Technical Guide - Deploying OpenShift Container Platform 3.2 in the Dell Red Hat OpenStack Cloud Solution - Version 5.0

Contents

Trademarks

DISCLAIMER: The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/ service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries, and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation or the OpenStack community.

Notes, Cautions, and Warnings

A **Note** indicates important information that helps you make better use of your system.

A **Caution** indicates potential damage to hardware or loss of data if instructions are not followed.

A **Warning** indicates a potential for property damage, personal injury, or death.

This document is for informational purposes only and may contain typographical errors and technical inaccuracies. The content is provided as is, without express or implied warranties of any kind.

Glossary

Director Node

The VM where the RHEL OSP Director Node was installed. It should have an Admin user (*osp_admin* in some cases) and the configuration files to access the OpenStack API.

docker-registry

Included in the OpenShift solution to store the docker-images provided by OpenShift and the images created when building applications. Implemented as an OpenShift application, but requires special routing considerations.

Infra/Master/Node Hosts

VM host types managed by OpenStack, and created by the Heat templates.

OpenShift Router

Routes requests for applications to proper pods. Part of the OpenShift solution which uses an HAProxy that routes HTTP1.1 requests to the proper pod for execution. Deployed on the Master hosts in this architecture, but can be deployed on Node hosts with the proper IP addressing and routing configuration.

OpenStack API IP Address

The IP address of the host serving the OpenStack APIs. It is located in the *overcloudrc* configuration file, as the `OS_AUTH_URL`.

osp_admin

The administrative user that was created on the Director Node for the Overcloud deployment. Referenced often for the installation and configuration of this solution. Found as the "user" value in the *director.cfg* file. Also found in the automation installation bastion host's *.ini* file as `settings_sample.ini:director_install_user=`.

overcloudrc

The configuration file that holds the `OS_AUTH_URL` and other critical configuration settings for accessing the OpenStack APIs of the Overcloud. Found on the Director Node, in the home directory of the *osp_admin* account.

Executive Summary

In order to meet the demands put on an organization by customers, developers need a way to provision environments and build and deploy applications with their components in a self-service fashion. IT Operations needs to be able to provide this with a secure, enterprise-grade environment that can have policy based control for automation of cluster services, scheduling and orchestration of the applications. By incorporating an OpenShift container management and OpenStack virtual machine clusters, these demands can be met quickly and effectively.

About OpenShift on OpenStack

Red Hat OpenShift Container Platform 3.2 is a Platform as a Service (PaaS) product. Its developer-centric approach enables developers to create and deploy applications with more predictability, greater ease, and less operator intervention. It manages deployments and provides application scalability services.

In the data center, OpenShift Container Platform 3.2 is deployed on Red Hat Enterprise Linux Server 7, and is comprised of application containers powered by Docker, and orchestration and management provided by Kubernetes.

Integration of OpenShift with OpenStack allows the organization to leverage existing operational techniques and organizational policies, adding a layer of deployment and redeployment flexibility not common in non-virtual deployments. This solution provides an example demonstrating how OpenShift Container Platform 3.2 basic usage can occur on a robust OpenStack infrastructure.

The configuration described in this document consists of three OpenShift master virtual machines and four OpenShift node virtual machines in single datacenter environment. The addition of more nodes is possible, but not documented here. In addition to the configuration, operational management tasks are shown to demonstrate functionality.

This version of the documentation introduces High Availability features of OpenShift Container Platform 3.2, to a robust, production-ready deployment of OpenShift on OpenStack.

OpenShift Container Platform 3.2 is hosted at *http://www.openshift.com*. It is based upon OpenShift Origin, the open source software project hosted at *http://www.openshift.org*.

Intended Audience

This technical guide shows the administrator how to build and deploy OpenShift in their Dell Red Hat OpenStack Cloud Solution. The end user is not directly addressed in this document.

Find out more about developing and managing the OpenShift Container Platform by accessing the Red Hat documentation here: *https://docs.openshift.com/enterprise/3.2/welcome/index.html*.

About OpenShift on OpenStack

Red Hat OpenShift Container Platform 3.2 is a Platform as a Service (PaaS) product. Its developer-centric approach enables developers to create and deploy applications with more predictability, greater ease, and less operator intervention. It manages deployments and provides application scalability services.

In the data center, OpenShift Container Platform 3.2 is deployed on Red Hat Enterprise Linux Server 7, and is comprised of application containers powered by Docker, and orchestration and management provided by Kubernetes.

Integration of OpenShift with OpenStack allows the organization to leverage existing operational techniques and organizational policies, adding a layer of deployment and redeployment flexibility not common in non-virtual deployments. This solution provides an example demonstrating how OpenShift Container Platform 3.2 basic usage can occur on a robust OpenStack infrastructure.

The configuration described in this document consists of three OpenShift master virtual machines and four OpenShift node virtual machines in single datacenter environment. The addition of more nodes is possible, but not documented here. In addition to the configuration, operational management tasks are shown to demonstrate functionality.

This version of the documentation introduces High Availability features of OpenShift Container Platform 3.2, to a robust, production-ready deployment of OpenShift on OpenStack.

OpenShift Container Platform 3.2 is hosted at *http://www.openshift.com*. It is based upon OpenShift Origin, the open source software project hosted at *http://www.openshift.org*.

About This Document

This document contains code and configuration samples in monospace fonts. While it is tempting for the user to copy and paste those values from this document into their system, it is inadvisable and not supported. While we make every effort to ensure that the documentation is correct and complete, documents rendered via some client applications make unpredictable changes to the actual spacing of the data elements, and lose fidelity to what a proper code or configuration setting should actually be to work properly. We see very impactful changes, for example, between the Firefox PDF display and the Adobe Acrobat reader PDF display.

Copy and paste from this document only with full understanding of the necessary formatting changes that you'll have to make. We have made efforts to provide online verbatim copies of the essential data, as well as pointing the user to appropriate external documentation to achieve the proper formatting.

Background

OpenShift will be running on multiple VMs. Some of them will require Internet access.

This solution uses several Open Source projects to install OpenShift, listed in *Table 1: Work Resources* on page 10:

Table 1: Work Resources

Project	Description	URL
OpenShift on OpenStack	Heat templates and scripts to set up the OpenStack environment and kick off the OpenShift Ansible installation. Not packaged by Red Hat.	*https://github.com/redhat-openstack/openshift-on-openstack*
OpenShift Ansible Playbooks	Ansible playbooks to deploy all manner of OpenShift on all manner of infrastructures. Packaged by Red Hat and available in repositories.	*https://github.com/openshift/openshift-ansible*
OpenShift Container Platform 3.2 Documentation	Information required to set up and manage an OpenShift Container Platform environment, as a cluster administrator or an application developer.	*https://docs.openshift.com/enterprise/3.2/welcome/index.html*

Architecture

Figure 1: Solution Architecture on page 11 displays a conceptual visualization of the Dell Red Hat OpenStack Cloud Solution architecture with OpenShift and CloudForms:

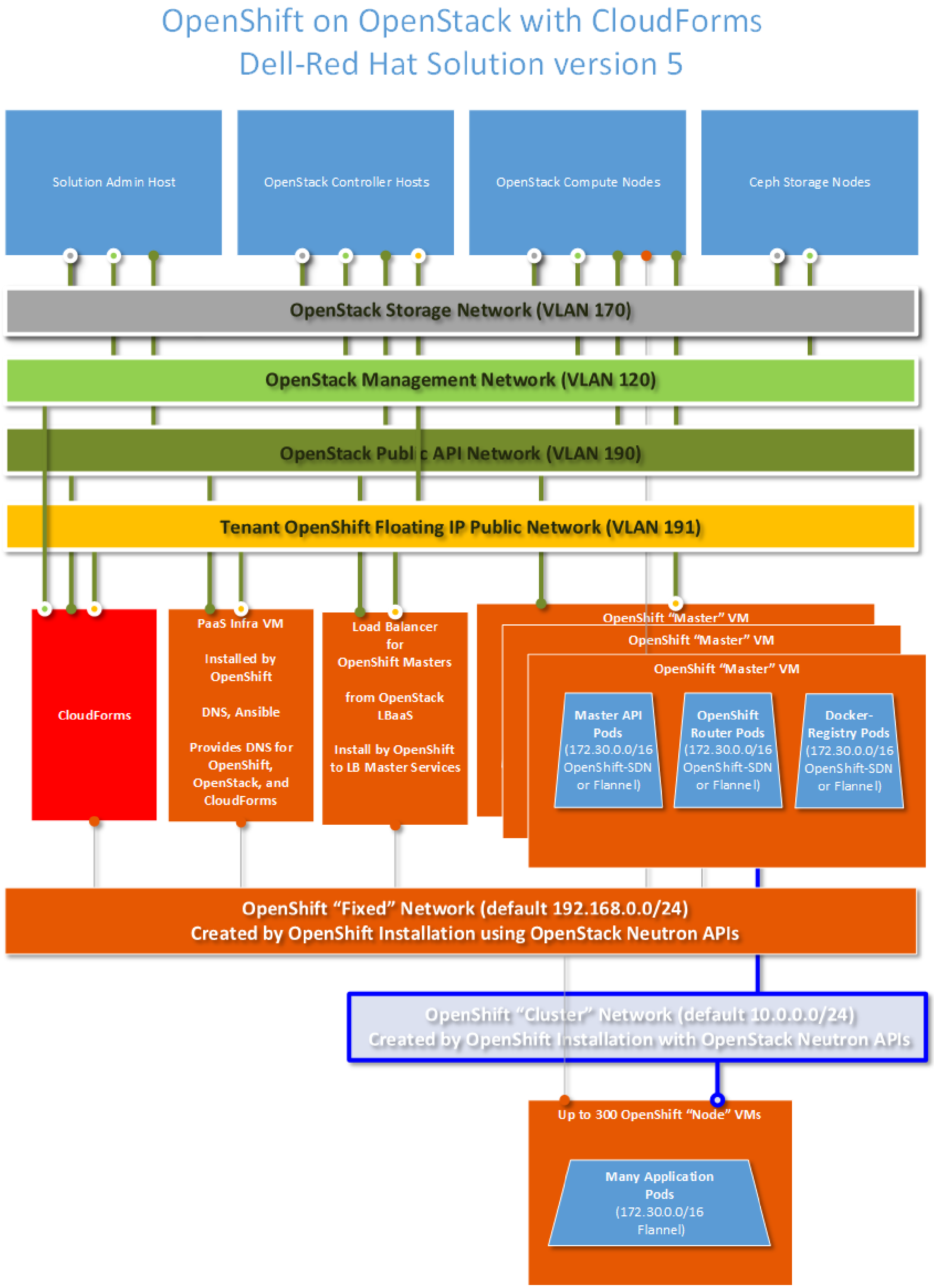

OpenShift on OpenStack with CloudForms
Dell-Red Hat Solution version 5
Solution Admin Host
OpenStack Controller Hosts
OpenStack Compute Nodes
Ceph Storage Nodes
OpenStack Storage Network (VLAN 170)
OpenStack Management Network (VLAN 120)
OpenStack Public API Network (VLAN 190)
Tenant OpenShift Floating IP Public Network (VLAN 191)
CloudForms
PaaS Infra VM
Installed by OpenShift
DNS, Ansible
Provides DNS for OpenShift, OpenStack, and CloudForms
Load Balancer for OpenShift Masters
from OpenStack LBaaS
Install by OpenShift to LB Master Services
OpenShift "Master" VM
OpenShift "Master" VM
OpenShift "Master" VM
Master API Pods (172.30.0.0/16 OpenShift-SDN or Flannel)
OpenShift Router Pods (172.30.0.0/16 OpenShift-SDN or Flannel)
Docker-Registry Pods (172.30.0.0/16 OpenShift-SDN or Flannel)
OpenShift "Fixed" Network (default 192.168.0.0/24)
Created by OpenShift Installation using OpenStack Neutron APIs
OpenShift "Cluster" Network (default 10.0.0.0/24)
Created by OpenShift Installation with OpenStack Neutron APIs
Up to 300 OpenShift "Node" VMs
Many Application Pods (172.30.0.0/16 Flannel)

Figure 1: Solution Architecture

Plan and Prepare to Install OpenShift

You must carefully plan the OpenShift installation, and prepare the environment. Topics discussed include:

- *Installation Plan* on page 12
- *Prepare the Environment* on page 12

Installation Plan

The major, critical steps for installing OpenShift on OpenStack include:

1. Prepare the environment
    a. Update OpenStack configuration
2. Director Node
    a. Clone the *openshift-on-openstack* repository.
    b. Customize the Heat templates
    c. Execute the installation
3. OpenShift master
    a. Configure and deploy a registry
    b. Configure and deploy a router
    c. Configure end users
    d. Configure management users (also for CloudForms)
4. Deploy a sample application

Prepare the Environment on page 12 delves deeper into each step.

Prepare the Environment

Follow these procedures to prepare the OpenShift installation environment:

- *Ensure Subscriptions Credentials and Pool ID* on page 12
- *Network Requirements* on page 13
- *Obtain the Guest Image* on page 13
- *Set Ample Default Quotas* on page 13
- *Configure LBaaS on All Controllers* on page 13
- *Configure Heat Services* on page 14
- *Increase Keystone Token Expiration Value* on page 16
- *Upload the Image into Glance* on page 17
- *Obtain the OpenShift-on-OpenStack Repository* on page 17
- *Prepare the Heat YAML Files* on page 18

Ensure Subscriptions Credentials and Pool ID

You must have Red Hat subscriptions to:

- Red Hat OpenStack Platform 8
- OpenShift Container Platform 3.2

Have ready the username, password, and pool ID for the Red Hat OpenStack Platform and OpenShift.

Network Requirements

Network requirements include:

- VMs require access to the Internet to download packages.
- (Optional) Complete access by the CloudForms VM to the Provisioning vLAN, to address the Undercloud VM and Overcloud Nodes.

See Integrating CloudForms 4.1 and OpenShift 3.2 in the Dell Red Hat OpenStack Cloud Solution - Version 5.0 for more information.

Obtain the Guest Image

Ensure that you have a RHEL 7.2 KVM Guest Image, in QCOW2 format.

The *rhel-guest-image-7.2-20160302.0.x86_64.qcow2* file can be found at *https://access.redhat.com/downloads/content/69/ver=/rhel---7/7.2/x86_64/product-software*.

Set Ample Default Quotas

To set default quotas:

1. Log into the **OpenStack Horizon dashboard** as the *Admin* user.
2. Select **Identities**.
3. Click to expand the **dropdown list** next to the Admin project, and then select **Manage Quotas**.
4. Set **all** your quotas **very high** (10,000 or higher.) This enables you to launch as many VMs as you require, create security groups, etc.
5. Click on **Save** to continue.

Configure LBaaS on All Controllers

To configure Load Balacing as a Service (LBaaS), perform the following procedures:

1. On **all** Controller Nodes, enable the *HAProxy* plug-in using the `service_provider` parameter, by editing the */etc/neutron/neutron_lbaas.conf* file.

   a. Find the following line in the file, and ensure that it is set as follows:

   Note: THIS LINE IS SPLIT HERE, BUT MUST BE PUT IN CONFIGURATION FILE AS ONE LINE

   ```
   service_provider=LOADBALANCER:Haproxy:neutron_lbaas.services.loadbalancer.
   drivers.haproxy.plugin_driver.HaproxyOnHostPluginDriver:default
   ```

2. On **all** Controller Nodes, enable the *LBaaS* plug-in using the `service_plugins` value, by editing the */etc/neutron/neutron.conf* file.

   a. Find the following line in the file, and ensure that it is set as follows:

   ```
   service_plugins = router,qos,lbaas
   ```

3. On **all** Controller Nodes, enable LBaaS Integration with Dashboard using the `enable_lb` option, by editing the */etc/openstack-dashboard/local_settings* file.

   a. Find the following line in the file, and ensure that it is set as follows:

   ```
   OPENSTACK_NEUTRON_NETWORK = {'enable_lb': True,
   ```

4. On **all** Controller Nodes, enable the *HAProxy* load balancer in the */etc/neutron/lbaas_agent.ini* file.

   a. Find the following line in the file, and ensure that it is set as follows:

Note: THIS LINE IS SPLIT HERE, BUT MUST BE PUT IN CONFIGURATION FILE AS ONE LINE

```
device_driver = neutron.services.loadbalancer.drivers.haproxy.
namespace_driver.HaproxyNSDriver
```

5. On **all** Controller Nodes, configure the `user_group` option in the */etc/neutron/lbaas_agent.ini* file.
   a. Find the following line in the file, and ensure that it is set as follows:

```
# The user group
# user_group = nogroup
user_group = haproxy
```

6. On **all** Controller Nodes, select the required driver in the */etc/neutron/lbaas_agent.ini* file.
   a. Find the following line in the file, and ensure that it is set as follows:

```
interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
```

7. On **any one** Controller Node **only**, restart all the services by executing the following command:

```
# pcs resource restart neutron-server
# pcs resource restart httpd.service
```

8. Log into to the Director Node as the *osp_admin* user.
9. Restart the LBaaS services by executing the following commands:

```
$ for mysystem in cntl0 cntl1 cntl2; do ssh $mysystem systemctl \
restart neutron-lbaas-agent ; done
$ for mysystem in cntl0 cntl1 cntl2; do ssh $mysystem  systemctl \
enable neutron-lbaas-agent ; done
```

10. Ensure that LBaaS is running on your system by executing the following command:

```
$ neutron agent-list
```

Configure Heat Services

Follow these procedures to set up and configure more Heat services:

- *Confirm or Create the heat-cfn Service* on page 14
- *Set Metadata Server URLs* on page 15

Confirm or Create the heat-cfn Service

The `heat-cfn` service must be present. If it is not, it must be created. For more information see *https://access.redhat.com/documentation/en/red-hat-openstack-platform/8/installation-reference/92-configure-the-orchestration-service*.

1. From the Director, Source the *overcloudrc* file to authenticate to the Overcloud, by executing the following command:

```
# . ./overcloudrc
```

2. Check the output of the following command to see if the service has been created:

```
# keystone service-list | grep heat-cfn
```

3. If there is no output, then the service has not been created. Create the service by executing the following command:

```
# keystone service-create --name heat-cfn --type cloudformation
```

4. Check the output of the following command to see if the `heat-cfn endpoint` has already been been created:

```
# keystone endpoint-list | grep heat-cfn
```

5. If there is no output (other than warnings associated with command deprecation), then create the endpoint using the value substitutions as in the following example.

   Note: The <`Heat VIP`> value can either be the VIP dedicated to Heat, or the IP address of the current Controller host on the OpenStack Public API network. In this example, the <`Heat VIP`> is *192.168.190.70*, as shown in the `ip address` command.

```
# keystone endpoint-create --service heat-cfn \
  --publicurl 'http://<OpenStack Public API IP>:8000/v1' \
  --adminurl 'http://<OpenStack Public API IP>:8000/v1' \
  --internalurl 'http://<Heat VIP>:8000/v1' \
  --region 'regionOne'
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |  http://192.168.190.70:8000/v1   |
|      id     | 996190feff1c48fbb8029f1723ff627c |
| internalurl |  http://192.168.140.71:8000/v1   |
|  publicurl  |  http://192.168.190.70:8000/v1   |
|    region   |            regionOne             |
|  service_id | d6653ae8e8294890b81b03de9056226a |
+-------------+----------------------------------+
```

Set Metadata Server URLs

The default installation has the Heat APIs incorrectly sending the `heat_metadata_server_url` and `heat_waitcondition_server_url` to clients as the *192.168.140.x* addresses of the Controllers. They should be telling the clients to access these services through the VIP (i.e., the OpenStack Public API IP address).

