Red Hat OpenShift Installation Lab (DO322)

# Chapter 1.  Describing the OpenShift Installation Process

[Introducing OpenShift Installation Methods](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01)

[Quiz: Introducing OpenShift Installation Methods](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s02)

[Running the OpenShift Installer](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s03)

[Quiz: Running the OpenShift Installer](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s04)

[Guided Exercise: Completing the OpenShift Installation Prerequisites](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s05)

[Introducing Hosted OpenShift](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s06)

[Quiz: Introducing Hosted OpenShift](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s07)

[Quiz: Chapter Review: Describing the OpenShift Installation Process](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s08)

[Summary](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch01s09)

**Abstract**

|  |  |
| --- | --- |
| **Goal** | Describe and compare the full-stack automation and pre-existing infrastructure installation methods. |
| **Objectives** | * Describe and compare the full-stack automation and pre-existing infrastructure installation methods. * Describe the OpenShift installer and its configuration files. * Describe the differences between a self-managed OpenShift cluster and hosted OpenShift offerings. |
| **Sections** | * Introducing OpenShift Installation Methods (and Quiz) * Running the OpenShift Installer (and Guided Exercise) (and Quiz) * Introducing Hosted OpenShift (and Quiz) |

## **Introducing OpenShift Installation Methods**

### **Objectives**

* Describe and compare the full-stack automation and pre-existing infrastructure installation methods.

### **Introducing OpenShift Installation**

Because the Red Hat OpenShift Container Platform 3 installation process was quite complex, customers found it difficult to run smoothly. In response to this feedback, Red Hat has redesigned the installer in OpenShift 4 with only those options that are required to create a functional cluster in an opinionated manner.

After the installation completes, administrators can customize and scale it using Day 2 operations.

The opinionated nature of the OpenShift 4 installation process has numerous advantages.

* Automates most of the installation steps to simplify the OpenShift installation process.
* Minimizes human errors during the installation.
* Applies OpenShift 4 best practices.
* Facilitates the integration with future automation efforts such as the OpenShift Assisted Installer, Red Hat Advanced Cluster Management for Kubernetes (ACM), and cluster deployment integration into CI/CD pipelines.

Throughout this course, references to OpenShift refer to Red Hat OpenShift 4.

### NOTE

Opinionated Software is a software product that assumes a certain business process is inherently best and the software is crafted around that approach. The Opinionated Software model helps the product consumer follow the best practices used in the product design.

#### **Describing Ignition Configuration Files**

To deploy an OpenShift Container Platform cluster, administrators must use the openshift-install binary. This command is also known as the OpenShift installer. The main assets generated by the OpenShift installer are the ignition configuration files for the bootstrap, control plane nodes, and compute nodes. With these three ignition configuration files and the underlying infrastructure correctly configured, administrators can start an OpenShift Container Platform cluster installation.

Ignition is a first boot provisioning tool designed to configure RHCOS systems early in the boot process. It uses the ignition configuration file (JSON-formatted .ign file) to declare the desired state of the RHCOS system. The ignition process applies the required configuration during the first boot of the RHCOS system to enforce the desired state on the system. The ignition process:

* Is based on the standard Linux startup process.
* Runs on physical nodes, virtual nodes, and cloud instances.
* Unifies the kickstart and cloud-init features on the RHCOS system boot.
* Is executed in the initramfs step of the RHCOS boot process.
* Configures storage, systemd units, certificates, users, and custom configurations on RHCOS systems when they first boot.
* Consumes the ignition configuration files generated with the openshift-install command and the machine config operator (MCO).
  + The openshift-install command uses the ignition configuration files to set the exact state of each node upon installation.
  + The machine config operator applies other changes to the nodes after installation, such as applying new certificates or ssh keys.

The ignition process loads the ignition file configuration files from one of the following locations:

* Local disk
* Cloud metadata
* Over the network using HTTP or HTTPS

You can use the following ignition configuration file example to configure an RHCOS with ignition:

[user@demo ~]$ **sudo yum install jq**

...output omitted...

[user@demo ~]$ **cat bootstrap.ign | jq .**

{

"ignition": {

**"version": "3.1.0"**

},

"**passwd**": {

"users": [

{

"name": "core",

"**sshAuthorizedKeys": [**

**"ssh-rsa AAA...hlw== lab@utility.lab.example.com\n",**

**"ssh-rsa AAA...3DR\n"**

]

}

]

},

**"storage": {**

"files": [

{

...output omitted...

{

"overwrite": false,

**"path": "/etc/motd",**

"user": {

"name": "root"

},

"append": [

{

"source": "data:text/plain;charset=utf-8;base64,**VGh...lCg==**"

}

],

"mode": 420

},

...output omitted...

**"systemd":** {

"units": [

{

**"contents": "[Unit]\nDescription=Bootstrap a Kubernetes cluster\nRequires=crio-configure.service\nWants=kubelet.service\nAfter=kubelet.service crio-configure.service\nConditionPathExists=!/opt/openshift/.bootkube.done\n\n[Service]\nWorkingDirectory=/opt/openshift\nExecStart=/usr/local/bin/bootkube.sh\n\nRestart=on-failure\nRestartSec=5s\n",**

**"name": "bootkube.service"**

},

...output omitted...

}

]

}

}

|  |  |
| --- | --- |
|  | Ignition version |
|  | Configures SSH public keys for the user core. |
|  | Storage section |
|  | Appends to the /etc/motd file the base64 encoded source data. |
|  | Creates a systemd service unit called bootkube.service that executes the script /usr/local/bin/bootkube.sh. |

For troubleshooting the ignition process, you can use the following commands.

* By default, the data stored in ignition files is encoded in base64. For inspection, you can decode this data in plain text using the base64 -d command.
* [user@demo ~]$ **echo "VGh...lCg==" | base64 -d**
* This is the bootstrap node; it will be destroyed when the master is fully up.
* The primary services are release-image.service followed by bootkube.service. To watch their status, run e.g.

journalctl -b -f -u release-image.service -u bootkube.service

### NOTE

For more information about ignition configuration files examples: <https://coreos.github.io/ignition/examples>

* After the RHCOS system boots, you can check the ignition configuration used to configure the system.
* [root@bootstrap ~]$ **cat /boot/ignition/config.ign**

{"ignition":{"version":"3.1.0"},"passwd":{"users":[{"name":"core","sshAuthorizedKeys":["ssh-rsa...]}}

* Also, you can check the ignition logs.
* [root@bootstrap ~]# **journalctl -t ignition**
* -- Logs begin at Wed 2021-01-27 17:48:36 UTC, end at Mon 2021-02-01 17:52:33 UTC. --
* Jan 27 17:48:38 localhost ignition[684]: **Ignition 2.6.0**
* Jan 27 17:48:38 localhost ignition[684]: Stage: fetch-offline
* Jan 27 17:48:38 localhost ignition[684]: reading system config file "/usr/lib/ignition/base.ign"
* ...output omitted...
* Jan 27 17:48:38 localhost ignition[666]: reading system config file "/usr/lib/ignition/user.ign"
* ...output omitted...
* Jan 27 17:48:44 bootstrap ignition[1176]: INFO : Stage: umount
* Jan 27 17:48:44 bootstrap ignition[1176]: INFO : reading system config file "/usr/lib/ignition/base.ign"
* ...output omitted...
* Jan 27 17:48:44 bootstrap ignition[1176]: INFO : umount: umount passed
* Jan 27 17:48:44 bootstrap ignition[1176]: INFO : **Ignition finished successfully**

...output omitted...

As discussed in this chapter, you do not need to create or modify any ignition configuration files manually for installing OpenShift.

* You can edit the install-config.yaml configuration file to customize the OpenShift installation.
* Then, execute the openshift-installer command to create the Kubernetes manifests from the install-config.yaml configuration file.
* Finally, execute the openshift-installer command again to create the ignition files from the Kubernetes manifests.

### NOTE

Modifying ignition configuration files is only supported if you follow Red Hat documented procedures or Red Hat support instructions. Otherwise, this is not supported.

### NOTE

For more information, refer to the Red Hat Enterprise Linux CoreOS section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/architecture>

#### **Explaining OpenShift Installation Process**

The following sequence of detailed steps explains the OpenShift installation process:

* Step 1: The user runs the OpenShift installer. The installer asks for the necessary cluster information and then creates the installation configuration file install-config.yaml accordingly.
* Step 2: From the install-config.yaml installation configuration file content, the OpenShift installer creates the Kubernetes manifests. The Kubernetes manifests contain the necessary instructions to build the resources for the OpenShift installation.
* Step 3: From the manifests content, the OpenShift installation process creates the ignition configuration files for the bootstrap node bootstrap.ign, control plane nodes master.ign, and compute nodes worker.ign.

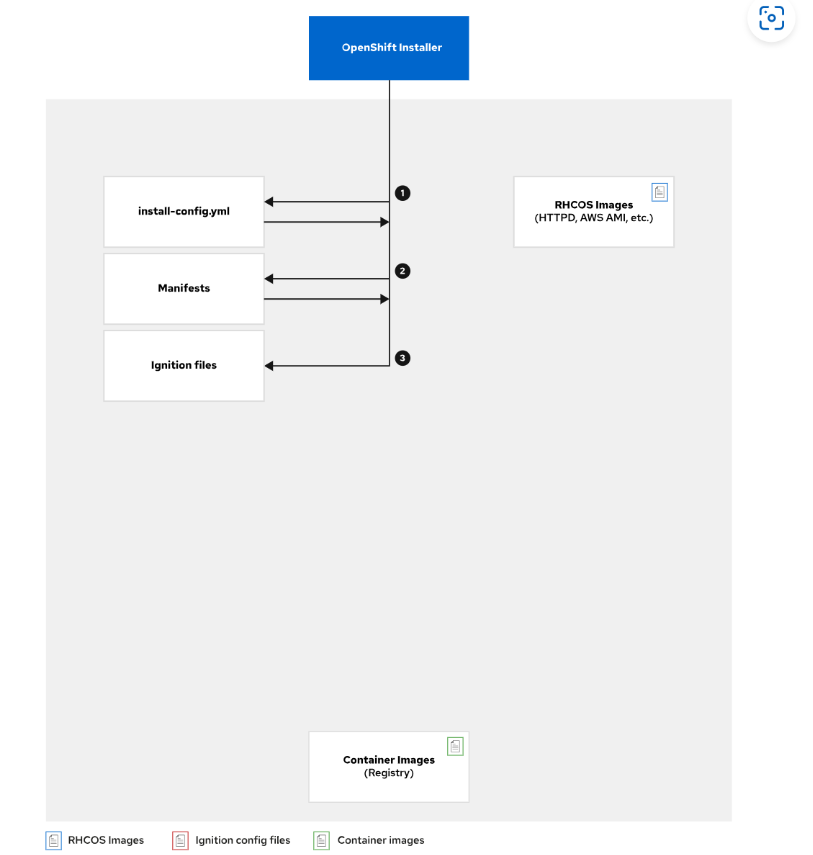


Figure 1.1: OpenShift installation process - Ignition configuration files stage

* Step 4: The bootstrap node boots and fetches its remote resources (bootstrap.ign) from the initial ignition data source, and then finishes booting. At this stage, the Kubernetes API is running on the bootstrap node.

The bootstrap node hosts the remote resources required for control plane nodes to boot (ignition configuration files) in the Machine Configuration Server (MCS). It also runs a single instance of the etcd cluster.

A screenshot of a computer

Description automatically generated

Figure 1.2: OpenShift installation process - Bootstrap (bootkube) stage

### NOTE

In this course, references to the Kubernetes API or the OpenShift API refer to the Kubernetes API Server.

During the OpenShift installation, the Kubernetes API Server runs first on the bootstrap node, and then it moves to the control plane nodes.

* Step 5: The control plane nodes boot and fetch their remote resources (the master.ign ignition configuration file) from the bootstrap node, and then finish booting.
* Step 6: The bootstrap node starts a temporary control plane and installs the etcd operator.

A screenshot of a computer

Description automatically generated

Figure 1.3: OpenShift installation process - Bootstrap (temporary control plane) stage

### NOTE

During the control plane nodes installation, fetching the ignition configuration files happens in two stages: stage-1 and stage-2. At the beginning of the control plane nodes installation (stage-1), the control plane nodes fetch their ignition configuration files (master.ign) from the initial ignition data source.

These ignition configuration files only contain a redirect instruction to get the corresponding ignition files from the Kubernetes API MCS. Finally, the control plane nodes fetch their ignition configuration from the Kubernetes API MCS (stage-2) and finish the installation.

* Step 7: The etcd operator running on the bootstrap node scales up the etcd cluster to 3 instances using two control plane nodes.
* Step 8: The temporary control plane running on the bootstrap node schedules the production control plane to the control plane nodes. The OpenShift installation process transfers the etcd cluster to the control plane nodes.

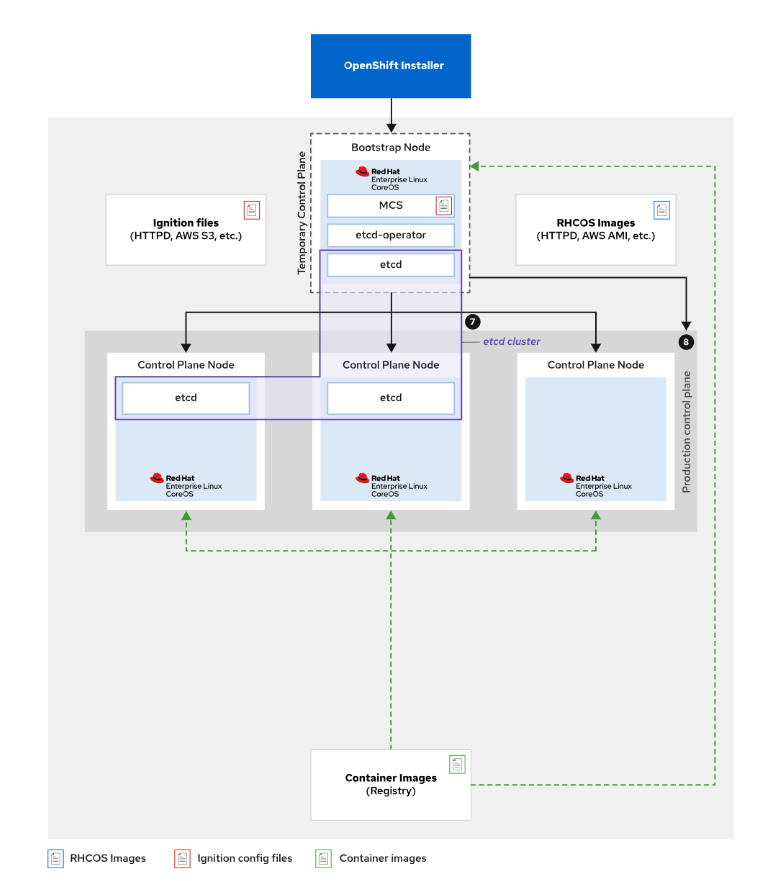


Figure 1.4: OpenShift installation process - Production control plane schedule stage

### NOTE

The temporary control plane is used only during the OpenShift installation. The OpenShift installation process transfers the temporary control plane to the production control plane running on the control plane nodes.

The production control plane is the definitive control plane that manages the OpenShift cluster.

* Step 9: The temporary control plane shuts down, yielding to the production control plane. At this stage, the Kubernetes API is running on the production control plane.
* Step 10: For full-stack automation installations, the installer shuts down the bootstrap node. Since this stage, the bootstrap node is no longer needed.

A screenshot of a computer

Description automatically generated

Figure 1.5: OpenShift installation process - Production control plane stage

### NOTE

The etcd cluster runs an etcd pod on each control plane node.

The Kubernetes API MCS service runs a Machine Config Server pod on each control plane node. The Kubernetes API MCS service hosts the master.ign and worker.ign ignition files.

* Step 11: At this stage, the production control plane hosts the cluster remote resources (ignition configuration files) for control plane nodes and compute nodes in their MCS. The compute nodes boot and fetch their remote resources (the worker.ign ignition configuration file) from the control plane nodes, finish booting, and join the cluster.

If the pre-existing infrastructure installation method is used, the OpenShift installation process can also install the compute nodes with the RHEL 7 operating system instead of using the default RHCOS operating system.

Support for using RHEL 7 compute nodes is deprecated and will be removed in a future release of OpenShift 4.

A screenshot of a computer

Description automatically generated

Figure 1.6: OpenShift installation process - Compute nodes installation stage

Regarding OpenShift installations:

* Red Hat only supports the use of three control plane nodes.
* Red Hat has tested a maximum of 2000 compute nodes on a Red Hat OpenShift Container Platform 4.6 cluster.
* Red Hat recommends the use of at least two compute nodes to ensure the high availability of the applications running on the cluster.
* For more information, refer to the Planning your environment according to object maximums section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/scalability_and_performance>

### NOTE

During the compute nodes installation, fetching the ignition configuration files happens in two stages: stage-1 and stage-2. At the beginning of compute nodes installation (stage-1), the compute nodes fetch their ignition configuration files (worker.ign) from the initial ignition data source.

These ignition configuration files only contain a redirect instruction to get the corresponding ignition files from the Kubernetes API MCS. Finally, the cluster nodes fetch their ignition configuration from the Kubernetes API MCS (stage-2) and finish the installation.

### **Describing OpenShift Installation Methods**

The OpenShift installer offers the following installation methods:

* **Full-stack Automation**

Using this installation method, administrators install OpenShift with minimal manual intervention in an opinionated manner.

The OpenShift installer deploys the cluster on infrastructure that the installer provisions and the cluster maintains.

* **Pre-existing Infrastructure**

Using this installation method, administrators have more flexibility when installing OpenShift than they do when using the full-stack automation method.

Administrators use the installer to deploy a cluster on infrastructure that they prepare and maintain themselves.

#### **Full-stack Automation Installation Method**

Using this installation method:

* Administrators can use the OpenShift installer to deploy an OpenShift cluster on infrastructure that the OpenShift installer provisions and the OpenShift cluster maintains.
* The OpenShift installer controls all installation areas, including infrastructure provisioning, with an opinionated best practices deployment of OpenShift.
* This installation method is frequently called installer-provisioned infrastructure (IPI).

The following diagram describes the OpenShift installation workflow for the full-stack automation installation method.

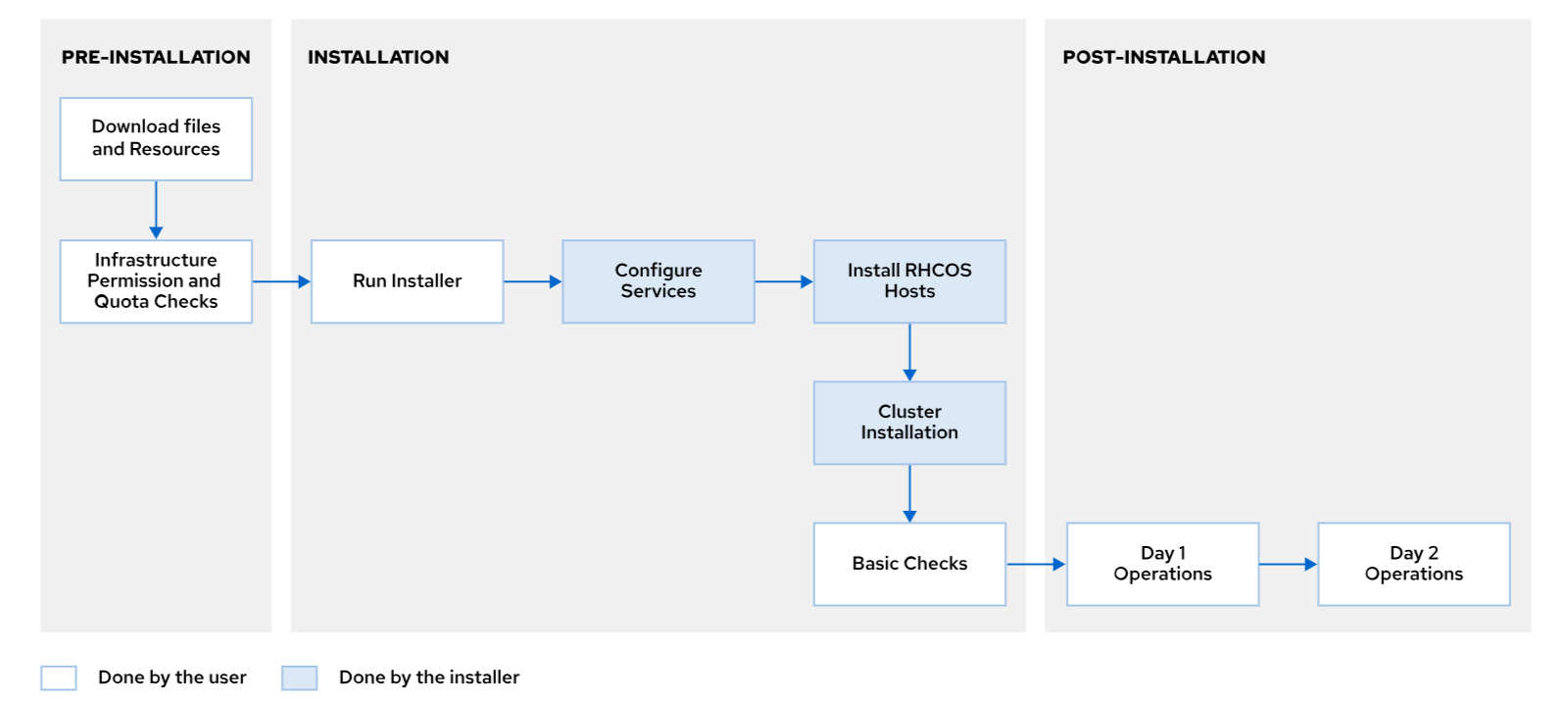


Figure 1.7: OpenShift installation workflow for full-stack automation method

#### **Pre-existing Infrastructure Installation Method**

Using this installation method:

* Administrators can use the OpenShift installer to deploy a cluster on infrastructure that they prepare and maintain themselves.
* Administrators are responsible for creating and managing their infrastructure, allowing greater infrastructure customization and operational flexibility.
* This installation method is frequently called user-provisioned infrastructure (UPI).

The following diagram describes the OpenShift installation workflow for the pre-existing infrastructure installation method.

A diagram of a process

Description automatically generated

Figure 1.8: OpenShift installation workflow for pre-existing infrastructure method

### NOTE

It is common to see the term installer-provisioned clusters when referring to clusters installed with the full-stack automation method, and the term user-provisioned clusters for clusters installed with the pre-existing infrastructure installation method.

### **Comparing Full-stack Automation and Pre-existing Infrastructure Installation Methods**

The following table shows the basic differences between full-stack automation and pre-existing infrastructure installation methods.

**Table 1.1. Full-stack Automation & Pre-existing Infrastructure Comparison**

| **Action** | **Full-stack Automation** | **Pre-existing Infrastructure** |
| --- | --- | --- |
| Build network | Installer | User |
| Setup load balancer | Installer | User |
| Configure DNS | Installer | User |
| Hardware or VM provisioning | Installer | User |
| OS installation | Installer | User |
| Generate ignition configs | Installer | Installer |
| Control plane node OS support | Installer: RHCOS | User: RHCOS |
| Compute node OS support | Installer: RHCOS (1) | User: RHCOS + RHEL 7 (1) |
| Configure persistent storage for the internal registry | Installer (2) | User |
| Configure dynamic storage provider | Installer (2) | User |
| Configure node provisioning and cluster autoscaling | Installer | Only for providers with OpenShift Machine API support (3) |

* (1) Full-stack automation and pre-existing infrastructure installation methods support adding RHEL 7 compute nodes deployed by the user as a Day 2 operation.

