Managing the Operation of an OpenShift Cluster

Andrew Block, Scott Collier, Jason DeTiberus, Vinny Valdez

Abstract:
Configuring distributed systems can be difficult. Fortunately, automation tools such as Ansible are available to help manage even the most complex environments. In this lab, you'll take the reigns of your own cluster and experience firsthand how Ansible can be used to install, configure, and maintain OpenShift to support mission-critical systems. Once you install Red Hat OpenShift, you'll learn how to diagnose, troubleshoot, and resolve common platform issues. Managing the platform doesn't stop once the installation is complete. You'll use Ansible to simplify ongoing maintenance in an automated fashion. Finally, the use of centralized management systems will be introduced into the environment in order to demonstrate its importance and to provide a streamlined experience for both platform maintainers and users.
Lab 0 - Pre-Lab Setup

Lab 1 - Lab Overview
   Introduction
   Environment Overview
   Target Environment
   Connectivity Details
   Virtualization level

Lab 2 - Exploring the Environment
   Exploring Red Hat OpenStack Platform Environment
      Connecting to Red Hat OpenStack Platform
      View Servers and Volumes
      Further Environment Exploration
   Exploring Ansible Tower
      Accessing Ansible Tower
      Job Templates
      Projects
      Inventory
      Credentials
   Monitor the Progress of the OpenShift Installation

Lab 3 - Verifying Installation of Red Hat OpenShift Container Platform Using Ansible Tower
   Reviewing Install of OpenShift
   Validate the OpenShift Installation

Lab 4 - Installing Red Hat CloudForms
   Deploy Red Hat CloudForms
   Instantiate CloudForms Templates
   Validating a Successful Deployment
   Accessing the CloudForms User Interface
   Configuring the Container Provider
   Configuring the OpenStack Cloud Provider

Lab 5 - Managing the Lifecycle of an Application
   Deploy a Sample Application
   Validating Application Deployment
   View Application
   Viewing Application Metrics
   Navigate through the OpenShift Web Console

Lab 6 - Expanding the OpenShift Container Platform Cluster
   Review Cluster
   Expand the Cluster
   Validate the Expanded Cluster
Lab 7 - Where do we go from here?

Appendices
  - Appendix A - Manually Cleanup Cinder Volume
  - Appendix B - Script For Deploying CloudForms
  - Appendix C - Recovering From Failed CloudForms Deployment
  - Appendix D - Average Tower Job Times
  - Appendix E - Troubleshooting CloudForms
Lab 0 - Pre-Lab Setup

Welcome! We are going to jump right into the lab implementation and then review the overall architecture and strategy afterward. You have been tasked with managing a Red Hat Container Platform environment running on the Red Hat OpenStack platform. Ansible Tower is also available and being used to execute and manage the overall installation of OpenShift.

Let’s perform some brief validation of the environment and kick off the OpenShift installation.

**NOTE**: The installation of OpenShift Container Platform can take 20-25 minutes so must be started immediately. If bullet point 1 below takes longer than 1 minute to complete, skip it and go directly to bullet point 2.

1. Connect to the running OpenStack environment to validate no servers exist:
   a. From the UI
      i. In a local web browser open [http://rhosp.admin.example.com](http://rhosp.admin.example.com)
      ii. Login with:
         1. Username: *user1*
         2. Password: *summit2017*
      iii. Click on **Compute -> Instances**
      iv. Verify there are no instances running
   b. From the CLI (for advanced OpenStack users)
      i. SSH with password *summit2017*

```
kiosk$ ssh user1@rhosp.admin.example.com
rhosp$ openstack server list
```

2. Connect to Ansible Tower to start the OpenShift deployment:
   a. From a local web browser open [https://tower.admin.example.com](https://tower.admin.example.com)
   b. **NOTE**: If you get an error **internal server error** then SSH to the Tower VM and restart services. SSH with password *summit2017*

```
kiosk$ ssh root@tower.admin.example.com
tower# ansible-tower-service restart
```

c. Login with the following credentials:
   i. Username: *admin*
   ii. Password: *summit2017*

d. On Ansible Tower overview page, select **Templates** on the menu bar at the top of the screen.

e. Locate the job template called **0-Provision and Install OpenShift**

f. Execute the job by clicking the rocket ship icon on the right hand side of the screen under the **Actions** column

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>ACTIVITY</th>
<th>LABELS</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-Provision and Install OpenShift</td>
<td>Workflow</td>
<td></td>
<td></td>
<td>![Rocket Ship Icon]</td>
</tr>
</tbody>
</table>
Lab 1 - Lab Overview

Introduction

With the OpenShift installation process kicked off, we can spend some time and provide an overview of the entire lab.

Managing an ecosystem of infrastructure and applications can be challenging. Fortunately, there are automation tools and technologies available to handle the most intense workloads. Today, we will leverage tools such as Ansible to automate the provisioning of the OpenShift Container Platform on top of Red Hat OpenStack Platform to provide the foundation for running containerized applications. Afterward, Red Hat CloudForms will be deployed to manage and monitor the underlying infrastructure and applications that run in the environment. Finally, we will walk through expanding the environment by adding new compute resources to the environment. By the conclusion of the lab, you will learn how each of these technologies complement one another to offer solutions to effectively manage the most complex environment.

Environment Overview

The lab environment that we will utilize today consists of multiple KVM virtual machines running within each student workstation. The details of each virtual machine are listed below:

- Student Workstation - KVM hypervisor (the system you are logged into now)
- Red Hat OpenStack Environment 10 - has been deployed for you and ready to host instances that will be used for the Red Hat OpenShift Container Platform 3.4.
  - KVM VM
  - hostname: rhosp.admin.example.com
  - Red Hat OpenShift Container Platform
    - 1 Master node
    - 1 Infrastructure Node
    - 2 Application Nodes
    - Red Hat CloudForms (containerized)
- Ansible Tower 3.1.2
  - KVM VM
  - hostname: tower.admin.example.com

In addition to the virtual machines that are running on each student workstation, an instructor machine is also contained within the environment and provides additional resources.

- Repository server
  - KVM VM on instructor machine
  - Hostname: repo.osp.example.com
- Hosts localized RPM's, docker registry, and git repository

The following diagram depicts the network layout within the environment.
Target Environment

As you progress through the series of labs, you will build increased capabilities for effectively managing containerized workloads. The diagram below represents the environment that we will be building today.