To set the proper metadata server URLs:

1. On **all** Controller Nodes, ensure that the `heat_metadata_server_url` is the URL with the OpenStack Public API IP Address port **8000**, by editing the */etc/heat/heat.conf* file:

```
# heat things to listen

# URL of the Heat metadata server. (string value)
#heat_metadata_server_url =
heat_metadata_server_url =http://<OpenStack Public API IP Address>:8000

# URL of the Heat waitcondition server. (string value)
#heat_waitcondition_server_url = <None>
heat_waitcondition_server_url = http://<OpenStack Public API IP
 Address>:8000/v1/waitcondition

# URL of the Heat CloudWatch server. (string value)
#heat_watch_server_url =
heat_watch_server_url =http://<OpenStack Public API IP Address>:8003
```

2. On **any one** Controller Node **only**, restart all the services by executing the following command:

```
# pcs resource show | grep heat | awk ' { print  $3 }' | xargs -n1 pcs \
resource restart
```

3. Ensure proper service restart:

```
# pcs resource | grep -A1 heat

 Clone Set: openstack-heat-engine-clone [openstack-heat-engine]
     Started: [ tan-controller-0 tan-controller-1 tan-controller-2 ]
--
 Clone Set: openstack-heat-api-clone [openstack-heat-api]
     Started: [ tan-controller-0 tan-controller-1 tan-controller-2 ]
--
 Clone Set: openstack-heat-api-cloudwatch-clone [openstack-heat-api-
cloudwatch]
     Started: [ tan-controller-0 tan-controller-1 tan-controller-2 ]
--
 Clone Set: openstack-heat-api-cfn-clone [openstack-heat-api-cfn]
     Started: [ tan-controller-0 tan-controller-1 tan-controller-2 ]
```

4. Ensure that the endpoints that can be accessed from the VMs are created by examining the endpoint with the following command:

```
# openstack
(openstack) endpoint show heat-cfn
+--------------+----------------------------------+
| Field        | Value                            |
+--------------+----------------------------------+
| adminurl     | http://192.168.140.70:8000/v1    |
| enabled      | True                             |
| id           | 84784ef00c3540a08bdd941b238f258f |
| internalurl  | http://192.168.140.70:8000/v1    |
| publicurl    | http://192.168.190.125:8000/v1   |
| region       | regionOne                        |
| service_id   | e80d0db1d99f42c784333072bf4bb61f |
| service_name | heat-cfn                         |
| service_type | cloudformation                   |
+--------------+----------------------------------+
(openstack)
```

5. Ensure that the watch server is listening by executing the following command:

```
$ ss -lntp | grep 8003

LISTEN     0      128    192.168.140.73:8003      *:*
LISTEN     0      128    192.168.190.125:8003     *:*
LISTEN     0      128    192.168.140.70:8003      *:*
```

Increase Keystone Token Expiration Value

Automated portions of the installation can exceed the default timeout of the Keystone token that the Heat client obtains when it first begins the automated installation.

To increase the token expiration value:

1. On **all** Controller Nodes, increase the value by executing the following command:

```
# . ~/overcloudrc
# openstack-config --verbose --set /etc/keystone/keystone.conf token \
 expiration 86400
```

2. On **any one** Controller Node **only**, restart the Keystone service by executing the following command:

```
# pcs resource restart openstack-keystone
```

3. Ensure that the new 24-hour setting (86400 seconds) appears by executing the following command:

```
# . ~/overcloudrc
# openstack token issue -c expires
```

Upload the Image into Glance

Now you can upload the RHEL 7.2 QCOW2 image, obtained in *Obtain the Guest Image* on page 13, into Glance.

To upload the image:

1. On the Director Node, become user *osp_admin*.
2. Source the *overcloudrc* file to authenticate to the Overcloud, by executing the following command:

```
# . ./overcloudrc
```

3. Check the Glance version by executing the following command:

```
$ glance --version
1.1.1
```

4. Create the Glance image:

Note: Some versions of the Glance client express "`--is-Public True`" differently.

```
$ glance image-create --name rhel72 --is-public True \
   --disk-format qcow2 --container-format bare \
   --file rhel-guest-image-7.2-20151102.0.x86_64.qcow2
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 486900b54f4757cb2d6b59d9bce9fe90     |
| container_format | bare                                 |
| created_at       | 2016-05-13T18:38:19.000000           |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | c0d6cf3d-196e-4fc8-a62c-39bae4a36152 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | rhel72                               |
| owner            | c1b740d8571f4cb5aa66f8ddcfdec015     |
| protected        | False                                |
| size             | 474909696                            |
| status           | active                               |
| updated_at       | 2016-05-13T18:38:26.000000           |
| virtual_size     | None                                 |
+------------------+--------------------------------------+
```

Obtain the OpenShift-on-OpenStack Repository

To obtain the *OpenShift-on-OpenStack* Github repository:

1. Log into the Director Node as user *root*.

2. Clone the repository by executing the following commands:

```
# cd
# git clone https://github.com/redhat-openstack/openshift-on-openstack.git
```

Prepare the Heat YAML Files

The installation is directed by Heat YAML files. You must prepare them with the proper parameters before they can be utilized.

To prepare the YAML files:

1. Create a file, named *openshift_parameters.yaml*, in the *openshift-on-openstack* directory that was created in *Obtain the OpenShift-on-OpenStack Repository* on page 17.
2. Paste the following into that file:

   Note: Ensure that this file is properly indented when you create your version. YAML is very whitespace sensitive.

```
parameters:
  internal_subnet: 192.168.1.0/24
  container_subnet: 10.1.0.0/24
  ssh_key_name: key_name
  server_image: rhel72
  flavor: 'm1.medium'
  external_network: public
  dns_nameserver: 8.8.4.4,8.8.8.8
  node_count: 2
  rhn_username: "dellcloudsol"
  rhn_password: ""
  rhn_pool: '8a85f98153a7bdce0153a971ce9373f1'
  extra_rhn_pools: '8a85f98153dfb4020153e15cdfc6618f'
  deployment_type: openshift-enterprise
  domain_name: "example.com"
  master_hostname: "openshift-master"
  node_hostname: "openshift-node"
  ssh_user: cloud-user
  master_docker_volume_size_gb: 25
  node_docker_volume_size_gb: 25
  master_count: 2
  master_server_group_policies: affinity
  deploy_registry: false
  os_auth_url: http://192.168.190.125:5000/v2.0
  os_username: admin
  os_password: pZ4CGyJU4XDbEuMVT7qMKdP6b
  os_tenant_name: admin

resource_registry:
  OOShift::LoadBalancer: loadbalancer_neutron.yaml
  OOShift::ContainerPort: sdn_flannel.yaml
```

3. Review *Table 2: openshift_parameters.yaml Parameters* on page 18 to ensure the correct parameters.

Table 2: openshift_parameters.yaml Parameters

Parameter: Example Value	Description
internal_subnet:	The subnet that is attached to **all** hosts.
container_subnet:	The subnet that is attached to all hosts, **except** the Infra host VM.

Parameter: Example Value	**Description**
ssh_key_name: key_name	The key name of the key uploaded to Nova with `nova keypair-add`.
server_image: rhel72	The name of the RHEL 7.2 image you uploaded to Glance.
ssh_user: cloud-user	The SSH username that the above *server_image* allows to login via `ssh_key_name`. RHEL 7.2 image uses cloud-user.
flavor: 'm1.medium'	The medium sized flavor of VM that OpenStack will launch for the Master and Node VMs.
external_network: nova	The name of the network in OpenStack that has access to the Internet. *nova* is the external network name created by the Dell Red Hat OpenStack Cloud Solution post-installation testing.
dns_nameserver: 8.8.4.4,8.8.8.8	DNS nameservers with public Internet query access. Use local DNS servers if available.
node_count: 4	The number of Nodes (OpenShift Compute VMs) to launch. Set to *2* in a POC. Set higher for multi-user deployments. OpenShift installation spreads the nodes out over *nova-compute* nodes.
rhn_username: ""	The Red Hat Network username to register your subscriptions.
rhn_password: ""	The password of your Red hat Network user.
rhn_pool: '8a85f98153a7bdce0153aaaaae9373f1'	The Pool ID that give you entitlements to OpenShift 3.2.
extra_rhn_pools: '8a85f98153dfb4020153e15cdfc6618f'	The Pool IDs that give you entitlements to the Red Hat OpenStack Platform. It provides the important configuration management and OpenStack integration with OpenShift.
deployment_type: openshift-enterprise	The OpenShift Container Platform product. The other possible option is *origin*, but OpenShift Origin is not supported by Red Hat.
domain_name: "example.com"	The domain that will be served by the DNS server on the OpenShift on OpenStack Infra host VM. The *example.com* domain is often used, because by IETF policy it is not supposed to be routed over the Internet. therefore, it is acceptable for example installations. If deploying to production, consult your naming administrators.
master_hostname: "openshift-master"	The hostname prefix that all OpenShift Master nodes will receive. For example, *delloss-openshift-master-49ouivsnsh2.example.com*. In this case *delloss* is the name of the Heat stack used in *Execute Heat Templates* on page 21.

Parameter: Example Value	Description
node_hostname: "openshift-node"	As above, but for the OpenShift Compute nodes where the containerized workloads are executed.
master_docker_volume_size_gb: 25	Volume size for Master nodes. Will be provisioned out of Cinder volumes, backed by Red Hat Ceph Storage.
node_docker_volume_size_gb: 25	Volume size for Node servers. Will be provisioned out of Cinder volumes, backed by Red Hat Ceph Storage.
master_count: 3	The number of Master replicas to launch and cluster. OpenShift has its own multi-master replication built in.
master_server_group_policies: affinity	The Master cluster policy type.
deploy_registry: false	Whether or not to deploy a docker registry.
os_auth_url: http://192.168.190.125:5000/v2.0	The `os_auth_url` is found in your *overcloudrc* file as `export OS_AUTH_URL=http://192.168.190.125:5000/v2.0`.
os_username: admin	The username in the *overcloudrc* file.
os_password: "pZ4CGyJU4XDbEuMVT7qMKdP6b"	The password in the *overcloudrc* file.
os_tenant_name: admin	The tenant name in the *overcloudrc* file.

