Reference Architectures 2017

Deploying Red Hat OpenShift Container Platform 3.6 on Red Hat Virtualization 4

Last Updated: 2018-01-23

Reference Architectures 2017 Deploying Red Hat OpenShift Container Platform 3.6 on Red Hat Virtualization 4

Chandler Wilkerson
refarch-feedback@redhat.com

Legal Notice

Abstract

The purpose of this document is to provide guidelines and considerations for installing and configuring Red Hat OpenShift Container Platform 3.6 on Red Hat Virtualization 4.

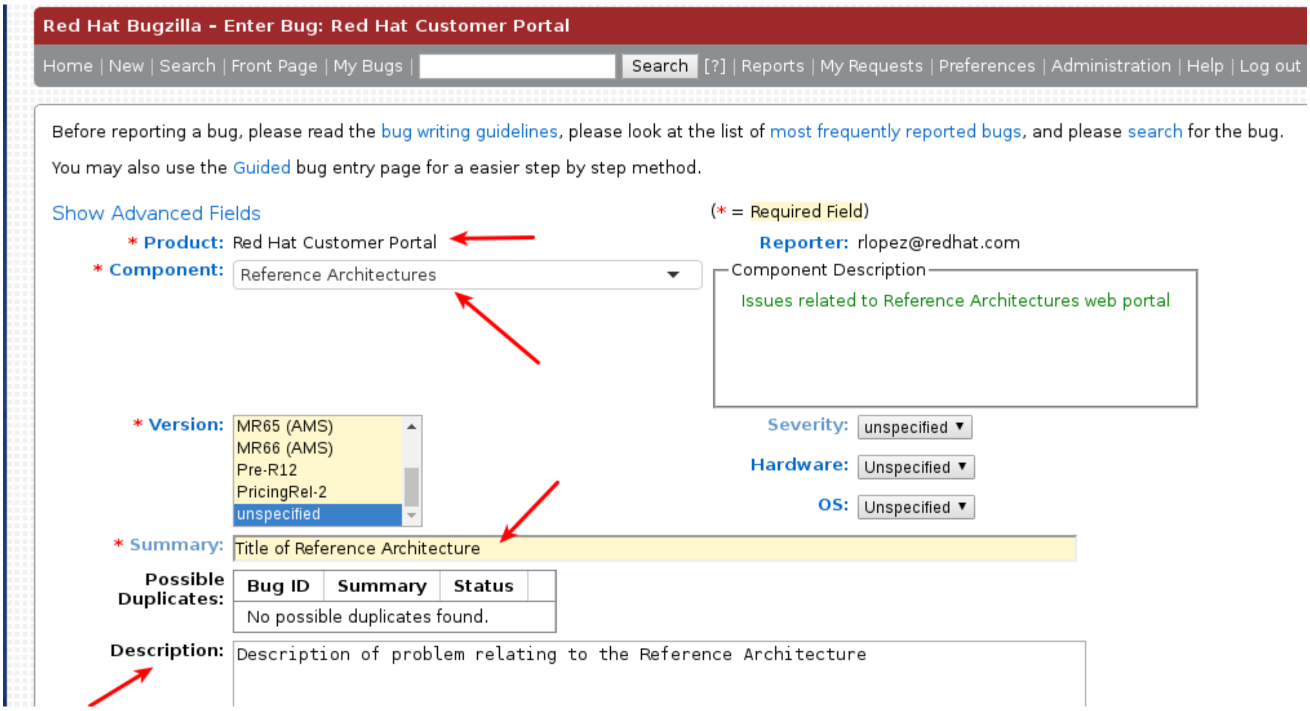
Table of Contents

COMMENTS AND FEEDBACK

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architecture. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers. Feedback on the papers can be provided by emailing refarch-feedback@redhat.com or creating a bug report at bugzilla.redhat.com. Please refer to the title within the email or bug report.

Providing feedback using a bug report allows both the person reporting the bug and us to track the feedback.

When filing a bug report for a reference architecture, create the bug under the Red Hat Customer Portal product and select the Component labeled Reference Architectures. Within the Summary, enter the title of the reference architecture. Within the Description, provide a URL (if available) along with any feedback.



CHAPTER 1. EXECUTIVE SUMMARY

Red Hat OpenShift Container Platform 3 is built around a core of application containers powered by **Docker**, with orchestration and management provided by **Kubernetes**, on a foundation of **Red Hat Enterprise Linux Atomic Host** and **Red Hat Enterprise Linux**. **OpenShift Origin** is the upstream community project that brings it all together along with extensions to accelerate application development and deployment.

This reference architecture provides a comprehensive example demonstrating how **Red Hat OpenShift Container Platform 3** can be set up to take advantage of the native high availability capabilities of Kubernetes and **Red Hat Virtualization** in order to create a highly available **OpenShift Container Platform** environment. The configuration consists of three **OpenShift Container Platform** master nodes, three **OpenShift Container Platform** infrastructure nodes, and two **OpenShift Container Platform** application nodes running as virtual machines within a highly available, self-hosted **Red Hat Virtualization** cluster. In addition, this document demonstrates key management tasks centered around validation and expansion of the environment.

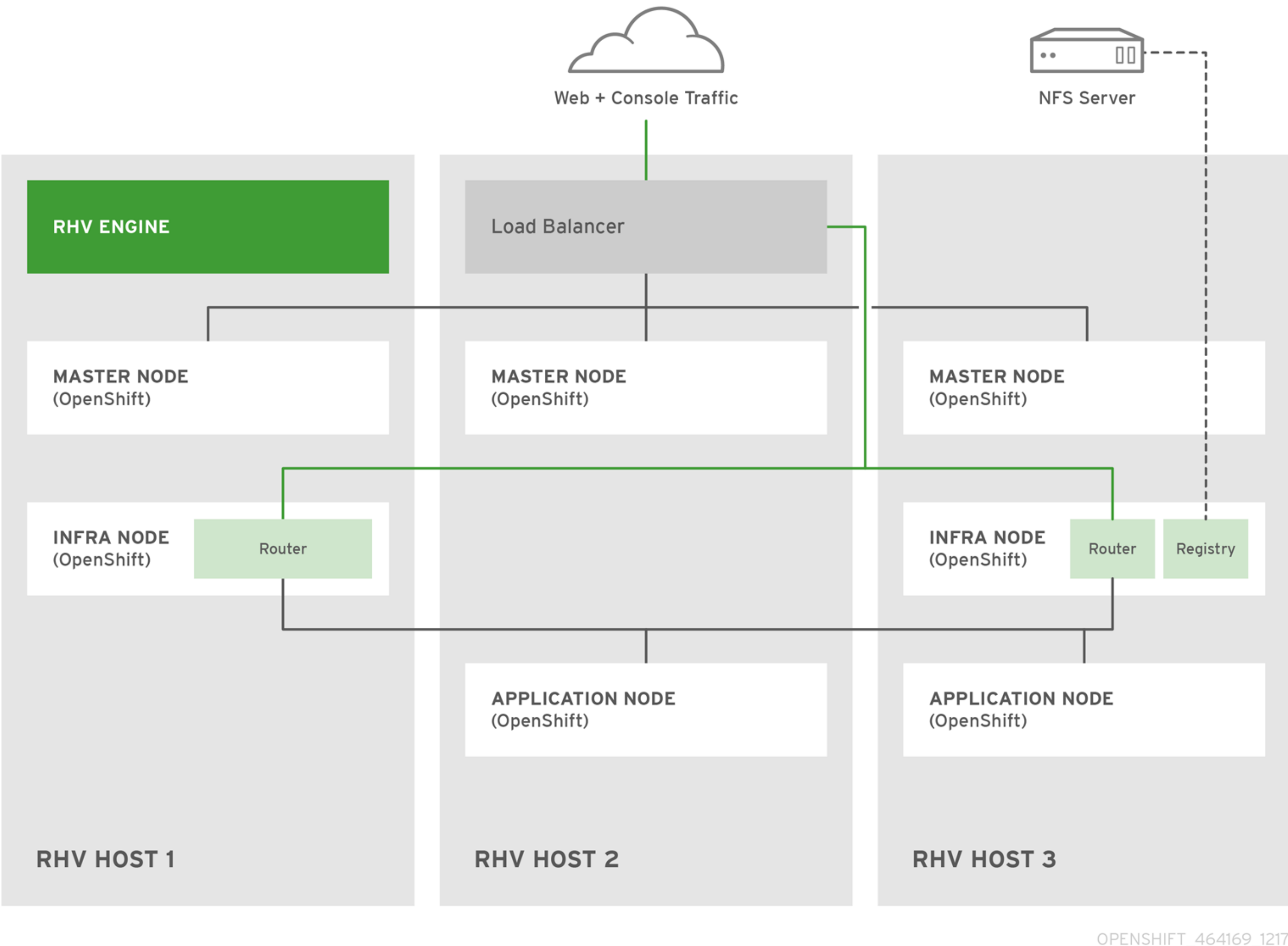
The target audience for this reference architecture would be a system administrator or system architect with solid background with **Red Hat Virtualization**. Experience with containers and **OpenShift Container Platform** helps, but is not required.

CHAPTER 2. COMPONENTS AND CONFIGURATION

This chapter describes the highly available **Red Hat OpenShift Container Platform 3** reference architecture environment.

The image below provides a high-level representation of the components within this reference architecture.

Figure 2.1. Red Hat OpenShift on RHV Architecture



By using **Red Hat Virtualization** in a self-hosted configuration, this reference architecture provides a complete high-availability environment with three physical hosts. In a production environment, it is recommended as a best practice to separate the **Red Hat Virtualization** cluster running **OpenShift Container Platform** from the cluster running the management engine. Proper set up of the **Red Hat Virtualization** environment is beyond the scope of this document. A guide to best practices in setting up **Red Hat Virtualization** can be found in the Links section of the Appendix at the end of this document.

