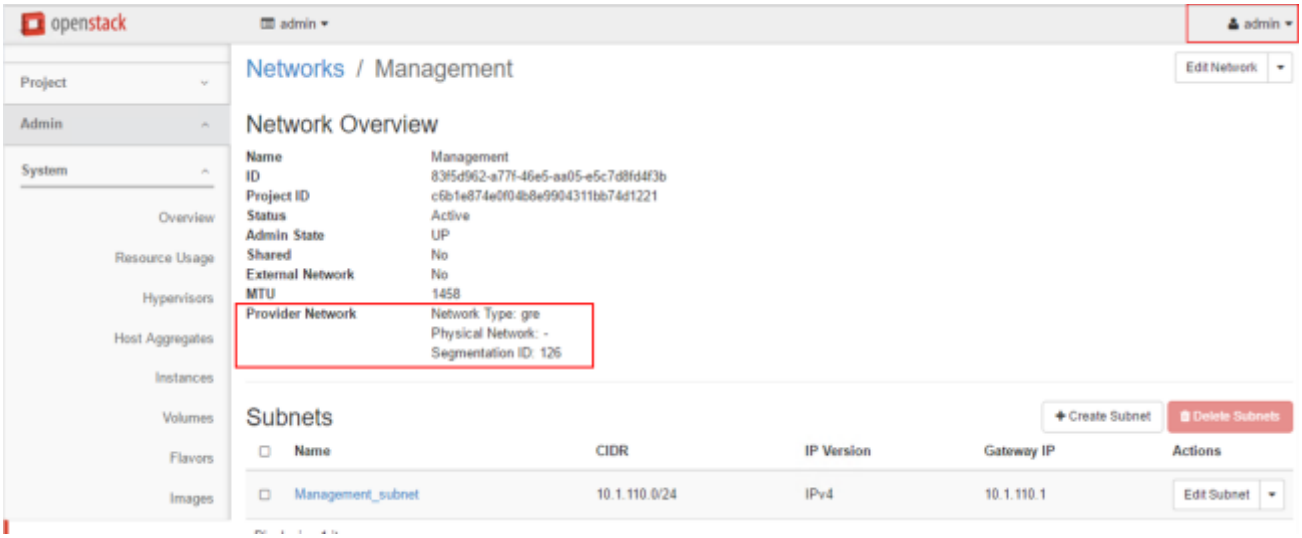
Check the "Network Overview" information. As you may see, there is nothing mentioning whether it's a network overlay or not.



If you want to learn this information, you'll have to log in as admin and review the same networks:

- Sign out (top right corner)

- Go to Admin > System > Networks. Here you'll have a summary of all the networks defined inside your OpenStack environment. Click on "Management" and review the network overview information:



Here we can see that the Management network is a GRE tunnel and its ID is 126 (can be another ID value for you).

You can try also in CLI (on your Controller).

As demo user:

1. `source /root/keystonerc_demo`
2. `neutron net-list`
3. retrieve your management network ID
4. `neutron net-show <management network ID>`

```
[root@Controller neutron(keystone_demo)]# source /root/keystonerc_demo
[root@Controller neutron(keystone_demo)]# neutron net-list
+--------------------------------------+------------+------------------------------------------------------+
| id                                   | name       | subnets                                              |
+--------------------------------------+------------+------------------------------------------------------+
| 860d13b1-c512-4cd4-a236-f61801596862 | public     | 5cdacde3-5e77-41b2-85bb-9668d4d5a19b                 |
| 816a00f5-9c02-4352-a1da-4f7c399971e7 | External   | f4db360b-6512-4e93-b418-9b21b51a996e 10.1.120.0/24 |
| 83f5d962-a77f-46e5-aa05-e5c7d8fd4f3b | Management | 655bbfad-944b-4696-b80a-b508c876713d 10.1.110.0/24 |
+--------------------------------------+------------+------------------------------------------------------+
[root@Controller neutron(keystone_demo)]#
```

```
[root@Controller neutron(keystone_demo)]# neutron net-show 83f5d962-a77f-46e5-aa05-e5c7d8fd4f3b
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        | nova                                 |
| created_at                | 2016-06-22T15:53:26                  |
| description               |                                      |
| id                        | 83f5d962-a77f-46e5-aa05-e5c7d8fd4f3b |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1458                                 |
| name                      | Management                           |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 655bbfad-944b-4696-b80a-b508c876713d |
| tags                      |                                      |
| tenant_id                 | c6b1e874e0f04b8e9904311bb74d1221     |
| updated_at                | 2016-06-22T15:53:26                  |
+---------------------------+--------------------------------------+
[root@Controller neutron(keystone_demo)]#
```

As admin:

1. `source /root/keystonerc_admin`
2. `neutron net-show <management network ID>`

```
-[root@Controller neutron(keystone_admin)]# neutron net-show 83f5d962-a77f-46e5-aa05-e5c7d8fd4f3b
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| availability_zone_hints   |                                      |
| availability_zones        | nova                                 |
| created_at                | 2016-06-22T15:53:26                  |
| description               |                                      |
| id                        | 83f5d962-a77f-46e5-aa05-e5c7d8fd4f3b |
| ipv4_address_scope        |                                      |
| ipv6_address_scope        |                                      |
| mtu                       | 1458                                 |
| name                      | Management                           |
| provider:network_type     | gre                                  |
| provider:physical_network |                                      |
| provider:segmentation_id  | 126                                  |
| router:external           | False                                |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 655bbfad-944b-4696-b80a-b508c876713d |
| tags                      |                                      |
| tenant_id                 | c6b1e874e0f04b8e9904311bb74d1221     |
| updated_at                | 2016-06-22T15:53:26                  |
+---------------------------+--------------------------------------+
```

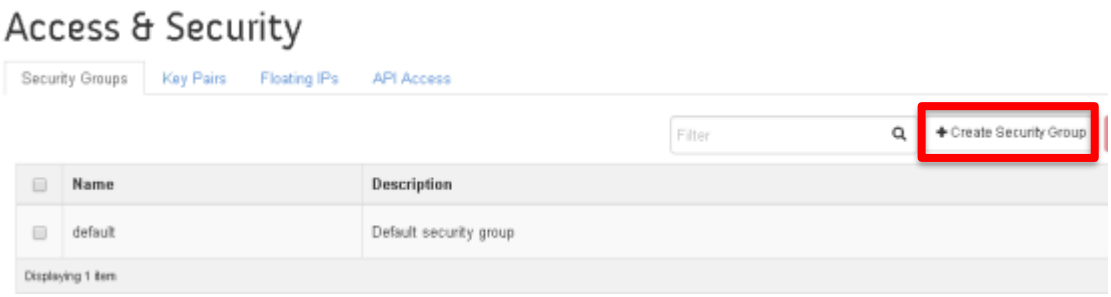
Deploy instances and test connectivity

Lab – Setup a security groups policy

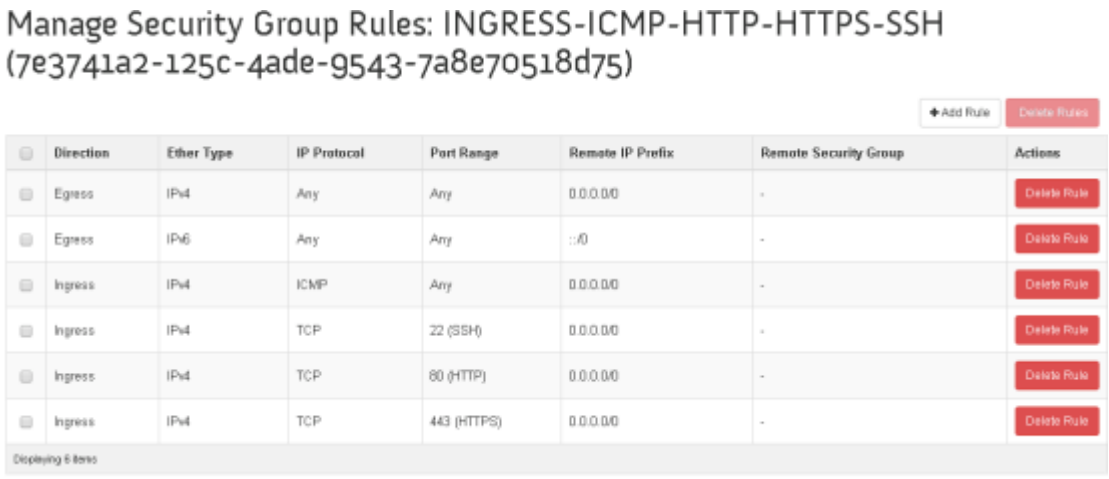
When we deploy instances in OpenStack, we need to assign a security group to each instance. This security group will specify security rules for ingress and egress traffic related to this instance.

There is a default security group, which authorizes everything. We will create something more specific.

1. **Log in the demo project (demo / demo) via the OpenStack dashboard**
2. Go to Compute > Access & Security and click on "+Create Security Group"



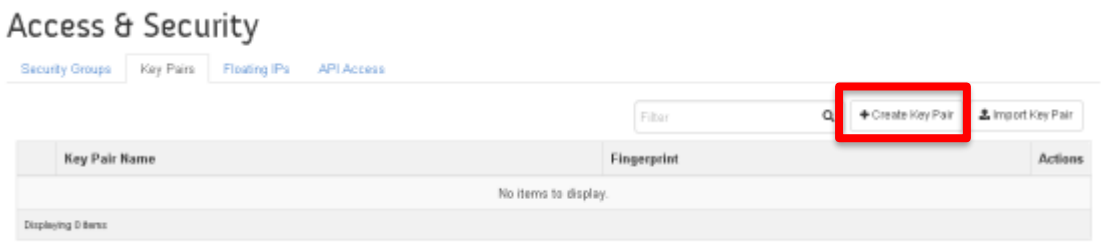
3. Name: INGRESS-ICMP-HTTP-HTTPS-SSH
4. Click on "manage rules" for this new policy and create the following rules (Click on "+add Rule")



Lab – Create a Key pair

To access your instances, you have the option to create a key pair that will be communicated to your instance when it's provisioned (via the metadata service).

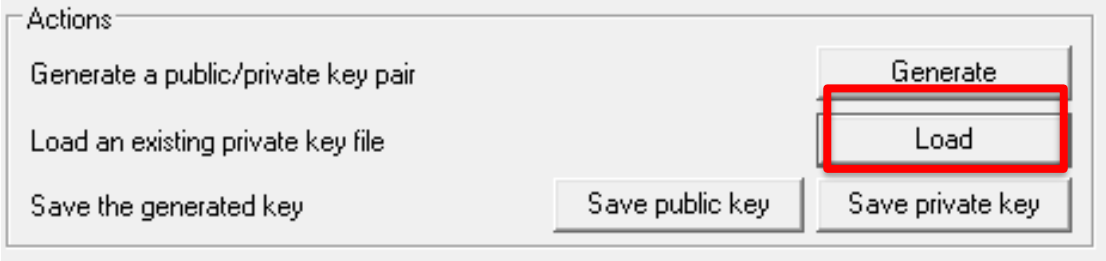
To create a Key Pair, Go to Project > Compute > Access & Security and click on the "Key Pairs" tab. Click on "+Create Key Pair"



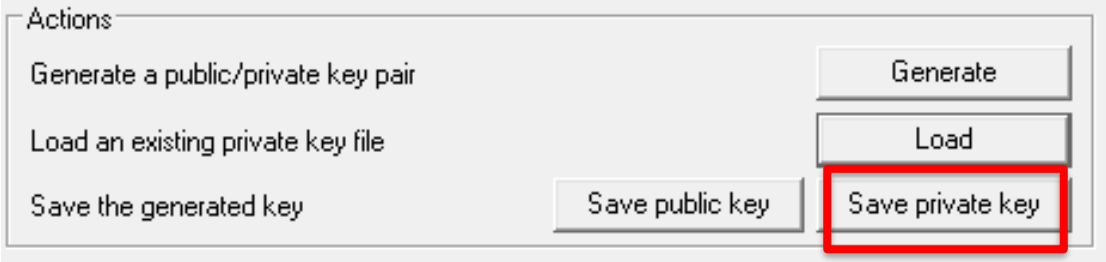
Specify the key pair name: DemoKeyPair and click on "Create Key Pair". The key will get automatically downloaded. It is in your "Downloads" folder.

The key you downloaded is in the PEM format which is not compatible with putty. You'll need to transform it. You may do it with the puttygen program (it's on your desktop).

1. Launch puttygen (On your desktop in the putty folder) and click on Load.



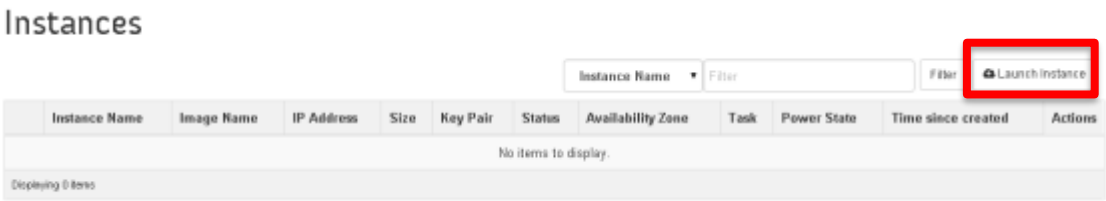
2. Select your pem key in the downloads folder (you need to select to show all types of file)
3. Once your key is loaded click on "save private key" to store it in a ppk format. Save it on your desktop.



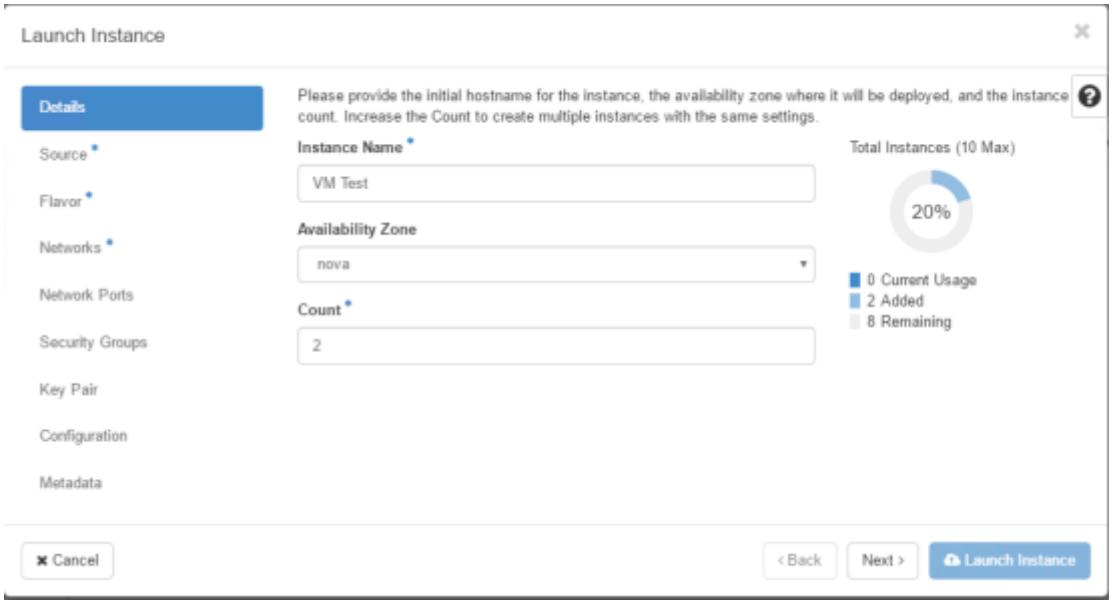
Lab – Deploy instances in the demo project

To test the networks, we will create two instances and see if they can speak to each other.

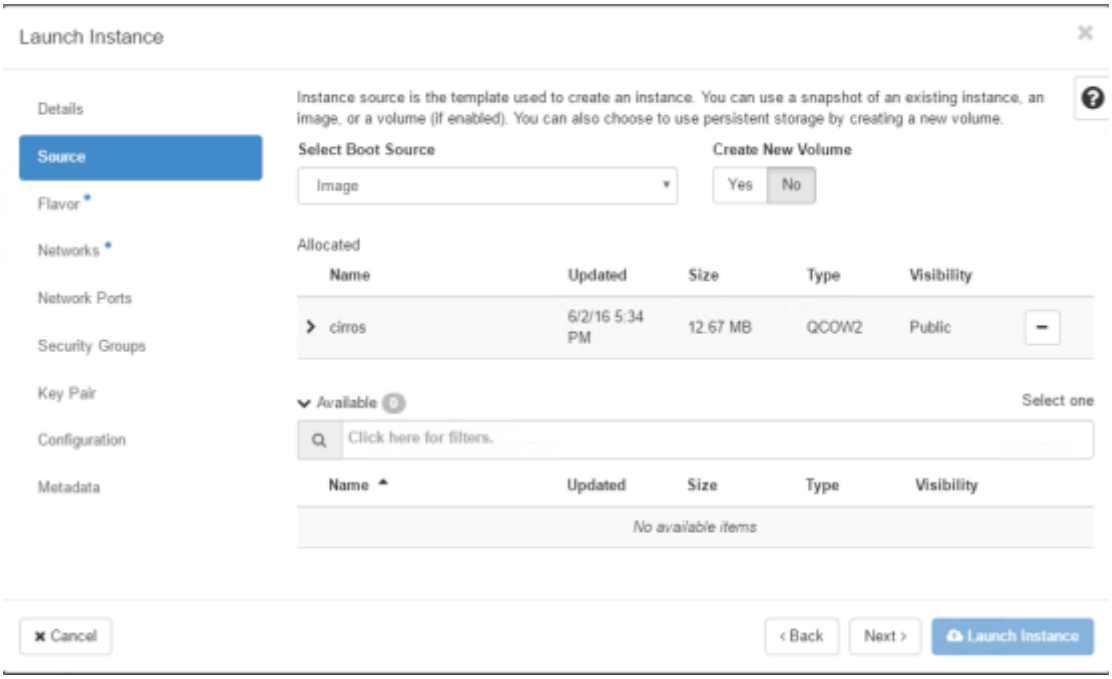
- Go to Project > Compute > Instances and click on "Launch instance" (you must be logged in as Demo)



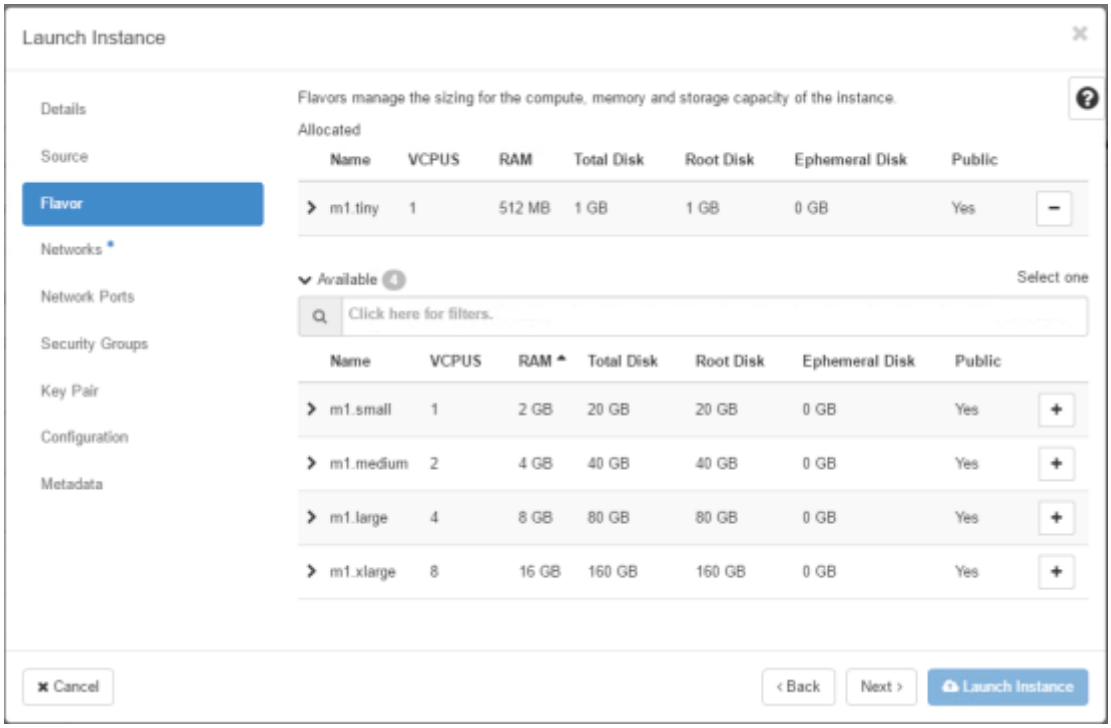
- Specify the following:
  - Availability Zone: nova
  - Instance Name : VM Test
  - Instance Count: 2