Installation and Configuration

Now that the solution installation environment is set up, you can install and configure the solution itself by following these procedures:

1. *Execute Heat Templates* on page 21
2. *Troubleshoot and Debug Failures* on page 21

Execute Heat Templates

This procedure strings together several YAML files to feed into Heat, in order to change the deployment configurations.

To execute the Heat templates:

1. Log into the Director Node.
2. Source the *overcloudrc* file by executing the following command:

```
$ . ./overcloudrc
```

3. Execute the templates with the following command, replacing <`stackname`> with a unique name without punctuation or numbers, to define your stack:

   Note: As indicated in *Prepare the Heat YAML Files* on page 18, our example uses the stack name *delloss*.

```
$ heat stack-create <stackname> --poll -t 120 \
 -e openshift_parameters.yaml \
 -f ~/openshift-on-openstack/openshift.yaml \
 -e ~/openshift-on-openstack/env_flannel.yaml
```

You will see output for about 30 minutes over the course of the entire installation. The `--poll` argument ensures that you see updates from the Heat subsystem.

Troubleshoot and Debug Failures

If the stack-create operation fails you will see some basic debugging output:

- If the problem was in OpenStack, the error message will likely be very clear and helpful.
- If the error message is in the OpenShift Ansible deployment portion, it is likely to be obscure.

Further depth can be found in the troubleshooting document in the *openshift-on-openstack* code repository: *https://github.com/redhat-openstack/openshift-on-openstack/blob/master/README_debugging.adoc*.

Some common troubleshooting information is presented in the following topics:

- *Heat Events* on page 21
- *Time-savers* on page 22
- *Debugging in Proper Order* on page 23
- *About Ansible Log Files* on page 23

Heat Events

Here are some common Heat events troubleshooting steps:

1. Get a list of all the events leading up to your error, by executing the following command on the Director.

   ```
   heat event-list -n 2 delloss
   ```

2. Ensure that all services are running correctly. See the Dell Red Hat OpenStack Cloud Solution Deployment Guide - Version 5.0 for suggestions about how to fix these errors.
3. You see the following error message:

   ```
   Error: No such flavor
   ```

   You might be set into the Undercloud, not the Overcloud. Remember to source the *overcloudrc* file with the following command:

   ```
   . ./overcloudrc
   ```

Time-savers

When debugging you are likely to log into the VMs several times. Here are some short scripts and actions to make that much quicker.

- To copy the `ssh` key from the Director to a Controller:

  ```
  $ scp key_name.pem heat-admin@cntl0:
  key_name.pem
  ```

- Here are some short scripts for the Controller node. Normally, you must look up the IP address of the VMs. This script that enables you to:
  - `./ssh-oc <stackname>-infra` to quickly `ssh` to your stack's OpenShift Infra VM
  - `./ssh-oc delloss-openshift-master-0` to get to *master-0*
  - **get-oc <stackname-hostname>:** make it easy to get the Floating IP address of a node:

    ```
    #!/bin/bash
    nova list| grep $1 | awk '{print $(NF-1) }'
    ```

  - Example: **Easy lookup of infra node IP**

    ```
    [heat-admin@tan-controller-0 ~]$ ./get-oc delloss-infra
    192.168.191.8
    ```

  - **ssh-oc: <stackname-hostname>**: use the above script inside the following script to ssh to the desired node quickly

    ```
    #!/bin/bash
    ssh -i key_name.pem cloud-user@`./get-oc $1`
    ```

  - Example: **Easy ssh to infra node.**

    ```
    [heat-admin@tan-controller-0 ~]$ ./ssh-oc delloss-infra
    Warning: Permanently added '192.168.191.8' (ECDSA) to the list of known
     hosts.
    Last login: Fri Jun 24 11:12:30 2016 from localhost
    [cloud-user@delloss-infra ~]$
    ```

- **Best log files to check: /var/log/cloud-init.log /var/log/messages**
  - Log into the Infra VM and check some log files:

    ```
    director:  ssh cntl0
    [heat-admin@tan-controller-0 ~]$ ./ssh-oc delloss-infra
    Last login: Fri Jun 24 11:36:37 2016 from 192.168.190.128
    ```

```
[cloud-user@flannel2-infra ~]$ sudo -i
[root@flannel2-infra ~]# less /var/log/cloud-init.log
[root@flannel2-infra ~]# less /var/log/messages
```

Debugging in Proper Order

Proper order is crucial to successfully debugging the installation. Follow the procedures below in the order presented:

1. Watch `heat event-list <stackname>` for failing events until the Infra VM comes up. This takes about 10 seconds.
2. Log into the Infra VM, and watch */var/log/cloud-init.log* while it configures itself and prepares Ansible. This takes about 15 minutes.
3. From the Director Node or a Controller, watch `heat event-list <stackname>` again, for it to create the masters and nodes VMs. This takes about 10 minutes, until the long wait at `"openshift-nodes"` with the notification of `"loadbalancer" CREATE_COMPLETE`.
4. Log into the Infra VM again, and watch */var/log/ansible** for Ansible events.
   a. Find failing events with `grep '^failed:' /var/log/ansible*`, or
   b. Search for `^failed:` in an editor or pager of your choice.

It will take over 30 minutes to execute all 20,000 Ansible plays to install a 3 Master, 4 node cluster on Dell's lab Internet link and standard 13G stamp.

You may have to log into the individual VMs, to track their */var/log/cloud-init.log* and */var/log/messages* for Ansible logging locally.

About Ansible Log Files

Ansible runs on the Infra VM, and installs servers independently to hasten deployment. There can be up to three Ansible log files. If there are none, then your installation did not succeed in */var/log/cloud-init.log*, so you should be looking there first.

The */var/log/cloud-init.log* files from the masters and nodes are not brought to the Infra VM. If there is an installation problem early in the boot process of the masters or nodes, you must `ssh` to them to examine them.

The */var/log/ansible.** files are **very** verbose. You can more easily watch Ansible progress by executing the following command:

```
tail -f /var/log/ansible.* | grep '^TASK'
```

Validate the Installation

Perform the following procedures to validate the installation:

1. *List the Nodes* on page 24
2. *Ensure a Working Router* on page 24
3. *Deploy the Router Pod to a Master* on page 25

List the Nodes

Validate the installation by making sure that the system knows about its components.

To list the nodes:

1. From the Director Node, `ssh` to a Controller node.
2. Source the *overcloudrc* file:

```
$ . ./overcloudrc
```

3. Display the nodes' floating IP addresses by executing the following command:

```
$ nova list
```

4. Then `ssh` to the floating IP address of a Master node from the list:

```
$ ssh -i key_name.pem cloud-user@192.168.191.19
```

5. List the nodes by executing the following command:

```
$ oc get nodes
```

The output should display as many masters and nodes as you requested in the Heat template.

Ensure a Working Router

Sometimes the Ansible installer does not properly deploy the router. Check it by ensuring that there is a pod associated with it.

To check the router status:

1. Execute the following command:

```
$ oc status
In project default on server https://ossdell-lb.example.com:8443

svc/kubernetes - 172.30.0.1 ports 443, 53, 53

svc/router - 172.30.15.96 ports 80, 443, 1936
  dc/router deploys docker.io/openshift3/ose-haproxy-router:v3.2.0.44
    deployment #1 deployed about an hour ago - 0 pods

View details with 'oc describe <resource>/<name>' or list everything with
 'oc get all'.
```

If there are no pods in the router deployment, you must perform the procedures in *Deploy the Router Pod to a Master* on page 25.

Deploy the Router Pod to a Master

To deploy the router pod to a Master node:

Note: Our example uses *Master-0*.

1. List the nodes:

```
# oc get nodes
NAME STATUS AGE ossdell-openshift-master-0.example.com
 Ready,SchedulingDisabled 1h
ossdell-openshift-master-1.example.com
 Ready,SchedulingDisabled 1h
ossdell-openshift-node-45j22uvs.example.com            Ready
      1h
ossdell-openshift-node-9g7fs5b4.example.com            Ready
      1h
ossdell-openshift-node-v8r3iny2.example.com            Ready
      1h
```

2. Set the *Master-0* node to `schedulable`:

```
# oadm manage-node ossdell-openshift-master-0.example.com --
schedulable=true
NAME STATUS AGE
ossdell-openshift-master-0.example.com Ready 1h
```

3. Verify that *Master-0* is now schedulable:

```
# oc get nodes
NAME STATUS AGE ossdell-openshift-master-0.example.com Ready
      1h
ossdell-openshift-master-1.example.com
 Ready,SchedulingDisabled 1h
ossdell-openshift-node-45j22uvs.example.com            Ready
      1h
ossdell-openshift-node-9g7fs5b4.example.com            Ready
      1h
ossdell-openshift-node-v8r3iny2.example.com            Ready
      1h
```

4. Delete the existing router deployment configuration and service, and redeploy the router.

   a. Examine the services created by the `oc adm router` command. Most importantly, you should see `"1 pod"` running under the `dc/router`.

```
# oc delete dc router; oc delete service router; oc adm router --
selector="region=infra"
```

5. Watch the pods that are available for the complete creation of the OpenShift Router on one of the Master node VMs:

```
# oc get pods
NAME                 READY     STATUS              RESTARTS   AGE
router-2-l98lm       0/1       ContainerCreating   0          8s

# oc get pods
NAME                 READY     STATUS    RESTARTS   AGE
```

```
router-2-l98lm   1/1       Running   0          44s

# oc status
In project default on server https://ossdell-lb.example.com:8443

svc/kubernetes - 172.30.0.1 ports 443, 53, 53

svc/router - 172.30.15.96 ports 80, 443, 1936
  dc/router deploys docker.io/openshift3/ose-haproxy-router:v3.2.0.44
    deployment #2 deployed 4 minutes ago - 1 pod
    deployment #1 deployed about an hour ago

View details with 'oc describe <resource>/<name>' or list everything with
 'oc get all'.
```

The OpenShift router is now properly deployed.