- Master nodes host the **OpenShift Container Platform** master components such as **ETCD** and the **OpenShift Container Platform** API.
- Infrastructure nodes are used for the **OpenShift Container Platform** infrastructure elements like the **OpenShift Container Platform router** and **OpenShift Container Platform integrated registry**.
- Application nodes are for users to deploy their containers.

Authentication in this reference architecture is managed by the **htpasswd** identity provider. **OpenShift Container Platform** can be configured to use any of the supported identity providers (including **GitHub**, **Google** or **LDAP**).

Storage in this reference architecture makes use of a **Red Hat Virtualization storage domain** for virtual machine local disk images, and a network file system (**NFS**) server for the registry.

Networking is kept simple, using the **Red Hat Virtualization** cluster's own **ovirtmgmt** network for the reference architecture. A **Red Hat Satellite** server provides **DNS**, **DHCP**, and **RHSM** subscriptions for the virtual machines.

A **load balancer** (**LB**) virtual machine provisioned during the OpenShift installation provides balanced access to **OpenShift Container Platform** services. For the administrative interface, the **LB** forwards requests to the **master** nodes. For application requests, the **LB** forwards to the **infrastructure** nodes where the **router** pods will proxy requests on to the relevant application pods served on the **application** nodes.

In order to properly direct browser traffic to the correct service, DNS must be set up by the systems administrator to point requests to the **LB** host as described later in this chapter.

IMPORTANT

The use of a **load balancer** virtual machine introduces a single point of failure into the environment. In a production environment, it is preferred to integrate an external load balancer appliance into the **OpenShift Container Platform** environment. For more information about integrating an external load balancer, see the **Red Hat Containers & PaaS Community of Practice** guide called Installing a Highly-Available OpenShift Cluster.

This reference architecture breaks down the deployment into three separate phases.

- Phase 1: Provision the Virtual Machines on **Red Hat Virtualization**
- Phase 2: Install **OpenShift Container Platform** on **Red Hat Virtualization**
- Phase 3: Post deployment activities

For Phase 1, provisioning of the virtual machines employs a series of **Ansible** playbooks provided in the openshift-ansible-contrib **GitHub** repository in conjunction with **Ansible** roles provided by the ovirt-ansible **GitHub** repository.

Phase 2 sees the provisioning of **OpenShift Container Platform** using **Ansible** playbooks included in the **openshift-ansible-playbooks** RPM package. The last part of phase 2 deploys the load balancer, routers, and registry.

Finally, Phase 3 concludes the deployment with a set of manually-run verifications in the form of an **Ansible** playbook and command-line tools.

NOTE

The scripts provided in the **GitHub** repository are not supported by **Red Hat**. They merely provide a mechanism that can be used to build out an **OpenShift Container Platform** environment.

2.1. RED HAT VIRTUALIZATION NODE DETAILS

Linux virtual machines in **Red Hat Virtualization** are created from a **Red Hat Enterprise Linux** virtual machine image according to a set of profiles described in `reference-architecture/rhv-ansible/playbooks/vars/ovirt-vm-infra.yaml`. Each profile is created with the base 10 GiB operating system disk, plus two extra virtual disks, a `docker storage disk` and a `local volume disk` where data persistence is not guaranteed.

Table 2.1. Defaults for Node Virtual Machines

Node Type	CPU	Memory	High Availability?
`master`	2	16 GiB	True
`infrastructure`	2	8 GiB	True
`application`	2	8 GiB	False
`load balancer`	2	8 GiB	True

Table 2.2. Defaults for Node Storage

Type	Name	Mount Point	Size	Purpose
`operating system disk`	vda	`/boot` & `/`	60 GiB	Root file system from qcow2 image
`docker disk`	vdb	LVM	100 GiB	Docker images storage
`localvol disk`	vdc	`/var/lib/origin/openshift.local.volumes`	50 GiB	**OpenShift Container Platform** `emptyDir` volumes on all nodes except for master

The following is a snippet from the `ovirt-vm-infra.yaml` configuration file that defines the profile for a non-master node:

```
node_vm:
  cluster: "{{ rhv_cluster }}"
  template: rhel7
  memory: 8GiB
  cores: 2
  disks:
    - size: 100GiB
      storage_domain: "{{ rhv_data_storage }}"
      name: docker_disk
      interface: virtio
```

```
    - size: 50GiB
      storage_domain: "{{ rhv_data_storage }}"
      name: localvol_disk
      interface: virtio
```

The node profile is then applied to the virtual machine entries as shown below, under the **`vms:`** variable definition. Note the application of the **`openshift_infra`** tag. This enables Ansible to identify the nodes for the **`infrastructure`** role while installing **OpenShift Container Platform**.

```
vms:

... [OUTPUT ABBREVIATED] ...

  # Infra VMs
  - name: openshift-infra-0
    tag: openshift_infra
    profile: "{{ node_vm }}"
    cloud_init:
      host_name: "openshift-infra-0.{{ public_hosted_zone }}"
      authorized_ssh_keys: "{{ root_ssh_key }}"
  - name: openshift-infra-1
    tag: openshift_infra
    profile: "{{ node_vm }}"
    cloud_init:
      host_name: "openshift-infra-1.{{ public_hosted_zone }}"
      authorized_ssh_keys: "{{ root_ssh_key }}"
```

The following is a sample output in a **OpenShift Container Platform** **`infrastructure`** virtual machine deployed using this reference architecture where the mount points as well as the disks can be seen as described:

NOTE

The name column below has been truncated for ease of reading.

```
[root@openshift-infra-0 ~]# lsblk

NAME                        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                          11:0    1 1024M  0 rom
sr1                          11:1    1  374K  0 rom
vda                         253:0    0   10G  0 disk
└─vda1                      253:1    0   10G  0 part /
vdb                         253:16   0  100G  0 disk
└─vdb1                      253:17   0  100G  0 part
  └─docker--vol-dockerlv 252:0    0   40G  0 lvm  /var/lib/docker
vdc                         253:32   0   50G  0 disk
/var/lib/origin/openshift.local.volumes
```

TIP

Swap is disabled automatically in the installation with the git repository scripts in nodes where pods run as a best practice

2.2. RED HAT VIRTUALIZATION LOAD BALANCER DETAILS

Phase 1 provisions one virtual machine designated as a `load balancer` which is supplied to the **OpenShift Container Platform** installer for use as an HA Proxy node.

The `load balancer` uses the public subnets and maps to infrastructure nodes. The infrastructure nodes run the router pod which then directs traffic directly from the outside world into pods when external routes are defined.

To avoid reconfiguring DNS every time a new route is created, an external wild-card A `DNS` entry record must be configured pointing to the `load balancer` IP.

For example, to create a wild-card DNS entry for `apps.example.com` that has a low (five minute) time-to-live value (TTL) and points to the public IP address of the `load balancer`:

```
*.apps.example.com. 300 IN A 192.168.155.111
```

2.3. SOFTWARE VERSION DETAILS

The following tables provide the installed software versions for the different servers comprising the **Red Hat OpenShift Container Platform** reference architecture.

Table 2.3. RHEL OSEv3 Details

Software	Version
Red Hat Enterprise Linux 7.4 x86_64	kernel-3.10.0-693
Atomic-OpenShift{master/clients/node/sdn-ovs/utils}	3.6
Docker	1.12.x
Ansible	2.3.2

2.4. REQUIRED CHANNELS

A subscription to the following channels is required in order to deploy this reference architecture environment.

Table 2.4. Required Channels - OSEv3 Nodes

Channel	Repository Name
Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rpms
Red Hat OpenShift Enterprise 3.6 (RPMs)	rhel-7-server-ose-3.6-rpms
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	rhel-7-server-extras-rpms

Channel	Repository Name
Red Hat Enterprise Linux 7 Server - Fast Datapath (RPMs)	rhel-7-fast-datapath-rpms

Supported methods for activating a subscription include providing either a URL to a local **Red Hat Satellite**, or credentials to a valid **Red Hat Customer Portal** account. Each virtual machine in the **Red Hat OpenShift Container Platform** reference architecture is subscribed to a subscription pool by way of either a `pool id` (in the case of **Customer Portal** accounts), or an `Activation Key` (in the case of a **Red Hat Satellite**) that must be configured prior to Phase 2.

NOTE

The `pool id` can be obtained in the Subscriptions section of the **Red Hat Customer Portal**. Select the appropriate subscription to open a detailed view of the subscription, including the Pool ID, and its included channels.

NOTE

If using an `activation key` with a **Red Hat Satellite**, the `activation key` should be configured within the **Red Hat Satellite** to force subscription to the `Red Hat Satellite Tools 6.2 (for RHEL 7 Server) (RPMs)` channel with repository name, `rhel-7-server-satellite-tools-6.2-rpms`.

2.5. PREREQUISITES

This section describes the environment and setup needed to execute the Ansible playbooks and perform pre and post installation tasks. All actions in this section should be performed on a workstation or bastion host (hereafter referred to as `workstation host`)

2.5.1. Workstation Host

The `workstation host` provides the platform from which all phases of this reference architecture environment are deployed. This host may be a virtual machine or a stand-alone workstation. The most important requirements of the `workstation host` are that it runs **Red Hat Enterprise Linux 7**, and has **SSH** and **HTTPS** access to the **Red Hat Virtualization** environment.

2.5.2. Deploy Host Playbook

A playbook, `deploy-host.yaml`, is provided in the `openshift-ansible-contrib` **GitHub** repository to automate the setup of the `workstation-host`. The `deploy-host.yaml` playbook requires the ability to become the `root` user on the `workstation-host`. If a manual approach is required, the following sections outline the tasks it performs on the `workstation-host`.

