Red Hat

Reference Architectures 2018

Deploying and Managing OpenShift 3.11 on VMware vSphere

Last Updated: 2019-05-09

Reference Architectures 2018 Deploying and Managing OpenShift 3.11 on VMware vSphere

Davis Phillips
refarch-feedback@redhat.com

Abstract

The purpose of this document is to provide guidelines and considerations for deploying and managing Red Hat OpenShift Container Platform on VMware vSphere.

Table of Contents

COMMENTS AND FEEDBACK 4

EXECUTIVE SUMMARY 5

WHAT IS RED HAT OPENSHIFT CONTAINER PLATFORM 6

REFERENCE ARCHITECTURE SUMMARY 7

CHAPTER 1. COMPONENTS AND CONSIDERATIONS 9
1.1. VMWARE VSPHERE ENVIRONMENT CONSIDERATIONS 9
1.2. INSTALLATION STEPS 9
1.3. VMWARE SOFTWARE DETAILS 9
1.4. LOAD BALANCERS 10
1.5. DNS 10
1.5.1. Application DNS 10
1.6. RED HAT OPENSHIFT CONTAINER PLATFORM COMPONENTS 11
1.6.1. OpenShift Instances 11
1.6.1.1. Master Instances 11
1.6.1.2. Infrastructure Instances 12
1.6.1.3. Application Instances 12
1.6.2. etcd 12
1.6.3. Labels 13
1.6.3.1. Labels as Alternative Hierarchy 13
1.6.3.2. Labels as Node Selector 13
1.7. SOFTWARE DEFINED NETWORKING 14
1.7.1. OpenShift SDN Plugins 14
1.8. CONTAINER STORAGE 15
1.9. PERSISTENT STORAGE 15
1.9.1. Storage Classes 15
1.9.1.1. Persistent Volumes 15
1.10. REGISTRY 15
1.11. AGGREGATED LOGGING 16
1.12. AGGREGATED METRICS 17

CHAPTER 2. RED HAT OPENSHIFT CONTAINER PLATFORM PREREQUISITES 19
2.1. NETWORKING 19
2.2. SHARED STORAGE 19
2.3. RESOURCE POOL, CLUSTER NAME AND FOLDER LOCATION 19
2.4. VMWARE VSPHERE CLOUD PROVIDER (VCP) 19
2.4.1. Enabling VCP 20
2.4.2. The VCP Configuration File 21
2.5. DOCKER VOLUME 22
2.6. ETCD VOLUME 23
2.7. OPENSHIFT LOCAL VOLUME 23
2.8. EXECUTION ENVIRONMENT 23
2.9. PREPARATIONS 24
2.9.1. Deployment host 24
2.9.1.1. Creating an SSH Keypair for Ansible 24
2.9.1.2. Enable Required Repositories and Install Required Playbooks 24
2.9.1.3. Configure Ansible 25
2.9.1.4. Prepare the Inventory File 25
2.10. VSPHERE VM INSTANCE REQUIREMENTS FOR RHOCP 28

2.10.1. Virtual Machine Hardware Requirements 28
2.11. SET UP DNS FOR RED HAT OPENSHIFT CONTAINER PLATFORM 29
2.11.1. Confirm Instance Deployment 30
2.12. CREATE AND CONFIGURE AN HAPROXY VMWARE VSPHERE INSTANCE 30
2.13. ENABLE REQUIRED REPOSITORIES AND PACKAGES TO OPENSHIFT INFRASTRUCTURE 31
2.14. OPENSHIFT AUTHENTICATION 31
2.15. INSTANCE VERIFICATION 32
2.16. PRIOR TO ANSIBLE INSTALLATION 33
2.17. INSTALLING RED HAT OPENSHIFT CONTAINER PLATFORM WITH VMWARE NSX-T (OPTIONAL) 35
2.17.1. Installing the VMware NSX Container Plug-in (NCP) 36
2.18. RED HAT OPENSHIFT CONTAINER PLATFORM PREQUISITES PLAYBOOK 37

CHAPTER 3. DEPLOYING RED HAT OPENSHIFT CONTAINER PLATFORM 38
3.1. REGISTRY VOLUME 38

CHAPTER 4. OPERATIONAL MANAGEMENT 40
4.1. OC ADM DIAGNOSTICS 40
4.2. USING THE DIAGNOSTICS TOOL 40
4.3. RUNNING DIAGNOSTICS IN A SERVER ENVIRONMENT 43
4.4. RUNNING DIAGNOSTICS IN A CLIENT ENVIRONMENT 43
4.5. ANSIBLE-BASED HEALTH CHECKS 43
4.5.1. Running Health Checks via ansible-playbook 46
4.6. RUNNING HEALTH CHECKS VIA DOCKER CLI 46

CHAPTER 5. CONCLUSION 48

APPENDIX A. CONTRIBUTORS 49

APPENDIX B. DEPLOYING A VSPHERE VM ENVIRONMENT (OPTIONAL) 50

APPENDIX C. CONFIGURING MASTERS 51

CHAPTER 6. CONFIGURING OPENSHIFT NODES FOR CLOUD PROVIDER 52

APPENDIX D. INSTALLING THE VMWARE NSX NSX CONTAINER PLUG-IN (NCP) 53

APPENDIX E. DEPLOYING RHOCS STORAGE FOR PERSISTENT STORAGE 54

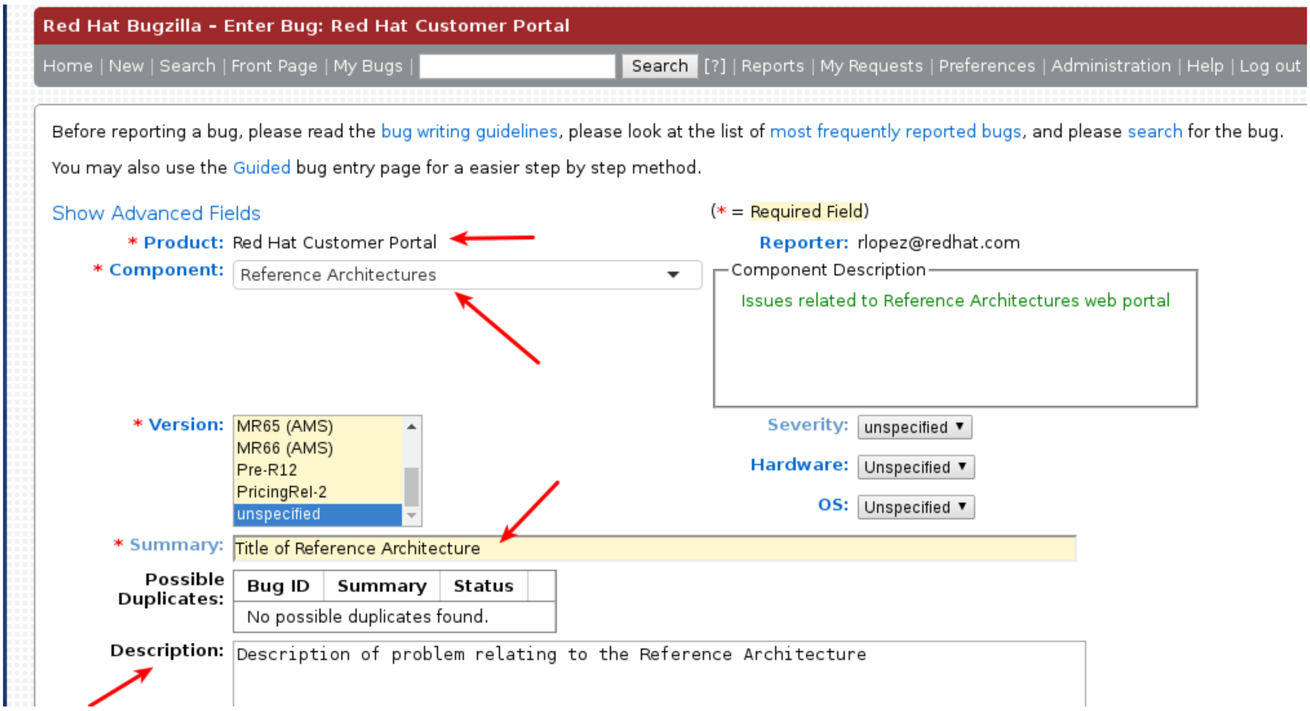
APPENDIX F. HOW TO CONFIGURE THE CLOUD PROVIDER FOR MULTIPLE VCENTER SERVERS 57

APPENDIX G. TROUBLESHOOTING ANSIBLE BY RED HAT 58

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. Please refer to the title within the email.

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.



EXECUTIVE SUMMARY

Staying ahead of the needs of an increasingly connected and demanding customer base demands solutions which are not only secure and supported, but robust and scalable, where new features may be delivered in a timely manner. In order to meet these requirements, organizations must provide the capability to facilitate faster development life cycles by managing and maintaining multiple products to meet each of their business needs. Red Hat solutions — for example Red Hat OpenShift Container Platform on VMware vSphere — simplify this process. Red Hat OpenShift Container Platform, providing a Platform as a Service (PaaS) solution, allows the development, deployment, and management of container-based applications while standing on top of a privately owned cloud by leveraging VMware vSphere as an Infrastructure as a Service (IaaS).

This reference architecture provides a methodology to deploy a highly available Red Hat OpenShift Container Platform on VMware vSphere environment by including a step-by-step solution along with best practices on customizing Red Hat OpenShift Container Platform.

This reference architecture is suited for system administrators, Red Hat OpenShift Container Platform administrators, and IT architects building Red Hat OpenShift Container Platform on VMware vSphere environments.

WHAT IS RED HAT OPENSHIFT CONTAINER PLATFORM

Red Hat OpenShift Container Platform is a Platform as a Service (PaaS) that provides developers and IT organizations with a cloud application platform for deploying new applications on secure, scalable resources with minimal configuration and management overhead. It allows developers to create and deploy applications by delivering a consistent environment for both development and during the runtime life cycle that requires no server management.

NOTE

For more information regarding about Red Hat OpenShift Container Platform visit: Red Hat OpenShift Container Platform Overview

REFERENCE ARCHITECTURE SUMMARY

The deployment of Red Hat OpenShift Container Platform varies among several factors that impact the installation process. Key considerations include:

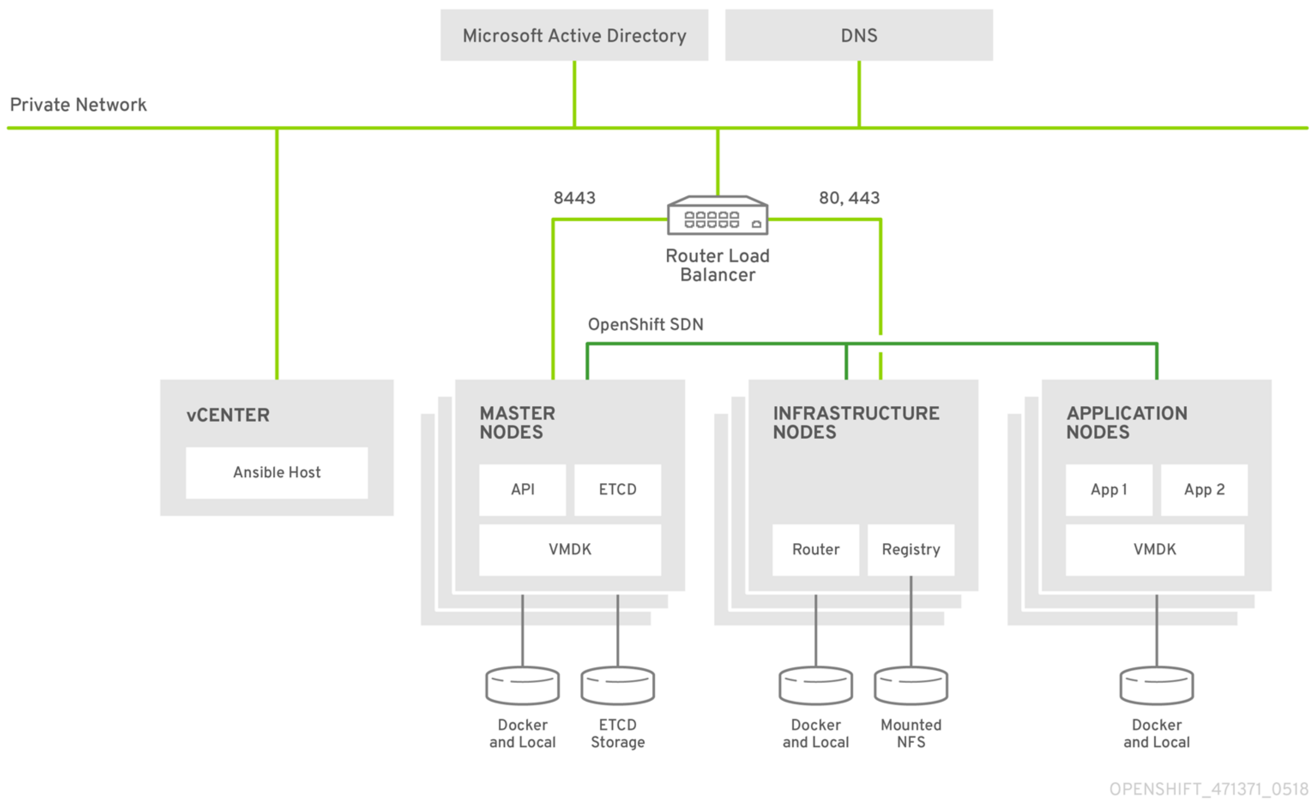
- *Which installation method do you want to use?*
- *How many instances do you require in the cluster?*
- *Is high availability required?*
- *Is my installation supported if integrating with other Red Hat technologies?*

For more information regarding the different options in installing an Red Hat OpenShift Container Platform cluster visit: Red Hat OpenShift Container Platform Chapter 2. Installing a Cluster

The initial planning process for this reference architecture answers these questions for this environment as follows:

- *Which installation method do you want to use?* Advanced Installation
- *How many instances do you require in the cluster?* 10
- *Is high availability required?* Yes
- *Is my installation supported if integrating with other Red Hat technologies?* Yes

A pictorial representation of the environment in this reference environment is shown below.