Connectivity Details

There are several components that will be utilized throughout the course of this lab. The following table outlines how to connect to each resource:

<table>
<thead>
<tr>
<th>Item</th>
<th>URL</th>
<th>Access</th>
<th>Virt Level</th>
</tr>
</thead>
</table>
| Red Hat OpenStack Platform| [http://rhosp.admin.example.com](http://rhosp.admin.example.com) | Username: user1  
Password: summit2017 | L1         |
| Ansible Tower             | [https://tower.admin.example.com](https://tower.admin.example.com) | Username: admin  
Password: summit2017 | L1         |
| OpenShift Container Platform | [https://master.osp.example.com:8443](https://master.osp.example.com:8443) | Username: user1  
Password: summit2017 | L2         |
| Red Hat CloudForms        | [https://cloudforms-cloudforms.apps.example.com](https://cloudforms-cloudforms.apps.example.com) | Username: admin  
Password: smartvm | L2 (container) |
Virtualization level

To understand the different layers of virtualization we will use the following classifications:

1. L0 - The hypervisor. In this lab this is the desktop you are sitting at
2. L1 - KVM virtual machine running on the L0 hypervisor
3. L2 - OpenStack Instance/Server running in nested virtualization in the OpenStack L1 VM
4. L2 (container) - Application running in a container on the L2 platform - in this case OpenShift

Keep in mind here, that we are using nested virtualization in this lab. So, while the performance is likely acceptable, it’s not reflective of a production deployment.

Each component plays a critical role into the overall management of the environment. Now let’s get started!
Lab 2 - Exploring the Environment

With the installation of the OpenShift Container Platform started and an understanding of the environment as as whole, we are going to take time waiting for the installation to complete to explore the environment in further detail.

Exploring Red Hat OpenStack Platform Environment

Red Hat OpenStack Platform (RHOSP) is used to host the servers used for the OpenShift Container Platform installation. Servers (also called Instances) are booted from LVM volumes on the RHOSP VM. If you view the list of servers and volumes on the Red Hat OpenStack Platform environment, you should see them in various states of BUILD and ACTIVE, though it is possible some may already be built by now. Connect to either the Horizon UI or the CLI to watch the status of servers and volumes.

The RHOSP environment is a KVM virtual machine running on each student machine. This environment will be used to host the Red Hat OpenShift Container Platform. Let’s verify the state of the instances and execute a few commands to validate it is in good working order prior to proceeding.

Connecting to Red Hat OpenStack Platform

From the physical hypervisor (Student Workstation), connect to the OpenStack virtual machine (rhosp.admin.example.com) using the following credentials:

Username: user1
Password: summit2017

You can use the provided SSH private key to connect:

```bash
kiosk$ eval "$(ssh-agent)"
curl -o ~/.L104353-tower.pem http://repo.osp.example.com/pub/L104353-tower.pem
chmod -v 600 L104353-tower.pem
mv L104353-tower.pem ~/.ssh
ssh-add ~/.ssh/L104353-tower.pem
```

**NOTE:** Although root access is not required to run any of the commands below in Red Hat OpenStack Platform, user1 does have sudo access in case you would like to view logs or config files. However, please DO NOT make any changes to the environment or the lab may not work properly.

```bash
kiosk$ ssh user1@rhosp.admin.example.com
```

To connect via the Horizon UI browse to [http://rhosp.admin.example.com](http://rhosp.admin.example.com)

Username: user1
Password: summit2017
View Servers and Volumes

Connect to the running OpenStack environment and view servers and volumes:

1. From the UI
   a. In a local web browser open http://rhosp.admin.example.com
   b. Click on **Compute -> Instances** to view server status
   c. Click on **Compute -> Volumes** to view block storage status

2. From the CLI
   a. SSH with user **user1** and password **summit2017**
   b. View server and volume status:

```
  kiosk$  ssh user1@rhosp.admin.example.com
  rhosp$  openstack server list & & openstack volume list
```

Further Environment Exploration

List the servers that have been started. Since we kicked off the Tower job, you should see the OpenShift servers in various states of **ACTIVE** or **BUILDING**. Use --format and --column to trim the output for easier viewing:

```
  rhosp$  openstack server list --format value --column Name --column Status

  node1.osp.example.com BUILD
  infra.osp.example.com ACTIVE
  master.osp.example.com ACTIVE
```

Since the Red Hat OpenShift environment makes use of persistent storage for the integrated router along with applications, Red Hat OpenStack provides Cinder volumes which the environment will make use of.

List the Cinder volumes by executing the following command:

```
  rhosp$  openstack volume list --format value --column ID --column "Attached to"

  eb8a3ad8-d059-47e5-9c84-cda926470b45 Attached to node1.osp.example.com on /dev/sda
  1b79b1c9-055d-41c1-84c4-17229841ffef1 Attached to infra.osp.example.com on /dev/sda
  903d7dc0-2b9b-423f-8f5f-95797f9d8bec6 Attached to master.osp.example.com on /dev/sda
```

If you list out the logical volumes (lvs), you will see the IDs of the volumes match the lvs:

```
  rhosp$  sudo lvs

  LV                                          VG             Attr       LSize  Pool
  Origin Data%  Meta%  Move Log Cpy%Sync Convert
  volume-1b79b1c9-055d-41c1-84c4-17229841ffef1 cinder-volumes -wi-ac---- 10.00g
  volume-903d7dc0-2b9b-423f-8f5f-95797f9d8bec6 cinder-volumes -wi-ac---- 10.00g
  volume-eb8a3ad8-d059-47e5-9c84-cda926470b45 cinder-volumes -wi-ac---- 10.00g
```
Next, each of the running instances are built from Red Hat Enterprise Linux 7.3. To list the images available for consumption within OpenStack, execute the following command:

```
rhosp$ openstack image list --format value --column Name --column ID
```

```
e5a369ea-f915-4a59-81e4-1015a7c13f6f openshift-base
```

Feel free to view the details of the openshift-base image which is used to instantiate the openshift servers by the Ansible Tower playbooks.

```
rhosp$ openstack image show openshift-base
```