To use the `deploy-host.yaml` playbook, first ensure the `workstation-host` is registered to an appropriate subscription which includes the repositories outlined in the next section. The `git` package is required to clone the `openshift-ansible-contrib` repository, `ansible` is required to run the playbook. These are found in the `rhel-7-server-rpms` and `rhel-7-server-extras-rpms` repositories respectively. To install these dependencies, run the following commands as root on the `workstation-host`:

```
[root@ws ~]# subscription-manager repos --enable rhel-7-server-rpms
[root@ws ~]# subscription-manager repos --enable rhel-7-server-extras-rpms
[root@ws ~]# yum -y install git ansible
```

Clone the **`openshift-ansible-contrib`** **GitHub** Repository as a regular user:

```
[user@ws ~]$ mkdir ~/git
[user@ws ~]$ cd ~/git
[user@ws git]$ git clone https://github.com/openshift/openshift-ansible-
contrib.git
Cloning into 'openshift-ansible-contrib'...

... [OUTPUT ABBREVIATED] ...
```

Run the **`deploy-host.yaml`** playbook, specifying **`rhv`** as the provider:

```
[user@ws git]$ cd openshift-ansible-contrib
[user@ws openshift-ansible-contrib]$ ansible-playbook playbooks/deploy-
host.yaml -e provider=rhv
```

The **`deploy-host.yaml`** playbook performs the following tasks:

- Subscribe to repositories
- Install prerequisite RPMs
- Clone the **`ovirt-ansible`** **GitHub** repository
- Generate SSH Keys

2.5.3. Subscriptions and RPMs

Install (if not already installed) the **Extra Packages for Enterprise Linux (EPEL)** repository and ensure it is disabled for normal installs:

```
[root@ws ~]# yum -y install https://dl.fedoraproject.org/pub/epel/epel-
release-latest-7.noarch.rpm
[root@ws ~]# yum-config-manager --disable epel
[root@ws ~]# yum-config-manager --disable epel-testing
```

Enable the following repositories and install the following packages to support communication with the **Red Hat Virtualization** API and installation of **OpenShift Container Platform**.

```
[root@ws ~]# subscription-manager repos --enable rhel-7-server-optional-
rpms \
              --enable rhel-7-server-ose-3.6-rpms \
              --enable rhel-7-fast-datapath-rpms \
              --enable rhel-7-server-rhv-4-mgmt-agent-rpms
[root@ws ~]# yum -y install atomic-openshift-utils python-ovirt-engine-
sdk4 pyOpenSSL
```

Install the following packages, enabling the **EPEL** repository:

```
[root@ws ~]# yum -y --enablerepo epel install python2-jmespath python-httplib2
```

2.5.4. GitHub Repositories

openshift-ansible-contrib

The code in the **openshift-ansible-contrib** repository referenced above handles the installation of **OpenShift Container Platform** and the accompanying infrastructure. The **openshift-ansible-contrib** repository is not explicitly supported by **Red Hat** but the Reference Architecture team performs testing to ensure the code operates as defined.

https://github.com/openshift/openshift-ansible-contrib

ovirt-ansible

The Ansible roles in the oVirt project's **ovirt-ansible** repository handle provisioning of **Red Hat Virtualization** elements such as networks, storage, and virtual machines. This reference architecture makes use of the virtual machine creation and destruction capabilities of the **ovirt-ansible** roles.

https://github.com/ovirt/ovirt-ansible

The playbooks in the **openshift-ansible-contrib** expect the **ovirt-ansible** repository to be cloned in the same directory as the **openshift-ansible-contrib** repository. It is recommended to create a common directory under a non-privileged user's home directory on the **workstation host** for these repositories, e.g. **~/git**.

```
[user@ws ~]$ cd ~/git
[user@ws git]$ git clone https://github.com/ovirt/ovirt-ansible.git
Cloning into 'ovirt-ansible'...

... [OUTPUT ABBREVIATED] ...

[user@ws git]$ ls
openshift-ansible-contrib  ovirt-ansible
```

2.5.5. SSH Public and Private Key

Ansible works by communicating with target servers via the **Secure Shell (SSH)** protocol. **SSH** keys are used in place of passwords in the **OpenShift Container Platform** installation process. These keys are generated on the **workstation host**. They must be provided during Phase 1 of the installation to ensure access to the virtual machines that are part of this reference architecture.

To avoid being asked repeatedly for the pass phrase, either employ an **ssh-agent** on the **workstation host**, or apply a blank pass phrase when creating keys.

The Ansible script from Phase 1 that installs the virtual machine hosts adds the public key to the **/root/.ssh/authorized_keys** file, and adds the private key **/root/.ssh/id_rsa** file on all the hosts to allow **SSH** communication within the environment (i.e.- from the **workstation host** to **openshift-master-0** without passwords).

SSH Key Generation

If **SSH** keys do not currently exist then it is required to create them. Generate an RSA key pair with a blank pass phrase by typing the following at a shell prompt:

```
[user@ws ~]$ ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa
```

A message similar to the following prints to indicate the keys are created successfully.

```
Your identification has been saved in ~/.ssh/id_rsa.
Your public key has been saved in ~/.ssh/id_rsa.pub.
The key fingerprint is:
e7:97:c7:e2:0e:f9:0e:fc:c4:d7:cb:e5:31:11:92:14 USER@ws.example.com
The key's randomart image is:
+--[ RSA 2048]----+
|             E.  |
|            . .  |
|             o . |
|              . .|
|         S .    . |
|          + o o ..|
|           * * +oo|
|            O +..=|
|            o*  o.|
+-----------------+
```

Adding SSH Key to Ansible

Once the **SSH** key pair has been generated, add the public key **`id_rsa.pub`** to the **`root_ssh_key`** variable in the **`ocp-vars.yaml`** variable file under **`openshift-ansible-contrib/reference-architecture/rhv-ansible`**

```
root_ssh_key: ssh-rsa AAAAB... [OMITTED] ... user@ws.example.com
```

2.5.6. RHEL 7 QCOW2 KVM Image

The **`ovirt-ansible role`**, **`ovirt-image-template`** requires a URL to download a QCOW2 KVM image to use as the basis for the VMs on which OpenShift will be installed. To download a suitable **Red Hat Enterprise Linux** image, log in at https://access.redhat.com/, navigate to Downloads, then Red Hat Enterprise Linux, and select the latest release (7.4 at the time of this writing), and copy the URL for "KVM Guest Image". It is preferable to download the image to a local server, e.g. the /pub/ directory of a satellite if available, and provide a URL to the Ansible playbook, because the download link will expire after a short while and need to be refreshed. For example:

```
qcow_url: http://satellite.example.com/pub/rhel-guest-image-
7.4.x86_64.qcow2
```

2.6. DYNAMIC INVENTORY

Ansible relies on inventory files and variables to perform playbook runs. Because this environment deals with virtual machines created on the fly within a **Red Hat Virtualization** cluster, its inventory must be created automatically. To that end, a Python script taken from the Ansible project is included in the **GitHub** repository for this reference architecture.

This dynamic inventory script queries the **Red Hat Virtualization** API to display information about its

VMs. The script's placement within the **`inventory`** subdirectory of the **`rhv-ansible`** reference architecture directory causes it to be automatically called by any Ansible script executed from the **`rhv-ansible`** directory. The script may be manually executed to provide information about the environment or to help in debugging its parameters.

The virtual machine image used by this reference architecture is a **Red Hat Enterprise Linux KVM Guest Image** provided on the **Red Hat Customer Portal** as a QCOW2 formatted disk image. The KVM guest image comes with the **`ovirt-guest-agent`** service installed and set to run at boot. This agent provides details about the guest operating system to the **Red Hat Virtualization** engine. For the purposes of this reference architecture, the most important detail provided is the IP address of any active network interfaces. Without this information, the dynamic inventory script will not be able to provide Ansible with a way to reach the VMs.

In order to collect information about the **Red Hat Virtualization** environment, the dynamic inventory script must be supplied with credentials that allow it to collect read-only statistics on all virtual machines relevant to the installation. Once the **`openshift-ansible-contrib`** repository has been cloned on the **`workstation host`**, the credentials must be entered into the **`ovirt.ini`** file under **`openshift-ansible-contrib/reference-architecture/rhv-ansible/inventory`**. An example file is included in the same directory called **`ovirt.ini.example`**. Rename it to **`ovirt.ini`** and add the credentials:

```
[ovirt]
ovirt_url = https://engine.example.com/ovirt-engine/api
ovirt_username = my_ro_user@internal
ovirt_password = mypassword
ovirt_ca_file = ca.pem
```

Note the **`ovirt_ca_file`** entry. This refers to the certificate authority (CA) for the **Red Hat Virtualization** engine's HTTPS connections. By default, this file is a self-signed SSL certificate on the engine VM (self-hosted) or host. Both the dynamic inventory script and the Python libraries used by the oVirt Ansible playbooks require a local copy of this CA certificate. Download the CA certificate to the **`openshift-ansible-contrib/reference-architecture/rhv-ansible/`** directory and test the setup, using the following commands, substituting **`engine.example.com`** with the proper value:

```
[user@ws rhv-ansible]$ curl -k 'https://engine.example.com/ovirt-engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA' > ca.pem
[user@ws rhv-ansible]$ inventory/ovirt4.py --list
```

NOTE

The paths used in the **`ovirt.ini`** file and in the **`ocp-vars.yaml`** file below refer to the same CA certificate. The paths differ due to a difference in how Python and Ansible handle relative paths to their current working directories.

This command returns a JSON description of virtual machines found on the cluster.

For the **OpenShift Container Platform** installation, the Python inventory script groups VMs by their role for use in later playbooks. This grouping is possible because tags are applied to each VM at creation time during Phase 1. The masters are assigned the **`openshift-master`** tag, the infrastructure nodes are assigned the **`openshift-infra`** tag, and the application nodes are assigned the **`openshift-app`** tag.