The Red Hat OpenShift Container Platform Architecture diagram shows the different components in the reference architecture.

The Red Hat OpenShift Container Platform instances:

- Deployment instance
- Three master instances
- Three infrastructure instances
- Three application instances

CHAPTER 1. COMPONENTS AND CONSIDERATIONS

1.1. VMWARE VSPHERE ENVIRONMENT CONSIDERATIONS

This chapter provides an overview and description of the reference architecture for a highly available Red Hat OpenShift Container Platform 3 environment deployed on a VMware private cloud.

The image shown above provides a high-level representation of the components within this reference architecture. Virtual machine (VM) resources are highly available using VMware technologies; VMware HA (high availability), storage IO (input/output) control, and resource allocation via hypervisor affinity and anti-affinity rules. The Ansible deployment host is a virtual machine and acts as the entry point for access to the hosts and performs configuration of the internal servers by ensuring that all Secure Shell (SSH) traffic passes through it.

Authentication is managed by Microsoft Active Directory via lightweight directory access protocol (LDAP) authentication. OpenShift on VMware has three cloud native storage options; virtual machine persistent storage, network file system (NFS) and Gluster file system (CNS/CRS).

Virtual machine persistent storage is housed on virtual machine disk VMDKs on datastores located on external logical unit numbers (LUNs) or NFS shares.

The other storage utilized is for container persistent storage including the OCP registry. The network is configured to leverage a single load balancer for access to the OpenShift API & Console (8443/tcp) and the OpenShift routers (80/tcp, 443/tcp).

Finally, the image shows that domain name system (DNS) is handled by an external DNS source. This DNS source should be pre-configured with the proper entries prior to deployment. In this case the solutions engineering team is managing all DNS entries through a BIND server and a conditional lookup zone in Microsoft DNS.

1.2. INSTALLATION STEPS

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

Phase 1: Provision the VM infrastructure on VMware (See Appendix B, *Deploying a vSphere VM Environment (Optional)*) Phase 2: Install Red Hat OpenShift Container Platform on VMware Phase 3: Post deployment activities

Provisioning of the VMware environment is a prerequisite, and outside the scope of this document. Phase 1 proceeds with the deployment of virtual machines, following requirements listed in the Section 2.10.1, "Virtual Machine Hardware Requirements"

Phase 2 is the installation of OpenShift Container Platform, which is done via the Ansible playbooks installed by the openshift-ansible-playbooks rpm package. During Phase 2 the router and registry are also deployed.

The last phase, Phase 3, concludes the deployment by confirming the environment was deployed properly. This is done by running some command line tools.

1.3. VMWARE SOFTWARE DETAILS

This reference architecture utilizes the following versions of VMware software:

Table 1.1. Software versions

Software	Version
vCenter Server via VCSA	6.7 U2 Build 13010631
vSphere Server	6.7.0 Build 13006603

1.4. LOAD BALANCERS

This guide uses an external load balancer running **haproxy** to offer a single entry point for the many Red Hat OpenShift Container Platform components. Organizations can provide their own currently deployed load balancers in the event that the service already exists.

The Red Hat OpenShift Container Platform console, provided by the Red Hat OpenShift Container Platform *master* nodes, can be spread across multiple instances to provide both load balancing and high availability properties.

Application traffic passes through the Red Hat OpenShift Container Platform Router on its way to the container processes. The Red Hat OpenShift Container Platform Router is a reverse proxy service container that multiplexes the traffic to multiple containers making up a scaled application running inside Red Hat OpenShift Container Platform. The load balancer used by *infra* nodes acts as the public view for the Red Hat OpenShift Container Platform applications.

The destination for the master and application traffic must be set in the load balancer configuration after each instance is created, the floating IP address is assigned and before the installation. A single **haproxy** load balancer can forward both sets of traffic to different destinations.

1.5. DNS

DNS service is an important component in the Red Hat OpenShift Container Platform environment. Regardless of the provider of DNS, an organization is required to have certain records in place to serve the various Red Hat OpenShift Container Platform components.

Since the load balancer values for the Red Hat OpenShift Container Platform master service and infrastructure nodes running router pods are known beforehand, entries should be configured into the DNS prior to starting the deployment procedure.

1.5.1. Application DNS

Applications served by OpenShift are accessible by the router on ports 80/TCP and 443/TCP. The router uses a *wildcard* record to map all host names under a specific sub domain to the same IP address without requiring a separate record for each name.

This allows Red Hat OpenShift Container Platform to add applications with arbitrary names as long as they are under that sub domain.

For example, a wildcard record for ***.apps.example.com** causes DNS name lookups for **tax.apps.example.com** and **home-goods.apps.example.com** to both return the same IP address: **10.19.x.y**. All traffic is forwarded to the OpenShift Routers. The Routers examine the HTTP headers of the queries and forward them to the correct destination.

With a load-balancer host address of 10.19.x.y, the wildcard DNS record can be added as follows:

Table 1.2. Load Balancer DNS records

IP Address	Hostname	Purpose
10.19.x.y	***.apps.example.com**	User access to application web services

1.6. RED HAT OPENSHIFT CONTAINER PLATFORM COMPONENTS

Red Hat OpenShift Container Platform comprises of multiple instances running on VMware vSphere that allow for scheduled and configured OpenShift services and supplementary containers. These containers can have persistent storage, if required, by the application and integrate with optional OpenShift services such as logging and metrics.

1.6.1. OpenShift Instances

Instances running the Red Hat OpenShift Container Platform environment run the **atomic-openshift-node** service that allows for the container orchestration of scheduling pods. The following sections describe the different instance and their roles to develop a Red Hat OpenShift Container Platform solution.

1.6.1.1. Master Instances

Master instances run the OpenShift master components, including the API server, controller manager server, and optionally **etcd**. The master manages nodes in its Kubernetes cluster and schedules pods to run on nodes.

NOTE

The master instances are considered nodes as well and run the **atomic-openshift-node** service.

For optimal performance, the **etcd** service should run on the masters instances. When collocating **etcd** with master nodes, at least three instances are required. In order to have a single entry-point for the API, the master nodes should be deployed behind a load balancer.

In order to create master instances, set the following in the inventory file as:

```
... [OUTPUT ABBREVIATED] ...
[etcd]
master1.example.com
master2.example.com
master3.example.com

[masters]
master1.example.com
master2.example.com
master3.example.com

[nodes]
master1.example.com openshift_node_group_name="node-config-master"
master2.example.com openshift_node_group_name="node-config-master"
master3.example.com openshift_node_group_name="node-config-master"
```

NOTE

See the official OpenShift documentation for a detailed explanation on master nodes.

1.6.1.2. Infrastructure Instances

The infrastructure instances run the **atomic-openshift-node** service and host the Red Hat OpenShift Container Platform components such as Registry, Prometheus and Hawkular metrics. The infrastructure instances also run the Elastic Search, Fluentd, and Kibana(**EFK**) containers for aggregate logging. Persistent storage should be available to the services running on these nodes.

Depending on environment requirements at least three infrastructure nodes are required to provide a sharded/highly available aggregated logging service and to ensure that service interruptions do not occur during a reboot.

NOTE

For more infrastructure considerations, visit the official OpenShift documentation.

When creating infrastructure instances, set the following in the inventory file as:

```
... [OUTPUT ABBREVIATED] ...
[nodes]
infra1.example.com openshift_node_group_name="node-config-infra"
infra2.example.com openshift_node_group_name="node-config-infra"
infra3.example.com openshift_node_group_name="node-config-infra"
```

NOTE

The router and registry pods automatically are scheduled on nodes with the role of 'infra'.

1.6.1.3. Application Instances

The Application (app) instances run the **atomic-openshift-node** service. These nodes should be used to run containers created by the end users of the OpenShift service.

When creating node instances, set the following in the inventory file as:

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

[nodes]
node1.example.com openshift_node_group_name="node-config-compute"
node2.example.com openshift_node_group_name="node-config-compute"
node3.example.com openshift_node_group_name="node-config-compute"
```

1.6.2. etcd

etcd is a consistent and highly-available key value store used as Red Hat OpenShift Container Platform's backing store for all cluster data. **etcd** stores the persistent master state while other components watch **etcd** for changes to bring themselves into the desired state.

Since values stored in **etcd** are critical to the function of Red Hat OpenShift Container Platform, firewalls should be implemented to limit the communication with **etcd** nodes. Inter-cluster and client-cluster communication is secured by utilizing x509 Public Key Infrastructure (PKI) key and certificate pairs.

etcd uses the RAFT algorithm to gracefully handle leader elections during network partitions and the loss of the current leader. For a highly available Red Hat OpenShift Container Platform deployment, an odd number (starting with three) of **etcd** instances are required.

1.6.3. Labels

Labels are key/value pairs attached to objects such as pods. They are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users but do not directly imply semantics to the core system. Labels can also be used to organize and select subsets of objects. Each object can have a set of labels defined at creation time or subsequently added and modified at any time.

NOTE

Each key must be unique for a given object.

```
"labels": {
  "key1" : "value1",
  "key2" : "value2"
}
```

Index and reverse-index labels are used for efficient queries, watches, sorting and grouping in UIs and CLIs, etc. Labels should not be polluted with non-identifying, large and/or structured data. Non-identifying information should instead be recorded using annotations.

1.6.3.1. Labels as Alternative Hierarchy

Service deployments and batch processing pipelines are often multi-dimensional entities (e.g., multiple partitions or deployments, multiple release tracks, multiple tiers, multiple micro-services per tier). Management of these deployments often requires cutting across the encapsulation of strictly hierarchical representations—especially those rigid hierarchies determined by the infrastructure rather than by users. Labels enable users to map their own organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings.

Example labels:

```
{"release" : "stable", "release" : "canary"}
{"environment" : "dev", "environment" : "qa", "environment" : "production"}
{"tier" : "frontend", "tier" : "backend", "tier" : "cache"}
{"partition" : "customerA", "partition" : "customerB"}
{"track" : "daily", "track" : "weekly"}
```

These are just examples of commonly used labels; the ability exists to develop specific conventions that best suit the deployed environment.

1.6.3.2. Labels as Node Selector

Node labels can be used as node selector where different nodes can be labeled to different use cases. The typical use case is to have nodes running **Red Hat OpenShift Container Platform** infrastructure components like the **Red Hat OpenShift Container Platform** registry, routers, metrics or logging components named "infrastructure nodes" to differentiate them from nodes dedicated to run user applications. Following this use case, the admin can label the "infrastructure nodes" with the label "region=infra" and the application nodes as "region=app". Other uses can be having different hardware in the nodes and have classifications like "type=gold", "type=silver" or "type=bronze".

The scheduler can be configured to use node labels to assign pods to nodes depending on the **node-selector**. At times it makes sense to have different types of nodes to run certain pods, the **node-selector** can be set to select which labels are used to assign pods to nodes.

1.7. SOFTWARE DEFINED NETWORKING

Red Hat OpenShift Container Platform offers the ability to specify how pods communicate with each other. This could be through the use of Red Hat provided Software-defined networks (SDN) or a third-party SDN.

Deciding on the appropriate internal network for an Red Hat OpenShift Container Platform step is a crucial step. Unfortunately, there is no right answer regarding the appropriate pod network to chose, as this varies based upon the specific scenario requirements on how a Red Hat OpenShift Container Platform environment is to be used.

For the purposes of this reference environment, the Red Hat OpenShift Container Platform **ovs-networkpolicy** SDN plug-in is chosen due to its ability to provide pod isolation using Kubernetes **NetworkPolicy**. The following section, “OpenShift SDN Plugins”, discusses important details when deciding between the three popular options for the internal networks - **ovs-multitenant**, **ovs-networkpolicy** and **ovs-subnet**.

1.7.1. OpenShift SDN Plugins

This section focuses on multiple plugins for pod communication within Red Hat OpenShift Container Platform using OpenShift SDN. The three plugin options are listed below.

- **ovs-subnet** - the original plugin that provides an overlay network created to allow pod-to-pod communication and services. This pod network is created using Open vSwitch (OVS).
- **ovs-multitenant** - a plugin that provides an overlay network that is configured using OVS, similar to the **ovs-subnet** plugin, however, unlike the **ovs-subnet** it provides Red Hat OpenShift Container Platform project level isolation for pods and services.
- **ovs-networkpolicy** - a plugin that provides an overlay network that is configured using OVS that provides the ability for Red Hat OpenShift Container Platform administrators to configure specific isolation policies using NetworkPolicy objects[1].

1: https://docs.openshift.com/container-platform/3.11/admin_guide/managing_networking.html#admin-guide-networking-networkpolicy

Network isolation is important, whichOpenShift SDN to choose?

With the above, this leaves two **OpenShift SDN** options: **ovs-multitenant** and **ovs-networkpolicy**. The reason **ovs-subnet** is ruled out is due to it not having network isolation.

While both **ovs-multitenant** and **ovs-networkpolicy** provide network isolation, the optimal choice comes down to what type of isolation is required. The **ovs-multitenant** plugin provides project-level isolation for pods and services. This means that pods and services from different projects cannot communicate with each other.

On the other hand, **ovs-networkpolicy** solves network isolation by providing project administrators the flexibility to create their own network policies using Kubernetes **NetworkPolicy** objects. This means that by default all pods in a project are accessible from other pods and network endpoints until **NetworkPolicy** objects are created. This in turn may allow pods from separate projects to communicate with each other assuming the appropriate **NetworkPolicy** is in place.

Depending on the level of isolation required, should determine the appropriate choice when deciding between **ovs-multitenant** and **ovs-networkpolicy**.

1.8. CONTAINER STORAGE

Container images are stored locally on the nodes running Red Hat OpenShift Container Platform pods. The **container-storage-setup** script uses the **/etc/sysconfig/docker-storage-setup** file to specify the storage configuration.

The **/etc/sysconfig/docker-storage-setup** file should be created before starting the **docker** service, otherwise the storage would be configured using a loopback device. The container storage setup is performed on all hosts running containers, therefore masters, infrastructure, and application nodes.

1.9. PERSISTENT STORAGE

Containers by default offer ephemeral storage but some applications require the storage to persist between different container deployments or upon container migration. **Persistent Volume Claims** (PVC) are used to store the application data. These claims can either be added into the environment by hand or provisioned dynamically using a **StorageClass** object.

1.9.1. Storage Classes

The **StorageClass** resource object describes and classifies different types of storage that can be requested, as well as provides a means for passing parameters to the backend 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 the **StorageClass** objects that users can use without needing any intimate knowledge about the underlying storage volume sources. Because of this the naming of the **storage class** defined in the **StorageClass** object should be useful in understanding the type of storage it maps whether that is storage from VMware vSphere or from **glusterfs** if deployed.

1.9.1.1. 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**.

1.10. REGISTRY

OpenShift can build container images from source code, deploy them, and manage their lifecycle. To enable this, OpenShift provides an internal, integrated registry that can be deployed in the OpenShift environment to manage images.

The registry stores images and metadata. For production environment, persistent storage should be used for the registry, otherwise any images that were built or pushed into the registry would disappear if the pod were to restart.

1.11. AGGREGATED LOGGING

One of the Red Hat OpenShift Container Platform optional components named Red Hat OpenShift Container Platform aggregated logging collects and aggregates logs from the pods running in the Red Hat OpenShift Container Platform cluster as well as **/var/log/messages** on nodes enabling Red Hat OpenShift Container Platform users to view the logs of projects which they have view access using a web interface.

Red Hat OpenShift Container Platform aggregated logging component it is a modified version of the **ELK** stack composed by a few pods running on the Red Hat OpenShift Container Platform environment:

- **Elasticsearch**: An object store where all logs are stored.
- **Kibana**: A web UI for Elasticsearch.
- **Curator**: Elasticsearch maintenance operations performed automatically on a per-project basis.
- **Fluentd**: Gathers logs from nodes and containers and feeds them to Elasticsearch.

NOTE

Fluentd can be configured to send a copy of the logs to a different log aggregator and/or to a different Elasticsearch cluster, see OpenShift documentation for more information.

Once deployed in the cluster, Fluentd (deployed as a **DaemonSet** on any node with the right labels) gathers logs from all nodes and containers, enriches the log document with useful metadata (e.g. namespace, container_name, node) and forwards them into Elasticsearch, where Kibana provides a web interface to users to be able to view any logs. Cluster administrators can view all logs, but application developers can only view logs for projects they have permission to view. To avoid users to see logs from pods in other projects, the Search Guard plugin for Elasticsearch is used.

A separate Elasticsearch cluster, a separate Kibana, and a separate Curator components can be deployed to form the **OPS cluster** where Fluentd send logs from the **default**, **openshift**, and **openshift-infra** projects as well as **/var/log/messages** on nodes into this different cluster. If the **OPS cluster** is not deployed those logs are hosted in the regular aggregated logging cluster.

Red Hat OpenShift Container Platform aggregated logging components can be customized for longer data persistence, pods limits, replicas of individual components, custom certificates, etc. The customization is provided by the **Ansible** variables as part of the deployment process.

The OPS cluster can be customized as well using the same variables using the suffix ***ops*** as in **openshift_logging_es_ops_pvc_size**.

NOTE

For more information about different customization parameters, see Aggregating Container Logs documentation.

Basic concepts for aggregated logging

- Cluster: Set of Elasticsearch nodes distributing the workload
- Node: Container running an instance of Elasticsearch, part of the cluster.
- Index: Collection of documents (container logs)

- Shards and Replicas: Indices can be split into sets of data containing the primary copy of the documents stored (primary shards) or backups of that primary copies (replica shards). Sharding allows the application to horizontally scaled the information and distributed/paralellized operations. Replication instead provides high availability and also better search throughput as searches are also executed on replicas.

> **WARNING**
>
> Using NFS storage as a volume or a persistent volume (or via NAS such as Gluster) is not supported for Elasticsearch storage, as Lucene relies on file system behavior that NFS does not supply. Data corruption and other problems can occur.

By default every Elasticsearch pod of the **Red Hat OpenShift Container Platform** aggregated logging components has the role of Elasticsearch master and Elasticsearch data node. If only 2 Elasticsearch pods are deployed and one of the pods fails, all logging stops until the second master returns, so there is no availability advantage to deploy 2 Elasticsearch pods.

> **NOTE**
>
> Elasticsearch shards require their own storage, but Red Hat OpenShift Container Platform **deploymentconfig** shares storage volumes between all its pods, therefore every Elasticsearch pod is deployed using a different **deploymentconfig** so it cannot be scaled using **oc scale**. In order to scale the aggregated logging Elasticsearch replicas after the first deployment, it is required to modify the **openshift_logging_es_cluster_size** in the inventory file and re-run the **openshift-logging.yml** playbook.

Below is an example of some of the best practices when deploying Red Hat OpenShift Container Platform aggregated logging. **Elasticsearch**, and **Kibana** are deployed on nodes with the role of "infras". Specifying the node role ensures that the **Elasticsearch** and **Kibana** components are not competing with applications for resources. A highly-available environment for Elasticsearch is deployed to avoid data loss, therefore, at least 3 Elasticsearch replicas are deployed and **openshift_logging_es_number_of_replicas** parameter is configured to be **1** at least. The settings below would be defined in a variable file or static inventory. The curator is now a scheduled job and no longer a deployment configuration.

```
openshift_logging_install_logging=true
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_size=30Gi
openshift_logging_es_cluster_size=1
openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_logging_kibana_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_logging_fluentd_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_logging_es_number_of_replicas=1
```

1.12. AGGREGATED METRICS

Red Hat OpenShift Container Platform has the ability to gather metrics from kubelet and store the values in **Heapster**. Red Hat OpenShift Container Platform Metrics provide the ability to view CPU, memory, and network-based metrics and display the values in the user interface. These metrics can allow for the horizontal autoscaling of pods based on parameters provided by an Red Hat OpenShift Container Platform user. It is important to understand capacity planning when deploying metrics into an Red Hat OpenShift Container Platform environment.

Red Hat OpenShift Container Platform metrics is composed by a few pods running on the Red Hat OpenShift Container Platform environment:

- **Heapster**: Heapster scrapes the metrics for CPU, memory and network usage on every pod, then exports them into Hawkular Metrics.
- **Hawkular Metrics**: A metrics engine that stores the data persistently in a Cassandra database.
- **Cassandra**: Database where the metrics data is stored.

Red Hat OpenShift Container Platform metrics components can be customized for longer data persistence, pods limits, replicas of individual components, custom certificates, etc. The customization is provided by the **Ansible** variables as part of the deployment process.

As best practices when metrics are deployed, persistent storage should be used to allow for metrics to be preserved. Node selectors should be used to specify where the Metrics components should run. In the reference architecture environment, the components are deployed on nodes with the role of "infra".

```
openshift_hosted_metrics_deploy=true
openshift_hosted_metrics_storage_kind=dynamic
openshift_hosted_metrics_storage_volume_size=10Gi
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_metrics_cassandra_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_metrics_heapster_nodeselector={"node-role.kubernetes.io/infra": "true"}
```

CHAPTER 2. RED HAT OPENSHIFT CONTAINER PLATFORM PREREQUISITES

A successful deployment of Red Hat OpenShift Container Platform requires many prerequisites. This consists of a set of infrastructure and host configuration steps prior to the actual installation of Red Hat OpenShift Container Platform using Ansible. In the following sections, details regarding the prerequisites and configuration changes required for an Red Hat OpenShift Container Platform on a VMware vSphere environment are discussed in detail.

For simplicity's sake, assume the vCenter environment is pre-existing and is configured with best practices for the infrastructure.

Technologies such as SIOC and VMware HA should already be configured where applicable. After the environment is provisioned, anti-affinity rules are established to ensure maximum uptime and optimal performance.

2.1. NETWORKING

An existing port group and virtual LAN (VLAN) are required for deployment. The environment can utilize a vSphere Distributed Switch (vDS) or vSwitch. The specifics of that are unimportant. However, to utilize network IO control and some of the quality of service (QoS) technologies that VMware employs, a vDS is required.

2.2. SHARED STORAGE

The vSphere hosts should have shared storage for the VMware virtual machine disk files (VMDKs). A best practice recommendation is to enable storage I/O control (SIOC) to address any performance issues caused by latency. This article discusses in depth how to do this.

NOTE

Some storage providers such as Dell Equallogic advise to disable storage I/O control (SIOC) as the array optimizes it. Check with the storage provider for details.

2.3. RESOURCE POOL, CLUSTER NAME AND FOLDER LOCATION

- Create a resource pool for the deployment
- Create a folder for the Red Hat OpenShift VMs for use with the vSphere Cloud Provider.
    - Ensure this folder exists under the datacenter then the cluster used for deployment

2.4. VMWARE VSPHERE CLOUD PROVIDER (VCP)

OpenShift Container Platform can be configured to access VMware vSphere VMDK Volumes, including using VMware vSphere VMDK Volumes as persistent storage for application data.

NOTE

The vSphere Cloud Provider steps below are for manual configuration. The OpenShift Ansible installer configures the cloud provider automatically when the proper variables are assigned during runtime. For more information on configuring masters and nodes see Appendix C, *Configuring Masters*

The vSphere Cloud Provider allows using vSphere managed storage within OpenShift Container Platform and supports:

- Volumes
- Persistent Volumes
- Storage Classes and provisioning of volumes.

2.4.1. Enabling VCP

To enable VMware vSphere cloud provider for OpenShift Container Platform:

1. Create a VM folder and move OpenShift Container Platform Node VMs to this folder.
2. Verify that the VM node names comply with the regex:

```
[a-z](([-0-9a-z]+)?[0-9a-z])?([a-z0-9](([-0-9a-z]+)?[0-9a-z])?)*
```

IMPORTANT

VM Names can not:

- Begin with numbers
- Have any capital letters
- Have any special characters except '-'
- Be shorter than three characters and longer than 63 characters

3. Set the **disk.EnableUUID** parameter to **TRUE** for each Node VM. This ensures that the VMDK always presents a consistent UUID to the VM, allowing the disk to be mounted properly. For every virtual machine node in the cluster, follow the steps below using the GOVC tool

   a. Download and install govc:

```
$ curl -LO https://github.com/vmware/govmomi/releases/download/v0.15.0/govc_linux_amd64.gz
$ gunzip govc_linux_amd64.gz
$ chmod +x govc_linux_amd64
$ cp govc_linux_amd64 /usr/bin/govc
```

a. Set up the GOVC environment:

```
$ export GOVC_URL='vCenter IP OR FQDN'
$ export GOVC_USERNAME='vCenter User'
$ export GOVC_PASSWORD='vCenter Password'
$ export GOVC_INSECURE=1
```

a. Find the Node VM paths:

```
$ govc ls /<datacenter>/vm/<vm-folder-name>
```

a. Set disk.EnableUUID to true for all VMs:

```
for VM in $(govc ls /<datacenter>/vm/<vm-folder-name>);do govc vm.change -
e="disk.enableUUID=1" -vm="$VM";done
```

> **NOTE**
>
> If Red Hat OpenShift Container Platform node VMs are created from a template VM, then **disk.EnableUUID=1** can be set on the template VM. VMs cloned from this template inherit this property.

1. Create and assign roles to the vSphere Cloud Provider user and vSphere entities. vSphere Cloud Provider requires the following privileges to interact with vCenter. See the vSphere Documentation Center for steps to create a custom role, user, and role assignment.

Roles	Privileges	Entities	Propagate to Children
manage-k8s-node-vms	Resource.AssignVMToPool System.Anonymous System.Read System.View VirtualMachine.Config.AddExistingDisk VirtualMachine.Config.AddNewDisk VirtualMachine.Config.AddRemoveDevice VirtualMachine.Config.RemoveDisk VirtualMachine.Inventory.Create VirtualMachine.Inventory.Delete	Cluster, Hosts, VM Folder	Yes
manage-k8s-volumes	Datastore.AllocateSpace Datastore.FileManagement (Low level file operations)	Datastore	No
k8s-system-read-and-spbm-profile-view	StorageProfile.View (Profile-driven storage view)	vCenter	No
ReadOnly	System.Anonymous System.Read System.View	Datacenter, Datastore Cluster, Datastore Storage Folder	No

2.4.2. The VCP Configuration File

Configuring Red Hat OpenShift Container Platform for VMware vSphere requires the */etc/origin/cloudprovider/vsphere.conf* file on each node.

If the file does not exist, create it, and add the following:

```
[Global]
        user = "username" 1
        password = "password" 2
        server = "10.10.0.2" 3
        port = "443" 4
        insecure-flag = "1" 5
        datacenter = "datacenter-name" 6
        datastore = "datastore-name" 7
        working-dir = "vm-folder-path" 8

[Disk]
        scsicontrollertype = pvscsi

[Network]
        network = "VM Network" 9
```

1. vCenter username for the vSphere cloud provider.
2. vCenter password for the specified user.
3. IP Address or FQDN for the vCenter server.
4. (Optional) Port number for the vCenter server. Defaults to port **443**.
5. Set to **1** if the vCenter uses a self-signed cert.
6. Name of the data center on which Node VMs are deployed.
7. Name of the datastore to use for provisioning volumes using the storage classes or dynamic provisioning. If datastore is located in a storage folder or datastore is a member of datastore cluster, specify the full datastore path. Verify that vSphere Cloud Provider user has the read privilege set on the datastore cluster or storage folder to be able to find datastore.
8. (Optional) The vCenter VM folder path in which the node VMs are located. It can be set to an empty path(**working-dir = ""**), if Node VMs are located in the root VM folder. The syntax resembles: **/<datacenter>/vm/<folder-name>/**
9. Specify the VM network portgroup to mark for the Internal Address of the node

2.5. DOCKER VOLUME

During the installation of Red Hat OpenShift Container Platform, the VMware instances created for RHOCP should include various **VMDK** volumes to ensure various OpenShift directories do not fill up the disk or cause disk contention in the **/var** partition.

Container images are stored locally on the nodes running Red Hat OpenShift Container Platform pods. The container-storage-setup script uses the **/etc/sysconfig/docker-storage-setup** file to specify the storage configuration.

The **/etc/sysconfig/docker-storage-setup** file must be created before starting the docker service, otherwise the storage is configured using a loopback device. The container storage setup is performed on all hosts running containers, therefore masters, infrastructure, and application nodes.

NOTE

The optional VM deployment in Appendix B, *Deploying a vSphere VM Environment (Optional)* takes care of Docker and other volume creation in addition to other machine preparation tasks like installing **chrony**, **open-vm-tools**, etc.