Finally, list the networks and subnets that have been configured in the OpenStack environment if curious.

```
rhosp$ openstack network list && openstack subnet list
```

The network is configured as a flat network to use the libvirt network for routing and DNS, so no floating IPs will be used. All server instances will use static IPs based on pre-configured network ports. You can view this with:

```
rhosp$ openstack port list --format value --column "Fixed IP Addresses" -c Name
```

```
openshift-master ip_address='172.20.17.5', subnet_id='28792deb-8e5f-459e-aa28-aec1d50838ef'
openshift-infra ip_address='172.20.17.6', subnet_id='28792deb-8e5f-459e-aa28-aec1d50838ef'
openshift-node1 ip_address='172.20.17.51', subnet_id='28792deb-8e5f-459e-aa28-aec1d50838ef'
openshift-node2 ip_address='172.20.17.52', subnet_id='28792deb-8e5f-459e-aa28-aec1d50838ef'
openshift-node3 ip_address='172.20.17.53', subnet_id='28792deb-8e5f-459e-aa28-aec1d50838ef'
```

Additional commands are available to investigate each one of the prior areas in greater detail. You are free to explore these areas later if time allots but be extremely careful not to change anything in this environment.

**Exploring Ansible Tower**

Since the installation of OpenShift can take anywhere from 20 - 30 mins, let us take this opportunity to explore the features and configurations of Ansible Tower in the lab environment.

Ansible is an agentless automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, along with many other IT needs. Ansible is used to provision, install and deploy the OpenShift Container Platform to a cluster of instances.

Ansible Tower provides the central management of Ansible workloads to enable complex workflows to manage environments big and small. The entire installation and management of
the OpenShift Container Platform can be managed from a centralized Ansible Tower environment.

Accessing Ansible Tower

As you saw previously, Ansible Tower has been provisioned as a standalone machine within the lab environment.

From the student machine, open a web browser and navigate to

Login with the following credentials:

Username admin
Password summit2017

If successful, will then be placed at the Ansible Tower overview page:

![Ansible Tower Dashboard](https://example.com)

Job Templates

First, let’s review the job template that we just executed to provision the OpenShift Container Platform. This workflow template consists of three chained job templates:

- OpenShift Pre-Install - Prepares the OpenStack environment by provisioning three instances
- OpenShift Install - Installs the OpenShift Container Platform
- OpenShift Post-Install - Customizes the OpenShift cluster for the lab
Projects

The Job Templates utilize Projects, or collections of Ansible playbooks, that in this lab are sourced from a Git repository. To view the projects that are being utilized, select the Projects link on the menu bar. Two projects are being leveraged:

- openshift-ansible - Installs and configures the OpenShift Container Platform
- summit-2017-ocp-operator - Customized Ansible tooling to prepare lab exercises

The configuration of each project can be viewed by selecting the pencil (edit) button under the Actions column.

Inventory

An Inventory within Ansible Tower is similar to a standalone inventory file and contains a collection of host in which jobs may be launched. The inventories defined within Tower can be accessed by clicking on the Inventories link on the menu bar. The OpenShift inventory defines the hosts organized within groups to install and configure the environment. Each group along with the host and variables that have been defined can be accessed by selecting the pencil icon under the Actions column next to each group.

Credentials

Credentials are a mechanism for authenticating against secure resources including target machines, inventory sources and projects leveraging version control systems. Every one of the previously explored areas makes use of a credential. Credentials are configured within the Ansible Tower settings and can be accessed by selecting the Settings icon (gear) on the menu bar. Once within the settings page, select the Credentials link. The following credentials have been defined:

- gitlab-creds - Access lab resources from source control
- osp-guest-creds - Execute actions against OpenStack instances
- osp-user-creds - Allows for communication with the OpenStack platform

Monitor the Progress of the OpenShift Installation

While browsing through the features of Ansible Tower, keep an eye out on the progress of the job template executing the OpenShift installation. OpenShift will be successfully installed when the status of the job template reports as Successful and the play recap reports no errors and appears similar to the following:
Click the Details link on each rectangle to see the details of each playbook. The overall workflow job is complete when all 3 playbooks are completed successfully.

This lab is concluded when the Ansible Tower job is completed successfully.
Lab 3 - Verifying Installation of Red Hat OpenShift Container Platform Using Ansible Tower

In this lab, we will review the install of the OpenShift Container Platform using Ansible Tower that we started at the beginning of this session.

Reviewing Install of OpenShift

The OpenShift Container Platform is installed through a collection of ansible resources. This automation toolset allows platform administrations the ability to quickly provision an environment with minimal effort. Ansible Tower has been configured with a Job Template that makes use of these assets to install OpenShift on instances available in the OpenStack environment.

To view the list of Job Templates configured in Ansible Tower, select Templates on the menu bar at the top of the screen.

All of the job templates configured in Ansible Tower are listed below. Earlier you launched the job template called 0-Provision and Install OpenShift. This is a workflow job type and will execute multiple chained job templates to provision OpenShift. Review the workflow jobs and playbooks that were run in the Jobs page.

When you execute the job template, you will be transferred to the jobs page where you will be able to track the progress and status of the installation. For more information on the Ansible playbooks see https://github.com/openshift/openshift-ansible
Validate the OpenShift Installation

With the OpenShift Container Platform installation complete, let’s perform a few tests to validate the status of the environment. There are two primary methods for accessing OpenShift: the web console and the Command Line tool (CLI).

From the student machine, open a web browser and navigate to the following address:

https://master.osp.example.com:8443

If successful, you should see the following page representing the OpenShift landing page:
Use the following credentials to access the web console:

Username: user1
Password: summit2017

The OpenShift web console provides an interactive way to interact with the OpenShift platform. After successfully authenticating, you are presented with an overview page containing all of the projects that you have access to. Since you are a normal user, you do not have access to any projects.

In subsequent labs, we will explore the OpenShift web console in further detail.

However, we will still use this opportunity to showcase the different items exposed within the web console.

Now that we have had an opportunity to login to the OpenShift web console from a developer's standpoint, let's shift over to an administrative and operations point of view and access the cluster directly using the terminal.

