Contents

Lab 1: Overview of Deploying OpenShift Enterprise 2.0 on Red Hat OpenStack 4.0 via Heat Templates

1.1 Assumptions

This lab manual assumes that you are attending an instructor-led training class and that you will be using this lab manual in conjunction with the lecture.

This manual also assumes that you have been granted access to a single Red Hat Enterprise Linux server with which to perform the exercises.

A working knowledge of SSH, git, and yum, and familiarity with a Linux-based text editor are assumed. If you do not have an understanding of any of these technologies, please let the instructors know.

1.2 What you can expect to learn from this training class

At the conclusion of this training class, you should have a solid understanding of how to configure Heat to deploy an OpenShift Enterprise 2.0 broker and node. In addition, you will learn how to expand the OpenShift node environment. You should also feel comfortable creating and deploying applications using the OpenShift Enterprise management console, using the OpenShift Enterprise administration console, as well as using the command line tools.

1.3 Overview of OpenShift Enterprise PaaS

Platform as a Service is changing the way developers approach developing software. Developers typically use a local sandbox with their preferred application server and only deploy locally on that instance. Developers typically start JBoss locally using the startup.sh command and drop their .war or .ear file in the deployment directory and they are done. Developers have a hard time understanding why deploying to the production infrastructure is such a time consuming process.

System Administrators understand the complexity of not only deploying the code, but procuring, provisioning, and maintaining a production level system. They need to stay up to date on the latest security patches and errata, ensure the firewall is properly configured, maintain a consistent and reliable backup and restore plan, monitor the application and servers for CPU load, disk IO, HTTP requests, etc.

OpenShift Enterprise provides developers and IT organizations an auto-scaling cloud application platform for quickly deploying new applications on secure and scalable resources with minimal configuration and management headaches. This means increased developer productivity and a faster pace with which IT can support innovation.

1.4 Overview of IaaS

OpenShift Enterprise is infrastructure agnostic. OpenShift Enterprise can be installed on bare metal, virtualized instances or on public/private cloud instances. At a basic level it requires Red Hat Enterprise Linux running on x86_64 architecture. Red Hat

Enterprise Linux provides the advantage of SELinux and other enterprise features to ensure the installation is stable and secure.

This means that in order to take advantage of OpenShift Enterprise any existing resources from your hardware pool may be used. Infrastructure may be based on EC2, VMware, RHEV, Rackspace, OpenStack, CloudStack or even bare metal: essentially any Red Hat Enterprise Linux operating system running on x86_64.

For this training class, Red Hat Linux OpenStack Platform 4.0 is the Infrastructure as a Service layer. The OpenStack environment has been installed on a single node with all the necessary components required to complete the lab.

1.5 Using the *openshift.sh* installation script

This training session will demonstrate the deployment mechanisms that Heat provides. Heat runs as a service on the OpenStack node in this environment. Heat also utilizes the `openshift.sh` installation script. `openshift.sh` automates the deployment and initial configuration of OpenShift Enterprise platform. For a deeper understanding of the internals of the platform refer to the official OpenShift Enterprise Deployment Guide.

Lab 1 Complete!

Lab 2: Lab Environment

2 Server Configuration

Each student will either recieve his / her own server or will share with another student. The server has Red Hat Enterprise Linux 6.5 install as the base operating system. The server was configured with OpenStack with packstack. Explore the environment to see what was pre-configured. The end result will consist of a Controller host (hypervisor) and 3 virtual machines: 1 OpenShift broker and 2 OpenShift nodes.

System Partitions

If you have to reboot the system, we are on partition X NEED TO FILL THIS OUT.

Look at the configuration options for Heat and Neutron:

```
vim /root/answer.txt
```

Each system has software repositories that are shared out via the local Apache web server:

```
ll /var/www/html
```

These will be utilized by the *openshift.sh* file when it is called by heat.