TIP

While the parameters are fresh at hand, it would be a good idea to also enter them into the **`ocp-vars.yaml`** file under the **`openshift-ansible-contrib/reference-architecture/rhv-ansible`** directory for use by the **`ovirt-ansible`** roles.

NOTE

There are two differences between the engine settings in the **`ovirt.ini`** and the **`ocp-vars.yaml`** file. First, the dynamic inventory script can make use of a non-administrative account with read-only access, but the **Red Hat Virtualization** installation requires the ability to create virtual machines. Second, the path to the ca file includes `../` here because playbooks expect to find files relative to the directory from which they are called.

2.7. INSTALLATION VARIABLES

The file **`ocp-vars.yaml`** in the **`openshift-ansible-contrib/reference-architecture/rhv-ansible`** directory holds all the user-configurable parts of this reference architecture. Naturally, the reader is free to edit any of the files in the repository as they see fit, but an attempt has been made to ensure the most important variables are all in the **`ocp-vars.yaml`** file.

The first part contains the **Red Hat Virtualization Engine admin** credentials. The user provided here needs to be able to create virtual machines, storage, and affinity groups, as well as to destroy virtual machines.

ocp-vars.yaml

```
---
### Red Hat Virtualization Engine Connection
#engine_url: https://engine.example.com/ovirt-engine/api
engine_url:
engine_user: admin@internal
engine_password:
# CA file copied from engine:/etc/pki/ovirt-engine/ca.pem; path is
relative to playbook directory
engine_cafile: ../ca.pem
```

This section defines the URL for downloading a QCOW2 formatted KVM image of a **Red Hat Enterprise Linux 7** guest. As noted in section 2.5.5, the **Red Hat Customer Portal** download links expire after a while, so it would be better to download and re-share the image from a local http server like a **Red Hat Satellite**.

```
### Red Hat Virtualization VM Image
## For CentOS 7:
#qcow_url: https://cloud.centos.org/centos/7/images/CentOS-7-x86_64-
GenericCloud.qcow2
## For RHEL: Find KVM Guest Image in Downloads -> RHEL on
https://access.redhat.com/ and use before the link expires:
#qcow_url:https://access.cdn.redhat.com//content/origin/files/<omitted>/rh
el-server-7.4-x86_64-kvm.qcow2?_auth_=<omitted>
## Alternatively, download the above KVM image, and re-host it on a local
satellite:
```

```
#qcow_url: https://satellite.example.com/pub/rhel-server-7.4-x86_64-
kvm.qcow2
qcow_url:
```

`Default` is the default name for a **Red Hat Virtualization** cluster set up through the hosted engine setup. If the local cluster name differs, or if it is desired to set up the **OpenShift Container Platform** installation in a separate cluster from other machines in a shared cluster, set the name here.

The second parameter in the section below sets which **Red Hat Virtualization** storage domain is used to create disk images for the VMs in this reference architecture. This will likely vary from installation to installation and the proper value should be suggested by the **Red Hat Virtualization** admin if not known. If multiple storage domains are desired, edit the VM definitions in the **`playbooks/vars/ocp-infra-vars.yaml`** file.

```
# Name of cluster to install on
rhv_cluster: Default

# Name of RHV storage domain to create disks
rhv_data_storage: vmstore
```

Set the credentials for either **Red Hat Customer Portal** or **Red Hat Satellite** here.

```
### Red Hat Content Subscriptions
## For subscriptions to Satellite:
rhsm_satellite: satellite.example.com
rhsm_activation_key: vm-key
rhsm_org_id: Default_Organization
rhsm_pool: none
rhsm_katello_url: http://satellite.example.com/pub/katello-ca-consumer-
latest.noarch.rpm

## For subscriptions to Red Hat's CDN
## Userid/Password could be moved to a vault file and encrypted for
safety, see the following link for details:
## http://docs.ansible.com/ansible/playbooks_vault.html
#rhsm_pool: OpenShift Enterprise, Premium*
#rhsm_user:
#rhsm_password:
```

The public **SSH** key should be entered here as outlined in section 2.5.4.

```
### PUBLIC SSH key here for access to all nodes.
## Use ssh-agent or a passwordless key in ~/.ssh/id_rsa for the PRIVATE
key.
root_ssh_key:
```

The first stanza of variables here set the deployment type and the expected console port (8443 by default) for the **OpenShift Container Platform** master console.

The second stanza defines host names for the deployment. Replace **`example.com`** with the local domain, and ensure the values defined make sense for the local installation.

```
### Openshift variables
## Choices of deployment type: openshift-enterprise, origin
```

```
deployment_type: openshift-enterprise
openshift_vers: v3_6
containerized: false
console_port: 8443

### DNS entries
## Wildcard *.{{app_dns_prefix}}.{{public_hosted_zone}} must point to IP
of LB
public_hosted_zone: example.com
app_dns_prefix: apps
load_balancer_hostname: openshift-lb.{{public_hosted_zone}}
openshift_master_cluster_hostname: {{load_balancer_hostname}}
openshift_master_cluster_public_hostname: openshift.{{public_hosted_zone}}
```

The following section deals with authentication into the **OpenShift Container Platform** master console. The example here uses **htpasswd**, but a number of options are available as outlined in the OpenShift Documentation.

```
### OpenShift Identity Providers
# htpasswd shown here, other options documented at
# https://docs.openshift.com/container-
platform/3.5/install_config/configuring_authentication.html
openshift_master_identity_providers:
  - name: htpasswd_auth
    login: true
    challenge: true
    kind: HTPasswdPasswordIdentityProvider
    filename: /etc/origin/master/htpasswd
# Defining htpasswd users
#openshift_master_htpasswd_users:
#  - user1: <pre-hashed password>
#  - user2: <pre-hashed password>
# Use 'htpasswd -n <user>' to generate password hash. (htpasswd from
httpd-tools RPM)
# Example with admin:changeme
openshift_master_htpasswd_users: {'admin':
'$apr1$zAhyA9Ko$rBxBOwAwwtRuuaw8OtCwH0'}
# or
#openshift_master_htpasswd_file=<path to local pre-generated htpasswd
file>
```

Persistent storage for the OpenShift Registry must be configured here. This example uses an **NFS** server.

```
### Registry storage
## NFS for registry storage
openshift_hosted_registry_storage_kind: nfs
openshift_hosted_registry_selector: region=infra
openshift_hosted_registry_storage_host: 192.168.155.10
openshift_hosted_registry_storage_nfs_directory: /var/lib/exports
openshift_hosted_registry_storage_volume_name: registryvol
```

CHAPTER 3. DEPLOY OPENSHIFT

This chapter focuses on Phases 1 and 2 of the deployment process. The prerequisites defined above in chapter 2 are required before running the commands in this chapter. The following checklist will help verify those prerequisites.

3.1. CHECKLIST

- Workstation Host with access to **Red Hat Virtualization** cluster.
  - RPMs installed for git, ansible, and python-ovirt-engine-sdk4 as outlined in section 2.5.2.
  - **GitHub** repositories checked out as outlined in section 2.5.3.
  - Configuration file in **`openshift-ansible-contrib/reference-architecture/rhv-ansible/inventory`** for dynamic inventory access to **Red Hat Virtualization Engine**
  - Configuration file **`openshift-ansible-contrib/reference-architecture/rhv-ansible/ocp-vars.yaml`** populated with:
    - Site specific host names.
    - **Red Hat Virtualization Engine** Credentials.
    - qcow2 image url for RHEL KVM image.
    - RHSM subscription info, either through **Red Hat Customer Portal** or a local **Red Hat Satellite**.
    - NFS server information for hosted registry storage.
- **Red Hat Virtualization** self-hosted cluster with access to storage domain.
- **DHCP** server set to provide pool of IP addresses to **Red Hat Virtualization** cluster network OR static set of predefined MAC addresses with static IP entries (specified in **`ovirt-vm-infra.yaml`** vars file under **`reference-architectures/rhv-ansible/playbooks/vars`** in the **`openshift-ansible-contrib`** repository.)
- DNS setup with:
  - Wild-card entry for applications sub domain, e.g. **`*.apps.example.com IN A 192.168.100.101`**.
  - Ability to add entries after the **Red Hat Virtualization Engine** assigns virtual machines MAC addresses from MAC address pool OR pre-defined host name entries corresponding to IPs assigned statically by DHCP server.