```
# cat /etc/sysconfig/docker-storage-setup
DEVS="/dev/sdb"
VG="docker-vol"
DATA_SIZE="95%VG"
STORAGE_DRIVER=overlay2
CONTAINER_ROOT_LV_NAME="dockerlv"
CONTAINER_ROOT_LV_MOUNT_PATH="/var/lib/docker"
```

NOTE

The value of the docker volume size should be at least 15 GB.

2.6. ETCD VOLUME

A VMDK volume should be created on the *master* instances for the storage of **/var/lib/etcd**. Storing **etcd** allows the similar benefit of protecting **/var** but more importantly provides the ability to perform snapshots of the volume when performing **etcd** maintenance.

NOTE

The value of the etcd volume size should be at least 25 GB.

2.7. OPENSHIFT LOCAL VOLUME

A **VMDK** volume should be created for the directory of **/var/lib/origin/openshift.local.volumes** that is used with the **perFSGroup** setting at installation and with the mount option of **gquota**. These settings and volumes set a quota to ensure that containers cannot grow to an unreasonable size.

NOTE

The value of OpenShift local volume size should be at least 30 GB.

```
# mkfs -t xfs /dev/sdc
# vi /etc/fstab
/dev/mapper/rhel-root   /                       xfs     defaults        0 0
UUID=8665acc0-22ee-4e45-970c-ae20c70656ef /boot                   xfs     defaults        0 0
/dev/sdc /var/lib/origin/openshift.local.volumes xfs gquota 0 0
```

2.8. EXECUTION ENVIRONMENT

Red Hat Enterprise Linux 7 is the only OS supported by the Red Hat OpenShift Container Platform installer therefore provider infrastructure deployment and installer must be run from one of the following locations:

- Local workstation/server/virtual machine

- Bastion instance
- Jenkins CI/CD build environment

This reference architecture focuses on deploying and installing Red Hat OpenShift Container Platform from local workstation/server/virtual machine. Jenkins CI/CD and Bastion are out of scope.

2.9. PREPARATIONS

2.9.1. Deployment host

2.9.1.1. Creating an SSH Keypair for Ansible

The VMware infrastructure requires an SSH key on the VMs for Ansible's use.

> **NOTE**
>
> The following task should be performed on the workstation/server/virtual machine where the Ansible playbooks are launched.

```
$ ssh-keygen -N '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:aaQHUf2rKHWvwwl4RmYcmCHswoouu3rdZiSH/BYgzBg root@ansible-test
The key's randomart image is:
+---[RSA 2048]----+
|   .. o=..       |
|E   ..o.. .      |
| * .  .... .     |
|. * o  +=.  .    |
|.. + o.=S    .   |
|o   + =o= . .    |
|.  . * = = +     |
|... . B . = .    |
+----[SHA256]-----+
```

> **NOTE**
>
> Add the ssh keys to the deployed virtual machines via **ssh-copy-id** or to the template prior to deployment.

2.9.1.2. Enable Required Repositories and Install Required Playbooks

Red Hat Subscription Manager registration and activate yum repositories

```
$ subscription-manager register

$ subscription-manager attach \
    --pool {{ pool_id }}
```

```
$ subscription-manager repos \
    --disable="*" \
    --enable=rhel-7-server-rpms \
    --enable=rhel-7-server-extras-rpms \
    --enable=rhel-7-server-ansible-2.7-rpms \
    --enable=rhel-7-server-ose-3.11-rpms

$ yum install -y \
    atomic-openshift-utils
```

2.9.1.3. Configure Ansible

ansible is installed on the deployment instance to perform the registration, installation of packages, and the deployment of the Red Hat OpenShift Container Platform environment on the master and node instances.

Before running playbooks, it is important to create a *ansible.cfg* to reflect the deployed environment:

```
$ cat ~/ansible.cfg

[defaults]
forks = 20
host_key_checking = False
roles_path = roles/
gathering = smart
remote_user = root
private_key = ~/.ssh/id_rsa
fact_caching = jsonfile
fact_caching_connection = $HOME/ansible/facts
fact_caching_timeout = 600
log_path = $HOME/ansible.log
nocows = 1
callback_whitelist = profile_tasks

[ssh_connection]
ssh_args = -C -o ControlMaster=auto -o ControlPersist=900s -o GSSAPIAuthentication=no -o
PreferredAuthentications=publickey
control_path = %(directory)s/%%h-%%r
pipelining = True
timeout = 10

[persistent_connection]
connect_timeout = 30
connect_retries = 30
connect_interval = 1
```

2.9.1.4. Prepare the Inventory File

NOTE

If using VMware NSX, see Section 2.17, "Installing Red Hat OpenShift Container Platform with VMware NSX-T (Optional)"

This section provides an example inventory file required for an advanced installation of Red Hat OpenShift Container Platform.

The inventory file contains both variables and instances used for the configuration and deployment of Red Hat OpenShift Container Platform. In the example below, some values are **bold** and must reflect the deployed environment from the previous chapter.

The **openshift_cloudprovider_vsphere_*** values are required for Red Hat OpenShift Container Platform to be able to create **vSphere** resources such as (VMDK)s on datastores for persistent volumes.

```
$ cat /etc/ansible/hosts

  [OSEv3:children]
  ansible
  masters
  infras
  apps
  etcd
  nodes
  lb

  [OSEv3:vars]

  ansible_ssh_user=cloud-user
  deployment_type=openshift-enterprise
  openshift_release="v3.11"
  become=yes
  ansible_become=yes
  ansible_user=root

  # Authentication for registry images and RHN network
  oreg_auth_user="registry_user"
  oreg_auth_password="registry_password"
  rhsub_user=username
  rhsub_pass=password
  rhsub_pool=8a85f9815e9b371b015e9b501d081d4b

  # Authentication settings for OCP
  openshift_master_ldap_ca_file=/home/cloud-user/mycert.crt
  openshift_master_identity_providers=[{'name': 'idm', 'challenge': 'true', 'login': 'true', 'kind':
'LDAPPasswordIdentityProvider', 'attributes': {'id': ['dn'], 'email': ['mail'], 'name': ['cn'],
'preferredUsername': ['uid']}, 'bindDN': 'uid=admin,cn=users,cn=accounts,dc=example,dc=com',
'bindPassword': 'ldapadmin', 'ca': '/etc/origin/master/ca.crt', 'insecure': 'false', 'url':
'ldap://ldap.example.com/cn=users,cn=accounts,dc=example,dc=com?uid?sub?(memberOf=cn=ose-
user,cn=groups,cn=accounts,dc=openshift,dc=com)'}]

  # Registry
  openshift_hosted_registry_storage_kind=vsphere
  openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
  openshift_hosted_registry_storage_annotations=['volume.beta.kubernetes.io/storage-provisioner:
kubernetes.io/vsphere-volume']
  openshift_hosted_registry_replicas=1

  # vSphere Cloud provider
  openshift_cloudprovider_kind=vsphere
  openshift_cloudprovider_vsphere_username="administrator@vsphere.local"
```

```
    openshift_cloudprovider_vsphere_password="password"
    openshift_cloudprovider_vsphere_host="vcenter.example.com"
    openshift_cloudprovider_vsphere_datacenter=datacenter
    openshift_cloudprovider_vsphere_cluster=cluster
    openshift_cloudprovider_vsphere_resource_pool=ocp311
    openshift_cloudprovider_vsphere_datastore="datastore"
    openshift_cloudprovider_vsphere_folder=ocp311

    #VM deployment
    openshift_cloudprovider_vsphere_template="ocp-server-template"
    openshift_cloudprovider_vsphere_vm_network="VM Network"
    openshift_cloudprovider_vsphere_vm_netmask="255.255.255.0"
    openshift_cloudprovider_vsphere_vm_gateway="192.168.1.1"
    openshift_cloudprovider_vsphere_vm_dns="192.168.2.250"
    openshift_required_repos=['rhel-7-server-rpms', 'rhel-7-server-extras-rpms', 'rhel-7-server-ose-
3.11-rpms']

    # OCP vars
    openshift_master_cluster_method=native
    openshift_node_local_quota_per_fsgroup=512Mi
    default_subdomain=example.com
    openshift_master_cluster_hostname=openshift.example.com
    openshift_master_cluster_public_hostname=openshift.example.com
    openshift_master_default_subdomain=apps.example.com
    os_sdn_network_plugin_name='redhat/openshift-ovs-networkpolicy'

    [ansible]
    localhost

    [masters]
    master-0 vm_name=master-0 ipv4addr=10.x.y.103
    master-1 vm_name=master-1 ipv4addr=10.x.y.104
    master-2 vm_name=master-2 ipv4addr=10.x.y.105

    [infras]
    infra-0  vm_name=infra-0 ipv4addr=10.x.y.100
    infra-1  vm_name=infra-1 ipv4addr=10.x.y.101
    infra-2  vm_name=infra-2 ipv4addr=10.x.y.102

    [apps]
    app-0  vm_name=app-0 ipv4addr=10.x.y.106
    app-1  vm_name=app-1 ipv4addr=10.x.y.107
    app-2  vm_name=app-2 ipv4addr=10.x.y.108

    [etcd]
    master-0
    master-1
    master-2

    [lb]
    haproxy-0 vm_name=haproxy-0 ipv4addr=10.x.y.200

    [nodes]
    master-0 openshift_node_group_name="node-config-master"
    master-1 openshift_node_group_name="node-config-master"
    master-2 openshift_node_group_name="node-config-master"
```

```
infra-0  openshift_node_group_name="node-config-infra"
infra-1  openshift_node_group_name="node-config-infra"
infra-2  openshift_node_group_name="node-config-infra"
app-0  openshift_node_group_name="node-config-compute"
app-1  openshift_node_group_name="node-config-compute"
app-2  openshift_node_group_name="node-config-compute"
```

NOTE

For a downloadable copy of this inventory file please see the following repo

2.10. VSPHERE VM INSTANCE REQUIREMENTS FOR RHOCP

This reference environment should consist of the following instances:

- three *master* instances
- three *infrastructure* instances
- three *application* instances
- one *loadbalancer* instance

2.10.1. Virtual Machine Hardware Requirements

Table 2.1. Virtual Machine Node Requirements

Node Type	Hardware
Master	2 vCPU
	16GB RAM
	1 x 60GB - OS RHEL 7.6
	1 x 40GB - Docker volume
	1 x 40Gb - EmptyDir volume
	1 x 40GB - ETCD volume
App or Infra Node	2 vCPU
	8GB RAM
	1 x 60GB - OS RHEL 7.6
	1 x 40GB - Docker volume
	1 x 40Gb - EmptyDir volume

The master instances should contain three extra disks used for Docker storage and ETCD and OpenShift volumes. The application node instances use their additional disks for Docker storage and OpenShift volumes.

etcd requires that an odd number of cluster members exist. Three masters were chosen to support high availability and **etcd** clustering. Three infrastructure instances allow for minimal to zero downtime for applications running in the OpenShift environment. Applications instance can be one to many instances depending on the requirements of the organization.

See Appendix B, *Deploying a vSphere VM Environment (Optional)* for steps on deploying the vSphere environment.

NOTE

infra and *app* node instances can easily be added after the initial install.

2.11. SET UP DNS FOR RED HAT OPENSHIFT CONTAINER PLATFORM

The installation process for Red Hat OpenShift Container Platform depends on a reliable name service that contains an address record for each of the target instances.

An example DNS configuration is listed below:

NOTE

Using **/etc/hosts** is not valid, a proper DNS service must exist.

```
$ORIGIN apps.example.com.
*           A       10.x.y.200
$ORIGIN example.com.
haproxy-0       A   10.x.y.200
infra-0       A   10.x.y.100
infra-1       A   10.x.y.101
infra-2       A   10.x.y.102
master-0       A   10.x.y.103
master-1       A   10.x.y.104
master-2       A   10.x.y.105
app-0         A   10.x.y.106
app-1         A   10.x.y.107
app-2         A   10.x.y.108
```

Table 2.2. Subdomain for RHOCP Network

Domain Name	Description
example.com	All interfaces on the internal only network

Table 2.3. Sample FQDNs

Fully Qualified Name	Description
master-0.example.com	Name of the network interface on the **master-0** instance
infra-0.example.com	Name of the network interface on the **infra-0** instance
app-0.example.com	Name of the network interface on the **app-0** instance
openshift.example.com	Name of the Red Hat OpenShift Container Platform console using the address of the **haproxy-0** instance on the network

2.11.1. Confirm Instance Deployment

After the 3 *master*, *infra* and *app* instances have been created in vCenter, verify the creation of the VMware vSphere instances via:

```
$ govc ls /<datacenter>/vm/<folder>/
```

Using the values provided by the command, update the DNS master **zone.db** file as shown in with the appropriate IP addresses. Do not proceed to the next section until the DNS resolution is configured.

Attempt to ssh into one of the Red Hat OpenShift Container Platform instances now that the ssh identity is setup.

NOTE

No password should be prompted if working properly.