Explore the Heat template:

```
egrep -i 'curl|wget' /root/heat-templates/openshift-enterprise/heat/neutron/OpenShift-1B1N-neutron.yaml
```

Here you can see that the Heat template was originally making calls to github for the *enterprise-2.0* and *openshift.sh* files. These lines were modified to point to local repositories for the purposes of this lab.

Look a the images that were pre-built for this lab:

```
ls /home/images/RHEL*
```

These two images were pre-built using disk image builder(DIB) for the purpose of saving time in the lab. The commands used to

Check out the software repositories:

```
yum repolist
```

Lab 2 Complete!

Lab 3: Configure Host Networking

3.1 Configure Interfaces

The server has a single network card. Configure both of the interface files at one time and then restart networking.

Explore the current network card interface setup:

```
ovs-vsctl show
ip a
```

Here you will notice that out of the box, packstack does not configure the interfaces. In it's current state, *em1* has the IP address. We need to migrate that IP address to the *br-em1* interface.

Set up the interfaces on the server:

Ensure the *ifcfg-em1* and *ifcfg-br-em1* files look as follows. The ifcfg-br-em1 file will have to be created - it does not exist out of box. The two files on the host should look exactly the same as what is listed below.

```
/etc/sysconfig/network-scripts/ifcfg-br-em1
DEVICE="br-em1"
ONBOOT="yes"
DEVICETYPE=ovs
TYPE="OVSBridge"
OVSBOOTPROTO="dhcp"
OVSDHCPINTERFACES="em1"
```

and configure em1, it exists already, just modify to make it look like:

```
/etc/sysconfig/network-scripts/ifcfg-em1
DEVICE="em1"
ONBOOT="yes"
TYPE="OVSPort"
OVS_BRIDGE="br-em1"
PROMISC="yes"
DEVICETYPE="ovs"
```

Restart Networking and review the interface configuration:

```
service network restart
```

Confirm the IP address moved to the bridge interface.

```
ovs-vsctl show
ip a
```

Now the IP address should be on the *br-em1* interface.

Lab 3 Complete!

Lab 4: Configure Neutron Networking

4.1 Create Keypair

All actions in this lab will performed by the *admin* tenant in this lab. In a production enviroinment there will likely be many tenants.

```
source /root/keystonerc_admin
```

Create a keypair and then list the key.

```
nova keypair-add adminkp > /root/adminkp.pem && chmod 400 /root/adminkp.pem
nova keypair-list
```

4.2 Set up Neutron Networking

Set up the neutron networking.

Create the *public* network. In the packstack answer file we specified the name *physnet1* for the physical external network. INSERT VINNY HERE - ack - vvaldez

```
neutron net-create public --provider:physical_network=physnet1 --provider:network_type flat --router:external=True
```

List the network after creation.

```
neutron net-list
```

Show the public network. If you have networks that are named the same thing, you can specify the UUID for the network instead of the name.

```
neutron net-show public
```

Create the *private* network that the virtual machines will be deployed to.

```
neutron net-create private --provider:network_type local
```

List the network after creation. This time you should see both **public** and **private**

```
neutron net-list
```

Show more details about the private network.

```
neutron net-show private
```

Create the *public* subnet. This command also creates a pool of IP addresses that will be *floating* IP addresses. In addition, set up the gateway here.

```
neutron subnet-create public --allocation-pool start=x.x.x.x,end=x.x.x.x \
--gateway x.x.x.x --enable_dhcp=False x.x.x.0/x --name pub-sub
```

List the *public* subnet.

```
neutron subnet-list
```

Show more details about the *public* subnet.

```
neutron subnet-show pub-sub
```

Create the private subnet.

```
neutron subnet-create private --gateway 192.168.0.1 192.168.0.0/24 --name priv-sub
```

List the *private* subnet.

```
neutron subnet-list
```

Show more details about the *pivate* subnet.

```
neutron subnet-show priv-sub
```