Support for using RHEL 7 compute nodes is deprecated and will be removed in a future release of OpenShift 4. Only RHCOS cluster nodes will be supported in future OpenShift versions. For more information, refer to the Deprecated features section at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/release_notes>

* (2) Except in full-stack automation, installations on bare metal.
* (3) Notice the distinction between a pre-existing infrastructure installation with cloud integration (UPI CI) and a pre-existing infrastructure installation on bare metal (UPI BM).
  + For UPI CI installations, if the infrastructure provider has integration with OpenShift Machine API, then the installer configures node provisioning and cluster autoscaling.
  + For UPI BM installations, where the infrastructure provider has no OpenShift Machine API integration support, the installer does not configure node provisioning and cluster autoscaling.

### **Describing OpenShift Installation Prerequisites**

Before installing OpenShift, administrators must ensure that the infrastructure environment used for the installation meets the required prerequisites. The OpenShift installation prerequisites include general prerequisites that apply to both installation methods, and prerequisites specific to the selected installation method.

#### **General Prerequisites**

* Provision a bastion host.
* Generate an SSH key on the bastion host.
* Download and install openshift-install on the bastion host.
* Download and install oc on the bastion host.
* Get the registry pull-secret.
* OpenShift cluster nodes must have access to a Network Time Protocol (NTP) server.
* Ensure that the infrastructure network firewall fulfills the OpenShift network access requirements.

#### **Full-stack Automation Installation Prerequisites**

* Verify infrastructure permissions and quotas.

For cloud environments, a cloud account with required permissions and quotas is required.

#### **Pre-existing Infrastructure Installation Prerequisites**

* Configure network services.
* Provide hardware (physical or virtual) for cluster nodes.
* Install RHCOS on cluster nodes.

### **Describing OpenShift DNS Prerequisites**

When installing OpenShift using the pre-existing infrastructure method, one of the network services that administrators must configure is the DNS service. Administrators must configure the DNS service using DNS records. A complete DNS record takes the form <component>.<cluster\_name>.<base\_domain>:

* <cluster\_name> is the name of the cluster (for instance ocp4).
* <base\_domain> is the cluster base domain configured in the install-config.yaml configuration file (for instance example.com).
* The reverse DNS records (PTR) are also required.

The following table describes the DNS records required for installing OpenShift:

**Table 1.2. DNS Records Required for Installing OpenShift**

| **Component** | **Record** | **Description** |
| --- | --- | --- |
| Kubernetes API | api.<cluster\_name>.<base\_domain>. | DNS A/AAAA or CNAME and PTR records to identify the API load balancer for the control plane nodes (resolvable by both clients external to the cluster and from all the nodes within the cluster). |
| Kubernetes API | api-int.<cluster\_name>.<base\_domain>. | DNS A/AAAA or CNAME and PTR records to identify the API load balancer for the control plane nodes (resolvable from all the nodes within the cluster). |
| Routes | \*.apps.<cluster\_name>.<base\_domain>. | Wildcard DNS A/AAAA or CNAME record to identify the Application Ingress load balancer that targets the cluster nodes that run the ingress router pod (resolvable by both clients external to the cluster and from all the nodes within the cluster). |
| Cluster nodes | <name>.<cluster\_name>.<base\_domain>. | DNS A/AAAA or CNAME and PTR records to identify each cluster node, including bootstrap node (resolvable by the nodes within the cluster). |

The following BIND configuration shows the DNS configuration used for installing an OpenShift cluster named ocp4 in the example.com base domain:

[root@utility ~]# **cat /etc/named.conf**

...output omitted...

zone "**example.com**" {

type master;

file "example.com.db";

allow-update { none; };

};

...output omitted...

[root@utility ~]# **cat /var/named/example.com.db**

**$TTL 1D**

@ IN SOA dns.ocp4.example.com. root.example.com. (

2019022400 ; serial

3h ; refresh

15 ; retry

1w ; expire

3h ; minimum

)

IN NS dns.ocp4.example.com.

dns.ocp4 IN A 192.168.50.254

api.ocp4 IN A 192.168.50.254

api-int.ocp4 IN A 192.168.50.254

\*.apps.ocp4 IN A 192.168.50.254

bootstrap.ocp4 IN A 192.168.50.9

master01.ocp4 IN A 192.168.50.10

master02.ocp4 IN A 192.168.50.11

master03.ocp4 IN A 192.168.50.12

worker01.ocp4 IN A 192.168.50.13

worker02.ocp4 IN A 192.168.50.14

[root@utility ~]# **cat /etc/named.conf**

...output omitted...

zone "**50.168.192.in-addr.arpa**" IN {

type master;

file "example.com.reverse.db";

allow-update { none; };

};

...output omitted...

[root@utility ~]# **cat /var/named/example.com.reverse.db**

**$TTL 1D**

@ IN SOA dns.ocp4.example.com. root.example.com. (

2019022400 ; serial

3h ; refresh

15 ; retry

1w ; expire

3h ; minimum

)

IN NS dns.ocp4.example.com.

254 IN PTR api.ocp4.example.com.

254 IN PTR api-int.ocp4.example.com.

9 IN PTR bootstrap.ocp4.example.com.

10 IN PTR master01.ocp4.example.com.

11 IN PTR master02.ocp4.example.com.

12 IN PTR master03.ocp4.example.com.

13 IN PTR worker01.ocp4.example.com.

14 IN PTR worker02.ocp4.example.com.

You can use the dig command to verify the DNS service configuration before installing OpenShift.

[lab@utility ~]$ **dig @dns.ocp4.example.com api.ocp4.example.com**

...output omitted...

;; ANSWER SECTION:

api.ocp4.example.com. 86400 IN A **192.168.50.254**

...output omitted...

[lab@utility ~]$ **dig @dns.ocp4.example.com api-int.ocp4.example.com**

...output omitted...

;; ANSWER SECTION:

api-int.ocp4.example.com. 86400 IN A **192.168.50.254**

...output omitted...

[lab@utility ~]$ **dig @dns.ocp4.example.com -x 192.168.50.254**

...output omitted...

;; ANSWER SECTION:

254.50.168.192.in-addr.arpa. 86400 IN PTR **api.ocp4.example.com.**

254.50.168.192.in-addr.arpa. 86400 IN PTR **api-int.ocp4.example.com.**

...output omitted...

[lab@utility ~]$ **dig @dns.ocp4.example.com master01.ocp4.example.com**

...output omitted...

;; ANSWER SECTION:

master01.ocp4.example.com. 86400 IN A **192.168.50.10**

...output omitted...

[lab@utility ~]$ **dig @dns.ocp4.example.com -x 192.168.50.10**

...output omitted...

;; ANSWER SECTION:

10.50.168.192.in-addr.arpa. 86400 IN PTR **master01.ocp4.example.com.**

...output omitted...

Repeat this procedure for all the cluster nodes, including the bootstrap node.

### NOTE

By default, when installing OpenShift using the full-stack automation method on supported cloud providers, the OpenShift installer automatically performs the DNS service configuration.

Administrators must configure the DNS service when installing OpenShift using the full-stack automation method on the following infrastructures:

* Red Hat OpenStack Platform cloud provider using an external DNS
* VMware vSphere
* Bare metal

### **Describing OpenShift Firewall Prerequisites**

As explained in the General Prerequisites section, you must ensure that your infrastructure network firewall configuration meets the OpenShift network access requirements before installing OpenShift. For connected installations, the OpenShift cluster needs access to the remote sites it requires to function. For more information, refer to the Configuring your firewall section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing>

Additionally, the OpenShift cluster nodes must have some network ports accessible on the cluster network. For more information, refer to the Installing on bare metal chapter of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing_on_bare_metal>

When using a network firewall, you must ensure network connectivity between the cluster nodes. Also, each cluster node must be able to resolve the host names of all other nodes in the cluster.

**Table 1.3. Connectivity Requirements from All Cluster Nodes to All Cluster Nodes**

| **Protocol** | **Port** | **Description** |
| --- | --- | --- |
| ICMP | N/A | Network reachability tests |
| TCP | 9000-9999 | Host level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099 |
| TCP | 10250-10259 | The default ports that Kubernetes reserves |
| TCP | 10256 | openshift-sdn |
| UDP | 4789 | VXLAN and Geneve |
| UDP | 6081 | VXLAN and Geneve |
| UDP | 9000-9999 | Host level services, including the node exporter on ports 9100-9101 |
| TCP/UDP | 30000-32767 | Kubernetes node port |

**Table 1.4. Connectivity Requirements from All Cluster Nodes to Control Plane Nodes**

| **Protocol** | **Port** | **Description** |
| --- | --- | --- |
| TCP | 2379-2380 | etcd server, peer, and metrics ports |
| TCP | 6443 | Kubernetes API |

Ensure that the load balancers used by the OpenShift cluster have both the front (load balancer) and back (cluster nodes) network ports accessible.

The API Load Balancer (API LB) provides a common endpoint to interact with and configure the platform. The Application Ingress Load Balancer (APP Ingress LB) provides an ingress point for application traffic flowing in from outside the cluster.

**Table 1.5. API Load Balancer Ports**

| **TCP Port** | **Back-end nodes (pool members)** | **Internal access** | **External access** | **Description** |
| --- | --- | --- | --- | --- |
| 6443 | Bootstrap (temporary) and control plane nodes | Yes | Yes | Kubernetes API server |
| 22623 | Bootstrap (temporary) and control plane nodes | Yes | No | Machine Config server |

**Table 1.6. Application Ingress Load Balancer Ports**

| **TCP Port** | **Back-end nodes (pool members)** | **Internal access** | **External access** | **Description** |
| --- | --- | --- | --- | --- |
| 443 | Cluster nodes that run the Ingress router pods | Yes | Yes | HTTPS traffic |
| 80 | Cluster nodes that run the Ingress router pods | Yes | Yes | HTTP traffic |

### NOTE

The firewall configuration explained in this section assumes the following:

* You are installing an OpenShift cluster using the pre-existing infrastructure installation method.
* The network infrastructure used to install the cluster has network restrictions managed by a network firewall.

When installing an OpenShift cluster with the full-stack automation in a supported cloud provider:

* By default, the installer creates a new virtual private network (VPC) for the cluster with all the network access requirements in place.
* If you want to install the cluster on an existing virtual private network (VPC), then you must verify that the existing VPC fulfills all the network access requirements.

### **Describing OpenShift Installation Modes**

Depending on the external connectivity of the cluster, administrators can use one of the following installation modes:

* **Connected**

The cluster nodes have internet access to pull container images from the quay.io and registry.redhat.io registries. The connected installation mode is supported when using the full-stack automation or the pre-existing infrastructure installation methods.

* **Disconnected**

Administrators use this installation mode when a connected installation is not possible. The OpenShift installer uses a local container registry to pull container images. Not all cloud providers support installation in disconnected mode.

#### **Introducing Disconnected Installations**

Administrators can install an OpenShift cluster on infrastructure that they provide in a restricted network. This installation mode is also known as the disconnected installation mode. The disconnected installation:

* Requires mirroring images to an API schema2 specification-compliant local container registry.
* Uses the exact version of images provided in the payload by digest.
* Requires access to the Internet to mirror the container images from the source registries to the local container registry.
* Requires mirroring the images required for the OpenShift installation and subsequent product updates to the local container registry.

Before starting a disconnected installation, you must mirror the required images to the local container registry and obtain the imageContentSources data for your OpenShift version. Mirror the following content in the local container registry:

* The OpenShift Container Platform image repository

This repository provides the container images to use during OpenShift cluster installation or upgrade.

* Cluster samples operator image streams

Because most image streams in the openshift namespace use images located in the Red Hat registry registry.redhat.io, you must mirror those images and configure the image streams accordingly.

* Remote OperatorHub sources
* Remote Operator Hub sources

In disconnected clusters, the operator lifecycle manager (OLM) cannot access the Red Hat provided Operator Hub sources hosted on quay.io. Administrators must mirror the Operator Hub sources in the local container registry and configure OLM to install and manage the operators from the local sources instead of the default remote sources.

Instructions to mirror the OpenShift Container Platform image repository and the cluster samples operator image streams in the local registry are available in the Creating a mirror registry for installation in a restricted network section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing>.

Instructions to mirror the remote Operator Hub sources in the local registry are available in the Using Operator Lifecycle Manager on restricted networks section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/operators/index#olm-restricted-networks>.

OpenShift clusters in restricted networks have the following limitations:

* The ClusterVersion status shows an Unable to retrieve available updates error message.
* By default, the Developer Catalog contents are unavailable because the cluster does not have access to the required image stream tags.
* The OpenShift Telemetry service is disabled.

#### **Customizing Disconnected Installations**

After mirroring the local container registry, start the OpenShift installation process. You must modify the install-config.yaml file to customize the disconnected OpenShift installation.

The following is a sample install-config.yaml file used in a connected installation:

[user@demo ~]$ **cat ${HOME}/ocp4-cluster/install-config.yaml**

apiVersion: v1

baseDomain: example.com

#proxy:

# httpProxy: http://<username>:<pswd>@<ip>:<port>

# httpsProxy: http://<username>:<pswd>@<ip>:<port>

# noProxy: example.com

compute:

- hyperthreading: Enabled

name: worker

replicas: 2

controlPlane:

hyperthreading: Enabled

name: master

replicas: 3

metadata:

name: ocp4

networking:

clusterNetwork:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: |

{"auths":...output omitted...}

sshKey: |

ssh-rsa AA...output omitted...

|  |  |
| --- | --- |
|  | If required, you can use a network proxy in a connected installation. |
|  | The pull-secret contains the credentials to authenticate against quay.io and registry.redhat.io. |

The following is a sample install-config.yaml file used in a disconnected installation:

[user@demo ~]$ **cat ${HOME}/ocp4-cluster/install-config.yaml**

apiVersion: v1

baseDomain: example.com

compute:

- hyperthreading: Enabled

name: worker

replicas: 2

controlPlane:

hyperthreading: Enabled

name: master

replicas: 3

metadata:

name: ocp4

networking:

clusterNetwork:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: |

{"auths":...output omitted...}

sshKey: |

ssh-rsa AA...output omitted...

**additionalTrustBundle: |**

**-----BEGIN CERTIFICATE-----**

**ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ**

**-----END CERTIFICATE-----**

**imageContentSources:**

**- mirrors:**

**- nexus-registry-int.apps.tools.dev.nextcle.com/openshift/ocp4**

**source: quay.io/openshift-release-dev/ocp-release**

**- mirrors:**

**- nexus-registry-int.apps.tools.dev.nextcle.com/openshift/ocp4**

**source: quay.io/openshift-release-dev/ocp-v4.0-art-dev**

|  |  |
| --- | --- |
|  | The pull-secret contains the credentials to authenticate against the nexus-registry-int.apps.tools.dev.nextcle.com local container registry. |
|  | If the certificate used by the local container registry is not trusted, provide the certificate bundle that you used for the local container registry. |
|  | Add the imageContentSources section from the output of the command used to mirror the repository. This customization will configure the file /etc/containers/registries.conf on each cluster node to enable image mirroring. |

### NOTE

The classroom environment used to perform the guided exercises for this course has full internet connectivity. There is a local registry with a mirror of the OpenShift release images and other resources in the classroom environment.

### REFERENCES

For more information, refer to the Installation and update chapter of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/architecture>

## **Running the OpenShift Installer**

### **Objectives**

* Describe the OpenShift installer and its configuration files.

### **Using OpenShift Installation Binary**

Regardless of the installation method, you must use the OpenShift Installer binary openshift-install to start an OpenShift installation. This installer ensures simple, uniform, and opinionated behavior during the OpenShift installation process.

### NOTE

Starting with OpenShift Container Platform 4.6, a full-stack installation on bare metal uses the OpenShift installer binary openshift-baremetal-install.

The following table shows some useful options available to the openshift-install command:

**Table 1.7.  openshift-install<command options>**

| **Option** | **Description** |
| --- | --- |
| help | Prints help information. |
| explain -h | Explains the fields related to each supported InstallConfig API. |
| explain installconfig | Explains the installconfig resource and its fields. |
| --log-level debug | Enables debug mode. |
| --dir | Configures the directory path to store the generated assets. |
| create install-config | Creates installation configuration file install-config.yaml. |
| create manifests | Creates Kubernetes manifests. |
| create ignition-configs | Creates ignition configuration files. |
| create cluster | Deploys the cluster. |
| wait-for bootstrap-complete | Waits until the bootstrap installation phase ends. |
| wait-for install-complete | Waits until the cluster install ends. |
| destroy cluster | Removes the cluster |

The following sequence shows the high-level steps required to install OpenShift:

1. Fulfill the installation prerequisites.
2. Create the installation directory.
3. Create the installer configuration file, install-config.yaml.
4. Generate the Kubernetes manifests.
5. Generate the ignition configuration files.
6. Deploy the OpenShift cluster.
7. Verify the OpenShift cluster health.

#### **Creating Installation Directory**

To start installing OpenShift, first create the installation directory. This directory stores all the files created by the OpenShift installer.

[user@demo ~]$ **mkdir ${HOME}/ocp4-cluster**

### NOTE

The OpenShift installer will create the installation directory if it does not exist.

#### **Creating Installer Configuration File**

After you create the installation directory, run the OpenShift installer to create the installer configuration file install-config.yaml. The installer prompts you for the necessary cluster information and then creates the install-config.yaml file accordingly.

[user@demo ~]$ **openshift-install create install-config \**

> **--dir=${HOME}/ocp4-cluster**

? SSH Public Key **/home/user/.ssh/ocp4-cluster.pub**

? Platform **aws**

INFO Credentials loaded from the "default" profile in file "/home/user/.aws/credentials"

? Region **us-east-2**

? Base Domain **mydomain.com**

? Cluster Name **ocp4**

? Pull Secret [? for help] **+++++**

INFO Install-Config created in: /home/user/ocp4-cluster

### NOTE

This example demonstrates the creation of the install-config.yaml configuration file for an OpenShift installation on AWS.

The following is a sample install-config.yaml configuration file:

[user@demo ~]$ **cat ${HOME}/ocp4-cluster/install-config.yaml**

apiVersion: v1

baseDomain: **mydomain.com**

compute:

- architecture: amd64

hyperthreading: Enabled

name: worker

platform: {}

replicas: 3

controlPlane:

architecture: amd64

hyperthreading: Enabled

name: master

platform: {}

replicas: 3

metadata:

creationTimestamp: null

name: **ocp4**

networking:

clusterNetwork:

- cidr: 10.128.0.0/14

hostPrefix: 23

machineNetwork:

- cidr: 10.0.0.0/16

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

aws:

region: **us-east-2**

publish: External

pullSecret: |

**{"auths":...}**

sshKey: |

**ssh-rsa AA...**

### NOTE

Before creating the Kubernetes manifests, you can edit the install-config.yaml configuration file to customize the OpenShift installation. The most common installation customizations are discussed elsewhere in this course.

#### **Generating Kubernetes Manifests**

After creating the install-config.yaml file, run the OpenShift installer to generate the Kubernetes manifests. A Kubernetes manifest is a file that describes one or more Kubernetes API objects, such as Pods, Services, Deployments or MachineConfigs. The Kubernetes manifests contain the instructions necessary to build the resources for the OpenShift installation.

[user@demo ~]$ **openshift-install create manifests \**

> **--dir=${HOME}/ocp4-cluster**

INFO Consuming Install Config from target directory

INFO Manifests created in: /home/user/ocp4-cluster/manifests and /home/user/ocp4-cluster/openshift

You can review the Kubernetes manifests generated by the OpenShift installer in the installation directory.

[user@demo ~]$ **find ${HOME}/ocp4-cluster/manifests**

/home/user/ocp4-cluster/manifests

/home/user/ocp4-cluster/manifests/04-openshift-machine-config-operator.yaml

...output omitted...

The following is an example of an official procedure from Red Hat that explains how to create a new Kubernetes manifest to add the loglevel=7 kernel argument to the control plane nodes during the installation. In the ${HOME}/ocp4-cluster/manifests/openshift directory, create a file called 99-openshift-machineconfig-master-kargs.yaml that defines a MachineConfig object for adding the kernel setting.

[user@demo ~]$ **cd ${HOME}/ocp4-cluster/manifests/openshift**

[user@demo ~]$ **cat << EOF > 99-openshift-machineconfig-master-kargs.yaml**

apiVersion: machineconfiguration.openshift.io/v1

kind: MachineConfig

metadata:

labels:

machineconfiguration.openshift.io/role: master

name: 99-openshift-machineconfig-master-kargs

spec:

kernelArguments:

- **'loglevel=7'**

EOF

When creating the ignition files from the manifests, this new manifest is wrapped and added to the control plane nodes ignition file master.ign.

### WARNING

Modifying Kubernetes manifests is only supported if you follow Red Hat documented procedures or Red Hat support instructions. Otherwise, this is not supported.

#### **Generating Ignition Configuration Files**

After you generate the Kubernetes manifests, run the OpenShift installer to create the ignition configuration files from the manifests content. The OpenShift installer creates the bootstrap, control plane nodes, and the compute nodes ignition configuration files. When deploying the OpenShift cluster, the OpenShift installer uses these ignition configuration files to install and configure RHCOS on the bootstrap node, control plane nodes, and compute nodes.

[user@demo ~]$ **openshift-install create ignition-configs \**

> **--dir=${HOME}/ocp4-cluster**

INFO Consuming Master Machines from target directory

INFO Consuming OpenShift Install (Manifests) from target directory

INFO Consuming Openshift Manifests from target directory

INFO Consuming Worker Machines from target directory

INFO Consuming Common Manifests from target directory

INFO Ignition-Configs created in: /home/user/ocp4-cluster and /home/user/ocp4-cluster/auth

You can review the ignition configuration files generated by the OpenShift installer in the installation directory.

[user@demo ~]$ **find ${HOME}/ocp4-cluster -name '\*.ign' | xargs ls -lrt**

-rw-r-----. 1 user user 1732 Dec 27 19:35 /home/user/ocp4-cluster/master.ign

-rw-r-----. 1 user user 1732 Dec 27 19:35 /home/user/ocp4-cluster/worker.ign

-rw-r-----. 1 user user 307594 Dec 27 19:35 /home/user/ocp4-cluster/bootstrap.ign

### NOTE

Ignition files are valid for 24 hours, after which the included certificates expire. If the cluster installation fails, check the ignition files to see if they are more than 24 hours old. If so, create new ignition files.

### WARNING

Modifying ignition configuration files is only supported if you follow Red Hat documented procedures or Red Hat support instructions. Otherwise, this is not supported.

#### **Deploying OpenShift Cluster**

After you generate the ignition configuration files, run the OpenShift installer to deploy the OpenShift cluster.

[user@demo ~]$ **openshift-install create cluster \**

> **--dir=${HOME}/ocp4-cluster --log-level=debug**

DEBUG OpenShift Installer 4.6.4

DEBUG Built from commit 6e02d049701437fa81521fe981405745a62c86c5

...output omitted...

**INFO Consuming Bootstrap Ignition Config from target directory**

INFO Consuming Worker Ignition Config from target directory

INFO Consuming Master Ignition Config from target directory

...output omitted...

DEBUG Apply complete! Resources: 122 added, 0 changed, 0 destroyed.

DEBUG OpenShift Installer 4.6.4

DEBUG Built from commit 6e02d049701437fa81521fe981405745a62c86c5

**INFO Waiting up to 20m0s for the Kubernetes API at https://api.ocp4.example.com:6443...**

**INFO API v1.19.0+9f84db3 up**

**INFO Waiting up to 30m0s for bootstrapping to complete...**

...output omitted...