3.2. PROVISION AND DEPLOY THE INFRASTRUCTURE

This section focuses on Phase 1 of the deployment process, creating a virtual machine environment within an existing **Red Hat Virtualization** cluster for the deployment of **OpenShift Container Platform.** Run the Ansible playbook included in the **`openshift-ansible-contrib`** repository by changing directory to **`~/git/openshift-ansible-contrib/reference-architecture/rhv-ansible/`**. This step is performed on the **`workstation host`**.

```
[user@ws ~]$ cd ~/git/openshift-ansible-contrib/reference-
architecture/rhv-ansible
[user@ws rhv-ansible]$ ansible-playbook -e@ocp-vars.yaml playbooks/ovirt-
vm-infra.yaml
```

3.3. INSTALL OPENSHIFT

Once the virtual machines have been provisioned, DNS will need to be configured to properly resolve the IP addresses assigned dynamically to the VMs. To assist in creating the DNS entries, run the **`output-dns.yaml`** Ansible playbook included in the same directory as the **`ovirt-vm-infra.yaml`** playbook, and with the same variable file, **`ocp-vars.yaml`**.

```
[user@ws rhv-ansible]$ ansible-playbook -e@ocp-vars.yaml playbooks/output-
dns.yaml

... [OUTPUT ABBREVIATED] ...

PLAY RECAP ***************************************************************
localhost          : ok=10   changed=8    unreachable=0    failed=0
```

This playbook creates two files, **`inventory.hosts`** with **`/etc/hosts`** style entries, and **`inventory.nsupdate`** with entries that can be used with the **`nsupdate`** dynamic DNS utility. In this reference architecture environment, the **Red Hat Satellite** **`satellite.example.com`** provides DNS, and must be updated manually with these entries:

```
[root@satellite ~]# nsupdate -k /etc/rndc.key -v
> server satellite.example.com
> update add 187.155.168.192.in-addr.arpa 86400 PTR  openshift-master-
0.example.com
> update add 188.155.168.192.in-addr.arpa 86400 PTR  openshift-master-
1.example.com
> update add 213.155.168.192.in-addr.arpa 86400 PTR  openshift-master-
2.example.com
> update add 189.155.168.192.in-addr.arpa 86400 PTR  openshift-infra-
0.example.com
> update add 186.155.168.192.in-addr.arpa 86400 PTR  openshift-infra-
1.example.com
> update add 184.155.168.192.in-addr.arpa 86400 PTR  openshift-node-
0.example.com
> update add 185.155.168.192.in-addr.arpa 86400 PTR  openshift-node-
1.example.com
> update add 206.155.168.192.in-addr.arpa 86400 PTR  openshift-
lb.example.com
> send
> zone example.com
> update add *.apps.example.com 86400 A 192.168.155.206
> update add openshift.example.com 86400 A 192.168.155.206
> update add openshift-master-0.example.com 86400 A 192.168.155.187
> update add openshift-master-1.example.com 86400 A 192.168.155.188
> update add openshift-master-2.example.com 86400 A 192.168.155.213
> update add openshift-infra-0.example.com 86400 A 192.168.155.189
> update add openshift-infra-1.example.com 86400 A 192.168.155.186
> update add openshift-node-0.example.com 86400 A 192.168.155.184
> update add openshift-node-1.example.com 86400 A 192.168.155.185
```

```
> update add openshift-lb.example.com 86400 A 192.168.155.206
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags:; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; ZONE SECTION:
;example.com.                   IN      SOA

;; UPDATE SECTION:
*.apps.example.com.     86400   IN      A       192.168.155.206
openshift.example.com.  86400   IN      A       192.168.155.206
openshift-master-0.example.com. 86400 IN A      192.168.155.187
openshift-master-1.example.com. 86400 IN A      192.168.155.188
openshift-master-2.example.com. 86400 IN A      192.168.155.213
openshift-infra-0.example.com. 86400 IN A       192.168.155.189
openshift-infra-1.example.com. 86400 IN A       192.168.155.186
openshift-node-0.example.com. 86400 IN  A       192.168.155.184
openshift-node-1.example.com. 86400 IN  A       192.168.155.185
openshift-lb.example.com. 86400 IN      A       192.168.155.206

> send
```

The first line, **`server satellite.example.com`** specifies which server to send updates to. After that, the reverse DNS entries are pasted in, then processed with the **send** command. Finally, the forward entries are pasted in and processed with the **send** command.

Now run the actual OpenShift installation playbook, from the same directory as the **`ovirt-vm-infra.yaml`** playbook, and with the same variable file, **`ocp-vars.yaml`**.

```
[user@ws ~]$ cd ~/git/openshift-ansible-contrib/reference-
architecture/rhv-ansible
[user@ws rhv-ansible]$ ansible-playbook -e@ocp-vars.yaml
playbooks/openshift-install.yaml

... [OUTPUT ABBREVIATED] ...

PLAY RECAP *************************************************************
localhost            : ok=16   changed=5    unreachable=0    failed=0
openshift-infra-0    : ok=221  changed=77   unreachable=0    failed=0
openshift-infra-1    : ok=221  changed=77   unreachable=0    failed=0
openshift-lb         : ok=238  changed=78   unreachable=0    failed=0
openshift-master-0 : ok=626  changed=204  unreachable=0    failed=0
openshift-master-1 : ok=414  changed=145  unreachable=0    failed=0
openshift-master-2 : ok=414  changed=145  unreachable=0    failed=0
openshift-node-0     : ok=221  changed=77   unreachable=0    failed=0
openshift-node-1     : ok=221  changed=77   unreachable=0    failed=0
```

Once the playbook finishes successfully, **OpenShift Container Platform** should be up and running on a **Red Hat Virtualization** cluster. Visit the administrative console at the host name set for the **Load Balancer** host, using https:// and appending port :8443, e.g. **`https://openshift.example.com:8443`**

3.3.1. Registry Console Selector (Optional)

The OpenShift Registry Console deployment is deployed on any Red Hat OpenShift Container Platform node by default, so the container may end up running on any of the application nodes.

From the first *master* node(openshift-master-0.example.com), ensure the OpenShift Registry Console pod runs on the *infra* nodes by modifying the *nodeSelector* as follows:

```
$ oc patch dc registry-console \
    -n default \
    -p '{"spec":{"template":{"spec":{"nodeSelector":{"role":"infra"}}}}}'
```

NOTE

There is a bugzilla ID: 1425022 being investigated by Red Hat at the time of writing this paper to fix this issue.

CHAPTER 4. OPERATIONAL MANAGEMENT

With the successful deployment of **Red Hat OpenShift Container Platform** , the following section demonstrates how to confirm proper functionality of the **OpenShift Container Platform**.

4.1. VALIDATE THE DEPLOYMENT

An Ansible script in the git repository deploys an application to test the functionality of the master and application nodes, registry, and router. This playbook tests the deployment and cleans up projects and pods created during the validation run.

The playbook performs the following steps:

Environment Validation

- Validate the public OpenShift **LB** address from the installation system.
- Validate the public OpenShift **LB** address from the master nodes.
- Validate the internal OpenShift **LB** address from the master nodes.
- Validate the master local master address.
- Validate the health of the **ETCD** cluster to ensure all **ETCD** nodes are healthy.
- Create a project in OpenShift called **`validate`**.
- Create an OpenShift Application.
- Add a route for the Application.
- Validate the URL returns a status code of 200 or healthy.
- Delete the validation project.

NOTE

Ensure the **`ocp-vars.yaml`** file matches that used during deployment.

```
[user@ws ~]$ cd ~/git/openshift-ansible-contrib/reference-
architecture/rhv-ansible
[user@ws rhv-ansible]$ ansible-playbook -e@ocp-vars.yaml
playbooks/openshift-validate.yaml

... [OUTPUT ABBREVIATED] ...

PLAY RECAP ************************************************************
localhost              : ok=11   changed=5    unreachable=0    failed=0
openshift-master-0     : ok=15   changed=6    unreachable=0    failed=0
openshift-master-1     : ok=7    changed=1    unreachable=0    failed=0
openshift-master-2     : ok=7    changed=1    unreachable=0    failed=0
```

4.2. GATHER HOST NAMES

With all of the steps that occur during the installation of **OpenShift Container Platform**, it is possible to lose track of the names of the nodes in the recently deployed environment. One option to get these host names is to manually run the dynamic inventory script detailed under Section 2.6, "Dynamic Inventory".

To help facilitate the **Operational Management** Chapter, the following host names are used.

- openshift-master-0.example.com
- openshift-master-1.example.com
- openshift-master-2.example.com
- openshift-infra-0.example.com
- openshift-infra-1.example.com
- openshift-app-0.example.com
- openshift-app-1.example.com

4.3. CHECK THE HEALTH OF ETCD

This section focuses on the **ETCD** cluster. It describes the different commands used to ensure the cluster is healthy. The internal **DNS** names of the nodes running **ETCD** must be used.

Perform the following steps from the first master node.

To run diagnostics, **SSH** into the first master node (**openshift-master-0.example.com**) as the **root** user.

```
[user@ws rhv-ansible]$ ssh root@openshift-master-0.example.com
```

```
[root@openshift-master-0 ~]# etcdctl2 cluster-health
member 41f00fdab9af3934 is healthy: got healthy result from
https://192.168.155.208:2379
member 513a3423c4bf3997 is healthy: got healthy result from
https://192.168.155.207:2379
member 981314c8bba47513 is healthy: got healthy result from
https://192.168.155.206:2379
cluster is healthy
```

NOTE

In this configuration the **ETCD** services are distributed among the **OpenShift Container Platform master** nodes.

4.4. DEFAULT NODE SELECTOR

As explained in section 2.6, node labels are an important part of the **OpenShift Container Platform** environment. In addition to Ansible groups, each node is assigned a role as an **openshift_node_label** that the **OpenShift Container Platform** installer will use to categorize the

nodes. In this reference architecture installation, the default node selector is set to "role=apps" in **/etc/origin/master/master-config.yaml** on all of the master nodes. This configuration parameter is set by the OpenShift installation playbooks on all masters.

Verify from the first master node that the **defaultNodeSelector** is defined.

```
[root@openshift-master-0 ~]# grep -A3 projectConfig
/etc/origin/master/master-config.yaml
projectConfig:
  defaultNodeSelector: "role=app"
  projectRequestMessage: ""
  projectRequestTemplate: ""
```

NOTE

For any changes to the master configuration to take effect, the master API service must be restarted on all masters.

4.5. MANAGEMENT OF MAXIMUM POD SIZE

Quotas are set on ephemeral volumes within pods to prevent a pod from growing too large and impacting the node. When persistent volume claims are not set a pod has the ability to grow as large as the underlying file system will allow. The required modifications are applied using a combination of user-data and Ansible.

OpenShift Volume Quota

At launch time, user-data creates a XFS partition on the **/dev/vdc** block device, adds an entry in **/etc/fstab**, and mounts the volume with the option of **gquota**. If **gquota** is not set the OpenShift node will not be able to start with the **perFSGroup** parameter defined below. Only **infrastructure** and **application** nodes are installed with local volume storage.

