Private Cloud Test-Drive

Test-Drive Guide

(Cloud Infrastructure)

Prepared by:

Roberto Calva - LATAM Management & Automation Specialist SA

Vagner Farias - LATAM Cloud Infrastructure Specialist SA

Creation date:	**06/28/18**
Date of last modification:	**06/02/19**
Version:	**1.1**

Lab 1: Self Service VM Creation

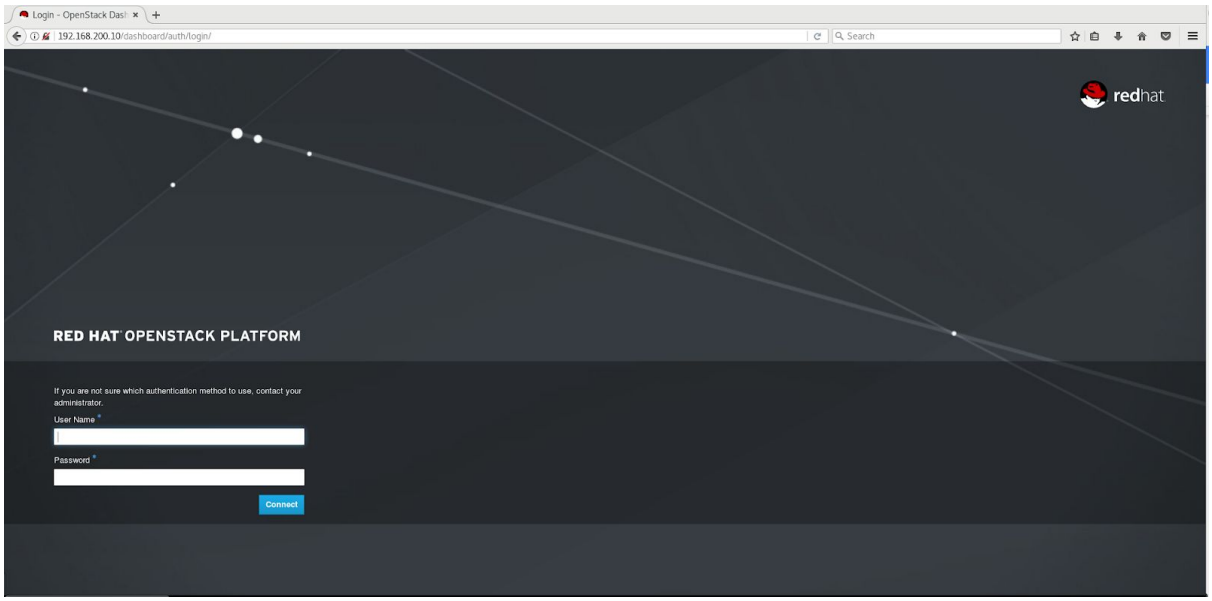
This first lab aims to provide an overview of the OpenStack Dashboard (Horizon) and you'll learn to:

- Create virtual machines (commonly called virtual instances in cloud environments) connected to an existing network.
- Access the console of a virtual instance.

Details on how to manage projects, users, networks and volumes will be covered in subsequent labs.

OpenStack Dashboard Overview

To access the OpenStack Dashboard, use the URL provided to you (eg. https://openstack-GUID.rhpds.opentlc.com).



Use the following credentials to login:

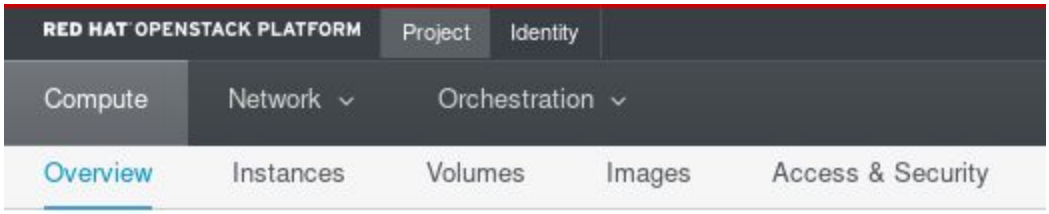
Login	td-user
Password	r3dh4t1!

This user will allow you to access a project called "test-drive" with a "_member_" role. This essentially means that this user is not an OpenStack administrator and is unable to

manage user permissions, create new projects or manage quotas, among other administrative tasks.

Inside the "test-drive" project, however, this user can manage virtual instances, networks, routers, images, volumes.

First thing you may do is to explore the dashboard. In the top left section of the screen you have two tabs: Project and Identity.



As you're logged as a "_member_", the *Identity* tab will only show information about your user. OpenStack administrators can manage users and projects.

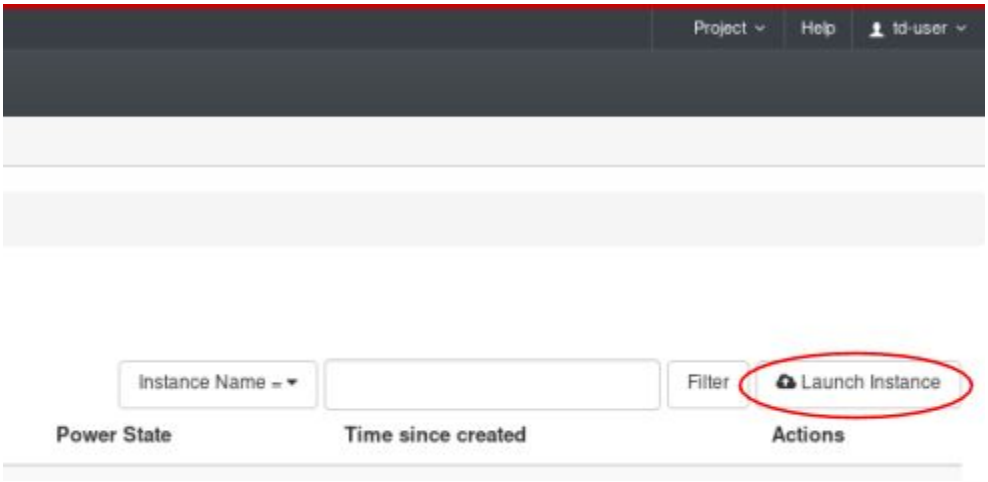
In the Project tab you can access several other sections that will allow you to execute several tasks in the environment, as summarized in the following table.

Section	Description
Compute	Provides an overview of used and available resources, considering user quota. It's also the section where you'll be able to create new instances, volumes, images or key pairs as well as manage existing ones.
Network	Provides a network topology diagram and allows you to manage networks, routers, security groups and floating IPs.
Orchestration	Provides an interface to the Orchestration service (Heat), which as the name implies allows orchestrating the creation/association of OpenStack resources to deploy applications based on a template.

At the top right corner of the screen you would be able to switch between projects if there were other projects you had access to, which is not the case at this moment. Also, clicking in your username you have the option to change some User Interface settings or your password. It's also through this menu that you may logout from the dashboard.

Create a new instance

To create a new instance go to Project → Compute → Instances and click on the "Launch Instance" button as shown in the diagram below.

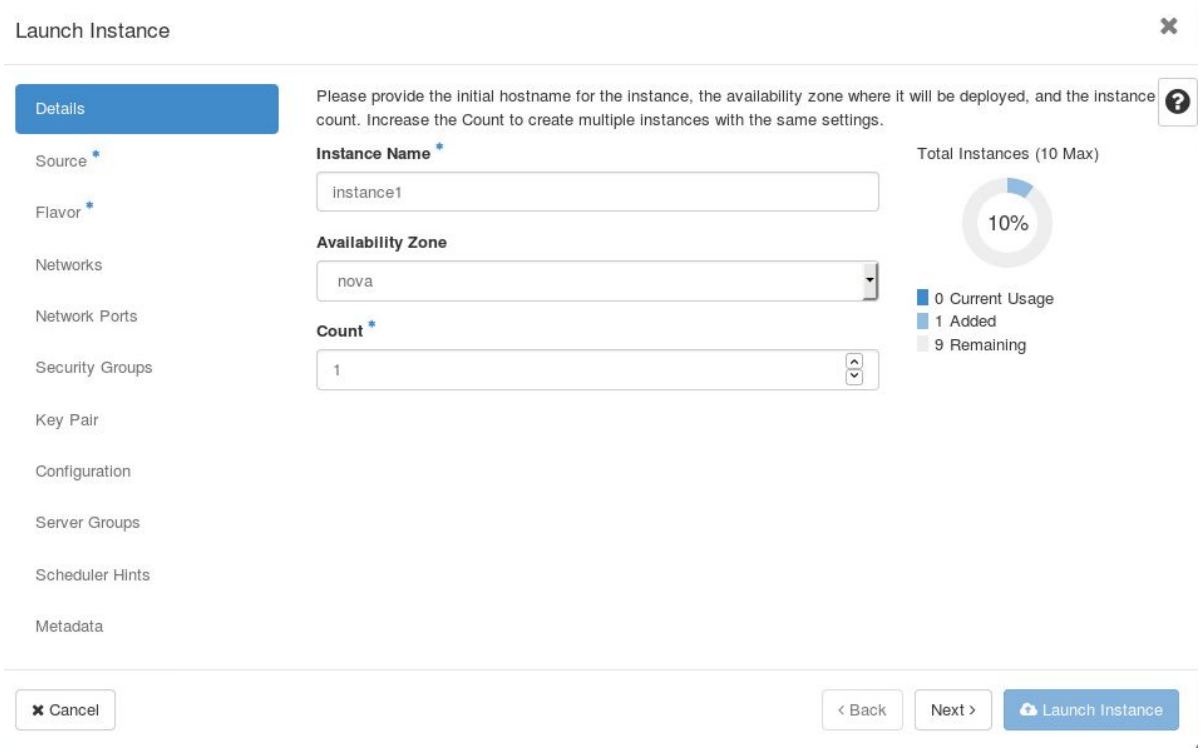


The Launch Instance window will be opened and you need to provide at least the information marked with a "*". For this lab, you should use the following information:

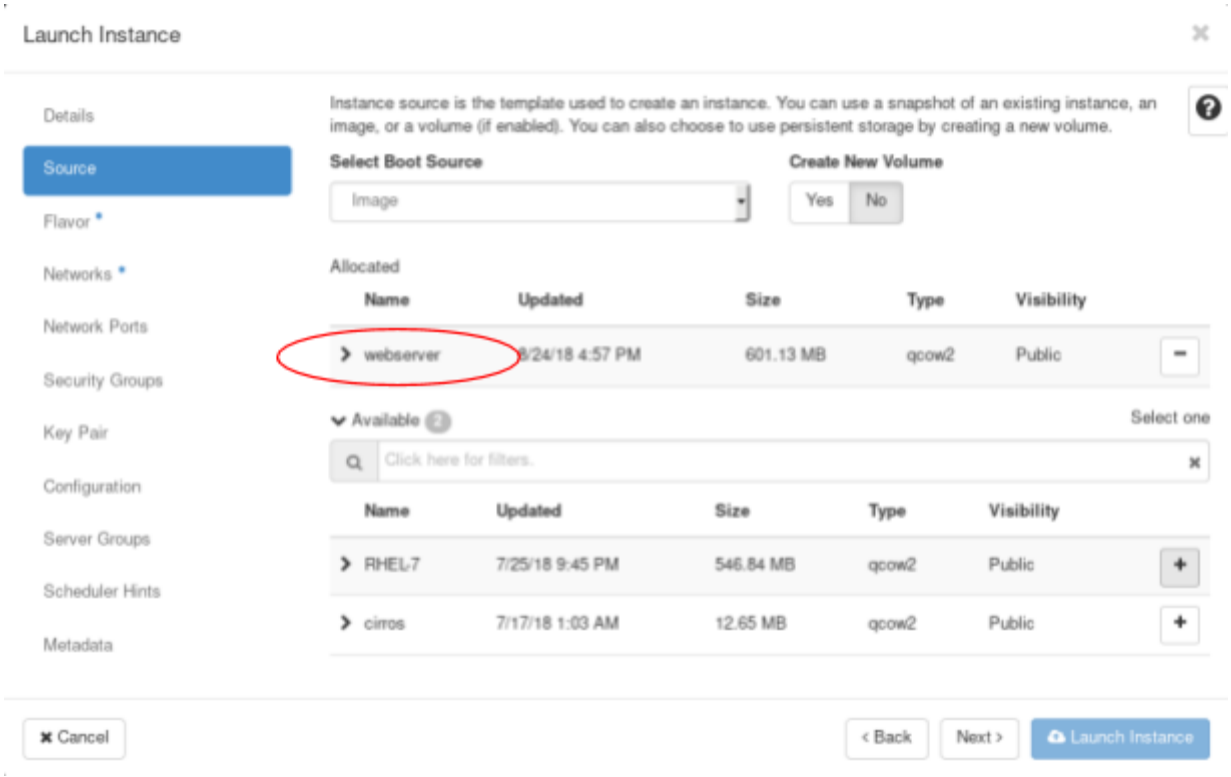
Instance name	instance1
Availability Zone	nova
Count	1
Select Boot Source	Image
Create New Volume	No
Image	webserver
Flavor	m1.small
Network	private

The following images demonstrate the process.