```
$ ssh master-1
```

2.12. CREATE AND CONFIGURE AN HAPROXY VMWARE VSPHERE INSTANCE

If an organization currently does not have a load balancer in place then **HAProxy** can be deployed. A load balancer such as **HAProxy** provides a single view of the Red Hat OpenShift Container Platform master services for the applications. The master services and the applications use different TCP ports so a single TCP load balancer can handle all of the inbound connections.

The load balanced DNS name that developers use must be in a DNS A record pointing to the **HAProxy** server before installation. For applications, a wildcard DNS entry must point to the **HAProxy** host.

The configuration of the HAProxy instance is completed within the subsequent steps as the deployment host configures the Red Hat subscriptions for all the instances and the Red Hat OpenShift Container Platform installer auto configures the HAProxy instance based upon the information found within the Red Hat OpenShift Container Platform inventory file.

2.13. ENABLE REQUIRED REPOSITORIES AND PACKAGES TO OPENSHIFT INFRASTRUCTURE

> **NOTE**
>
> The optional VM deployment in Appendix B, *Deploying a vSphere VM Environment (Optional)* takes care of this and all volume creation and other machine preparation like chrony, open-vm-tools, etc. Additionally, the **haproxy** instance will be deployed from it.

Ensure connectivity to all instances via the deployment instance via:

```
$ ansible all -m ping
```

Once connectivity to all instances has been established, register the instances via Red Hat Subscription Manager. This is accomplished using credentials or an activation key.

Via credentials the **ansible** command is as follows:

```
$ ansible all -m command -a "subscription-manager register --username <user> --password
'<password>'"
```

Via activation key, the **ansible** command is as follows:

```
$ ansible all -m command -a "subscription-manager register --org=<org_id> --activationkey=
<keyname>"
```

where the following options:

- -m module to use
- -a module argument

Once all the instances have been successfully registered, enable all the required RHOCP repositories on all the instances via:

```
$ ansible all -m command -a "subscription-manager repos \
   --enable="rhel-7-server-rpms" \
   --enable="rhel-7-server-extras-rpms" \
   --enable="rhel-7-server-ose-3.11-rpms" \
   --enable="rhel-7-server-ansible-2.7-rpms""
```

2.14. OPENSHIFT AUTHENTICATION

Red Hat OpenShift Container Platform provides the ability to use many different authentication platforms. For this reference architecture, LDAP is the preferred authentication mechanism. A listing of other authentication options are available at Configuring Authentication and User Agent .

When configuring LDAP as the authentication provider the following parameters can be added to the ansible inventory. An example is shown below.

```
openshift_master_identity_providers=[{'name': 'idm', 'challenge': 'true', 'login': 'true', 'kind':
'LDAPPasswordIdentityProvider', 'attributes': {'id': ['dn'], 'email': ['mail'], 'name': ['cn'],
```

```
'preferredUsername': ['uid']}, 'bindDN': 'uid=admin,cn=users,cn=accounts,dc=openshift,dc=com',
'bindPassword': 'ldapadmin', 'ca': '/etc/origin/master/ca.crt', 'insecure': 'false', 'url':
'ldap://ldap.example.com/cn=users,cn=accounts,dc=openshift,dc=com?uid?sub?
(memberOf=cn=ose-user,cn=groups,cn=accounts,dc=openshift,dc=com)'}]
```

NOTE

If using LDAPS, all the masters must have the relevant *ca.crt* file for LDAP in place prior to the installation, otherwise the installation fails. The file should be placed locally on the deployment instance and be called within the inventory file from the variable *openshift_master_ldap_ca_file*

2.15. INSTANCE VERIFICATION

It can be useful to check for potential issues or misconfigurations in the instances before continuing the installation process. Connect to every instance using the deployment host and verify the disks are properly created and mounted.

```
$ ssh deployment.example.com
$ ssh <instance>
$ lsblk
$ sudo journalctl
$ free -m
$ sudo yum repolist
```

where *instance* is for example *master-0.example.com*

For reference, below is example output of **lsblk** for the master nodes.

```
$ lsblk
NAME                     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                      8:0    0   60G  0 disk
├─sda1                     8:1    0  500M  0 part /boot
├─sda2                     8:2    0 39.5G  0 part
│ ├─rhel-root            253:0    0   55G  0 lvm  /
│ └─rhel-swap              253:1    0  3.9G  0 lvm
└─sda3                     8:3    0   20G  0 part
  └─rhel-root            253:0    0   55G  0 lvm  /
sdb                      8:16   0   40G  0 disk
└─sdb1                     8:17   0   40G  0 part
  └─docker--vol-dockerlv 253:2    0   40G  0 lvm  /var/lib/docker
sdc                      8:32   0   40G  0 disk /var/lib/origin/openshift.local.volumes
sdd                      8:48   0   40G  0 disk /var/lib/etcd
```

For reference, below is an example of output of **lsblk** for the infra and app nodes.

```
$ lsblk
sda                      8:0    0   60G  0 disk
├─sda1                     8:1    0  500M  0 part /boot
├─sda2                     8:2    0 39.5G  0 part
│ ├─rhel-root            253:0    0   55G  0 lvm  /
│ └─rhel-swap              253:1    0  3.9G  0 lvm
└─sda3                     8:3    0   20G  0 part
  └─rhel-root            253:0    0   55G  0 lvm  /
```

```
sdb                 8:16   0   40G  0 disk
└─sdb1                 8:17   0   40G  0 part
  └─docker--vol-dockerlv 253:2    0   40G  0 lvm  /var/lib/docker
sdc                 8:32   0   40G  0 disk /var/lib/origin/openshift.local.volumes
sdd                 8:48   0  300G  0 disk
```

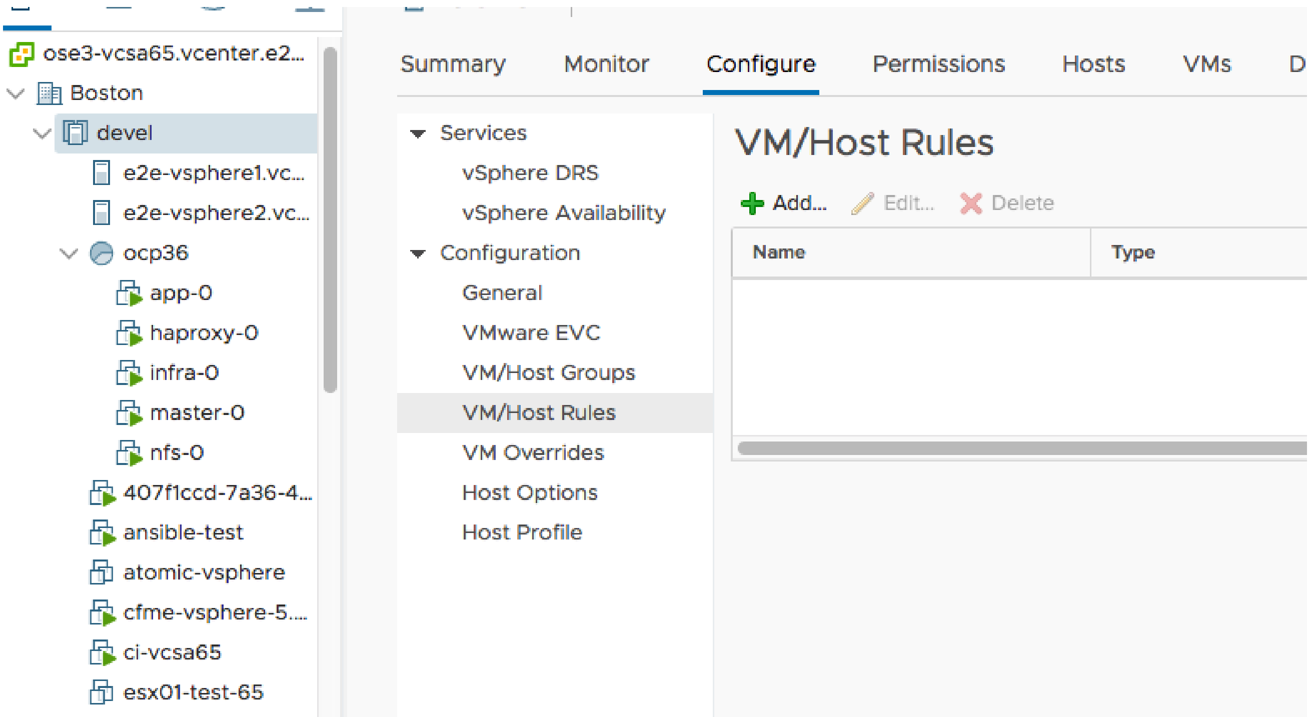
NOTE

The **docker-vol** LVM volume group may not be configured on **sdb** on all nodes at this stage, as this step is completed via the prerequisites playbook in the following section.

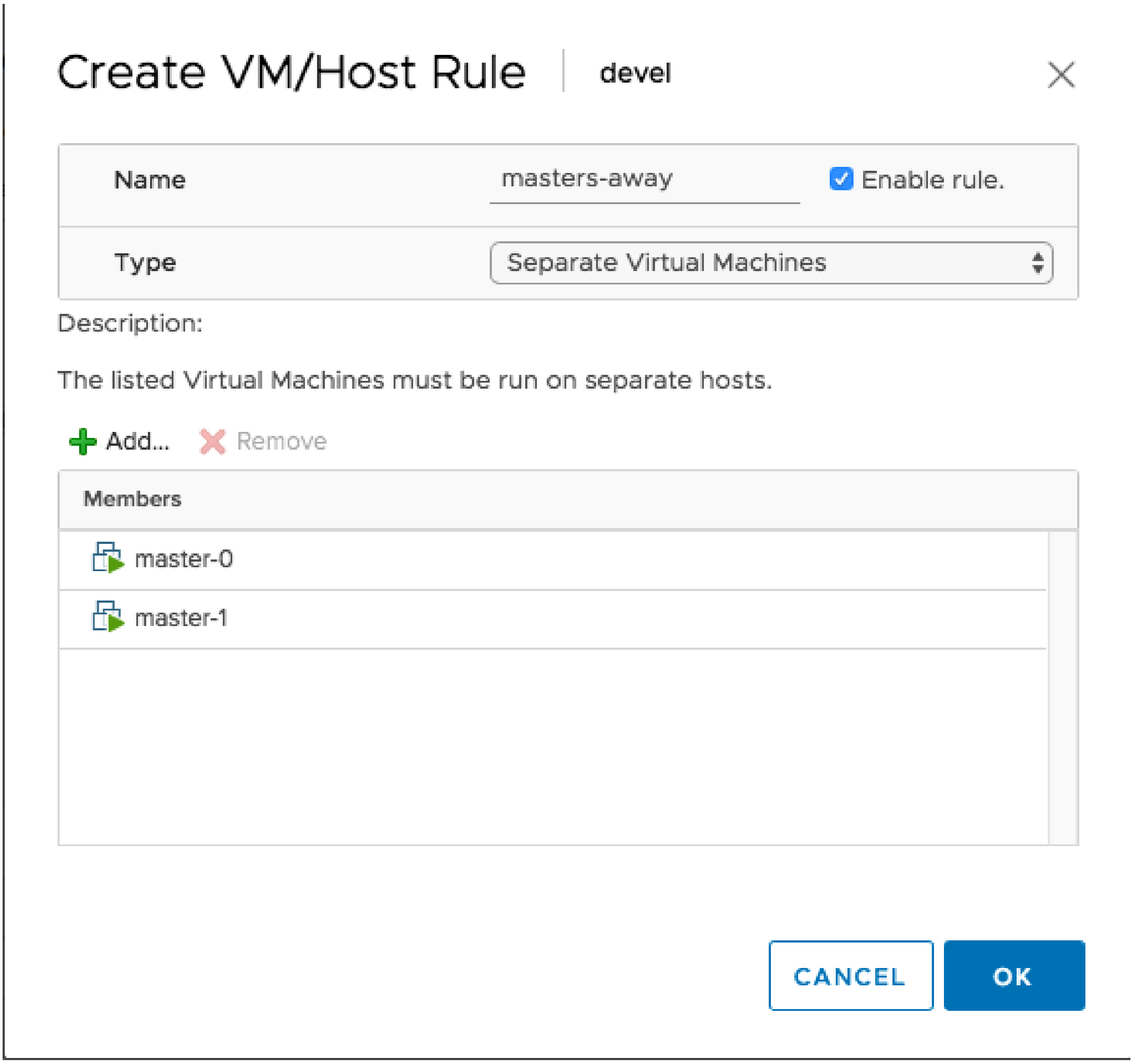
2.16. PRIOR TO ANSIBLE INSTALLATION

Prior to Chapter 4, *Operational Management*, create DRS anti-affinity rules to ensure maximum availability for the cluster.

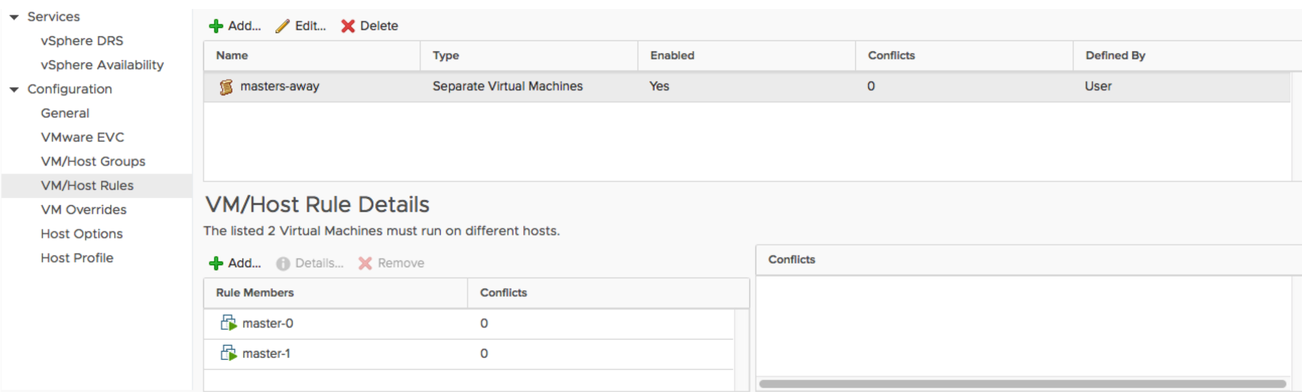
1. Open the VMware vCenter web client, select the cluster, choose configure.



1. Under Configuration, select VM/Host Rules.



1. Click add, and create a rules to keep the masters separate.



The following VMware documentation goes over creating and configuring anti-affinity rules in depth.

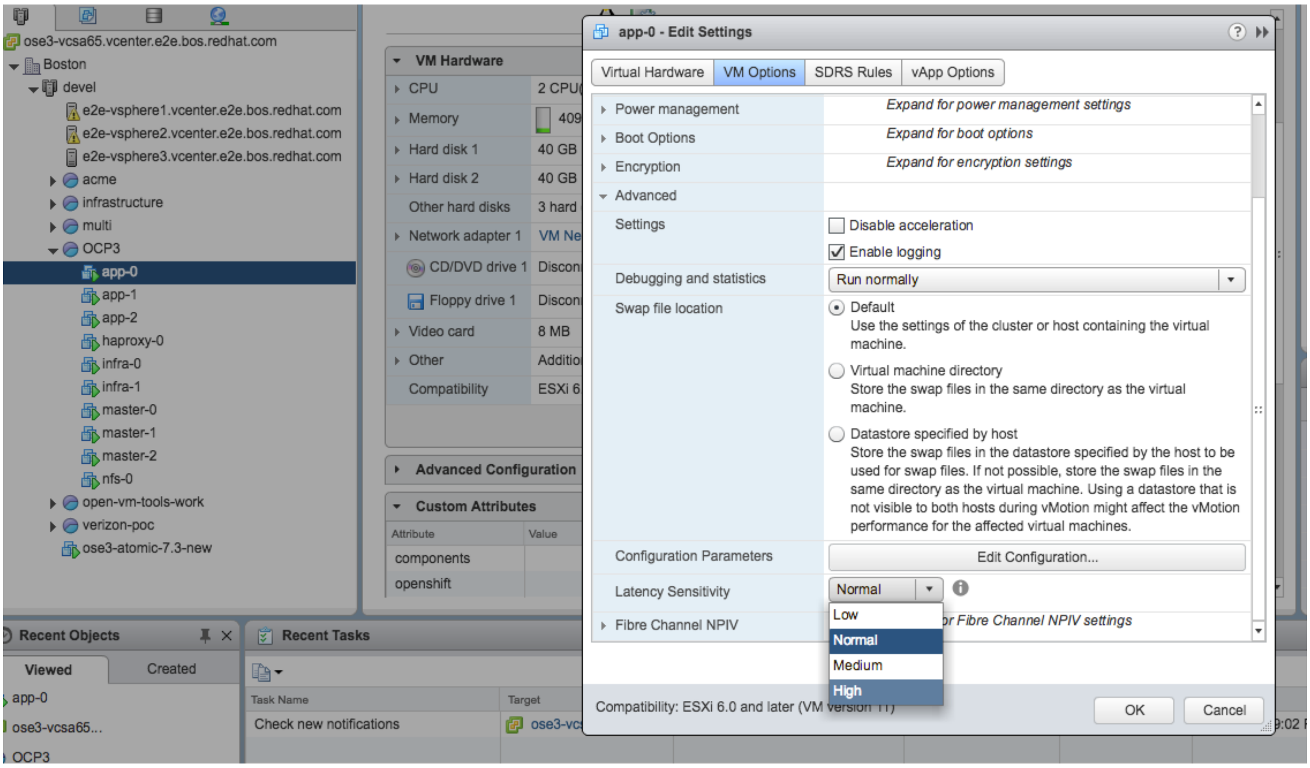
Lastly, set all of the VMs created to High VM Latency to ensure some additional tuning recommended by VMware for latency sensitive workloads as described here.

1. Open the VMware vCenter web client and under the virtual machines summary tab, in the 'VM Hardware' box select 'Edit Settings'.

2. Under, 'VM Options', expand 'Advanced'.

3. Select the 'Latency Sensitivity' dropbox and select 'High'.

Figure 2.1. VMware High Latency



2.17. INSTALLING RED HAT OPENSHIFT CONTAINER PLATFORM WITH VMWARE NSX-T (OPTIONAL)

NSX-T (NSX Transformers) provides network virtualization for hypervisors. NSX Container Plug-in (NCP) provides integration between NSX-T Data Center and container orchestrators such as Kubernetes, as well as integration between NSX-T Data Center and container based PaaS (platform as a service) software products such as OpenShift. This guide describes setting up NCP with OpenShift.

Table 2.4. Compatibility Requirements

Software Product	Version
NSX-T Data Center	2.3, 2.4
vSphere Server	6.5 U2, 6.7 U1, 6.7 U2
Container Host Operating System	RHEL 7.4, 7.5, 7.6
Platform as a Service	OpenShift 3.10, 3.11
Container Host Open vSwitch	2.10.2 (packaged with NSX-T Data Center 2.4)

The setup and configuration of VMware NSX is outside of the scope of this reference architecture. For more information on how to prepare NSX for the use of the VMware NSX Container Plug-in the following article describes in full how to prepare the environment.

2.17.1. Installing the VMware NSX Container Plug-in (NCP)

OpenShift node VMs must have two vNICs:

- A management vNIC connected to the logical switch that has an uplink to the management tier-1 router.
- The second vNIC on all VMs must have the following tags in NSX-T so that NCP knows which port is used as a parent VIF for all PODs running on the particular OpenShift node.