SSH into the first infrastructure node (**openshift-infra-0.example.com**) to verify the entry exists within **/etc/fstab**.

```
[root@openshift-infra-0 ~]# cat /etc/fstab

... [OUTPUT ABBREVIATED] ...

/dev/vdc /var/lib/origin/openshift.local.volumes xfs gquota 0 0
```

Docker Storage Setup

The docker-storage-setup file is created at launch time by user-data. This file tells the Docker service to use **/dev/vdb** and create the volume group of **docker-vol**. Docker storage setup is performed on all master, infrastructure, and application nodes.

SSH into the first infrastructure node, (**openshift-infra-0.example.com**) to verify **/etc/sysconfig/docker-storage-setup** matches the information below.

```
[root@openshift-infra-0 ~]# cat /etc/sysconfig/docker-storage-setup
DEVS=/dev/vdb
VG=docker-vol
DATA_SIZE=95%VG
```

```
STORAGE_DRIVER=overlay2
CONTAINER_ROOT_LV_NAME=dockerlv
CONTAINER_ROOT_LV_MOUNT_PATH=/var/lib/docker
CONTAINER_ROOT_LV_SIZE=100%FREE
```

4.6. YUM REPOSITORIES

Section 2.4, Required Channels, defines the repositories required for a successful OpenShift installation. All systems should have the same subscriptions. To verify subscriptions match those defined in Required Channels, perform the following. The repositories below are enabled during the **rhsm-repos** playbook during the installation. The installation will be unsuccessful if the repositories are missing from the system.

```
[root@openshift-infra-0 ~]# yum repolist
Loaded plugins: package_upload, search-disabled-repos
repo id                                              repo name
status
!rhel-7-fast-datapath-rpms/7Server/x86_64 Red Hat Enterprise Linux Fast
Datapath (RHEL 7 Serve     38
!rhel-7-server-extras-rpms/x86_64         Red Hat Enterprise Linux 7
Server - Extras (RPMs)    653+24
!rhel-7-server-ose-3.6-rpms/x86_64        Red Hat OpenShift Container
Platform 3.6 (RPMs)      512+20
!rhel-7-server-rpms/7Server/x86_64        Red Hat Enterprise Linux 7
Server (RPMs)             17,461
repolist: 15,682
```

4.7. CONSOLE ACCESS

This section will cover logging into the **OpenShift Container Platform** management console via the GUI and the CLI. After logging in via one of these methods applications can then be deployed and managed.

4.7.1. Log into GUI console and deploy an application

Perform the following steps from the **workstation host**.

Open a browser and access the host name for your master console, e.g. https://openshift.example.com:8443/. Log in using the account created during the install. (See the **openshift_master_htpasswd_users** entry in **ocp-vars.yaml** if the default was left in.)

To deploy an application, click on the **New Project** button. Provide a **Name** and click **Create**. Next, deploy the **jenkins-ephemeral** instant app by either searching for it in the search bar, or by clicking the **Continuous Integration** box. Accept the defaults and click **Create**. Instructions along with a URL will be provided for how to access the application on the next screen. Click **Continue to Overview** and bring up the management page for the application. Click on the link provided and access the application to confirm functionality.

4.7.2. Log into CLI and Deploy an Application

Perform the following steps from the **workstation host**.

Install the **oc** client by visiting the public URL of the OpenShift deployment. For example,

https://openshift.example.com:8443/console/command-line and click latest release. When directed to https://access.redhat.com, log in with the valid Red Hat customer credentials and download the client relevant to the current workstation. Follow the instructions located on the production documentation site for getting started with the CLI.

```
[user@ws ~]$ mkdir ~/bin
[user@ws ~]$ tar xf Downloads/oc-3.5*linux.tar.gz -C ~/bin
[user@ws ~]$ oc login https://openshift.example.com:8443
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server
could be intercepted by others.
Use insecure connections? (y/n): y

Authentication required for https://openshift.example.com:8443 (openshift)
Username: myuser
Password:
Login successful.
```

After the **oc** client is configured, create a new project and deploy an application.

```
[user@ws ~]$ oc new-project test-app
Now using project "test-app" on server
"https://openshift.example.com:8443".

You can add applications to this project with the 'new-app' command. For
example, try:

    oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-
ex.git

    to build a new example application in Ruby.
```

```
[user@ws ~]$ oc new-app https://github.com/openshift/cakephp-ex.git --
name=php
--> Found image 3fb0dea (4 weeks old) in image stream "openshift/php"
under tag "7.0" for "php"

    Apache 2.4 with PHP 7.0
    -----------------------
    Platform for building and running PHP 7.0 applications

    Tags: builder, php, php70, rh-php70

    * The source repository appears to match: php
    * A source build using source code from
https://github.com/openshift/cakephp-ex.git will be created
      * The resulting image will be pushed to image stream "php:latest"
      * Use 'start-build' to trigger a new build
    * This image will be deployed in deployment config "php"
    * Port 8080/tcp will be load balanced by service "php"
      * Other containers can access this service through the hostname
"php"

--> Creating resources ...
    imagestream "php" created
```

```
    buildconfig "php" created
    deploymentconfig "php" created
    service "php" created
--> Success
    Build scheduled, use 'oc logs -f bc/php' to track its progress.
    Run 'oc status' to view your app.
[user@ws ~]$ oc expose service php
route "php" exposed
```

Display the status of the application.

```
[user@ws ~]$ oc status
In project test-app on server https://openshift.example.com:8443

http://php-test-app.apps.example.com to pod port 8080-tcp (svc/php)
  dc/php deploys istag/php:latest <-
    bc/php source builds https://github.com/openshift/cakephp-ex.git on
openshift/php:7.0
    deployment #1 deployed 2 minutes ago - 1 pod

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

Access the application by accessing the URL provided by **`oc status`**. The **`CakePHP`** application should be visible now.

4.8. EXPLORE THE ENVIRONMENT

4.8.1. List Nodes and Set Permissions

If you try to run the following command, it should fail.

```
[user@ws ~]$ oc get nodes
Error from server: User "myuser" cannot list all nodes in the cluster
```

The reason it is failing is because the permissions for that user are incorrect. Get the username and configure the permissions.

```
[user@ws ~]$ oc whoami
myuser
```

Once the username has been established, log back into a master node and enable the appropriate permissions for your user. Perform the following step from the first master (**`openshift-master-0.example.com`**).

```
[root@openshift-master-0 ~]# oc adm policy add-cluster-role-to-user
cluster-admin myuser
cluster role "cluster-admin" added: "myuser"
```

Attempt to list the nodes again and show the labels.

```
[user@ws ~]$ oc get nodes
NAME                                        STATUS                            AGE
```

```
openshift-infra-0.example.com    Ready                         1d
openshift-infra-1.example.com    Ready                         1d
openshift-lb.example.com         Ready                         1d
openshift-master-0.example.com   Ready,SchedulingDisabled   1d
openshift-master-1.example.com   Ready,SchedulingDisabled   1d
openshift-master-2.example.com   Ready,SchedulingDisabled   1d
openshift-node-0.example.com     Ready                         1d
openshift-node-1.example.com     Ready                         1d
```

(The VERSION column is omitted here for readability)

4.8.2. List Router and Registry

List the router and registry by changing to the **default** project.

> **NOTE**
>
> If the OpenShift account configured on the workstation has cluster-admin privileges perform the following. If the account does not have this privilege **SSH** to one of the OpenShift masters and perform the steps.

```
[user@ws ~]$  oc project default
Now using project "default" on server
"https://openshift.example.com:8443".
[user@ws ~]$  oc get all
NAME                   DOCKER REPO
TAGS      UPDATED
is/registry-console   172.30.12.144:5000/default/registry-console   3.5
28 hours ago

NAME                   REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/docker-registry    1          1         1         config
dc/registry-console   1          1         1         config
dc/router             1          2         2         config

NAME                     DESIRED   CURRENT   READY     AGE
rc/docker-registry-1    1         1         1         1d
rc/registry-console-1   1         1         1         1d
rc/router-1             2         2         2         1d

NAME                          HOST/PORT
PATH      SERVICES           PORT      TERMINATION   WILDCARD
routes/docker-registry    docker-registry-default.apps.example.com
docker-registry    <all>     passthrough   None
routes/registry-console   registry-console-default.apps.example.com
registry-console   <all>     passthrough   None

NAME                    CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
svc/docker-registry    172.30.12.144   <none>        5000/TCP
1d
svc/kubernetes         172.30.0.1      <none>
443/TCP,53/UDP,53/TCP     1d
svc/registry-console   172.30.66.63    <none>        9000/TCP
1d
```

```
svc/router              172.30.5.177    <none>
80/TCP,443/TCP,1936/TCP   1d

NAME                           READY     STATUS    RESTARTS   AGE
po/docker-registry-1-sw6ms     1/1       Running   0          1d
po/registry-console-1-vdw1k    1/1       Running   0          1d
po/router-1-kw8sd              1/1       Running   0          1d
po/router-1-zm9fh              1/1       Running   0          1d
```

Observe the output of **oc get all**

4.8.3. Explore the Registry

The OpenShift Ansible playbooks configure an NFS share for the docker registry.

```
[user@ws ~]$ oc describe svc/docker-registry
Name:           docker-registry
Namespace:      default
Labels:         <none>
Selector:       docker-registry=default
Type:           ClusterIP
IP:         172.30.12.144
Port:           5000-tcp    5000/TCP
Endpoints:      172.16.8.3:5000
Session Affinity:   ClientIP
No events.
```

Observe the IP (for reaching the registry) and the Endpoint (actual pod on which the registry runs).

The **oc** client allows similar functionality to the **docker** command. To find out more information about the registry storage perform the following.