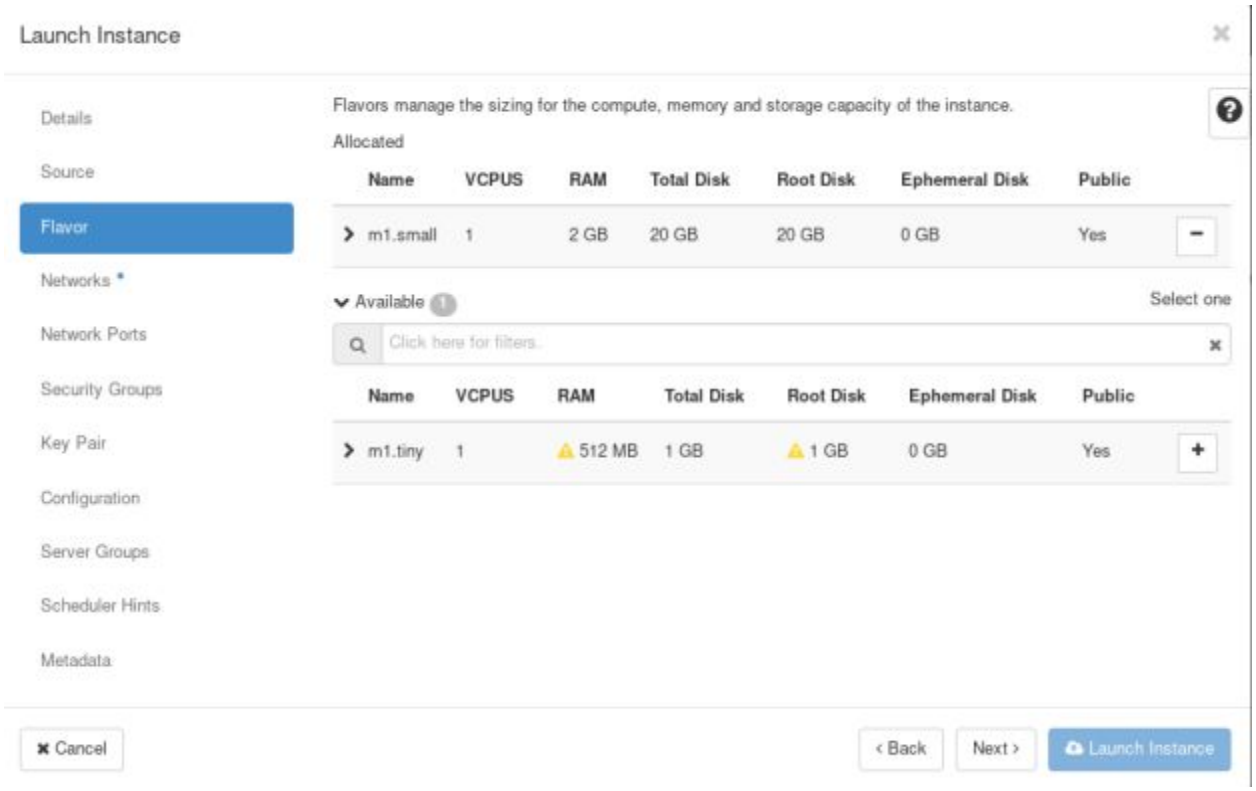
Details tab


Launch Instance
Details
Source *
Flavor *
Networks
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.
Instance Name *
instance1
Availability Zone
nova
Count *
1
Total Instances (10 Max)
10%
0 Current Usage
1 Added
9 Remaining
Cancel
Back
Next
Launch Instance

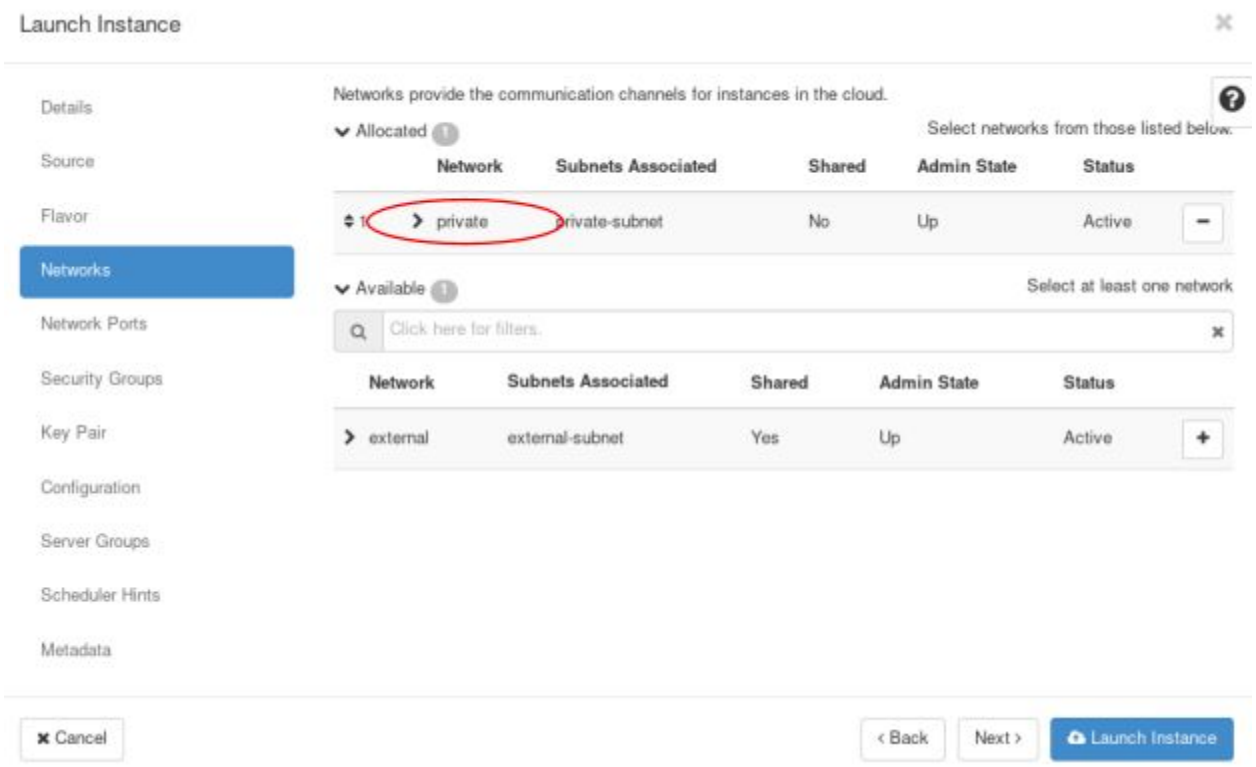
Source tab


Launch Instance
Details
Source
Flavor *
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Instance source is the template used to create an instance. You can use a snapshot of an existing instance, an image, or a volume (if enabled). You can also choose to use persistent storage by creating a new volume.
Select Boot Source
Image
Create New Volume
Yes
No
Allocated
Name
Updated
Size
Type
Visibility
webserver
8/24/18 4:57 PM
601.13 MB
qcow2
Public
Available 2
Select one
Click here for filters.
Name
Updated
Size
Type
Visibility
RHEL-7
7/25/18 9:45 PM
546.84 MB
qcow2
Public
cirros
7/17/18 1:03 AM
12.65 MB
qcow2
Public
Cancel
Back
Next
Launch Instance

Flavor tab


Launch Instance
Details
Source
Flavor
Networks *
Network Ports
Security Groups
Key Pair
Configuration
Server Groups
Scheduler Hints
Metadata
Flavors manage the sizing for the compute, memory and storage capacity of the instance.
Allocated
Name
VCPUS
RAM
Total Disk
Root Disk
Ephemeral Disk
Public
m1.small
1
2 GB
20 GB
20 GB
0 GB
Yes
Available 1
Select one
Click here for filters.
Name
VCPUS
RAM
Total Disk
Root Disk
Ephemeral Disk
Public
m1.tiny
1
512 MB
1 GB
1 GB
0 GB
Yes
Cancel
Back
Next
Launch Instance

Networks tab



After the instance creation finishes you'll be able to see it in the listing at Project → Compute → Instances. At this point, you can use the Actions menu and access the instance console.

To log into this instance, use:

Login	root
Password	r3dh4t1!

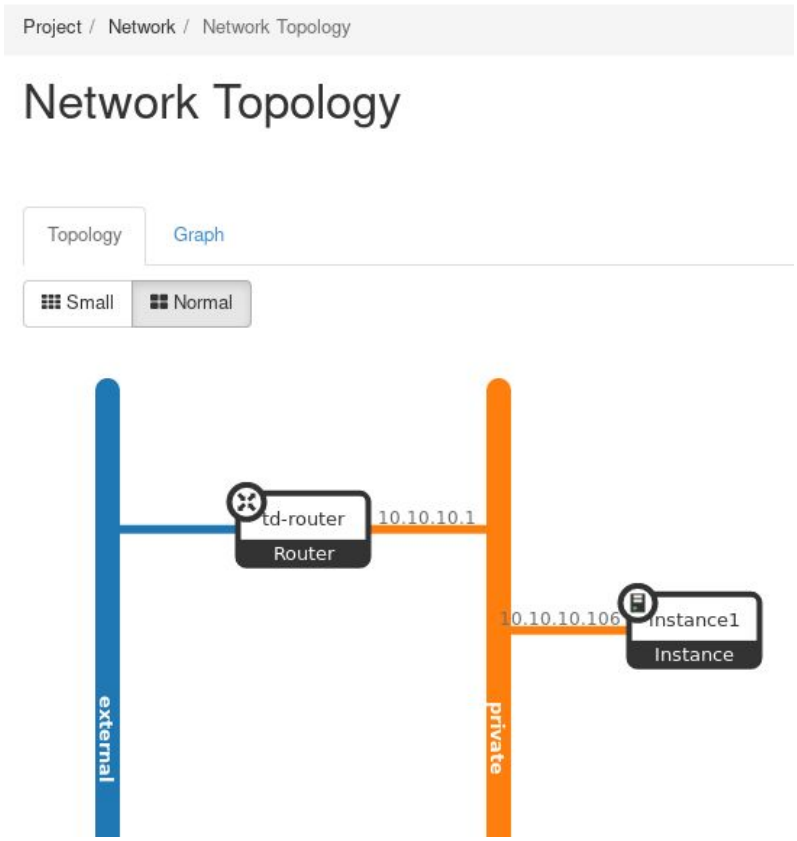
Test connectivity to the virtual router (10.10.10.1) and to an Internet address (eg. 8.8.8.8)

```
[root@instance1 ~]# ping -c3 10.10.10.1
PING 10.10.10.1 (10.10.10.1) 56(84) bytes of data.
64 bytes from 10.10.10.1: icmp_seq=1 ttl=64 time=1.64 ms
64 bytes from 10.10.10.1: icmp_seq=2 ttl=64 time=0.768 ms
64 bytes from 10.10.10.1: icmp_seq=3 ttl=64 time=0.489 ms

--- 10.10.10.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.489/0.966/1.641/0.490 ms
[root@instance1 ~]#
[root@instance1 ~]# ping -c3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=121 time=4.84 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=121 time=1.82 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=121 time=1.96 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.821/2.876/4.844/1.393 ms
[root@instance1 ~]#
```

You can check the network topology at Project → Network → Network Topology.



Create a second instance

In order to test connectivity between two instances, launch a new one (Project → Compute → Instances → Launch Instance), using the following information:

Instance name	**Instance2**
Availability Zone	nova
Count	1
Select Boot Source	Image
Create New Volume	No
Image	webserver
Flavor	m1.small
Network	private

From the Instances list, take note of the IP address of **instance1**. After that, connect to **instance2** console, log into it and ping the IP address of **instance1** and the router (10.10.10.1).

Lab 2: External Inbound Access to Instances

In this lab you'll learn:

- How to use floating IPs to allow inbound access to virtual instances.

Virtual Instance IP addresses

In order to allow inbound access to a virtual instance connected to a virtual private network it's required to associate a floating IP to it. At this time, it's important to differentiate Fixed IPs and Floating IPs.

Fixed IP Address

When an instance is created it receives at least one IP address, which is known as the *fixed IP* and is provided by DHCP[1] according to the network to which the virtual NIC is connected to. The number of fixed IPs the instance will have depends on the number of virtual NICs configured.

This IP address is visible from within the instance (eg. you can see it running "ip address show") and is typically part of a virtual private network.

Floating IP Address

Floating IP addresses are used to allow external inbound access to instances connected to virtual private networks. The floating IPs are created on an *external network* and when it's attached to a virtual instance, neutron will configure a Destination NAT (DNAT) from the external IP address to the fixed IP address used by the instance.

Virtual instances don't have any information about floating IP addresses and the address translation will be handled by neutron L3 agent (in the case of lab environment).

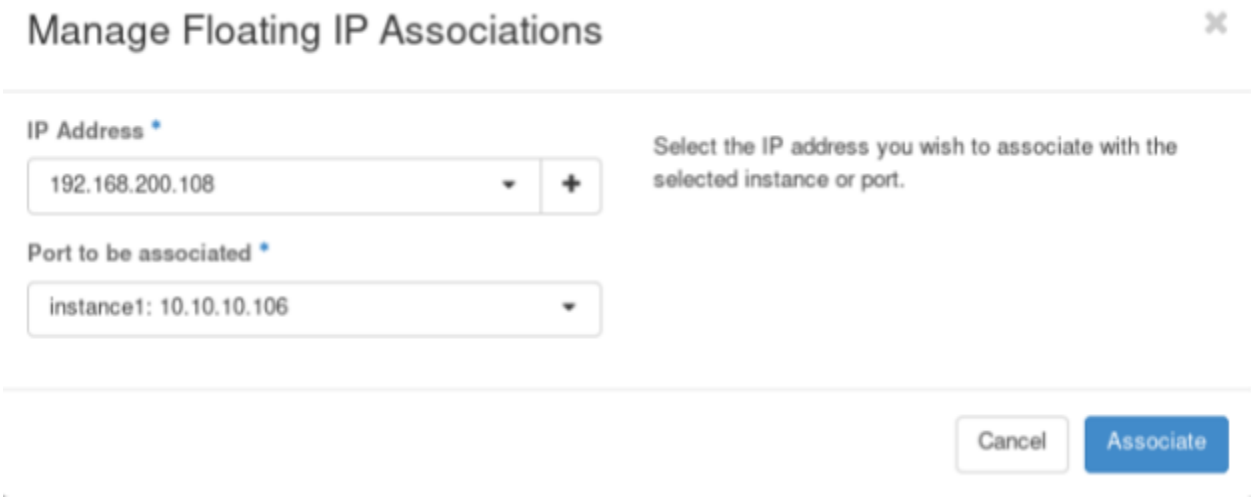
Associate Floating IP to instance

1. Log in as td-user.
2. Go to Projects → Compute → Instances.

[1] Actually IPv6 may use SLAAC to configure IP addresses.

3. In “instance1” Actions menu, select “Associate Floating IP”

4. In the next dialog, select one of the available IP addresses and click on “Associate”



“Port to be associated” sets to which IP address the DNAT configuration should be done. In this case, the instance has only one port/IP address.

5. Repeat the process for *instance2*.

⚠ Only Floating IP addresses **192.168.200.106** and **192.168.200.108** will be externally accessible.

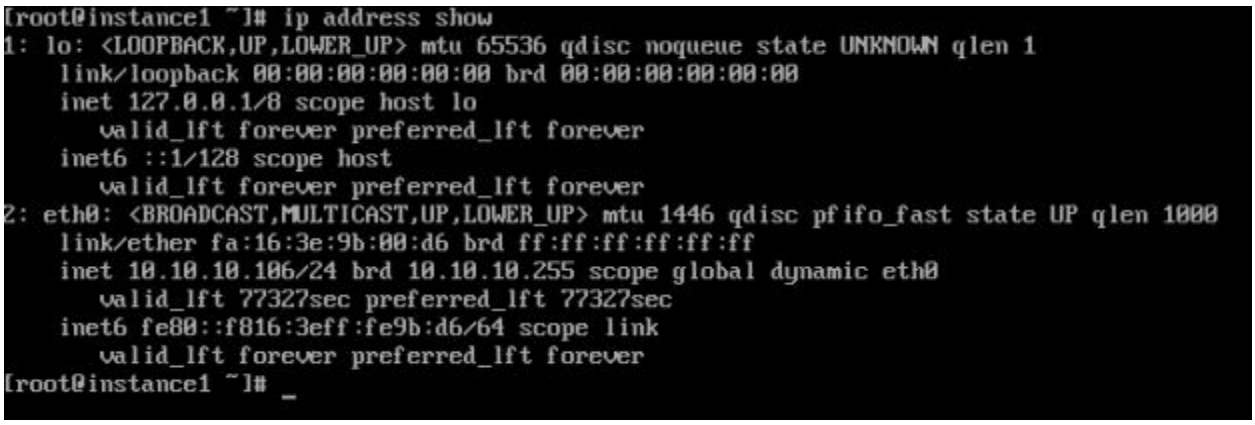
When you finish this procedure, you'll be able to see the floating IP associated to the instance, in addition to its fixed IP (Project → Compute → Instances):

⚠ Take note of the floating IP associated to your instance as you'll use it in the next step.