```
{'ncp/node_name': '<node_name>'}
{'ncp/cluster': '<cluster_name>'}
```

NSX-T Requirement:

- A tier-0 router.
- An overlay transport zone.
- An IP block for POD networking.
- (Optional) An IP Block for routed (no NAT) POD networking.
- An IP Pool for SNAT. By default the IP Block for POD networking is routable only inside NSX-T. NCP uses this IP Pool to provide connectivity to the outside.
- (Optional) Top and bottom firewall sections. NCP will place Kubernetes network policy rules between these two sections.
- Open vSwitch and CNI plugin RPMs must be hosted on an HTTP server reachable from the OpenShift node VMs.

NCP Docker Image

Currently the NCP docker image is not publicly available. You must have the image nsx-ncp in a local private registry, or download to the OpenShift nodes.

To push this image to a private registry, download the image to a system with Docker installed.

```
$ podman load -i nsx-ncp-rhel-xxx.yyyyyyyy.tar
$ podman push nsx-ncp registry.local/xxx.yyyyyyyy/nsx-ncp-rhel
```

Alternatively, the container image can be loaded on all of the cluster nodes locally:

```
    The prerequisites playbook will install and configure a container runtime.

    Now, the normal installation process of OpenShift can take place.

    Preparing the Ansible Hosts File You must specify NCP parameters in the Ansible hosts file for
NCP to be integrated with OpenShift. After you specify the following parameters in the Ansible hosts
```

file, installing OpenShift will nstall NCP automatically.

2.18. RED HAT OPENSHIFT CONTAINER PLATFORM PREQUISITES PLAYBOOK

The Red Hat OpenShift Container Platform Ansible installation provides a playbook to ensure all prerequisites are met prior to the installation of Red Hat OpenShift Container Platform. This includes steps such as registering all the nodes with Red Hat Subscription Manager and setting up the docker on the docker volumes.

Via the **ansible-playbook** command on the deployment instance, ensure all the prerequisites are met using **prerequisites.yml** playbook:

```
$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/prerequisites.yml
```

In the event that OpenShift fails to install or the prerequisites playbook fails, follow the steps in Appendix Appendix G, *Troubleshooting Ansible by Red Hat* to troubleshoot Ansible.

CHAPTER 3. DEPLOYING RED HAT OPENSHIFT CONTAINER PLATFORM

With the prerequisites met, the focus shifts to the installation of Red Hat OpenShift Container Platform. The installation and configuration is done via a series of **Ansible** playbooks and roles provided by the OpenShift RPM packages.

Run the installer playbook to install Red Hat OpenShift Container Platform:

```
$ ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/deploy_cluster.yml
```

The playbook runs through the complete process of installing Red Hat OpenShift Container Platform and reports a play recap showing the number of changes and errors (if any).

```
PLAY RECAP ***********************************************************************************
app1.example.com : ok=233  changed=40   unreachable=0    failed=0
app2.example.com : ok=233  changed=40   unreachable=0    failed=0
app3.example.com : ok=233  changed=40   unreachable=0    failed=0
infra1.example.com : ok=233  changed=40   unreachable=0    failed=0
infra2.example.com : ok=233  changed=40   unreachable=0    failed=0
infra3.example.com : ok=233  changed=40   unreachable=0    failed=0
localhost                  : ok=12   changed=0    unreachable=0    failed=0
master1.example.com : ok=674  changed=161  unreachable=0    failed=0
master2.example.com : ok=442  changed=103  unreachable=0    failed=0
master3.example.com : ok=442  changed=103  unreachable=0    failed=0

Tuesday 29 August 2018  10:34:49 -0400 (0:00:01.002)       0:29:54.775 ********
===============================================================================

openshift_hosted : Ensure OpenShift router correctly rolls out (best-effort today) -- 92.44s
openshift_hosted : Ensure OpenShift registry correctly rolls out (best-effort today) -- 61.93s
openshift_health_check ------------------------------------------------ 53.92s
openshift_common : Install the base package for versioning ------------- 42.15s
openshift_common : Install the base package for versioning ------------- 36.36s
openshift_hosted : Sanity-check that the OpenShift registry rolled out correctly -- 31.43s
cockpit : Install cockpit-ws ------------------------------------------ 27.65s
openshift_version : Get available atomic-openshift version ------------- 25.27s
etcd_server_certificates : Install etcd -------------------------------- 15.53s
openshift_master : Wait for master controller service to start on first master -- 15.21s
openshift_master : pause ---------------------------------------------- 15.20s
openshift_node : Configure Node settings ------------------------------- 13.56s
openshift_excluder : Install openshift excluder ------------------------ 13.54s
openshift_node : Install sdn-ovs package ------------------------------- 13.45s
openshift_master : Create master config -------------------------------- 11.92s
openshift_master : Create the scheduler config ------------------------- 10.92s
openshift_master : Create the policy file if it does not already exist -- 10.65s
openshift_node : Install Node service file ----------------------------- 10.43s
openshift_node : Install Node package ---------------------------------- 10.39s
openshift_node : Start and enable node --------------------------------- 8.96s
```

3.1. REGISTRY VOLUME

The OpenShift image registry requires a volume to ensure that images are saved in the event that the registry needs to migrate to another node.

The initial installation of OpenShift will configure vSphere-volume and make it the default storage class.

Add the following lines to the **/etc/ansible/hosts** file in the **[OSEv3:vars]** section to allow for the default vSphere storage class to serve as the backend storage.