Configure a Docker Registry with Persistent Storage

Perform the following procedures to configure a Docker registry with persistent storage:

1. *Create a Cinder Volume* on page 27
2. *Configure Red Hat Ceph Storage and Cinder Persistent Storage* on page 28
3. *Create the Docker-Registry* on page 29
4. *Create a Sample User in OpenShift* on page 34

Create a Cinder Volume

From the Director Node, you can create a Cinder Volume for the registry.

To create a Cinder volume:

1. Check to see if there was already a registry created. If so, note its ID.

   Note: In this example, a Cinder volume for the `docker-registry` was created: `ossdell-registry_volume-cr4v3l33qm5n`.

   ```
   $ cinder list | grep registry_volume

   | 108226c8-7de4-4905-9e67-7c61699aba0a | available | - | ossdell-
   registry_volume-cr4v3l33qm5n | 10 | - | false | False | |
   ```

2. If no volume was created, create one and note its ID:

   ```
   $ cinder create --name docker-registry 100
   ```

The output will contain information similar to *Table 3: Cinder Volume Values* on page 27.

Table 3: Cinder Volume Values

Property	Value
attachments	[]
availability_zone	nova
bootable	false
consistencygroup_id	None
created_at	2016-06-03T02:49:46.000000
description	None
encrypted	False
id	81fd05bf-8653-45eb-baf4-6df7fffb643e
metadata	{}
migration_status	None
multiattach	False
name	docker-registry
os-vol-host-attr:host	rbd:volumes@tripleo_ceph#tripleo_ceph

Property	Value
os-vol-mig-status-attr:migstat	None
os-vol-mig-status-attr:name_id	None
os-vol-tenant-attr:tenant_id	c1b740d8571f4cb5aa66f8ddcfdec015
os-volume-replication:driver_data	None
os-volume-replication:extended_status	None
replication_status	disabled
size	100
snapshot_id	None
source_volid	None
status	creating
user_id	3d0e3655d9224e5292b5f0cd3646a5ac
volume_type	None

Configure Red Hat Ceph Storage and Cinder Persistent Storage

To configure Red Hat Ceph Storage and Cinder persistent storage for the registry:

1. Access an OpenShift master.
2. Create a new file, called *pv.yaml*.
   a. Paste the calues below into *pv.yaml*. Be sure to substitute the `volume_ID` at the end for the `id` from *Table 3: Cinder Volume Values* on page 27.

      Note: Ensure that the following formatting pastes correctly into your new file. Use two spaces for each level of indentation. This is YAML.

```
apiVersion: "v1"
kind: "PersistentVolume"
metadata:
  name: "docker-reg-volume"
spec:
  capacity:
    storage: "100Gi"
  accessModes:
    - "ReadWriteOnce"
  cinder:
    fsType: "ext3"
    volumeID: "<volume_ID>"
```

3. Create the persistent volume objects, and verify them by executing the following commands:

```
# oc create -f pv.yaml
persistentvolume "docker-reg-volume" created

# oc get pv
NAME              CAPACITY ACCESSMODES STATUS    CLAIM REASON AGE
docker-reg-volume 100Gi    RWO         Available              36s
```

4. Have the docker-registry pod claim that persistent volume by creating a Persistent Volume Claim file, called *pvc.yaml*, on the same Master host:

Note: Ensure that the following formatting pastes correctly into your new file. Use two spaces for each level of indentation. This is YAML.

```
apiVersion: "v1"
kind: "PersistentVolumeClaim"
metadata:
  name: "docker-registry-pvc1"
spec:
  accessModes:
    - "ReadWriteOnce"
  resources:
    requests:
      storage: "100Gi"
```

5. Create those Persistent Volume Claim objects, and verify them by executing the following commands on the Master host:

```
# oc create -f pvc.yaml
persistentvolumeclaim "docker-registry-pvc1" created

# oc get pvc
NAME                 STATUS VOLUME            CAPACITY ACCESSMODES AGE
docker-registry-pvc1 Bound  docker-reg-volume 100Gi    RWO         2d
```

Create the Docker-Registry

To create the docker-registry right the first time, you must create a custom docker-registry YAML file with the proper IP address.

Note: Other pods cannot access this registry unless it resides on a network managed by Flannel; only one network per host is managed by Flannel. That network must be selected for the docker-registry to be accesible.

To create the docker-registry:

1. On a Master node, create a registry YAML file, called, *reg.yaml*:

```
# oc create -f pvc.yaml
persistentvolumeclaim "docker-registry-pvc1" created

# oc get pvc
NAME                 STATUS VOLUME            CAPACITY ACCESSMODES AGE
docker-registry-pvc1 Bound  docker-reg-volume 100Gi    RWO         2d
```

2. Find an appropriate `clusterIP` address by finding out which 172.30.x.0/24 address range is hosted on **this** server.
   a. Note which range is on the *docker0* bridge interface.
   b. In the example below, 172.30.9.0/24 is on the *docker0* bridge interface. So, choose an unused IP address on the 172.30.9.0/24 network.
   c. 192.168.9.215 appears to be unused, so check that by pinging that IP address and waiting for a failed response, indicating that there is no host listening on the IP address.
   d. That will be the `clusterIP` address, since that is Kubernetes' terminology.

```
# ip r

default via 192.168.10.1 dev eth0 proto static metric 100
10.10.0.0/24 dev eth1 proto kernel scope link src 10.10.0.5 metric 100
172.30.9.0/24 dev docker0 proto kernel scope link src 172.30.9.1
```

```
172.30.21.0/24 via 10.10.0.4 dev eth1
172.30.90.0/24 via 10.10.0.8 dev eth1
172.30.92.0/24 via 10.10.0.7 dev eth1
172.30.94.0/24 via 10.10.0.6 dev eth1
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.7 metric
 100
```

3. Edit *~/reg.yaml* to add a `clusterIP` and `portalIP` (same as `clusterIP`) address to the **"kind: Service"** section as in the following example, then save the file:

   Note: Ensure that the following formatting pastes correctly into your new file. Use two spaces for each level of indentation. This is YAML.

```
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    docker-registry: default
    name: docker-registry
  spec:
    clusterIP: 172.30.9.215
    portalIP: 172.30.9.215
    ports:
    - name: 5000-tcp
      port: 5000
      targetPort: 5000
```

4. Build the docker-registry objects, and kick off the build and deployment of the docker-registry pods and containers, by executing the following command:

```
# oc create -f ~/reg.yaml
```

5. Watch the deployment build for success by executing the following command:

   Note: It might take some time for the results to appear. You can keep checking by executing the `oc status` command several times.

```
# oc status
In project default on server https://ossdell-lb.example.com:8443

svc/docker-registry - 172.30.105.235:5000
  dc/docker-registry deploys registry.access.redhat.com/openshift3/ose-
docker-registry:v3.2.0.44
  deployment #1 deployed 3 minutes ago - 1 pod

svc/kubernetes - 172.30.0.1 ports 443, 53, 53

svc/router - 172.30.15.96 ports 80, 443, 1936
  dc/router deploys docker.io/openshift3/ose-haproxy-router:v3.2.0.44
    deployment #2 deployed 15 minutes ago - 1 pod
    deployment #1 deployed about an hour ago

View details with 'oc describe <resource>/<name>' or list everything with
 'oc get all'.
```

6. Now that the docker-registry is running, update it to use the Persistent Volume Claim by executing the following commands on the Master host:

```
# oc volume deploymentconfigs/docker-registry --add \
--name=docker-registry-volume -t pvc \
--claim-name=docker-registry-pvc1 \
--overwrite deploymentconfigs/docker-registry
```

```
# oc describe dc/docker-registry
-----Output truncated-----
  Volumes:
  registry-storage:
    Type:       EmptyDir (a temporary directory that shares a pod's
 lifetime)
    Medium:
  docker-registry-volume:
    Type:       PersistentVolumeClaim (a reference to a
 PersistentVolumeClaim in the same namespace)
    ClaimName:  docker-registry-pvc1
    ReadOnly:   false
-----Output truncated-----
```

7. You can observe the *docker-registry-2-deploy* pod coming up to deploy your docker-registry again with the persistent storage attached, by executing the following commands:

```
# oc status
In project default on server https://ossdell-lb.example.com:8443

svc/docker-registry - 172.30.105.235:5000
  dc/docker-registry deploys registry.access.redhat.com/openshift3/ose-
docker-registry:v3.2.0.44
    deployment #2 pending 41 seconds ago
    deployment #1 deployed 34 minutes ago - 1 pod

svc/kubernetes - 172.30.0.1 ports 443, 53, 53

    svc/router - 172.30.15.96 ports 80, 443, 1936
  dc/router deploys docker.io/openshift3/ose-haproxy-router:v3.2.0.44
    deployment #2 deployed 46 minutes ago - 1 pod
    deployment #1 deployed 2 hours ago

View details with 'oc describe <resource>/<name>' or list everything with
 'oc get all'.
# oc get pods
NAME                     READY STATUS            RESTARTS AGE
docker-registry-1-vghsw  1/1   Running           0        33m
docker-registry-2-deploy 0/1   ContainerCreating 0        46s
router-2-l98lm           1/1   Running           0        43m
```