Since the instances deployed within the OpenStack environment are utilizing cloud-init, login to the OpenShift Master instance as cloud-user:

```
$ ssh -i ~/.ssh/L104353-tower.pem cloud-user@master.osp.example.com
```

Access to the cluster is available using the system:admin user which has the cluster-admin role. This can be confirmed by executing the following command which should confirm the currently logged in user is system:admin

```
$ oc whoami
```

As one would expect, users with the cluster-admin role have elevated permissions in comparison to normal users, such as user1 which was utilized when browsing the web console.

Cluster administrators can view all of the nodes that have constitute the cluster:

```
$ oc get nodes
```

View all of the Projects that have been created by users or to support the platform:

```
$ oc get projects
```

Along with listing all of the Persistent Volumes that have been defined:

```
$ oc get pv
```

Now check out the OpenShift on OpenStack cloud provider integration.

```
$ cat /etc/origin/cloudprovider/openstack.conf
```

```
[Global]
auth-url = http://rhosp.admin.example.com:5000/v2.0/
```
The cloud provider integration file tells OpenShift how to interact with OpenStack. You can see that it’s doing so via the OpenStack API which requires an auth-url, credentials, and a tenant name. This integration between OpenShift and OpenStack enable capabilities like dynamic storage provisioning for applications. Cloud Provider configurations are specific to each provider, for example, you also have cloud provider configurations for AWS, Azure, VMware, etc…

Let’s check out the storage class as well, continuing on the integration story.

```
master$ oc get storageclass
master$ oc describe storageclass ocp
```

Notice that the provisioner is the cinder provisioner and the is-default-class is set to 'true'.

You can use the OpenShift Command line tool as a user with cluster administrator role to access the entire set of configurations for the platform.

**Note:** With great power comes great responsibility. Executing commands as a user with cluster administrator rights has the potential to negatively impact the overall health of the environment.

**IMPORTANT:** If you need to teardown the OpenShift Environment and start over, execute the OpenShift Teardown job template. However, please raise your hand and inform one of the lab instructors. *If you do this too late into the lab you may not have enough time to finish.* See this table for a reference of typical times for the Tower jobs: Appendix D - Average Tower Job Times

This concludes lab 3
Lab 4 - Installing Red Hat CloudForms

Red Hat CloudForms Management Engine (CFME) delivers the insight, control, and automation necessary to address the challenges of managing complex environments. CloudForms is available as a standalone appliance, but is also available as a containerized solution that can be deployed on the OpenShift Container Platform.

In this lab, you will deploy a single instance/replica of Red Hat CloudForms to the OpenShift Container Platform cluster and configure the container provider to monitor the OpenShift environment.

Deploy Red Hat CloudForms

**NOTE:** If you are repeating this lab due to an issue encountered, consider using Appendix B - Script For Deploying CloudForms

Since Red Hat CloudForms is available as a container, it can be deployed to the OpenShift Container Platform in a few short steps.

A user with cluster-admin permissions must be used to configure the environment as CloudForms requires access to privileged resources.

First, using the OpenShift Command Line, create a new project called `cloudforms`

```
master$ oc new-project cloudforms
```

By creating a new project, the context of the CLI is automatically switched into the `cloudforms` project:

```
master$ oc config current-context for context
```

When creating a new project, a set of service accounts are automatically provisioned. These accounts are used when building, deploying and running containers. The `default` service account is the de facto service account used by pods. Since CloudForms is deployed within a pod and requires access to key metrics in the OpenShift environment along with the host, it must be granted elevated access as a privileged resource. In OpenShift, permissions associated to pods are managed by Security Context Constraints and the service account that is used to run them.

Execute the following command to add the default service account in the `cloudforms` project to the privileged SCC:

```
master$ oc adm policy add-scc-to-user privileged \ system:serviceaccount:cloudforms:default
```

Confirm the user is associated to the privileged SCC:

```
master$ oc get scc privileged -o yaml
```
CloudForms retrieves metrics from applications deployed within OpenShift, and its leverages the data exposed by the onboard metrics infrastructure (Hawkular). Since the platform metrics are deployed in the `openshift-infra` project and CloudForms is deployed in the `cloudforms` project, they cannot communicate with each other due to use of the multitenant SDN plugin which isolates each project at a network level.

Fortunately, as a cluster administrator, you can manage the configuration of the pod overlay network to allow traffic to traverse between specific projects or be exposed to all projects. Execute the following command to join the `cloudforms` project to the `openshift-infra` project:

```
master$ oc adm pod-network join-projects cloudforms --to=openshift-infra
```

Verify the NETID is the same for these projects:

```
master$ oc get netnamespace | egrep 'cloudforms|openshift-infra'
```

### Instantiate CloudForms Templates

The components representing the containerized deployment of Red Hat CloudForms is available as a template and located on the repository server. Execute the following command to download the file to the openshift master VM and explore it:

```
master$ curl -o cfme.yaml
http://repo.osp.example.com/ocp/templates/cfme-template.yaml
master$ cat cfme-template.yaml
```

Notice how the services are set up, how variables are passed along, which containers are used, etc... This is how we are defining how CloudForms is being configured.

Add the template to the OpenShift `cloudforms` project:

```
master$ oc create -n cloudforms -f cfme-template.yaml
```

**NOTE**: The `-n cloudforms` parameters specifies the namespace explicitly. You can omit this if you are sure you are in the `cloudforms` project. Use `oc project -q` to verify.

Verify the template is available in the OpenShift environment:

```
master$ oc get -n cloudforms template cloudforms
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>PARAMETERS</th>
<th>OBJECTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudforms</td>
<td>CloudForms appliance with persistent storage</td>
<td>23 (1 blank)</td>
<td>12</td>
</tr>
</tbody>
</table>

The persistent storage required by CloudForms will be dynamically provisioned by the [OpenStack cloud provider](#).

Instantiate the template to deploy Red Hat CloudForms. Since no parameters were specified, the default values as defined in the template will be utilized.
Red Hat CloudForms will now be deployed into the cloudforms project

Validating a Successful Deployment

There are several steps that can be taken in order to verify the deployment of Red Hat CloudForms in OpenShift.

First validate that all pods are successfully running by watching the status of the pods. When all pods are running and the -deploy pods are terminated, stop the command with CTRL+C. The following output is a full deployment which took just over 4 minutes:

```
NAME                  READY     STATUS              RESTARTS   AGE
cloudforms-1-deploy   0/1       ContainerCreating   0          0s
memcached-1-deploy    0/1       ContainerCreating   0          0s
NAME                  READY     STATUS    RESTARTS   AGE
postgresql-1-deploy   0/1       Pending   0          0s
postgresql-1-deploy   0/1       Pending   0         0s
postgresql-1-deploy   0/1       ContainerCreating   0         0s
memcached-1-nih8c     0/1       Pending   0          0s
memcached-1-nih8c     0/1       Pending   0         0s
memcached-1-nih8c     0/1       ContainerCreating   0         0s
memcached-1-deploy    1/1       Running   0         7s
cloudforms-1-sc191    0/1       Pending   0          0s
cloudforms-1-sc191    0/1       Pending   0         0s
cloudforms-1-sc191    0/1       ContainerCreating   0         0s
cloudforms-1-deploy   1/1       Running   0         8s
postgresql-1-deploy   1/1       Running   0         8s
postgresql-1-244w2    0/1       Pending   0         0s
postgresql-1-244w2    0/1       Pending   0         0s
postgresql-1-244w2    0/1       ContainerCreating   0         1s
memcached-1-nih8c     0/1       Running   0         5s
memcached-1-nih8c     1/1       Running   0         10s
memcached-1-deploy    0/1       Completed   0         19s
memcached-1-deploy    0/1       Terminating   0         19s
memcached-1-deploy    0/1       Terminating   0         19s
cloudforms-1-sc191    0/1       Running   0         15s
postgresql-1-244w2    0/1       Running   0         33s
postgresql-1-244w2    1/1       Running   0         51s
postgresql-1-deploy   0/1       Completed   0         59s
postgresql-1-deploy   0/1       Terminating   0         59s
postgresql-1-deploy   0/1       Terminating   0         59s
cloudforms-1-sc191    1/1       Running   0         4m
cloudforms-1-deploy   0/1       Completed   0         4m
cloudforms-1-deploy   0/1       Terminating   0         4m
cloudforms-1-deploy   0/1       Terminating   0         4m
```

Red Hat CloudForms may take up to 5 minutes to start up for the first time as it builds the content of the initial database. As noted above, the deployment of CloudForms will be complete when the status has changed to “Running” for the containers.

Execute the following command to view the overall status of the pods in the cloudforms project

```
master$ oc status -n cloudforms
```

22
For full details of the deployed application run

```
master$ oc describe -n cloudforms pod/cloudforms-<pod_name>
```

Next, in order to validate the cloudforms pod is running with the proper privileged SCC, export the contents and inspect the openshift.io/scc annotation to confirm the privileged value is present.

```
master$ oc -n cloudforms get -o yaml pod cloudforms-<pod_name>
...
metadata:
  annotations:
    openshift.io/scc: privileged
...
```

For more details check events:

```
master$ oc -n cloudforms get events
```

You can also check volumes:

```
master$ oc -n cloudforms get pv
```

**NOTE**: If the project may have to be removed and start over again. **Only perform this task if there was an irrecoverable failure. Let and instructor know before doing this.** See Recovering From CloudForms Failed Deployment

## Accessing the CloudForms User Interface

As part of the template instantiation, a route was created that allows for accessing resources from outside the OpenShift cluster. Execute the following command to locate the name of the route that was created for CloudForms:

```
master$ oc -n cloudforms get routes
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>HOST/PORT</th>
<th>PATH</th>
<th>SERVICES</th>
<th>PORT</th>
<th>TERMINATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudforms</td>
<td>cloudforms-cloudforms.apps.example.com</td>
<td>cloudforms</td>
<td>https</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Open a web browser and navigate securely to the to the hostname retrieved above:

```
https://cloudforms-cloudforms.apps.example.com
```

**NOTE**: If you get an error such as Application Not Available see Appendix E - Troubleshooting CloudForms

Since Red Hat CloudForms in the lab environment uses a self signed certificate, add an exception in the browser to add an exception.

Use the following credentials to access the console:
Username: admin
Password: smartvm

Once successfully authenticated, you should be taken to the overview page.

Configuring the Container Provider

Red Hat CloudForms gathers metrics from infrastructure components through the use of providers. An OpenShift container provider is available that queries the OpenShift API and platform metrics. As part of the OpenShift installation completed previously, cluster metrics were automatically deployed and configured. CloudForms must be configured to consume from each of these resources.

Configure the container provider:

1. Hover your mouse over the Compute tab.
2. Once over the compute tab, additional panes will appear. (do not click anything yet)
3. Hover over Containers and then click on Providers.
4. No container providers are configured by default. Add a new container provider by clicking on Configuration (with a gear icon)
5. Lastly select Add Existing Container Provider
Start adding a new Container Provider by specifying **OCP Summit Lab** as the name and **OpenShift Container Platform** as the type.

As mentioned previously, there are two endpoints in which CloudForms retrieves metrics from. First, configure the connection details to the OpenShift API. Since CloudForms is deployed within OpenShift, we can leverage the internal service associated with API called *kubernetes* in the default project. Internal service names can be referenced across projects in the form `<service_name>,<namespace>`

Enter **kubernetes.default** in the *hostname* field and **443** in the *port* field.

The token field refers to the OAuth token used to authenticate CloudForms to the OpenShift API. The management-infra project is a preconfigured project as part of the OpenShift installation. A service account called *management-admin* is available that has access to the requisite resources needed by CloudForms. Each service account has an OAuth token associated with its account. Execute the following command to retrieve the token.

```
master$ oc serviceaccounts get-token -n management-infra management-admin
```

Copy the value returned into the token fields. Click the **Validate** button to verify the configuration.
Next, click on the *Hawkular* tab to configure CloudForms to communicate with the cluster metrics.

Enter `hawkular-metrics.openshift-infra` in the *hostname* field and `443` in the port field.

Click **Add** to add the new container provider.

You have now configured Red Hat CloudForms to retrieve metrics from OpenShift. It may take a few minutes to data to be displayed.