```
openshift_hosted_registry_storage_kind=vsphere
openshift_hosted_registry_storage_access_modes=['ReadWriteOnce']
openshift_hosted_registry_storage_annotations=['volume.beta.kubernetes.io/storage-provisioner:
kubernetes.io/vsphere-volume']
openshift_hosted_registry_replicas=1
```

NOTE

The vSphere volume provider only supports the **ReadWriteOnce** access mode. Because of this the replica count of the registry must be set to 1.

If registry storage was not configured during the installation. After the deployment is finished, the following steps reconfigure the registry to use the vSphere-volume storage class:

```
cat << EOF > pvc-registry.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: vsphere-registry-storage
  annotations:
    volume.beta.kubernetes.io/storage-class: vsphere-standard
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 30Gi
EOF

# submit the PVC
oc create -f pvc-registry.yaml

# update the volume config to use our new PVC
oc volume dc docker-registry --add --name=registry-storage -t pvc --claim-name=vsphere-registry-
storage --overwrite

# rollout the new registry
oc rollout latest docker-registry

# verify the new volume
oc volume dc docker-registry
```

NOTE

The registry volume size should be at least 30GB.

CHAPTER 4. OPERATIONAL MANAGEMENT

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

NOTE

The following subsections are from OpenShift Documentation - Diagnostics Tool site. For the latest version of this section, reference the link directly.

4.1. OC ADM DIAGNOSTICS

The **oc adm diagnostics** command runs a series of checks for error conditions in the host or cluster. Specifically, it:

- Verifies that the default registry and router are running and correctly configured.
- Checks **ClusterRoleBindings** and **ClusterRoles** for consistency with base policy.
- Checks that all of the client configuration contexts are valid and can be connected to.
- Checks that SkyDNS is working properly and the pods have SDN connectivity.
- Validates master and node configuration on the host.
- Checks that nodes are running and available.
- Analyzes host logs for known errors.
- Checks that systemd units are configured as expected for the host.

4.2. USING THE DIAGNOSTICS TOOL

Red Hat OpenShift Container Platform may be deployed in numerous scenarios including:

- built from source
- included within a VM image
- as a container image
- via enterprise RPMs

Each method implies a different configuration and environment. The diagnostics were included within **openshift** binary to minimize environment assumptions and provide the ability to run the diagnostics tool within an Red Hat OpenShift Container Platform server or client.

To use the diagnostics tool, preferably on a master host and as cluster administrator, run a **sudo** user:

```
$ sudo oc adm diagnostics
```

The above command runs all available diagnostic skipping any that do not apply to the environment.

The diagnostics tool has the ability to run all or specific diagnostics via name or as an enabler to address issues within the Red Hat OpenShift Container Platform environment. For example:

```
$ sudo oc adm diagnostics <name1>
```

The options provided by the diagnostics tool require working configuration files. For example, the **NodeConfigCheck** does not run unless a node configuration is readily available.

Diagnostics verifies that the configuration files reside in their standard locations unless specified with flags (respectively, **--config**, **--master-config**, and **--node-config**)

The standard locations are listed below:

- Client:
  - As indicated by the **$KUBECONFIG** environment variable variable
  - *~/.kube/config file*
- Master:
  - */etc/origin/master/master-config.yaml*
- Node:
  - */etc/origin/node/node-config.yaml*

If a configuration file is not found or specified, related diagnostics are skipped.

Available diagnostics include:

Diagnostic Name	Purpose
AggregatedLogging	Check the aggregated logging integration for proper configuration and operation.
AnalyzeLogs	Check systemd service logs for problems. Does not require a configuration file to check against.
ClusterRegistry	Check that the cluster has a working Docker registry for builds and image streams.
ClusterRoleBindings	Check that the default cluster role bindings are present and contain the expected subjects according to base policy.
ClusterRoles	Check that cluster roles are present and contain the expected permissions according to base policy.
ClusterRouter	Check for a working default router in the cluster.
ConfigContexts	Check that each context in the client configuration is complete and has connectivity to its API server.

Diagnostic Name	Purpose
DiagnosticPod	Creates a pod that runs diagnostics from an application standpoint, which checks that DNS within the pod is working as expected and the credentials for the default service account authenticate correctly to the master API.
EtcdWriteVolume	Check the volume of writes against etcd for a time period and classify them by operation and key. This diagnostic only runs if specifically requested, because it does not run as quickly as other diagnostics and can increase load on etcd.
MasterConfigCheck	Check this particular hosts master configuration file for problems.
MasterNode	Check that the master node running on this host is running a node to verify that it is a member of the cluster SDN.
MetricsApiProxy	Check that the integrated Heapster metrics can be reached via the cluster API proxy.
NetworkCheck	Create diagnostic pods on multiple nodes to diagnose common network issues from an application standpoint. For example, this checks that pods can connect to services, other pods, and the external network. If there are any errors, this diagnostic stores results and retrieved files in a local directory (***/tmp/openshift/***, by default) for further analysis. The directory can be specified with the **--network-logdir** flag.
NodeConfigCheck	Checks this particular hosts node configuration file for problems.
NodeDefinitions	Check that the nodes defined in the master API are ready and can schedule pods.
RouteCertificateValidation	Check all route certificates for those that might be rejected by extended validation.
ServiceExternalIPs	Check for existing services that specify external IPs, which are disallowed according to master configuration.

Diagnostic Name	Purpose
UnitStatus	Check systemd status for units on this host related to Red Hat OpenShift Container Platform. Does not require a configuration file to check against.

> **NOTE**
>
> The logging curator is now a scheduled task instead of a deployment configuration. The diagnostics will return: Did not find a DeploymentConfig to support component 'curator'. Please see Bugzilla for more details.

4.3. RUNNING DIAGNOSTICS IN A SERVER ENVIRONMENT

An Ansible-deployed cluster provides additional diagnostic benefits for nodes within Red Hat OpenShift Container Platform cluster due to:

- Standard location for both master and node configuration files
- Systemd units are created and configured for managing the nodes in a cluster
- All components log to journald.

Standard location of the configuration files placed by an Ansible-deployed cluster ensures that running **sudo oc adm diagnostics** works without any flags. In the event, the standard location of the configuration files is not used, options flags as those listed in the example below may be used.

```
$ sudo oc adm diagnostics --master-config=<file_path> --node-config=<file_path>
```

For proper usage of the log diagnostic, systemd units and log entries within **journald** are required. If log entries are not using the above method, log diagnostics won't work as expected and are intentionally skipped.

4.4. RUNNING DIAGNOSTICS IN A CLIENT ENVIRONMENT

The diagnostics runs using as much access as the existing user running the diagnostic has available. The diagnostic may run as an ordinary user, a **cluster-admin** user or **cluster-admin** user.

A client with ordinary access should be able to diagnose its connection to the master and run a diagnostic pod. If multiple users or masters are configured, connections are tested for all, but the diagnostic pod only runs against the current user, server, or project.

A client with **cluster-admin** access available (for any user, but only the current master) should be able to diagnose the status of the infrastructure such as nodes, registry, and router. In each case, running **sudo oc adm diagnostics** searches for the standard client configuration file location and uses it if available.

4.5. ANSIBLE-BASED HEALTH CHECKS

Additional diagnostic health checks are available through the Ansible-based tooling used to install and manage Red Hat OpenShift Container Platform clusters. The reports provide common deployment problems for the current Red Hat OpenShift Container Platform installation.

These checks can be run either using the **ansible-playbook** command (the same method used during Advanced Installation) or as a containerized version of **openshift-ansible**. For the **ansible-playbook** method, the checks are provided by the **atomic-openshift-utils** RPM package.

For the containerized method, the **openshift3/ose-ansible** container image is distributed via the Red Hat Container Registry.

Example usage for each method are provided in subsequent sections.

The following health checks are a set of diagnostic tasks that are meant to be run against the Ansible inventory file for a deployed Red Hat OpenShift Container Platform cluster using the provided ***health.yml*** playbook.

> **WARNING**
>
> Due to potential changes the health check playbooks could make to the environment, the playbooks should only be run against clusters that have been deployed using Ansible with the same inventory file used during deployment. The changes consist of installing dependencies in order to gather required information. In some circumstances, additional system components (i.e. **docker** or networking configurations) may be altered if their current state differs from the configuration in the inventory file. These health checks should **only** be run if the administrator does not expect the inventory file to make any changes to the existing cluster configuration.

Table 4.1. Diagnostic Health Checks

Check Name	Purpose
etcd_imagedata_size	This check measures the total size of Red Hat OpenShift Container Platform image data in an etcd cluster. The check fails if the calculated size exceeds a user-defined limit. If no limit is specified, this check fails if the size of image data amounts to 50% or more of the currently used space in the etcd cluster. A failure from this check indicates that a significant amount of space in etcd is being taken up by Red Hat OpenShift Container Platform image data, which can eventually result in etcd cluster crashing. A user-defined limit may be set by passing the **etcd_max_image_data_size_bytes** variable. For example, setting **etcd_max_image_data_size_bytes=40000000000** causes the check to fail if the total size of image data stored in etcd exceeds 40 GB.

Check Name	Purpose
etcd_traffic	This check detects higher-than-normal traffic on an etcd host. The check fails if a **journalctl** log entry with an etcd sync duration warning is found. For further information on improving etcd performance, see Recommended Practices for Red Hat OpenShift Container Platform etcd Hosts and the Red Hat Knowledgebase.
etcd_volume	This check ensures that the volume usage for an etcd cluster is below a maximum user-specified threshold. If no maximum threshold value is specified, it is defaulted to **90%** of the total volume size. A user-defined limit may be set by passing the **etcd_device_usage_threshold_percent** variable.
docker_storage	Only runs on hosts that depend on the **docker** daemon (nodes and containerized installations). Checks that **docker**'s total usage does not exceed a user-defined limit. If no user-defined limit is set, **docker**'s maximum usage threshold defaults to 90% of the total size available. The threshold limit for total percent usage can be set with a variable in the inventory file, for example **max_thinpool_data_usage_percent=90**. This also checks that **docker**'s storage is using a supported configuration.
curator, **elasticsearch**, **fluentd**, **kibana**	This set of checks verifies that Curator, Kibana, Elasticsearch, and Fluentd pods have been deployed and are in a **running** state, and that a connection can be established between the control host and the exposed Kibana URL. These checks run only if the **openshift_logging_install_logging** inventory variable is set to **true**. Ensure that they are executed in a deployment where cluster logging has been enabled.

Check Name	Purpose
logging_index_time	This check detects higher than normal time delays between log creation and log aggregation by Elasticsearch in a logging stack deployment. It fails if a new log entry cannot be queried through Elasticsearch within a timeout (by default, 30 seconds). The check only runs if logging is enabled. A user-defined timeout may be set by passing the **openshift_check_logging_index_timeout_seconds** variable. For example, setting **openshift_check_logging_index_timeout_seconds=45** causes the check to fail if a newly-created log entry is not able to be queried via Elasticsearch after 45 seconds.

NOTE

A similar set of checks meant to run as part of the installation process can be found in Configuring Cluster Pre-install Checks. Another set of checks for checking certificate expiration can be found in Redeploying Certificates.

4.5.1. Running Health Checks via ansible-playbook

The **openshift-ansible** health checks are executed using the **ansible-playbook** command and requires specifying the cluster's inventory file and the ***health.yml*** playbook:

```
# ansible-playbook -i <inventory_file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-checks/health.yml
```

In order to set variables in the command line, include the **-e** flag with any desired variables in **key=value** format. For example:

```
# ansible-playbook -i <inventory_file> \
    /usr/share/ansible/openshift-ansible/playbooks/openshift-checks/health.yml
    -e openshift_check_logging_index_timeout_seconds=45
    -e etcd_max_image_data_size_bytes=40000000000
```

To disable specific checks, include the variable **openshift_disable_check** with a comma-delimited list of check names in the inventory file prior to running the playbook. For example:

```
openshift_disable_check=etcd_traffic,etcd_volume
```

Alternatively, set any checks to disable as variables with **-e openshift_disable_check=<check1>, <check2>** when running the **ansible-playbook** command.

4.6. RUNNING HEALTH CHECKS VIA DOCKER CLI

The **openshift-ansible** playbooks may run in a Docker container avoiding the requirement for installing and configuring Ansible, on any host that can run the **ose-ansible** image via the Docker CLI.

This is accomplished by specifying the cluster's inventory file and the ***health.yml*** playbook when running the following **docker run** command as a non-root user that has privileges to run containers:

```
# docker run -u `id -u` \ 1
    -v $HOME/.ssh/id_rsa:/opt/app-root/src/.ssh/id_rsa:Z,ro \ 2
    -v /etc/ansible/hosts:/tmp/inventory:ro \ 3
    -e INVENTORY_FILE=/tmp/inventory \
    -e PLAYBOOK_FILE=playbooks/openshift-checks/health.yml \ 4
    -e OPTS="-v -e openshift_check_logging_index_timeout_seconds=45 -e
etcd_max_image_data_size_bytes=40000000000" \ 5
    openshift3/ose-ansible
```

1. These options make the container run with the same UID as the current user, which is required for permissions so that the SSH key can be read inside the container (SSH private keys are expected to be readable only by their owner).

2. Mount SSH keys as a volume under ***/opt/app-root/src/.ssh*** under normal usage when running the container as a non-root user.

3. Change ***/etc/ansible/hosts*** to the location of the cluster's inventory file, if different. This file is bind-mounted to ***/tmp/inventory***, which is used according to the **INVENTORY_FILE** environment variable in the container.

4. The **PLAYBOOK_FILE** environment variable is set to the location of the ***health.yml*** playbook relative to ***/usr/share/ansible/openshift-ansible*** inside the container.

5. Set any variables desired for a single run with the **-e key=value** format.

In the above command, the SSH key is mounted with the **:Z** flag so that the container can read the SSH key from its restricted SELinux context. This ensures the original SSH key file is relabeled similarly to **system_u:object_r:container_file_t:s0:c113,c247**. For more details about **:Z**, see the **docker-run(1)** man page.

It is important to note these volume mount specifications because it could have unexpected consequences. For example, if one mounts (and therefore relabels) the ***$HOME/.ssh*** directory, **sshd** becomes unable to access the public keys to allow remote login. To avoid altering the original file labels, mounting a copy of the SSH key (or directory) is recommended.

It is plausible to want to mount an entire ***.ssh*** directory for various reasons. For example, this enables the ability to use an SSH configuration to match keys with hosts or modify other connection parameters. It could also allow a user to provide a ***known_hosts*** file and have SSH validate host keys, which is disabled by the default configuration and can be re-enabled with an environment variable by adding **-e ANSIBLE_HOST_KEY_CHECKING=True** to the **docker** command line.

CHAPTER 5. CONCLUSION

Red Hat solutions involving the Red Hat OpenShift Container Platform provide an excellent foundation for building a production ready environment which simplifies the deployment process, provides the latest best practices, and ensures stability by running applications in a highly available environment.

The steps and procedures described in this reference architecture provide system, storage, and Red Hat OpenShift Container Platform administrators the blueprints required to create solutions to meet business needs. Administrators may reference this document to simplify and optimize their Red Hat OpenShift Container Platform on VMware vSphere environments with the following tasks:

- Deciding between different internal network technologies
- Provisioning instances within VMware vSphere for Red Hat OpenShift Container Platform readiness
- Deploying Red Hat OpenShift Container Platform 3.11
- Using dynamic provisioned storage
- Verifying a successful installation
- Troubleshooting common pitfalls

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

APPENDIX A. CONTRIBUTORS

Davis Phillips, content provider

Joel Davis, content provider

Robbie Jerrom, content provider

Roger Lopez, content provider

Ryan Cook, content provider

Chandler Wilkerson, content provider

Chris Callegari, content provider

APPENDIX B. DEPLOYING A VSPHERE VM ENVIRONMENT (OPTIONAL)

To deploy a working vSphere environment on the deployment host, first prepare it.