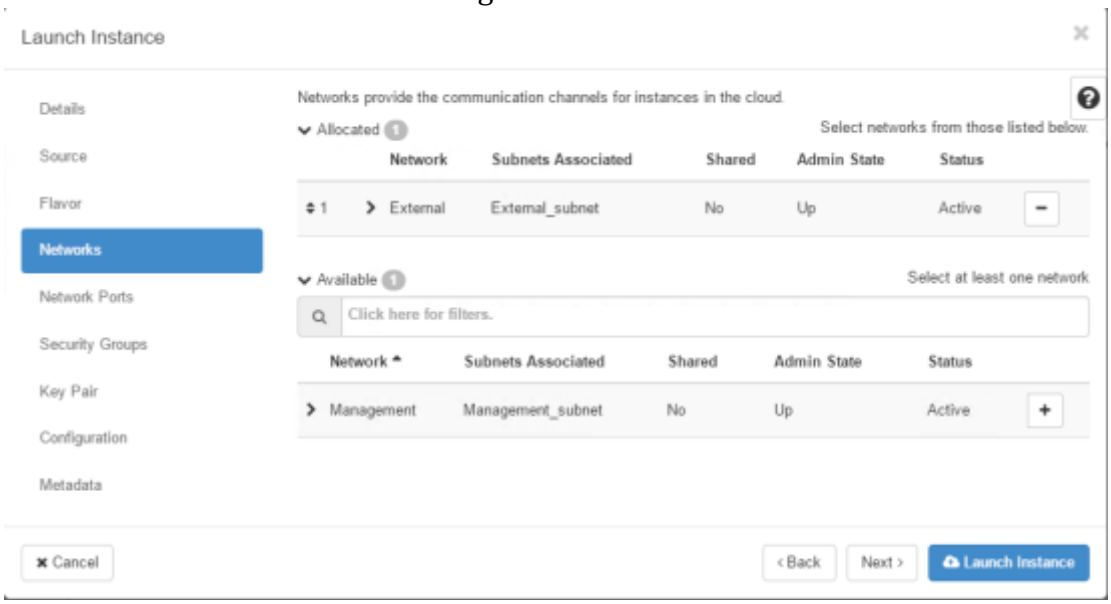
- CLICK Next
- Boot source: Boot from Image
- Image Name: cirros, click on the "+"



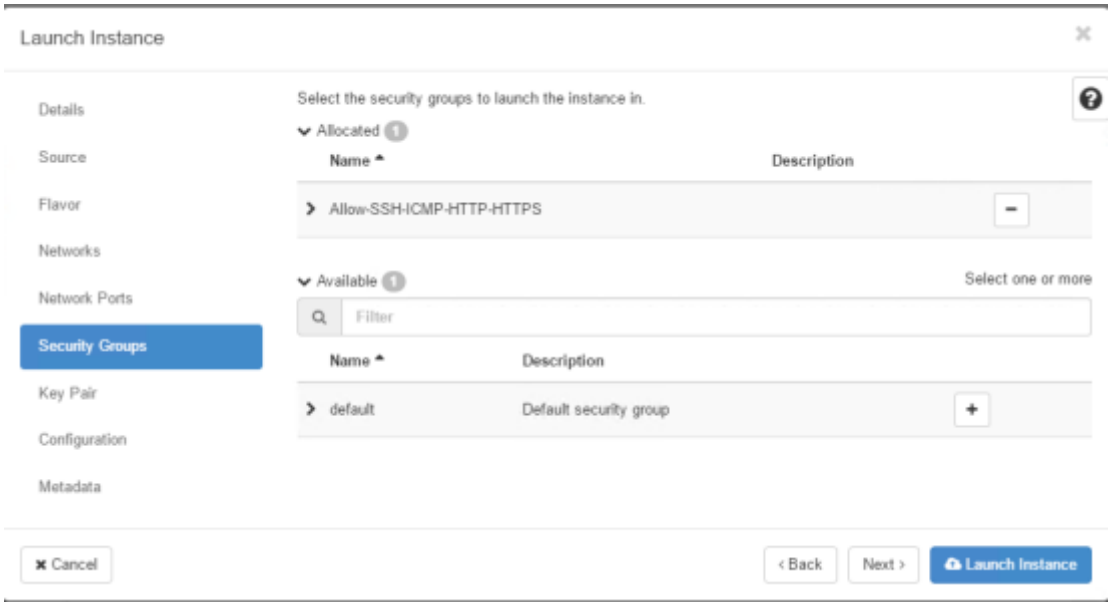
- CLICK Next
- Flavor: m1.tiny (this specifies the resources allocated to the VM – this is defined in the admin environment – we will review this later) / Click on the "+"



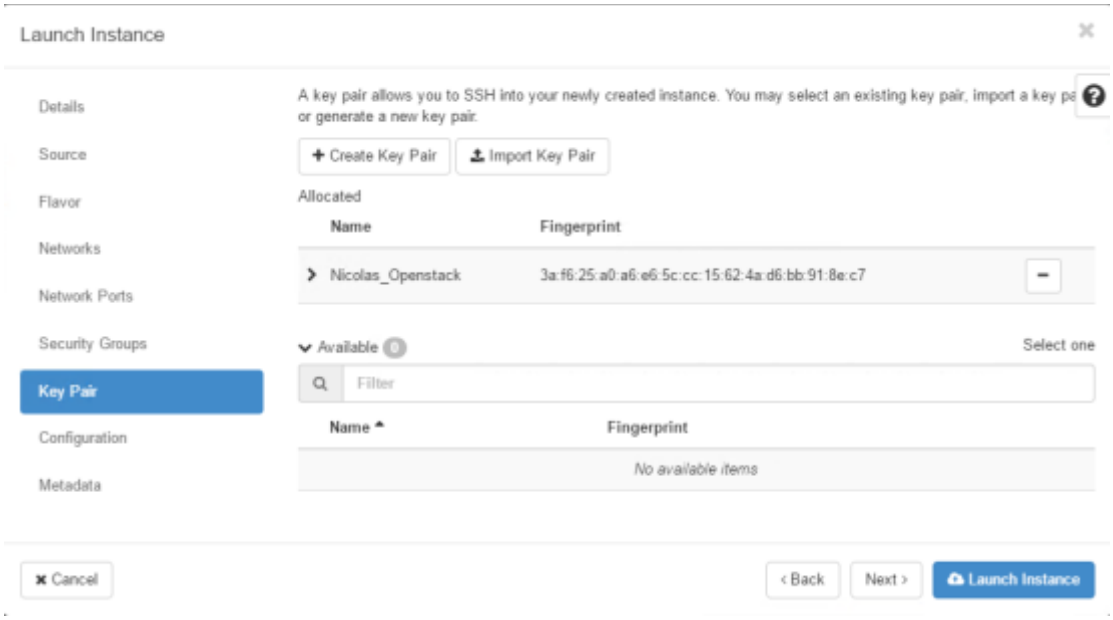
- CLICK Next
- Click on the "+" on the right of the external network to select it



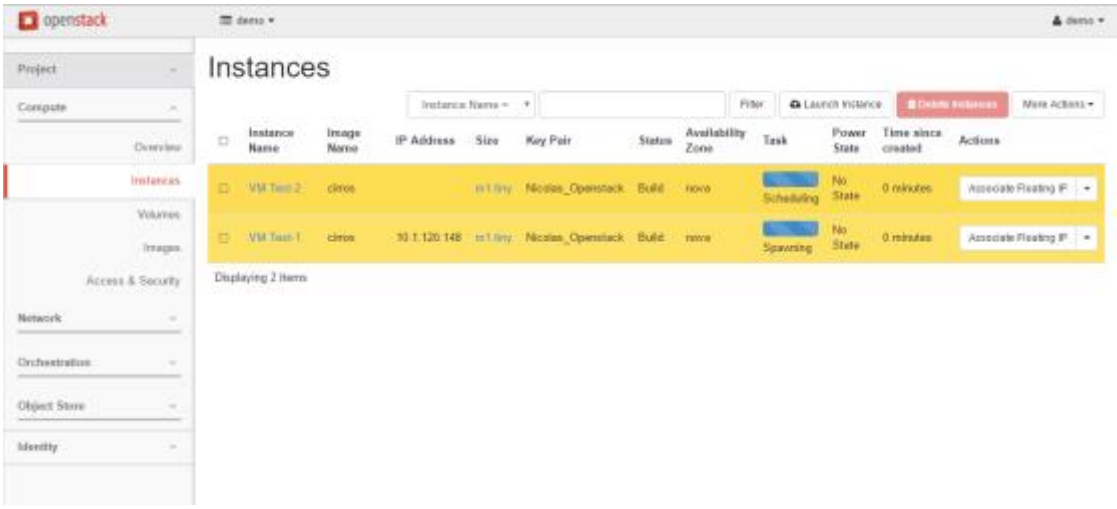
- CLICK on Next
- Click on Next without doing anything with the Network Ports interface
- Select the security groups you created previously



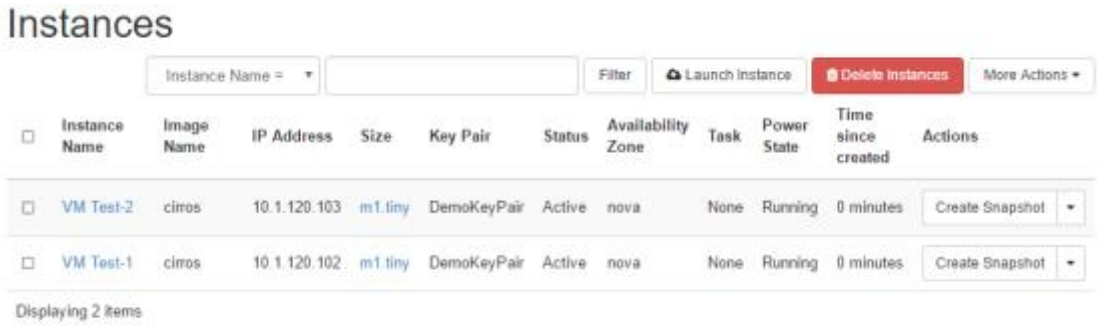
- Click Next
- Select the Key you created previously



- Click on “Launch instance. You should see two instances getting provisioned.



- Once it's finalized, you should see "Running" as a Power State and IP addresses assigned to each interface.



Lab – Access instances from the public network

You have two options to access your instances that are connected to the router:

- You route traffic through the router by adding a network route to your windows client:

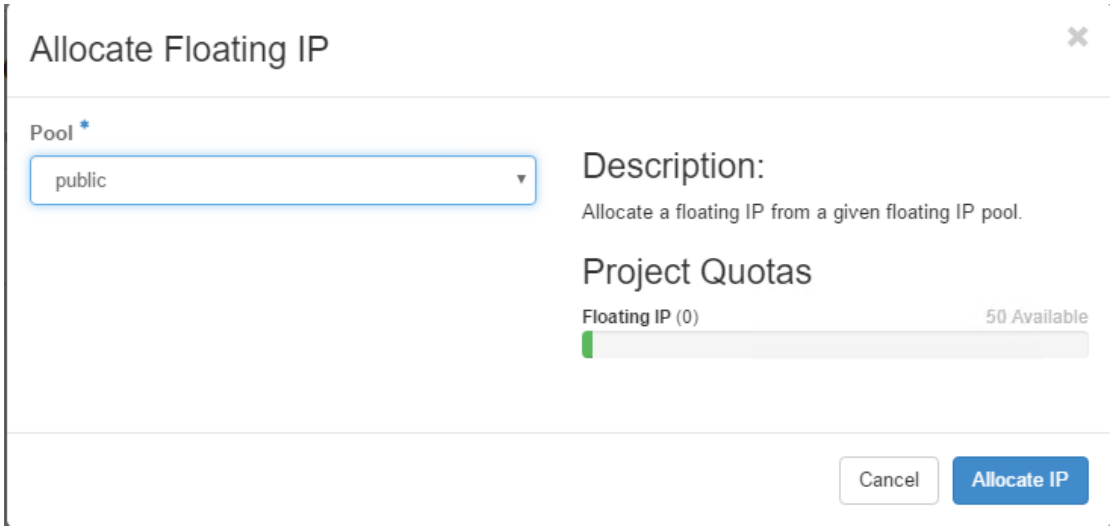
  Example `route add 10.1.120.0 MASK 255.255.255.0 10.1.30.176`

  In this example, `10.1.30.176` is the IP that was assigned to the router in the public network (see previous lab – your IP may be different). As soon as you add this route, you should be able to connect to your instances (ICMP, SSH – login "cirros" / password:" cubswin:)")

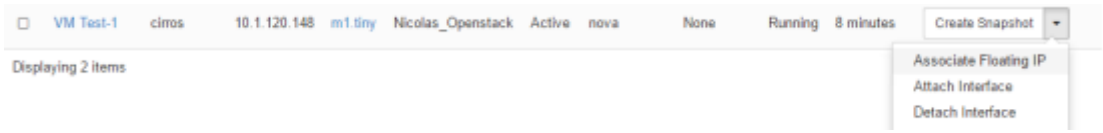
- You assign "public IPs" to your VM. To do so you need to follow this process:
  - Request some "Floating IPs" (IP from your public network)
  - Assign a Floating IP to the instance that should be accessible from the outside

To assign a Floating IP to an instance:

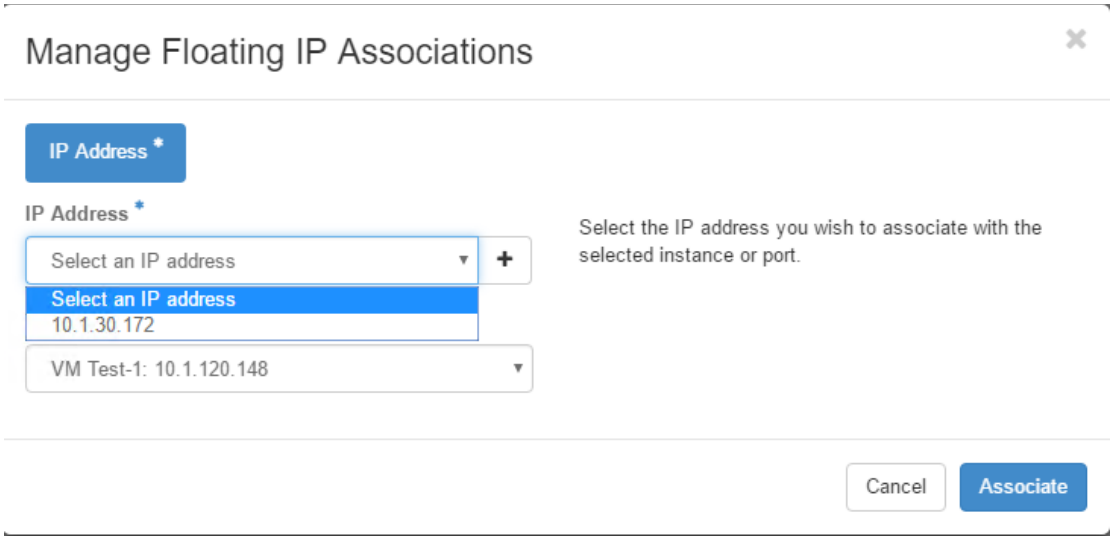
1. Go to Project > Compute > Access & Security > "Floating IPs" tab
2. Click "Allocate IP To Project" and click on "Allocate IP"



3. Go to Project > Compute > Instances
4. For VM Test-1, click on the down arrow and click on "Associate Floating IP"



5. Select the IP address that was provided previously as a Floating IP and Select VM Test-1 (The system offer only interfaces that are in a subnet connected to the router) and click on "Associate".



6. If the assignment was successful you should have a new IP address (classified as Floating IP) for VM Test-1

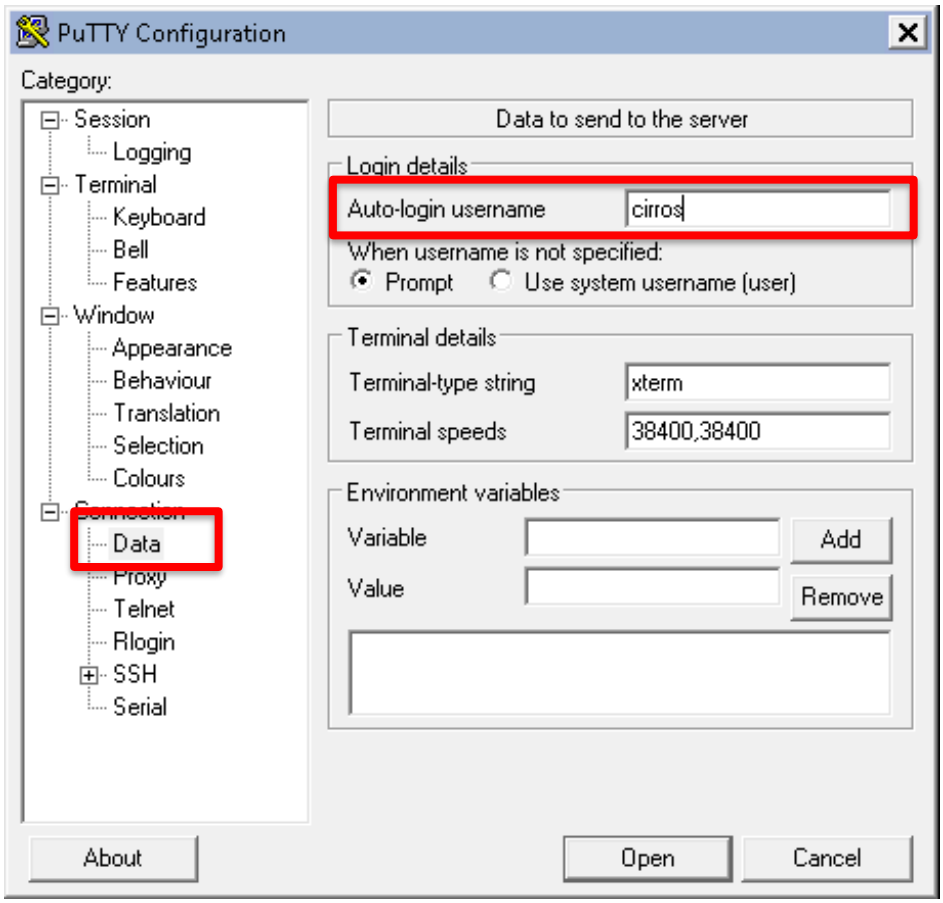
VM Test-1	cirros	10.1.120.148 Floating IPs: 10.1.30.172	m1.tiny	Nicolas_Openstack	Active	nova

We can access VM Test-1 from this public IP:

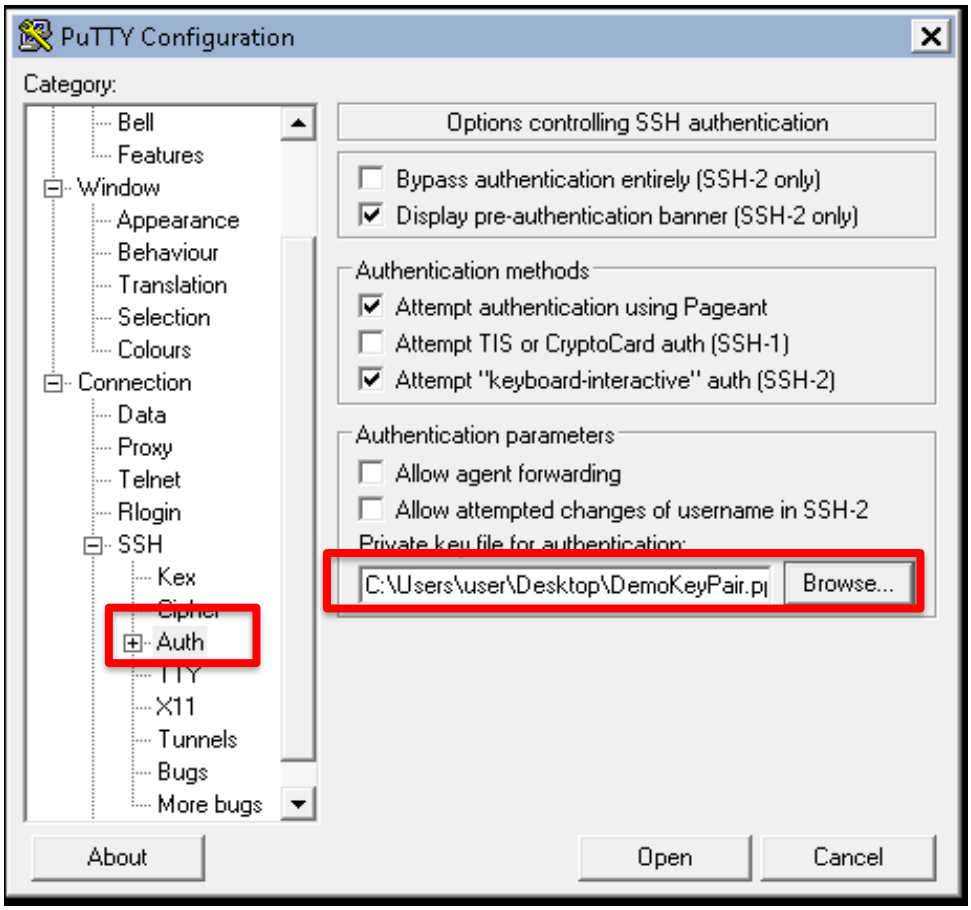
- Try to ping the Floating IP (here it would be 10.1.30.172)
- SSH with our Key. When we created our instances, we assigned our key to it. We should be able to use it to bypass the authentication process.

Here is the process to SSH with our keypair:

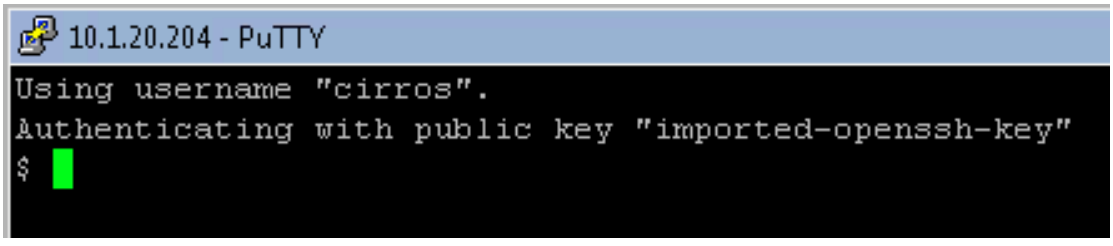
1. Launch Putty
2. Specify your Floating IP in the Host Name field
3. Go to Connection > Data with the left navigation pane
   - Specify "cirros" in the auto-login username field



4. Go to Connection > SSH > Auth to specify our Key. It has to be the key in the ppk format (where you saved it in the previous lab).