Check Instance IP Addresses

To confirm the floating IP address is unknown by the virtual instance, access the instance console using the following steps:

1. Through the OpenStack Dashboard, go to Projects → Compute → Instances.
2. From the instance "Actions" menu, select "Console".
3. Run "ip address show" and check only the fixed IP address is known by the instance.



Test External Inbound Access

In this environment, each of the two pre-configured IP addresses is associated to a resolvable hostname. Using other floating IP addresses won't work as these are the only ones with mapping configured.

192.168.200.106	floating106-GUID.rhpds.opentlc.com
192.168.200.108	floating108-GUID.rhpds.opentlc.com

Although the instance was configured with a floating IP, it won't be possible to access any service on it as the OpenStack security groups are preventing such access. This will be covered on the next lab.

Lab 3: Security Groups

In this lab you'll learn:

- What are security groups.
- How to use security groups to allow or block access to instances.

Overview of Security Groups

In a nutshell, security groups act as a firewall at the instance level, controlling both inbound and outbound access.

In this case, "at the instance level" doesn't mean to be running a firewall *inside* the virtual instance operating system which could cause too much overhead at scale, but to apply rules at the virtual switch level. This means there's no need to configure firewall rules inside the virtual instance.

In the reference implementation of security groups, used in this environment and in production clouds, the rules are applied using *iptables* in the compute nodes, directly on the virtual switch ports used by the virtual instance.

This is different from the most common approach in legacy infrastructures where there's a firewall in the perimeter in addition to network switch Access Control Lists (ACLs) controlling access to and from devices.

Information needed to configure security groups are the following:

- Source / destination
- Protocol
- Port

Check Security Groups for Virtual Instance

It's time to understand why you could successfully get *ping* replies from *instance1* but couldn't connect to it using *ssh*.

1. Log into the OpenStack Dashboard as *td-user*.
2. Go to Project → Compute → Instances, click on *instance1*.

3. At the *Overview* tab, look at the Security Groups section, as illustrated below

The highlighted rule allowed the instance to reply to *ping* from any host. Examining each rule in the security group:

ALLOW IPv4 to 0.0.0.0/0	Allows *egress* traffic from the instance *to* any host, to any port, using IPv4.
ALLOW IPv4 from default	Allows *ingress* traffic from any instance that also has the *default* security group applied, using IPv4.
ALLOW IPv4 icmp from 0.0.0.0/0	Allows *ingress* traffic of *icmp* protocol *from* any host, using IPv4.
ALLOW IPv6 to ::/0	Allows *egress* traffic from the instance *to* any host, to any port, using IPv6.
ALLOW IPv6 from default	Allows *ingress* traffic from any instance that also has the *default* security group applied, using IPv6.

Apply Security Group to Virtual Instance

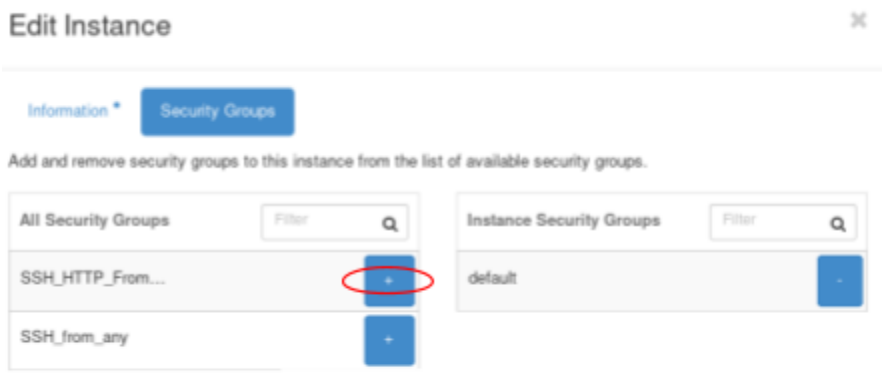
To allow HTTP and SSH access to the instances, apply the security group SSH_HTTP_From_any to them, executing the following steps:

1. Log into the OpenStack Dashboard as *td-user*.
2. Go to Project → Compute → Instances

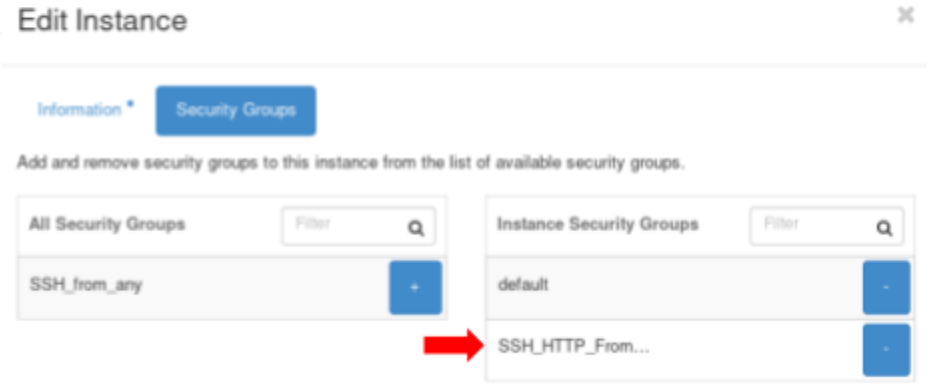
3. In the “Actions” menu for the instance, click on “Edit Security Groups”.

4. In the following dialog, you’ll be able to see all available security groups on the left and all applied security groups on the right. Click on the “+” sign for the “SSH_HTTP_from_any” security group to add it to the instance.

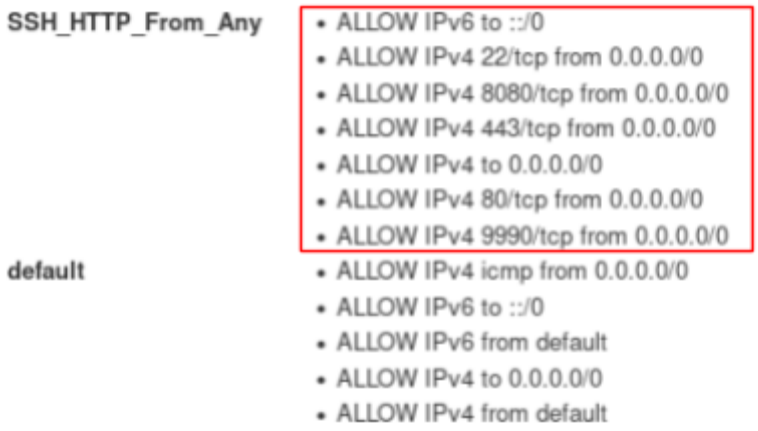
Before



After



5. Check applied security groups going to the Overview tab for the instance (Project → Compute → Instances → Instance1 → Overview).

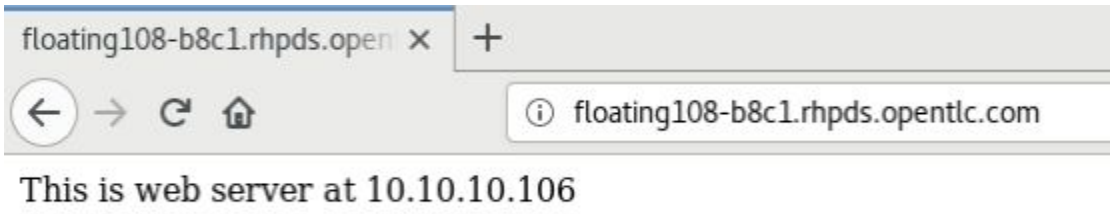


Test HTTP Access From External Host

Identify the floating IP address and access the the corresponding URL in your web browser.

192.168.200.106	http://floating106-GUID.rhpds.opentlc.com
192.168.200.108	http://floating108-GUID.rhpds.opentlc.com

Example:



Lab 4: Persistent Volume Management

In this lab you'll learn:

- What are volumes.
- How to use volumes to manage data life cycle.

Overview of Volumes

A volume is a persistent block storage device that can be attached to or detached from instances. Currently, volumes can only be attached to one instance at a time.

Without volumes, instances only have ephemeral storage provided by the flavor configuration, meaning that all data is lost when an instance are deleted.

Besides, ephemeral storage commonly use local compute nodes disks, so if the node is unavailable, data is unavailable as well. An exception to this is when OpenStack Compute service (nova) is configured to use a shared storage solution, like NFS or Red Hat Ceph Storage.

Volumes can be used as the primary disk for the instance (where the operating system resides) or secondary disks used for storing data.

Volumes are provided by OpenStack Block Storage service, widely known by its project name *cinder*.

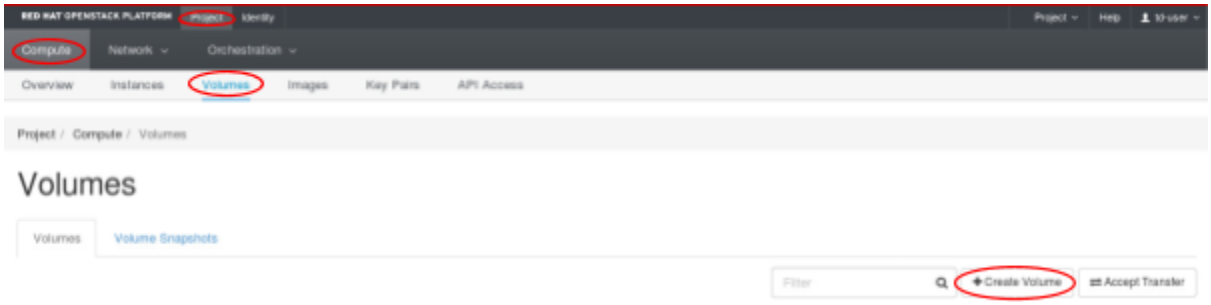
The OpenStack Block Storage service creates an abstraction layer between the real storage device in use and the instances. No matter if the storage device is a SAN, a NAS or a scale-out storage solution like Red Hat Ceph Storage, the operating system inside the instances will see a standard block device.

Volumes for Secondary Disks

Suppose you have an application that needs to store data and that you need to ensure this data won't be lost or made unavailable if there are issues with the compute node. In this case, you can keep the operating system in ephemeral disk and use a secondary disk for data. To experiment, do the following:

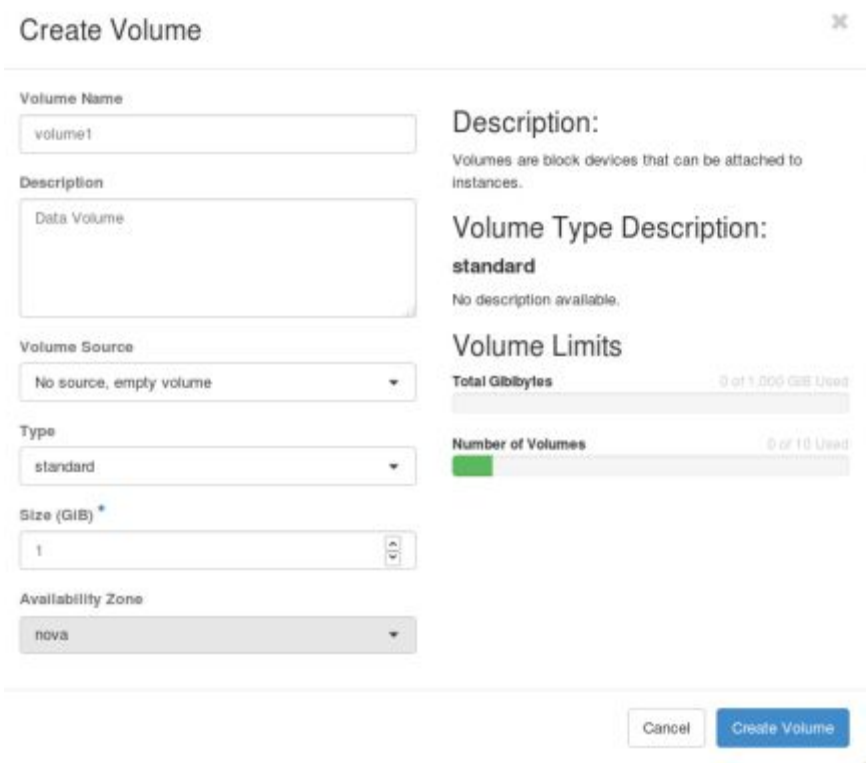
1. Log into the OpenStack Dashboard as td-user.