To force an immediate refresh of the newly added Provider:
1. Select the **OCP Summit Lab** provider icon
2. Notice all of the *Relationships* have 0 items
3. Now select the *Configuration* drop-down again
4. Choose **Refresh Items and Relationships**
5. Lastly, click the **Refresh** icon just to the left of *Configuration*
6. Now the *Relationships* should be populated with data from OpenShift
Select **Compute -> Containers -> Overview** to view the collected data. Once baseline metrics similar to what is shown below appears, you can move on to the next lab. Feel free to explore the CloudForms web console as time permits to view additional details exposed from the OpenShift cluster.

Configuring the OpenStack Cloud Provider
NOTE: This lab should be considered optional and/or stretch goal. If you are behind just skip this section and move onto the next lab.

Red Hat CloudForms can also gather metrics and infrastructure data from our Red Hat OpenStack Platform environment, in the same manner that it is now collecting information from our OpenShift Container Platform.

Configure the OpenStack cloud provider:
1. Hover your mouse over the Compute tab.
2. Once over the compute tab, additional panes will appear. (do not click anything yet)
3. Hover over Clouds and then click on Providers.
4. No cloud providers are configured by default. Add a new cloud provider by clicking on Configuration (with a gear icon)
5. Lastly select Add New Cloud Provider
6. For the Add New Cloud Provider section use these values:
   a. For Name: enter RHOSP Summit Lab
   b. For Type: choose OpenStack
   c. Leave the other items in this upper section default (including empty Region)
   d. For Tenant Mapping Enabled toggle this option to Yes
7. In the lower section labeled Endpoints in the first tab labeled Default
   a. For Hostname enter rhosp.admin.example.com
   b. Leave API Port at 5000
   c. For Security Protocol change the drop-down to Non-SSL
   d. For Username enter admin
   e. For the Password fields use summit2017
   f. Select Validate
8. In the Events section leave Ceilometer selected
9. Lastly, Add the cloud provider to CloudForms.
You have now configured Red Hat CloudForms to retrieve metrics from Red Hat OpenStack Platform. It may take a few minutes for data to be displayed.

To force a refresh of the newly added Provider:

1. Select the **RHOSP Summit Lab** provider icon
2. Notice all of the **Relationships** have 0 items
3. Now select the **Configuration** drop-down again
4. Choose **Refresh Items and Relationships**
5. Lastly, click the **Refresh** icon just to the left of **Configuration**
6. Now the **Relationships** should be populated with data from OpenStack in a few short minutes
7. Feel free to browse the new objects and get familiar with your newly connected OpenStack environment. In other words, click everything.
This concludes lab 4.
Lab 5 - Managing the Lifecycle of an Application

In this lab, you will deploy an application to Red Hat OpenShift Container Platform and use the tools previously deployed to investigate how to manage the application.

Deploy a Sample Application

One of the steps to validate the successful installation of an OpenShift Container Platform cluster is to build and deploy a sample application. OpenShift contains a number of quickstart templates that can be used to demonstrate different application frameworks along with the integration with a backend data store. One of these example applications consists of a CakePHP based web application with state stored in a MySQL database.

We will now put our cluster administrator hat aside and complete the majority of this lab as a developer by using the OpenShift web console to build and deploy the sample application.

Navigate to https://master.osp.example.com:8443 and login using the following credentials.

Username: user1
Password: summit2017

Since user1 does not currently have access to any projects, the only actions that can be taken in the web console is to create a new project. Click on the New Project button.

Enter the following information on the new project wizard:

Name: cakephp-mysql-persistent
Display Name: CakePHP MySQL Persistent
Description: Sample Project Demonstrating A CakePHP MySQL Application Using Persistent Storage

Click the Create button to create the project

You are presented with a catalog of items that you can add to your project. In a typical OpenShift cluster, this catalog would be filled with numerous programming languages emphasizing polyglot development and tools to implement Continuous Integration. In the lab environment, there is only one programming language option, PHP. Click on the PHP language to display the available options.
You are presented with one option; an OpenShift template which contains the various OpenShift components to build and deploy a CakePHP based application along with a MySQL database backed by persistent storage. The goal of this lab is to use this template to validate the build and deployment capabilities of the platform along with the dynamic allocation of Persistent Volumes for the storage of the backend database.

Click the Select button under the CakePHP + MySQL (Persistent) card which will display the images that will be used as part of this template instantiation along with parameters that can be used to inject custom logic.

One of the parameters that we will customize is the location of the Git repository containing the source code of the CakePHP application. The location will point to the Git repository that is running on the repository machine:

Modify the Git Repository URL parameter with the following value:

Git Repository URL: http://repo.osp.example.com/git/openshift/cakephp-ex.git

Scroll to the bottom of the page and select the Create button to instantiate the template

A page displaying the successful instantiation of the template will be displayed along with a set of next steps that you can take against the application. Click the Continue to Overview link to return to the project homepage.
Validating Application Deployment

After triggering instantiating the template, a new Source to Image build of the CakePHP application will begin.

View the build by selecting **Builds** and the **Builds**

Select **cakephp-mysql-persistent** to view the builds for the application. From this page, you can view build status along with the logs produced.

To investigate the status of all pods within the project, select **Application** and then **Pods**
Pods that are in a healthy condition will either have a status of Running or completed.

**NOTE**: If either the mysql or cakephp are not in a healthy state, triggering a new deployment may rectify the issue.

New deployments can be initiated from the deployments page by selecting Applications and the Deployments.

Select either mysql and then cakephp-mysql-persistent depending on the application to be deployed.

On the top right corner, click Deploy to trigger a new deployment if needed.
View Application

Click on **Overview** from the left hand navigation bar to return to the overview page.

**NOTE:** You may see an error getting metrics. This is safe to ignore for now as it will be covered in a subsequent section.

You should be able to see both the CakePHP and MySQL applications running.

The template automatically creates a route to provide external access to the application. The link is available at the top right corner of the page. Click the link to navigate to the application:

http://cakephp-mysql-persistent-cakephp-mysql-persistent.apps.example.com
Viewing Application Metrics

Application users and administrators have the ability to leverage several facilities for monitoring the state of an application deployed to the OpenShift Container Platform. While not deployed to the lab environment, OpenShift provides an aggregated logging framework based on the ELK (Elasticsearch, Fluentd and Kibana) stack. However, you can still utilize the telemetry captured by the cluster metrics mechanisms. Cluster metrics were deployed as part of the OpenShift installation and are being used to drive Red Hat CloudForms.