5. Click on Open. You should be connected without any challenge for a password (if there was an issue, you can type the password “cubswin:)”)



6. You can try to ping VM Test-2

Troubleshooting the networks

There are different things you can review when troubleshooting the network:

1. Is Neutron running as expected including its agents?
2. Review the network related log files.
3. Access your network environment via its router and execute some commands.

Lab – review neutron and its agents 'status

- On the controller:
  - You can make sure that neutron-server is running as expected (this command may vary based on the distribution you're using):
    `systemctl status neutron-server`

```
+--------------------------+--------------------------
----------+
 [student@Openstack-Controller ~(keystone_admin)]$
systemctl status neutron-server
 neutron-server.service - OpenStack Neutron Server
    Loaded: loaded (/usr/lib/systemd/system/neutron-
server.service; enabled)
    Active: active (running) since Thu 2015-10-22
08:24:55 EDT; 3h 17min ago
  Main PID: 1181 (neutron-server)
    CGroup: /system.slice/neutron-server.service
            ├─1181 /usr/bin/python2 /usr/bin/neutron-
server --config-file /usr/share/neutron/neutron-dist.conf
--config-dir /usr/share/neutron/server --config-...
            ├─3809 /usr/bin/python2 /usr/bin/neutron-
server --config-file /usr/share/neutron/neutron-dist.conf
--config-dir /usr/share/neutron/server --config-...
            ├─3810 /usr/bin/python2 /usr/bin/neutron-
server --config-file /usr/share/neutron/neutron-dist.conf
--config-dir /usr/share/neutron/server --config-...
            ├─3811 /usr/bin/python2 /usr/bin/neutron-
server --config-file /usr/share/neutron/neutron-dist.conf
--config-dir /usr/share/neutron/server --config-...
            └─3812 /usr/bin/python2 /usr/bin/neutron-
server --config-file /usr/share/neutron/neutron-dist.conf
--config-dir /usr/share/neutron/server --config-...
```

  - You can review the agent's status with the command previously seen:
    `neutron agent-list`
  - Even if the status of the agent is fine, it's recommended to make sure that its configuration is loaded properly by doing a

`neutron agent-show <agent ID>`. If something looks wrong, you may review the relevant log file (see next)

- On the network and compute nodes:
  - Make sure openvswitch is running as expected:
    `systemctl status openvswitch`

```
[root@Openstack-Compute2 neutron]# systemctl status
openvswitch
openvswitch.service - Open vSwitch
   Loaded: loaded
(/usr/lib/systemd/system/openvswitch.service;
enabled)
   Active: active (exited) since Thu 2015-10-22
09:43:20 EDT; 2h 7min ago
  Process: 1441 ExecStart=/bin/true (code=exited,
status=0/SUCCESS)
 Main PID: 1441 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/openvswitch.service

Oct 22 09:43:20 Openstack-Compute2 systemd[1]: Starting
Open vSwitch...
Oct 22 09:43:20 Openstack-Compute2 systemd[1]: Started
Open vSwitch.
```

Lab – Review neutron related log files

To review the log files, you'll need to be identified as root:

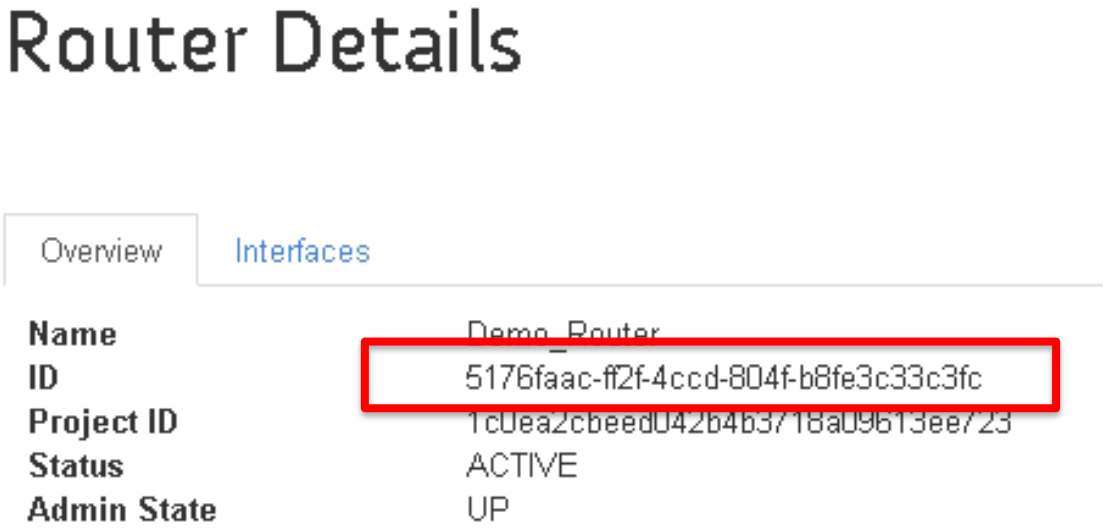
- On the controller – review the following log files
  - You have a /var/log/neutron/server.log file
- On the network and compute nodes – review the following log files
  - /var/log/neutron/openvswitch-agent.log
  - /var/log/neutron/l3-agent.log
  - /var/log/neutron/metadata-agent.log
  - /var/log/neutron/dhcp-agent.log

Lab – Test your project network

If you're not able to reach your instances, you may want to test the network's setup in your project. This can be done from the routers setup in the project. Here, it will be via the Demo_Router:

The first thing we need to retrieve is our router ID:

1. Via the GUI:
   - Connect to the Openstack dashboard and log in as demo.
   - Go to Project > Network > Routers and click on Demo_Router.
   - The router ID is in the overview tab



2. Via the CLI (on the Controller node / source keystonerc_admin):
   - `neutron router-list`

Once you know your router ID, we can use it to test your networks:

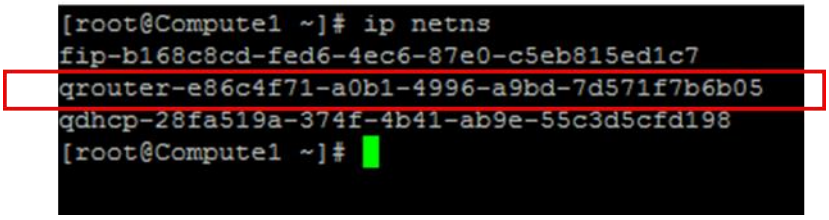
There are 2 possible scenarios:

- You don't use Neutron Distributed Virtual Routers: Then you must connect to the network node and do everything from here
- You use Neutron DVR: here it means that if you want to troubleshoot an instance, you need to connect to its hypervisor. In this lab, Neutron was set to use DVR.

We don't need to source any keystonerc* file since we won't be using an OpenStack client to do this.

Connect to your Compute1 component via SSH.

Type the following command: `ip netns` – this will list the different namespaces available. You may see that one of the qrouter entries contain our router ID. For each router created in OpenStack, there is a specific namespace for it.



If you want to execute commands within this namespace, you may use the command `ip netns exec <namespace name > <your command>`

Try the following commands:

- `ip netns exec <namespace name> ip a`
- `ip netns exec <namespace name> ping <VM Test-1 Management IP>`
- `ip netns exec <namespace name> ping <VM Test-2 Management IP>`

Here, only the ping related to the instance hosted on this hypervisor should work.

- `IP netns exec <namespace name> ssh cirros@<VM Test-1 Management IP> (use the VM that was successfully pinged)`
  - `Password : cubswin:)`
  - `From this ssh instance, try to ping the other instance. It works`

LAB 2 – F5 LBaaSv2 plugin

Introduction

Openstack has deprecated LBaaSv1 since Liberty. Starting Liberty, it is advised to use LBaasv2.

F5 has created a LBaasv1 and LBaaSv2 (Load Balancing as a Service) plugins for OpenStack. Depending on the Openstack version the customer use, you may use one or the other. To review our support matrix, it's available here: http://f5-openstack-docs.readthedocs.io/en/latest/releases_and_versioning.html

For this lab, we will use the LBaaSv2 plugin. It's available here https://github.com/F5Networks/f5-openstack-lbaasv2-driver (click on Releases to see the different version of plugin available)

Our user guide related to LBaaSv2 is here: http://f5-openstack-lbaasv2-driver.readthedocs.io/en/latest/map_f5-lbaasv2-user-guide.html

When using the F5 LBaaS plugin, the user need to consider many factors. One of the thing to consider is whether we want to do ADC "over the cloud" or "under the cloud".

Over the cloud:

- The provider offers generic virtual machine to the tenant
- The customer selects, installs, provisions and operates network services

Under the cloud:

- The provider deploys network services into the infrastructure and exposes them "as a service". This is exactly what our LBaaS plugin enables.
- The customer subscribes to network services as needed.

The F5 LBaaS plugin support "under the cloud" deployment and we will see how it works with the next labs.

F5 under the cloud deployment

Introduction / requirements

If we want to offer LBaaS "under the cloud" to a tenant, we have two choices:

- Deploy BIG-IP(s) in an admin tenant in OpenStack
- Use external BIG-IP(s)

For this lab, we will need either a standalone BIG-IP or a cluster of BIG-IPs. Up to the student.

If we want our BIG-IP(s) to be able to process traffic for each tenant, we need to remember:

- Tenant's networks are GRE tunnels running on top of the "Underlay Network" network which has been setup in the admin tenant (GRE tunnels are specified in the /etc/neutron/plugin.ini file).
- "Underlay Network" network has been setup on the 10.1.20.0/24 network.
- Our BIG-IPs will have to be in the same network to be able to act as VTEP and join the GRE tunnels.

Lab – Install your BIG-IP(s)

For this lab and for performance reason, we will deploy our BIG-IPs in UDF itself and not within Openstack.

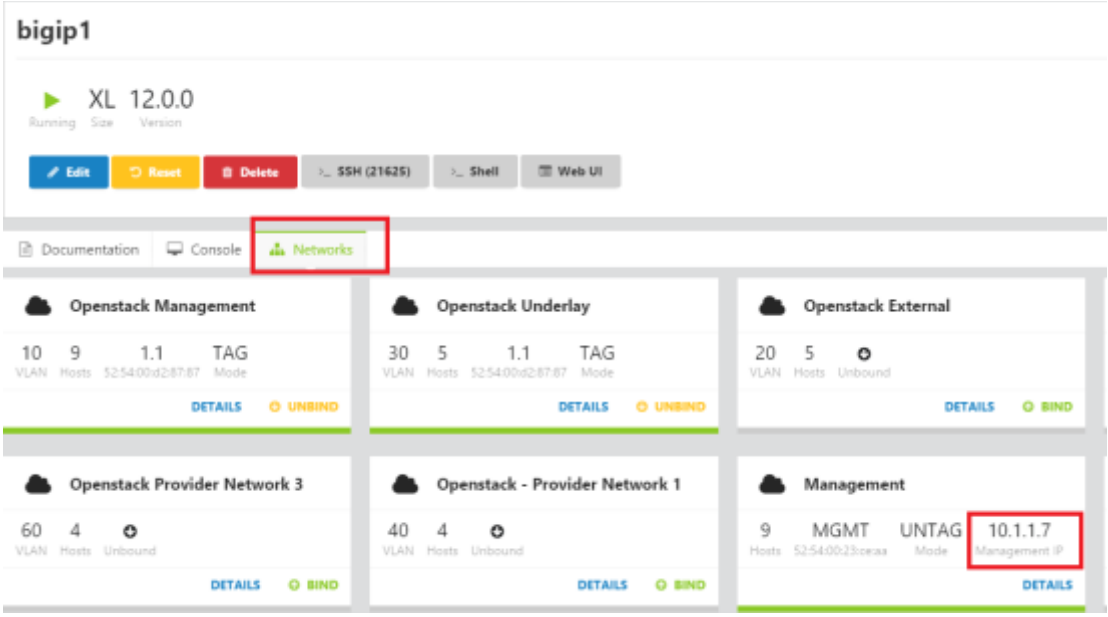
THIS IS ALREADY DONE. Below is the process if you need to deploy another or need to replace the existing one. You may need to renew the license

To deploy a BIG-IP in UDF, go to your deployment page. In the components tab and in the "F5 products" column, click on "+NEW". **For this lab you MUST deploy a BIG-IP running v11.6**. The reason for this requirement is that the latest plugin we will use has not been yet tested with v12.0.
Size XL is enough regarding the instance size and put any name to it.

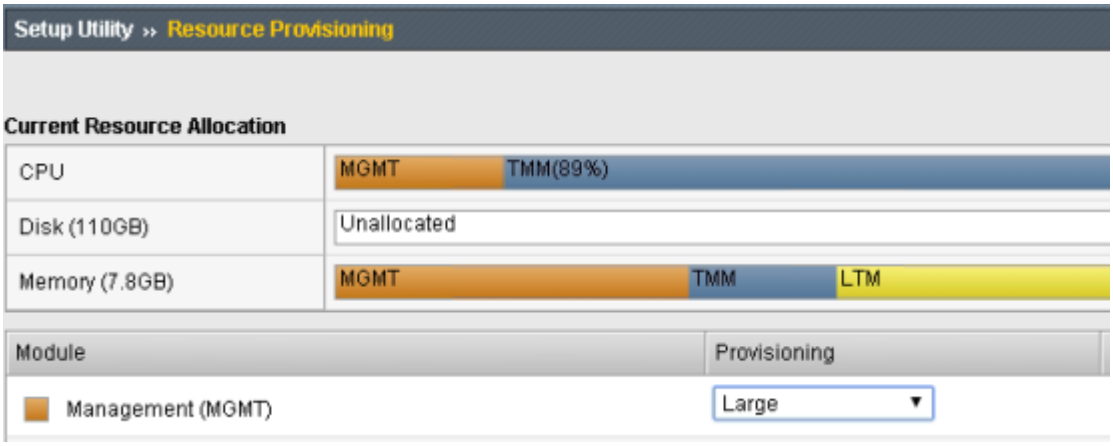
You'll need license(s) for your BIG-IP(s) and they **must have the SDN license**. Without this license, we can setup GRE tunnels but we won't be able to pass traffic.

Once your BIG-IP is deployed, you need to do the following in UDF:

- Click on the Networks tab to retrieve its management address (it should be within the 10.1.1.0/24 Networks.
- "Bind" some more networks to your BIG-IP(s). Click on "Bind' for the following networks:
    - Openstack Management – this network will be used to let our LBaaS agents communicate with the BIG-IP(s) and push the required configuration
    - Openstack Underlay – this network will be used to setup GRE tunnels with the relevant tenants.

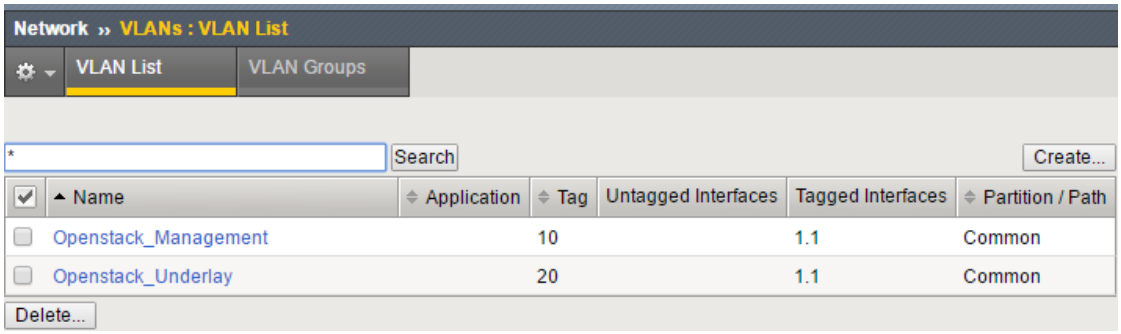


You should be able to access your BIG-IP(s) via HTTPS (via the UDF Web UI button or from your RDP session). License your device as needed. In the provisioning page, **you must setup the management resource to LARGE**. If you fail to do so, your agent won't start and this will be explained in the log file.

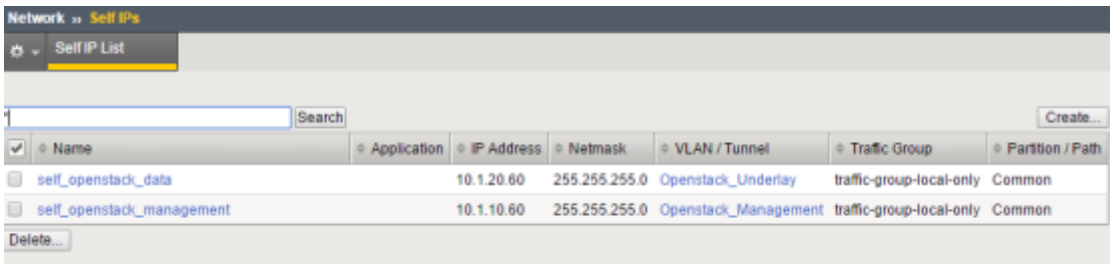


Next step is to define your default VLANs and SELF-IPs.

Define the following VLANS (TAGGED/only 1.1 is used in UDF):



Define the following SELF-IPs (make sure to respect the naming or you'll have to update the lbaas agent configuration to use the name you have specified - for the SELF-IP related to the network overlays)



Setup port lockdown to "Allow All" (default should be good enough). You'll need to be able to do REST API call against the Openstack Mangement Self-IP

Repeat this process if you have multiple BIG-IPs and make sure to set them up as a cluster.

Install F5 LBaaSv2 plugin – installation process

To install the LBaaSv2 plugin, you'll have different installation steps to follow:

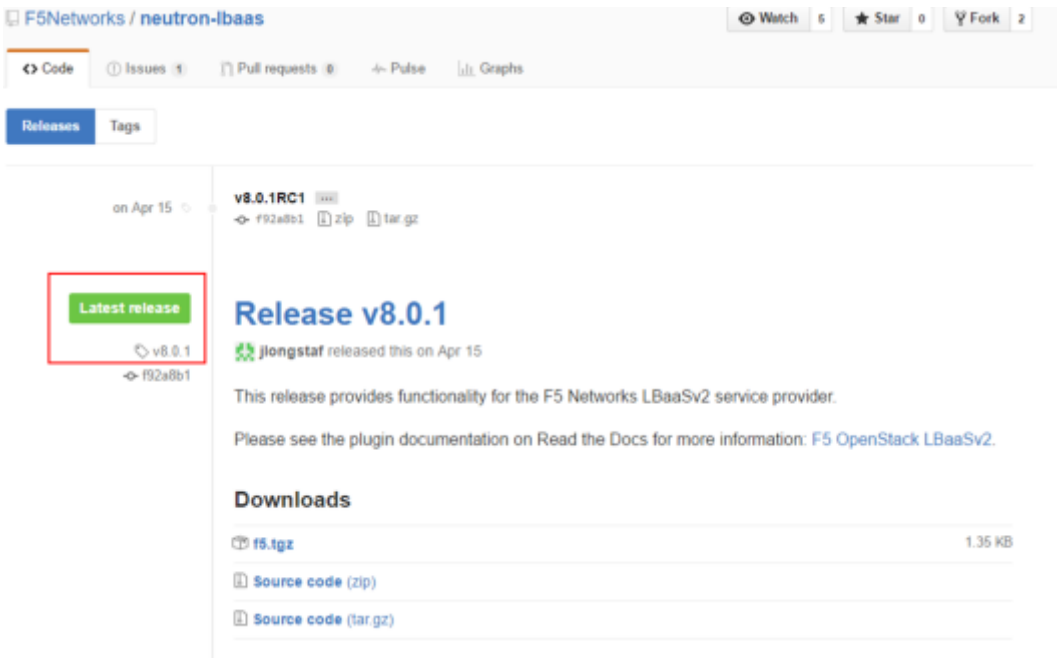
- Install F5 Service Provider Package – on the Controller Node (it hosts the neutron server)
- Install F5 Agent
- Install the F5 LBaaSv2 Driver – on the Controller Node (it hosts the neutron server)

All those steps are documented in our user guide: http://f5-openstack-lbaasv2-driver.readthedocs.io/en/latest/map_f5-lbaasv2-user-guide.html

Lab – Install F5 LBaaS plugin – Service provider package setup

The process is documented here: http://f5-openstack-lbaasv2-driver.readthedocs.io/en/latest/includes/topic_before-you-begin.html#install-the-f5-service-provider-package

Go to https://github.com/F5Networks/neutron-lbaas/releases to see what is the latest version of the package.