DEBUG Bootstrap status: complete

INFO Destroying the bootstrap resources...

...output omitted...

**INFO Waiting up to 40m0s for the cluster at https://api.ocp4.example.com:6443 to initialize...**

DEBUG Still waiting for the cluster to initialize: Working towards 4.6.4: 82% complete

...output omitted...

**INFO Install complete!**

INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home/user/ocp4-aws-cluster/auth/kubeconfig'

INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4.example.com

INFO Login to the console with user: "kubeadmin", and password: "xxxx"

...output omitted...

INFO Time elapsed: 34m10s

In a pre-existing infrastructure installation, you cannot use the openshift-install create cluster command to deploy the cluster because you have already installed the cluster nodes. The bootstrap node installation triggers the cluster installation, so execute the following command sequence to monitor the cluster installation:

[user@demo ~]$ **openshift-install wait-for bootstrap-complete \**

> **--dir=${HOME}/ocp4-cluster --log-level=debug**

[user@demo ~]$ **openshift-install wait-for install-complete \**

> **--dir=${HOME}/ocp4-cluster --log-level=debug**

### NOTE

The openshift-install wait-for commands do not trigger the cluster installation. It is a recommended practice to use them to monitor the cluster installation.

### **Monitoring OpenShift Installations**

In addition to reviewing the installation logs, you can perform active installation process monitoring using the oc command. You can start using the oc command once the bootstrap has the Kubernetes API running on the temporary control plane.

To start using the oc command, authenticate against the Kubernetes API with cluster-admin privileges using one of the following methods:

* Configuring the KUBECONFIG environment variable to use the kubeconfig file.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

* Using the kubeadmin credentials stored in the kubeadmin-password file.

### WARNING

Save the kubeconfig and kubeadmin-password files in a secure location. These files grant cluster-admin privileges in the cluster. Handle them accordingly.

After logging in to the Kubernetes API as cluster-admin, use the following sequence of oc commands to monitor the installation process:

[user@demo ~]$ **watch 'oc get clusterversion; oc get clusteroperators; \**

> **oc get pods --all-namespaces | grep -v -E "Running|Completed"; oc get nodes'**

...output omitted...

Using the watch command, you obtain a real-time view of how the cluster version operator (CVO) installs the cluster operators, which pods are not yet running on the cluster, and the status of cluster nodes. The watch command automatically runs the oc command sequence every 2 seconds, so it is useful for monitoring the progression of the cluster installation.

[user@demo ~]$ **watch 'oc get clusterversion; oc get clusteroperators; \**

> **oc get pods --all-namespaces | grep -v -E "Running|Completed"; oc get nodes'**

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version False **True** 13m **Working towards 4.6.4: 98% complete**

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication 4.6.4 False False True 7m59s

cloud-credential True False False 11m

cluster-autoscaler 4.6.4 True False False 6m34s

config-operator 4.6.4 True False False 8m4s

console 4.6.4 False True False 60s

...output omitted...

NAMESPACE NAME READY STATUS RESTARTS AGE

openshift-apiserver apiserver-5c4b794d87 0/2 Pending 0 2m13s

openshift-kube-apiserver kube-apiserver-ip-10 0/5 Init:0/1 0 61s

...output omitted...

NAME STATUS ROLES AGE VERSION

master01 Ready master 7m19s v1.19.0+9f84db3

master02 Ready master 7m18s v1.19.0+9f84db3

master03 Ready master 7m19s v1.19.0+9f84db3

The oc get events command prints the OpenShift event messages in real-time, so it is useful monitoring installation progress and background events in detail.

[user@demo ~]$ **oc get events -A -w**

...output omitted...

openshift-infra 0s Warning NetworkNotReady

### **Troubleshooting OpenShift Installations**

The OpenShift cluster deployment process has three main stages:

* **Bootstrap (Bootkube) Stage**

The OpenShift installation process deploys the bootstrap node. Next, the release-image systemd service running on the bootstrap node downloads the container images required to start the temporary control plane. Finally, bootkube systemd service running on the bootstrap node starts the temporary control plane.

* **Bootstrap (Temporary Control Plane) Stage**

The Kubernetes API and the temporary control plane are running on the bootstrap node. Next, the bootstrap node waits until the control plane nodes boot and form an etcd cluster. Finally, the bootstrap node schedules the production control plane to the control plane nodes.

* **Production Control Plane stage**

After the production control plane runs on the control plane nodes, the cluster version operator (CVO) finishes the OpenShift cluster deployment.

For troubleshooting installation issues, you must use different procedures depending on the installation stage. You can determine the stage by examining the OpenShift installation logs.

#### **Troubleshooting Bootstrap (Bootkube) Stage**

At this stage, the Kubernetes API is not yet available on the bootstrap node. For troubleshooting, you must get the logs from the bootstrap node using SSH. You can log in to the bootstrap node using SSH from the bastion host and inspect the journal and container logs. One of the prerequisites before installing OpenShift is generating an SSH key (cluster SSH key) for enabling SSH access from the bastion host to the cluster nodes.

### WARNING

Do not run the ssh-keygen command at this stage because it will overwrite the cluster SSH key. You will not have access to the cluster nodes with the new cluster SSH key. You must execute the ssh-keygen command before running the OpenShift installation.

[user@demo ~]$ **ssh-keygen -t rsa -b 4096 -N '' -f ${HOME}/.ssh/ocp4-cluster**

Before running the cluster deployment, you must include the cluster SSH public key ocp4-cluster.pub in the install-config.yaml file. After you install the cluster nodes, the cluster SSH key will give you SSH access from the bastion host. To troubleshoot the bootstrap node, run the ssh command from the bastion host to connect to the bootstrap node. The following example assumes that 192.168.50.9 is the bootstrap node IP address. Use the cluster SSH private key to connect to the bootstrap node from the bastion host:

[user@demo ~]$ **ssh -i ${HOME}/.ssh/ocp4-cluster core@192.168.50.9**

The bootstrap node runs two systemd services, the release-image service and the bootkube service.

* The release-image service downloads the OpenShift release container images used during the installation.
* The bootkube service orchestrates the bootstrap stage steps.

After logging in to the bootstrap node, run the following command to debug both services:

sh-4.2# **journalctl -b -f -u release-image.service -u bootkube.service**

Also, run the crictl command on the bootstrap node to look for failed containers and print their logs:

sh-4.2# **sudo bash**

sh-4.2# **crictl ps -a**

sh-4.2# **crictl logs <container\_id>**

Typical errors at this stage include:

* The bootstrap node cannot download container images from quay.io or the local registry due to authentication or networking issues.
* The cluster nodes cannot obtain the IP address of the Kubernetes API due to DNS issues.

#### **Troubleshooting Bootstrap (Temporary Control Plane) Stage**

At this stage, the Kubernetes API and the temporary control plane are running on the bootstrap node.

For troubleshooting, you must get the logs from the bootstrap node. You can use SSH to get logs from the bootstrap node, as explained elsewhere. Because the Kubernetes API is available on the bootstrap node, you can also use the openshift-install and oc commands to obtain the bootstrap and control plane nodes logs.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

[user@demo ~]$ **openshift-install gather bootstrap \**

> **--dir=${HOME}/ocp4-cluster**

INFO Pulling debug logs from the bootstrap machine

INFO Bootstrap gather logs captured here "/home/user/ocp4-cluster/log-bundle-20210107135825.tar.gz"

[user@demo ~]$ **cd ocp4-cluster**

[user@demo ~]$ **tar -xvzf log-bundle-20210107135825.tar.gz**

[user@demo ~]$ **cd log-bundle-20210107135825**

[user@demo ~]$ **ls**

bootstrap/ control-plane/ failed-units.txt rendered-assets/ resources/ unit-status/

For troubleshooting the bootstrap systemd services, use the log files stored in the bootstrap/journals/ directory:

[user@demo ~]$ **ls bootstrap/journals/**

approve-csr.log bootkube.log crio-configure.log crio.log ironic.log kubelet.log master-update-bmh.log release-image.log

For troubleshooting the bootstrap containers, use the log files stored in the bootstrap/containers/ directory:

[user@demo ~]$ **ls bootstrap/containers/**

...output omitted...

cluster-version-operator-652...f5.inspect

cluster-version-operator-652...f5.log

...output omitted...

For troubleshooting the control plane nodes, use the log files stored in the control-plane/ directory:

[user@demo ~]$ **find control-plane/**

control-plane/

control-plane/10.0.203.117

control-plane/10.0.203.117/unit-status

control-plane/10.0.203.117/journals

control-plane/10.0.203.117/journals/kubelet.log

control-plane/10.0.203.117/journals/crio.log

...output omitted...

### NOTE

When using the pre-existing infrastructure installation method, you must specify the IP addresses of the bootstrap and control plane nodes when running the openshift-install gather bootstrap command.

[user@demo ~]$ **openshift-install gather bootstrap \**

> **--dir=${HOME}/ocp4-cluster \**

> **--bootstrap <bootstrap\_address> \**

> **--master <master\_1\_address> \**

> **--master <master\_2\_address> \**

> **--master <master\_3\_address>"**

You can also access the Kubernetes API running on the bootstrap node and monitor the cluster installation progress. Use the standard oc commands from the bastion host for troubleshooting.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

[user@demo ~]$ **oc get nodes**

NAME STATUS ROLES AGE VERSION

master01 Ready master 2m35s v1.19.0+9f84db3

master02 Ready master 2m34s v1.19.0+9f84db3

master03 Ready master 2m35s v1.19.0+9f84db3

[user@demo ~]$ **oc get clusterversion**

NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version False True 8m32s Unable to apply 4.6.4: an unknown error has occurred: MultipleErrors

[user@demo ~]$ **oc get clusteroperators**

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication

cloud-credential True False False 10m

console

csi-snapshot-controller

dns

etcd 4.6.4 Unknown Unknown False 1s

...output omitted...

You can follow the installation process in detail by watching the cluster events.

[user@demo ~]$ **oc get events -A -w**

...output omitted...

openshift-infra 0s Warning NetworkNotReady

At this stage, you can also use the oc adm node-logs command to gather the cluster node logs.

[user@demo ~]$ **oc adm node-logs -u crio master01**

...output omitted...

Jan 27 18:25:57.423858 master01 crio[1627]: time="2021-01-27 18:25:57.423599752Z" level=info msg="Checking image status: quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:009...fd5" id=39...fe name=/runtime.v1alpha2.ImageService/ImageStatus

[user@demo ~]$ **oc adm node-logs -u kubelet master01**

...output omitted...

Jan 27 18:31:13.904091 master01 hyperkube[1661]: I0127 18:31:13.903991 1661 prober.go:126] Readiness probe for "etcd-master01\_openshift-etcd(63...40):etcd" succeeded

[user@demo ~]$ **oc adm node-logs master01**

...output omitted...

Jan 27 18:32:47.487425 master01 hyperkube[1661]: I0127 18:32:47.487445 1661 kubelet.go:1914] SyncLoop (housekeeping)

Jan 27 18:32:47.536556 master01 hyperkube[1661]: I0127 18:32:47.536513 1661 exec.go:60] Exec probe response: ""

Jan 27 18:32:47.536556 master01 hyperkube[1661]: I0127 18:32:47.536546 1661 prober.go:126] Readiness probe for "ovs-z57pn\_openshift-sdn(9e...2c):openvswitch" succeeded

### NOTE

The OpenShift cluster nodes run very few local services because most of the system services run as containers. The main exceptions are the cri-o container engine and the kubelet, which are systemd service units.

Also, use the oc debug command to create a debug session on the control plane nodes for troubleshooting.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

[user@demo ~]$ **oc get nodes**

NAME STATUS ROLES AGE VERSION

master01 Ready master 2m35s v1.19.0+9f84db3

master02 Ready master 2m34s v1.19.0+9f84db3

master03 Ready master 2m35s v1.19.0+9f84db3

[user@demo ~]$ **oc debug node/master01**

...output omitted...

sh-4.2#

Once you log in, run the following command to debug the node logs:

sh-4.2# **chroot /host**

sh-4.2# **journalctl -f**

Also, use the command crictl on the control plane node to search for failed containers and get their logs:

sh-4.2# **sudo bash**

sh-4.2# **crictl ps -a**

sh-4.2# **crictl logs <container\_id>**

### NOTE

Red Hat does not recommend using SSH to access the cluster nodes after the installation has finished. During the installation, if you are not able to use the oc debug command, then you can access the control plane nodes using SSH from the bastion host. The following example assumes that 192.168.50.10 is the IP address for one of the control plane nodes. Use the cluster SSH private key to connect to that control plane node from the bastion host.

[user@demo ~]$ **ssh -i ${HOME}/.ssh/ocp4-cluster core@192.168.50.9**

Typical errors at this stage include:

* Control plane nodes installation issues
* DNS resolution issues

#### **Troubleshooting the Production Control Plane Stage**

At this stage, the Kubernetes API and the production control plane are running on the control plane nodes. The cluster version operator (CVO) running on the production control plane installs all the operators that build the OpenShift cluster and finish the installation. You can apply the same troubleshooting techniques used in the previous section. Typical errors at this stage include:

* OpenShift operators installation issues

### **Verifying OpenShift Installations**

After the OpenShift installation finishes successfully, administrators must ensure that the installed cluster is healthy and ready for day-2 tasks to onboard users and applications.

#### **OpenShift Cluster Health**

From the bastion host, you can perform a basic health check using the oc command.

* Configure the KUBECONFIG environment variable to authenticate against the Kubernetes API with cluster-admin permissions.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

* Verify that all the cluster nodes have their system clock synchronized with a Network Time Protocol (NTP) server.
* [user@demo ~]$ **oc debug node/master01**
* ...output omitted...
* sh-4.4# **chroot /host**
* sh-4.4# **cat /etc/chrony.conf**
* # Use public servers from the pool.ntp.org project.
* # Please consider joining the pool (http://www.pool.ntp.org/join.html).
* **pool 2.rhel.pool.ntp.org iburst**
* ...output omitted...
* sh-4.4# **sudo chronyc tracking**
* Reference ID : 8AEC8070 (time.gac.edu)
* Stratum : 3
* Ref time (UTC) : Thu Feb 11 13:06:57 2021
* System time : 0.000034756 seconds fast of NTP time
* Last offset : **-0.000001187 seconds**
* RMS offset : 0.004707427 seconds
* Frequency : 28.194 ppm fast
* Residual freq : -0.000 ppm
* Skew : 0.136 ppm
* Root delay : 0.052070152 seconds
* Root dispersion : 0.018801220 seconds
* Update interval : 64.9 seconds

Leap status : **Normal**

The chronyd systemd service running on the cluster node uses the NTP pool 2.rhel.pool.ntp.org. The system clock is synchronized with the NTP server time.gac.edu.

Repeat this procedure on all the cluster nodes.

* Verify that all the cluster nodes are in a Ready status.

If a cluster node is not in a Ready status, it cannot communicate with the OpenShift control plane and is unavailable to the cluster.

[user@demo ~]$ **oc get nodes**

NAME STATUS ROLES AGE VERSION

master01 **Ready** master 15h v1.19.0+9f84db3

master02 **Ready** master 15h v1.19.0+9f84db3

master03 **Ready** master 15h v1.19.0+9f84db3

worker01 **Ready** worker 15h v1.19.0+9f84db3

worker02 **Ready** worker 15h v1.19.0+9f84db3

* Check that all the cluster nodes are reporting usage metrics.
* [user@demo ~]$ **oc adm top node**
* NAME CPU(cores) CPU% MEMORY(bytes) MEMORY%
* master01 677m 19% 4747Mi 31%
* master02 391m 11% 3300Mi 22%
* master03 519m 14% 4037Mi 27%
* worker01 273m 7% 2435Mi 35%

worker02 313m 8% 2906Mi 42%

* Ensure that there are no certificate signing requests (CSRs) pending approval.

[user@demo ~]$ **oc get csr | grep Pending**

* Confirm that the cluster version operator report shows that the OpenShift cluster is available and ready.
* [user@demo ~]$ **oc get clusterversion**
* NAME VERSION AVAILABLE PROGRESSING SINCE STATUS

version 4.6.4 **True** **False** 22h **Cluster version is 4.6.4**

* Check that all the cluster operators are available and ready.

If the cluster is healthy, all the cluster operators should be available and not progressing unless one or more operators are still applying the configuration.

[user@demo ~]$ **oc get clusteroperators**

NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE

authentication 4.6.4 True False False 22h

cloud-credential 4.6.4 True False False 22h

cluster-autoscaler 4.6.4 True False False 22h

config-operator 4.6.4 True False False 22h

console 4.6.4 True False False 22h

...output omitted...

* Verify that there are not any pods with scheduling or execution issues in the cluster.
* [user@demo ~]$ **oc get pods --all-namespaces | grep -v -E 'Running|Completed'**

NAMESPACE NAME READY STATUS RESTARTS AGE

#### **OpenShift Etcd Health**

* Ensure that all the etcd cluster members are healthy.
* [user@demo ~]$ **oc get pods -n openshift-etcd | grep etcd-master**
* etcd-master01 **3/3** **Running** 0 22h
* etcd-master02 **3/3** **Running** 0 22h

etcd-master03 **3/3** **Running** 0 22h

[user@demo ~]$ **oc rsh -n openshift-etcd etcd-master01**

sh-4.4# **etcdctl endpoint health --cluster**

https://192.168.50.10:2379 is **healthy**: successfully committed proposal: took=10.8ms

https://192.168.50.12:2379 is **healthy**: successfully committed proposal: took=11.8ms

https://192.168.50.11:2379 is **healthy**: successfully committed proposal: took=12.1ms

#### **OpenShift API and Console Health**

* Verify that the OpenShift API DNS record api.ocp4.example.com is configured to use the external load balancer IP address 192.168.50.254.
* [user@demo ~]$ **dig api.ocp4.example.com**
* ...output omitted...
* ;; QUESTION SECTION:
* ;api.ocp4.example.com. IN A
* ;; ANSWER SECTION:
* api.ocp4.example.com. 85333 IN A **192.168.50.254**
* ;; Query time: 0 msec
* ;; SERVER: 172.25.250.254#53(172.25.250.254)
* ;; WHEN: Thu Jan 28 05:11:46 EST 2021

;; MSG SIZE rcvd: 71

* Verify that the OpenShift API is available by requesting the Kubernetes version.
* [user@demo ~]$ **curl -k https://api.ocp4.example.com:6443/version**
* ...output omitted...
* "gitVersion": "**v1.19.0+9f84db3**",

...output omitted...

* Check that you can connect to the OpenShift Console.
* [user@demo ~]$ **curl -kIs \**
* > **https://console-openshift-console.apps.ocp4.example.com**
* ...output omitted...
* **HTTP/1.1 200 OK**

...output omitted...

[user@demo ~]$ **firefox https://console-openshift-console.apps.ocp4.example.com**

#### **OpenShift Registry Health**

* Ensure that the number of internal registry pods running on the OpenShift cluster matches its deployment configuration.
* [user@demo ~]$ **oc -n openshift-image-registry get deployment.apps/image-registry**
* NAME READY UP-TO-DATE AVAILABLE AGE

image-registry **2/2 2 2** 24h

* If there are multiple compute nodes, verify that each registry pod is running on a different compute node.
* [user@demo ~]$ **oc -n openshift-image-registry get pods -o wide**
* ...output omitted...
* NAME READY STATUS RESTARTS AGE IP NODE
* image-registry-69d88984fb-tjpkl 1/1 Running 0 16m 10.128.2.32 **worker02**
* image-registry-59b67d44f6-n7wkq 1/1 Running 0 16m 10.131.2.12 **worker01**

...output omitted...

* From any cluster node, check the internal registry health.
* [user@demo ~]$ **oc debug node/master01**
* ...output omitted...
* sh-4.4# **chroot /host**
* sh-4.4# **curl -kIs \**
* > **https://image-registry.openshift-image-registry.svc:5000/healthz**
* ...output omitted...
* **HTTP/2 200**

...output omitted...

* Verify that the internal registry deployment is using persistent storage. Also, ensure that the image registry operator is in the Managed management state.
* [user@demo ~]$ **oc get configs.imageregistry.operator.openshift.io cluster -o yaml**
* ...output omitted...
* spec:
* **managementState: Managed**
* ...output omitted...
* **storage:**
* **pvc:**
* **claim: registry-claim**

...output omitted...

#### **OpenShift Ingress Health**

* Verify that the wildcard DNS record for applications, \*.apps.ocp4.example.com, is configured to use the external load balancer IP address 192.168.50.254.
* [user@demo ~]$ **dig test.apps.ocp4.example.com**
* ...output omitted...
* ;; QUESTION SECTION:
* ;test.apps.ocp4.example.com. IN A
* ;; ANSWER SECTION:
* test.apps.ocp4.example.com. 86358 IN A **192.168.50.254**
* ;; Query time: 0 msec
* ;; SERVER: 172.25.250.254#53(172.25.250.254)
* ;; WHEN: Thu Jan 28 05:11:46 EST 2021

;; MSG SIZE rcvd: 71

* Check that you can access an application exposed by an OpenShift Ingress route.
* [user@demo ~]$ **oc get routes -A | grep downloads**

openshift-console downloads **downloads-openshift-console.apps.ocp4.example.com**

[user@demo ~]$ **curl -kIs \**

> **https://downloads-openshift-console.apps.ocp4.example.com**

...output omitted...

**HTTP/1.0 200 OK**

...output omitted...

* Ensure that the number of router pods running on the OpenShift cluster matches its deployment configuration.
* [user@demo ~]$ **oc -n openshift-ingress get deployment.apps/router-default**
* NAME READY UP-TO-DATE AVAILABLE AGE

router-default **2/2 2 2** 24h

* If there are multiple compute nodes, verify that each router pod is running on a different compute node.
* [user@demo ~]$ **oc -n openshift-ingress get pods -o wide | grep router**
* NAME READY STATUS RESTARTS AGE IP NODE
* router-default-b7567-l2z4d 1/1 Running 1 16h 192.168.50.13 **worker01**

router-default-b7567-qf8x4 1/1 Running 1 16h 192.168.50.14 **worker02**

#### **OpenShift Dynamic Storage Provider Health**

When installing OpenShift in a supported cloud provider, the installer configures a dynamic storage provider. In this case, you must verify the status of the dynamic storage provider.

During the OpenShift installation on AWS using the full-stack automation method, the installer configures an AWS EBS dynamic storage provider. This dynamic storage provider uses the aws-ebs storage provisioner.

The OpenShift installation process creates a storage class named gp2 that uses the AWS EBS dynamic storage provider as the back end. The gp2 storage class dynamically provisions persistent storage for the containerized applications running on the OpenShift cluster. The OpenShift installation process configures the gp2 storage class as the default storage class. Unless you specify a different storage class in the PVC definition, any PVC request will use the gp2 storage class to create and bound the PV dynamically.

* Check the AWS EBS gp2 storage class status.
* [user@demo ~]$ **oc get sc**
* NAME PROVISIONER RECLAIMPOLICY BINDINGMODE EXPANSION AGE
* **gp2 (default) kubernetes.io/aws-ebs** Delete **WaitForFirstConsumer** true 32m

gp2-csi ebs.csi.aws.com Delete WaitForFirstConsumer true 32m

The gp2 storage class uses the WaitForFirstConsumer volume binding mode. This volume binding mode delays the binding and provisioning of a PersistentVolume until a pod using the PersistentVolumeClaim is created. This configuration is immutable for this storage class.

* Verify that the gp2 storage class works as expected.