2. Go to Project → Compute → Volumes and click on “+Create Volume” button.

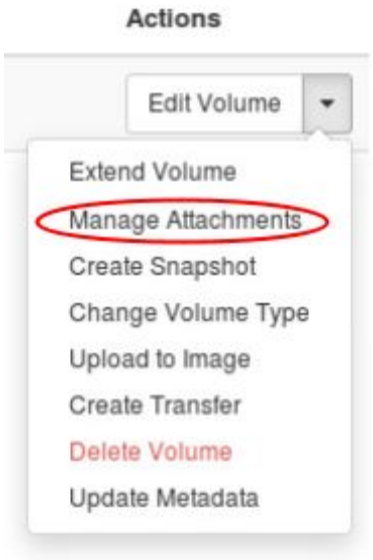


3. Create a volume with the following configuration.

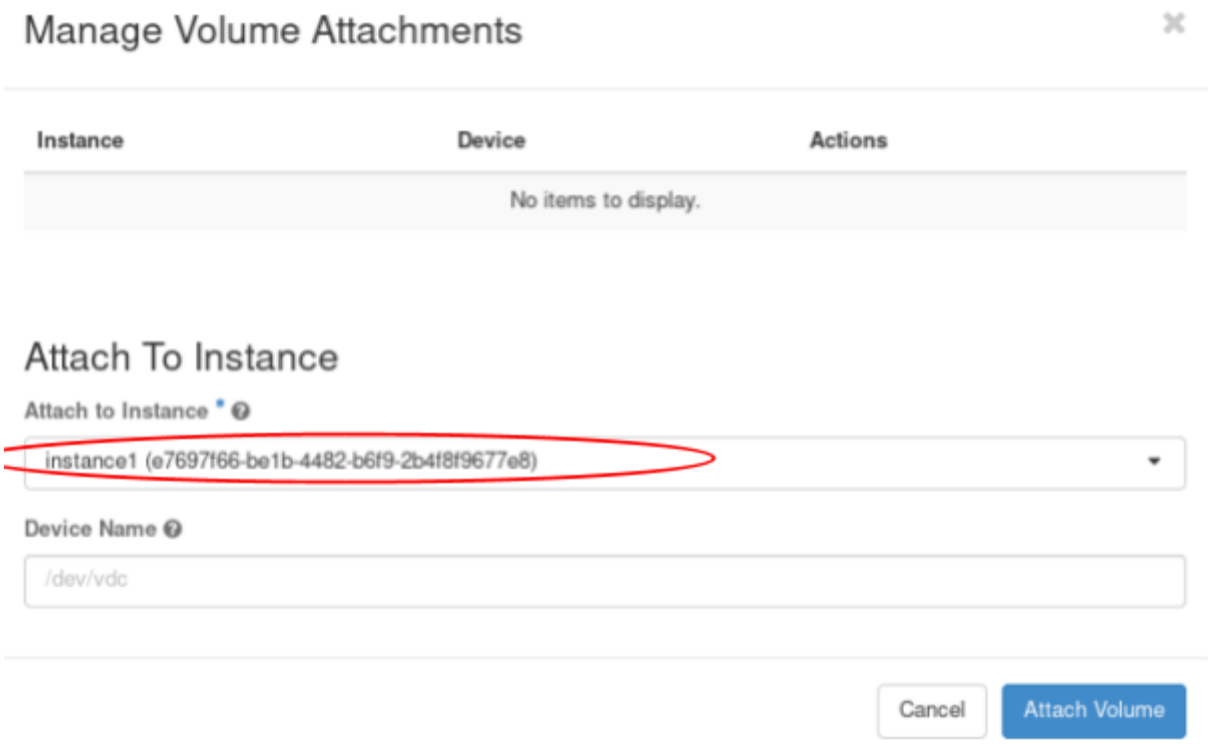
Volume Name	volume1
Description	Data volume
Volume Source	No source, empty volume
Volume Type	standard
Size (GB)	1
Availability Zone	nova



4. From the volume list (Project → Compute → Volumes), select the “Manage Attachments” option in Actions Menu for *volume1*.



5. In the following dialog, select *instance1* in the “Attach to Instance” drop down menu.



6. Observe that the volume is attached to the desired instance.

Name	Description	Size	Status	Type	Attached To
volume1	Data volume	1GiB	In-use	standard	Attached to instance1 on /dev/vdb

7. Access *instance1* console (Project → Compute → Instances → Instance1 → Console) and login as root (password r3dh4t1!).

8. Run the volume-config.sh script (/usr/local/sbin/volume-config.sh)

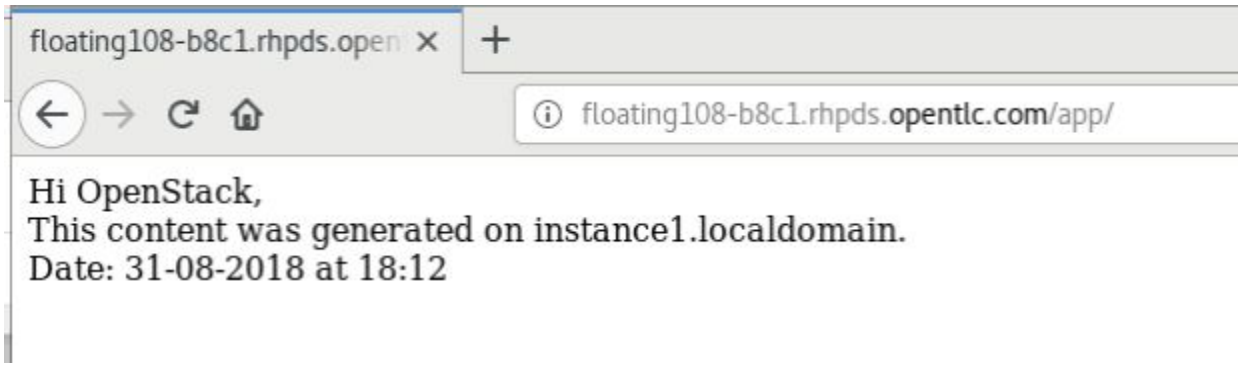
```
[root@instance1 ~]# volume-config.sh
preparing data disk...
[
[

Type your name: OpenStack
done!
[root@instance1 ~]#
```

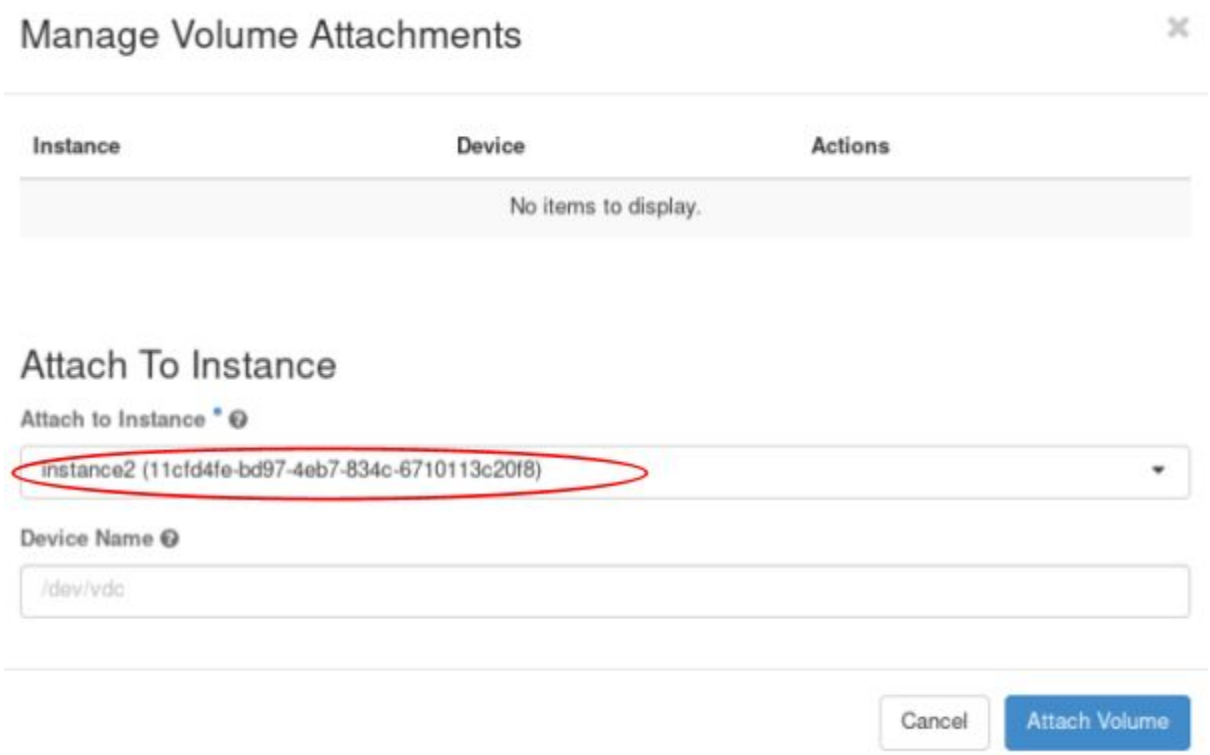
This script was created for this Test Drive and does the following:
If unused disk:
- Create partition
- Create filesystem
- Mount filesystem
- Create an HTML file based on user input

If it finds data in disk:
- Mount filesystem

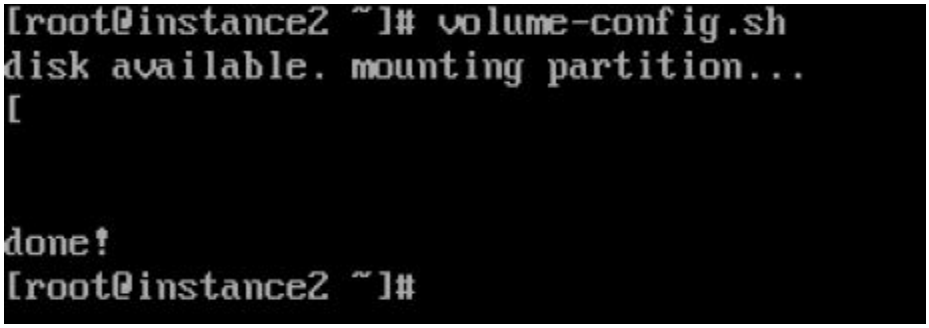
9. Access *instance1* via HTTP. If instance1 has the floating IP 192.168.200.108, use the following URL: http://floating108-GUID.rhpds.opentlc.com/app.



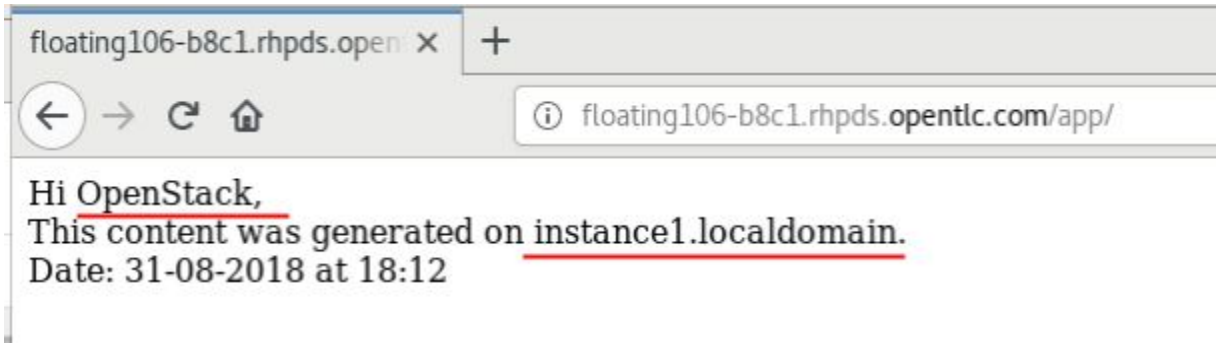
10. From OpenStack Dashboard, **delete instance1** (Project → Compute → Instances → instance1 Actions → Delete Instance).
11. From OpenStack Dashboard (Project → Compute → Volumes, Manage Attachments), attach *volume1* to *instance2*.



12. Access *instance2* console (Project → Compute → Instances → Instance2 → Console) and login as root (password r3dh4t1!).
13. Run the volume-config.sh script (/usr/local/sbin/volume-config.sh)



14. Access *instance2* via HTTP. If instance2 has the floating IP 192.168.200.106, use the following URL: http://floating106-GUID.rhpds.opentlc.com/app.


floating106-b8c1.rhpds.open
floating106-b8c1.rhpds.opentlc.com/app/
Hi OpenStack,
This content was generated on instance1.localdomain.
Date: 31-08-2018 at 18:12

Lab 5: Key Pairs

In this lab you'll learn:

- What is a key pair.
- How to create key pairs in OpenStack.

Overview of Key Pairs

Key pairs are SSH credentials based on public and private keys. Public keys are usually injected to instances when they are created, so that they can be accessed by whoever has the corresponding private key, eliminating the need to configure default passwords on images.

To enable key pair injection to the instances, the most common solution is to have the cloud-init package installed on images, which is true for Red Hat provided images.

Although it is possible to use cloud-init to set password for users, this isn't recommended as it's technically possible for any instance in the same network to retrieve this information.

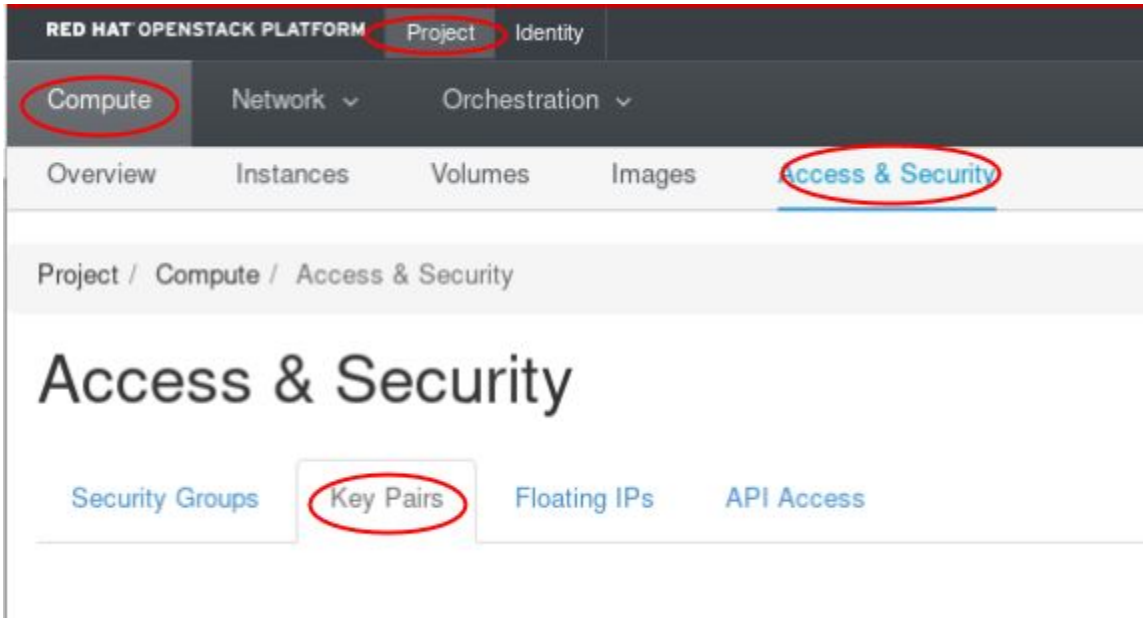
Create a Key Pair

Although it's possible to upload an existing public key to OpenStack, in this lab you'll be creating a new key pair.

To create a key pair, use the following steps.

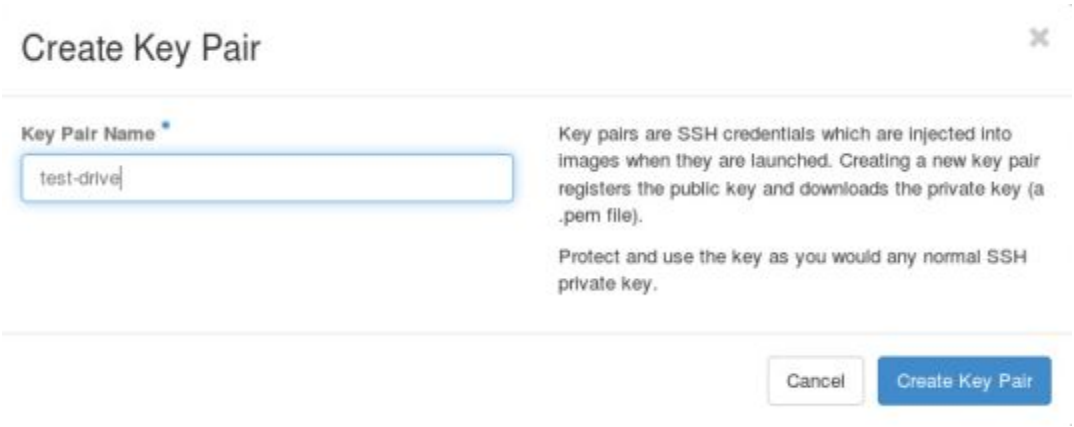
1. Log into the OpenStack Dashboard as *td-user*.