Here we see that the latest release is version 8.0.1

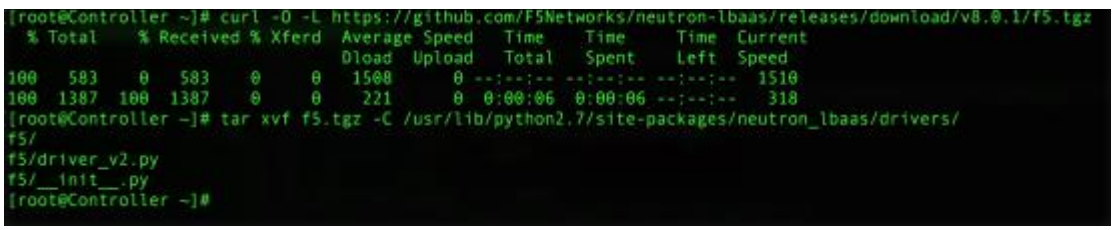
Here is the process to install the service package:

1. Connect via ssh to the Controller
2. Download from GitHub

```
$ curl -O -L https://github.com/F5Networks/neutron-lbaas/releases/download/v8.0.1/f5.tgz
```

3. Install the service provider package.

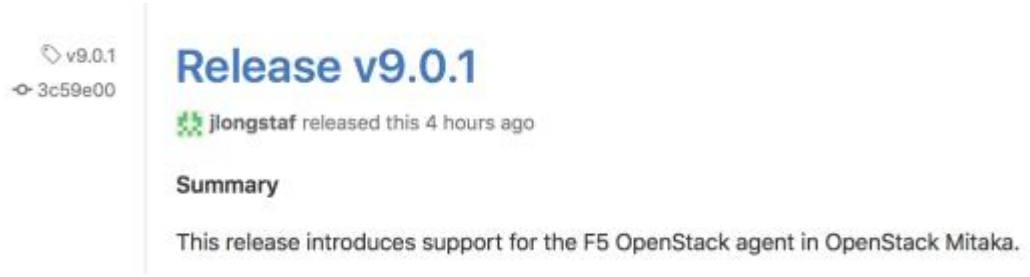
```
$ tar xvf f5.tgz -C /usr/lib/python2.7/site-packages/neutron_lbaas/drivers/
```



Lab – install F5 LBaaS plugin – agent setup

The process is documented here: http://f5-openstack-agent.readthedocs.io/en/latest/

Go to https://github.com/F5Networks/f5-openstack-agent/releases to identify the latest version supporting our release

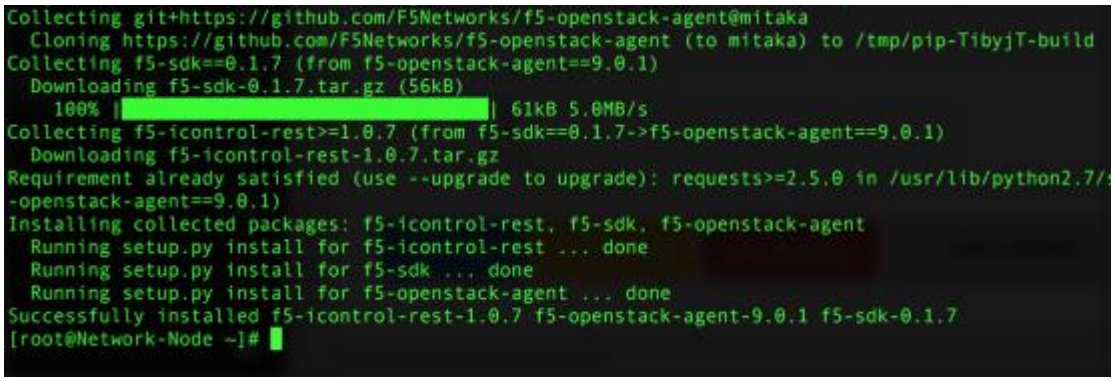


Here the latest release is v9.0.1 that support Mitaka

We can install the agent on any components. It may be on any system, tied or not to the openstack infrastructure.

We will install the agent on the Network node.

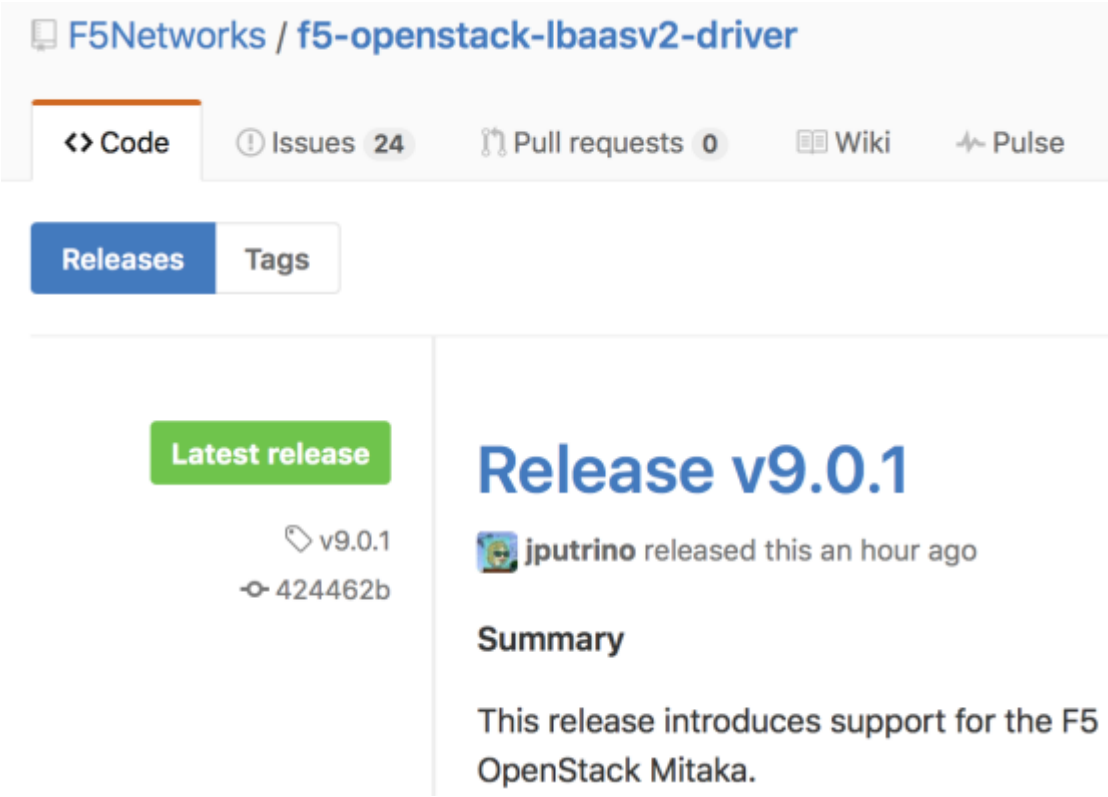
1. Connect to the Network node via SSH
2. Type this command: *pip install git+https://github.com/F5Networks/f5-openstack-agent@mitaka*



Lab – install F5 LBaaS plugin – Driver setup

The process is documented here: http://f5-openstack-lbaasv2-driver.readthedocs.io/en/latest/includes/topic_install-f5-lbaasv2-driver.html

Identify the latest driver release we have: https://github.com/F5Networks/f5-openstack-lbaasv2-driver/releases



Here the latest release is v9.0.1

The driver need to be installed with on the neutron server which is hosted on the Controller node

1. Connect to the Controller via SSH
2. Type this command: *pip install git+https://github.com/F5Networks/f5-openstack-lbaasv2-driver@mitaka*

```
[root@Controller ~]# pip install git+https://github.com/F5Networks/f5-openstack-lbaasv2-driver@mitaka
Collecting git+https://github.com/F5Networks/f5-openstack-lbaasv2-driver@mitaka
  Cloning https://github.com/F5Networks/f5-openstack-lbaasv2-driver (to mitaka) to /tmp/pip-7HWZ_P-build
Installing collected packages: f5-openstack-lbaasv2-driver
  Running setup.py install for f5-openstack-lbaasv2-driver ... done
Successfully installed f5-openstack-lbaasv2-driver-9.0.1
```

Lab – Setup neutron for LBaaSv2

We have to do a few configuration change to the Neutron server configuration to be able to use LBaaSv2

1. Setup F5 as a new LBaasv2 service provider
2. Enable Neutron LBaaSv2 Service plugin in Neutron

The process is documented here: http://f5-openstack-lbaasv2-driver.readthedocs.io/en/latest/includes/topic_configure-neutron-lbaasv2.html

Here are the steps:

1. Connect to the Controller node via SSH
2. Edit /etc/neutron/neutron_lbaas.conf :
    - Edit the service_providers section to set F5Networks as the LBaaSv2 service provider. Add the following line at the end of the filei

```
service_provider =
LOADBALANCERV2:F5Networks:neutron_lbaas.drivers.f5.driver_v2.F5LBaaSV2Driver:defaul
t
```

3. Edit /etc/neutron/neutron.conf
    - ADD the lbaasv2 service to the service_plugins statement:

```
service_plugins = [already defined
plugins],neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
```

In this lab, your service_plugins statement should look like:

```
service_plugins =
router,metering,neutron_lbaas.services.loadbalancer.plugin.LoadBalancerPluginv2
```

4. Restart the neutron server and make sure that it is up and running

```
systemctl restart neutron-server

systemctl status neutron-server
```

```
[root@Controller ~(keystone_admin)]# systemctl restart neutron-server
[root@Controller ~(keystone_admin)]# systemctl status neutron-server
● neutron-server.service - OpenStack Neutron Server
   Loaded: loaded (/usr/lib/systemd/system/neutron-server.service; enabled; vendor preset: disabled)
   Active: active (running) since Thu 2016-07-07 05:28:44 EDT; 1min 7s ago
 Main PID: 14823 (neutron-server)
   CGroup: /system.slice/neutron-server.service
           ├─14823 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron
           ├─14836 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron
           ├─14837 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron
           ├─14838 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron
           ├─14839 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron
           └─14840 /usr/bin/python2 /usr/bin/neutron-server --config-file /usr/share/neutron/neutron

Jul 07 05:28:39 Controller systemd[1]: Starting OpenStack Neutron Server...
Jul 07 05:28:39 Controller neutron-server[14823]: Guru mediation now registers SIGUSR1 and SIGUSR2 b
Jul 07 05:28:40 Controller neutron-server[14823]: Option "verbose" from group "DEFAULT" is deprecate
Jul 07 05:28:40 Controller neutron-server[14823]: Option "notification_driver" from group "DEFAULT"
Jul 07 05:28:44 Controller systemd[1]: Started OpenStack Neutron Server.
Hint: Some lines were ellipsized, use -l to show in full.
[root@Controller ~(keystone_admin)]#
```

If neutron-server doesn't start, you may check your /var/log/neutron/server.log file for any hint on what failed.

Lab – Configure the F5 LBaaS agent

We have installed our LBaaS agent on the network node. The last step is to set it up so that it can communicate with Neutron and our BIG-IP cluster:

- The agent will use the neutron configuration files available on this component to know/learn how to communicate with Neutron. We don't need to do anything here
- Regarding the communication with the BIG-IP, we will need to explain how the configuration should be done and provide a few more information.

Here are the steps needed:

1. Edit the /etc/neutron/services/f5/f5-openstack-agent.ini
   Be aware that this file is sensitive when you manipulate it: if you have a space at the beginning of a line, it may not load properly for example. Everything in red below are comments and should not be put in the file

```
debug = True
periodic_interval = 10
f5_ha_type = standalone OR pair depending on your
setup
f5_sync_mode = replication
f5_external_physical_mappings = default:1.1:True
f5_vtep_folder = 'Common'
f5_vtep_selfip_name = 'self_openstack_data' If you
didn't name your self-IP like this, you must change
it
advertised_tunnel_types = gre
f5_populate_static_arp = True
l2_population = True
f5_global_routed_mode = False
use_namespaces = True
max_namespaces_per_tenant = 1
f5_route_domain_strictness = False
f5_snat_mode = True
f5_snat_addresses_per_subnet = 1
f5_common_external_networks = True
f5_bigip_lbaas_device_driver =
f5_openstack_agent.lbaasv2.drivers.bigip.icontrol_dr
iver.iControlDriver
icontrol_hostname = 10.1.10.60 if multiple BIG-IP in
your cluster, add all their self management IP here
with a comma as a delimiter
icontrol_username = admin
icontrol_password = admin
icontrol_connection_retry_interval = 10
icontrol_connection_timeout = 10
icontrol_config_mode = objects
```

2. Enable the agent as a service and start it:
    - `systemctl enable f5-openstack-agent`
    - `systemctl start f5-openstack-agent`
3. Once it's done you may execute `systemctl status f5-openstack-agent` to see if it's loaded/running properly

```
[root@Network-Node ~]# systemctl status f5-openstack-agent
● f5-openstack-agent.service - F5 LBaaSv2 BIG-IP Agent
   Loaded: loaded (/usr/lib/systemd/system/f5-openstack-agent.service; enabled;
   Active: active (running) since Thu 2016-07-07 05:40:13 EDT; 1min 38s ago
 Main PID: 1539 (f5-oslbaasv2-ag)
   CGroup: /system.slice/f5-openstack-agent.service
           └─1539 /usr/bin/python /usr/bin/f5-oslbaasv2-agent --log-file /var/l

Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.270
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.271
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.271
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.272
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.273
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.291
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.292
Jul 07 05:41:15 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:15.292
Jul 07 05:41:44 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:44.962
Jul 07 05:41:44 Network-Node f5-oslbaasv2-agent[1539]: 2016-07-07 05:41:44.964
Hint: Some lines were ellipsized, use -l to show in full.
[root@Network-Node ~]#
```

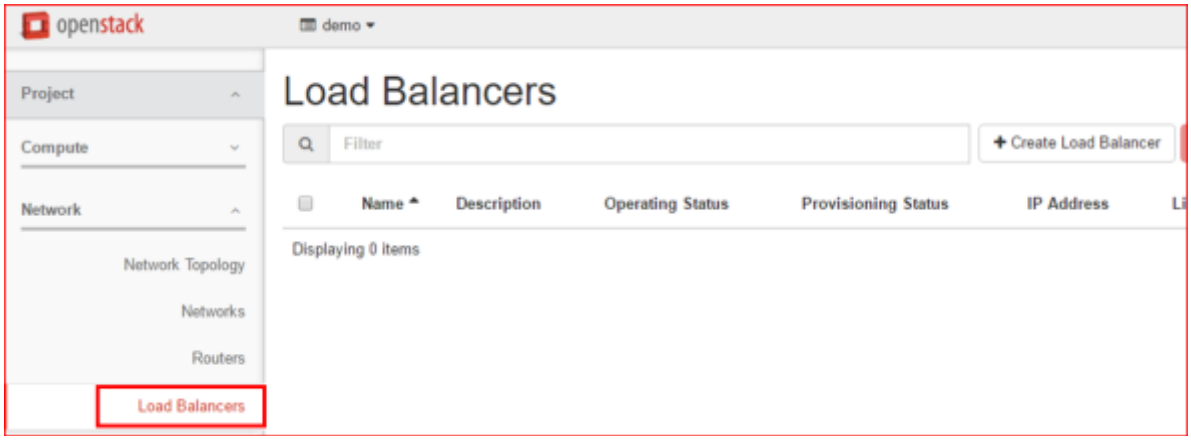
4. Check that a new F5 agent is registered in Neutron (check on the controller) :
   `neutron agent-list`

```
[root@Controller ~(keystone_admin)]# neutron agent-list
+--------------------------+----------------------+--------------------------
| id                       | agent_type           | host
+--------------------------+----------------------+--------------------------
| 21d7c929-d645-49c2-82cb- | Open vSwitch agent   | Compute1
| 39e048a8ebd8             |                      |
| 24ce3f8a-4bbb-41db-      | DHCP agent           | Compute1
| a0a0-458b121c9faa        |                      |
| 3d383d2d-8bef-           | L3 agent             | Compute1
| 44c2-9651-0c25b8cc7bd5   |                      |
| 3db4dc19-e92b-4bf0-ba2b- | Open vSwitch agent   | Compute2
| 5ce1b5a5cefa             |                      |
| 5b2169bb-3813-401e-      | Metadata agent       | Compute2
| ab25-5dde1b388b1c        |                      |
| 65775c4c-d75d-4d79-85bc- | Loadbalancerv2 agent | Network-Node:14f88b7d-36a
| ab17187551b1             |                      | e-5272-9d91-388245eefb95
```

`neutron agent-show <Agent ID>`

```
[root@Controller ~(keystone_admin)]# neutron agent-show 65775c4c-d75d-4d79-85bc-ab17187551b1
+---------------------+--------------------------------------------------------------------------+
| Field               | Value                                                                    |
+---------------------+--------------------------------------------------------------------------+
| admin_state_up      | True                                                                     |
| agent_type          | Loadbalancerv2 agent                                                     |
| alive               | True                                                                     |
| availability_zone   |                                                                          |
| binary              | f5-oslbaasv2-agent                                                       |
| configurations      | {                                                                        |
|                     |      "icontrol_endpoints": {                                             |
|                     |           "10.1.10.60": {                                                |
|                     |                "device_name": "bigip1.lab-nico.org",                     |
|                     |                "platform": "",                                           |
|                     |                "version": "11.6.0",                                      |
|                     |                "serial_number": "12ac8a23-030e-5b49-e592bbee5acc"        |
|                     |           }                                                              |
|                     |      },                                                                  |
|                     |      "request_queue_depth": 0,                                           |
|                     |      "environment_prefix": "Project",                                    |
|                     |      "common_networks": {},                                              |
|                     |      "tunneling_ips": [                                                  |
|                     |           "10.1.20.60"                                                   |
|                     |      ],                                                                  |
|                     |      "services": 0,                                                      |
|                     |      "f5_common_external_networks": true,                                |
|                     |      "environment_capacity_score": 0,                                    |
|                     |      "tunnel_types": [                                                   |
|                     |           "gre"                                                          |
|                     |      ],                                                                  |
|                     |      "environment_group_number": 1,                                      |
|                     |      "device_drivers": [                                                 |
|                     |           "f5-lbaasv2-icontrol"                                          |
|                     |      ],                                                                  |
|                     |      "bridge_mappings": {                                                |
|                     |           "default": "1.1"                                               |
|                     |      },                                                                  |
|                     |      "global_routed_mode": false                                         |
|                     | }                                                                        |
| created_at          | 2016-07-07 09:40:14                                                      |
| description         |                                                                          |
| heartbeat_timestamp | 2016-07-07 09:46:14                                                      |
| host                | Network-Node:14f88b7d-36ae-5272-9d91-388245eefb95                        |
| id                  | 65775c4c-d75d-4d79-85bc-ab17187551b1                                     |
| started_at          | 2016-07-07 09:40:14                                                      |
| topic               | f5-lbaasv2-process-on-agent                                              |
+---------------------+--------------------------------------------------------------------------+
```

If you close your Openstack dashboard and access it again with the user demo, you should see a new option in the UI: Load balancers (Projet > Network > Load Balancers)



Lab – Use the new F5 provider to deploy a load balancing service

Our new F5 provider "lab" is up and running. We can now use it within the OpenStack dashboard to push some load balancing configuration.

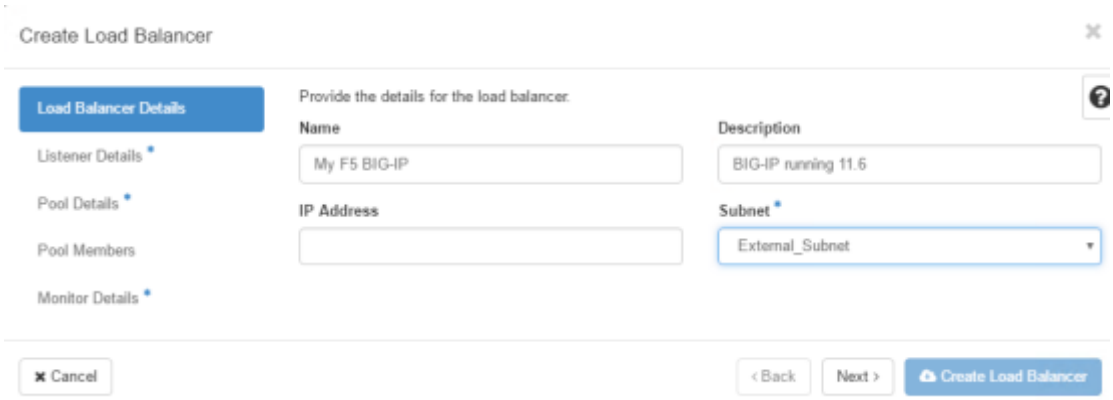
To do this we need to complete our configuration in Project > Network > Load Balancers (OpenStack dashboard)

The cirros image deployed doesn't have any service except ssh. We will use this service to test load balancing.