Create a simple httpd application that uses persistent storage for its DocumentRoot directory at /var/www/html.

[user@demo ~]$ **oc new-project httpd-persistent**

...output omitted...

[user@demo ~]$ **cat /tmp/httpd-persistent.yaml**

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: httpd-claim

namespace: httpd-persistent

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

---

apiVersion: v1

kind: Pod

metadata:

name: httpd

namespace: httpd-persistent

spec:

containers:

- image: registry.redhat.io/rhel8/httpd-24:latest

name: httpd

ports:

- containerPort: 8080

name: http

protocol: TCP

volumeMounts:

- mountPath: /var/www/html

name: httpd-claim

volumes:

- name: httpd-claim

persistentVolumeClaim:

claimName: httpd-claim

[user@demo ~]$ **oc create -f /tmp/httpd-persistent.yaml**

persistentvolumeclaim/httpd-claim created

pod/httpd created

Verify that the PVC creation automatically triggers the PV provisioning and bounding through the gp2 default storage class.

[user@demo ~]$ **oc get pvc**

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

httpd-claim **Bound** pvc-d965cb1f 3Gi RWO **gp2** 29s

[user@demo ~]$ **oc rsh httpd**

sh-4.4$ **df -h**

Filesystem Size Used Avail Use% Mounted on

...output omitted...

**/dev/nvme2n1 2.9G 9.0M 2.9G 1% /var/www/html**

...output omitted...

#### **OpenShift Application Build and Deployment Test**

* Build and deploy a test application to verify the OpenShift application build cycle.
* [user@demo ~]$ **oc new-project validate**
* ...output omitted...
* [user@demo ~]$ **oc new-app django-psql-example**

...output omitted...

[user@demo ~]$ **oc get pods -n validate**

NAME READY STATUS RESTARTS AGE

django-psql-example-1-build 0/1 Completed 0 11m

django-psql-example-1-deploy 0/1 Completed 0 10m

django-psql-example-1-vfb5l 1/1 Running 0 10m

postgresql-1-bdgkk 1/1 Running 0 11m

postgresql-1-deploy 0/1 Completed 0 11m

[user@demo ~]$ **oc logs -f django-psql-example-1-build**

...output omitted...

**Successfully pushed image-registry.openshift-image-registry.svc:5000/validate/django-psql-example@sha256:b97b...ff82**

**Push successful**

[user@demo ~]$ **oc logs -f django-psql-example-1-deploy**

...output omitted...

--> Scaling django-psql-example-1 to 1

--> **Success**

[user@demo ~]$ **oc get routes -n validate**

NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

django-psql-example **django-psql-example-validate.apps.ocp4.example.com** django-psql-example <all> None

[user@demo ~]$ **curl -Is \**

> **django-psql-example-validate.apps.ocp4.example.com**

...output omitted...

**HTTP/1.1 200 OK**

...output omitted...

[user@demo ~]$ **firefox http://django-psql-example-validate.apps.ocp4-aws.example.com**

#### **OpenShift Cluster Network**

* Verify the OpenShift cluster network configuration.
* [user@demo ~]$ **oc get network.config/cluster -o yaml**
* apiVersion: config.openshift.io/v1
* kind: Network
* metadata:
* ...output omitted...
* **status:**
* **clusterNetwork:**
* **- cidr: 10.128.0.0/14**
* **hostPrefix: 23**
* **clusterNetworkMTU: 8142**
* **networkType: OpenShiftSDN**
* **serviceNetwork:**

**- 172.30.0.0/16**

#### **OpenShift Etcd Storage Performance**

* Verify the etcd storage performance.
* [user@demo ~]$ **oc get pods -n openshift-etcd | grep etcd-master**
* etcd-master01 3/3 Running 0 22h
* etcd-master02 3/3 Running 0 22h

etcd-master03 3/3 Running 0 22h

[user@demo ~]$ **oc rsh -n openshift-etcd etcd-master01**

...output omitted...

sh-4.4# **etcdctl check perf --load="s"**

60 / 60 Boooooooooooooooooooooooooooooooooooooooooooooooooo! 100.00% 1m0s

PASS: Throughput is 150 writes/s

PASS: Slowest request took 0.220329s

PASS: Stddev is 0.018010s

**PASS**

sh-4.4# **etcdctl check perf --load="m"**

60 / 60 Boooooooooooooooooooooooooooooooooooooooooooooooooo! 100.00% 1m0s

PASS: Throughput is 964 writes/s

PASS: Slowest request took 0.379547s

PASS: Stddev is 0.022218s

**PASS**

sh-4.4# **etcdctl check perf --load="l"**

60 / 60 Boooooooooooooooooooooooooooooooooooooooooooooooooo! 100.00% 1m0s

FAIL: Throughput too low: 4586 writes/s

PASS: Slowest request took 0.258474s

PASS: Stddev is 0.032695s

**FAIL**

From the test result, the etcd cluster performs well for a medium cluster (--load="m") and fails for a large cluster (--load="l"). For more information about etcd performance, visit the etcd documentation page at <https://etcd.io/docs/current/op-guide/hardware/>

For more detailed etcd storage performance information, use the fio tool from the etcd-perf container to run a performance test on the control plane nodes. The performance test output reports whether the disk is fast enough to host etcd by comparing the 99th percentile of the fsync metric captured from the run to see if it is less than **10 ms**.

[user@demo ~]$ **oc debug node/master01**

...output omitted...

sh-4.4# **chroot /host**

sh-4.4# **podman run --volume /var/lib/etcd:/var/lib/etcd:Z quay.io/openshift-scale/etcd-perf**

...output omitted...

{

"fio version" : "fio-3.7",

...output omitted...

"global options" : {

**"rw" : "write",**

"ioengine" : "sync",

"fdatasync" : "1",

**"directory" : "/var/lib/etcd",**

**"size" : "22m",**

**"bs" : "2300"**

},

...output omitted...

**"write"** : {

**"iops" : 328.808892,**

...output omitted..

},

...output omitted...

}

]

}

-----------------------------------------------

99th percentile of fsync is 6193152 ns

**99th percentile of the fsync is within the recommended threshold - 10 ms, the disk can be used to host etcd**

The fio performance test produces the following result:

* 1. This test writes 22 MiB of data in blocks of 2300 bytes on the /var/lib/etcd directory.
  2. The 99th percentile of the fsync is 6193152 ns, which is equivalent to **6 ms** of write latency.
  3. The operating system has achieved an average of **328 IOPS** during the test.

#### **OpenShift Machine API**

When installing OpenShift in a supported cloud provider, the installer configures the compute node autoscaling using the OpenShift Machine API component. In this case, you must verify the status of the compute node autoscaling. One of the essential advantages of using the full-stack automation installation method on AWS is that the OpenShift installation process configures the compute node autoscaling.

The OpenShift Machine API is the component that defines and manages the OpenShift Machines resource. The OpenShift Machines resource represents the OpenShift cluster nodes. The OpenShift Machine API:

* Creates, updates, and deletes Machines
* Creates the infrastructure (instance or VM) for the node

You can use the OpenShift MachineSets resource to control sets of Machines. A Machineset represents:

* A set of Machines
* An abstraction of the underlying infrastructure

When installing OpenShift on AWS using the full-stack automation method, the OpenShift installer creates and configures a MachineSet for each availability zone in the selected region.

[user@demo ~]$ **oc get machines -n openshift-machine-api**

NAME PHASE TYPE REGION ZONE AGE

ocp4-aws-9r678-master-0 Running m5.2xlarge us-east-2 us-east-2a 16h

ocp4-aws-9r678-master-1 Running m5.2xlarge us-east-2 us-east-2b 16h

ocp4-aws-9r678-master-2 Running m5.2xlarge us-east-2 us-east-2c 16h

ocp4-aws-9r678-worker-us-east-2a-gq2ps Running m5.4xlarge us-east-2 us-east-2a 16h

ocp4-aws-9r678-worker-us-east-2b-slp7l Running m5.4xlarge us-east-2 us-east-2b 16h

ocp4-aws-9r678-worker-us-east-2c-vj7pj Running m5.4xlarge us-east-2 us-east-2c 16h

[user@demo ~]$ **oc get machinesets -n openshift-machine-api**

NAME DESIRED CURRENT READY AVAILABLE AGE

ocp4-aws-9r678-worker-us-east-2a 1 1 1 1 16h

ocp4-aws-9r678-worker-us-east-2b 1 1 1 1 16h

ocp4-aws-9r678-worker-us-east-2c 1 1 1 1 16h

[user@demo ~]$ **oc get nodes --label-columns \**

> **failure-domain.beta.kubernetes.io/region,failure-domain.beta.kubernetes.io/zone**

NAME STATUS ROLES AGE VERSION REGION ZONE

ip-10-0-151-177.us-east-2.compute.internal Ready master 16h v1.19.0+9f84db3 us-east-2 us-east-2a

ip-10-0-157-4.us-east-2.compute.internal Ready **worker** 16h v1.19.0+9f84db3 us-east-2 **us-east-2a**

ip-10-0-166-182.us-east-2.compute.internal Ready **worker** 16h v1.19.0+9f84db3 us-east-2 **us-east-2b**

ip-10-0-180-27.us-east-2.compute.internal Ready master 17h v1.19.0+9f84db3 us-east-2 us-east-2b

ip-10-0-205-233.us-east-2.compute.internal Ready master 17h v1.19.0+9f84db3 us-east-2 us-east-2c

ip-10-0-217-153.us-east-2.compute.internal Ready **worker** 16h v1.19.0+9f84db3 us-east-2 **us-east-2c**

Using the worker MachineSets, you can scale up (or down) the number of compute nodes running on the cluster. When scaling up a worker MachineSet:

* The OpenShift Machine API automatically provisions and starts an AWS EC2 instance for the new compute node.
* The new compute node gets its ignition configuration file and installs RHCOS.
* The new compute node joins the OpenShift cluster automatically.

[user@demo ~]$ **oc scale machineset ocp4-aws-9r678-worker-us-east-2c \**

> **--replicas=2 -n openshift-machine-api**

machineset.machine.openshift.io/ocp4-aws-9r678-worker-us-east-2c scaled

[user@demo ~]$ **oc get machinesets -n openshift-machine-api**

NAME DESIRED CURRENT READY AVAILABLE AGE

ocp4-aws-9r678-worker-us-east-2a 1 1 1 1 17h

ocp4-aws-9r678-worker-us-east-2b 1 1 1 1 17h

**ocp4-aws-9r678-worker-us-east-2c 2 2 1 1 17h**

After a few minutes, the new compute node must be in Ready status.

[user@demo ~]$ **oc get machinesets -n openshift-machine-api**

NAME DESIRED CURRENT READY AVAILABLE AGE

ocp4-aws-9r678-worker-us-east-2a 1 1 1 1 17h

ocp4-aws-9r678-worker-us-east-2b 1 1 1 1 17h

**ocp4-aws-9r678-worker-us-east-2c 2 2 2 2 17h**

[user@demo ~]$ **oc get machines -n openshift-machine-api**

...output omitted..

**ocp4-aws-9r678-worker-us-east-2c-65hln Running m5.4xlarge us-east-2 us-east-2c 3m41s**

ocp4-aws-9r678-worker-us-east-2c-vj7pj Running m5.4xlarge us-east-2 us-east-2c 17h

[user@demo ~]$ **oc get nodes --label-columns \**

> **failure-domain.beta.kubernetes.io/region,failure-domain.beta.kubernetes.io/zone**

...output omitted..

**ip-10-0-196-32.us-east-2.compute.internal Ready worker 2m31s v1.19.0+9f84db3 us-east-2 us-east-2c**

ip-10-0-205-233.us-east-2.compute.internal Ready master 17h v1.19.0+9f84db3 us-east-2 us-east-2c

As you can see in the preceding example, the OpenShift Machine API automatically provisioned and added a new compute node to the cluster (ip-10-0-196-32.us-east-2.compute.internal) on the desired us-east-2c AWS AZ.

### **Gathering OpenShift Data**

When interacting with Red Hat Support to solve any OpenShift issue, administrators are asked to provide cluster debug information. Depending on the OpenShift component to troubleshoot, administrators can use different debug mechanisms.

#### **OpenShift Cluster Data**

You can gather cluster debug information using the oc adm must-gather CLI command as the cluster-admin user. This CLI command collects the information from your cluster, such as:

* Resource definitions
* Audit logs
* Service logs

The execution of the oc adm must-gather command creates a new pod on the cluster. That pod collects the cluster data and stores it in a new directory. The new directory name starts with must-gather.local. The oc adm must-gather command creates this directory in the current working directory.

[user@demo ~]$ **export KUBECONFIG=${HOME}/ocp4-cluster/auth/kubeconfig**

[user@demo ~]$ **cd ${HOME}**

[user@demo ~]$ **oc adm must-gather**

[must-gather ] OUT Using must-gather plugin-in image: quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:47..86

[must-gather ] OUT namespace/openshift-must-gather-ndwcz created

[must-gather ] OUT clusterrolebinding.rbac.authorization.k8s.io/must-gather-5ptk2 created

[must-gather ] OUT pod for plug-in image quay.io/openshift-release-dev/ocp-v4.0-art-dev@sha256:47..86 created

[must-gather-276db] POD Wrote inspect data to must-gather.

...output omitted...

[user@demo ~]$ **ls**

install-config.yaml **must-gather.local.1227184995617480385/** ocp4-cluster/

[user@demo ~]$ **find must-gather.local.1227184995617480385/**

must-gather.local.1227184995617480385/

must-gather.local.1227184995617480385/timestamp

...output omitted...

As you can see, this directory contains the OpenShift resources definitions, services logs, and audit logs.

### NOTE

When opening an OpenShift support case in the Red Hat Customer Portal, create a tar file with the output generated by the oc adm must-gather execution and attach it to the support case.

[user@demo ~]$ **tar cvaf must-gather.tar.gz \**

> **must-gather.local.1227184995617480385/**

...output omitted...

You can gather debug information about specific features using the oc adm must-gather CLI command with the --image or --image-stream argument.

[user@demo ~]$ **oc adm must-gather \**

> **--image-stream=openshift/must-gather \**

> **--image=registry.redhat.io/container-native-virtualization/\**

> **cnv-must-gather-rhel8:v2.5.2**

...output omitted...

For instance, using the cnv-must-gather-rhel8 image the oc adm must-gather command collects OpenShift Virtualization specific data.

**Table 1.8. Most Commonly Used Must-gather Images**

| **Image** | **Purpose** |
| --- | --- |
| registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v2.5.2 | Data collection for OpenShift Virtualization |
| registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8 | Data collection for OpenShift Serverless |
| registry.redhat.io/openshift-service-mesh/istio-must-gather-rhel7 | Data collection for Red Hat OpenShift Service Mesh |
| registry.redhat.io/rhcam-1-2/openshift-migration-must-gather-rhel8 | Data collection for migration-related information |
| registry.redhat.io/ocs4/ocs-must-gather-rhel8 | Data collection for Red Hat OpenShift Container Storage |
| registry.redhat.io/openshift4/ose-cluster-logging-operator | Data collection for Red Hat OpenShift cluster logging |

#### **OpenShift Node Data**

In some scenarios, Red Hat Support will ask you to collect a sosreport file from a specific OpenShift cluster node. The sosreport command is a tool that collects configuration details, system information, and diagnostic data from Red Hat Enterprise Linux (RHEL) and Red Hat Enterprise  Linux CoreOS (RHCOS) systems.

Red Hat recommends using a debug pod to generate a sosreport from an OpenShift cluster node.

[user@demo ~]$ **oc debug node/ip-10-0-151-177.us-east-2.compute.internal**

...output omitted...

sh-4.4# **chroot /host**

sh-4.4# **toolbox**

Trying to pull registry.redhat.io/rhel8/support-tools...

...output omitted...

[root@ip-10-0-151-177 /]# **sosreport -k crio.all=on -k crio.logs=on**

...output omitted...

Your sosreport has been generated and saved in:

/host/var/tmp/sosreport-ip-10-0-151-177-1234-2021-01-11-crlsmlr.tar.xz

Size 31.07MiB

Owner root

md5 3ddfb7a774002fc8fb18da9c9c1534bc

Please send this file to your support representative.

[root@ip-10-0-151-177 /]# **exit**

sh-4.4# **exit**

sh-4.4# **ls -lrt \**

> **/host/var/tmp/sosreport-ip-10-0-151-177-1234-2021-01-11-crlsmlr.tar.xz**

-rw-------. 1 root root 32578896 Jan 11 13:42 /host/var/tmp/sosreport-ip-10-0-151-177-1234-2021-01-11-crlsmlr.tar.xz

#### **OpenShift Remote Health Monitoring**

OpenShift collects anonymized aggregated information about the health, usage, and size of the clusters. With this information, Red Hat can proactively react to issues that can impact customers.

The OpenShift cluster reports this information to Red Hat using two components:

* Telemetry
* Insights Operator

The Telemetry component sends a chosen subset of the cluster monitoring metrics to Red Hat. These metrics are sent continuously and describe:

* OpenShift cluster size
* OpenShift components health
* OpenShift upgrades health
* Limited OpenShift usage information
* OpenShift alerts summary info

The Insights Operator periodically gathers the cluster configuration and component failure status and reports that data to Red Hat. Using this information, the Red Hat OpenShift Cluster Manager proactively identifies potential cluster issues and provides solutions and preventive actions.

### NOTE

For more information, refer to the Remote health monitoring with connected clusters guide in the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/support>

You can access your cluster information using the [Red Hat OpenShift Cluster Manager Console](https://cloud.redhat.com/openshift). If you manage several OpenShift clusters, you will need your cluster id to identify your cluster in the Red Hat OpenShift Cluster Manager Console.

[user@demo ~]$ **oc get clusterversion \**

> **-o jsonpath='{.items[].spec.clusterID}{"\n"}'**

9d1c5e73-9deb-452b-b327-376d04315246

After retrieving the cluster id, open your web browser and navigate to [Red Hat OpenShift Cluster Manager Console](https://cloud.redhat.com/openshift) using your Red Hat account. Then click your cluster id link and review your cluster information under the Overview, Monitoring, Insights, and Support navigation tabs.

A screenshot of a computer

Description automatically generated

Figure 1.9: Red Hat OpenShift Cluster Manager Console

### **Deleting an OpenShift Cluster**

If you need to remove your OpenShift cluster, then you can run the OpenShift installer using the destroy cluster option.

[user@demo ~]$ **openshift-install destroy cluster \**

> **--dir=${HOME}/ocp4-cluster**

...output omitted...

### REFERENCES

* For more information, refer to the Installation configuration chapter of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing>
* For more information, refer to the Installing a cluster on AWS with customizations section of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing_on_aws>

# Chapter 3.  Installing OpenShift on a Virtualized Environment

[Introducing OpenShift Installation on Hypervisors](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03)

[Quiz: Introducing OpenShift Installation on Hypervisors](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s02)

[Describing the Installation of OpenShift on vSphere Using Full-stack Automation](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s03)

[Quiz: Describing the Installation of OpenShift on vSphere Using Full-stack Automation](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s04)

[Describing the Installation of OpenShift on vSphere Using Pre-existing Infrastructure](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s05)

[Quiz: Describing the Installation of OpenShift on vSphere Using Pre-existing Infrastructure](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s06)

[Quiz: Chapter Review: Installing OpenShift on a Virtualized Environment](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s07)

[Summary](https://rol.redhat.com/rol/app/courses/do322-4.6/pages/ch03s08)

**Abstract**

|  |  |
| --- | --- |
| **Goal** | Provision OpenShift clusters on hypervisors, with common customizations, using the full-stack automation and the pre-existing infrastructure installation methods. |
| **Objectives** | * Describe the architecture and workflow for installing OpenShift on hypervisors using the full-stack automation and pre-existing infrastructure methods. * Describe the installation of OpenShift on vSphere using full-stack automation with common customizations. * Describe the installation of OpenShift on vSphere using pre-existing infrastructure with common customizations. |
| **Sections** | * Introducing OpenShift Installation on Hypervisors (and Quiz) * Describing the Installation of OpenShift on vSphere Using Full-stack Automation (and Quiz) * Describing the Installation of OpenShift on vSphere Using Pre-existing Infrastructure (and Quiz) |

## **Introducing OpenShift Installation on Hypervisors**

### **Objectives**

* Describe the architecture and workflow for installing OpenShift on hypervisors using the full-stack automation and pre-existing infrastructure methods.

### **Introducing OpenShift Installation on Hypervisors**

Prior to the adoption of hypervisors, physical computers only leveraged one operating system. Although this resulted in a stable physical server that could effectively and efficiently handle requests from a single operating system, unused resources were wasted. The hypervisor provides a software layer that enables multiple operating systems to run in parallel.

A hypervisor is a specific layer of software that builds and runs virtual machines. A hypervisor is sometimes called a virtual machine (VM) monitor due to its operating system isolation features.

VM resources are logically isolated from one another. Resources such as memory and CPU are allocated and scheduled by the hypervisor to existing and new virtual machines based on VM requirements and the pool of resources available from the physical host.

The primary benefit of virtualization is enabling multiple operating systems that share the same virtualized hardware resources managed by a hypervisor. A successful deployment of OpenShift Container Platform also requires other services besides the hypervisor, such as a traffic load balancer or DNS zone management.

### **Introducing Hypervisors**

Virtualization hypervisors are categorized as either a bare metal hypervisor or hosted hypervisor.

A native hypervisor is commonly defined as a bare metal hypervisor, due to its ability to replace the host operating system, run directly on the host’s hardware, manage the guest operating systems, and schedule VM resources directly to the hardware. The following bare metal hypervisors are commonly deployed in enterprise data centers and server-based environments.

* RHV
* Microsoft Hyper-V
* vSphere ESXi

A software layer hypervisor is considered a hosted hypervisor because it can run on a standard operating system as a software layer or application.

In contrast to bare metal hypervisors, software layer hypervisors schedule VM resources against the host operating system and are executed against the hardware. They are typically suitable for personal computer users who require access to multiple operating systems. The following are examples software layer hypervisors:

* VMware Workstation
* Oracle VirtualBox

#### **Reasons for Installing OpenShift on Hypervisors Using Full-stack Automation**

On supported hypervisors, Red Hat recommends using the Full-stack installation method for the following reasons:

* Cluster administrators install OpenShift with minimal manual intervention in an opinionated installer wizard.
* The OpenShift installer creates the virtual machines, installs RHCOS, and then starts the OpenShift install process.
* The OpenShift installer uses the credentials provided to connect to the virtualization management software, upload a template, then clone that template to be used for the bootstrap and control plane nodes.
* The OpenShift installer fully integrates the OpenShift Machine API resource with the virtualization management services. The installer creates the OpenShift MachineSets resource, so automatic node provisioning and cluster autoscaling are ready to use after the installation completes.
* The Only DNS requirement are two entries, Ingress and API.

#### **Reasons for Installing OpenShift on Hypervisors Using Pre-existing Infrastructure**

Deploying OpenShift in a pre-existing infrastructure can be advantageous for the following reasons:

* The cluster administrator maintains and manages the infrastructure, allowing for detailed customizations.
* OpenShift clusters can be deployed on infrastructure specifically tailored to their needs.
* Cluster administrators have more flexibility and can more easily respond to resource requirements.

#### **Comparing Full-stack Automation and Pre-existing Infrastructure Installation Methods**

The following table displays the common deployment differences and requirements between Full-stack Automation and Pre-existing Infrastructure.