Create the router.

```
neutron router-create router1
```

Set the gateway for the router to reside on the *public* subnet.

```
neutron router-gateway-set router1 public
```

List the router.

```
neutron router-list
```

Update the *public* subnet with a valid DNS entry. **THIS WILL NEED TO BE MODIFIED, IT MAY NEED TO BE REMOVED**

```
neutron subnet-update pub-sub --dns_nameservers list=true 10.16.143.247
```

Add an interface for the private subnet to the router.

```
neutron router-interface-add router1 priv-sub
```

Display router1 configuration.

```
neutron router-show router1
```

4.3 Configure the Neutron plugin.ini

Ensure the */etc/neutron/plugin.ini* has this configuration at the bottom of the file in the [OVS] stanza. The key part is to ensure the *vxlan_udp_port* and *network_vlan_ranges* are commented out.

```
# vxlan_udp_port=4789
# network_vlan_ranges=physnet1:1113:1114
tenant_network_type=local
enable_tunneling=False
integration_bridge=br-int
bridge_mappings=physnet1:br-em1
```

Reboot the server.

```
reboot
```

Lab 4 Complete!

Lab 5: Explore the Openstack Environment

5 Server Configuration

FILL OUT THIS

FILL OUT THIS

FILL OUT THIS

Lab 5 Complete!

Lab 6: Deploy Heat Stack

6.1 Import the Images into Glance

All actions in this lab will performed by the *admin* tenant in this lab. In a production enviroinment there will likely be many tenants.

```
source /root/keystonerc_admin
```

The names of these images are hard coded in the heat template. Do not change the name here.

```
glance add name=RHEL65-x86_64-broker is_public=true disk_format=qcow2 \
container_format=bare < /home/images/RHEL65-x86_64-broker-v2.qcow2

glance add name=RHEL65-x86_64-node is_public=true disk_format=qcow2 \
container_format=bare < /home/images/RHEL65-x86_64-node-v2.qcow2

glance image-list
```

6.2 Ensure the heat.conf file is confirgured correctly

Ensure the following variables are set in the /etc/heat/heat.conf file:

```
heat_metadata_server_url=http://IP of Controller:8000

heat_waitcondition_server_url=http://IP of Controller:8000/v1/waitcondition

heat_watch_server_url=http://IP of Controller:8003
```

6.3 Create the openshift-environment file

Create the openshift-environment.yaml file:

Get the private and public network IDs as well as the private subnet ID out of the first column of the output of the below commands. Place those parameters in the following file in the following fields: private_net_id: public_net_id: private_subnet_id: to replace FIXME.

```
neutron net-list
neutron subnet-list
```

Create the */root/openshift-environment.yaml* file and copy the following contents into it. For the IP address of the repo locations, please replace with the IP address of the host you are on.

```
parameters:
  key_name: rootkp
  prefix: novalocal
  broker_hostname: openshift.brokerinstance.novalocal
  node_hostname: openshift.nodeinstance.novalocal
  conf_install_method: yum
  conf_rhel_repo_base: http://IP_OF_HOST/rhel6.5
  conf_jboss_repo_base: http://IP_OF_HOST
  conf_ose_repo_base: http://IP_OF_HOST/ose-latest
  conf_rhscl_repo_base: http://IP_OF_HOST
  private_net_id: FIXME
  public_net_id: FIXME
  private_subnet_id: FIXME
  yum_validator_version: "2.0"
  ose_version: "2.0"
```

6.4 Open the port for Return Signals

The *broker* and *node* VMs need to be able to deliver a completed signal to the metadata service.

```
iptables -I INPUT -p tcp --dport 8000 -j ACCEPT
service iptables save
```

6.5 Launch the stack

Now run the *heat* command and launch the stack. The -f option tells *heat* where the template file resides. The -e option points *heat* to the environment file that was created in the previous section.