2. Go to Project → Compute → Access & Security → Key Pairs.



3. Click on "+Create Key Pair" button.

4. In the next dialog, enter "test-drive" as the name for the Key Pair and click on the Create Key Pair button.



5. Proceed to save the *test-drive.pem* file in your local machine. This is the private key which should be kept secure.

The location where the private key file will be saved depends on your browser configuration.

Use a Key Pair When Creating a RHEL instance

Creating an instance with an associated key pair is done the same way you did before with the exception that you need to ensure the key pair is selected in the corresponding dialog, as follows.

1. Log into the OpenStack Dashboard as *td-user*.
2. Go to Project → Compute → Instances, select *instance1* and *instance2* and **delete both** - this will automatically release the floating IPs.
3. Go to Project → Compute → Instances and click on the "Launch Instance" button.
4. Use the following information to create the instance.

Instance name	**instance3**
Availability Zone	nova
Count	1
Select Boot Source	Image
Create New Volume	No
Image	**RHEL-7**
Flavor	td.small
Network	private
Security Groups	default SSH_from_any
Key Pair	**test-drive**

5. After the instance is launched, associate floating IP **192.168.200.106** to the instance, selecting the corresponding option in the "Actions" menu.

SSH into the instance

As *instance3* instance has a floating IP, any device that has access to network 192.168.200.0/24 should be able to reach this instance. Use the following steps to ssh into the instance.

1. From Linux ssh client:
   ```
   $ ssh -i /path/to/test-drive.pem \
         cloud-user@floating106-GUID.rhpds.opentlc.com
   ```
2. From Windows client with PuTTY:
   a. Download PuTTYgen from here.
   b. Open PuTTYgen.
   c. Click Conversions, then click import key.
   d. Locate test-drive.pem key file, then click open.
   e. You can now save your key as a PPK file by clicking the Save private key button. Name the file **test-drive.ppk.**
   f. Use *floating106-GUID.rhpds.opentlc.com* as the host to connect to.
   g. In Connection → SSH → Auth, click on *Browse* and select test-drive.ppk.
   h. Click on Open.
   i. Login name is *cloud-user*.

Lab 6: Multi-tenancy

In this lab you'll learn:

- How the use of OpenStack Projects enable multi-tenancy

Overview of Multi-tenancy

In a multi-tenant environment, resources like compute, networking and storage are isolated among different users. OpenStack implements this isolation through *projects*, allowing different organizations, customers or accounts, to use the same shared environment. This is managed by the Identity Service, known by the project name keystone.

Each project has its own configured quota for several kind of infrastructure resources, like vCPUs, RAM, disk space, number of persistent volumes, floating IP addresses, etc. This allows the administrator to restrict how much resources each project will use.

In order to have access to some resource, the user must have access to the project where that resource belongs, unless the resource is public or shared, like some networks or images.

Besides, users may have different roles. Default Red Hat OpenStack Platform roles follows:

admin	OpenStack administrator with access to all operations. Can manage projects and users.
member	Regular user that can create/delete instances, create/delete project networks, create/delete volumes on projects he has access to.
heat_stack_user	Regular user that can use heat orchestration service to manage resources.

Identity V3 included support to domains, which are high-level containers for projects, users and groups. This allows administrators to configure roles for the user at the project or at the domain level, enabling more fine grained Role Based Access Control (RBAC) capabilities. For example, it's possible to configure an administrator that can manage other users and projects for an specific domain.

It's common to use the terms *project* and *tenant* interchangeably.

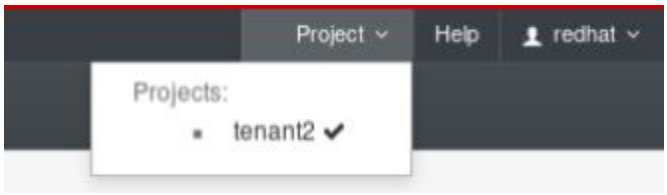
Access a Different Project

In this environment there's a user which is associated to a project different than the "test-drive" being used so far.

1. Login using the following credentials:

Login	redhat
Password	r3dh4t1!

2. Notice on the top right corner that this user sees only a project called "tenant2".



3. Navigate to Project → Compute → Instances and start the existing *instance1*, which although may have the same name of instances of other projects, is a completely different one.
4. Access instance console and log into it using the username "cirros" and the password "cubswin:)", without the quotes. This instance is using a lightweight linux system called CirrOS, used mainly for tests like this.
5. Try pinging 8.8.8.8 or any other Internet public address and notice it won't work, as this project doesn't have a virtual router connecting the private network to the external one.
6. From OpenStack Dashboard, visualize Network Topology (Project → Network → Network Topology). Note how different this is from the "test-drive" tenant.

Lab 7: Provisioning Instances using CloudForms

Red Hat® CloudForms® is an infrastructure management platform that allows IT departments to control users' self-service abilities to provision, manage, and ensure compliance across virtual machines and private clouds.

Provisioning refers to the capacity an infrastructure has to deliver a resource and manage its life cycle. Provisionable resources can include virtual machines, Instances, storage space, or any resource a given infrastructure can manage. The web interface on a *CloudForms* appliance provides an easy way for end users and administrators to provision new virtual machines and Instances. Virtual machines can be provisioned from virtual machine templates, PXE boot, or ISO images. Instances can be provisioned from images.

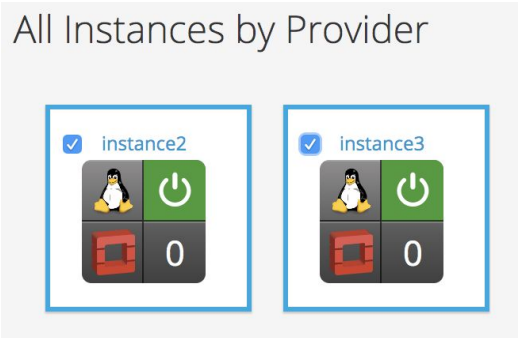
Cleaning Up Our Environment

First, we need to clean up our environment so that we can reuse resources such as CPU, Memory, vDisks and of course ours Floating IPs.

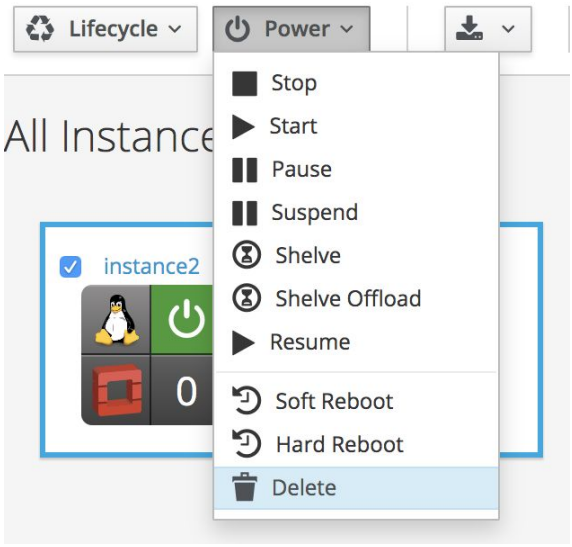
1. Access to **CloudForms Cloud Management Portal**, using the URL provided to you (eg. **https://cloudforms-GUID.rhpds.opentlc.com**)

Login	td-admin
Password	r3dh4t1!

2. Go to: **Compute → Clouds → Instances** and select **instance2** and **instance3**:

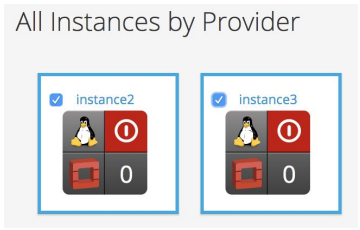


Then click on **Power → Delete**:

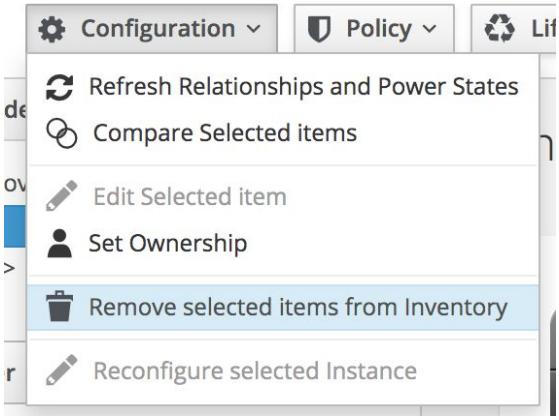


Click on **Refresh** button until both instances are in **Terminate** state .

Select again **instance2** and **instance3**:

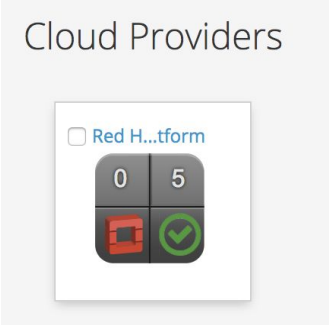


Then click on **Configuration → Remove selected items from inventory**:

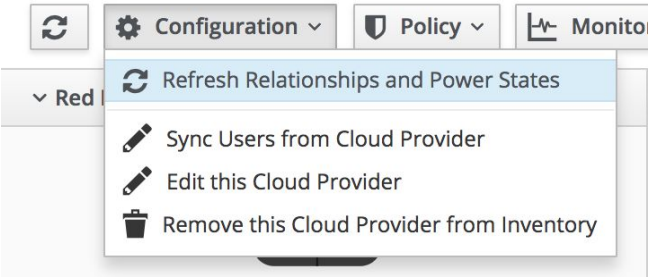


Click again on **Refresh** button until both instances disappears.

Go to **Compute → Clouds → Providers** and then click on **Red Hat OpenStack Platform** provider:



After that, click on **Configuration → Refresh Relationships and Power States** to update all information about our Red Hat OpenStack Provider:



Finally, click on **Refresh** button, until you can see "**Last Refresh: Success - Less Than A Minute Ago**" Status

Red Hat OpenStack Platform (Summary)

Properties	
Region	OpenStack
Host Name	192.168.200.10
Type	OpenStack
API Port	13000
Management Engine GUID	640d6a41-823e-4124-bbee-19c657a3ab72
Region	regionOne

Status	
Default Credentials	Valid
Last Refresh	Success - Less Than A Minute Ago
Last Refresh Date	2018-08-31 20:28:22 -0500

Now we're ready to next Labs!

Creating a New Instance Using CloudForms

1. Access to **CloudForms Cloud Management Portal**, using the URL provided to you (eg. **https://cloudforms-GUID.rhpds.opentlc.com**)

Login	td-admin
Password	r3dh4t1!

2. Go to: **Compute → Clouds → Instances** and click on **Lifecycle → +Provision Instances**

Select the Template **"RHEL-7"** as Image. Click on **Continue** and fill out the information as follows:

Request Tab:

Email	your.email@yourdomain.com
First/Last Name	your name (if you want. Not necessary)

Purpose Tab:

Purpose	Select **Environment → Test**

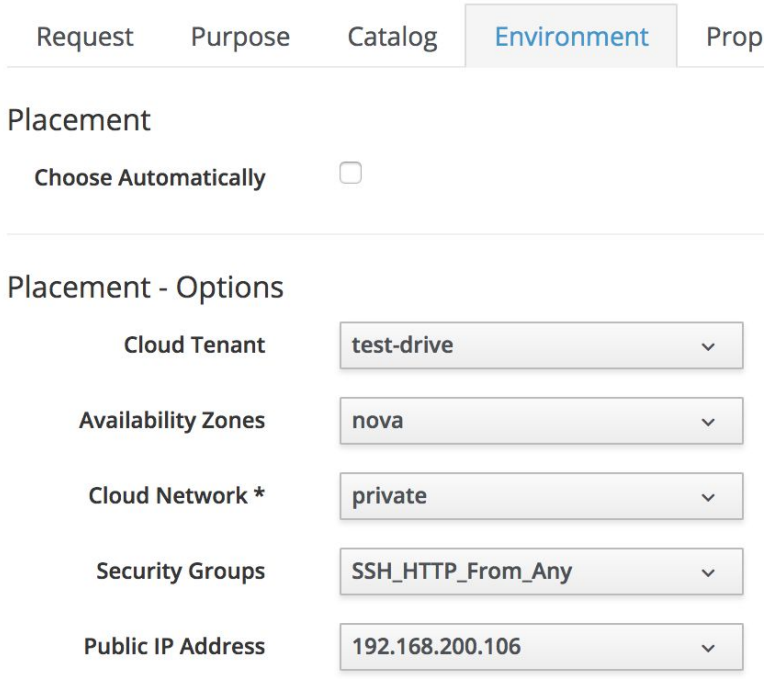
Catalog Tab:

Number of instances	1
Instance Name	wordpress-instance

Environment Tab:

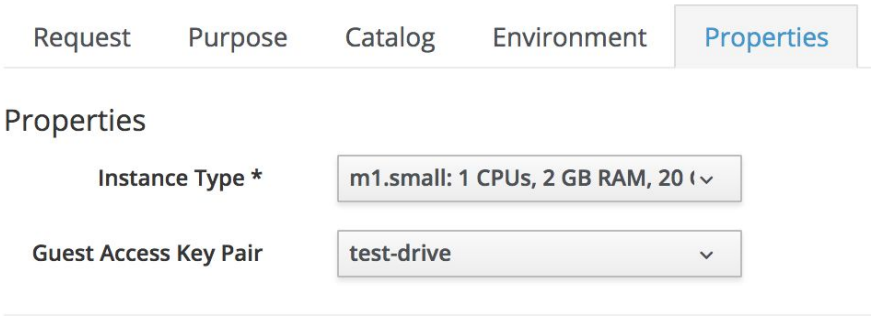
- Set **Placement - Options** as follows:

Cloud Tenant	test-drive
Availability Zones	nova
Cloud Network	private
Security Groups	SSH_HTTP_From_Any
Public IP Address	192.168.200.106



Properties Tab:

Instance Type	m1.small
Guest Access Key Pair	test-drive



Schedule Tab:

Time until Retirement	1 Month
Retirement Warning	1 Week

3. Click on **Submit** to create the new Instance.

4. Switch to **OpenStack Dashboard**, using the URL provided to you (eg. **https://openstack-GUID.rhpds.opentlc.com**)

Observe the deployment of the new OpenStack instance. Click on **Instances → wordpress-instance** to see the instance details:

Login	td-admin
Password	r3dh4t1!