**Table 3.1. Full-stack Automation & Pre-existing Infrastructure Comparison**

| **Action** | **Full-stack Automation** | **Pre-existing Infrastructure** |
| --- | --- | --- |
| Build network | Installer | User |
| Setup load balancer | Installer | User |
| Configure DNS | Installer | User |
| Hardware or VM provisioning | Installer | User |
| OS installation | Installer | User |
| Generate ignition configs | Installer | Installer |
| OS support | Installer: RHCOS | User: RHCOS |
| Configure persistent storage for the internal registry | Installer | User |
| Configure dynamic storage provider | Installer | User |
| Configure node provisioning and autoscaling | Yes | Only for providers with OpenShift Machine API support. |

#### **Describing the General Architecture of a Full-stack Cluster in a Hypervisor.**

|  |
| --- |
|  |

A Full Stack cluster architecture consists of the following:

**Bastion host (Optional)**

The bastion host requires network connectivity to the cloud provider API to install OpenShift and to the OpenShift API for managing OpenShift after installation.

**Control plane nodes**

Manages workloads for the compute nodes and runs services required to control the cluster.

**Compute nodes**

Location where the actual workloads requested by Kubernetes users run and are managed.

**Infrastructure nodes**

Run core infrastructure components such as service brokers and logging.

**Temporary bootstrap node**

A temporary node that runs a minimal Kubernetes used to deploy the OpenShift control plane. It is deleted at the end of the installation.

**Ingress load balancer**

The virtual IP (VIP) is managed by Keepalived and is only hosted on nodes that have a router instance. Traffic destined for the \\*.apps Ingress VIP are passed directly to the router instance.

**API load balancer**

When a client creates a new request the API, HAproxy on the node hosting the API IP load balances across nodes using round robin.

#### **Describing OpenShift Network Plug-ins for vSphere**

OpenShift software defined networking (SDN) is available for layer 2 VM connectivity. However, there are vSphere plug-ins that can integrate or replace features provided by OpenShift SDN.

**Networking (NSX-T)**

Networking (NSX-T) is the Software Defined Network (SDN) integrated into vSphere. NSX-T creates overlay networks for VM connectivity with additional features, such as micro-segmentation, load balancers, and granular security policies. NSX-T enables any traffic to or from the VM to be firewalled at the network layer.

**NSX Container Plug-in (NCP)**

The NSX Container plug-in integrates NSX-T and OpenShift. It replaces OpenShift SDN and creates load balancer objects inside the cluster. NSX manager provides visibility into which pods are connected to which networks. NCP is deployed during the install process via previously created manifests.

#### **Describing Cluster Storage**

VMware developed the vSphere Cloud Provider to support the persistent storage requirements of containers. The vSphere Cloud Provider is a storage provider that provides volume plug-ins that are accessed by the OpenShift platform, backed by VMware vSAN or any supported vSphere Datastore. Virtual Machine File System (VMFS), Network File System (NFS) or Virtual Storage Area Network (vSAN) datastores, are available storage offerings.

**Persistent Volumes (PV)**

Persistent Volumes, or PVs, allow a cluster administrator to provision persistent storage for a cluster. PVs are resources that are not scoped to a particular project and have a life cycle that is independent of any individual pod that uses the PV. After a PV is bound to a PVC, that PV cannot then be bound to any additional PVCs. The following table displays the storage access modes available for a targeted volume plug-in.

**Table 3.2. Storage Access Modes**

| **Volume Plug-in** | **ReadWriteOnce** | **ReadOnlyMany** | **ReadWriteMany** |
| --- | --- | --- | --- |
| VMware vSphere | X |  |  |
| Cinder | X |  |  |
| NFS | X | X | X |

The accessibility of storage components and features can vary depending on the type of virtualization solution.

**Virtual Storage Area Network (vSAN)**

A virtual storage area network, or VSAN, is a hyperconverged, software-defined, storage solution. Local resources on the physical servers are pooled, or combined together. Hybrid and all-flash versions contain a flash-based cache tier for storing and retrieving the initial set of data. Each node contains a group of disks, hard drives, or solid state drives (SSDs) that defines the capacity tier.

A flash-based cache device such as SATA or SSD are pooled together across the nodes to form the vSAN data store. Attempts to write to the VSAN data store will traverse the cache first and then are flushed to the capacity tier. The data is cached as it is read into the cache tier. For example, if the same set of blocks are repeatably accessed for the VM, then you can conclude that they are staying in cache.

VSAN version 7 and above supports file and block storage. This enables the cluster administrator to create RWO and RWX PVCs from the VSAN.

**vSphere ESXi Plug-ins**

VSphere storage for Kubernetes, also referred to as vSphere Cloud Provider, is a cloud provisioner. By default, an OpenShift deployment on vSphere enables In-tree Storage.

In-tree storage integration is supported by Red Hat in vSphere 6.5 and later. OpenShift Container Platform allows the use of VMware vSphere Virtual Machine Disk (VMDK) volumes. Both VSAN and VMFS are supported volume types, and VMware vSphere volumes can be provisioned dynamically.

The vSphere disk is created by OpenShift and attached to the correct image. The deployment storage is block only and supports ReadWriteOnce. The cluster administrator can specify any data store whether block, file, or VSAN, and the provisioner creates the Virtual Machine Disk (VMDK) for that data store.

* Storage In-tree drivers
  + Formerly maintained by VMware, this is the default storage deployed when using both full-stack and pre-existing infrastructure installation methods. The plug-in creates vSphere storage using the in-tree storage drivers for vSphere included in OpenShift Container Platform, and is used when vSphere CSI drivers are not available. These drivers are supported by Red Hat on vSphere 6.5 and later.
* vSphere CSI driver
  + The Container Storage Integration (CSI) provisioner is not deployed by Red Hat. It is a Day 2 operation deployed by the customer. The plug-in creates vSphere storage using the standard Container Storage Interface. The vSphere CSI driver is provided and supported by VMware.

### NOTE

RHEL NFS does not fully support all POSIX locking and caching semantics. Testing uncovered issues with certain apps using shared storage, such as the internal registry and HA databases. Therefore, using RHEL NFS to back PVs used by core services is not recommended. However, other NFS implementations on the marketplace might not have these issues.

Contact the NFS implementation vendor regarding completed testing against these OpenShift Container Platform core components.

### REFERENCES

* For more information, refer to the Installation and update chapter of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/architecture>

## **Describing the Installation of OpenShift on vSphere Using Full-stack Automation**

### **Objectives**

* Describe the installation of OpenShift on vSphere using full-stack automation with common customizations.

### **Installing OpenShift on Using Full-stack Automation on vSphere with Common Customizations**

The cluster administrator must ensure that the specific prerequisites for vSphere are completed before deploying the cluster.

The OpenShift installer prompts the cluster administrator for values, and then uses the input from those values to configure the install-config.yaml file. The default automated install-config.yaml might not be completely suitable for your environment, however, the cluster administrator can customize the install-config.yaml file. Parameters for customization are required before deploying the cluster. Finally, the OpenShift installer creates the required vSphere resources and installs an OpenShift Cluster.

### **Verifying Prerequisites for OpenShift Full-stack Installation on vSphere**

Before to deploying an OpenShift full-stack cluster on vSphere, cluster administrators are required to verify the following prerequisites:

* Use persistent storage with ReadWriteMany access mode for the OpenShift internal registry as a Day 2 task.
* Verify that the vSphere server has only one data center and one cluster. If there is more than one resource pool, then worker nodes will not provision during installation.
* Review the general and full-stack automation prerequisites detailed in the Describing OpenShift Installation Prerequisites lecture.
* Use VMware vSphere, version 6 or 7, to deploy an OpenShift Container Platform cluster.
* Configure vCenter for the infrastructure before deploying an OpenShift cluster.

#### **Describing OpenShift Full-stack Architecture on vSphere**

The following diagram displays the architecture of a full-stack deployment on vSphere:

This workflow description assumes that you have configured the vSphere ESXi host.

* Verify that the two DNS records, API and .apps are valid.
* Use the OpenShift installer binary interactive mode to prompt for the targeted values that the installer requires to configure the install-config.yaml file.
  + After invoking the openshift-install create cluster command, provide the SSH key when the interactive installer prompts you for the key. The SSH key is essential for authenticating nodes during the installation.
  + Provide the information required by the target platform, vSphere, including the vCenter URL, username, and password.
  + Provide values for the base domain, cluster name, and the pull secret when prompted by the the installer. If you do not specify answers to the interactive queries, then the installer relies on default selections. For example, if there is only one data center or cluster, the installer will automatically use those values. You must select the appropriate data store and network for the deployment, and also ensure that the virtual IP address for API and for the \\*apps wildcard matches the entries in DNS.

The bootstrap and control plane virtual machines are configured by invoking the values specified during the OpenShift installer interactive prompts.

* The Red Hat Enterprise Linux CoreOS image is downloaded and used to create the virtual machines. After The bootstrap and control plane virtual machines are cloned, they are automatically powered on and begin the deployment.
* When bootstrapping completes, the virtual machine is automatically destroyed. Then, the control plane nodes create the worker nodes using the same template downloaded previously.
* After the worker nodes are created and the deployment completes, use the kubeadmin password to log in to the cluster.

The Ingress and API addresses require static IP addresses.

* Communication with the cluster API requires the Ingress virtual IP.
* The API requires a virtual IP for processing Ingress traffic.

Configuring DNS records for the Ingress and API static IPs is mandatory for the vCenter instance that hosts your OpenShift Container Platform cluster.

* The DNS records use the following form: <component>.<cluster\_name>.<base\_domain>.

A DHCP server is necessary for the network and provides persistent IP addresses to the cluster machines.

#### **Required vCenter Account Privileges**

vCenter requires specific access to an account configured with privileges to both read and create resources, such as the VMs.

* You can give administrator privileges to an account instead of granting specific privileges. However, unintended security consequences might result from granting more access then necessary.

Review the appropriate documentation prior to deployment.

#### **Installing OpenShift Using Full-stack on vSphere Mandatory Requirement Checklist**

The following lists the mandatory requirements that the cluster administrator should consider before deploying an OpenShift cluster using the full-stack automation method on vSphere.

**vSphere version**

The following vSphere versions are supported:

* VMware vSphere 6.5
* VMware vSphere 6.7
* VMware vSphere 7.0

**vSphere data center**

* vCenter has only one data center.
* vSphere data center has only one cluster.

**vSphere data store**

Create a data store for persistent volumes that is accessible by all the machine nodes in the data center for Day 1 installation.

**vSphere VM folder**

Create a new folder to contain all the cluster VMs.

**Resource pool**

The installer binary creates the machine in the default resource pool.

* Configuring another resource pool is **NOT** possible and must be considered when planning an OpenShift cluster deployment.

**Physical hosts**

Use separate physical hosts for the cluster machines to maintain high availability of the cluster.

**VM / VM anti affinity**

Configure anti affinity rules for control plane and infra VMs when they do not reside in the same physical host.

**DRS disabled**

VMware vMotion is intended to provide live migration of virtual machines between hosts while preventing downtime. VMware Distributed Resource Scheduler (DRS) is disabled for control plane and infra worker nodes.

* A node draining procedure is implemented before migrating compute worker nodes.
* In VMware DRS, virtual machines are migrated when triggered by the level of utilization on specific hosts.
* A VMware DRS migration might, in turn, trigger another VMware DRS migration if the OpenShift node being migrated becomes NotReady and the workloads on that node are scheduled on other nodes.

### NOTE

Currently, vMotion is not extensively tested with OpenShift. However, there is an feature enhancement request under consideration to address formally testing vMotion with OpenShift. Red Hat does not recommend the use of vMotion at this time.

**Network**

Configure DHCP, DHCP IP reservations, or static IP addresses for the cluster VMs.

**Image registry**

File storage is installed as a storage technology for high available cluster internal registry.

* NFS on RHEL is not supported.
* Others NFS implementations are supported (NetApp, HPE, DELL, etc).
* OCS using CephFS is supported.

**Monitoring & logging**

Install block storage for both monitoring and logging.

**Storage provisioning strategy**

Choose either a static or a dynamic storage provisioning strategy.

**vSphere account**

Create a service account in vSphere with the roles and permissions annotated on the vSphere Storage for Kubernetes Permission web page: <https://vmware.github.io/vsphere-storage-for-kubernetes/documentation/vcp-roles.html>

**Restart policy**

Configure the restart policy in the following order:

1. Stop compute nodes
2. Stop infra nodes
3. Stop master nodes
4. Start master nodes
5. Start infra nodes
6. Start compute nodes

#### **Installing OpenShift Using Full-stack Automation on vSphere Recommendation Checklist**

This section describes the recommendations for a full-stack installation of OpenShift on vSphere.

* Three additional worker machines (CPU: 4 cores, RAM: 24 GB, Storage: 120 GB) dedicated for infra (registry, haproxy, etc.) are provisioned.
* The other worker machines are dedicated for computing (compute machines).

#### **Installing OpenShift Using Full-stack Automation on vSphere**

* Download the installer binary, oc tools, and pull secret from the Red Hat OpenShift Cluster Manager site.
* Download the root CA certificates from vCenter and add them to your bastion VM.
* Run the openshift-install binary to create the install-config.yaml file.
* Input values when prompted by the installer binary.
* Run the installer binary to create the cluster.

### NOTE

At the time this course was written, there is an error on try.openshift.com/cloud.redhat.com that displays only UPI as available for vSphere. However, the pull-secret and tools on the pre-existing infrastructure web page are suitable for full-stack deployments as well.

* Install the openshift-install binary and oc tools on the bastion host.
* [user@bastion ~]$ **sudo -i**
* [root@bastion ~]# **OCP\_VERSION=4.6.4**
* [root@bastion ~]# **MIRROR=mirror.openshift.com/pub/openshift-v4/clients**
* [root@bastion ~]# **wget \**
* > **https://{MIRROR}/ocp/${OCP\_VERSION}/openshift-install-linux-${OCP\_VERSION}.tar.gz**
* [root@bastion ~]# **tar zxvf openshift-install-linux-${OCP\_VERSION}.tar.gz \**
* > **-C /usr/bin**
* [root@bastion ~]# **rm -f openshift-install-linux-${OCP\_VERSION}.tar.gz**
* [root@bastion ~]# **chmod +x /usr/bin/openshift-install**
* [root@bastion ~]# **openshift-install version**
* openshift-install 4.6.4
* built from commit 6e02d049701437fa81521fe981405745a62c86c5

release image quay.io/openshift-release-dev/ocp-release@sha256:668...6fc

[root@bastion ~]# **wget \**

> **https://{MIRROR}/ocp/${OCP\_VERSION}/openshift-client-linux-${OCP\_VERSION}.tar.gz**

[root@bastion ~]# **tar zxvf openshift-client-linux-${OCP\_VERSION}.tar.gz \**

> **-C /usr/bin**

[root@bastion ~]# **rm -f openshift-client-linux-${OCP\_VERSION}.tar.gz**

[root@bastion ~]# **chmod +x /usr/bin/oc**

[root@bastion ~]# **oc completion bash >/etc/bash\_completion.d/openshift**

[root@bastion ~]# **oc version**

Client Version: 4.6.4

* Generate an SSH key on the bastion host.
* [root@bastion ~]# **su - user**

[user@bastion ~]$ **ssh-keygen -f ${HOME}/.ssh/ocp46-key -N ''**

* Download and extract the root CA certificates from vCenter.
* [user@bastion ~]$ **sudo wget \**
* > **vcenter.sddc.vmwaremc.com/downloads/certs/download.zip**
* ...ouput omitted...
* Saving to: `download.zip`
* 100% [==========================>] 5,708 --.-K/s in 0s

2020-10-09 22:08:15 (781 MB/s) - `download.zip` saved [6708/6708]

[user@bastion ~]$ **sudo unzip download.zip**

Archive: download.zip

inflating: download/certs//lin/000cec1a.0

inflating: download/certs//mac/000cec1a.0

inflating: download/certs//win/000cec1a.0.crt

inflating: download/certs//lin/000cec1a.r0

inflating: download/certs//mac/000cec1a.r0

inflating: download/certs//win/000cec1a.r0.cr1

* Update the Linux certificates to enable vCenter authentication. vCenter requires an established trust relationship with the configuration host. The installer will not connect to vCenter without this trust relationship.
* [user@bastion ~]$ **sudo cp certs/lin/\* /etc/pki/ca-trust/source/anchors**

[user@bastion ~]$ **sudo update-ca-trust extract**

* Create an install directory named ocp46.

[user@bastion ~]# **mkdir ocp46**

* Run the openshift-install binary to create the install-config.yaml file.
* [user@demo ~]$ **./openshift-install create install-config --dir=ocp46 \**
* > **--log-level=info**
* ? SSH Public Key /user/.ssh/ocp4ipi.pub
* ? Platform vsphere
* ? vCenter vcenter.vmwarecloud.com
* ? Usernanme cloudadmin@vmclocal.com
* Info connecting to vcenter vcenter.vmwarecloud.com
* Info Defaulting to only available datacenter: SDDC
* Info Defaulting to only available cluster: cluster1
* ? Default Datastore: WorkloadData
* ? Network network-segment1
* ? Virtual IP Address for API 192.168.1.100
* ? Virtual IP Address for Ingress 192.168.1.110
* ? Base Domain example.com
* ? Cluster Name ocp4

? Pull Secret ...output omitted...

* Run the openshift-install binary to deploy the cluster.
* [user@bastion ~]$ **./openshift-install create cluster --dir=ocp46 \**

> **--log-level=debug**

**Video**

### REFERENCES

* For more information, refer to the Installing on vSphere chapter of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing_on_vsphere/index>

## **Describing the Installation of OpenShift on vSphere Using Pre-existing Infrastructure**

### **Objectives**

* Describe the installation of OpenShift on vSphere using pre-existing infrastructure with common customizations.

### **Installing OpenShift Using Pre-existing Infrastructure on vSphere**

The administrator must ensure that the specific prerequisites for the vSphere infrastructure are completed before deploying the cluster.

### **Describing OpenShift Architecture Using Pre-existing Infrastructure on vSphere**

The administrator is responsible for the architecture resources displayed in the following diagram of an OpenShift deployment using pre-existing infrastructure on vSphere:

|  |
| --- |
| A diagram of a computer  Description automatically generated |

The following diagram displays the different phases of the installation process using pre-existing infrastructure.

|  |
| --- |
|  |

### **Describing the Deployment Workflow Using Pre-existing Infrastructure on vSphere**

The following diagram illustrates the workflow for deploying an OpenShift architecture using pre-existing infrastructure on vSphere.

|  |
| --- |
|  |

This section provides an overview of the process steps for installing OpenShift using pre-existing infrastructure on vSphere.

Notice that the numbers on the preceding workflow diagram align to the following numbered steps.

1. Power on the bootstrap node to initialize the RHCOS template image.
2. The bootstrap node receives an IP address from the DHCP server. The DNS server contains an entry mapping the bootstrap host name to the appropriate IP address.
3. The vApp specifies the base64 encoded URL to the bootstrap ignition configuration file located on the HTTPD server.
4. The bootstrap node downloads the ignition file located at the URL specified in the vApp.
5. Using the information specified in the ignition configuration file, the control plane nodes begin installing RHCOS and receive the ignition files from the Kubernetes API MCS running on the bootstrap node.

On each of the control plane VM images:

1. The RHCOS template image completes and the control plane node boots.
2. The control plane node receives an IP address from the DHCP server.
3. The vApp specifies the base64 encoded ignition configuration information used to install the control plane nodes.
4. The control plane nodes fetch their ignition configuration files from the bootstrap node.
5. The compute nodes fetch their ignition configuration files from the Kubernetes API MCS, running on the control plane nodes after the control plane is transferred from the bootstrap node to the control plane.

#### **Installing OpenShift Using Pre-existing Infrastructure in vSphere Mandatory Requirement Checklist**

This section describes the requirements for a pre-existing infrastructure installation of OpenShift on vSphere.

**Machines**

The following machines are provisioned:

* One bootstrap machine (CPU: 4 cores, RAM: 16 GB, Storage: 120 GB) - This machine is removed after installation.
* Three control plane machines (CPU: 4 cores, RAM: 16 GB, Storage: 120 GB)
* Two compute machines (CPU: 2 cores, RAM: 8 GB, Storage: 120 GB)

**vSphere version**

The following vSphere versions are supported:

* VMware vSphere 6.5
* VMware vSphere 6.7
* VMware vSphere 7.0

**Network**

Each VM must have DHCP access, DHCP IP reservations, or static IP addresses configured.

**Network ports**

For inbound traffic, the following ports are opened:

* All machines:
  + ICMP
  + TCP: 22
* All machines except bastion:
  + TCP: 9000-9999, 10249-10259, 30000-32767
  + UDP: 4789, 6081, 9000-9999, 30000-32767
  + Control plane and bootstrap:
  + TCP: 2379-2380, 6443, 22623
* Workers:
  + TCP: 443. If optional infra worker machine configuration chosen, open 443 only to infra worker

For outbound traffic, all ports are opened.

**API load balancer**

* One highly-available logical layer-4 load balancer is configured with 2 virtual IPs (VIPs):
  + VIP1 (External API): Dispatch the traffic to all control plane nodes IPs and the bootstrap node IP with the following port mapping: 6443 → 6443
  + VIP2 (Internal API): Dispatch the traffic to all control plane nodes IPs and the bootstrap node IP with the following port mapping: 6443 → 6443, 22623 → 22623
* Redirection to the bootstrap node IP is removed during installation.
* Layer-7 configured load balancers are not supported.
* Both VIP health checks are configured for the following
  + The endpoint for the API server is ready.
  + The timeout is set to 30 seconds before the removal of the API server instance.
  + Probing occurs every 10 seconds, requiring two successful requests to become healthy and three unsuccessful requests to become unhealthy.
* The load balancing algorithm is stateless.

**Apps load balancer**

* One highly available logical layer-4 load balancer is configured with one VIP.
  + The applications VIP dispatches the traffic to all compute nodes IP with the following port mapping: 443 → 443
* Redirection to non-infra nodes is removed after the infra node configuration.
* Layer-7 configured load balancers are not supported.
* The basic VIP health check is configured.
* The load balancing algorithm is stateless.

**External DNS records**

The following external DNS records are required:

* API:
  + api.<cluster\_name>.<base\_domain>: VIP1
  + \\*.apps.<cluster\_name>.<base\_domain>: VIP2

The DNS records must be resolvable by both clients external to the cluster and from all the nodes within the cluster.

**Internal DNS records**

The following are the required internal DNS records:

* API:
  + api-int.<cluster\_name>.<base\_domain>: VIP3

The DNS records must be resolvable from all the nodes within the cluster.

**HTTP server**

The HTTP server must be accessible from both the bastion machine running the installation and all the cluster machines that are installed.

* If the HTTP server is not available, use the bastion machine as an HTTP server.
* Ensure that port 80 is opened in the bastion machine for inbound traffic.

**Red Hat Enterprise Linux CoreOS**

Provision the Red Hat Enterprise Linux CoreOS VMs from the Red Hat Enterprise Linux CoreOS OVA file or ISO with iPXE files provided by Red Hat.

**Latency sensitivity**

Set Latency Sensitivity to **High** for all VMs, except for the bastion machine.

**Image registry storage**

If file storage is installed as a storage technology for the internal registry on a highly-available cluster, then:

* NFS on RHEL is not supported.
* Other NFS implementations are supported, for example, NetApp, HPE, and DELL.
* OCS using CephFS is supported.

**Storage provisioning strategy**

Select either static or dynamic for the storage provisioning strategy.

**vSphere data store**

Create a data store for persistent volumes that are accessible by all machine nodes in the data center for Day 1 installation.

**vSphere VM folder**

Create a folder that contains all the VMs that are created for the deployment.