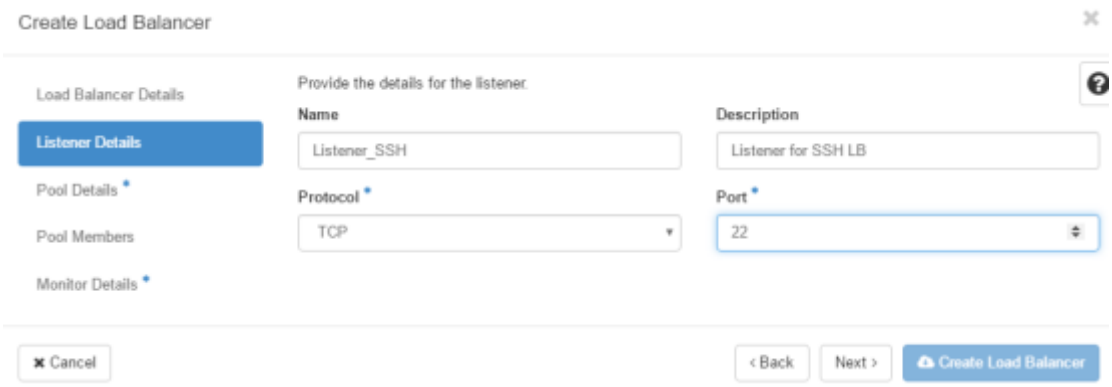
Create a new service

In the demo project, go to Project > Network > Load Balancers

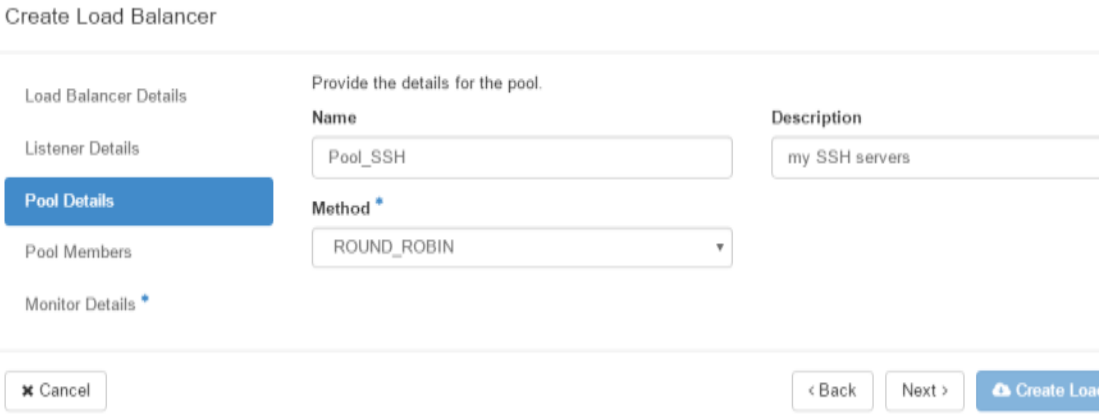
- Click on +Create Load Balancer. On the Load Balancer Details section:
  - Name: My F5 BIG-IP
  - Subnet: External_Subnet
  - IP Address: leave empty



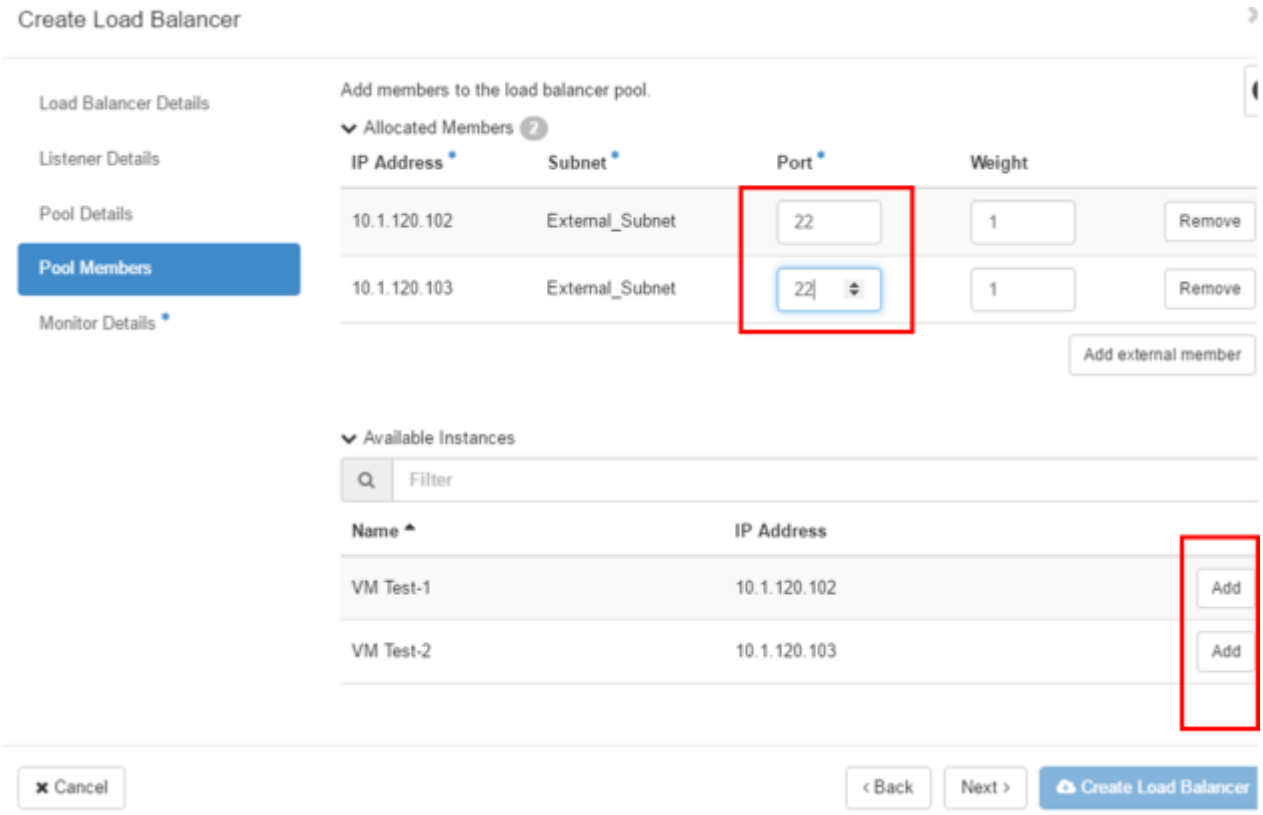
- Click Next (or "Listener Details")
  - Name: Listener_SSH
  - Protocol: TCP
  - Port 22



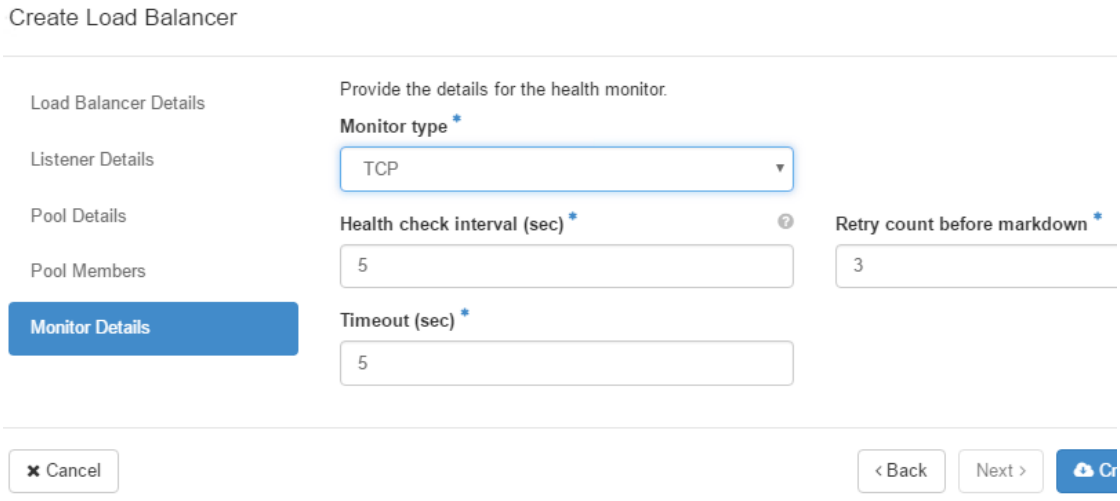
- Click Next (or "Pool Details")
  - Name: Pool_SSH
  - Method: ROUND_ROBIN



- Click Next (or "Pool Members")
  - Select our cirros instances (click on Add next to each IP)
  - Specify port 22 once the instances are added

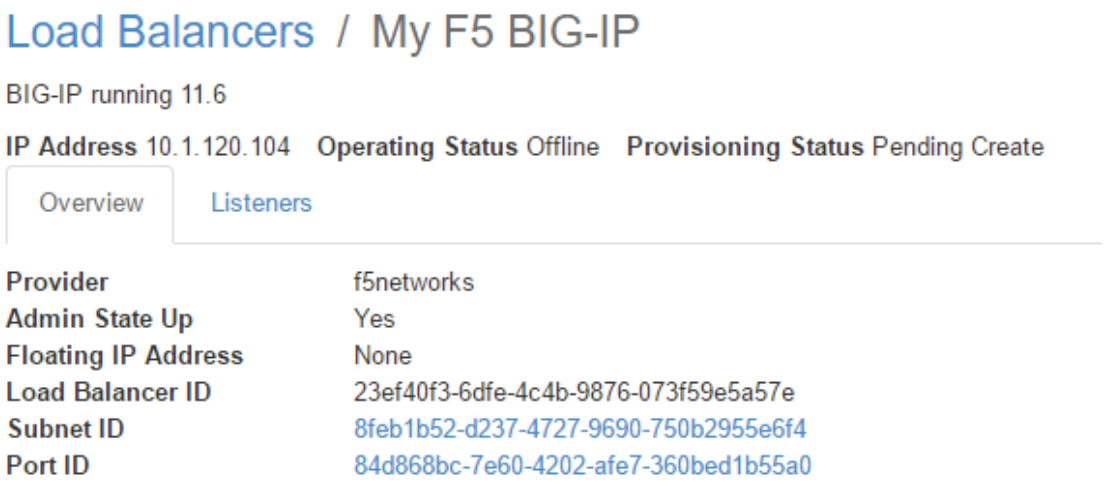


- Click Next (or “Monitor Details”)
  - Monitor type: TCP
  - Leave everything else by default

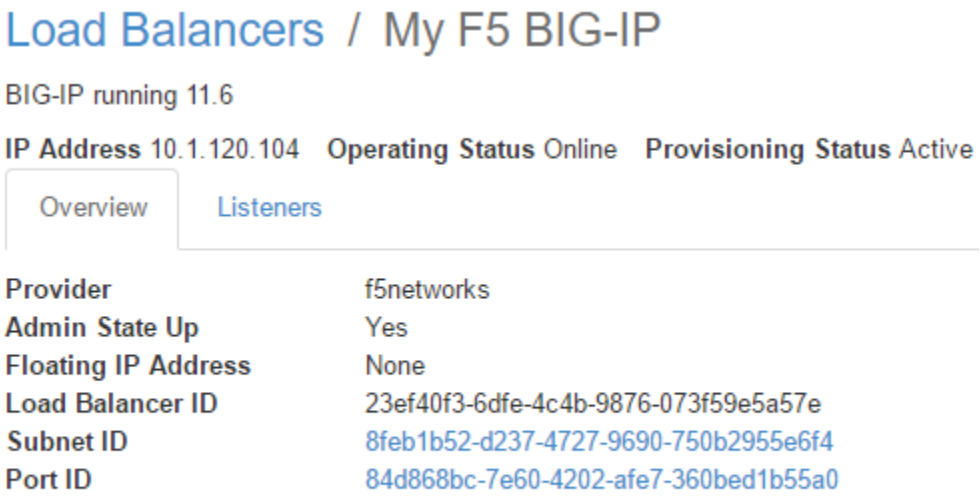


- Click on “Create Load Balancer”.

Click on your Load Balancer called “My F5 BIG-IP”. You should see a page like this:

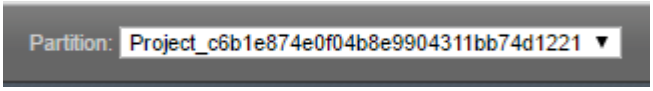


After a few seconds, everything should be online and active



If you connect to your BIG-IP, you should see the pushed configuration:

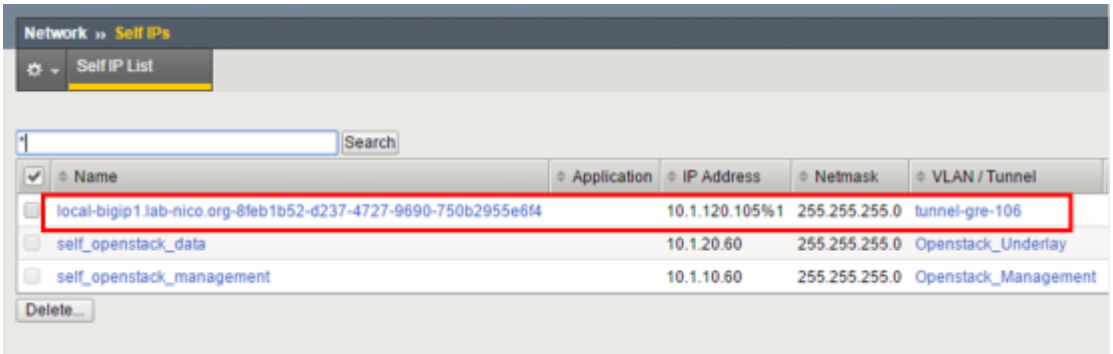
- Once logged in on your BIG-IP, select the partition that was created



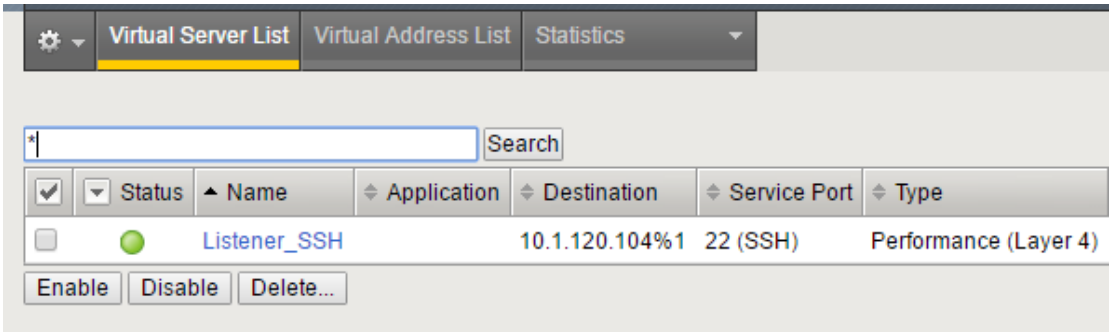
- Review the self IPs setup

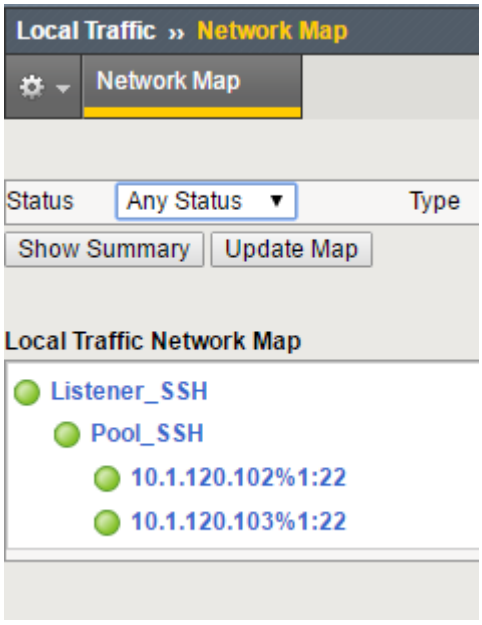
Our BIG-IP(s) didn't had demo's external network configured. This was completed automatically by the plugin. You may review the configuration that was added:

- Network > Self IPs
  You can see that a new Self IP has been created and it is part of demo's external network. If you check the attached VLAN, you'll see that it's a GRE tunnel. You may click on the tunnel name to review its configuration.

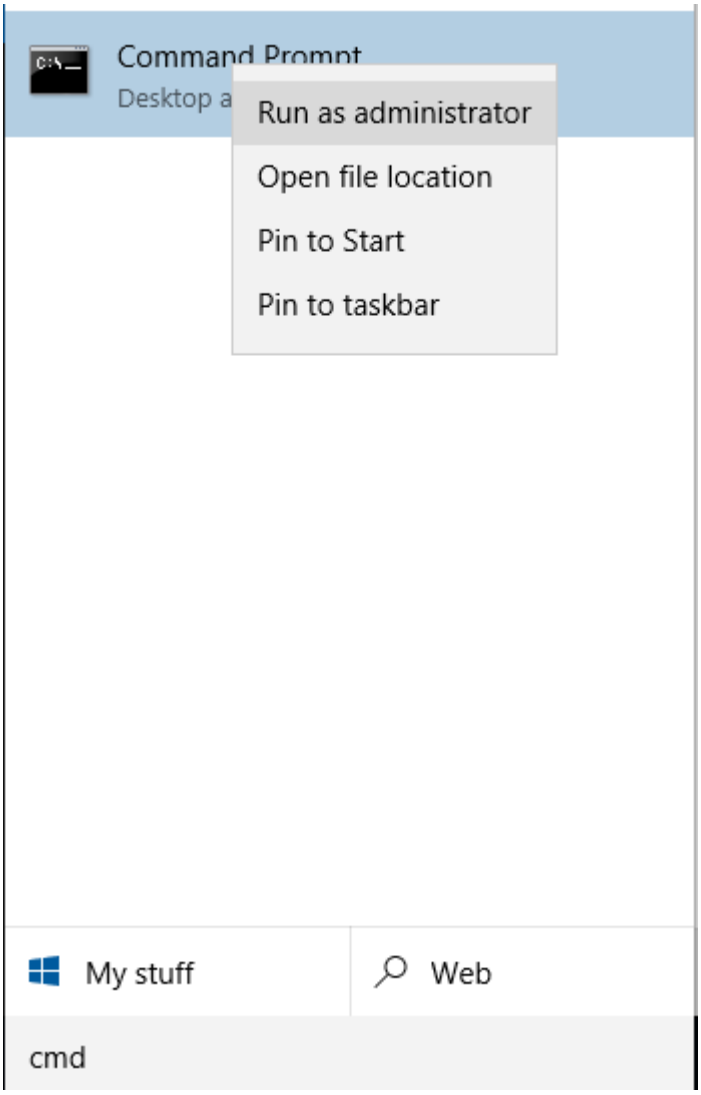


- Review the VS / Pool / Pool members setup

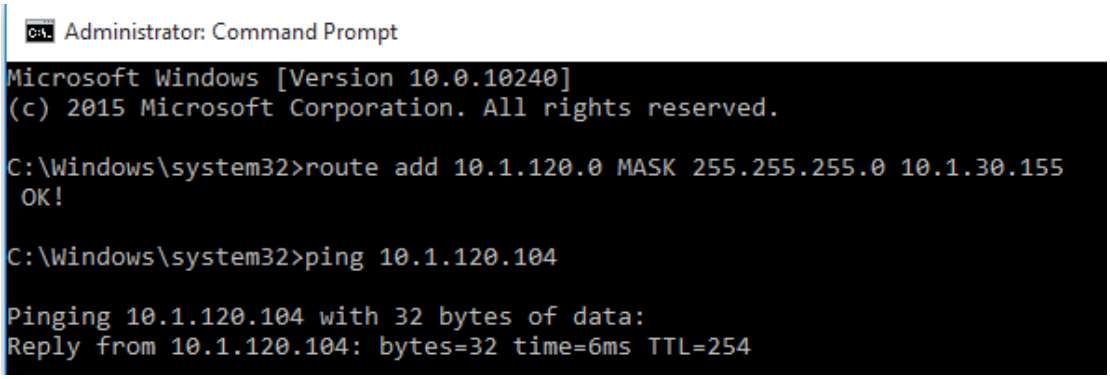




- To test your setup, use your RDP client
  - Launch a command line with administrator privileges

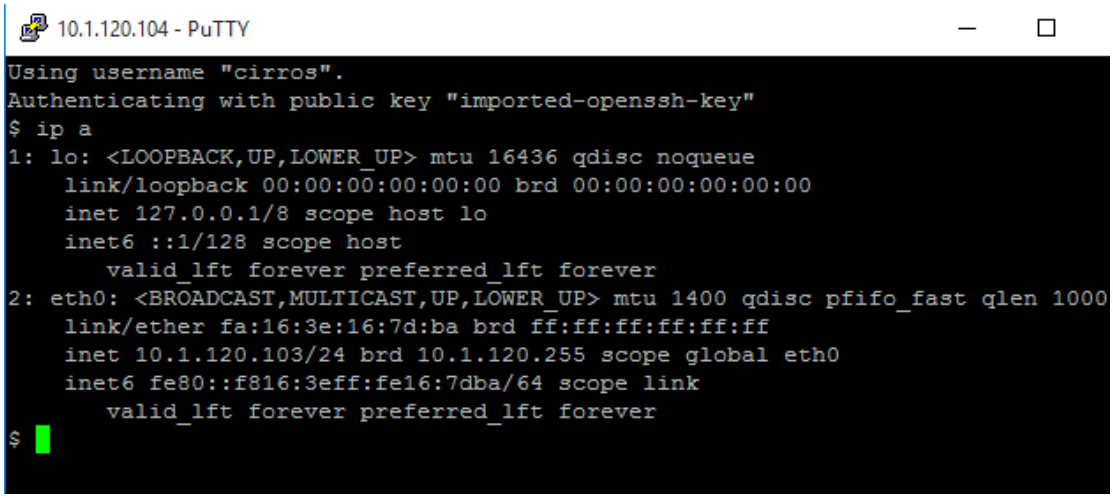


- Type the command route add 10.1.120.0 MASK 255.255.255.0 <Demo_Router public IP>
- Try to ping your BIG-IP VS address



- Launch putty and use the following setup
  - Hostname: IP of your VS
  - Connection > Data > Auto-login username: cirros

    - Connection > SSH > Auth > Private key file for authentication: select the private key (ppk format) you generated previously
- Open your SSH connection and check you reached one of your Openstack cirros instances



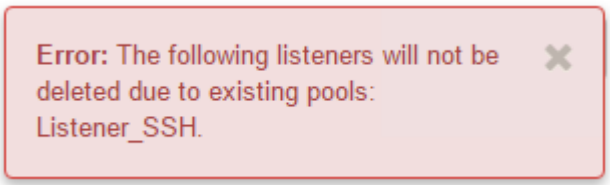
Delete a service

Let's remove the SSH service we created. To do so, we will need to delete the listener setup.