5. Return to **CloudForms Management Portal**. Go to **Services → Requests**. Click on **Reload** button, until the **Last Message "CF_Worker1: VM [wordpress-instance] Provisioned Successfully"** appears and **Request State** is in **Finished** state:

Last Message		Status	Request State	Request ID	Requester
[CF_Worker1] VM [wordpress-instance] IP [10.10.10.112] Provisioned Successfully		Ok	Finished	18	Test-Drive Admin

Lab 8: Configuring Instances Automatically using Ansible Tower

Ansible seamlessly unites workflow orchestration with configuration management, provisioning, and application deployment in one easy-to-use and deploy platform.

Regardless of where you start with Ansible, you'll find our simple, powerful and agentless automation platform has the capabilities to solve your most challenging problems.

Centralizing configuration file management and deployment is a common use case for Ansible, and it's how many power users are first introduced to the Ansible automation platform. In that way, it's very easy to automate the configuration of your bare-metal servers, virtual machines and cloud instances.

When you define your application with Ansible, and manage the deployment with Ansible Tower, teams are able to effectively manage the entire application lifecycle from development to production.

Access to **Ansible Tower Portal**, using the URL provided to you (eg.

https://tower-GUID.rhpds.opentlc.com)

Login	td-admin
Password	r3dh4t1!

Setting Up Credentials in Ansible Tower

Credentials are utilized by Tower for authentication when launching Jobs against machines, synchronizing with inventory sources, and importing project content from a version control system.

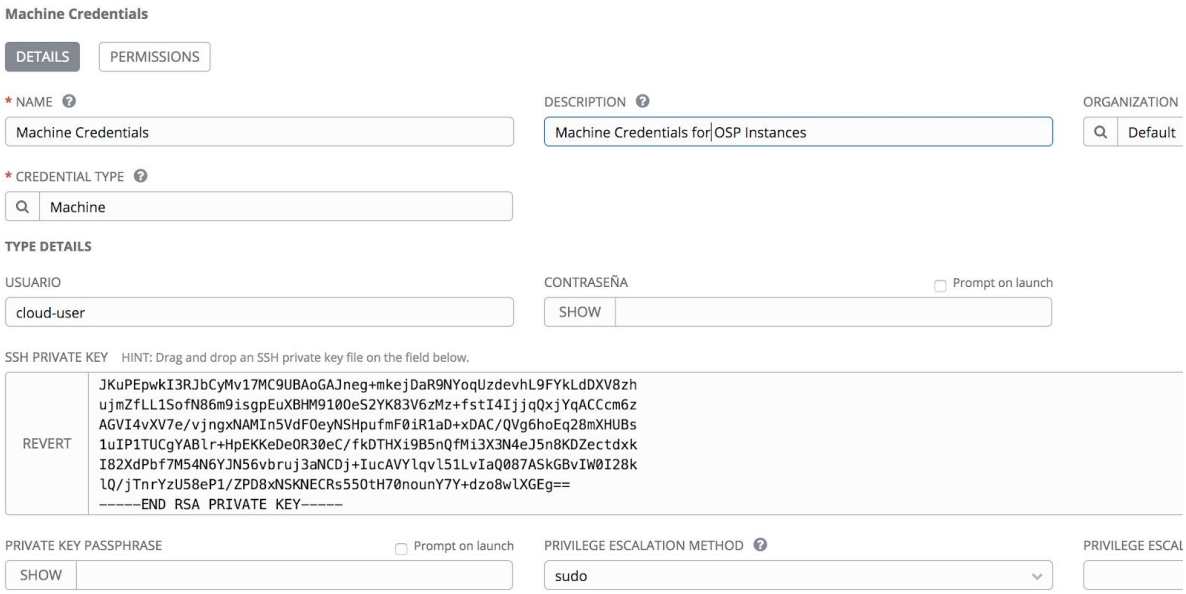
You can grant users and teams the ability to use these credentials, without actually exposing the credential to the user. If you have a user move to a different team or leave the organization, you don't have to re-key all of your systems just because that credential was available in Tower.

Navigate within the **Ansible Tower Portal** and click on **Credentials** (within **RESOURCES** Section). Click on  to add new Credentials:

Create the new Credentials as follows:

NAME	Machine Credentials
DESCRIPTION	Machine Credentials for OSP Instances
ORGANIZATION	Default
TYPE	Machine
USERNAME	cloud-user
SSH PRIVATE KEY	Drag and drop your **test-drive.pem** file
PRIVILEGE ESCALATION METHOD	sudo

Click on **SAVE** to save the changes.



Creating Ansible Tower Job Templates

A job template is a definition and set of parameters for running an Ansible job. Job templates are useful to execute the same job many times. Job templates also encourage the reuse of Ansible playbook content and collaboration between teams. While the REST API allows for the execution of jobs directly, Tower requires that you first create a job template.

Navigate within the **Ansible Tower Portal** and click on **TEMPLATES**.

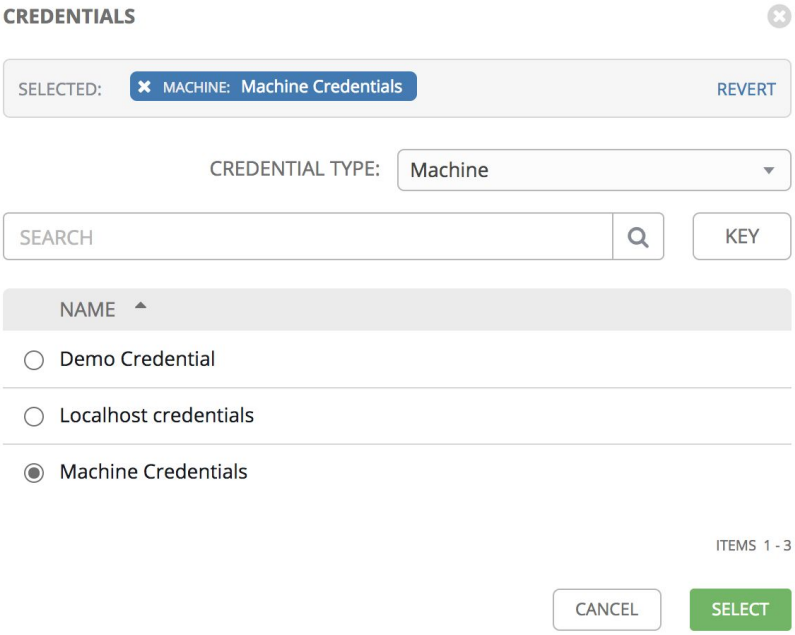
This menu opens a list of the job templates that are currently available. The Templates tab also enables the user to launch, schedule, modify, and remove a job template.

1. Click on **Templates** (within **RESOURCES** Section) and then click on  button (selecting **Job Template**) to create a new Job Template to deploy a **Wordpress Application** within our **wordpress-instance**:

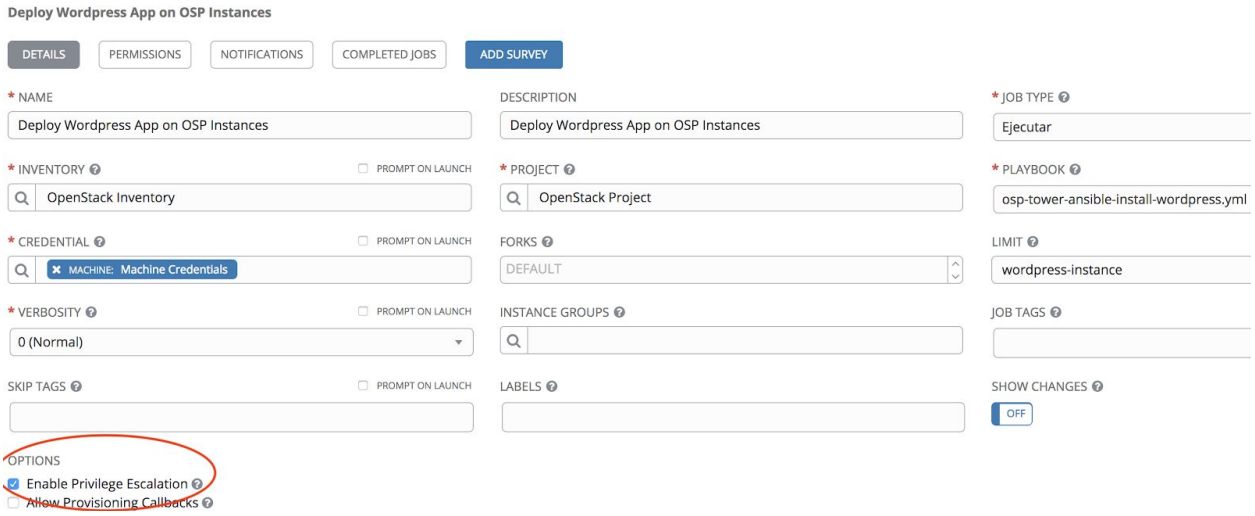
NAME	Deploy Wordpress App on OSP Instances
DESCRIPTION	Deploy Wordpress App on OSP Instances
JOB TYPE	Run
INVENTORY	OpenStack Inventory
PROJECT	OpenStack Project
PLAYBOOK	osp-tower-ansible-install-wordpress.yml
CREDENTIAL	Machine Credentials
LIMIT	wordpress-instance
VERBOSITY	0 (Normal)
OPTIONS	Enable Privilege Escalation: **checked**

Click on **SAVE** to save the changes.

TIP: Selecting a Credential: just click on  icon and then click on **Machine Credentials**. Finally click on **SELECT** button:



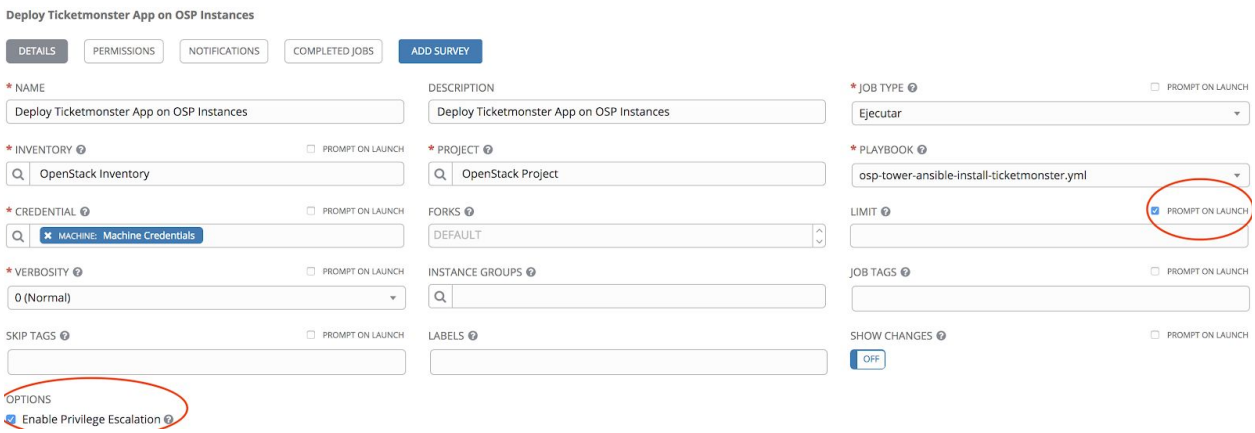
Your new Job Template now looks as follows:



2. Click again on **Templates** (within **RESOURCES** Section) and then click on  button (selecting **Job Template**) to create a new Job Template to deploy a **Ticketmonster Web Application** within an instance. We'll be creating this instance via *CloudForms* using a Service Bundle and ***we'll be using this Job Tamplate later within this Service Bundle*** *to deploy the application within the instance*:

NAME	Deploy Ticketmonster App on OSP Instances
DESCRIPTION	Deploy Ticketmonster App on OSP Instances
JOB TYPE	Run
INVENTORY	OpenStack Inventory
PROJECT	OpenStack Project
PLAYBOOK	osp-tower-ansible-install-ticketmonster.yml
CREDENTIAL	Machine Credentials
LIMIT	**Keep it in blank** (*just –> Prompt on Launch: **checked***)
VERBOSITY	0 (Normal)
OPTIONS	Enable Privilege Escalation: **checked**

Click on **SAVE** to save the changes.



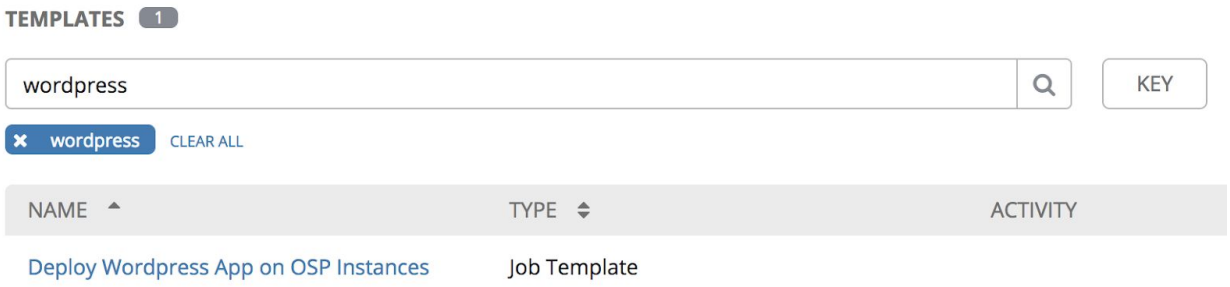
Launching Ansible Tower Job Templates

Now it's time to deploy our new **Wordpress** App within **wordpress-instance** previously created via CloudForms.