```
[user@ws ~]$ oc get pods --selector=docker-registry
NAME                      READY     STATUS    RESTARTS   AGE
docker-registry-1-sw6ms   1/1       Running   0          1d
```

```
[user@ws ~]$ oc exec docker-registry-1-sw6ms cat /config.yml
version: 0.1
log:
  level: debug
http:
  addr: :5000
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /registry
  delete:
    enabled: true
auth:
  openshift:
    realm: openshift
    audit:
      enabled: false
```

```
    # tokenrealm is a base URL to use for the token-granting registry
endpoint.
    # If unspecified, the scheme and host for the token redirect are
determined from the incoming request.
    # If specified, a scheme and host must be chosen that all registry
clients can resolve and access:
    #
    # tokenrealm: https://example.com:5000
middleware:
  registry:
    - name: openshift
  repository:
    - name: openshift
      options:
        acceptschema2: false
        pullthrough: true
        mirrorpullthrough: true
        enforcequota: false
        projectcachettl: 1m
        blobrepositorycachettl: 10m
  storage:
    - name: openshift
```

4.8.4. Explore Docker Storage

This section explores the Docker storage on an infrastructure node.

The example below can be performed on any node in the cluster. For this example, the infrastructure node (**`openshift-infra-0.example.com`**) is used. The output below describing the **`Storage Driver: docker— vol-docker— pool`** demonstrates that docker storage is set to use an **`overlay2`** file system.

```
[root@openshift-infra-0 ~]# docker info
Containers: 2
 Running: 2
 Paused: 0
 Stopped: 0
Images: 3
Server Version: 1.12.6
Storage Driver: overlay2
 Backing Filesystem: xfs
Logging Driver: journald
Cgroup Driver: systemd
Plugins:
 Volume: local
 Network: null host bridge overlay
 Authorization: rhel-push-plugin
Swarm: inactive
Runtimes: docker-runc runc
Default Runtime: docker-runc
Security Options: seccomp selinux
Kernel Version: 3.10.0-693.el7.x86_64
Operating System: OpenShift Enterprise
OSType: linux
```

```
Architecture: x86_64
Number of Docker Hooks: 3
CPUs: 2
Total Memory: 7.636 GiB
Name: openshift-infra-0.example.com
ID: WLBN:TJCN:JQTZ:2HYP:QPDU:PJWD:VZI4:HD3D:CENT:7M2M:MIYY:XW3L
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://registry.access.redhat.com/v1/
WARNING: bridge-nf-call-iptables is disabled
WARNING: bridge-nf-call-ip6tables is disabled
Insecure Registries:
 127.0.0.0/8
Registries: registry.access.redhat.com (secure),
registry.access.redhat.com (secure), docker.io (secure)
```

Verify 3 disks are attached to the node. The disk **/dev/vda** is used for the OS, **/dev/vdb** is used for docker storage, and **/dev/vdc** is used for emptyDir storage for containers that do not use a persistent volume.

```
[root@openshift-infra-0 ~]# fdisk -l

Disk /dev/vda: 64.4 GB, 64424509440 bytes, 125829120 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000a73bb

   Device Boot      Start         End      Blocks   Id  System
/dev/vda1   *        2048   125829086    62913519+  83  Linux

Disk /dev/vdb: 107.4 GB, 107374182400 bytes, 209715200 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0003fdcb

   Device Boot      Start         End      Blocks   Id  System
/dev/vdb1            2048   209715199   104856576   8e  Linux LVM

Disk /dev/vdc: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/docker--vol-dockerlv: 107.4 GB, 107369988096 bytes,
209707008 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

4.9. TEST LOAD BALANCER

If console access (https://openshift.example.com:8443) is working, and HTTP and HTTPS access to created applications are working, then normal function of the load balancer has already been established. If either of these are not working, then troubleshooting should start with the HA Proxy service on the **openshift-lb** node itself. **SSH** as root to the **openshift-lb** node, and view the **/etc/haproxy/haproxy.cfg** file.

/etc/haproxy/haproxy.cfg

```
frontend  atomic-openshift-api
    bind *:8443
    default_backend atomic-openshift-api
    mode tcp
    option tcplog
frontend  apps-http
    bind *:80
    default_backend apps-http
frontend  apps-https
    bind *:443
    default_backend apps-http

backend atomic-openshift-api
    balance source
    mode tcp
    server      master0 192.168.155.176:8443 check
    server      master1 192.168.155.177:8443 check
    server      master2 192.168.155.178:8443 check
backend apps-http
    balance source
    server      infra0 192.168.155.182:80 check
    server      infra1 192.168.155.183:80 check
backend apps-https
    balance source
    server      infra0 192.168.155.182:443 check
    server      infra1 192.168.155.183:443 check
```

Ensure there are **frontend** entries for ports 8443, 80, and 443, and the **backends** include the master nodes for the 8443 **frontend**, and the application nodes for the 80 and 443 **backends**.

4.10. TESTING FAILURE

In this section, reactions to failure are explored. After a successful install and some of the smoke tests noted above have been completed, failure testing is executed.

4.10.1. Generate a Master Outage

NOTE

Perform the following steps from the **Red Hat Virtualization Engine Administration** Portal and the OpenShift public URL.

Log in to the **Red Hat Virtualization Engine Administration** Portal, click on the Virtual Machines tab, and select the **openshift-master-1** VM. Right-click the entry and select Power Off, then confirm the action. This will simulate a hard power-off with no warning to the operating system.

Ensure the console can still be accessed by opening a browser and accessing https://openshift.example.com:8443/. At this point, the cluster is in a degraded state because only 2/3 master nodes are running, but complete functionality remains.

4.10.2. Observe the Behavior of ETCD with a Failed Master Node

SSH into the first master node (**openshift-master-0.example.com**). Issue the **etcdctl2** command to confirm the cluster is healthy.

```
[user@ws ~]$ ssh root@openshift-master-0.example.com
```

```
[root@openshift-master-0 ~]# etcdctl2 cluster-health
member 880fa69fded1439 is healthy: got healthy result from
https://192.168.155.178:2379
failed to check the health of member 671af8bcf83da922 on
https://192.168.155.177:2379: Get https://192.168.155.177:2379/health:
dial tcp 192.168.155.177:2379: i/o timeout
member 671af8bcf83da922 is unreachable: [https://192.168.155.177:2379] are
all unreachable
member e91cf1059ca0c86f is healthy: got healthy result from
https://192.168.155.176:2379
cluster is healthy
```

Return the VM to service by right-clicking on the VM entry, selecting Run, and confirming the action.

4.10.3. Generate an Infrastructure Node outage

This section shows what to expect when an infrastructure node fails or is brought down intentionally.

4.10.3.1. Confirm Application Accessibility

NOTE

Perform the following steps from the browser on the **workstation host**.

Before bringing down an infrastructure node, check behavior and ensure things are working as expected. The goal of testing an infrastructure node outage is to see how the OpenShift routers and registries behave. Confirm the simple application deployed from before is still functional. If it is not, deploy a new version. Access the application to confirm connectivity. The next example provides **oc** commands to list projects, change to the project the application is deployed in, print the status of the application, and access the application via URL.

```
[user@ws ~]$ oc get projects
NAME                DISPLAY NAME   STATUS
default                            Active
kube-system                        Active
logging                            Active
management-infra                   Active
openshift                          Active
```

```
openshift-infra                    Active
test                               Active
test-app                           Active

[user@ws ~]$ oc project test-app1
Now using project "test-app" on server
"https://openshift.example.com:8443".
[user@ws ~]$ oc status
In project test-app on server https://openshift.example.com:8443

http://php-test-app.apps.example.com to pod port 8080-tcp (svc/php)
  dc/php deploys istag/php:latest <-
    bc/php source builds https://github.com/openshift/cakephp-ex.git on
openshift/php:7.0
    deployment #1 deployed 30 hours ago - 1 pod

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

Open a browser and ensure the application is still accessible.

4.10.3.2. Get Location of Router and Registry.

NOTE

Perform the following steps from the CLI of a local workstation.

Change to the default OpenShift project and check the router and registry pod locations.

```
[user@ws ~]$ oc project default
Now using project "default" on server
"https://openshift.example.com:8443".
[user@ws ~]$ oc get pods -o wide
NAME                         READY     STATUS    RESTARTS   AGE       IP
NODE
docker-registry-1-sw6ms      1/1       Running   0          2d
172.16.8.3         openshift-infra-1.example.com
registry-console-1-vdw1k     1/1       Running   0          2d
172.16.14.2        openshift-node-0.example.com
router-1-kw8sd               1/1       Running   0          2d
192.168.155.183    openshift-infra-1.example.com
router-1-zm9fh               1/1       Running   0          2d
192.168.155.182    openshift-infra-0.example.com
```

Note the registry is running on **`openshift-infra-1`**. This is the node that should be shut down to test fail-over.

4.10.3.3. Initiate the Failure and Confirm Functionality

NOTE

Perform the following steps from the **Red Hat Virtualization Engine Administration Portal** and a browser.

Log into the **Red Hat Virtualization Engine Administration Portal** , click on the Virtual Machines tab, and select the **openshift-infra-1** VM. Right click, and select Shut Down, then confirm the action. Wait a minute or two for the registry and pod to migrate over to **openshift-infra-0**. Check the registry locations and confirm they are on the same node.