In your Demo project, go to Network > Load Balancers, click on your Load Balancers "My F5 BIG-IP". Click on the Listeners tab

- Select your Listener_SSH
- Click on "Delete Listeners"

You should have an error message:



The horizon LBaaS dashboard has a few limitations/issues so we should use CLI when possible

1. Connect to your controller
2. Source /root/keystonerc_demo
3. Type the following commands to review our existing load balancer configuration
    1. *neutron lbaas-loadbalancer-list*
    2. *neutron lbaas-loadbalancer-show 'My F5 BIG-IP'*

```
[root@Controller ~(keystone_admin)]# source /root/keystonerc_demo
[root@Controller ~(keystone_demo)]# neutron lbaas-loadbalancer-list
+--------------------------------------+-------------+--------------+---------------------+------------+
| id                                   | name        | vip_address  | provisioning_status | provider   |
+--------------------------------------+-------------+--------------+---------------------+------------+
| 23ef40f3-6dfe-4c4b-9876-073f59e5a57e | My F5 BIG-IP | 10.1.120.104 | ACTIVE              | f5networks |
+--------------------------------------+-------------+--------------+---------------------+------------+
[root@Controller ~(keystone_demo)]# neutron lbaas-loadbalancer-show 'My F5 BIG-IP'
+---------------------+------------------------------------------------+
| Field               | Value                                          |
+---------------------+------------------------------------------------+
| admin_state_up      | True                                           |
| description         | BIG-IP running 11.6                            |
| id                  | 23ef40f3-6dfe-4c4b-9876-073f59e5a57e           |
| listeners           | {"id": "f8dd246f-5f0f-4d2d-996c-1c56c2c55610"} |
| name                | My F5 BIG-IP                                   |
| operating_status    | ONLINE                                         |
| pools               | {"id": "7da598f3-16b7-450d-a75f-37c4ecf518d3"} |
| provider            | f5networks                                     |
| provisioning_status | ACTIVE                                         |
| tenant_id           | c6b1e874e0f04b8e9904311bb74d1221               |
| vip_address         | 10.1.120.104                                   |
| vip_port_id         | 84d868bc-7e60-4202-afe7-360bed1b55a0           |
| vip_subnet_id       | 8feb1b52-d237-4727-9690-750b2955e6f4           |
+---------------------+------------------------------------------------+
[root@Controller ~(keystone_demo)]#
```

4. type the following commands to review our listener configuration
    1. *neutron lbaas-listener-list*
    2. *neutron lbaas-listener-show Listener_SSH*

```
[root@Controller ~(keystone_demo)]# neutron lbaas-listener-list
+--------------------------------------+--------------------------------------+-------------+----------+
| id                                   | default_pool_id                      | name        | protocol |
+--------------------------------------+--------------------------------------+-------------+----------+
| f8dd246f-5f0f-4d2d-996c-1c56c2c55610 | 7da598f3-16b7-450d-a75f-37c4ecf518d3 | Listener_SSH | TCP      |
+--------------------------------------+--------------------------------------+-------------+----------+
[root@Controller ~(keystone_demo)]# neutron lbaas-listener-show Listener_SSH
+---------------------------+------------------------------------------------+
| Field                     | Value                                          |
+---------------------------+------------------------------------------------+
| admin_state_up            | True                                           |
| connection_limit          | -1                                             |
| default_pool_id           | 7da598f3-16b7-450d-a75f-37c4ecf518d3           |
| default_tls_container_ref |                                                |
| description               | Listener for SSH LB                            |
| id                        | f8dd246f-5f0f-4d2d-996c-1c56c2c55610           |
| loadbalancers             | {"id": "23ef40f3-6dfe-4c4b-9876-073f59e5a57e"} |
| name                      | Listener_SSH                                   |
| protocol                  | TCP                                            |
| protocol_port             | 22                                             |
| sni_container_refs        |                                                |
| tenant_id                 | c6b1e874e0f04b8e9904311bb74d1221               |
+---------------------------+------------------------------------------------+
[root@Controller ~(keystone_demo)]#
```

5. Type the following commands to review the pool configuration
    1. *neutron lbaas-member-list <default pool id specified in the previous command>*

```
[root@Controller ~(keystone_demo)]# neutron lbaas-member-list 7da598f3-16b7-450d-a75f-37c4ecf518d3
+--------------------------------------+------+--------------+---------------+--------+-------------
| id                                   | name | address      | protocol_port | weight | subnet_id
+--------------------------------------+------+--------------+---------------+--------+-------------
| 567a4d5d-d7f3-49a6-9f71-bf9831b82c84 |      | 10.1.120.102 |            22 |      1 | 8feb1b52-d23
| 6010177b-3b6d-401b-b084-756fbb5e9877 |      | 10.1.120.103 |            22 |      1 | 8feb1b52-d23
+--------------------------------------+------+--------------+---------------+--------+-------------
[root@Controller ~(keystone_demo)]#
```

6. Type the following command to review the pool / monitor configuration
    - *neutron lbaas-pool-show Pool_SSH*

```
a[root@Controller ~(keystone_demo)]# neutron lbaas-pool-show Pool_SSH
+---------------------+----------------------------------------------------+
| Field               | Value                                              |
+---------------------+----------------------------------------------------+
| admin_state_up      | True                                               |
| description         | my SSH servers                                     |
| healthmonitor_id    | 08d5bf35-b02e-4759-a7b3-23a2ea6f53f3               |
| id                  | 7da598f3-16b7-450d-a75f-37c4ecf518d3               |
| lb_algorithm        | ROUND_ROBIN                                        |
| listeners           |                                                    |
| loadbalancers       | {"id": "23ef40f3-6dfe-4c4b-9876-073f59e5a57e"}     |
| members             | 567a4d5d-d7f3-49a6-9f71-bf9831b82c84               |
|                     | 6010177b-3b6d-401b-b084-756fbb5e9877               |
| name                | Pool_SSH                                           |
| protocol            | TCP                                                |
| session_persistence |                                                    |
| tenant_id           | c6b1e874e0f04b8e9904311bb74d1221                   |
+---------------------+----------------------------------------------------+
```

7. If you want to see all the neutron lbaas command, type the following command:
   neutron help | grep lbaas
8. Let's delete our configuration. The order to delete object is important, we must delete object in the following order: healthmonitor, pool members, pool, listener and load balancer.
    1. *neutron lbaas-pool-show Pool_SSH* – identify your healthmonitor_id, members id

```
[root@Controller ~(keystone_demo)]# neutron lbaas-pool-show Pool_SSH
+---------------------+--------------------------------------------------+
| Field               | Value                                            |
+---------------------+--------------------------------------------------+
| admin_state_up      | True                                             |
| description         | my SSH servers                                   |
| healthmonitor_id    | 08d5bf35-b02e-4759-a7b3-23a2ea6f53f3             |
| id                  | 7da598f3-16b7-450d-a75f-37c4ecf518d3             |
| lb_algorithm        | ROUND_ROBIN                                      |
| listeners           |                                                  |
| loadbalancers       | {"id": "23ef40f3-6dfe-4c4b-9876-073f59e5a57e"}   |
| members             | 567a4d5d-d7f3-49a6-9f71-bf9831b82c84             |
|                     | 6010177b-3b6d-401b-b084-756fbb5e9877             |
| name                | Pool_SSH                                         |
| protocol            | TCP                                              |
| session_persistence |                                                  |
| tenant_id           | c6b1e874e0f04b8e9904311bb74d1221                 |
+---------------------+--------------------------------------------------+
```

2. *neutron lbaas-healthmonitor-delete* <your health monitor ID>
3. *neutron lbaas-member-delete* <your member ID> <your pool name> (2 members to delete so need to run this command twice)
4. *neutron lbaas-pool-delete* <your pool name or pool ID >
5. *neutron lbaas-listener-delete <your listener name or listener id >* (you can get it via the command neutron lbaas-listener-list)
6. *neutron lbaas-loadbalancer-delete <your load balancer name or id>*
7. Connect to your BIG-IP and review if the configuration has been removed

LBaaS v2 – troubleshooting

If your LBaaS agent doesn't seem to behave as expected, you may review its log file.

Connect to the Network node and review the log file /var/log/neutron/f5-openstack-agent.log.

If you want to see the whole startup process for the agent do the following:

- Open 2 SSH connections on your network node
- Do tail -f /var/log/neutron/f5-openstack-agent.log in one session
- On the other ssh session, execute the following command: systemctl restart f5-openstack-agent

In your first session you can review the whole startup process and the periodic checkings done by the openstack agent

Lab 3 - Orchestration / Automation with HEAT

Introduction

Heat is the main project of the OpenStack orchestration program. It allows users to describe deployments of complex cloud applications in text files called *templates*. These templates are then parsed and executed by the Heat engine.

Heat was born as the counterpart to the CloudFormation service in AWS. It accepts AWS templates and provides a compatible API, but in recent OpenStack releases it has also begun to grow outside of the shadow of CloudFormation, providing a template syntax (the Heat Orchestration Template, or HOT) and new features.

HOT templates are written as structured YAML text files. If you want to learn more about HOT templates: Openstack docs

Heat can be accessed via the CLI, and using RESTful queries. It can also be accessed easily via Horizon, which can be quite powerful, since Horizon knows how to interpret and present HOT templates as easy-to-use web UI, complete with multiple-choice popdowns and input-checking to help you populate them error-free.

Lab – Test HEAT

Let's make sure that HEAT works as expected with a small HEAT stack

Here is a basic HEAT template

```
heat_template_version: 2013-05-23

description: Simple template to deploy a single compute instance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      image: cirros
      flavor: m1.small
      key_name: DemoKeyPair
      networks:
        - network: external
```

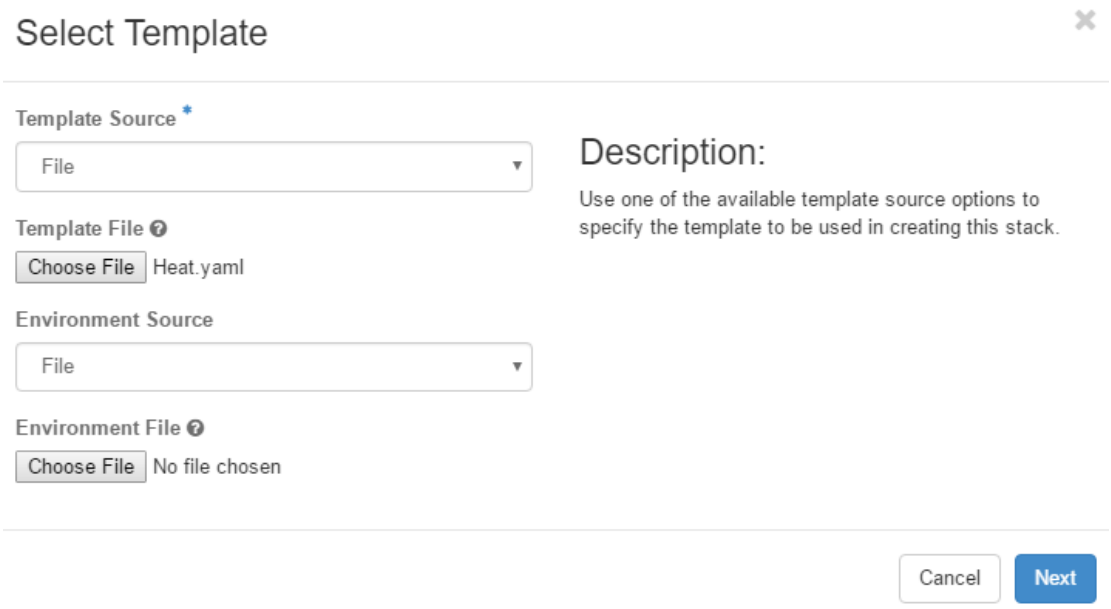
Save this template in a .yaml file. You may need to change the following:

- Change the key_name attribute to the name of the key you created in Openstack (Compute > Access & Security > Key Pair)
- The network name to match the name of the network you created in your demo project

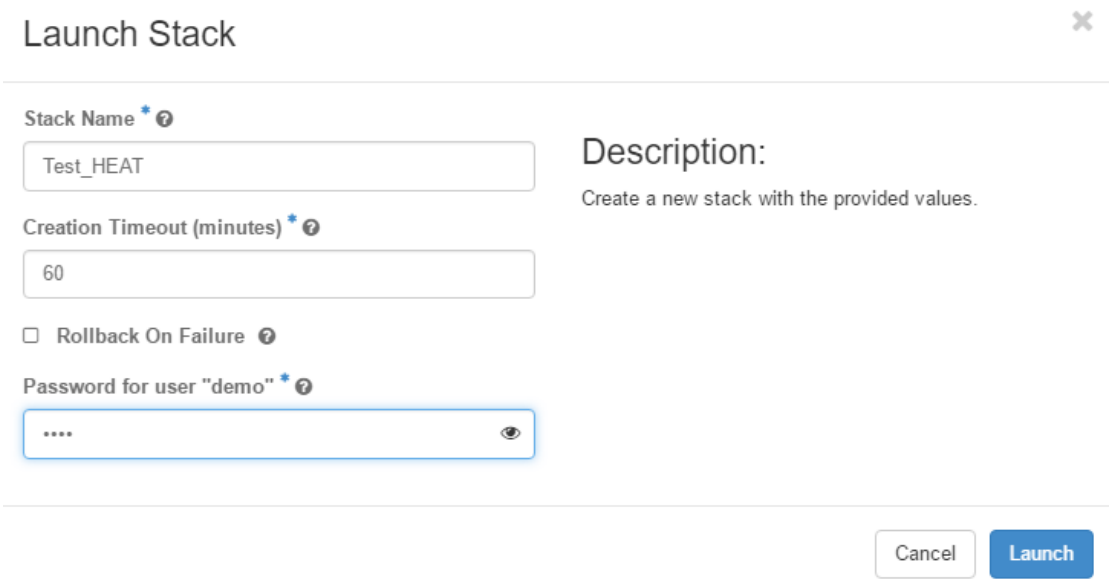
Connect to the dashboard for your demo project.

Go to Orchestration > Stacks and Click on "+ Launch Stack"

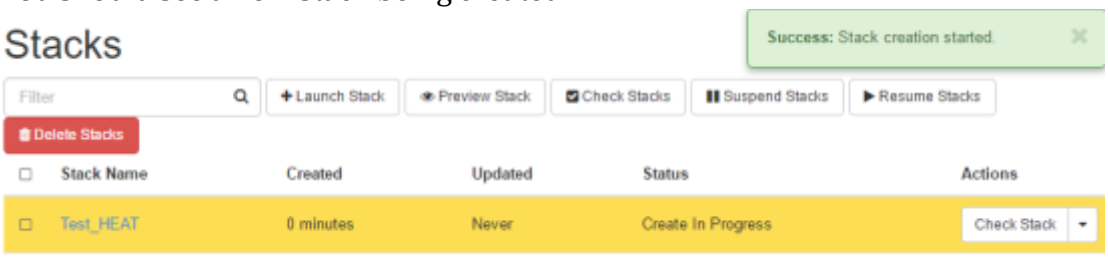
- Template Source: File
- Template File: Select the yaml file you just created
- Click on Next



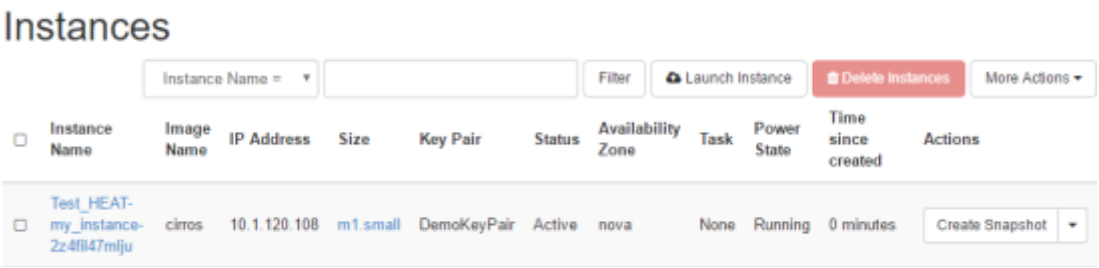
- Stack Name: Test_HEAT
- Specify your password for the demo user
- Click on the "Launch" Button



You should see a new stack being created.

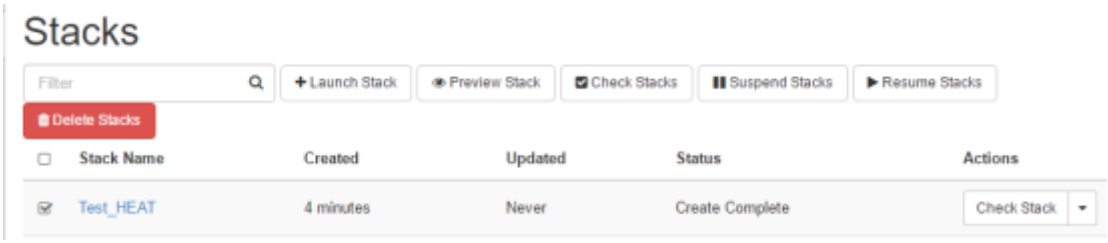


If you go to Project > Compute > Instances. You should have a new cirros instance that got launched automatically and attached to your external networks. Your key should also have been assigned to it



Go back to Project > Orchestration > Stacks.

Select the stack that got created and click on “Delete stacks”



Once the request is processed:

- The stack should be removed from the page
- Your instance should have been deleted

LAB – Push a LTM config via HEAT

Your customer may request to be able to leverage features from F5 that are not in Openstack's LBaaS specifications. A way to do this is to leverage HEAT to push F5 services.

To push this kind of services from HEAT, you'll need the F5 HEAT plugin: http://f5-openstack-heat-plugins.readthedocs.io/en/latest/ . It is hosted here: https://github.com/F5Networks/f5-openstack-heat-plugins/

This plugin must be installed where the HEAT engine reside. In our lab, this is on the Controller.

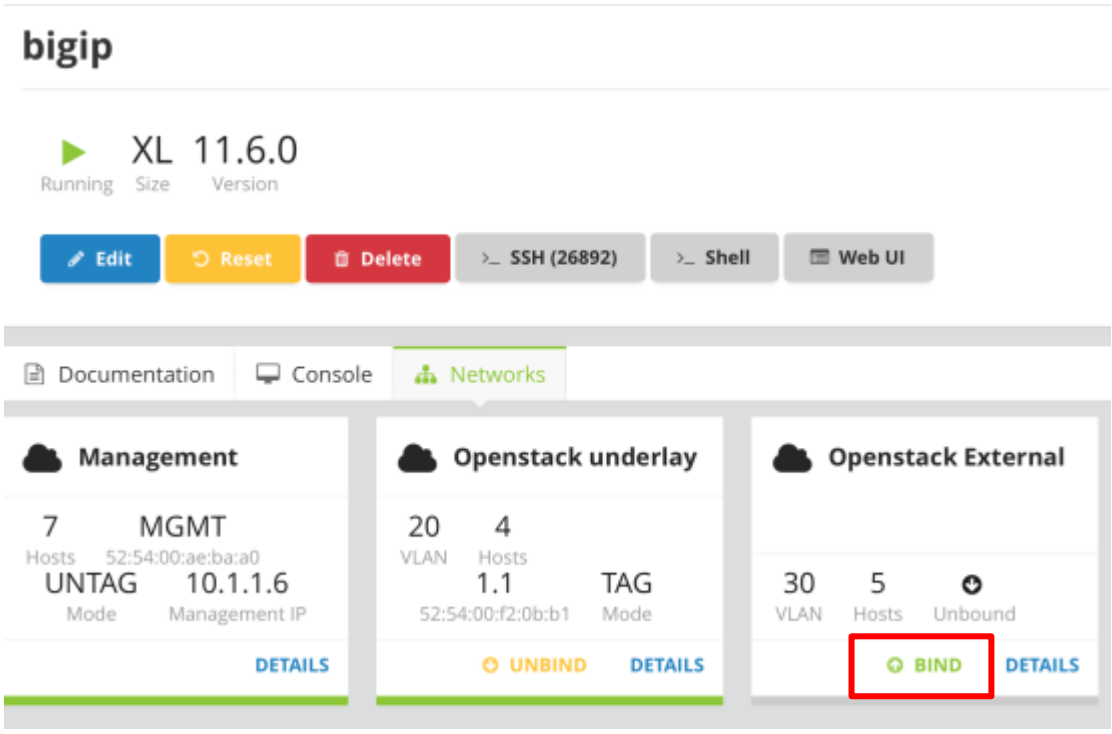
The plugin is already installed and you don't need to do it.

The following command was used to specify that we needed the plugin for the Mitaka tree:

pip install git+https://github.com/F5Networks/f5-openstack-heat-plugins@mitaka

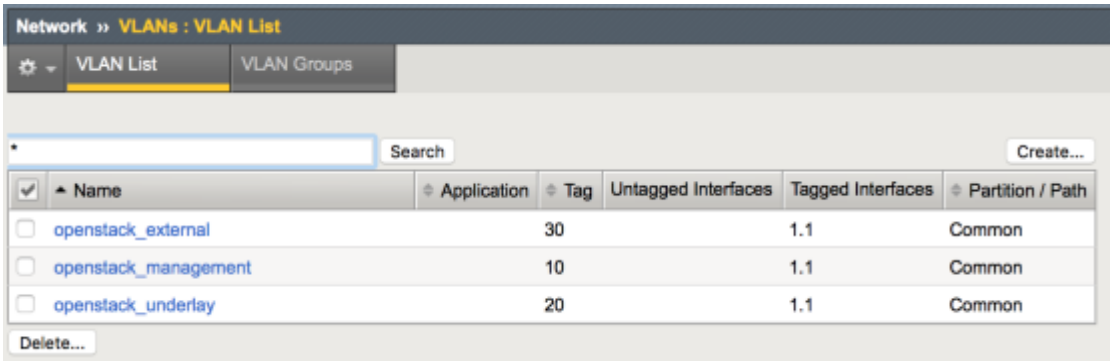
To be able to do this lab, we will need to connect our BIG-IP to the external network of our lab.