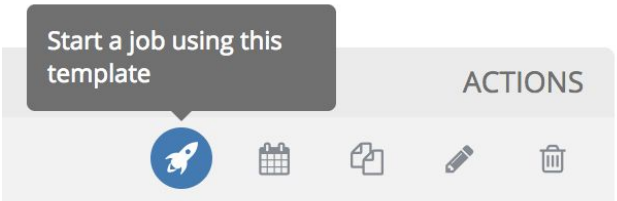
First, use the URL provided to you to be sure **Wordpress** is not installed, using it with a web browser:

http://floating106-GUID.rhpds.opentlc.com/wordpress

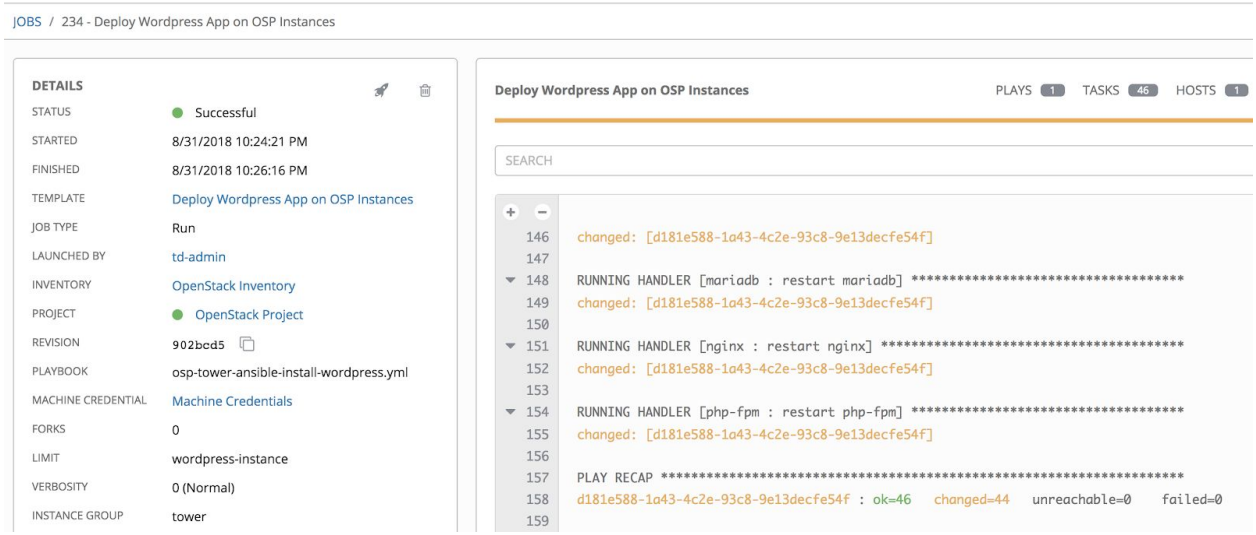
Then, go back to **Ansible Tower Portal**. Click on **Templates** (within **RESOURCES** Section) and find the **"Deploy Wordpress App on OSP Instances"** Job Template, using the **SEARCH** textbox as follows:



Click on  to start a job creation using this template.

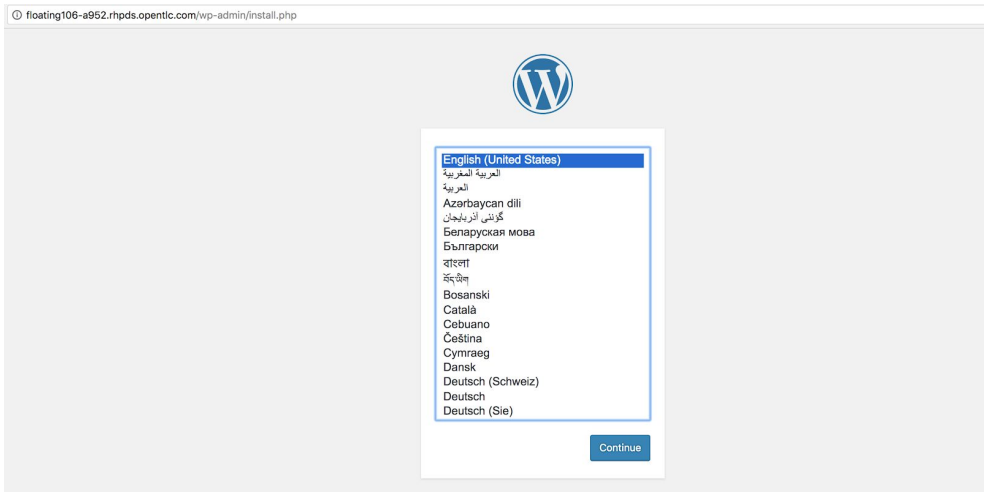


This Action will launch the **“Deploy Wordpress App on OSP Instances”** Job Template, starting the Website installation. You will be redirected to a dynamic Job Output page:



You will see the Status of the Job, Started / Finished Time, Tasks, etc.

If your Job finished **Ok**, with Status **Successful**, go to your **Wordpress URL** again and make sure your Wordpress Server is **Up & Running**. Play a moment with your new Wordpress!



You can take a look at the **Ansible Playbook** we are using for this Job Template:

https://github.com/rcalvaga/openstack-ansible/blob/master/osp-tower-ansible-install-wordpress.yml

Lab 9: Automating Services Delivery using CloudForms and Ansible Tower

Your apps have to live somewhere. If you're PXE booting and kickstarting bare-metal servers or VMs, or creating virtual or cloud instances from templates, Ansible and Red Hat® Ansible® Tower help streamline the process in conjunction with Red Hat® CloudForms®.

We are using *Ansible Tower* as an Automation Provider for *CloudForms*, and we want to successfully provision a Web Application (Ticketmonster Service Portal) within an OpenStack Instance. For this task, we need to create a new **Service Bundle** based on an OpenStack Instance Service Item to create a new Instance + Ansible Tower Job Template Service Item to deploy our Ticketmonster Web Application within it, and then access to *CloudForms Self-Service Portal* and select our new service to build our own Application.

1. Access to **CloudForms Cloud Management Portal,**

Login	td-admin
Password	r3dh4t1!

Creating Service Catalog Items and Bundles

Through the use of catalogs, Red Hat CloudForms provides support for multi-tier service provisioning to deploy layered workloads across hybrid environments. You can create customized dialogs that will give consumers of the services the ability to input just a few parameters and provision the entire service. The following table lists the terminology associated with catalogs that you will use within the CloudForms user interface for service provisioning.

Now we need to create first two Service Catalog Items:

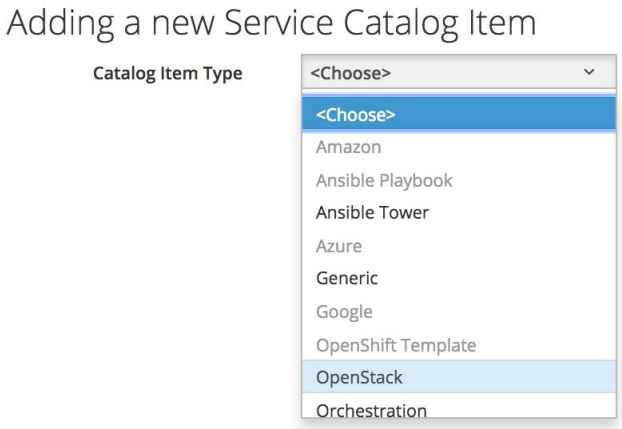
- **Create OpenStack Instance**. Based on Catalog Item Type: OpenStack.
- **Deploy Ticketmonster App on OSP Instance**. Based on Catalog Item Type: AnsibleTower.

Later, we'll be creating a Service Catalog Bundle using this Service Catalog Items.

A. Create an OpenStack Service Catalog Item:

1. Navigate to the **Catalogs** section in the accordion, **Services → Catalogs** and then **Catalog Items → All Catalog Items → OpenStack Catalog** and click on **Configuration → Add a New Catalog Item.**

Select **OpenStack** from the drop-down list:



Fill out information as follows:

Basic Info:

Name/Description	Create OSP Instance
Display in Catalog	**Keep it unchecked**
Catalog	OpenStack Catalog
Dialog	**<No Dialog>**

Request Info:

Click on **Request Info** Tab and then:

Catalog Tab:

Select Image Name	RHEL-7

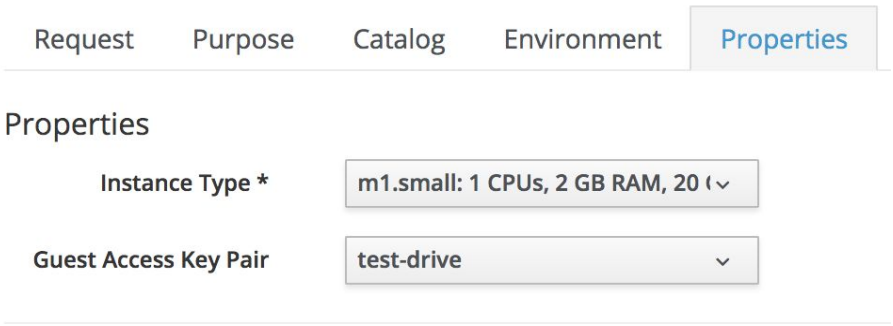
Number of Instances	1
Instance Name	changeme

Environment Tab:

Cloud Tenant	test-drive
Availability Zones	nova
Cloud Network	**private**
Security Groups	SSH_HTTP_From_Any
Public IP Address	**<None>** (we will select it using a Service Dialog later)

Properties Tab:

Instance Type	m1.small
Guest Access Key Pair	test-drive

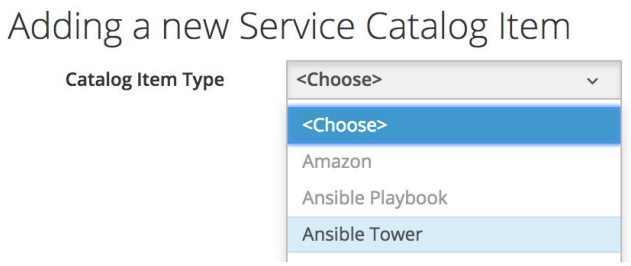


2. Click on **Add** to create the new OSP Service Item.

B. Create an Ansible Tower Service Catalog Item:

1. Navigate to the **Catalogs** section in the accordion, **Services → Catalogs** and then **Catalog Items → All Catalog Items → OpenStack Catalog** and click on **Configuration → Add a New Catalog Item.**

Select **AnsibleTower** from the drop-down list:



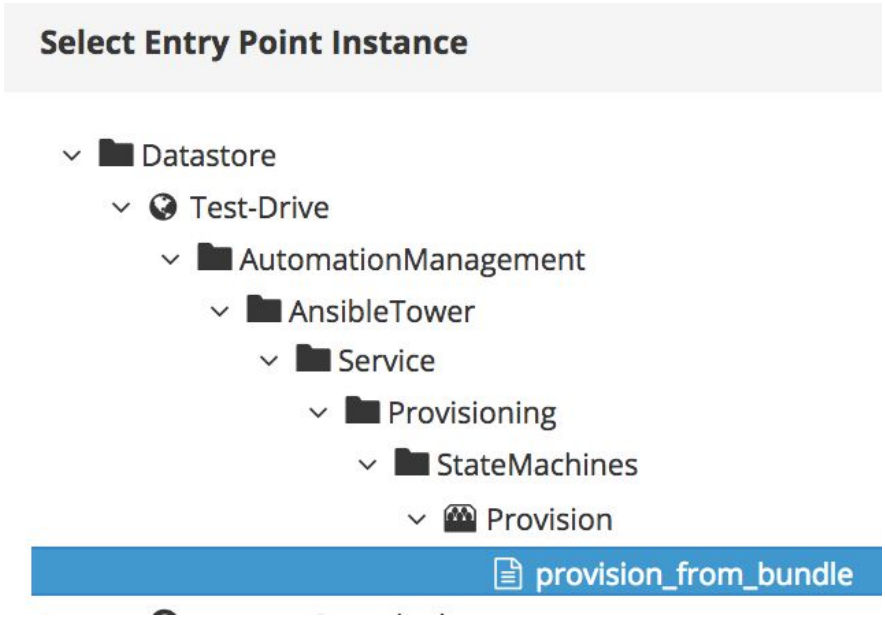
Fill out information as follows:

Basic Info:

Name/Description	Deploy Ticketmonster App on OSP Instance
Display in Catalog	**Keep it unchecked**
Catalog	OpenStack Catalog
Dialog	**<No Dialog>**
Provider	Ansible Tower 3 Automation Manager
Ansible Tower Job Template	Deploy Ticketmonster App on OSP Instances
Provisioning Entry Point	/AutomationManagement/AnsibleTower/Service/Provisioning/StateMachines/Provision/provision_from_bundle

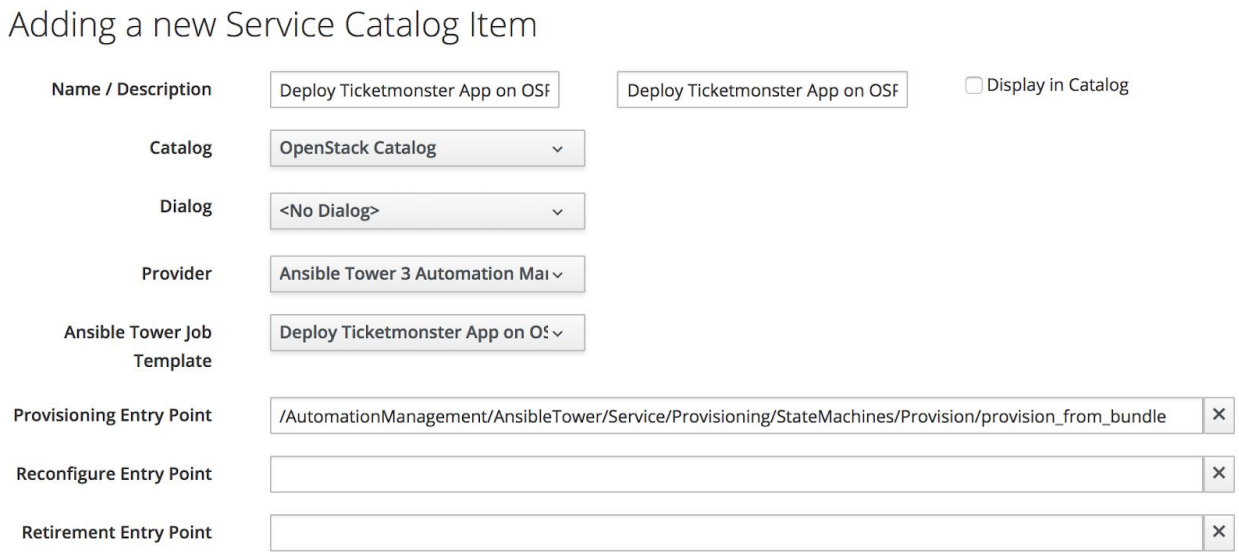
TIP: Selecting the Provisioning Entry Point Path:

Click on the **Provisioning Entry Point Path** and then select it as follows:



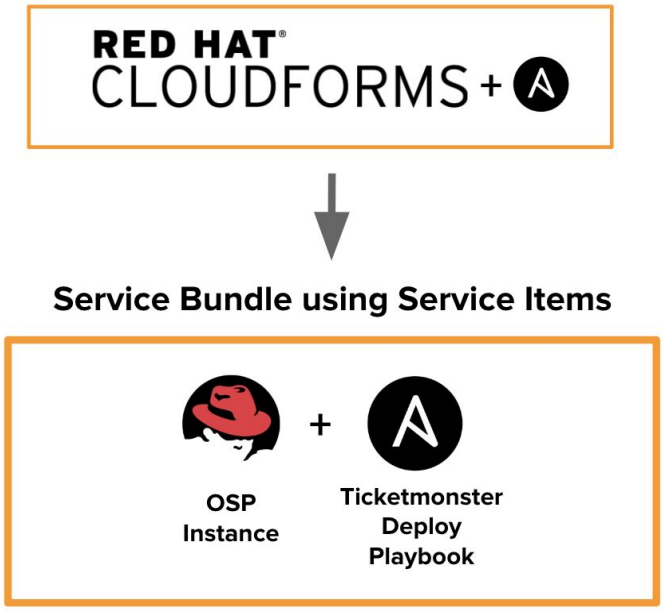
Click on **Apply** to select the Provisioning Entry Point

Then click on **Add** to create our new AnsibleTower Service Item.