```
# yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
# yum install -y python2-pyvmomi
$ git clone -b vmw-3.10 https://github.com/openshift/openshift-ansible-contrib
$ cd openshift-ansible-contrib/reference-architecture/vmware-ansible/
```

Verify that the inventory file has the appropriate variables including IPv4 addresses for the virtual machines in question and logins for the Red Hat Subscription Network. All of the appropriate nodes should be listed in the proper groups: masters, infras, apps.

```
$ cat /etc/ansible/hosts | egrep 'rhsub|ip'
rhsub_user=rhn_username
rhsub_pass=rhn_password
rhsub_pool=8a85f9815e9b371b015e9b501d081d4b
infra-0 vm_name=infra-0 ipv4addr=10.x.y.8
infra-1 vm_name=infra-1 ipv4addr=10.x.y.9
...omitted...
$ ansible-playbook playbooks/provision.yaml
```

If an HAproxy instance is required it can also be deployed.

```
$ ansible-playbook playbooks/haproxy.yaml
```

This will provide the necessary nodes to fulfill Section 2.10, “vSphere VM Instance Requirements for RHOCP”

APPENDIX C. CONFIGURING MASTERS

Edit or create the master configuration file on all masters (***/etc/origin/master/master-config.yaml*** by default) and update the contents of the **apiServerArguments** and **controllerArguments** sections with the following:

```
kubernetesMasterConfig:
  admissionConfig:
    pluginConfig:
      {}
  apiServerArguments:
    cloud-provider:
    - "vsphere"
    cloud-config:
    - "/etc/origin/cloudprovider/vsphere.conf"
  controllerArguments:
    cloud-provider:
    - "vsphere"
    cloud-config:
    - "/etc/origin/cloudprovider/vsphere.conf"
```

CHAPTER 6. CONFIGURING OPENSHIFT NODES FOR CLOUD PROVIDER

1. Edit or create the node configuration file on all nodes (*/etc/origin/node/node-config.yaml* by default) and update the contents of the **kubeletArguments** section:

```
kubeletArguments:
  cloud-provider:
    - "vsphere"
  cloud-config:
    - "/etc/origin/cloudprovider/vsphere.conf"
```

IMPORTANT

For all installations, */etc/origin* and */var/lib/origin* directories are mounted to the master and node container. Therefore, *master-config.yaml* must be in */etc/origin/master* rather than */etc/*.

APPENDIX D. INSTALLING THE VMWARE NSX NSX CONTAINER PLUG-IN (NCP)

OpenShift node VMs must have two vNICs:

- A management vNIC connected to the logical switch that has an uplink to the management tier-1 router.
- The second vNIC on all VMs must have the following tags in NSX-T so that NCP knows which port is used as a parent VIF for all PODs running on the particular OpenShift node.

```
{'ncp/node_name': '<node_name>'}
{'ncp/cluster': '<cluster_name>'}
```

NSX-T Requirement:

- A tier-0 router.
- An overlay transport zone.
- An IP block for POD networking.
- (Optional) An IP Block for routed (no NAT) POD networking.
- An IP Pool for SNAT. By default the IP Block for POD networking is routable only inside NSX-T. NCP uses this IP Pool to provide connectivity to the outside.
- (Optional) Top and bottom firewall sections. NCP will place Kubernetes network policy rules between these two sections.
- Open vSwitch and CNI plugin RPMs must be hosted on an HTTP server reachable from the OpenShift node VMs.

NCP Docker Image

APPENDIX E. DEPLOYING RHOCS STORAGE FOR PERSISTENT STORAGE

To deploy a working vSphere **CNS** environment on the deployment host, first prepare it.

```
$ sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ sudo yum install -y python2-pyvmomi
$ git clone -b vmw-3.10 https://github.com/openshift/openshift-ansible-contrib
$ cd openshift-ansible-contrib/reference-architecture/vmware-ansible/
```

Next, make sure that the appropriate variables are assigned in the inventory file:

```
$ cat /etc/ansible/hosts
rhsub_user=rhn_username
rhsub_pass=rhn_password
rhsub_pool=8a85f9815e9b371b015e9b501d081d4b
[glusterfs]
storage-0  vm_name=storage-0 ipv4addr=10.x.y.33
storage-1  vm_name=storage-1 ipv4addr=10.x.y.34
storage-2  vm_name=storage-2 ipv4addr=10.x.y.35
...omitted...
```

Note the storage group for the **CNS** nodes.

```
$ ansible-playbook playbooks/ocs-cv.yaml
```

During the Red Hat OpenShift Container Platform Installation, the following inventory variables are used to add **Gluster CNS** to the registry for persistent storage:

```
$ cat /etc/ansible/hosts
[OSEv3:children]
...omitted...
nodes
glusterfs
glusterfs_registry

[OSEv3:vars]
...omitted...
# registry
openshift_hosted_registry_storage_kind=glusterfs
openshift_hosted_registry_storage_volume_size=10Gi
openshift_hosted_registry_selector="node-role.kubernetes.io/infra=true"

# logging
openshift_logging_install_logging=true
openshift_logging_es_pvc_dynamic=true
openshift_logging_es_pvc_size=50Gi
openshift_logging_es_cluster_size=3
openshift_logging_es_pvc_storage_class_name='glusterfs-registry-block'
openshift_logging_kibana_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_logging_curator_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra": "true"}

# metrics
```

```
openshift_metrics_install_metrics=true
openshift_metrics_storage_kind=dynamic
openshift_metrics_storage_volume_size=20Gi
openshift_metrics_cassandra_pvc_storage_class_name='glusterfs-registry-block'
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_metrics_cassandra_nodeselector={"node-role.kubernetes.io/infra": "true"}
openshift_metrics_heapster_nodeselector={"node-role.kubernetes.io/infra": "true"}

# Container image to use for glusterfs pods
openshift_storage_glusterfs_image="registry.access.redhat.com/rhgs3/rhgs-server-rhel7:v3.10"

# Container image to use for gluster-block-provisioner pod
openshift_storage_glusterfs_block_image="registry.access.redhat.com/rhgs3/rhgs-gluster-block-
prov-rhel7:v3.10"

# Container image to use for heketi pods
openshift_storage_glusterfs_heketi_image="registry.access.redhat.com/rhgs3/rhgs-volmanager-
rhel7:v3.10"

# OCS storage cluster for applications
openshift_storage_glusterfs_namespace=app-storage
openshift_storage_glusterfs_storageclass=true
openshift_storage_glusterfs_storageclass_default=false
openshift_storage_glusterfs_block_deploy=false

# OCS storage cluster for OpenShift infrastructure
openshift_storage_glusterfs_registry_namespace=infra-storage
openshift_storage_glusterfs_registry_storageclass=false
openshift_storage_glusterfs_registry_block_deploy=true
openshift_storage_glusterfs_registry_block_host_vol_create=true
openshift_storage_glusterfs_registry_block_host_vol_size=200
openshift_storage_glusterfs_registry_block_storageclass=true
openshift_storage_glusterfs_registry_block_storageclass_default=false

[glusterfs]
storage-0.example.com glusterfs_zone=1 glusterfs_devices='[ "/dev/sdd" ]'
storage-1.example.com glusterfs_zone=2 glusterfs_devices='[ "/dev/sdd" ]'
storage-2.example.com glusterfs_zone=3 glusterfs_devices='[ "/dev/sdd" ]'
storage-3.example.com glusterfs_zone=1 glusterfs_devices='[ "/dev/sdd" ]'

[glusterfs_registry]
infra-0.example.com glusterfs_zone=1 glusterfs_devices='[ "/dev/sdd" ]'
infra-1.example.com glusterfs_zone=2 glusterfs_devices='[ "/dev/sdd" ]'
infra-2.example.com glusterfs_zone=3 glusterfs_devices='[ "/dev/sdd" ]'
...omitted...
```

Verify connectivity to the **PVC** for services:

```
[root@master-0 ~]# oc get pvc --all-namespaces
NAMESPACE         NAME                 STATUS    VOLUME                                    CAPACITY
ACCESS MODES   STORAGECLASS              AGE
default           registry-claim           Bound     registry-volume                        25Gi       RWX
31d
logging           logging-es-0            Bound     pvc-22ee4dbf-48bf-11e8-b36e-005056b17236   10Gi
RWO           glusterfs-registry-block   8d
logging           logging-es-1            Bound     pvc-38495e2d-48bf-11e8-b36e-005056b17236   10Gi
```

```
RWO            glusterfs-registry-block   8d
logging            logging-es-2             Bound     pvc-5146dfb8-48bf-11e8-b871-0050568ed4f5   10Gi
RWO            glusterfs-registry-block   8d
mysql              mysql                    Bound     pvc-b8139d85-4735-11e8-b3c3-0050568ede15   10Gi
RWO            app-storage          10d
openshift-metrics   prometheus               Bound     pvc-1b376ff8-489d-11e8-b871-0050568ed4f5
100Gi      RWO            glusterfs-registry-block   8d
openshift-metrics   prometheus-alertbuffer    Bound     pvc-1c9c0ba1-489d-11e8-b871-0050568ed4f5
10Gi       RWO            glusterfs-registry-block   8d
openshift-metrics   prometheus-alertmanager   Bound     pvc-1be3bf3f-489d-11e8-b36e-
005056b17236   10Gi       RWO            glusterfs-registry-block   8d
```

APPENDIX F. HOW TO CONFIGURE THE CLOUD PROVIDER FOR MULTIPLE VCENTER SERVERS

The release of OpenShift 3.9 (Kubernetes 1.9.0) brought support for multiple vCenter Servers are now supported in the vSphere Cloud Provider.

In the previous release of vSphere Cloud Provider (VCP), the configuration file supported a single vCenter Server listing as such:

```
vi /etc/origin/cloudprovider/vsphere.conf

[Global]
user = "administrator@vsphere.local"
password = "password"
server = "vcsa65.example.com"
port = 443
insecure-flag = 1
datacenter = Datacenter
datastore = ose3-vmware-datastore
working-dir = /Datacenter/vm/ocp39/

[Disk]
scsicontrollertype = pvscsi
```

In the new configuration file layout, there are some similiarities to the previous file. Note, that the **Workspace** section shows the endpoint that will be used to create the disk. The **default-datastore** entry should be a shared storage that is accessible to **BOTH** vCenter servers.

In OpenShift 3.11 both formats can be used and work fine.

```
vi /etc/origin/cloudprovider/vsphere.conf

[Global]
user = "administrator@vsphere.local"
password = "password"
port = 443
insecure-flag = 1

[VirtualCenter "vcsa.example.com"]

[VirtualCenter "vcsa-2.example.com"]

[Workspace]
server = vcsa.example.com
default-datastore = ose3-vmware-datastore
folder = /Datacenter/vm/ocp311/
datacenter = "Datacenter"
[Disk]
scsicontrollertype = pvscsi
```

APPENDIX G. TROUBLESHOOTING ANSIBLE BY RED HAT

In the event of a deployment failure, there are a couple of options to use to troubleshoot Ansible.

- Run ansible-playbook with the **-vvv** option.

This can be helpful in determining connection issues or run-time playbook errors.

```
TASK [rhn-subscription : Is the host already registered?] **********************
task path: /opt/ansible/roles/rhn-subscription/tasks/main.yaml:16
Using module file /usr/lib/python2.7/site-packages/ansible/modules/core/commands/command.py
<10.19.114.224> ESTABLISH SSH CONNECTION FOR USER: root
<10.19.114.224> SSH: ansible.cfg set ssh_args: (-C)(-o)(ControlMaster=auto)(-o)
(ControlPersist=900s)(-o)(GSSAPIAuthentication=no)(-o)(PreferredAuthentications=publickey)
<10.19.114.224> SSH: ANSIBLE_HOST_KEY_CHECKING/host_key_checking disabled: (-o)
(StrictHostKeyChecking=no)
<10.19.114.224> SSH:
ANSIBLE_PRIVATE_KEY_FILE/private_key_file/ansible_ssh_private_key_file set: (-o)
(IdentityFile="ssh_key/ocp3-installer")
<10.19.114.224> SSH: ansible_password/ansible_ssh_pass not set: (-o)
(KbdInteractiveAuthentication=no)(-o)(PreferredAuthentications=gssapi-with-mic,gssapi-
keyex,hostbased,publickey)(-o)(PasswordAuthentication=no)
<10.19.114.224> SSH: ANSIBLE_REMOTE_USER/remote_user/ansible_user/user/-u set: (-o)
(User=root)
<10.19.114.224> SSH: ANSIBLE_TIMEOUT/timeout set: (-o)(ConnectTimeout=10)
<10.19.114.224> SSH: PlayContext set ssh_common_args: ()
<10.19.114.224> SSH: PlayContext set ssh_extra_args: ()
<10.19.114.224> SSH: found only ControlPersist; added ControlPath: (-o)(ControlPath=/var/run/%h-
%r)
<10.19.114.224> SSH: EXEC ssh -vvv -C -o ControlMaster=auto -o ControlPersist=900s -o
GSSAPIAuthentication=no -o PreferredAuthentications=publickey -o StrictHostKeyChecking=no -o
'IdentityFile="ssh_key/ocp3-installer"' -o KbdInteractiveAuthentication=no -o
PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o
PasswordAuthentication=no -o User=root -o ConnectTimeout=10 -o ControlPath=/var/run/%h-%r
10.19.114.224 '/bin/sh -c '"'"'/usr/bin/python && sleep 0'"'"''
```

- If there is a failure during the playbook, occasionally the playbooks may be rerun.