8. Examine the docker-registry, and ensure that its second deployment comes up cleanly, by executing the following commands:

```
# oc get pods
NAME                    READY STATUS  RESTARTS AGE
docker-registry-2-xjmuo 1/1   Running 0        2m

# oc describe pod docker-registry-2-xjmuo

-----Output truncated-----

Events: (truncated to show only From and Message fields)
  From                                                  Message
  ----                                                  --------

  {default-scheduler}                                   Successfully
 assigned docker-registry-2-xjmuo to ossdell-openshift-node-
vm4047k3.example.com
  {kubelet ossdell-openshift-node-vm4047k3.example.com} Container image
 "registry.access.redhat.com/openshift3/ose-docker-registry:v3.2.0.20"
 already present on machine
```

```
   {kubelet ossdell-openshift-node-vm4047k3.example.com} Created container
  with docker id 2a204472c9fc
   {kubelet ossdell-openshift-node-vm4047k3.example.com} Started container
  with docker id 2a204472c9fc

# oc describe service docker-registry
Name:                   docker-registry
Namespace:              default
Labels:                 docker-registry=default
Selector:               docker-registry=default
Type:                   ClusterIP
IP:                     172.30.19.92
Port:                   5000-tcp        5000/TCP
Endpoints:              10.1.6.2:5000
Session Affinity:       ClientIP
No events.

# oc get pods
NAME                     READY STATUS            RESTARTS AGE
docker-registry-1-vghsw  1/1   Running           0        34m
docker-registry-2-deploy 1/1   Running           0        1m
docker-registry-2-po9az  0/1   ContainerCreating 0        28s
router-2-l98lm           1/1   Running           0        43m

# oc get pods
NAME                    READY  STATUS  RESTARTS AGE
docker-registry-2-po9az 1/1    Running 0        55s
router-2-l98lm          1/1    Running 0        44m
```

Note: By using `PersistentVolumes`, these docker-registry pods can fail at any time, and the underlying replicated storage is still intact. OpenShift will quickly bring up another docker-registry, thanks to the deployment configuration: docker-registry.

9. To test that the docker-registry has been properly deployed and networked, `curl` the docker-registry **ClusterIP** that you set above from the Master host, and from any of the other OpenShift masters or nodes:

```
# curl -v http://<ClusterIP>:5000
* About to connect() to 172.30.207.92 port 5000 (#0)
* Trying 172.30.207.92...
* Connected to 172.30.207.92 (172.30.207.92) port 5000 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.29.0
> Host: 172.30.207.92:5000
> Accept: */*
>
< HTTP/1.1 200 OK
```

10. Make sure your registry is populated by image streams and templates, and very importantly, have the same correct `ClusterIP` for the docker-registry, by executing the following commands on a Master VM:

```
# oc get is
NAME         DOCKER REPO                                     TAGS UPDATED
jenkins      172.30.21.10:5000/default/jenkins
mongodb      172.30.21.10:5000/default/mongodb
mysql        172.30.21.10:5000/default/mysql
nodejs       172.30.21.10:5000/default/nodejs
perl         172.30.21.10:5000/default/perl
php          172.30.21.10:5000/default/php
postgresql   172.30.21.10:5000/default/postgresql
python       172.30.21.10:5000/default/python
ruby         172.30.21.10:5000/default/ruby
```

11. If the docker-registry's `ClusterIP` does not match the docker-registry `ClusterIP`, or there is no output from the command perform the following steps:

    a. Remove all the image streams and templates by executing the following once on a Master VM:

    ```
    # oc delete is --all
    ```

    b. On **each** Master, restart all the OpenShift processes:

    ```
    # systemctl restart atomic-openshift-node
    # systemctl restart atomic-openshift-master-controllers
    # systemctl restart atomic-openshift-master-api
    ```

    c. Recreate the image streams by executing the following commands on a master-0

    ```
    # oc create -f /usr/share/openshift/examples/image-streams/  \
    image-streams-rhel7.json

    # oc get is

    NAME       DOCKER REPO                          TAGS             UPDATED
    jenkins    172.30.21.10:5000/default/jenkins    1,latest         8
     minutes ago
    mongodb    172.30.21.10:5000/default/mongodb    2.4,2.6,latest   8
     minutes ago
    mysql      172.30.21.10:5000/default/mysql      5.5,5.6,latest   8
     minutes ago
    nodejs     172.30.21.10:5000/default/nodejs     0.10,latest      7
     minutes ago
    perl       172.30.21.10:5000/default/perl       5.16,5.20,latest 8
     minutes ago
    php        172.30.21.10:5000/default/php        5.5,5.6,latest   7
     minutes ago
    postgresql 172.30.21.10:5000/default/postgre.. latest,9.2,9.4    8
     minutes ago
    python     172.30.21.10:5000/default/python     2.7,3.3,3.4 ...  8
     minutes ago
    ruby       172.30.21.10:5000/default/ruby       latest,2.0,2.2   8
     minutes ago
    ```

    d. Reinstate the templates by executing the following commands on a master-0:

    ```
    # ls /usr/share/openshift/examples/db-templates/*.json| xargs -n 1 oc
     create -f

    Error from server: error when creating "/usr/share/openshift/examples/
    db-templates/mongodb-ephemeral-template.json": templates "mongodb-
    ephemeral" already exists
    Error from server: error when creating "/usr/share/openshift/examples/
    db-templates/mongodb-persistent-template.json": templates "mongodb-
    persistent" already exists
    template "mysql-ephemeral" created
    template "mysql-persistent" created
    template "postgresql-ephemeral" created
    template "postgresql-persistent" created
    ```

    e. Reinstate the quickstart-templates by executing the following commands on a master-0:

    ```
    # ls /usr/share/openshift/examples/quickstart-templates/*.json | xargs -
    n 1 oc create -f

    template "cakephp-example" created
    template "cakephp-mysql-example" created
    template "dancer-example" created
    ```

```
template "dancer-mysql-example" created
template "django-example" created
template "django-psql-example" created
template "jenkins-ephemeral" created
template "jenkins-persistent" created
template "nodejs-example" created
template "nodejs-mongodb-example" created
template "rails-postgresql-example" created

# oc get templates

NAME                        DESCRIPTION               PARAMETERS
 OBJECTS
cakephp-example             An example CakePHP app... 17 (8 blank) 5
cakephp-mysql-example       An example CakePHP app... 18 (3 blank) 7
dancer-example              An example Dancer appl... 10 (4 blank) 5
dancer-mysql-example        An example Dancer appl... 17 (4 blank) 7
django-example              An example Django appl... 15 (9 blank) 5
django-psql-example         An example Django appl... 16 (4 blank) 7
jenkins-ephemeral           Jenkins service, witho...
The username is 'admin' ... 4 (all set)               3
jenkins-persistent          Jenkins service, with ...
The username is 'admin' ... 5 (all set)               4
mongodb-ephemeral           MongoDB database servi... 7 (3 generated) 2
mongodb-persistent          MongoDB database servi... 8 (3 generated) 3
mysql-ephemeral             MySQL database service... 6 (2 generated) 2
mysql-persistent            MySQL database service... 7 (2 generated) 3
nodejs-example              An example Node.js app... 14 (8 blank)    5
nodejs-mongodb-example      An example Node.js app... 15 (3 blank)    7
postgresql-ephemeral        PostgreSQL database se... 6 (2 generated) 2
postgresql-persistent       PostgreSQL database se... 7 (2 generated) 3
rails-postgresql-example    An example Rails appli... 19(3 blank)     7
```

You have now successfully created a docker-registry in the OpenStack cluster.

Create a Sample User in OpenShift

The final procedure before logging into the OpenShift GUI is to create a sample user in OpenShift.

To create a sample user in OpenShift:

1. Execute the following command on **all** of the OpenShift masters, creating a password of your choosing.

   ⚠ **Caution:** You must use the same one on each host.

   ```
   # htpasswd /etc/origin/openshift-passwd osadmin
   New password:
   Re-type new password:
   Adding password for user osadmin
   ```

You can now proceed to *Configure OpenShift Web GUI Access* on page 35.

Configure OpenShift Web GUI Access

Perform the following procedures to configure OpenShift web GUI access:

1. *Make DNS World Accessible* on page 35
2. *Ensure the Wildcard Domain Resolves to the Router* on page 35
3. *Add DNS to the Bastion Host* on page 36
4. *Navigate to the OpenShift Web Console* on page 36

Make DNS World Accessible

To ensure that DNS is accessible by everyone:

1. On the Director Node execute the following command:

```
$ nova list
```

2. Note the floating IP address of the OpenShift *master-0* VM.
3. `ssh` into the Infra VM.
4. Fix `named/bind` to accept queries from anywhere, by editing the */etc/named.conf* file, changing the allow-query value to be **any;**:

```
# vi /etc/named.conf

allow-query { any; };
```

Ensure the Wildcard Domain Resolves to the Router

To enable the wildcard domain to resolve to the router:

1. On the Infra VM, edit /var/named/openshift-cluster.zone.
   a. Increment the `; Serial` entry.
   b. Add the wildcard domain entry like so: `*.cloudapps IN A <FloatingIP of Master-0>`, with the Floating IP address of the *master-0* that you noted in *Make DNS World Accessible* on page 35.