Note: it can take up to 10 minutes for this to complete

```
source /root/keystonerc_admin

cd /root/

heat create openshift \
-f heat-templates/openshift-enterprise/heat/neutron/OpenShift-1B1N-neutron.yaml \
-e /root/openshift-environment.yaml
```

6.6 Monitor the stack

List the *heat* stack

```
heat stack-list
```

Watch the heat events.

```
heat event-list openshift

heat resource-list openshift

nova list
```

Get a VNC console address and open it in the browser. Firefox must be launched from the hypervisor host, the host that is running the VM's.

```
nova get-vnc-console broker_instance novnc

nova get-vnc-console node_instance novnc
```

6.7 Confirm Connectivity

Ping the public IP. Get the public IP by running *nova list* on the controller.

```
ping x.x.x.x
```

SSH into the broker instance. This may take a minute or two while they are spawning. This will use the key that was created with *nova keypair* earlier.

Confirm which IP address belongs to the broker and to the node.

```
nova list
```

SSH into the broker

```
ssh -i ~/adminkp.pem ec2-user@IP.OF.BROKER
```

Once logged in, gain root access and explore the environment.

```
sudo su -
```

Check the OpenShift install output.

```
cat /tmp/openshift.out
```

Check mcollective traffic. You should get a response from the node that was deployed as part of the stack.

```
oo-mco ping

oo-diagnostics -v

oo-accept-broker -v
```

SSH into the node, using the IP that was obtained above.

```
ssh -i ~/rootkp.pem ec2-user@IP.OF.NODE
```

Check node configuration

```
oo-accept-node
```

Confirm Console Access by opening a browser and putting in the IP address of the broker.

http://IP.OF.BROKER/console

FILL OUT THIS

FILL OUT THIS

Lab 6 Complete!

Lab 7: Installing the RHC client tools

Server used:

- localhost

Tools used:

- ruby
- sudo
- git
- yum
- gem
- rhc

The OpenShift Client tools, known as **rhc**, are built and packaged using the Ruby programming language. OpenShift Enterprise integrates with the Git version control system to provide powerful, decentralized version control for your application source code.

OpenShift Enterprise client tools can be installed on any operating system with Ruby 1.8.7 or higher. Instructions for specific operating systems are provided below. It is assumed that you are running the commands from a command line window, such as Command Prompt, or Terminal. If you are using Ruby Version Manager (rvm) see the instructions below.

Microsoft Windows

Installing Ruby for Windows

RubyInstaller 1.9 provides the best experience for installing Ruby on Windows XP, Vista, and Windows 7. Download the latest 1.9 version from the download page and launch the installer.

Important: During the installation, you should accept all of the defaults. It is mandatory that you select the "Add Ruby executables to your PATH" check box in order to run Ruby from the command line.

After the installation is complete, to verify that the installation is working, run:

```
C:\Program Files\> ruby -e 'puts "Welcome to Ruby"'
Welcome to Ruby
```

If the 'Welcome to Ruby' message does not display, the Ruby executable may not have been added to the path. Restart the installation process and ensure the "Add Ruby executables to your PATH" check box is selected.

Installing Git for Windows

The next step is to install Git for Windows so that you can synchronize your local application source and your OpenShift application. Git for Windows offers the easiest Git experience on the Windows operating system and is the recommended default - if you use another version of Git, please ensure it can be executed from the command line, and continue to the next section.

Download and install the latest version of Git for Windows. Ensure that Git is added to your PATH so that it can be run from the command line. After the installation has completed, verify that Git is correctly configured by runing:

```
C:\Program Files\> git --version
git version 1.7.11.msysgit.1
```

Installing RHC for Windows

After Ruby and Git are correctly installed, use the RubyGems package manager (included in Ruby) to install the OpenShift Enterprise client tools. Run:

```
C:\Program Files\> gem install rhc
```

RubyGems downloads and installs the rhc gem from www.rubygems.org/gems/rhc. The installation typically proceeds without errors. After the installation has completed, run:

```
C:\Program Files\> rhc
```

Mac OS X

Installing Ruby for OS X

From OS X Lion onwards, Ruby 1.8.7 is installed by default. On older Mac systems, Ruby is shipped as part of the Xcode development suite and can be installed from your installation CD. If you are familiar with Mac development, you can also use MacRuby or see the Ruby installation page for help installing with homebrew.