To do this, connect to the UDF interface and bind the BIG-IP into the Openstack External Network

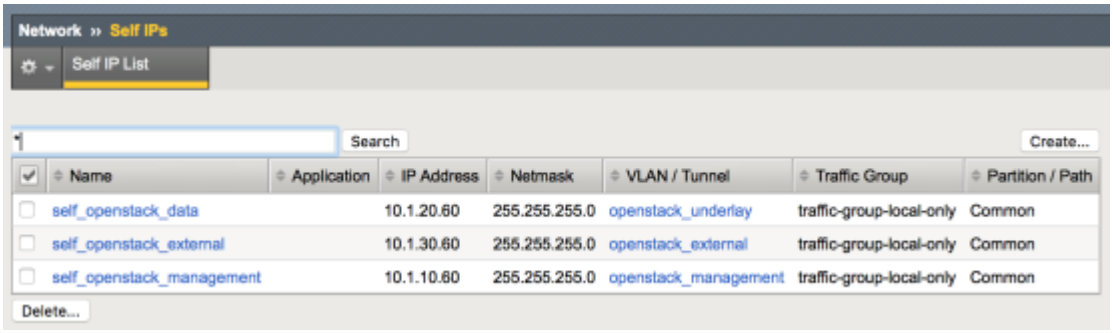


Once this is done, connect to your BIG-IP UI and add the following vlan / self IP:

- Vlan Openstack_External, tagged 30 with interface 1.1

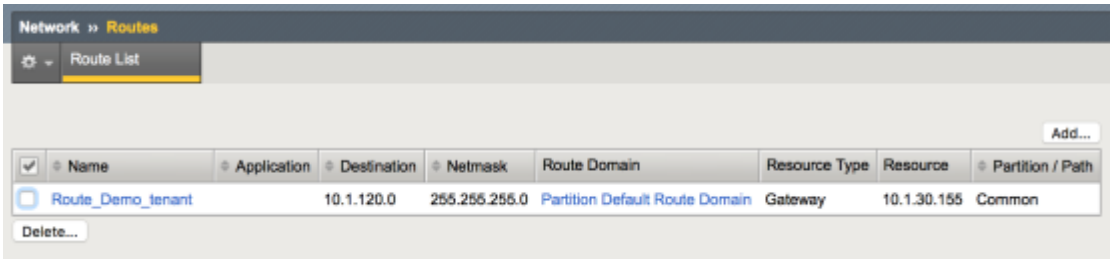


- Self Self_openstack_external, 10.1.30.60/24



We will also need a route to be able to send traffic toward the demo project external network (10.1.120.0/24).

Review your Demo_Router IP in the dashboard: Network > Routers > Click on Demo_Router and then on the overview tab



the HEAT template we will use will do the following:

- Deploy a Client and Server
- Deploy the LTM config specified in the HEAT template
- Server:
    - Will simulate a web server
- Client:
    - Will try a HTTPS connection via our VIP

To deploy your Openstack HEAT template, go to your Openstack dashboard and Orchestration > Stacks. Click on “Launch Stack”

For your stack, select the file in Desktop > HEAT Templates > deploy_lb.yaml

Specify the information as follow:

Launch Stack

Stack Name *

deploy-lb-config

Creation Timeout (minutes) *

60

Rollback On Failure

Password for user "demo" *

••••

Server Glance Image *

ubuntu-server (293.9PB)

Server Nova Flavor *

m1.medium

Key Name *

DemoKeyPair

Security Group

INGRESS-HTTP-ICMP-HTTPS-SSH

Server Network *

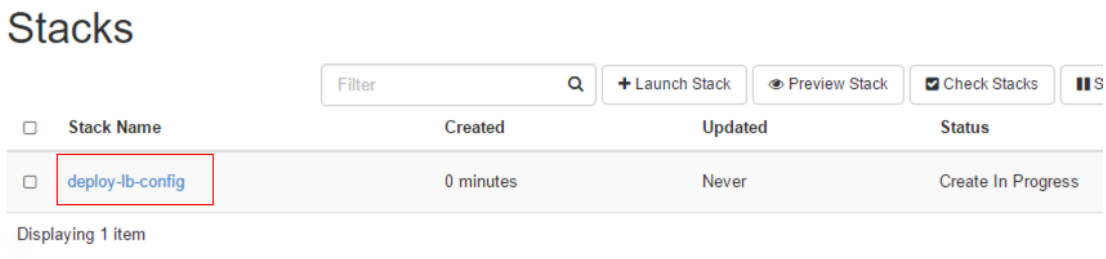
External

BigIP FIP

10.1.10.60

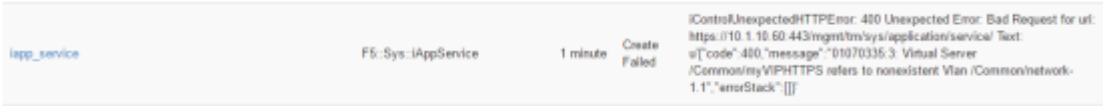
Click on Launch.

Click on your stack name to review its progress



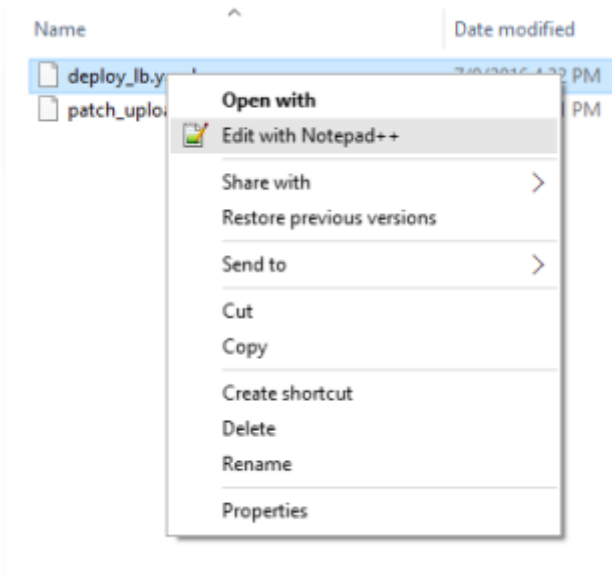
Click on the Resources tab and check the different information provided here.

After a few seconds, you should have the iApp_service resource showing an error message:

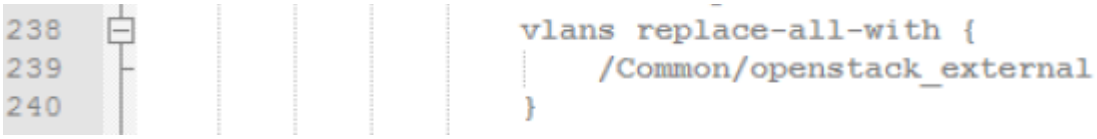


We can see here that we tried to setup a non-existing vlan in our VIP. This stack is due to failure so let's delete the stack. Go back to Orchestration > Stacks and delete your stack.

Edit the HEAT template in the Desktop > HEAT Templates directory. Notepad++ is installed to make it more friendly



Edit line 239 and replace /Common/network-1.1 with /Common/openstack_external



Save your file and try to launch again the stack.

Go into your stack and the resources tab, make sure everything is proceeding as expected this time

Stacks / test

Topology | Overview | Resources | Events | Template

Stack Resource	Resource	Stack Resource Type	Date Updated	Status
client_wait_condition		OS::Heat::WaitCondition	0 minutes	Create In Progress
client_data_port	70ef6638-2c67-461d-8bb4-1ac8711e84ec	OS::Neutron::Port	0 minutes	Create Complete
server_wait_condition		OS::Heat::WaitCondition	0 minutes	Create In Progress
iapp_template		F5::Sys::iAppCompositeTemplate	0 minutes	Create Complete
partition		F5::Sys::Partition	0 minutes	Create Complete
client_wait_handle	256b5aa0eb9141c3b5a0cb5eda89dd6f	OS::Heat::WaitConditionHandle	0 minutes	Create Complete
server	b2cf2fc9-0797-463d-88a2-c8d1191f42d6	OS::Nova::Server	0 minutes	Create Complete
bigip	test-bigip-ie32mlx5mjot	F5::BigIP::Device	0 minutes	Create Complete
server_wait_handle	58a0d597836a4e2c9640eb6a030e828d	OS::Heat::WaitConditionHandle	0 minutes	Create Complete
client		OS::Nova::Server	0 minutes	Init Complete
iapp_service		F5::Sys::iAppService	0 minutes	Create Complete
server_data_port	18deaae7-b81c-42df-8228-731196bbb94b	OS::Neutron::Port	0 minutes	Create Complete

Displaying 12 items

Check that the servers have been created:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State
server2	ubuntu-server	10.1.120.121	m1.medium	DemoKeyPair	Active	nova	None	Running
server	ubuntu-server	10.1.120.120	m1.medium	DemoKeyPair	Active	nova	None	Running

Review your BIG-IP configuration and check what has been created

Local Traffic » Virtual Servers : Virtual Server List

Virtual Server List | Virtual Address List | Statistics

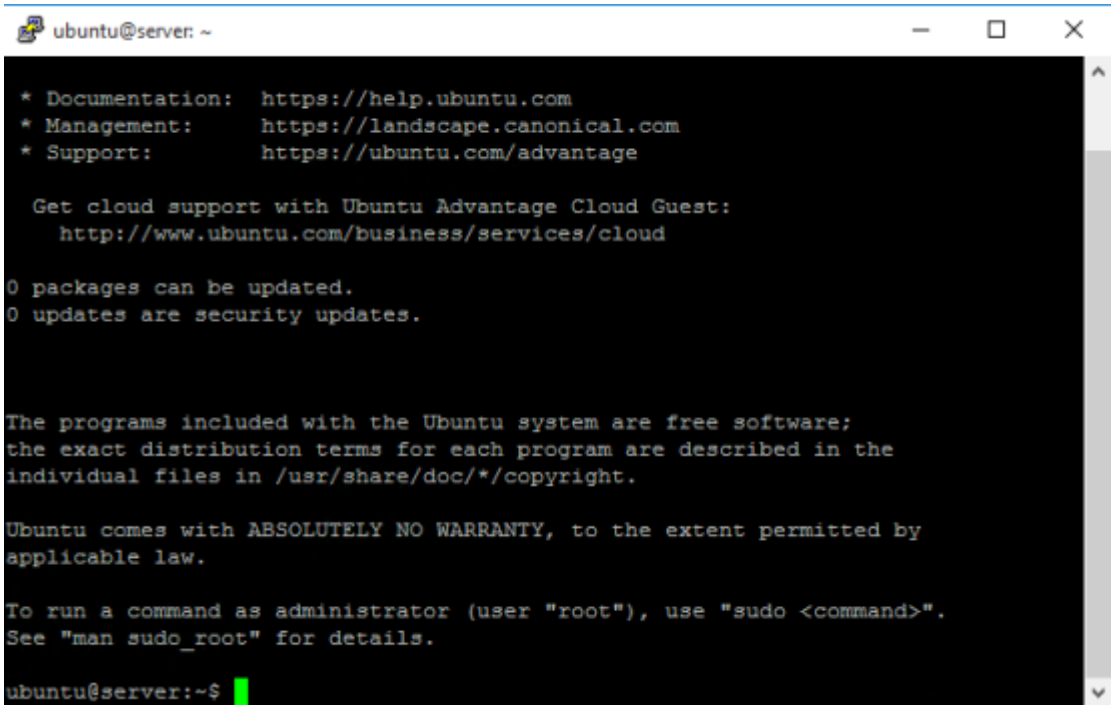
Search

Status	Name	Application	Destination	Service Port
	My_VIP_SSH	lb_service	10.1.30.22	22 (SSH)

Enable | Disable | Delete...

Try to access this VIP from your RDP session with PUTTY:

- User Ubuntu (Connection > Data > auto-login username: Ubuntu)
- Connection > SSH > Auth > Specify your key



Once you're done, try to delete your stack and review if everything is removed successfully.

Try to play with the template (backup it first) to change a few things and see how it impacts the deployment

LAB – F5 Virtual Edition onboarding – Patch a BIG-IP image (optional)

This lab is optional. The main reason is that the process can take quite some time due to all the required step to do this automatically. It is strongly recommended to do this at the end of the day or later. Expect around 30 to 40 min for the process to be done.

Overview of the metadata service

OpenStack is a cloud computing architecture that provides self-service infrastructure for tenants. In addition to self-service, OpenStack provides mechanisms to simplify and accelerate device onboarding. Every instance deployed in OpenStack has access to a metadata (this term is used interchangeably with user-data) service.

Metadata is provided at the time of instance creation in an arbitrary format of the user's choice. JSON is a common choice for this data and the format we've chosen. Other common formats include cloud-init, raw scripts, etc. It is up to the instance to ingest and make use of this data, which makes this method of device onboarding extremely flexible. The metadata service is HTTP based and is available via a link-local address (169.254.169.254).

The metadata is returned upon a request to http://169.254.169.254/latest/user-data. The metadata service also provides access to SSH keys, which can then be injected into a running instance providing password-less authentication.

How BIG-IP can access this metadata service

In order to retrieve and parse this data in TMOS, a Bash script (/config/startup - https://support.f5.com/kb/en-us/solutions/public/11000/900/sol11948.html?sr=48643683) is injected into the qcow disk image prior to first boot.

The startup script is a special file that lives in the BIG-IP's configuration directory and is executed automatically via /etc/rc.local on every boot. As the script is launched on every boot, parameters provided via metadata such as SSH keys, passwords, provisioned modules, etc. may be overwritten when the BIG-IP system is rebooted. The script can also be executed manually at any time from a Bash shell on the BIG-IP.

The OpenStack initialization script automates many of the manual steps needed to bring a BIG-IP online. These settings include: licensing, module provisioning, SSH key injection, admin and root passwords, and interface configuration (VLAN tagging, DHCP or static address assignment). Once the BIG-IP has completed its auto-initialization, the unit will be bootstrapped with the minimal configuration necessary to begin building virtual services. All initialization actions and errors will be logged to /var/log/messages.

Lab - Injecting a new startup script in a BIG-IP qcow image

To let the BIG-IP access the metadata service, we need to include a specific startup script in the BIG-IP image that will be deployed to enable F5 services.

We have a fully documented/maintained process to do it here: https://github.com/F5Networks/f5-openstack-image-prep or we could use a heat template to do this automatically: https://f5-openstack-heat.readthedocs.io/en/latest/templates/supported/ref_images_patch-upload-ve-image.html

We will use the second example.

This example will use a HEAT template to automatically patch a BIG-IP image and push it into Glance.

You need the following:

- A web server hosting the BIG-IP image to patch. It is already installed and is available at the following URL: http://10.1.10.80/ BIGIP-11.6.1.0.0.317.LTM_1SLOT.qcow2.zip
- The yaml file : the yaml we used as a base for this exercise comes from here https://f5-openstack-heat.readthedocs.io/en/latest/_downloads/patch_upload_ve_image.yaml

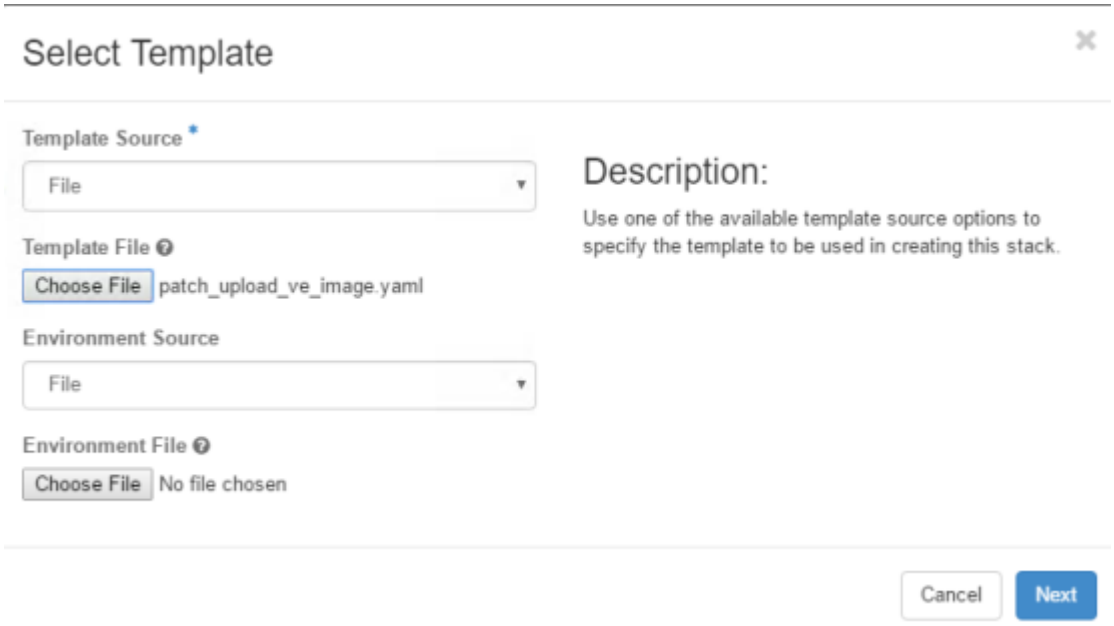
This HEAT template will do the following:

- Deploy a unbuntu instance and patch it to the latest version
- Install all the relevant packages to be able to patch our BIG-IP image
- Retrieve the BIG-IP image from our web server and patch it.
- Upload it into Glance

To do this lab, **Connect to the dashboard as ADMIN**
we will use a provider network that has access to the Openstack management network in UDF (our instance will need to be able to communicate with the Openstack infrastructure to load our image in Glance)

Go to Project > Orchestration > Stacks and launch a new stack.

Select the File that is in the Desktop > Heat Templates folder called patch_upload_ve_image.yaml and click Next



In the Stack specify:

- Ubuntu image: Ubuntu-server
- Flavor: m1.medium
- Onboard server mgmt network: Openstack_Mgmt
- SSH Key: AdminKey (you have its ppk already created in the Desktop > HEAT Templates folder)
- All the required mentioned passwords: admin

For all the other field, leave the default value.

Launch Stack

Stack Name *

patch-bigip-image

Creation Timeout (minutes) *

60

Rollback On Failure

Password for user "admin" *

•••••

Onboard Ubuntu Image *

ubuntu-server (293.9PB)

F5 Onboard Server Flavor

m1.medium

F5 Onboard Server Use Config Drive

F5 Onboard Server Management Network *

Openstack_Mgmt

SSH Key *

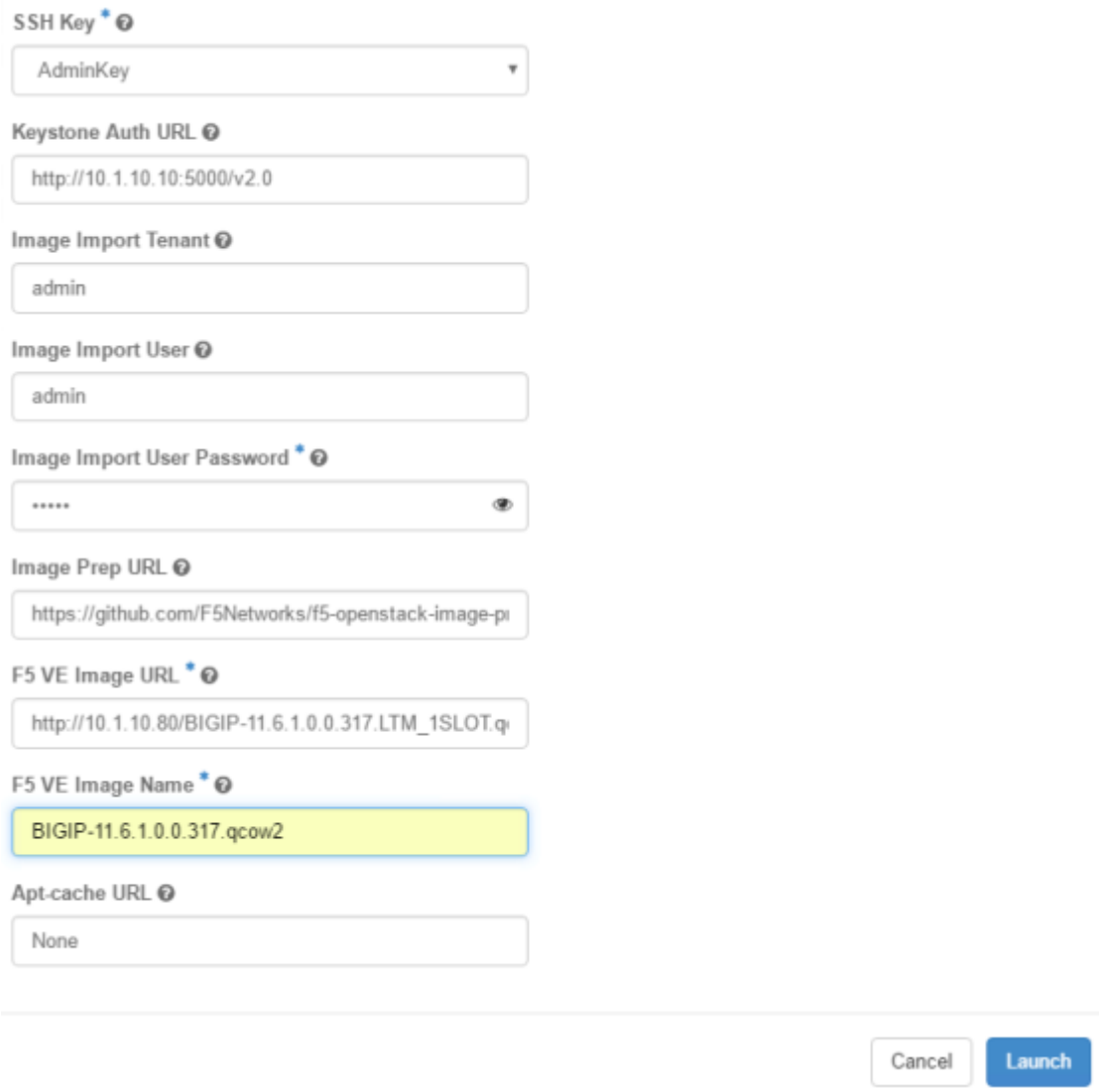
AdminKey

Keystone Auth URL

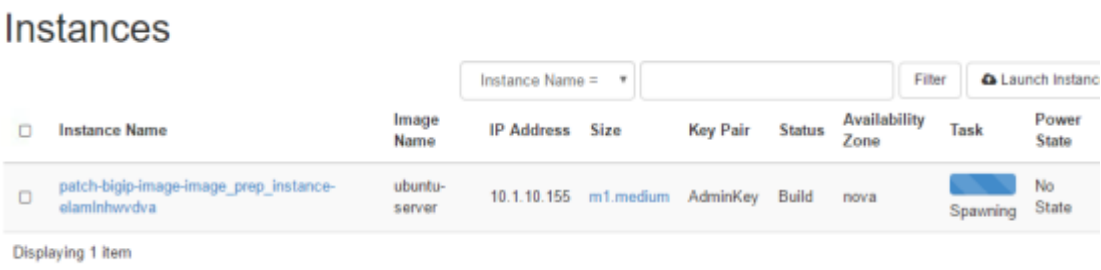
http://10.1.10.10:5000/v2.0

Description:

Create a new stack with the provided values.