Note: Ensure that the following formatting pastes correctly into your new file. This is a BIND9 Zone File.

```
$TTL 1d
@     IN  SOA  flannel-infra openshift (
               1466012102      ; Serial <- INCREMENT THIS NUMBER
               12h     ; Refresh
               3m      ; Retry
               4w      ; Expire
               3h      ; TTL for negative replies
          )
      IN NS flannel-infra
flannel-infra  IN A  192.168.10.5
flannel-openshift-master-0  IN A  192.168.10.7
flannel-openshift-master-1  IN A  192.168.10.6
flannel-openshift-node-x9hh4188  IN A  192.168.10.11
flannel-openshift-node-4dzubun2  IN A  192.168.10.9
```

```
flannel-openshift-node-xl8df21a  IN A  192.168.10.10
flannel-lb  IN A  192.168.191.14
*.cloudapps IN A <FloatingIP of Master-0>
```

2. Restart the DNS server process to pickup all of the changes:

```
# systemctl restart named
```

Add DNS to the Bastion Host

To add DNS to your Windows bastion host:

1. In Windows, navigate to **Network and Sharing Center**, then select **Change Adapter Settings**.
2. Right-click on the **adapter that you use to connect to the OpenShift console** (usually the public network), then choose **Properties**.
3. Double-click on **Internet Protocol Version 4**, then change the **Preferred DNS Server** to the public IP address of the Infra VM.
4. Click on **OK**, then on **OK** again to close the Network Properties dialogue box and ensure that the setting was saved.
5. Run `cmd.exe` to display a command console.
6. Ensure that the DNS server was accepted by the configuration, and applied to the Network settings, by executing the following command:

```
C:\> ipconfig /all
```

7. Ensure network connectivity by executing the following command:

```
C:\> ping <infra host IP address>
```

8. Execute the following command to return the IP address of the load balancer:

```
C:\> nslookup <lb FQDN> <infra host IP address>
```

Navigate to the OpenShift Web Console

To log into the console on the bastion host:

1. In a web browser, navigate to http://openshift-lb.example.com/console.
2. Accept the Security Exception for the self-signed certificate.
3. Login with these credentials:
   a. Username: *osadmin*
   b. Password: *password* (the password that you set, above)
4. If the URL is not accessible immediately, or becomes inaccesible after some time, the keys may have been rotated. To regain accessibility:
   a. Access the OpenStack Dashboard in a web browser.
   b. Navigate to the **Admin Project > Network > Load Balancers > Members**.
   c. Ensure the status of each member is **Active**.
   d. If the status is not Active, select **Edit Member**, then save it with no changes to force OpenStack to refresh the members properly.
   e. Browse to openshift-lb.example.com:8443/console.
   f. Accept the new certificates.

5. Do not log out - you will need this access for the next procedure, *Deploy a Sample Application* on page 38.

Deploy a Sample Application

Now you can deploy a sample application in the OpenShift Container Platform web GUI.

To deploy a sample application:

1. Log into the OpenShift Enterprise console.
2. Create a new project with a name that you like.

   Note: Projects are namespaces in which you deploy one or more applications that will share resources.
3. Create a new application from the Quickstart Templates:

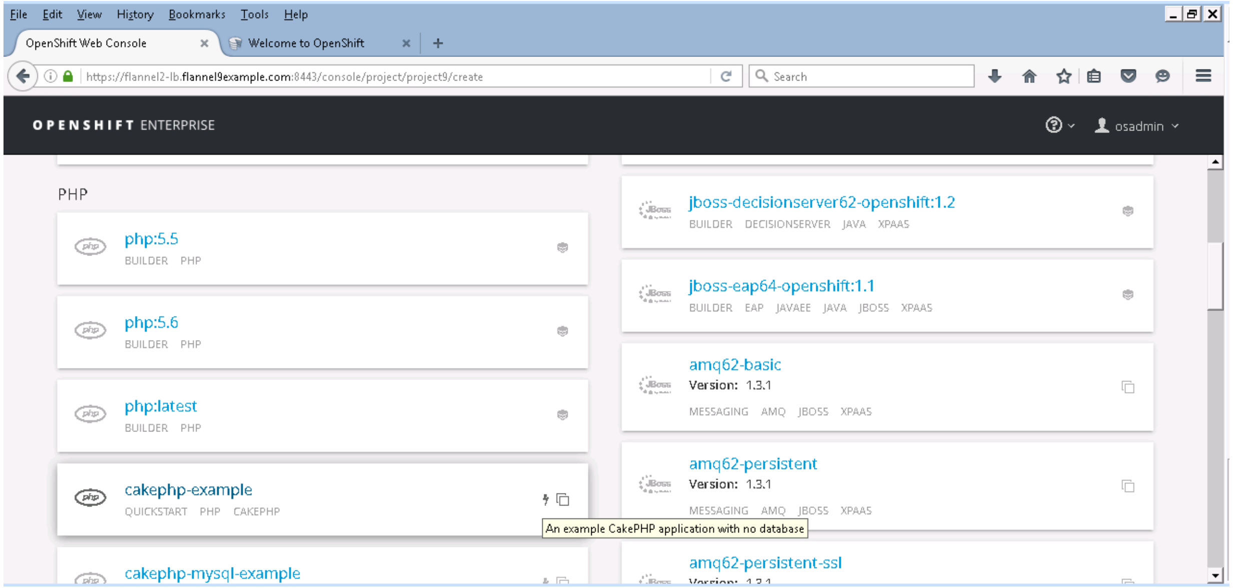
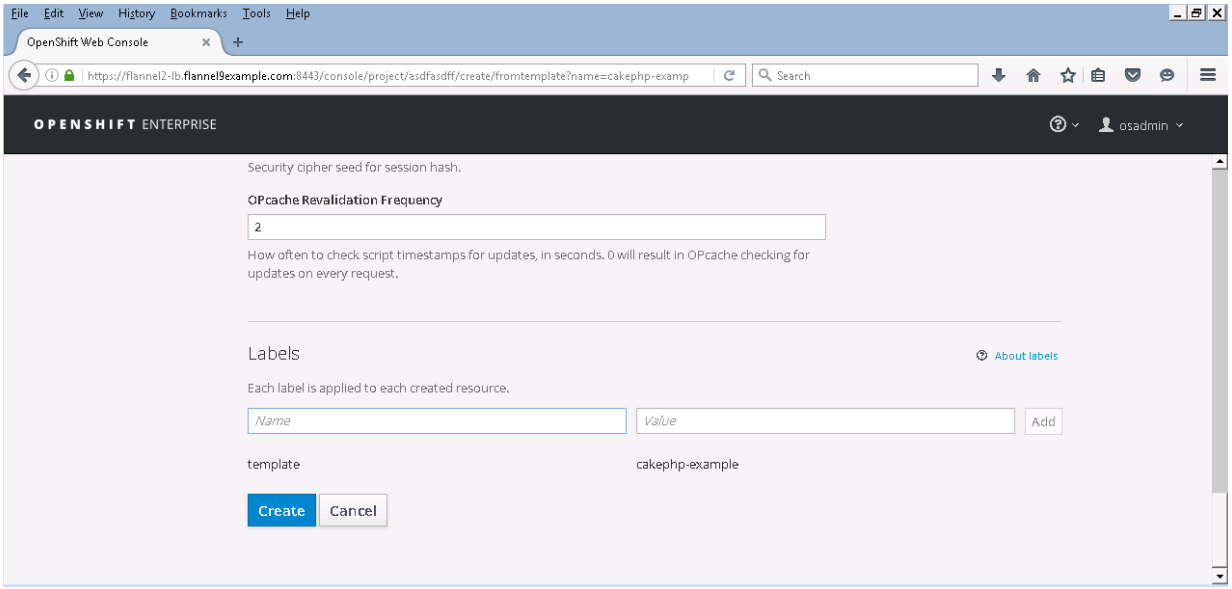

File Edit View History Bookmarks Tools Help
OpenShift Web Console
Welcome to OpenShift
https://flannel2-lb.flannel9example.com:8443/console/project/project9/create
Search
OPENSHIFT ENTERPRISE
osadmin
PHP
php:5.5
BUILDER PHP
php:5.6
BUILDER PHP
php:latest
BUILDER PHP
cakephp-example
QUICKSTART PHP CAKEPHP
An example CakePHP application with no database
cakephp-mysql-example
jboss-decisionserver62-openshift:1.2
BUILDER DECISIONSERVER JAVA XPAAS
jboss-eap64-openshift:1.1
BUILDER EAP JAVAEE JAVA JBOSS XPAAS
amq62-basic
Version: 1.3.1
MESSAGING AMQ JBOSS XPAAS
amq62-persistent
Version: 1.3.1
MESSAGING AMQ JBOSS XPAAS
amq62-persistent-ssl

Figure 2: Example Application with No Database

4. Accept all the defaults and click **Create**:


File Edit View History Bookmarks Tools Help
OpenShift Web Console
https://flannel2-lb.flannel9example.com:8443/console/project/asdfasdff/create/fromtemplate?name=cakephp-examp
Search
OPENSHIFT ENTERPRISE
osadmin
About labels
Name
Value
Add
Create
Cancel

Security cipher seed for session hash.

OPcache Revalidation Frequency

2

How often to check script timestamps for updates, in seconds. 0 will result in OPcache checking for updates on every request.

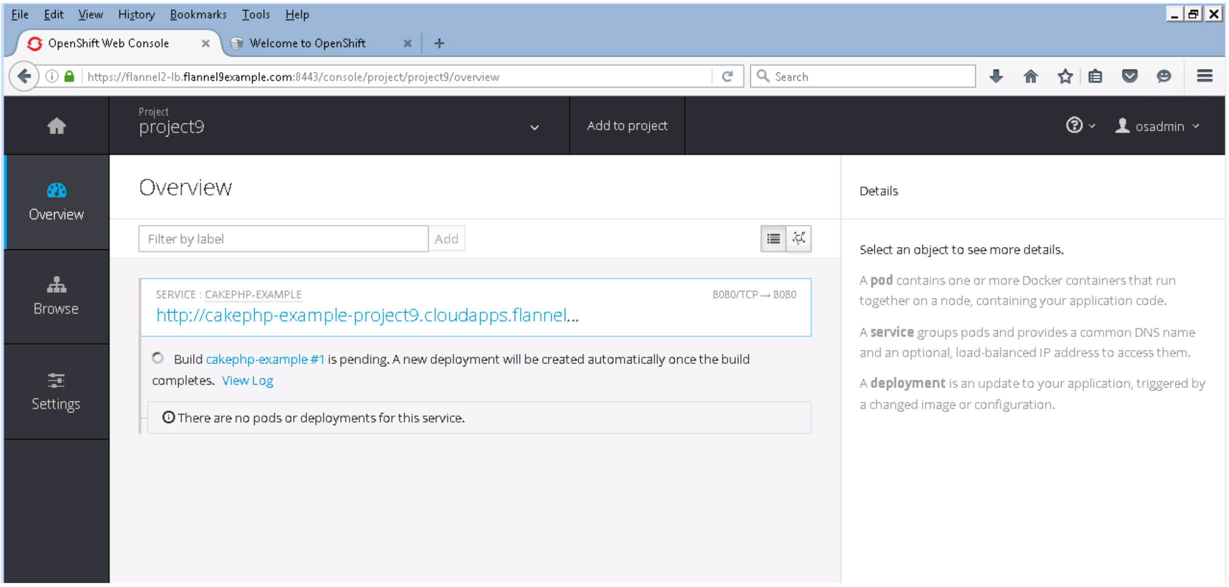
Labels

Each label is applied to each created resource.

template cakephp-example

Figure 3: New Application

5. Click through the informational screen, and watch the application build on the overview screen:



SERVICE : CAKEPHP-EXAMPLE 8080/TCP → 8080

http://cakephp-example-project9.cloudapps.flannel...