With the cakephp-mysql-persistent application deployed, you can use the OpenShift web console to view metrics that has been gathered by the cluster metrics facility. Since the metrics facility within the web console reaches out to Hawkular deployed in OpenShift from your web browser, you will need to perform one additional step to configure your browser to trust the self signed certificate configured before metrics can start to be displayed.

1. From the overview page, click on **Applications** on the lefthand side
2. Select **Pods**
3. Select the **Running cakephp pod**
4. Navigate to the **Metrics** tab.
Click on the link displayed which will connect to the Hawkular endpoint. Accept the self signed certificate and if successful, you will see the Hawkular logo along with additional details about the status of the service.

NOTE: After clicking on the URL noted above, it may hang for a bit as it tries to go online. It will continue after a while.

Return to the OpenShift overview page for the `cakephp-mysql-persistent` project by clicking the **Overview** link on the left side where you should be able to see metrics displaying next to each pod.
Additional details relating to the performance of the application can be viewed by revisiting the Metrics tab within each pod as previously described.

While normal consumers of the platform are able to view metrics for only the applications they have permissions to access, cluster administrators can make use of Red Hat CloudForms to view metrics from all applications deployed to the OpenShift Container platform from a single pane of glass.

Navigate through the OpenShift Web Console

With an application deployed to the OpenShift cluster, we can navigate through the various options exposed by the OpenShift web console. Use this time as an opportunity to explore the following sections at your own pace:

- Various details provided with each pod including pod details, application logs and the ability to access a remote shell
  - Hover over Applications from the left hand navigation bar and select Pods. Select one of the available pods and navigate through each of the provided tabs
- Secrets used by the platform and the CakePHP application
  - Hover over Resources from the left hand navigation bar and select Secrets
- Persistent storage dynamically allocated by the cluster to support MySQL
  - Click on the Storage tab

If desired, connect to OpenStack and view the volumes created using the steps described in a prior lab.

This concludes Lab 5
Lab 6 - Expanding the OpenShift Container Platform Cluster

In this lab, you will use Ansible Tower to add an additional application node to the OpenShift Container Platform cluster.

One of the benefits of the OpenShift Container Platform architecture is the effective scheduling of workloads onto compute resources (nodes). However, available capacity may result in the need to add additional resources. As an OpenShift cluster administrator, having a defined process for adding resources in an automated manner helps guarantee the stability of the overall cluster.

The OpenShift Container Platform provides methods for adding resources to an existing cluster, whether it be a master or node. The method for executing the scale up task depends on the installation method used for the cluster. Both methods make use of an Ansible playbook to automate the process. The execution of the playbook can be driven through Ansible Tower to further simplify adding resources to a cluster.

Review Cluster

Recall the number of nodes in the cluster by either visiting CloudForms or OpenStack.

From the OpenStack server:

```
rhosp$ openstack server list && openstack volume list
```

From the OpenShift master:

```
master$ oc get nodes
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>infra.osp.example.com</td>
<td>Ready</td>
<td>1h</td>
</tr>
<tr>
<td>master.osp.example.com</td>
<td>Ready,SchedulingDisabled</td>
<td>1h</td>
</tr>
<tr>
<td>node1.osp.example.com</td>
<td>Ready</td>
<td>1h</td>
</tr>
</tbody>
</table>

Expand the Cluster

Once again, using the web browser from the student machine, navigate to the Ansible Tower instance:

`https://tower.admin.example.com`

If the web session has not been retained from a prior lab, login with the following credentials:

Username **admin**
Password **summit2017**
After logging in, navigate to the **Templates** page and locate the **1-Provision and Scale OpenShift** workflow job template. Click the 'rocket' icon to start the job.

The workflow first creates a new OpenStack instance and once the instance has been created, the scaleup Ansible playbook will be executed to expand the cluster. The workflow job will take a few minutes to complete. Monitor the status until the workflow job completes successfully by selecting **Details** as with the initial workflow job.

**Validate the Expanded Cluster**

Once the Tower job is completed, there are multiple methods in which to validate the successful expansion of the OpenShift cluster.

First, as an OpenShift cluster administrator, you can use the OpenShift command line interface from the OpenShift master to view the available nodes and their status.

As the *root* user on the OpenShift master (*master.osp.example.com*), execute the following command to list the available nodes:

```
master$ oc get nodes
```

If successful, you should see four (4) total nodes (1 master and 3 worker nodes) with **Ready** under the **Status** column, as opposed to (3) total nodes before (1 master and 2 worker nodes).

Red Hat CloudForms can also be used to confirm the total number of nodes has been expanded to four.

From the OpenStack server:

```
rhosp$ openstack server list && openstack volume list
```

Login to CloudForms and once authenticated, hover over *Compute*, then *Containers*, and finally select **Container Nodes**. Confirm four nodes are displayed.

This concludes lab 6.
Lab 7 - Where do we go from here?

The lab may be coming to a close, but that does not mean that you need to stop once you leave the session.

Let's recap what you have accomplished during this session.

- Ansible Tower was used to execute Ansible playbooks to provision a Red Hat OpenShift Container Platform cluster
  - Instances were created with Red Hat OpenStack
  - Red Hat OpenShift Container Platform was installed and configured
    - Platform metrics were automatically deployed
- Red Hat Cloudforms was deployed within the Red Hat Container Platform cluster
  - Integrated with Red Hat OpenShift Container Platform to monitor the cluster
- Sample application using persistent storage deployed on the Red Hat OpenShift Container Platform
- Ansible Tower was used to execute Ansible platforms to expand the cluster
  - New instance deployed within Red Hat OpenStack
  - Red Hat OpenShift Container Platform node installed and cluster updated

The following resources are available for your reference:

- Source Code
- Lab Guide
- Official Documentation
  - Red Hat OpenShift Container Platform
  - Ansible Tower
  - Red Hat CloudForms
  - Red Hat OpenStack
Appendices

Appendix A - Manually Cleanup Cinder Volume

How to manually clean up a volume that will not delete with openstack volume delete

From the OpenStack server:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rhosp$ openstack volume list</td>
<td>List volumes</td>
</tr>
<tr>
<td>rhosp$ sudo -i</td>
<td>Switch to root user</td>
</tr>
<tr>
<td>rhosp# source ~/.keystonerc_admin</td>
<td>Source keystonerc_admin configuration file</td>
</tr>
<tr>
<td>rhosp# openstack volume set --state available 09d601f8-4159-4979-ae77-441920564230</td>
<td>Set volume state to available</td>
</tr>
<tr>
<td>rhosp$ mysql -u root cinder</td>
<td>Connect to MariaDB</td>
</tr>
<tr>
<td># MariaDB [cinder]&gt; delete from volumes where id='09d601f8-4159-4979-ae77-441920564230'</td>
<td>Delete volume</td>
</tr>
<tr>
<td># MariaDB [cinder]&gt; update volumes set attach_status=&quot;detached&quot; where id='09d601f8-4159-4979-ae77-441920564230'</td>
<td>Update volume attach status to detached</td>
</tr>
<tr>
<td>rhosp$ openstack volume delete 09d601f8-4159-4979-ae77-441920564230</td>
<td>Delete volume again if needed</td>
</tr>
</tbody>
</table>
# Appendix B - Script For Deploying CloudForms

These are pulled directly from [Lab 4 - Installing Red Hat CloudForms](http://repo.admin.example.com/pub/scripts/lab4-cloudforms-validation.sh)

**NOTE:** This is also available at [http://repo.admin.example.com/pub/scripts/lab4-cloudforms-validation.sh](http://repo.admin.example.com/pub/scripts/lab4-cloudforms-validation.sh)