**Physical hosts**

Use separate physical hosts for the cluster machines to maintain high availability of the cluster.

**VM / VM anti affinity**

Configure anti affinity rules for control plane and infra VMs when they do not reside in the same physical host.

**DRS disabled**

VMware vMotion is intended to provide live migration of virtual machines between hosts while preventing downtime. VMware Distributed Resource Scheduler (DRS) is disabled for control plane and infra worker nodes.

* A node draining procedure is implemented before migrating compute worker nodes.
* In VMware DRS, virtual machines are migrated when triggered by the level of utilization on specific hosts.
* A VMware DRS migration might, in turn, trigger another VMware DRS migration if the OpenShift node being migrated becomes NotReady and the workloads on that node are scheduled on other nodes.

### NOTE

Currently, vMotion is not extensively tested with OpenShift. However, there is an feature enhancement request under consideration to address formally testing vMotion with OpenShift. Red Hat does not recommend the use of vMotion at this time.

**vSphere account**

Create a service account in vSphere with the roles/permissions annotated on vSphere Storage for Kubernetes Permission web page: <https://vmware.github.io/vsphere-storage-for-kubernetes/documentation/vcp-roles.html>

**Restart policy**

Configure the restart policy in the following order:

1. Stop compute nodes
2. Stop infra nodes
3. Stop control plane nodes
4. Start control plane nodes
5. Start infra nodes
6. Start compute nodes

#### **Installing an OpenShift Cluster on vSphere with Pre-existing Infrastructure.**

* Download the installer binary, oc tools, and pull secret from the Red Hat OpenShift Cluster Manager site.
* Download the root CA certificates for vCenter and add them to your bastion VM.
* Download the bare metal install image.
* Verify DNS, HAproxy, and Apache Web server configurations.
* Manually create the install-config.yaml file.
* Create the manifest files.
* Modify the manifest/cluster-schedular-02 file to reflect uppercase False.
* Create the ignition files.
* Make the ignition files available via HTTPD.
* Create append-bootstrap.ign to point to the URL for bootstrap.ign.
* Convert the ignition files using base64 encoding.
* Download the RHCOS.ova file and deploy the template.
* Clone the template to create cluster VMs.
* Add base64 encoding of ignition files to the corresponding VMs.
* Power on the VM and pass kernel line arguments to customize the cluster VMs.
* Use the openshift-install binary to monitor the bootstrapping process.
* Remove the bootstrap entry from the load balancer after bootstrapping completes.
* Use the openshift-install binary to monitor the deployment progress until completion.

**Video**

### REFERENCES

* For more information, refer to the Installing vSphere guide of the Red Hat OpenShift Container Platform 4.6 documentation at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/installing_on_vsphere/index>

## **Guided Exercise: Performing Day 1 and Day 2 Operations**

* Perform a basic configuration before releasing an OpenShift cluster to production.

**Outcomes**

You should be able to:

* Save a copy of the kubeconfig file.
* Configure a dynamic storage provider.
* Configure the registry operator, adding persistent storage.
* Deploy an application to perform a functional test of the cluster.

To perform this exercise, ensure you have completed the install-perform-practice guided exercise.

**Procedure 6.1. Instructions**

1. Save the kubeconfig file.
   1. As the student user, copy the kubeconfig file from /home/lab/ocp4upi/auth/kubeconfig to the workstation machine.
   2. [student@workstation ~]$ **ssh lab@utility**
   3. [lab@utility ~]$ **scp /home/lab/ocp4upi/auth/kubeconfig student@workstation:**
   4. The authenticity of host 'workstation (172.25.250.9)' can't be established.
   5. ECDSA key fingerprint is SHA256:DDmgpJODDjGdZHJRAhDuN6XyPrL43F4IXMYhj5orlnQ.
   6. Are you sure you want to continue connecting (yes/no/[fingerprint])? **yes**
   7. Warning: Permanently added 'workstation,172.25.250.9' (ECDSA) to the list of known hosts.
   8. student@workstation's password: **student**

kubeconfig 100% 8942 5.2MB/s 00:00

1. Configure the image registry with persistent storage.
   1. Use the kubeconfig file to authenticate to OpenShift.

[lab@utility ~]$ **export KUBECONFIG=~/ocp4upi/auth/kubeconfig**

* 1. Investigate the NFS exports on the utility server.
  2. [lab@utility ~]$ **cat /etc/exports**
  3. **/exports** \*(rw,sync,no\_wdelay,no\_root\_squash,insecure,fsid=0)
  4. [lab@utility ~]$ **lsblk**
  5. NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
  6. vda 252:0 0 10G 0 disk
  7. ├─vda1 252:1 0 1M 0 part
  8. ├─vda2 252:2 0 100M 0 part /boot/efi
  9. └─vda3 252:3 0 9,9G 0 part /
  10. vdb 252:16 0 40G 0 disk
  11. └─vdb1 252:17 0 **40G** 0 part **/exports**
  12. [lab@utility ~]$ **df -h**
  13. Filesystem Size Used Avail Use% Mounted on
  14. ...output omitted...
  15. /dev/vdb1 40G 318M **40G** 1% **/exports**
  16. /dev/vda2 100M 6,8M 94M 7% /boot/efi

tmpfs 183M 0 183M 0% /run/user/1000

The /exports NFS export has approximately 40 GB of disk space available.

* 1. Create the persistent volume (PV) file named pv.yaml.
  2. [lab@utility ~]$ **vi pv.yaml**
  3. apiVersion: v1
  4. kind: **PersistentVolume**
  5. metadata:
  6. name: **registry-pv**
  7. spec:
  8. capacity:
  9. storage: **5Gi**
  10. accessModes:
  11. - ReadWriteMany
  12. nfs:
  13. path: **/exports/registry**
  14. server: **192.168.50.254**

persistentVolumeReclaimPolicy: Recycle

Save the file and then close the editor. You can retrieve that file by running the wget http://classroom.example.com/materials/solutions/etherpad/pv.yaml command.

* 1. Create the /exports/registry folder with the appropriate permissions and the persistent volume (PV) from the pv.yaml file:

[lab@utility ~]$ **mkdir /exports/registry**

[lab@utility ~]$ **sudo chmod 777 /exports/registry/**

[lab@utility ~]$ **oc create -f pv.yaml**

persistentvolume/registry-pv created

* 1. Create a file named pvc.yaml with the following content:
  2. [lab@utility ~]$ **vi pvc.yaml**
  3. kind: PersistentVolumeClaim
  4. apiVersion: v1
  5. kind: PersistentVolumeClaim
  6. metadata:
  7. name: **registry-claim**
  8. namespace: **openshift-image-registry**
  9. spec:
  10. accessModes:
  11. - ReadWriteMany
  12. resources:
  13. requests:

storage: **5Gi**

Save the file and then close the editor. You can retrieve that file by running the wget http://classroom.example.com/materials/solutions/etherpad/pvc.yaml command.

* 1. Create the persistent volume claim (PVC) from the pvc.yaml file:
  2. [lab@utility ~]$ **oc create -f pvc.yaml**

persistentvolumeclaim/registry-claim created

* 1. In the spec section, set the image registry operator in a "Managed" state. Edit the configuration of the cluster image registry to add the PVC. Also, configure the image registry to have two pod replicas.
  2. [lab@utility ~]$ **oc edit configs.imageregistry/cluster**
  3. ...output omitted...
  4. spec:
  5. ...output omitted...
  6. **managementState: Managed**
  7. ...output omitted...
  8. proxy: {}
  9. **replicas: 2**
  10. requests:
  11. ...output omitted...
  12. rolloutStrategy: RollingUpdate
  13. **storage:**
  14. **pvc:**
  15. **claim: registry-claim**

...output omitted...

### NOTE

Ensure you are editing the spec section, not the status section. Editing the status section does not have any effect in the configuration.

* 1. Check the PV, PVC, and image registry operator status.
  2. [lab@utility ~]$ **oc get pv -A**
  3. NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE
  4. registry-pv **5Gi** RWX Recycle **Bound** openshift-image-registry/registry-claim 111m
  5. [lab@utility ~]$ **oc get pvc -A**
  6. NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE
  7. openshift-image-registry registry-claim **Bound** registry-pv **5Gi** RWX 105m
  8. [lab@utility ~]$ **oc get clusteroperator**
  9. NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE
  10. authentication 4.6.4 True False False 174m
  11. cloud-credential 4.6.4 True False False 3d
  12. cluster-autoscaler 4.6.4 True False False 3d
  13. config-operator 4.6.4 True False False 3d
  14. console 4.6.4 True False False 175m
  15. csi-snapshot-controller 4.6.4 True False False 3d
  16. dns 4.6.4 True False False 3d
  17. etcd 4.6.4 True False False 3d
  18. **image-registry** 4.6.4 **True** **False** **False** 99m

...output omitted...

* 1. Check the number of registry pods in the openshift-image-registry namespace.
  2. [lab@utility ~]$ **oc get pods -n openshift-image-registry -o wide**
  3. NAME READY STATUS RESTARTS AGE IP NODE
  4. cluster-image-registry-operator-8455b85cc6-z7wcv 1/1 Running 1 78m 10.130.0.18 master03 ...
  5. **image-registry-76467959f4-kmdf8** 1/1 Running 0 8m38s 10.131.0.21 **worker01** ...
  6. **image-registry-76467959f4-tcczp** 1/1 Running 0 9m45s 10.128.2.39 **worker02** ...
  7. node-ca-dczt7 1/1 Running 0 66m 192.168.50.14 worker02 ...
  8. node-ca-gsbtk 1/1 Running 0 67m 192.168.50.13 worker01 ...
  9. node-ca-mpbkm 1/1 Running 0 76m 192.168.50.12 master03 ...
  10. node-ca-xjbp8 1/1 Running 0 76m 192.168.50.10 master01 ...

node-ca-xwfsf 1/1 Running 0 76m 192.168.50.11 master02 ...

1. Configure a dynamic storage provisioner.
   1. Clone the kubernetes-sigs/nfs-subdir-external-provisioner GitHub repository:
   2. [lab@utility ~]$ **git clone \**
   3. > **https://github.com/kubernetes-sigs/nfs-subdir-external-provisioner/**

...output omitted...

### WARNING

The nfs-subdir-external-provisioner is not supported by Red Hat nor recommended for production environments.

* 1. Edit the files in the deploy folder using information for the NFS server in utility. The output should be as follows:
  2. [lab@utility ~]$ **vi nfs-subdir-external-provisioner/deploy/class.yaml**
  3. apiVersion: storage.k8s.io/v1
  4. kind: StorageClass
  5. metadata:
  6. name: nfs-client
  7. **annotations:**
  8. **storageclass.kubernetes.io/is-default-class: "true"**
  9. provisioner: **nfs-dynamic-provisioner**
  10. **reclaimPolicy: Retain**

...output omitted...

[lab@utility ~]$ **vi nfs-subdir-external-provisioner/deploy/rbac.yaml**

apiVersion: v1

kind: ServiceAccount

metadata:

name: nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

---

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: nfs-client-provisioner-runner

rules:

- apiGroups: [""]

resources: ["persistentvolumes"]

verbs: ["get", "list", "watch", "create", "delete"]

- apiGroups: [""]

resources: ["persistentvolumeclaims"]

verbs: ["get", "list", "watch", "update"]

- apiGroups: ["storage.k8s.io"]

resources: ["storageclasses"]

verbs: ["get", "list", "watch"]

- apiGroups: [""]

resources: ["events"]

verbs: ["create", "update", "patch"]

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: run-nfs-client-provisioner

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

roleRef:

kind: ClusterRole

name: nfs-client-provisioner-runner

apiGroup: rbac.authorization.k8s.io

---

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

rules:

- apiGroups: [""]

resources: ["endpoints"]

verbs: ["get", "list", "watch", "create", "update", "patch"]

---

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: leader-locking-nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

subjects:

- kind: ServiceAccount

name: nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

roleRef:

kind: Role

name: leader-locking-nfs-client-provisioner

apiGroup: rbac.authorization.k8s.io

[lab@utility ~]$ **vi nfs-subdir-external-provisioner/deploy/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nfs-client-provisioner

labels:

app: nfs-client-provisioner

namespace: **nfs-dynamic-namespace**

spec:

replicas: 1

strategy:

type: Recreate

selector:

matchLabels:

app: nfs-client-provisioner

template:

metadata:

labels:

app: nfs-client-provisioner

spec:

serviceAccountName: nfs-client-provisioner

containers:

- name: nfs-client-provisioner

image: registry.k8s.io/sig-storage/nfs-subdir-external-provisioner:v4.0.2

volumeMounts:

- name: nfs-client-root

mountPath: /persistentvolumes

env:

- name: PROVISIONER\_NAME

value: **nfs-dynamic-provisioner**

- name: NFS\_SERVER

value: **192.168.50.254**

- name: NFS\_PATH

value: **/exports**

volumes:

- name: nfs-client-root

nfs:

server: **192.168.50.254**

path: **/exports**

* 1. Create the objects in the nfs-dynamic-namespace namespace.
  2. [lab@utility ~]$ **oc create namespace nfs-dynamic-namespace**

namespace/nfs-dynamic-namespace created

[lab@utility ~]$ **oc create -f nfs-subdir-external-provisioner/deploy/rbac.yaml**

serviceaccount/nfs-client-provisioner created

clusterrole.rbac.authorization.k8s.io/nfs-client-provisioner-runner created

clusterrolebinding.rbac.authorization.k8s.io/run-nfs-client-provisioner created

role.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner created

rolebinding.rbac.authorization.k8s.io/leader-locking-nfs-client-provisioner created

* 1. Create the use-scc-hostmount-anyuid role in the nfs-dynamic-namespace namespace. Add the use-scc-hostmount-anyuid role to the nfs-client-provisioner service account.
  2. [lab@utility ~]$ **oc create role use-scc-hostmount-anyuid --verb=use \**
  3. > **--resource=scc --resource-name=hostmount-anyuid -n nfs-dynamic-namespace**

role.rbac.authorization.k8s.io/use-scc-hostmount-anyuid created

[lab@utility ~]$ **oc project nfs-dynamic-namespace**

Now using project "nfs-dynamic-namespace" on server "https://api.ocp4.example.com:6443".

[lab@utility ~]$ **oc adm policy add-role-to-user use-scc-hostmount-anyuid \**

> **-z nfs-client-provisioner --role-namespace='nfs-dynamic-namespace'**

role.rbac.authorization.k8s.io/use-scc-hostmount-anyuid added: "nfs-client-provisioner"

* 1. Create the deployment.
  2. [lab@utility ~]$ **oc create -f \**
  3. > **nfs-subdir-external-provisioner/deploy/deployment.yaml**

deployment.apps/nfs-client-provisioner created

* 1. Verify all the resources created in the nfs-dynamic-namespace namespace.
  2. [lab@utility ~]$ **oc get all**
  3. NAME READY STATUS RESTARTS AGE
  4. pod/nfs-client-provisioner-69785747-h66hh 1/1 Running 0 25s
  5. NAME READY UP-TO-DATE AVAILABLE AGE
  6. deployment.apps/nfs-client-provisioner 1/1 1 1 25s
  7. NAME DESIRED CURRENT READY AGE

replicaset.apps/nfs-client-provisioner-69785747 1 1 1 25s

* 1. Create the NFS StorageClass.
  2. [lab@utility ~]$ **oc create -f \**
  3. > **nfs-subdir-external-provisioner/deploy/class.yaml**

storageclass.storage.k8s.io/nfs-client created

[lab@utility ~]$ **oc get storageclass**

NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE

nfs-client (default) nfs-dynamic-provisioner Retain Immediate false 46s

* 1. Test the NFS dynamic volume provisioning.
  2. [lab@utility ~]$ **oc create -f \**
  3. > **nfs-subdir-external-provisioner/deploy/test-claim.yaml**

persistentvolumeclaim/test-claim created

[lab@utility ~]$ **oc get pvc**

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

test-claim Bound pvc-7c628580-6de8-416c-8750-9e3d315e2e69 1Mi RWX nfs-client 17s

[lab@utility ~]$ **oc create -f \**

> **nfs-subdir-external-provisioner/deploy/test-pod.yaml**

pod/test-pod created

[lab@utility ~]$ **oc get all**

NAME READY STATUS RESTARTS AGE

pod/nfs-client-provisioner-69785747-h66hh 1/1 Running 0 13m

**pod/test-pod** 0/1 **Completed** 0 9s

NAME READY UP-TO-DATE AVAILABLE AGE

deployment.apps/nfs-client-provisioner 1/1 1 1 13m

NAME DESIRED CURRENT READY AGE

replicaset.apps/nfs-client-provisioner-69785747 1 1 1 13m

* 1. If the NFS dynamic provisioner is working properly, there will be a SUCCESS file in the NFS export folder.
  2. [lab@utility ~]$ **tree /exports/**
  3. /exports/
  4. └── nfs-dynamic-namespace-test-claim-pvc-7c628580-6de8-416c-8750-9e3d315e2e69
  5. │ └── **SUCCESS**
  6. └── registry

2 directories, 1 file

### NOTE

If the creation of the test PVC or test pod fails, then verify the changes made to the YAML files in the previous steps. If you forgot to change a parameter, then delete the resources using the oc delete -f command and recreate them using the correct YAML file.

1. Perform a functional test of the cluster.
   1. Create a new project named etherpad and set it as the working project.
   2. [lab@utility ~]$ **oc new-project etherpad**

...output omitted...

* 1. Create a new folder named etherpad and change into it.
  2. [lab@utility ~]$ **mkdir etherpad**
  3. [lab@utility ~]$ **cd etherpad**

[lab@utility etherpad]$

* 1. Download the following files and save the content into the etherpad folder.
  2. [lab@utility etherpad]$ **SOL\_URL=http://classroom.example.com/materials/solutions**
  3. [lab@utility etherpad]$ **wget ${SOL\_URL}/etherpad/etherpad-svc.yaml**
  4. ...output omitted...
  5. [lab@utility etherpad]$ **cat etherpad-svc.yaml**
  6. apiVersion: v1
  7. kind: Service
  8. metadata:
  9. name: etherpad
  10. labels:
  11. app.kubernetes.io/name: etherpad
  12. app.kubernetes.io/version: "latest"
  13. spec:
  14. type: ClusterIP
  15. ports:
  16. - port: 9001
  17. targetPort: http
  18. protocol: TCP
  19. name: http
  20. selector:

app.kubernetes.io/name: etherpad

[lab@utility etherpad]$ **wget ${SOL\_URL}/etherpad/etherpad-route.yaml**

...output omitted...

[lab@utility etherpad]$ **cat etherpad-route.yaml**

apiVersion: route.openshift.io/v1

kind: Route

metadata:

annotations:

openshift.io/host.generated: "true"

name: etherpad

labels:

app.kubernetes.io/name: etherpad

app.kubernetes.io/version: "latest"

spec:

host:

port:

targetPort: http

to:

kind: Service

name: etherpad

weight: 100

tls:

insecureEdgeTerminationPolicy: Redirect

termination: edge

[lab@utility etherpad]$ **wget ${SOL\_URL}/etherpad/etherpad-pvc.yaml**

...output omitted...

[lab@utility etherpad]$ **cat etherpad-pvc.yaml**

kind: PersistentVolumeClaim

apiVersion: v1

metadata:

name: etherpad

labels:

app.kubernetes.io/name: etherpad

app.kubernetes.io/version: "latest"

spec:

accessModes:

- "ReadWriteOnce"

resources:

requests:

storage: "1Gi"

[lab@utility etherpad]$ **wget ${SOL\_URL}/etherpad/etherpad-deployment.yaml**

...output omitted...

[lab@utility etherpad]$ **cat etherpad-deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: etherpad

labels:

app.kubernetes.io/name: etherpad

app.kubernetes.io/version: "latest"

spec:

replicas: 1

selector:

matchLabels:

app.kubernetes.io/name: etherpad

template:

metadata:

labels:

app.kubernetes.io/name: etherpad

spec:

securityContext:

{}

containers:

- env:

- name: TITLE

value: DO322 Etherpad

- name: DEFAULT\_PAD\_TEXT

value: Etherpad for sharing ideas between the students.

name: etherpad

securityContext:

{}

image: "quay.io/redhattraining/etherpad:latest"

imagePullPolicy: IfNotPresent

ports:

- name: http

containerPort: 9001

protocol: TCP

livenessProbe:

httpGet:

path: /

port: http

readinessProbe:

httpGet:

path: /

port: http

resources:

{}

volumeMounts:

- name: etherpad-data

mountPath: /opt/etherpad-lite/var

volumes:

- name: etherpad-data

persistentVolumeClaim:

claimName: etherpad

* 1. Create the OpenShift resources using the files that you created in the etherpad folder.
  2. [lab@utility etherpad]$ **oc create -f etherpad-pvc.yaml**

persistentvolumeclaim/etherpad created

[lab@utility etherpad]$ **oc create -f etherpad-svc.yaml**

service/etherpad created

[lab@utility etherpad]$ **oc create -f etherpad-route.yaml**

route.route.openshift.io/etherpad created

[lab@utility etherpad]$ **oc create -f etherpad-deployment.yaml**

deployment.apps/etherpad created

* 1. Verify the creation of the resources:
  2. [lab@utility etherpad]$ **oc get all**
  3. NAME READY STATUS RESTARTS AGE
  4. pod/etherpad-c7476d8d8-8b8j5 1/1 Running 0 25s
  5. NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
  6. service/etherpad ClusterIP 172.30.15.113 <none> 9001/TCP 31s
  7. NAME READY UP-TO-DATE AVAILABLE AGE
  8. deployment.apps/etherpad 1/1 1 0 25s
  9. NAME DESIRED CURRENT READY AGE
  10. replicaset.apps/etherpad-c7476d8d8 1 1 1 25s
  11. NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD
  12. route.route.openshift.io/etherpad etherpad-etherpad.apps.ocp4.example.com etherpad http edge/Redirect None
  13. [lab@utility etherpad]$ **oc get events -o template --template \**
  14. > **'{{range .items}}{{.message}}{{"\n"}}{{end}}'**
  15. ...output omitted...
  16. Created pod: etherpad-c7476d8d8-pnv4r
  17. **waiting for a volume to be created, either by external provisioner "nfs-dynamic-provisioner" or manually created by system administrator**
  18. **External provisioner is provisioning volume for claim "etherpad/etherpad"**
  19. Successfully provisioned volume pvc-17d90bb5-ae24-46e5-a264-4e057383cdd4

...output omitted...

The PV was automatically created by the nfs-dynamic-provisioner.

* 1. Open the Firefox web browser on the workstation machine and navigate to the route etherpad-etherpad.apps.ocp4.example.com.

### NOTE

When prompted by Firefox about a potential security risk, click **Advanced** and then click **Accept the Risk and Continue**.

A screenshot of a computer

Description automatically generated

* 1. Create a new etherpad named "DO322". Remove the text in the etherpad and type: "DO322 etherpad".

A screenshot of a computer

Description automatically generated

* 1. In the terminal, delete the etherpad pod and verify that the database is preserved.
  2. [lab@utility etherpad]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE

etherpad-**c7476d8d8-pnv4r** 1/1 Running 0 23m

[lab@utility etherpad]$ **oc delete pod etherpad-c7476d8d8-pnv4r**

pod "etherpad-`c7476d8d8-pnv4r`" deleted

Verify the creation of the new pod:

[lab@utility etherpad]$ **oc get pods**

NAME READY STATUS RESTARTS AGE

etherpad-**c7476d8d8-sfvsj** 1/1 Running 0 26s

* 1. Return to the Firefox web browser. Reload the web page and verify that the text "DO322 etherpad" is still present.