To verify that Ruby is correctly installed run:

```
$ ruby -e 'puts "Welcome to Ruby"'
Welcome to Ruby
```

Installing Git for OS X

There are a number of options on Mac OS X for Git. We recommend the Git for OS X installer - download and run the latest version of the dmg file on your system. To verify the Git for OS X installation, run:

```
$ git --version
git version 1.7.11.1
```

Installing RHC for OS X

With Ruby and Git installed, use the RubyGems library system to install and run the OpenShift Enterprise gem. Run:

```
$ sudo gem install rhc
```

After the installation has completed, run:

```
$ rhc -v
```

Fedora 16 or later

To install from yum on Fedora, run:

```
$ sudo yum install rubygem-rhc
```

This installs Ruby, Git, and the other dependencies required to run the OpenShift Enterprise client tools.

After the OpenShift Enterprise client tools have been installed, run:

```
$ rhc -v
```

Red Hat Enterprise Linux 6 with OpenShift entitlement

The most recent version of the OpenShift Enterprise client tools are available as a RPM from the OpenShift Enterprise hosted Yum repository. We recommend this version to remain up to date, although a version of the OpenShift Enterprise client tools RPM is also available through EPEL.

With the correct entitlements in place, you can now install the OpenShift Enterprise 2.0 client tools by running the following command:

```
$ sudo yum install rubygem-rhc
```

If you do not have an OpenShift Enterprise on the system you want to install the client tools on, you can install ruby and rubygems and then issue the following command:

```
$ sudo gem install rhc
```

Ubuntu

Use the apt-get command line package manager to install Ruby and Git before you install the OpenShift Enterprise command line tools. Run:

```
$ sudo apt-get install ruby-full rubygems git-core
```

After you install both Ruby and Git, verify they can be accessed via the command line:

```
$ ruby -e 'puts "Welcome to Ruby"'
$ git --version
```

If either program is not available from the command line, please add them to your PATH environment variable.

With Ruby and Git correctly installed, you can now use the RubyGems package manager to install the OpenShift Enterprise client tools. From a command line, run:

```
$ sudo gem install rhc
```

Lab 7 Complete!

Lab 08: Using *rhc setup*

Server used:

- localhost

Tools used:

- rhc

Configuring RHC setup

By default, the RHC command line tool will default to use the publicly hosted OpenShift environment. Since we are using our own enterprise environment, we need to tell *rhc* to use our broker.hosts.example.com server instead of openshift.com. In order to accomplish this, the first thing we need to do is run the *rhc setup* command using the optional *--server* parameter.

```
$ rhc setup --server broker.hosts.example.com
```

Once you enter in that command, you will be prompted for the username that you would like to authenticate with. For this training class, use the *demo* user account.

The first thing that you will be prompted with will look like the following:

```
The server's certificate is self-signed, which means that a secure connection can't be established to
'broker.hosts.example.com'.

You may bypass this check, but any data you send to the server could be intercepted by others.
Connect without checking the certificate? (yes|no):
```

Since we are using a self signed certificate, go ahead and select *yes* here and press the enter key.

At this point, you will be prompted for the username. Enter in demo and specify the password for the demo user.

After authenticating, OpenShift Enterprise will prompt if you want to create a authentication token for your system. This will allow you to execute command on the PaaS as a developer without having to authenticate. It is suggested that you generate a token to speed up the other labs in this training class.

The next step in the setup process is to create and upload our SSH key to the broker server. This is required for pushing your source code, via Git, up to the OpenShift Enterprise server.