```bash
#!/bin/bash

oc new-project cloudforms
oc config current-context for context
oc adm policy add-scc-to-user privileged \
    system:serviceaccount:cloudforms:default
oc get scc privileged -o yaml | grep cloudforms
oc adm pod-network join-projects cloudforms --to=openshift-infra
oc get netnamespace | egrep 'cloudforms|openshift-infra'
curl -O http://repo.osp.example.com/ocp/templates/cfme-template.yaml
oc create -n cloudforms -f cfme-template.yaml
oc get -n cloudforms template cloudforms
oc new-app -n cloudforms --template=cloudforms
oc -n cloudforms get pods -w
```

Proceed to [Accessing the CloudForms User Interface](http://repo.admin.example.com/pub/scripts/lab4-cloudforms-validation.sh)
Appendix C - Recovering From Failed CloudForms Deployment

The following output represents a failed deployment:

```
master$ oc get pods -w

NAME                  READY     STATUS              RESTARTS   AGE
cloudforms-1-deploy   1/1       Running             0          10s
cloudforms-1-dgvv6    0/1       ContainerCreating   0          4s
memcached-1-deploy    1/1       Running             0          10s
memcached-1-s78jr     0/1       ContainerCreating   0          2s
postgresql-1-deploy   0/1       ContainerCreating   0          10s
NAME                 READY     STATUS    RESTARTS   AGE
postgresql-1-oqoyw   0/1       Pending   0          0s
postgresql-1-oqoyw   0/1       Pending   0          0s
postgresql-1-oqoyw   0/1       ContainerCreating   0         0s
postgresql-1-deploy   1/1       Running   0         11s
memcached-1-s78jr   0/1       Running   0         18s
memcached-1-s78jr   1/1       Running   0         30s
memcached-1-deploy   0/1       Completed   0         41s
memcached-1-deploy   0/1       Terminating   0         41s
memcached-1-deploy   0/1       Terminating   0         41s
cloudforms-1-dgvv6   0/1       Running   0         1m
postgresql-1-deploy   0/1       Error     0         10m
postgresql-1-oqoyw   0/1       Terminating   0         10m
cloudforms-1-dgvv6   0/1       Running   1         10m
postgresql-1-oqoyw   0/1       Terminating   0         10m
postgresql-1-oqoyw   0/1       Terminating   0         10m
cloudforms-1-dgvv6   0/1       Running   2         19m
cloudforms-1-deploy   0/1       Error     0         20m
cloudforms-1-dgvv6   0/1       Terminating   2         20m
cloudforms-1-dgvv6   0/1       Terminating   2         20m
cloudforms-1-dgvv6   0/1       Terminating   2         20m
```

The quickest way to remedy this is to delete the project and start over:

```
master$ oc delete project cloudforms
```

Now return the lab and try again [Lab 4 - Installing Red Hat CloudForms](#)
Appendix D - Average Tower Job Times

<table>
<thead>
<tr>
<th>Tower Workflow Job</th>
<th>Ansible Playbook</th>
<th>Elapsed Time</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-Provision and Install OpenShift</td>
<td>OpenShift Pre-Install</td>
<td>00:02:38</td>
<td>Crease servers on OpenStack</td>
</tr>
<tr>
<td>0-Provision and Install OpenShift</td>
<td>OpenShift Install</td>
<td>00:12:34</td>
<td>Install OpenShift</td>
</tr>
<tr>
<td>0-Provision and Install OpenShift</td>
<td>OpenShift Post-Install</td>
<td>00:02:20</td>
<td>Setup templates and image streams for labs</td>
</tr>
<tr>
<td>1-Provision and Scale OpenShift</td>
<td>OpenShift Pre-Scaleup</td>
<td>00:01:19</td>
<td>Create server on OpenStack</td>
</tr>
<tr>
<td>1-Provision and Scale OpenShift</td>
<td>OpenShift Scaleup</td>
<td>00:05:24</td>
<td>Run openshift-ansible to add new node to the OCP</td>
</tr>
</tbody>
</table>

Appendix E - Troubleshooting CloudForms

Try to curl the CloudForms application, this may fail.

```
master$ curl -Ik https://cloudforms-cloudforms.apps.example.com
```

If this matches the web browser’s output of Application Not Available or status code of 503 then something failed in the deployment.

List the pods in the default project

```
master$ oc get pods -n default
```

List services in the default project

```
master$ oc get services
```

Try curl against the cloudforms service IP
If the router is in error state, delete it

```bash
master$ oc delete pod router -n default
```

Watch the router get deployed

```bash
master$ oc get pods -n default -w
```

The cloudforms application should work now if the router came up cleanly

```bash
master$ curl -Ik https://cloudforms-cloudforms.apps.example.com
```

Return to [Accessing the CloudForms User Interface](#)