1. Return to the lab home folder and clean up the environment:
2. [lab@utility etherpad]$ **cd ~**
3. [lab@utility ~]$ **oc delete project etherpad**

project.project.openshift.io "etherpad" deleted

**Finish**

Do not make any other changes to the lab environment until the next guided exercise. You will continue using it in later guided exercises.

This concludes the guided exercise.

## **Replacing a Control Plane Node**

### **Objectives**

* Backup and restore a control plane node.

### **Explaining Etcd Backups**

The OpenShift etcd database is a key-value store containing the cluster status data. To restore an OpenShift cluster to a previous state, administrators first must perform an etcd backup.

An etcd backup contains the status of the cluster and the configuration of the static pods running on a control plane node. It is not necessary to save a backup from each of the three control plane nodes. The backup of one of the control plane nodes is enough.

Performing regular etcd backups is necessary to ensure that a cluster can be recovered if all control plane nodes are lost, or the cluster becomes unstable because of human error or a hardware failure.

### NOTE

The default etcd space quota size is approximately 7.5 GB, and is defined by the variable etcd\_server\_quota\_backend\_bytes. If an etcd database exceeds this quota, you must defragment and compact it.

For more information, refer to the Recommended etcd practices section of the Scalability and Performance Guide at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/scalability_and_performance/index#recommended-etcd-practices_>

### **Restoring the Cluster to a Previous State**

OpenShift provides automatic healing capabilities because of the nature of Kubernetes operators. Sometimes, such as when a cluster is misconfigured or data corrupted, an OpenShift cluster might not be able to heal itself. If the cluster cannot self-heal, OpenShift enables restoring the cluster to a previous state using an etcd backup.

The restore procedure of an OpenShift cluster requires:

* SSH access to the control plane nodes
* A user with cluster-admin permissions
* An etcd backup

You can find more information about disaster recovery in the Disaster recovery chapter of the Red Hat OpenShift Container Platform Backup and Restore Guide at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/backup_and_restore/index#disaster-recovery>

### **Restoring a Lost Control Plane Node**

In production environments, it is possible to lose one or more control plane nodes due to human error or a hardware failure.

To replace an unhealthy control plane node or etcd member, the cluster administrators must use the same machine configuration to create the new node as was used to create the unhealthy node being replaced.

The machine configuration is stored in different formats depending on the original installation method. For Full-stack Automation OpenShift installations, retrieve the machine configuration from the OpenShift machine API. For installations on Pre-existing Infrastructure, the ignition files are required.

Elsewhere in this chapter, you will learn how to redeploy a lost control plane node using the classroom environment.

### NOTE

In the future, Red Hat plans to include automatic handling and replacement of lost control plane members by the etcd operator.

### REFERENCES

For more information, refer to the Post-Installation Cluster Tasks chapter in the Red Hat OpenShift Container Platform Post-Installation Configuration Guide at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/post-installation_configuration/>

For more information, you can also refer to the Red Hat OpenShift Container Platform Backup and Restore Guide at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/backup_and_restore/>

## **Guided Exercise: Replacing a Control Plane Node**

* Reinstall a failed control plane node.

**Outcomes**

You should be able to:

* Generate an etcd backup.
* Recover a failed control plane node.

To perform this exercise, ensure that you have completed all prior exercises in this course.

The classroom environment is unchanged, apart from the course exercises.

If you have made any additional changes to the environment, recreate a new classroom environment before you begin. You must repeat the steps from the guided exercises in the previous chapters to prepare for this exercise.

**Procedure 6.2. Instructions**

1. Use the kubeconfig file to authenticate into OpenShift from the utility server.
   1. From the workstation machine, log into the utility server as the lab user:
   2. [student@workstation ~]$ **ssh lab@utility**

...output omitted...

* 1. Export the KUBECONFIG variable to authenticate into OpenShift:

[lab@utility ~]$ **export KUBECONFIG=~/ocp4upi/auth/kubeconfig**

1. Create the etcd backup.
   1. Start a debug session in the master01 control plane node. Change the root directory to /host.
   2. [lab@utility ~]$ **oc get nodes**
   3. NAME STATUS ROLES AGE VERSION
   4. **master01** Ready master 2d18h v1.19.0+9f84db3
   5. master02 Ready master 2d18h v1.19.0+9f84db3
   6. master03 Ready master 2d18h v1.19.0+9f84db3
   7. worker01 Ready worker 2d17h v1.19.0+9f84db3
   8. worker02 Ready worker 2d17h v1.19.0+9f84db3
   9. [lab@utility ~]$ **oc debug node/master01**
   10. Creating debug namespace/openshift-debug-node-sws5d ...
   11. Starting pod/master01-debug ...
   12. To use host binaries, run `chroot /host`
   13. Pod IP: 192.168.50.10
   14. If you don't see a command prompt, try pressing enter.
   15. sh-4.4# **chroot /host**

sh-4.4#

* 1. Run the etcd backup script in /usr/local/bin/cluster-backup.sh. Save the output in /home/core/backup.
  2. sh-4.4# **/usr/local/bin/cluster-backup.sh /home/core/backup**
  3. ...output omitted...

snapshot db and kube resources are successfully saved to /home/core/backup

* 1. List the resources created by the backup script.
  2. sh-4.4# **ls /home/core/backup/**

snapshot\_2020-12-03\_110352.db static\_kuberesources\_2020-12-03\_110352.tar.gz

* + - The snapshot\_2020-12-03\_110352.db file is the etcd snapshot.
    - The static\_kuberesources\_2020-12-03\_110352.tar.gz contains the static pods resources: kube-apiserver, kube-controller-manager, kube-scheduler and etcd.
  1. Change the backup folder permissions so that the new owner is the core user and the core group.

sh-4.4# **chown -R core:core /home/core/backup**

* 1. Close the debug session.
  2. sh-4.4# **exit**
  3. exit
  4. sh-4.4# **exit**
  5. exit
  6. Removing debug pod ...

Removing debug namespace/openshift-debug-node-85dg7 ...

* 1. Create a new folder, /home/lab/etcd\_backup, and save the backup files to the utility server. Use the ssh key ocp4upi that you generated as a prerequisite. Type yes to establish the connection to the master01 server.
  2. [lab@utility ~]$ **mkdir /home/lab/etcd\_backup**
  3. [lab@utility ~]$ **scp -i .ssh/ocp4upi \**
  4. > **core@192.168.50.10:/home/core/backup/\* /home/lab/etcd\_backup/**
  5. The authenticity of host '192.168.50.10 (192.168.50.10)' can't be established.
  6. ECDSA key fingerprint is SHA256:lK205E7PCvFYMAcnHH26Sy1wHowbuEN1bACvyu+WjCU.
  7. Are you sure you want to continue connecting (yes/no/[fingerprint])? **yes**
  8. Warning: Permanently added '192.168.50.10' (ECDSA) to the list of known hosts.
  9. snapshot\_2020-12-16\_105145.db 100% 73MB 181.3MB/s 00:00

static\_kuberesources\_2020-12-16\_105145.tar.gz 100% 65KB 52.1MB/s 00:00

1. Remove a control plane node.

### NOTE

This procedure for recovering a failed control plane member is valid when the failed machine is not running or in a NotReady status.

If the machine is running and the control plane member is listed as Ready, but the etcd pod is in an Error status, you must follow the procedure described in the Replacing an unhealthy etcd member whose etcd pod is crashlooping section in the Red Hat OpenShift Container Platform Backup and Restore Guide at <https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/backup_and_restore/>

* 1. Reset the master01 machine in the lab environment.

From the workstation machine, use Firefox to navigate to the **Lab Environment** tab on the Red Hat OpenShift Installation Lab web page. Click **ACTION** for the master01 server, and then select **Reset**, as shown in the following image:

A screenshot of a computer

Description automatically generated

When prompted to confirm the reset, click **YES, RESET IT**.

|  |
| --- |
| A screenshot of a phone  Description automatically generated |

This restores the master01 server to the initial status when the lab environment was created (a blank disk, ready for PXE boot).

* 1. From the utility server, verify the status of master01 server and the etcd cluster. Wait until the master01 server displays NotReady. Press **Ctrl+C** to exit.
  2. [lab@utility ~]$ **oc get nodes -w**
  3. NAME STATUS ROLES AGE VERSION
  4. **master01** **NotReady** master 73m v1.19.0+9f84db3
  5. master02 Ready master 73m v1.19.0+9f84db3
  6. master03 Ready master 73m v1.19.0+9f84db3
  7. worker01 Ready worker 47m v1.19.0+9f84db3

worker02 Ready worker 47m v1.19.0+9f84db3

[lab@utility ~]$ **oc get etcd -o=jsonpath='{range .items[0].status.conditions[?(@.type=="EtcdMembersAvailable")]}{.message}{"\n"}'**

2 of 3 members are available, master01 is unhealthy

[lab@utility ~]$ **oc get nodes -o jsonpath='{range .items[\*]}{"\n"}{.metadata.name}{"\t"}{range .spec.taints[\*]}{.key}{" "}' | grep unreachable**

**master01** node-role.kubernetes.io/master **node.kubernetes.io/unreachable** node.kubernetes.io/unreachable

The node master01 is NotReady and has the taint node.kubernetes.io/unreachable.

### NOTE

The pods running on the master01 server might show the status Running for some time after the server connection is lost. This is because the kubelet running on master01 was unable to report the NotReady status.

### NOTE

Because you deleted a member of the control plane, you might encounter temporary errors in pods or services of the cluster. If so, ignore the error messages and retry your commands to complete the exercise.

1. Remove the unhealthy etcd member.
   1. From a running etcd pod, remove the missing etcd member.
   2. [lab@utility ~]$ **oc get pods -n openshift-etcd | grep etcd**
   3. etcd-master01 3/3 Running 0 89m
   4. **etcd-master02** 3/3 Running 0 88m
   5. **etcd-master03** 3/3 Running 0 90m
   6. etcd-quorum-guard-644f5747b8-bb8vw 1/1 Running 0 93m
   7. etcd-quorum-guard-644f5747b8-c9b5k 1/1 Running 0 93m
   8. etcd-quorum-guard-644f5747b8-fbc5f 1/1 Running 0 93m
   9. [lab@utility ~]$ **oc rsh -n openshift-etcd etcd-master02**
   10. Defaulting container name to etcdctl.
   11. Use 'oc describe pod/etcd-master02 -n openshift-etcd' to see all of the containers in this pod.
   12. sh-4.4# **etcdctl member list -w table**
   13. +------------------+---------+----------+----------------------------+----------------------------+------------+
   14. | ID | STATUS | NAME | PEER ADDRS | CLIENT ADDRS | IS LEARNER |
   15. +------------------+---------+----------+----------------------------+----------------------------+------------+
   16. | **6c05b85443152afa** | started | **master01** | https://192.168.50.10:2380 | https://192.168.50.10:2379 | false |
   17. | a8fc53b7e8e11c19 | started | master02 | https://192.168.50.11:2380 | https://192.168.50.11:2379 | false |
   18. | f955e62306188c83 | started | master03 | https://192.168.50.12:2380 | https://192.168.50.12:2379 | false |

+------------------+---------+----------+----------------------------+----------------------------+------------+

### WARNING

The etcd members are sorted by the ID field, not the NAME. Verify that you are using the master01 member ID, which might not be first on the list.

Next, remove the unhealthy etcd member.

sh-4.4# **etcdctl member remove 6c05b85443152afa**

Member 6c05b85443152afa removed from cluster c73435d0ce2db908

sh-4.4# **etcdctl member list -w table**

+------------------+---------+----------+----------------------------+----------------------------+------------+

| ID | STATUS | NAME | PEER ADDRS | CLIENT ADDRS | IS LEARNER |

+------------------+---------+----------+----------------------------+----------------------------+------------+

| a8fc53b7e8e11c19 | started | master02 | https://192.168.50.11:2380 | https://192.168.50.11:2379 | false |

| f955e62306188c83 | started | master03 | https://192.168.50.12:2380 | https://192.168.50.12:2379 | false |

+------------------+---------+----------+----------------------------+----------------------------+------------+

sh-4.4# **exit**

exit

The etcd member master01 is now removed.

* 1. Remove the secrets used by the removed etcd member.
  2. [lab@utility ~]$ **oc get secrets -n openshift-etcd | grep master01**
  3. etcd-peer-master01 kubernetes.io/tls 2 102m
  4. etcd-serving-master01 kubernetes.io/tls 2 102m

etcd-serving-metrics-master01 kubernetes.io/tls 2 102m

[lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-peer-master01**

secret "etcd-peer-master01" deleted

[lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-serving-master01**

secret "etcd-serving-master01" deleted

[lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-serving-metrics-master01**

secret "etcd-serving-metrics-master01" deleted

* 1. Delete the master01 node object.
  2. [lab@utility ~]$ **oc delete node master01**

node "master01" deleted

[lab@utility ~]$ **oc get nodes**

NAME STATUS ROLES AGE VERSION

master02 Ready master 105m v1.19.0+9f84db3

master03 Ready master 106m v1.19.0+9f84db3

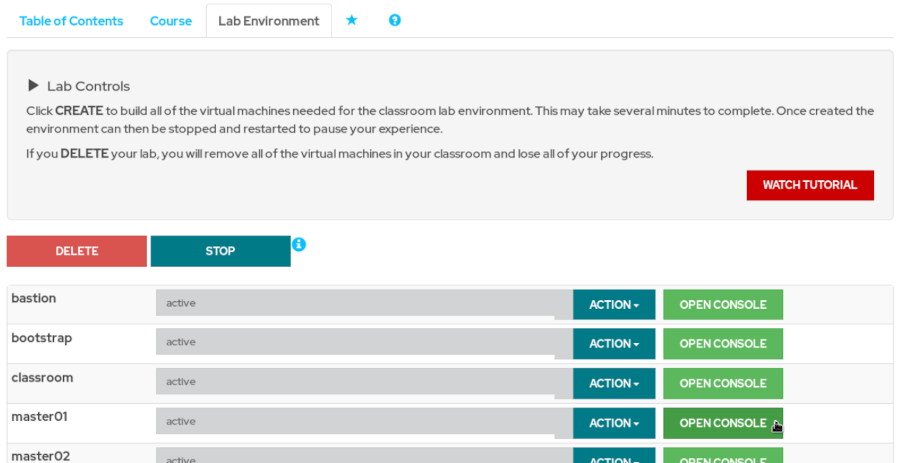
worker01 Ready worker 79m v1.19.0+9f84db3

worker02 Ready worker 79m v1.19.0+9f84db3

* 1. Remove the secrets used by the removed etcd member.
  2. [lab@utility ~]$ **oc get secrets -n openshift-etcd | grep master01**
  3. etcd-peer-master01 kubernetes.io/tls 2 102m
  4. etcd-serving-master01 kubernetes.io/tls 2 102m
  5. etcd-serving-metrics-master01 kubernetes.io/tls 2 102m
  6. [lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-peer-master01**
  7. secret "etcd-peer-master01" deleted
  8. [lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-serving-master01**
  9. secret "etcd-serving-master01" deleted
  10. [lab@utility ~]$ **oc delete secret -n openshift-etcd etcd-serving-metrics-master01**

secret "etcd-serving-metrics-master01" deleted

1. From the workstation machine, use Firefox to provision the new master01 node from the **Lab Environment** tab within the Red Hat Learning portal.
   1. Use the **Console** button to open the web console and install the master01 node using the PXE Boot Menu.



### NOTE

If the RHCOS installation does not start, use the **Ctrl+Alt+Del** button on the server web console to trigger a restart.

### NOTE

A warning message displays in the Firefox browser. Click **Preferences** and allow Firefox to show the pop-up windows for rol.redhat.com.

Reboot the master01 virtual machine from within the Red Hat Learning portal console by clicking **Ctrl Alt Del** in the top right corner.

After the reboot is completed, press **Enter** to start the RHCOS installation in the PXE Boot menu.

A screenshot of a computer

Description automatically generated

After two reboots, the new master01 control plane node is installed.

|  |
| --- |
| A black screen with white text  Description automatically generated |

1. Use the terminal on the utility server to add the new master01 node to the cluster.
   1. Approve the kubelet and the node certificates.
   2. [lab@utility ~]$ **oc get csr | grep Pending**

csr-7qqzn 2m1s kubernetes.io/kube-apiserver-client-kubelet system:serviceaccount:openshift-machine-config-operator:node-bootstrapper Pending

[lab@utility ~]$ **oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve**

certificatesigningrequest.certificates.k8s.io/csr-7qqzn approved

[lab@utility ~]$ **oc get csr | grep Pending**

csr-48rhz 4s kubernetes.io/kubelet-serving system:node:master01 Pending

[lab@utility ~]$ **oc get csr -o go-template='{{range .items}}{{if not .status}}{{.metadata.name}}{{"\n"}}{{end}}{{end}}' | xargs oc adm certificate approve**

certificatesigningrequest.certificates.k8s.io/csr-48rhz approved

### NOTE

There might be more than one pending CSR in the cluster. Do not worry if the output of oc get csr | grep Pending shows more than one entry.

* 1. Verify the status of the nodes in the cluster.
  2. [lab@utility ~]$ **oc get nodes**
  3. NAME STATUS ROLES AGE VERSION
  4. **master01 Ready master 6m2s v1.19.0+9f84db3**
  5. master02 Ready master 120m v1.19.0+9f84db3
  6. master03 Ready master 120m v1.19.0+9f84db3
  7. worker01 Ready worker 93m v1.19.0+9f84db3

worker02 Ready worker 93m v1.19.0+9f84db3

* 1. Check the status of the etcd pods in the cluster.
  2. [lab@utility ~]$ **oc get pods -n openshift-etcd | grep etcd**
  3. **etcd-master01** 3/3 Running 0 **9m7s**
  4. etcd-master02 3/3 Running 0 7m35s
  5. etcd-master03 3/3 Running 0 8m37s
  6. etcd-quorum-guard-644f5747b8-bb8vw 1/1 Running 0 121m
  7. etcd-quorum-guard-644f5747b8-c9b5k 1/1 Running 0 121m

etcd-quorum-guard-644f5747b8-dfrcj 1/1 Running 0 13m

* 1. Verify the members of the etcd cluster from within the etcd pod in master01.
  2. [lab@utility ~]$ **oc rsh -n openshift-etcd etcd-master01**
  3. Defaulting container name to etcdctl.
  4. Use 'oc describe pod/etcd-master01 -n openshift-etcd' to see all of the containers in this pod.
  5. sh-4.4# **etcdctl member list -w table**
  6. +------------------+---------+----------+----------------------------+----------------------------+------------+
  7. | ID | STATUS | NAME | PEER ADDRS | CLIENT ADDRS | IS LEARNER |
  8. +------------------+---------+----------+----------------------------+----------------------------+------------+
  9. | a8fc53b7e8e11c19 | started | master02 | https://192.168.50.11:2380 | https://192.168.50.11:2379 | false |
  10. | f9423fa20fccb5de | started | master01 | https://192.168.50.10:2380 | https://192.168.50.10:2379 | false |
  11. | f955e62306188c83 | started | master03 | https://192.168.50.12:2380 | https://192.168.50.12:2379 | false |
  12. +------------------+---------+----------+----------------------------+----------------------------+------------+
  13. sh-4.4# **etcdctl endpoint health --cluster**
  14. https://192.168.50.11:2379 is healthy: successfully committed proposal: took = 8.418937ms
  15. https://192.168.50.10:2379 is healthy: successfully committed proposal: took = 8.842156ms
  16. https://192.168.50.12:2379 is healthy: successfully committed proposal: took = 8.2554ms
  17. sh-4.4# **exit**

exit

* 1. Verify that the three etcd members are available to the OpenShift cluster.
  2. [lab@utility ~]$ **oc get etcd -o=jsonpath='{range .items[0].status.conditions[?(@.type=="EtcdMembersAvailable")]}{.message}{"\n"}'**

**3 members are available**

### NOTE

The etcd operator automatically detects a new etcd member and joins it into the cluster. In this scenario, you do not need to use the etcd backup.

**Finish**

After completing this guided exercise, delete the complete lab environment using the **DELETE** button from within the Red Hat Learning portal.

This concludes the guided exercise.

## **Lab: Installing a Compact OpenShift Cluster**

* Install a compact cluster (three control plane nodes), and verify the installation.

**Outcomes**

You should be able to complete the installation of a compact OpenShift cluster and verify the installation.

From within the Red Hat Learning portal, click **DELETE** to delete the complete lab environment.

After the environment is deleted, click **CREATE**, and then wait until all the servers are active.

After all the servers become active, click **+** to extend the auto-stop timer and increase the interval to at least two hours. This will help ensure that your environment is not stopped in the middle of the exercise. As the student user on the workstation machine, use the lab command to prepare your system for this exercise.

This command ensures that the necessary files to complete the comprehensive review are available on the workstation machine.

[student@workstation ~]$ **lab comprehensive-review start**

### NOTE

The Firefox version in the workstation machine does not render properly the <https://cloud.redhat.com/openshift/> site. To complete this guided exercise, install and use the Chromium web browser in the workstation:

[student@workstation ~]$ **sudo yum install chromium**

**Procedure 7.1. Instructions**

1. Find the local registry FQDN for the region of your classroom environment. You can locate your lab environment region on the **Lab Environment** tab on the Red Hat OpenShift Installation Lab course web page.

The following table shows the FQDN of the local registry for each region.

**Table 7.1. Local Registry FQDN per Classroom Environment Region**

| **Region** | **Local registry FQDN** |
| --- | --- |
| northamerica | nexus-registry-int.apps.tools-na150.prod.ole.redhat.com |
| emea | nexus-registry-int.apps.tools-emea150.prod.ole.redhat.com |
| apac | nexus-registry-int.apps.tools-apac150.prod.ole.redhat.com |

During this exercise, you will use a local registry containing the OpenShift release images mirrored from quay.io. The local registry runs in the same region as the classroom environment, and its fully qualified domain name (FQDN) depends on that region.

* 1. Install the Chromium web browser from a terminal on the workstation machine.
  2. [student@workstation ~]$ **sudo yum install chromium**

...output omitted...

When prompted, type yes and press **Enter** to accept the installation.

* 1. Find the region of your classroom environment.

From the workstation machine, use the Chromium web browser to navigate to the **Lab Environment** tab on the Red Hat OpenShift Installation Lab course web page.

Click **information** to display information about the classroom environment. You can find the classroom environment region in the Published region field.

A screenshot of a computer

Description automatically generated

* 1. Locate the FQDN of the local registry for your region using the preceding table.
  2. As the lab user on the utility server, use the curl command to verify that you can communicate with the FQDN of the local registry for your classroom environment.

The following command assumes that the classroom environment region is northamerica, with the local registry FQDN nexus-registry-int.apps.tools-na150.prod.ole.redhat.com.

[student@workstation ~]$ **ssh lab@utility**

[lab@utility ~]$ **curl -s \**

> **https://nexus-registry-int.apps.tools-na150.prod.ole.redhat.com -o /dev/null; echo $?**

0

If you have used the wrong local registry FQDN, the output of the curl command will not be 0.

### NOTE

The local registry of the northamerica region nexus-registry-int.apps.tools-na150.prod.ole.redhat.com is used in the following steps.

Ensure that you use the local registry FQDN of the region of your classroom environment. Using a local registry from other regions will cause the failure of the guided exercises.

[Hide Solution](https://rol.redhat.com/rol/app/)

1. Create a valid pull secret for installing OpenShift in the classroom environment using a pull secret from cloud.redhat.com. The credentials for the quay.io registry must be replaced with the credentials for the local registry.