C. **Create a Ticketmonster App Service Catalog Bundle:**

Now it's time to create a Service Catalog Bundle, using our Service Catalog Items created previously, so that we can create first an OpenStack Instance and then automate a configuration and an application deployment within it.



1. Navigate to the **Catalogs** section in the accordion, **Services → Catalogs** and then **Catalog Items → All Catalog Items → OpenStack Catalog** and click on **Configuration → Add a New Catalog Bundle**.

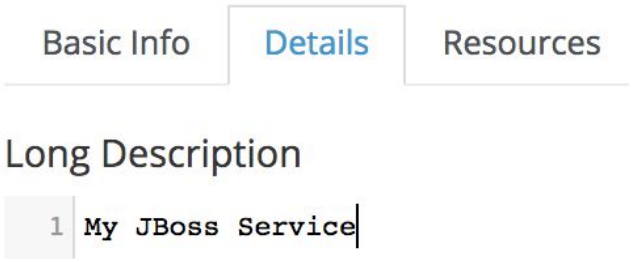
Fill out information as follows:

Basic Info:

Name/Description	Ticketmonster Service
Display in Catalog	**Checked**
Catalog	OpenStack Catalog
Dialog	**Ticketmonster Service Dialog**

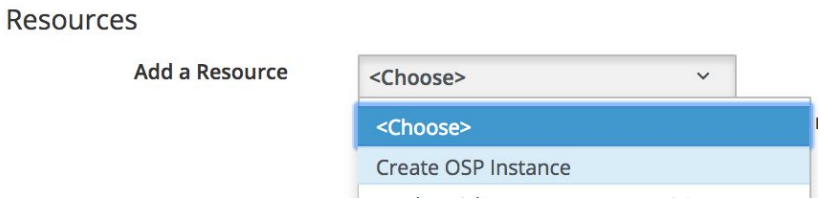
Details:

Click on **Details Tab** and put **"My JBoss Service"** as Long Description:

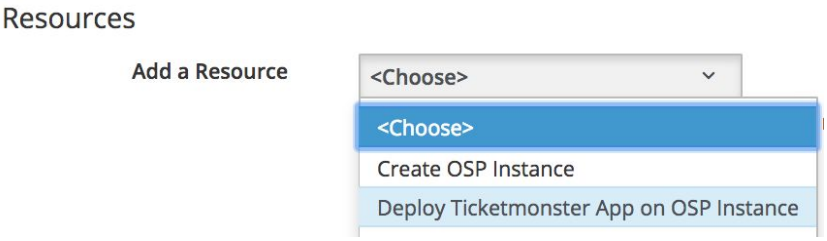


Resources:

Click on **Add a Resource** Dropdown and Select: **"Create OSP Instance"**:



Click again on **Add a Resource** Dropdown and Select: "**Deploy Ticketmonster App on OSP Instance**":



After that, it's necessary to adjust some stuff for our selected resources, such as **Action Order**, **Provision Order** and **Delay** as follows:

Name	Create OSP Instance
Action Order	1
Provision Order	1

Name	Deploy Ticketmonster App on OSP Instance
Action Order	2
Provision Order	2
Delay (mins) Start	1

You will have something like that:

Then click on **Add** to create the new Service Bundle.

Now it's time to order our new Service Bundle!

Ordering a new Service using CloudForms Self-Service Portal

Red Hat CloudForms Self Service is a web-based graphical user interface for ordering and managing IT service requests. You can enable self-service tenant end users, who can easily access their services, track requests, and manage their accounts using the Self Service user interface (SSUI), which has widgets, dashboard controls and feedback. The Self Service user interface supports role-based access control (RBAC) of menus and features, similar to in the full administrative user interface.

1. Access to **CloudForms Self-Service Portal**, using the URL provided to you, adding **/self_service** context-root

(eg. **https://cloudforms-GUID.rhpds.opentlc.com/self_service**)

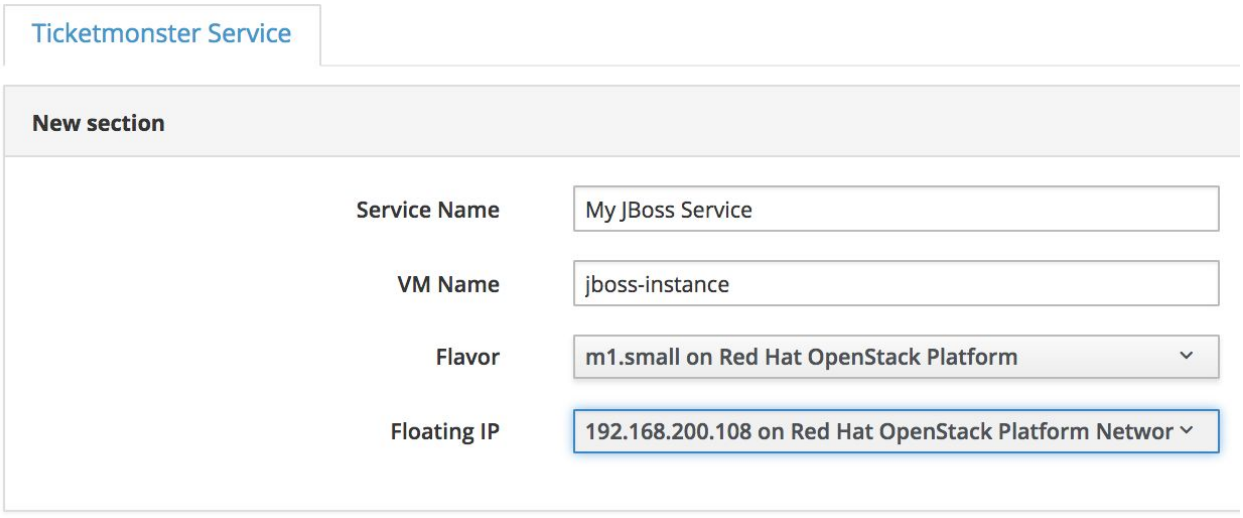
Login	td-admin
Password	r3dh4t1!

2. Click on **Service Catalog**

3. Click on **Ticketmonster Service**

Fill out required information as follows:

Service Name	My JBoss Service
VM Name	jboss-instance
Flavor	m1.small
Floating IP	192.168.200.108



4. Click on **Add to Shopping Cart** button. Then click on **your Shopping Cart icon** and click on **Order** to order your service.

Meanwhile, use the URL provided to you to be sure **Ticketmonster Web App** is **not** installed, using it with a web browser and adding **:8080** port and context-root **/ticket-monster** as follows:

http://floating108-GUID.rhpds.opentlc.com:8080/ticket-monster

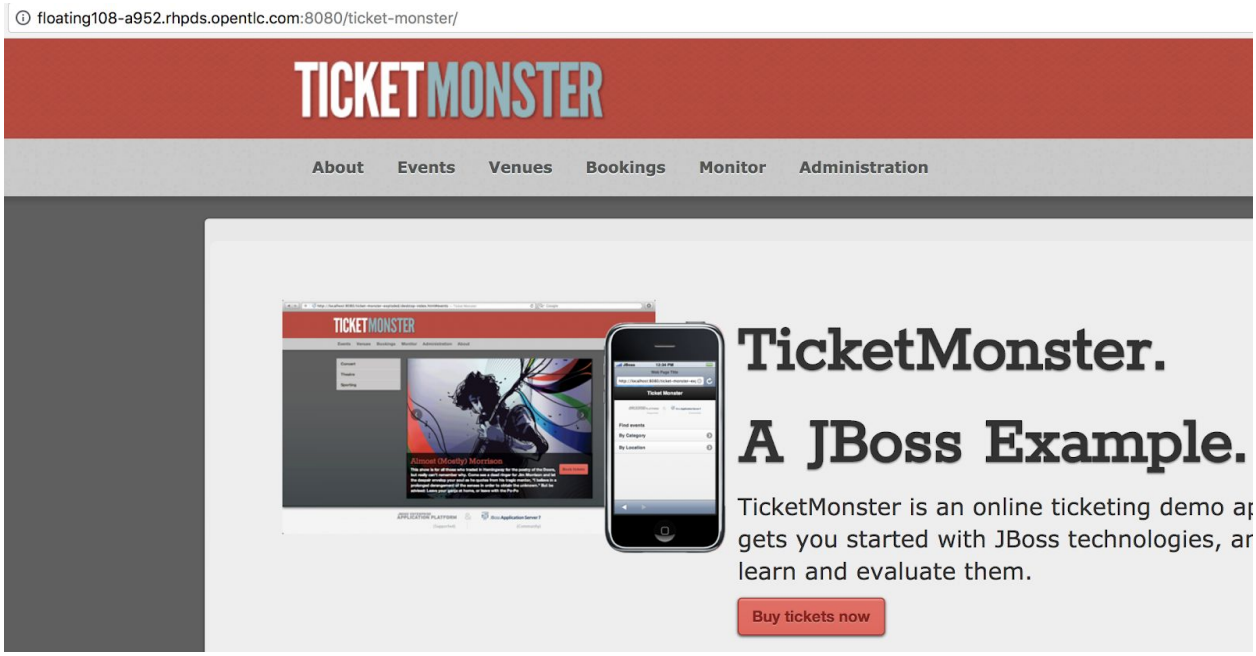
5. Return to the **CloudForms Cloud Management Engine Portal**:

Go to **Services → Requests** and click on **Reload** button until Request State is in **Finished** state.

6. Switch to **RHEL OpenStack Platform Portal** to observe the deployment of the new OpenStack instance from a Service. Click on **Instances → jboss-instance** to see the instance details.

7. You can access to **Ansible Tower Portal**. Click on **Jobs**. Click on **Deploy Ticketmonster App on OSP Instances** Job and take a look at all information.

8. Go to your URL provided but adding **:8080** port and context-root **/ticket-monster**:
http://floating108-GUID.rhpds.opentlc.com:8080/ticket-monster



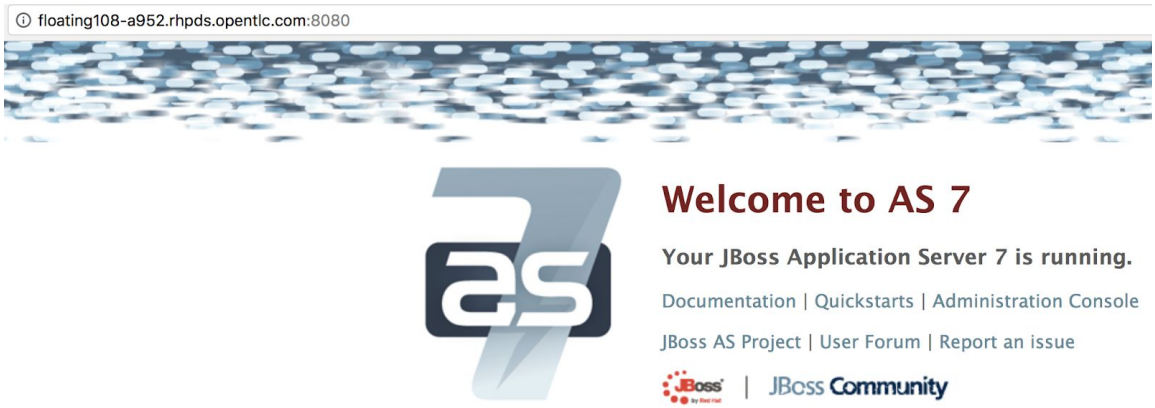
Now you can access your new **Ticketmonster Web App** using your link and it's time to play with it!

Go to your **JBoss Application Server URL** and make sure your App Server is Up & Running, **accessing via :8080 Port!**

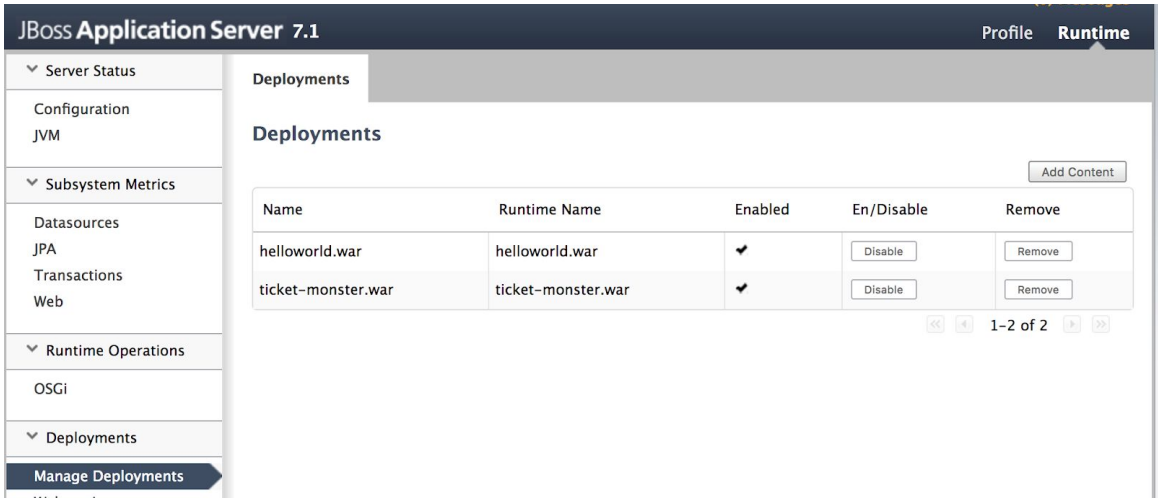
http://floating108-GUID.rhpds.opentlc.com:8080

Click on **Administration Console** link. You can access the **JBoss Administration Console:**

User: admin / **Password:** Redhat1!



Click on **Deployments → Manage Deployments.** Refresh page if necessary. You will see your new deployments:



TEST-DRIVE SURVEY:

If you finished the Test-Drive, please answer this Survey, selecting **"Private Cloud"** as **Test Drive Name:**

https://docs.google.com/forms/d/e/1FAIpQLSdauHtguNMYICRE5x1nrE0Y11ASfDNnptSEqbLZi_TCsNgb2g/viewform