Finally, you will be asked to create a namespace for the provided user account. The namespace is a unique name which becomes part of your application URL. It is also commonly referred to as the user's domain. The namespace can be at most 16 characters long and can only contain alphanumeric characters. There is currently a 1:1 relationship between usernames and namespaces. For this lab, create the following namespace:

```
ose
```

Under the covers

The *rhc setup* tool is a convenient command line utility to ensure that the user's operating system is configured properly to create and manage applications from the command line. After this command has been executed, a *.openshift* directory will have been created in the user's home directory with some basic configuration items specified in the *express.conf* file. The contents of that file are as follows:

```
# Default user login
default_rhlogin=‘demo’

# Server API
libra_server = 'broker.hosts.example.com'
```

This information will be read by the *rhc* command line tool for every future command that is issued. If you want to run commands as a different user than the one listed above, you can either change the default login in this file or provide the *-l* switch to the *rhc* command.

Lab 8 Complete!

Lab 9: Create a PHP Application

9.1 Create a PHP Application

FILL OUT THIS

FILL OUT THIS

FILL OUT THIS

Lab 9 Complete!

Lab 10: Extending the OpenShift Environment

As applications are added additional node hosts may be added to extend the capacity of the OpenShift Enterprise environment.

10.1 Create the node environment file

A separate heat template to launch a single node host is provided. A heat environment file will be used to simplify the heat deployment.

Create the */root/node-environment.yaml* file and copy the following contents into it.

```
parameters:
  key_name: rootkp
  domain: novalocal
  broker1_floating_ip: 10.16.138.100
  load_bal_hostname: openshift.brokerinstance.novalocal
  node_hostname: openshift.nodeinstance2.novalocal
  node_image: RHEL65-x86_64-node
  hosts_domain: novalocal
  replicants: ""
  install_method: yum
  rhel_repo_base: http://10.16.138.52/rhel6.5
  jboss_repo_base: http://10.16.138.52
  openshift_repo_base: http://10.16.138.52/ose-latest
  rhscl_repo_base: http://10.16.138.52
  activemq_admin_pass: FIXME
  activemq_user_pass: FIXME
  mcollective_pass: FIXME
  private_net_id: FIXME
  public_net_id: FIXME
  private_subnet_id: FIXME
```

10.2 Launch the node heat stack

Now run the *heat* command and launch the stack. The -f option tells *heat* where the template file resides. The -e option points *heat* to the environment file that was created in the previous section.

```
cd /root/

heat stack-create ose_node \
-f  heat-templates/openshift-enterprise/heat/neutron/highly-available/ose_node_stack.yaml \
-e /root/node-environment.yaml
```

10.3 Monitor the stack

List the *heat* stack

```
heat stack-list
```

Watch the heat events.

```
heat event-list openshift

heat resource-list openshift

nova list
```

10.4 Confirm Connectivity

Ping the public IP of node 2

```
ping x.x.x.x
```

SSH into the node2 instance. This may take a minute or two while they are spawning.
This will use the key that was created with *nova keypair* earlier.

SSH into the node

```
ssh -i ~/rootkp.pem ec2-user@IP.OF.NODE2
```

Once logged in, gain root access and explore the environment.

```
sudo su -
```

Check the OpenShift install output.

```
cat /tmp/openshift.out
```

Check node configuration

```
oo-accept-node
```

SSH into the broker instance to update the DNS zone file.

```
ssh -i ~/rootkp.pem ec2-user@IP.OF.BROKER
```

Once logged in, gain root access.

```
sudo su -
```

Append the node 2 instance *A* record to the zone file so node 2 hostname resolves.

```
echo "openshift.nodeinstance2    A    IP.OF.NODE2" >> /var/named/dynamic/novalocal.db
```

Check mcollective traffic. You should get a response from node 2 that was deployed
as part of the stack.

```
oo-mco ping
```

Lab 10 Complete!