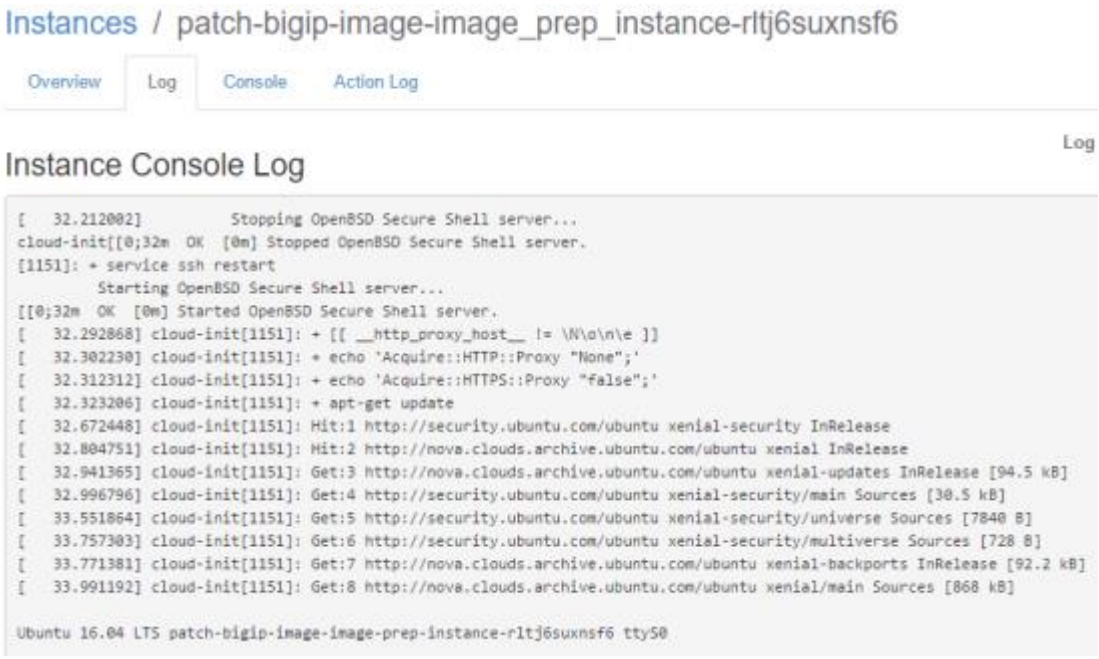
Once you've clicked on launch, go to Project > Compute > Instances. You should see an instance starting



Once it is in a running state, click on the instance name and go to the log tab

Here you'll be able to follow the ongoing process. Click on "View full log" if you want to review the whole process in details

Startup of the image:



Install all the required packages to patch our image:

```
Reading package lists...
+ apt-get -y install unzip qemu-utils lvm2 python-keystoneclient python-glanceclient python-eventlet python-suds python-paramiko git
Reading package lists...
```

Retrieving the github repo containing the script to use to patch our image: You can see that this is the link mentioned previously if you wanted to do this manually.

```
+ cd /home/imageprep
+ git clone https://github.com/F5Networks/f5-openstack-image-prep.git
Cloning into 'f5-openstack-image-prep'...
```

```
2016-07-09 22:22:38 (5.49 MB/s) - 'f5_ve_image.zip' saved [1131226575/1131226575]
+ unzip f5_ve_image.zip BIGIP-11.6.1.0.0.317.qcow2
Archive:  f5_ve_image.zip
  inflating: BIGIP-11.6.1.0.0.317.qcow2
+ python ve_image_sync.py -i BIGIP-11.6.1.0.0.317.qcow2
sudo: unable to resolve host test-image-prep-instance-36cy34o4iqq3
+ '[' -n '' ']'
+ '[' -n '' ']'
+ check_oldfile_full_path
+ '[' -f BIGIP-11.6.1.0.0.317.qcow2 ']'
++ basename BIGIP-11.6.1.0.0.317.qcow2
+ ofname=BIGIP-11.6.1.0.0.317.qcow2
+ '[' -z os_ready-BIGIP-11.6.1.0.0.317.qcow2 ']'
+ cp BIGIP-11.6.1.0.0.317.qcow2 /home/imageprep/os_ready-BIGIP-11.6.1.0.0.317.qcow2
```

Here is the process to patch the image

```
+ qemu-nbd -d /dev/nbd0
+ sleep 2
+ qemu-nbd --connect=/dev/nbd0 /home/imageprep/os_ready-BIGIP-11.6.1.0.0.317.qcow2
+ sleep 2
+ pvscan
+ sleep 2
+ echo 'The following command may cause '\''Can'\''t deactivate'\'' messages.'
+ echo 'These do not necessarily indicate a problem.'
+ vgchange -ay
+ sleep 2
+ mkdir -p /mnt/bigip-config
+ '[' -n '' ']'
+ echo 'Waiting 15 seconds'
+ sleep 15
+ umount /mnt/bigip-config
umount: /mnt/bigip-config: not mounted
+ '[' 32 -eq 1 ']'
+ umount /mnt/bigip-shared
umount: /mnt/bigip-shared: mountpoint not found
+ '[' 32 -eq 1 ']'
++ get_dev set.1._config
++ ls -l /dev/vg-db-hda
++ grep set.1._config
++ cut '-d>' -f2
++ cut -d/ -f2-
ls: cannot access '/dev/vg-db-hda': No such file or directory
+ mount /dev/ /mnt/bigip-config
mount:  /dev is not a block device
+ inject_files
+ '[' -f /home/imageprep/f5-openstack-image-prep/lib/f5_image_prep/startup.tar ']'
+ tar -xf /home/imageprep/f5-openstack-image-prep/lib/f5_image_prep/startup.tar -C /mnt/bigip-config/
+ true
+ touch /mnt/bigip-config/firstboot
```

```
+ touch /mnt/bigip-config/firstboot
+ '[' -f none ']'
+ '[' -n '' ']'
+ '[' -n '' ']'
+ sleep 2
+ umount /mnt/bigip-config
umount: /mnt/bigip-config: not mounted
+ sleep 2
+ '[' -n '' ']'
+ sleep 2
+ vgchange -an
+ sleep 2
+ qemu-nbd -d /dev/nbd0
ve control failed (result -32)
+ echo 'Patched image located at /home/imageprep/os_ready-BIGIP-11.6.1.0.0.317.qcow2'
```

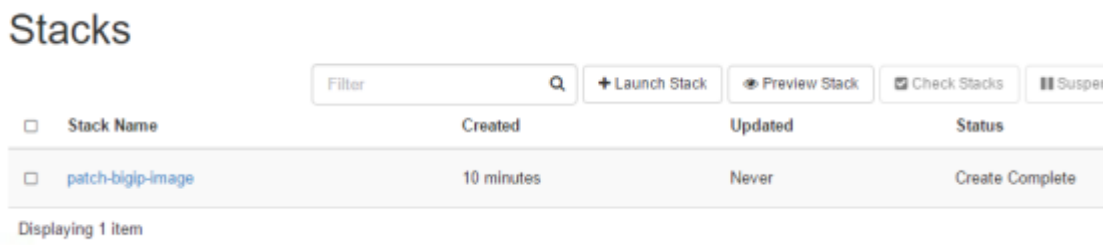
Loading the image in Glance

```
[  405.393195] cloud-init[1162]: Patching image...
[  405.395460] cloud-init[1162]: Uploading patched image to glance...
[  405.399649] cloud-init[1162]: Image Model:
[  405.403449] cloud-init[1162]: <Image {u'status': u'active', u'created_at': u'2016-07-10T18:04:06.000000', u'virtu
u'container_format': u'bare', u'min_ram': 0, u'disk_format': u'qcow2', u'updated_at': u'2016-07-10T18:04:34.000000',
u'bfe146f6215e992ff7076b5e2e55d264', u'owner': u'0b02c277107a4c318c72ed7f60227413', u'is_public': False, u'deleted_a
[  405.478208] cloud-init[1162]: + curl -i -X POST -H 'X-Auth-Token: ee4bc2e98cca4bd58138e2c3a87e821e' -H 'Content-T
http://10.1.10.10:8004/v1/0510aa17707c4583a41e1913841248bf/stacks/patch-bigip-image/d666dbab-1296-41d6-a42f-cb2203f5
[  405.718962] cloud-init[1162]:   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
[  405.722613] cloud-init[1162]:                                  Dload  Upload   Total   Spent    Left  Speed
[  406.218970] cloud-init[1162]:   0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0100
25  100     4  100    21      7     41 --:--:-- --:--:-- --:--:--    41
[  406.219569] cloud-init[1162]: HTTP/1.1 200 OK
[  406.222590] cloud-init[1162]: Content-Type: application/json; charset=UTF-8
[  406.224492] cloud-init[1162]: Content-Length: 4
[  406.226615] cloud-init[1162]: X-Openstack-Request-Id: req-06100c01-29e2-4a59-95fe-c85ea4203214
[  406.235003] cloud-init[1162]: Date: Sun, 10 Jul 2016 18:04:35 GMT
```

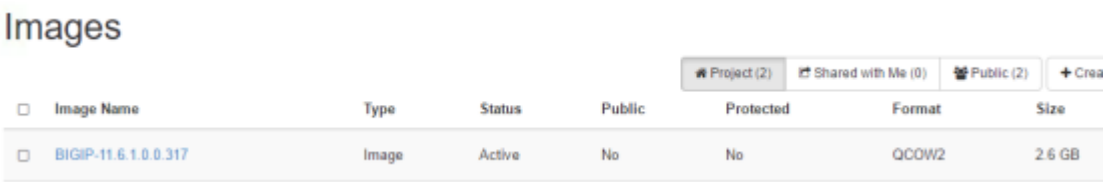
In the default HEAT template, you have a lots of logging enabled as shown in the previous screenshot (apt-get logs, wget process). However, in this nested environment, this is killing the cpu

(disk I/O) and therefore the heat teamplate has been updated to be quiet on those transactions (qq option for apt-get and –quiet for wget). This improve drastically the time needed to process this template (from hours to minutes). We also updated the command pvscan for pvscan –cache otherwise often the mounting of the image to patch would fail.

Once the HEAT template is done you should see this:



and if you go to Project > Compute > Images, you should see your BIG-IP image.



You can delete the stack to remove the instance that is no longer required.

Lab – Deploy F5 VE via HEAT

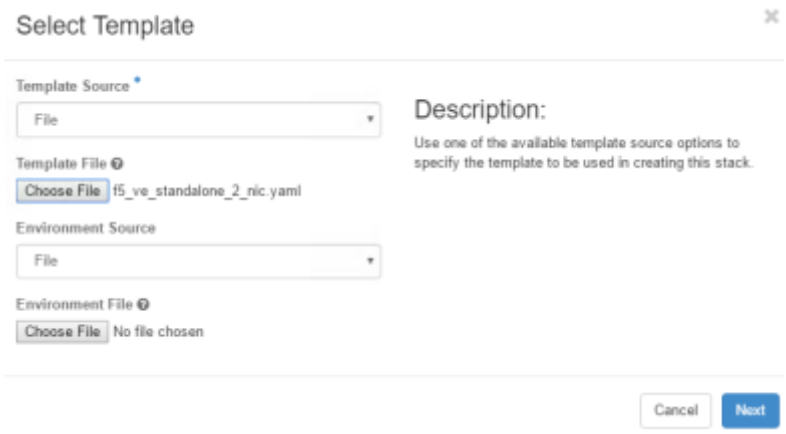
Now that we have a patched BIG-IP image in our environment, we can try to deploy it via HEAT.

To do so, we will use the following HEAT template:
https://github.com/F5Networks/f5-openstack-heat/blob/kilo/f5_supported/ve/standalone/f5_ve_standalone_2_nic.yaml

It is already available in your Desktop > HEAT Templates Folder.

Retrieve a license key for the VE that we will deploy

Create a new stack using this file and launch it.



Launch Stack

Stack Name *

deploy-VE

Creation Timeout (minutes) *

60

Rollback On Failure

Password for user "admin" *

•••••

F5 VE Image *

BIGIP-11.6.1.0.0.317 (2616.6PB)

F5 VE Flavor

m1.large

Use Config Drive

F5 FW Root SSH Key Name *

AdminKey

F5 VE Admin User Password *

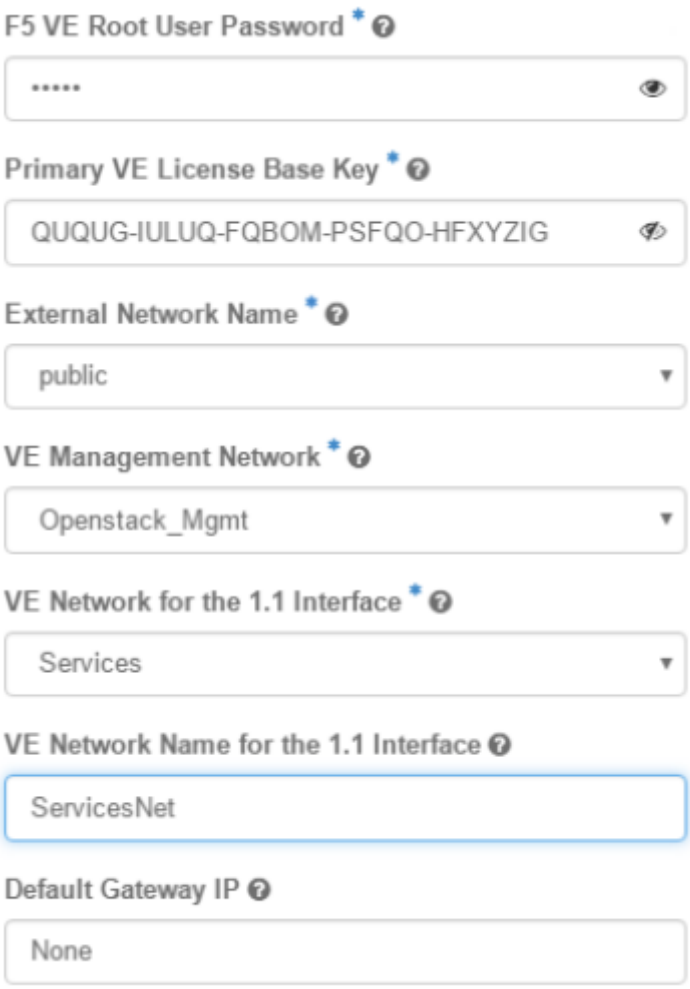
•••••

F5 VE Root User Password *

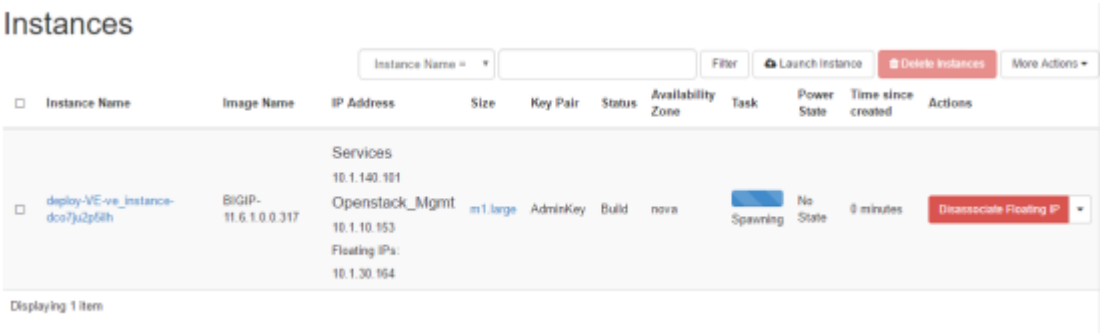
•••••

Description

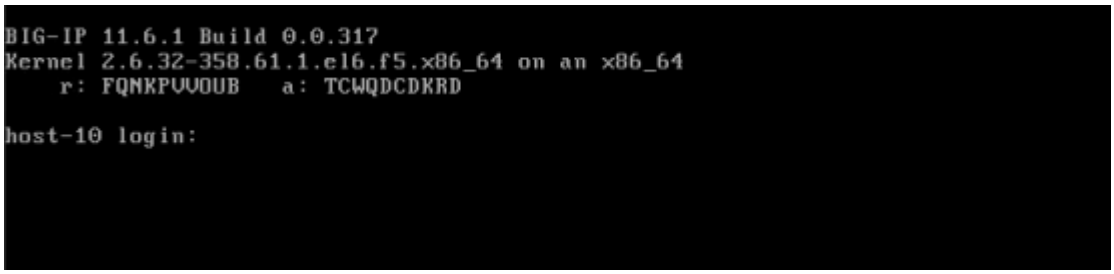
Create a new stack



You should see a new instance in your Project > Compute > Instances page



You can monitor it's progress by clicking on the instance name and check the console tab.



You can now try to access your big-ip from your RDP session on the 10.1.10.XX IP that you're BIG-IP received (the use of the floating is not needed with the architecture we have). Don't forget to use your admin private key for authentication or the password you specified

Once you're connected, you can review the updated startup script (vi /config/startup)

```
[root@host-10:INOPERATIVE:Standalone] config # cat /config/startup
#!/bin/bash

# Copyright 2015-2016 F5 Networks Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

source /config/os-functions/openstack-datasource.sh
source /config/os-functions/openstack-license.sh
source /config/os-functions/openstack-network.sh
source /config/os-functions/openstack-ssh.sh
source /config/os-functions/openstack-password.sh

function restore_issue() {
    cat /etc/issue | head -n 2 > /etc/issue
}

function force_platform_id() {
        log "Forcing /PLATFORM..."
        printf 'platform=Z100\nfamily=0xC000000\nhost=Z100\nsystype=0x71\n
}
```

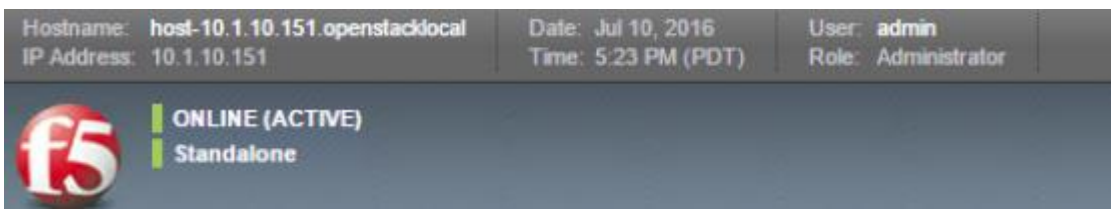
All the action triggered by the startup script are logged into /var/log/messages

```
Jul 10 17:15:57 host-10 notice openstack-init: Saving Ethernet mapping...done
Jul 10 17:15:58 host-10 notice openstack-init: Licensing BIG-IP using license key QUQUG-IULUQ-FQBOM-PSFQO-HFXYZI
...
Jul 10 17:16:08 host-10 notice setsebool: The allow_dynad policy boolean was changed to on by root
Jul 10 17:16:21 host-10 notice openstack-init: Successfully licensed BIG-IP using user-data from instance metada
a...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for LTM module...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for ASM module...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for APM module...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for APM module...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for AFM module...
Jul 10 17:16:22 host-10 notice openstack-init: Found license for AFM module...
Jul 10 17:16:34 host-10 notice openstack-init: Successfully provisioned LTM with level nominal...
Jul 10 17:16:35 host-10 notice openstack-init: Successfully provisioned ASM with level none...
Jul 10 17:16:35 host-10 notice openstack-init: Successfully provisioned APM with level none...
Jul 10 17:16:45 host-10 notice openstack-init: Successfully provisioned AFM with level none...
Jul 10 17:16:45 host-10 notice openstack-init: Successfully provisioned AFM with level none...
Jul 10 17:16:47 host-10 notice openstack-init: Successfully provisioned AFM with level none...
Jul 10 17:17:00 host-10 notice logger: Started writing core file: /var/core/tmm.0.bld0.0.317.core.gz for PID 181
9
Jul 10 17:17:07 host-10 notice openstack-init: Successfully connected to mcpd...
Jul 10 17:17:10 host-10 notice openstack-init: Successfully connected to mcpd...
Jul 10 17:17:10 host-10 notice openstack-init: detected tmm started
Jul 10 17:17:16 host-10 notice openstack-init: Saving running configuration...
Jul 10 17:17:16 host-10 notice openstack-init:   /config/bigip.conf
Jul 10 17:17:16 host-10 notice openstack-init:   /config/bigip_base.conf
Jul 10 17:17:16 host-10 notice openstack-init:   /config/bigip_user.conf
Jul 10 17:17:17 host-10 notice openstack-init: Saving Ethernet mapping...done
Jul 10 17:17:24 host-10 notice openstack-init: Completed OpenStack auto-configuration in 255 seconds...
Jul 10 17:18:46 host-10 notice logger: Finished writing 95933304 bytes for core file: /var/core/tmm.0.bld0.0.317
core.gz for PID 18179
Jul 10 17:19:09 host-10 notice setsebool: The allow_dynad policy boolean was changed to on by root
Jul 10 17:19:42 host-10 notice syslog-ng[13097]: Configuration reload request received, reloading configuration;
```

Once the startup script is done you should have a fully functional BIG-IP

```
[root@host-10:Active:Standalone] config #
```

You can access it also via your browser



Check if the network has been setup as expected