Build cakephp-example #1 is pending. A new deployment will be created automatically once the build completes. View Log

There are no pods or deployments for this service.

Select an object to see more details.

A **pod** contains one or more Docker containers that run together on a node, containing your application code.

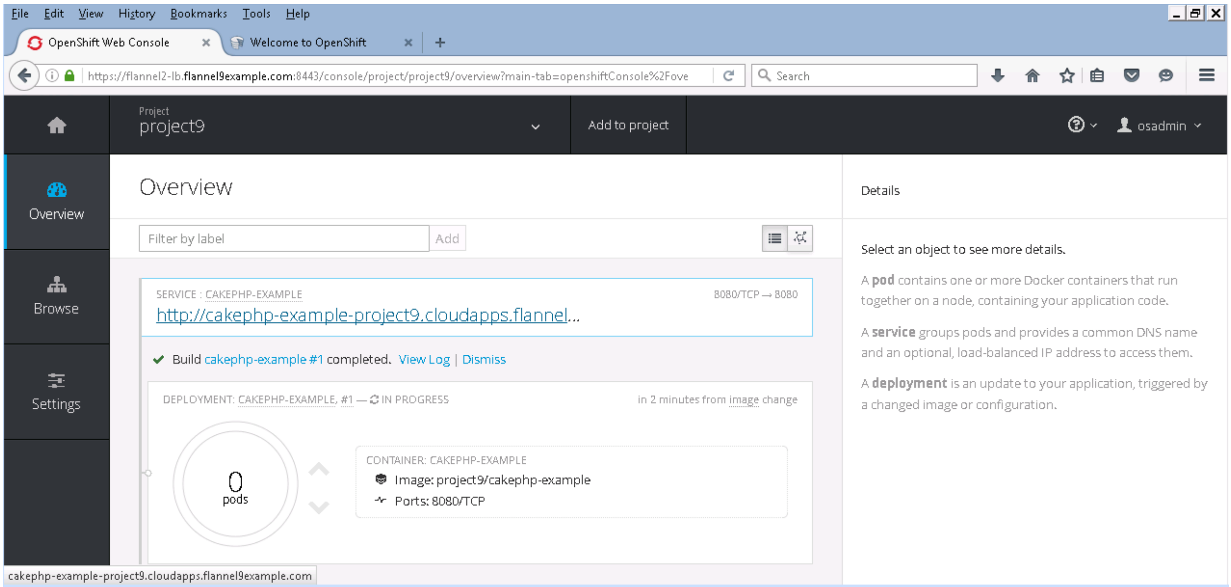
A **service** groups pods and provides a common DNS name and an optional, load-balanced IP address to access them.

A **deployment** is an update to your application, triggered by a changed image or configuration.

Figure 4: Overview

6. In a matter of moments, the build is complete; the deployment completes shortly thereafter.

Note: You can click the **View Log** link to see the progress.


File Edit View History Bookmarks Tools Help
OpenShift Web Console
Welcome to OpenShift
https://flannel2-lb.flannel9example.com:8443/console/project/project9/overview?main-tab=openshiftConsole%2Fove
Search
Project
project9
Add to project
osadmin
Overview
Browse
Settings
Filter by label
Add
Details
0
pods

SERVICE : CAKEPHP-EXAMPLE 8080/TCP → 8080

http://cakephp-example-project9.cloudapps.flannel...

Build cakephp-example #1 completed. View Log | Dismiss

DEPLOYMENT: CAKEPHP-EXAMPLE, #1 — IN PROGRESS in 2 minutes from image change

CONTAINER: CAKEPHP-EXAMPLE

Image: project9/cakephp-example

Ports: 8080/TCP

Select an object to see more details.

A **pod** contains one or more Docker containers that run together on a node, containing your application code.

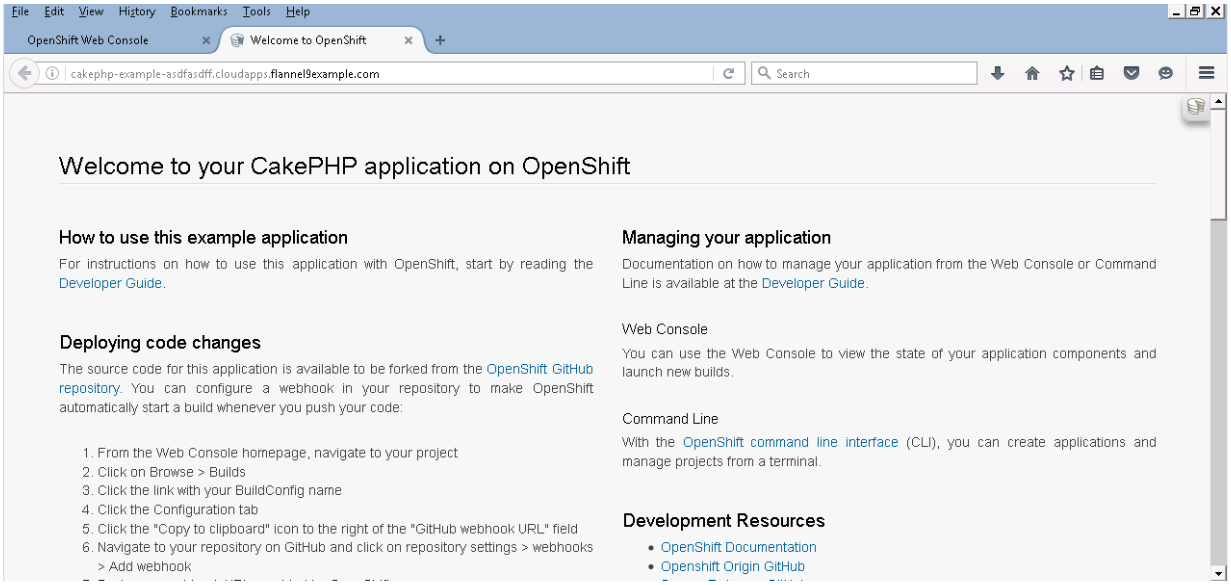
A **service** groups pods and provides a common DNS name and an optional, load-balanced IP address to access them.

A **deployment** is an update to your application, triggered by a changed image or configuration.

cakephp-example-project9.cloudapps.flannel9example.com

Figure 5: Build Complete

7. Click on the **application's link** to display the working application:


File Edit View History Bookmarks Tools Help
OpenShift Web Console
Welcome to OpenShift
cakephp-example-asdfasdff.cloudapps.flannel9example.com
Search

Welcome to your CakePHP application on OpenShift

How to use this example application

For instructions on how to use this application with OpenShift, start by reading the Developer Guide.

Deploying code changes

The source code for this application is available to be forked from the OpenShift GitHub repository. You can configure a webhook in your repository to make OpenShift automatically start a build whenever you push your code:

1. From the Web Console homepage, navigate to your project
2. Click on Browse > Builds
3. Click the link with your BuildConfig name
4. Click the Configuration tab
5. Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field
6. Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook

Managing your application

Documentation on how to manage your application from the Web Console or Command Line is available at the Developer Guide.

Web Console

You can use the Web Console to view the state of your application components and launch new builds.

Command Line

With the OpenShift command line interface (CLI), you can create applications and manage projects from a terminal.

Development Resources

- OpenShift Documentation
- Openshift Origin GitHub

Figure 6: Working Application

The OpenShift Container Platform is now deployed in the Dell Red Hat OpenStack Cloud Solution.

Next Steps

Now that the OpenShift Container Platform is now deployed in the Dell Red Hat OpenStack Cloud Solution, follow the procedures in the guide listed below to deploy Red Hat CloudForms on the Dell Red Hat OpenStack Cloud Solution:

- Technical Guide - Deploying CloudForms 4.1 in the Dell Red Hat OpenStack Cloud Solution - Version 5.0

Getting Help

This appendix details contact and reference information for the Dell Red Hat® OpenStack Cloud Solution with Red Hat OpenStack Platform.

Contacting Dell

For customers in the United States, call 800-WWW-DELL (800-999-3355).

Note: If you do not have an active Internet connection, you can find contact information on your purchase invoice, packing slip, bill, or Dell product catalog.

Dell provides several online and telephone-based support and service options. Availability varies by country and product, and some services may not be available in your area. To contact Dell for sales, technical support, or customer service issues:

1. Visit *dell.com/support*.
2. Click your country/region at the bottom of the page. For a full listing of country/region, click **All**.
3. Click **All Support** from the **Support** menu.
4. Select the appropriate service or support link based on your need.
5. Choose the method of contacting Dell that is convenient for you.

References

Additional information can be obtained at *http://www.dell.com/en-us/work/learn/openstack-cloud* or by e-mailing *openstack@dell.com*.

If you need additional services or implementation help, please contact your Dell sales representative.

To Learn More

For more information on the Dell Red Hat® OpenStack Cloud Solution visit *http://www.dell.com/learn/us/en/04/solutions/red-hat-openstack*.