```
[user@ws ~]$ oc get pods -o wide
NAME                        READY     STATUS    RESTARTS   AGE       IP
NODE
docker-registry-1-sw6ms    1/1       Unknown   0          2d
172.16.8.3        openshift-infra-1.example.com
docker-registry-1-v5rn7    1/1       Running   0          2m
172.16.4.3        openshift-infra-0.example.com
registry-console-1-vdw1k   1/1       Running   0          2d
172.16.14.2       openshift-node-0.example.com
router-1-83n6s             0/1       Pending   0          2m
<none>
router-1-kw8sd             1/1       Unknown   0          2d
192.168.155.183   openshift-infra-1.example.com
router-1-zm9fh             1/1       Running   0          2d
192.168.155.182   openshift-infra-0.example.com
```

Confirm the router is still working by refreshing the application browser page.

CHAPTER 5. PERSISTENT STORAGE

Container storage by default is not persistent. For example, if a new build triggers on a running container then data on the running container is lost. If a container terminates, all of the changes to its local file system are lost.

OpenShift offers different types of persistent storage. Persistent storage ensures data which should persist between builds and container migrations is available. The different storage options can be found at https://docs.openshift.com/container-platform/3.5/architecture/additional_concepts/storage.html#types-of-persistent-volumes. When choosing a persistent storage back-end ensure the back-end supports the scaling, speed, and redundancy the project requires.

5.1. PERSISTENT VOLUMES

`Persistent volumes` (**PV**) provide pods with non-ephemeral storage by configuring and encapsulating underlying storage sources. A **`persistent volume claim`** (**PVC**) abstracts an underlying **PV** to provide provider agnostic storage to OpenShift resources. A **PVC**, when successfully fulfilled by the system, mounts the persistent storage to a specific directory (**`mountPath`**) within one or more pods. From the container point of view, the **`mountPath`** is connected to the underlying storage mount points by a **`bind-mount`**.

5.1.1. Create an NFS Persistent Volume

Log in to the first master node **`openshift-master-0`** to define a new persistent volume. Creation of persistent volumes requires privileges that a default user account does not have. For this example, the **`system:admin`** account is used due to the account having cluster-admin privileges. This example provides an outline for any future **PVC**.

NOTE

In a production environment, different tiers of storage are assigned as needed for different workloads.

nfs-pv.yaml

```
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv001
spec:
  capacity:
    storage: 5Gi
  accessModes:
  - ReadWriteOnce
  nfs:
    path: /pv1
    server: nfs-0.example.com
  persistentVolumeReclaimPolicy: Recycle
```

Create the **PV** object using the **`yaml`** file above.

```
[root@openshift-master-0 ~]# oc create -f nfs-pv.yaml
```

5.1.2. Create an NFS Persistent Volumes Claim

The persistent volume claim (**PVC**) changes the pod from using **EmptyDir** non-persistent storage to storage backed by an NFS volume or **PV** as created above. The **PVC** will be created as long as the storage size is equal or greater to the **PV** and the **accessModes** are the same, e.g. **ReadWriteOnce**.

nfs-pvc.yaml

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: db
spec:
  accessModes:
  - ReadWriteOnce
  resources:
     requests:
       storage: 5Gi
```

```
[root@openshift-master-0 ~]# oc create -f nfs-pvc.yaml

persistentvolumeclaim "db" created
```

5.1.3. Deleting a Persistent Volumes Claim

There may come a point in which a **PVC** is no longer necessary for a project. The following removes the **PVC**.

```
[root@openshift-master-0 ~]# oc delete pvc db
persistentvolumeclaim "db" deleted

[root@openshift-master-0 ~]# oc get pvc db
No resources found.
Error from server: persistentvolumeclaims "db" not found
```

5.2. STORAGE CLASSES

The **StorageClass** resource object describes and classifies different types of storage by providing the means for passing parameters between user projects and the cluster storage back-end for dynamically provisioned storage on demand. **StorageClass** objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. **Cluster Administrators (cluster-admin)** or **Storage Administrators (storage-admin)** define and create **StorageClass** objects so users and developers can request storage without needing intimate knowledge about the underlying storage back ends. Thus, the naming scheme of the **storage class** defined in the **StorageClass** object should reflect the types and capabilities of storage it maps to (i.e. **HDD** vs **SDD** or **NVS** vs **Gluster**).

CHAPTER 6. CONCLUSION

Red Hat solutions involving the **OpenShift Container Platform** are created to deliver a production-ready foundation that simplifies the deployment process, shares the latest best practices, and provides a stable highly-available environment on which to run production applications.

This reference architecture covered the following topics:

- A completely provisioned infrastructure in **Red Hat Virtualization**.
- OpenShift Masters distributed across multiple **Red Hat Virtualization** hypervisor nodes utilizing anti-affinity groups.
- Infrastructure nodes likewise distributed across multiple **Red Hat Virtualization** hypervisor nodes with Router and Registry pods scaled accordingly.
- Native integration with **Red Hat Virtualization** services like thin-provisioned disks and HA.
- Creation of applications.
- Validation of the environment including fail-over tests.

For any questions or concerns, please email refarch-feedback@redhat.com and be sure to visit the Red Hat Reference Architecture page to find about all of our Red Hat solution offerings.

APPENDIX A. CONTRIBUTORS

Ken Bell, content reviewer

Ryan Cook, content reviewer

Rayford Johnson, content reviewer

Davis Phillips, content reviewer

APPENDIX B. REINSTALL OPENSHIFT CONTAINER PLATFORM

In case of a misconfiguration that puts the OpenShift installation into an unrecoverable state, it is easiest to delete all environment VMs and start over. To aid in this, the openshift-ansible-contrib repository includes a playbook under **`reference_architectures/rhv-ansible/playbooks`** called **`ovirt-vm-uninstall.yaml`** which will connect to the Red Hat Virtualization cluster and instruct it to remove all VMs created for the installation. Run it in the same manner as the **`ovirt-vm-infra.yaml`** script:

```
[user@ws rhv-ansible]$ ansible-playbook -e@ocp-vars.yaml playbooks/ovirt-vm-uninstall.yaml
```

APPENDIX C. INSTALLING ON RHEL ATOMIC HOST

While this reference architecture uses **Red Hat Enterprise Linux 7** as the base for its virtual machines, other options include **Red Hat Enterprise Linux Atomic Host** as it is designed and optimized to run Linux Containers. This is a particularly good fit for **OpenShift Container Platform**, and much work has been done in the **OpenShift Container Platform** installer so it can be run seamlessly on **Red Hat Enterprise Linux Atomic Host**

In order to build this reference architecture on top of **Red Hat Enterprise Linux Atomic Host**, obtain an image of the latest version from the **Red Hat Customer Portal**. At the time of publishing, there are two virtual machine images available for download. Select the **Red Hat Virtualization** (OVA) format as the dynamic inventory script described in Section 2.6, “Dynamic Inventory” requires the **rhevm-guest-agent** container.

NOTE

The **ovirt-ansible** roles used in this reference architecture do not support uploading **OVA** formatted files at the time of this publication.

As a workaround, download and unpack the file with the following **tar** command:

```
$ tar xvf rhel-atomic-rhevm-7.4.2-8.x86_64.rhevm.ova images
images/
images/f556c960-9c4d-469e-a191-14c5c9a045e5/
images/f556c960-9c4d-469e-a191-14c5c9a045e5/da9e9854-3055-4029-b92d-
132053c999cf.meta
images/f556c960-9c4d-469e-a191-14c5c9a045e5/da9e9854-3055-4029-b92d-
132053c999cf
```

The images directory contains a subdirectory labeled **f556c960-9c4d-469e-a191-14c5c9a045e5** with an image file named **da9e9854-3055-4029-b92d-132053c999cf**. This file is a **QCOW2** file that can be uploaded to the **Red Hat Virtualization** environment using the **ovirt-ansible** roles.

Rename the file for easier reference later:

```
$ mv images/f556c960-9c4d-469e-a191-14c5c9a045e5/da9e9854-3055-4029-b92d-
132053c999cf \
  rhel-atomic-rhevm-7.4.2-8.x86_64.rhevm.qcow2
```

NOTE

The **ovirt-ansible** role **ovirt-image-template** calls for a downloadable URL, but also provides an image cache variable, **image_path** to specify the local file. If the **image_path** exists, the URL download will be skipped.

To avoid having to re-host the extracted image, set the **qcow2_url** variable to a random URL and set the **image_path** variable to the path of the extracted **QCOW2** image.

```
qcow2_url: http://www.redhat.com/
image_path: rhel-atomic-rhevm-7.4.2-8.x86_64.rhevm.qcow2
```

APPENDIX D. MANAGE VIRTUAL GUEST SUBSCRIPTIONS WITH VIRT-WHO

Red Hat sells cloud-related subscriptions that apply to a fixed number of hypervisors by CPU socket pairs, but provide unlimited guest subscriptions for virtual machines running on those hypervisors. In order to allow the guest subscriptions to associate with the hypervisor's subscription, Red Hat provides a virtualization agent called **`virt-who`**. The role of **`virt-who`** is to keep the satellite updated with a current list of virtual machines running on each licensed hypervisor to enable the virtual machines to properly access the correct unlimited guest subscriptions.

For this reference architecture, installation and subscriptions are managed through a **Red Hat Satellite** (version 6.2). The **`virt-who`** service runs on the satellite, and is configured to log in to the **Red Hat Virtualization** cluster to query its hypervisor nodes. Although it is beyond the scope of this document to guide the installation and configuration of the **`virt-who`** service, it is highly recommended to use the Red Hat Virtualization Agent (virt-who) Configuration Helper .

APPENDIX E. LINKS

- Product Documentation for Red Hat Virtualization 4.1
- Product Documentation for OpenShift Container Platform 3.5
- Best Practices for Red Hat Virtualization 4

APPENDIX F. REVISION HISTORY

Revision 3.6.2-0	**January 16, 2018**	**Chandler Wilkerson**

- **Updated architectural diagram with brand team's work.**

Revision 3.6.1-0	**November 16, 2017**	**Chandler Wilkerson**

- **Added appendix entry for Atomic Host**

Revision 3.6.0-0	**October 31, 2017**	**Chandler Wilkerson**

- **Update for OpenShift 3.6**

Revision 3.5.0-0	**August 4, 2017**	**Chandler Wilkerson**

- **Initial OpenShift 3.5 on RHV release**