Assuming that your classroom environment region is northamerica, the content of the pull-secret for the local registry is:

"nexus-registry-int.apps.tools-na150.prod.ole.redhat.com":{"auth":"cmVndXNlcjpJbnN0YWxsTTM=","email":"nobody@example.com"}

Finally, copy the pull secret file to the utility server as the lab user.

* 1. Obtain a pull secret from cloud.redhat.com.

On the workstation machine, use the Chromium web browser to navigate to <https://cloud.redhat.com/openshift/install/metal/user-provisioned>. Log in using your Red Hat account credentials.

Click **Download pull secret**. Then select **Save File** and click **OK**.

The file is saved as /home/student/Downloads/pull-secret.

* 1. Format the pull secret as a JSON file.

Open a terminal on the workstation machine and format the pull-secret file in the /home/student/Downloads folder as a JSON.

[lab@utility ~]$ **exit**

...output omitted...

[student@workstation ~]$

[student@workstation ~]$ **python3 -m json.tool \**

> **Downloads/pull-secret > pull-secret.json**

[student@workstation ~]$ **cat pull-secret.json**

{

"auths": {

"cloud.openshift.com": {

"auth": "UxUUjYwSTMyb3BlUxUUjYwSTMybnNUxUUjYwSTMyoaWZUxUU...YMy3NfNGUxUUjYw==",

"email": "student@redhat.com"

},

**"quay.io"**: {

"auth": **"UxUUjYwSTMyb3BlUxUUjYwSTMybnNUxUUjYwSTMyoaWZUxUU...YMy3NfNGUxUUjYw=="**,

"email": **"student@redhat.com"**

},

"registry.connect.redhat.com": {

"auth": "CvmsWaJUROSkhCkEQ71NiM1BsracE9ZOVmBMIDrS3R20KOH8Eq...TxO9phtozeXpqKLJN=",

"email": "student@redhat.com"

},

"registry.redhat.io": {

"auth": "CvmsWaJUROSkhCkEQ71NiM1BsracE9ZOVmBMIDrS3R20KOH8Eq...TxO9phtozeXpqKLJN=",

"email": "student@redhat.com"

}

}

}

* 1. Edit the pull-secret.json file to replace the credentials for quay.io with the credentials for your local registry.
  2. [student@workstation ~]$ **vi pull-secret.json**
  3. {
  4. "auths": {
  5. "cloud.openshift.com": {
  6. "auth": "UxUUjYwSTMyb3BlUxUUjYwSTMybnNUxUUjYwSTMyoaWZUxUU...YMy3NfNGUxUUjYw==",
  7. "email": "student@redhat.com"
  8. },
  9. **"nexus-registry-int.apps.tools-na150.prod.ole.redhat.com"**: {
  10. "auth": **"cmVndXNlcjpJbnN0YWxsTTM="**,
  11. "email": **"nobody@example.com"**
  12. },
  13. "registry.connect.redhat.com": {
  14. "auth": "CvmsWaJUROSkhCkEQ71NiM1BsracE9ZOVmBMIDrS3R20KOH8Eq...TxO9phtozeXpqKLJN=",
  15. "email": "student@redhat.com"
  16. },
  17. "registry.redhat.io": {
  18. "auth": "CvmsWaJUROSkhCkEQ71NiM1BsracE9ZOVmBMIDrS3R20KOH8Eq...TxO9phtozeXpqKLJN=",
  19. "email": "student@redhat.com"
  20. }
  21. }

}

Save the file and close the editor.

* 1. Use the jq tool to generate the compact version of the pull-secret.json file in the pull-secret-oneline.json file.

[student@workstation ~]$ **cat pull-secret.json | jq . -c > pull-secret-oneline.json**

### NOTE

The jq parsing used in the preceding command also checks that the pull-secret.json file is valid. If you do not get any errors when running this command, it means that the pull-secret.json and pull-secret-oneline.json files are valid.

[student@workstation ~]$ **cat pull-secret-oneline.json**

{"auths":{"cloud.openshift.com":{"auth":"UxUUj...UUjYw==","email":"student@redhat.com"},"nexus-registry-int.apps.tools-na150.prod.ole.redhat.com":{"auth":"cmVnd...sTTM=","email":"nobody@example.com"},"registry.connect.redhat.com":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"},"registry.redhat.io":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"}}}

* 1. Copy the pull-secret-oneline.json file to the utility server.
  2. [student@workstation ~]$ **scp pull-secret-oneline.json lab@utility:**
  3. ...output omitted...

pull-secret-oneline.json 100% 2624 4.6MB/s 00:00

You will use the pull-secret-oneline.json file later on this lab to complete the install-config.yaml file.

[Hide Solution](https://rol.redhat.com/rol/app/)

1. Run the Ansible Playbook in the folder /home/student/DO322/labs/comprehensive-review/ansible/, using the inventory file included in this folder.
   1. Inspect the content of the folder /home/student/DO322/.
   2. [student@workstation ~]$ **tree DO322/**
   3. DO322/
   4. └── labs
   5. └── comprehensive-review
   6. └── ansible
   7. ├── inventory
   8. ├── prereq.yaml
   9. └── roles
   10. └── ocp\_install\_prereq
   11. ├── files
   12. │ ├── haproxy.cfg
   13. │ └── install-config.yaml
   14. ├── tasks
   15. │ └── main.yaml
   16. └── vars

└── main.yml

* 1. Change to the /home/student/DO322/labs/comprehensive-review/ansible/ folder.
  2. [student@workstation ~]$ **cd DO322/labs/comprehensive-review/ansible/**

[student@workstation ansible]$

* 1. Run the prereq.yaml Ansible Playbook. Use the inventory file included in the same folder.
  2. [student@workstation ansible]$ **ansible-playbook -i inventory prereq.yaml**
  3. PLAY [Configure Prerequisites] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  4. TASK [ocp\_install\_prereq : Download "oc"] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  5. changed: [utility.lab.example.com]
  6. TASK [ocp\_install\_prereq : Download "openshift-install"] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  7. changed: [utility.lab.example.com]
  8. TASK [ocp\_install\_prereq : Extract "oc" to /usr/bin/] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  9. changed: [utility.lab.example.com]
  10. TASK [ocp\_install\_prereq : Extract "openshift-install" to /usr/bin/] \*\*\*\*\*\*\*\*\*
  11. changed: [utility.lab.example.com]
  12. TASK [ocp\_install\_prereq : Download RHCOS rootfs] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  13. changed: [utility.lab.example.com]
  14. TASK [ocp\_install\_prereq : Download RHCOS kernel] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  15. changed: [utility.lab.example.com]
  16. TASK [ocp\_install\_prereq : Download RHCOS initramfs] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  17. changed: [utility.lab.example.com]
  18. TASK [ocp\_install\_prereq : Download bootstrap PXE Boot file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  19. changed: [utility.lab.example.com]
  20. TASK [ocp\_install\_prereq : Download master01 PXE boot file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  21. changed: [utility.lab.example.com]
  22. TASK [ocp\_install\_prereq : Download master02 PXE Boot file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  23. changed: [utility.lab.example.com]
  24. TASK [ocp\_install\_prereq : Download master03 PXE Boot file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  25. changed: [utility.lab.example.com]
  26. TASK [ocp\_install\_prereq : Configure haproxy.cfg file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  27. changed: [utility.lab.example.com]
  28. TASK [ocp\_install\_prereq : Reload HAProxy service] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  29. changed: [utility.lab.example.com]
  30. TASK [ocp\_install\_prereq : Create the OpenShift installation directory if it does not exist] \*\*\*\*\*\*\*
  31. changed: [utility.lab.example.com]
  32. TASK [ocp\_install\_prereq : Create custom install-config.yaml file] \*\*\*\*\*\*\*\*\*\*\*
  33. changed: [utility.lab.example.com]
  34. PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

utility.lab.example.com : ok=15 changed=15 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

As you can see, this playbook performs several prerequisite tasks in utility.lab.example.com.

* + - Downloads and installs the oc and openshift-install CLI tools
    - Downloads the RHCOS files
    - Downloads the PXE Boot files for the bootstrap node and the control plane nodes
    - Configures the HAProxy Load Balancer for the API and Ingress traffic
    - Downloads a base install-config.yaml file (that you will complete later)
  1. Return to the home folder belonging to the student user.
  2. [student@workstation ansible]$ **cd ~**

[student@workstation ~]$

1. [Hide Solution](https://rol.redhat.com/rol/app/)
2. Create a SSH key on the utility server. Save it to the /home/lab/.ssh/ocp4upi file.
   1. Connect to the utility server via SSH as the lab user.
   2. [student@workstation ~]$ **ssh lab@utility**
   3. ...output omitted...

[lab@utility ~]$

* 1. Generate the SSH key, and then save it in the /home/lab/.ssh/ocp4upi file.
  2. [lab@utility ~]$ **ssh-keygen -t rsa -b 4096 -N '' -f .ssh/ocp4upi**
  3. Generating public/private rsa key pair.
  4. Your identification has been saved in .ssh/ocp4upi.
  5. Your public key has been saved in .ssh/ocp4upi.pub.
  6. ...output omitted...

[lab@utility ~]$

The public key, which you will use later, is in the file /home/lab/.ssh/ocp4upi.pub.

1. [Hide Solution](https://rol.redhat.com/rol/app/)
2. Complete the install-config.yaml file in the /home/lab/ocp4upi/ folder. Use the pull secret generated elsewhere in this exercise. Use the SSH key generated elsewhere in this exercise. Include the imageContentSources information to use the local registry to retrieve the OpenShift release images. There must be only three control plane nodes without any compute nodes.
   1. Edit the install-config.yaml file in the /home/lab/ocp4upi/ folder. Use the compact pull secret in the pull-secret-oneline.json file.
   2. [lab@utility ~]$ **vi ocp4upi/install-config.yaml**
   3. apiVersion: v1
   4. baseDomain: example.com
   5. compute:
   6. - hyperthreading: Enabled
   7. **name: worker**
   8. **replicas: 0**
   9. controlPlane:
   10. hyperthreading: Enabled
   11. **name: master**
   12. **replicas: 3**
   13. metadata:
   14. name: ocp4
   15. networking:
   16. clusterNetwork:
   17. - cidr: 10.128.0.0/14
   18. hostPrefix: 23
   19. networkType: OpenShiftSDN
   20. serviceNetwork:
   21. - 172.30.0.0/16
   22. platform:
   23. none: {}
   24. fips: false
   25. pullSecret: |
   26. **{"auths":{"cloud.openshift.com":{"auth":"UxUUj...UUjYw==","email":"student@redhat.com"},"nexus-registry-int.apps.tools-na150.prod.ole.redhat.com":{"auth":"cmVnd...sTTM=","email":"nobody@example.com"},"registry.connect.redhat.com":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"},"registry.redhat.io":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"}}}**
   27. sshKey: |
   28. <CHANGE\_ME\_KEEPING\_THE\_INDENTATION\_LEVEL>
   29. imageContentSources:
   30. - mirrors:
   31. - **nexus-registry-int.apps.tools-na150.prod.ole.redhat.com**/openshift/ocp4
   32. source: quay.io/openshift-release-dev/ocp-release
   33. - mirrors:
   34. - **nexus-registry-int.apps.tools-na150.prod.ole.redhat.com**/openshift/ocp4

source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

Save the file and close the editor.

### NOTE

The number of replicas of the compute machines is 0. This makes the control plane nodes schedulable automatically.

* 1. Copy the public SSH key from the /home/lab/.ssh/ocp4upi.pub file. Edit the install-config.yaml file again to include the key in the sshKey section.
  2. [lab@utility ~]$ **cat .ssh/ocp4upi.pub**

ssh-rsa AAAA...UgUsz2w== lab@utility.lab.example.com

[lab@utility ~]$ **vi ocp4upi/install-config.yaml**

apiVersion: v1

baseDomain: example.com

compute:

- hyperthreading: Enabled

name: worker

replicas: 0

controlPlane:

hyperthreading: Enabled

name: master

replicas: 3

metadata:

name: ocp4

networking:

clusterNetwork:

- cidr: 10.128.0.0/14

hostPrefix: 23

networkType: OpenShiftSDN

serviceNetwork:

- 172.30.0.0/16

platform:

none: {}

fips: false

pullSecret: |

{"auths":{"cloud.openshift.com":{"auth":"UxUUj...UUjYw==","email":"student@redhat.com"},"nexus-registry-int.apps.tools-na150.prod.ole.redhat.com":{"auth":"cmVnd...sTTM=","email":"nobody@example.com"},"registry.connect.redhat.com":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"},"registry.redhat.io":{"auth":"CvmsW...qKLJN=","email":"student@redhat.com"}}}

sshKey: |

**ssh-rsa AAAA...UgUsz2w== lab@utility.lab.example.com**

imageContentSources:

- mirrors:

- nexus-registry-int.apps.tools-na150.prod.ole.redhat.com/openshift/ocp4

source: quay.io/openshift-release-dev/ocp-release

- mirrors:

- nexus-registry-int.apps.tools-na150.prod.ole.redhat.com/openshift/ocp4

source: quay.io/openshift-release-dev/ocp-v4.0-art-dev

Save the file and close the editor.

* 1. Make a copy of the install-config.yaml file in the /home/lab/ folder for troubleshooting purposes.

[lab@utility ~]$ **cp ocp4upi/install-config.yaml .**

1. [Hide Solution](https://rol.redhat.com/rol/app/)
2. Create the Kubernetes manifests and ignition configuration files using the openshift-install CLI tool. You must configure the installer to use the classroom local registry to retrieve the OpenShift release version. To do so, you must export the following variable:

OPENSHIFT\_INSTALL\_RELEASE\_IMAGE\_OVERRIDE="<LOCAL\_REGISTRY\_FQDN>/openshift/ocp4:4.6.4-x86\_64"

Place the ignition files in the /var/www/html/openshift4/4.6.4/ignitions/ with world-readable permissions.

* 1. Set the OPENSHIFT\_INSTALL\_RELEASE\_IMAGE\_OVERRIDE environment variable to use the local registry for your cluster region. The release image is in the path: openshift/ocp4:4.6.4-x86\_64.
  2. [lab@utility ~]$ **reg="nexus-registry-int.apps.tools-na150.prod.ole.redhat.com"**
  3. [lab@utility ~]$ **releaseimg="/openshift/ocp4:4.6.4-x86\_64"**

[lab@utility ~]$ **export OPENSHIFT\_INSTALL\_RELEASE\_IMAGE\_OVERRIDE=$reg$releaseimg**

Verify the complete URL path to the release image as follows:

[lab@utility ~]$ **echo $OPENSHIFT\_INSTALL\_RELEASE\_IMAGE\_OVERRIDE**

nexus-registry-int.apps.tools-na150.prod.ole.redhat.com/openshift/ocp4:4.6.4-x86\_64

### NOTE

The commands above assume that your cluster is in the northamerica region. Use the FQDN of the local registry for your region.

* 1. Generate the Kubernetes manifests.
  2. [lab@utility ~]$ **openshift-install create manifests --dir=./ocp4upi**
  3. INFO Consuming Install Config from target directory
  4. **WARNING Making control-plane schedulable by setting MastersSchedulable to true for Scheduler cluster settings**

INFO Manifests created in: ocp4upi/manifests and ocp4upi/openshift

* 1. Generate the ignition configuration files.
  2. [lab@utility ~]$ **openshift-install create ignition-configs --dir=./ocp4upi**
  3. WARNING Found override for release image. Please be warned, this is not advised
  4. INFO Consuming Master Machines from target directory
  5. INFO Consuming Common Manifests from target directory
  6. INFO Consuming OpenShift Install (Manifests) from target directory
  7. INFO Consuming Worker Machines from target directory
  8. INFO Consuming Openshift Manifests from target directory

INFO Ignition-Configs created in: ocp4upi and ocp4upi/auth

* 1. Copy the ignition files to the /var/www/html/openshift4/4.6.4/ignitions/ folder. Set the files permissions readable by anyone.
  2. [lab@utility ~]$ **sudo cp ./ocp4upi/\*.ign /var/www/html/openshift4/4.6.4/ignitions/**

[lab@utility ~]$ **sudo chmod +r /var/www/html/openshift4/4.6.4/ignitions/\*.ign**

Verify the read permissions of the ignition config files.

[lab@utility ~]$ **ls -l /var/www/html/openshift4/4.6.4/ignitions/**

total 296

**-rw-r--r--.** 1 root root 293214 Feb 12 05:37 bootstrap.ign

**-rw-r--r--.** 1 root root 1718 Feb 12 05:37 master.ign

**-rw-r--r--.** 1 root root 1718 Feb 12 05:37 worker.ign

[Hide Solution](https://rol.redhat.com/rol/app/)

1. From the PXE menu, install RHCOS in the bootstrap, master01, master02, and master03 machines to start the OpenShift cluster deployment.
   1. Reboot the bootstrap machine from within the Red Hat Learning portal console by clicking **Ctrl Alt Del** in the top right corner.
   2. Wait until the reboot is completed and the machine boots into the PXE menu.
   3. From the PXE menu, press **Enter** to proceed with the installation of Red Hat Enterprise Linux CoreOS on the machine.
   4. Repeat these steps to install master01, master02, and master03.
   5. For information purposes, run the openshift-install command to wait for the bootstrap installation.
   6. [lab@utility ~]$ **openshift-install --dir=./ocp4upi wait-for bootstrap-complete**
   7. INFO Waiting up to 20m0s for the Kubernetes API at https://api.ocp4.example.com:6443...

INFO API v1.19.0+9f84db3 up

### NOTE

As soon as you run the previous step, continue with the next steps in the lab. The previous command will take approximately 10 minutes to finish.

* 1. Open a new terminal on the workstation machine and connect via SSH to the utility server as the lab user.
  2. [student@workstation ~]$ **ssh lab@utility**
  3. ...output omitted...

[lab@utility ~]$

* 1. Look at the output of the openshift-install command, and wait until the API is running.

Then, configure the KUBECONFIG environment variable to interact with the cluster.

[lab@utility ~]$ **export KUBECONFIG=~/ocp4upi/auth/kubeconfig**

* 1. Watch the creation and configuration of the control plane nodes using the watch oc get nodes command.

### NOTE

It can take a few minutes to create the control plane nodes.

[lab@utility ~]$ **watch oc get nodes**

The status in the output is updated every two seconds by default.

When you first run the watch command, it will not show any nodes.

After a few minutes, the output will display the following:

Every 2.0s: oc get nodes utility.lab.example.com: Wed Feb 3 08:10:51 2021

NAME STATUS ROLES AGE VERSION

master01 **NotReady** **master** 5s v1.19.0+9f84db3

master02 **NotReady** **master** 12s v1.19.0+9f84db3

master03 **NotReady** **master** 13s v1.19.0+9f84db3

The control plane nodes have been created but are not ready yet.

After a few more minutes, the output will display the updated status.

Every 2.0s: oc get nodes utility.lab.example.com: Wed Feb 3 08:11:33 2021

NAME STATUS ROLES AGE VERSION

master01 **Ready** **master** 47s v1.19.0+9f84db3

master02 **Ready** **master** 54s v1.19.0+9f84db3

master03 **Ready** **master** 55s v1.19.0+9f84db3

The control plane nodes are not yet configured as "schedulable". Eventually, the output will show:

Every 2.0s: oc get nodes utility.lab.example.com: Wed Feb 3 08:11:59 2021

NAME STATUS ROLES AGE VERSION

master01 Ready **master,worker** 73s v1.19.0+9f84db3

master02 Ready **master,worker** 80s v1.19.0+9f84db3

master03 Ready **master,worker** 81s v1.19.0+9f84db3

At this point, the control plane nodes are properly configured as "schedulable"

### NOTE

The control plane nodes CSRs are approved automatically during the installation.

* 1. After the control plane nodes are Ready and schedulable, the openshift-install command displays the following output, and then exits.
  2. INFO Waiting up to 20m0s for the Kubernetes API at https://api.ocp4.example.com:6443...
  3. INFO API v1.19.0+9f84db3 up
  4. INFO Waiting up to 30m0s for bootstrapping to complete...
  5. INFO It is now safe to remove the bootstrap resources

INFO Time elapsed: 8m1s

Use the openshift-install command to monitor the remainder of the installation.

[lab@utility ~]$ **openshift-install --dir=./ocp4upi \**

> **wait-for install-complete --log-level=debug**

DEBUG OpenShift Installer 4.6.4

DEBUG Built from commit 6e02d049701437fa81521fe981405745a62c86c5

DEBUG Loading Install Config...

DEBUG Loading SSH Key...

DEBUG Loading Base Domain...

DEBUG Loading Platform...

DEBUG Loading Cluster Name...

DEBUG Loading Base Domain...

DEBUG Loading Platform...

DEBUG Loading Pull Secret...

DEBUG Loading Platform...

DEBUG Using Install Config loaded from state file

INFO Waiting up to 40m0s for the cluster at https://api.ocp4.example.com:6443 to initialize...

DEBUG Still waiting for the cluster to initialize: Working towards 4.6.4: 99% complete

DEBUG Still waiting for the cluster to initialize: Multiple errors are preventing progress:

\* Cluster operator authentication is reporting a failure: WellKnownReadyControllerDegraded: kube-apiserver oauth endpoint https://192.168.50.12:6443/.well-known/oauth-authorization-server is not yet served and authentication operator keeps waiting (check kube-apiserver operator, and check that instances roll out successfully, which can take several minutes per instance)

\* Cluster operator monitoring is reporting a failure: Failed to rollout the stack. Error: running task Updating Prometheus-k8s failed: reconciling Prometheus rules PrometheusRule failed: updating PrometheusRule object failed: Internal error occurred: failed calling webhook "prometheusrules.openshift.io": Post "https://prometheus-operator.openshift-monitoring.svc:8080/admission-prometheusrules/validate?timeout=5s": x509: certificate signed by unknown authority

DEBUG Still waiting for the cluster to initialize: Cluster operator authentication is reporting a failure: WellKnownReadyControllerDegraded: need at least 3 kube-apiservers, got 2

DEBUG Still waiting for the cluster to initialize: Cluster operator authentication is reporting a failure: WellKnownReadyControllerDegraded: need at least 3 kube-apiservers, got 2

DEBUG Still waiting for the cluster to initialize: Cluster operator authentication is reporting a failure: WellKnownReadyControllerDegraded: need at least 3 kube-apiservers, got 2

DEBUG Cluster is initialized

INFO Waiting up to 10m0s for the openshift-console route to be created...

DEBUG Route found in openshift-console namespace: console

DEBUG Route found in openshift-console namespace: downloads

DEBUG OpenShift console route is created

INFO Install complete!

INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home/lab/ocp4upi/auth/kubeconfig'

INFO Access the OpenShift web-console here: https://console-openshift-console.apps.ocp4.example.com

INFO Login to the console with user: "kubeadmin", and password: "rTbDK-j5rqe-JQKVZ-ftEGH"

DEBUG Time elapsed per stage:

DEBUG Cluster Operators: 12m44s

INFO Time elapsed: 12m44s

### NOTE

It takes around 20 minutes to complete the installation.

At this point, you can monitor the status of the installation from another terminal with commands such as:

[lab@utility ~]$ **oc get clusteroperator**

[lab@utility ~]$ **oc get clusterversion**

[lab@utility ~]$ **watch "oc get pods \**

> **--all-namespaces | grep -v -E 'Running|Completed'"**

### NOTE

Transient error messages while the installation progresses are normal and can be ignored safely.

The installation takes around 25 minutes in total to complete.

1. [Hide Solution](https://rol.redhat.com/rol/app/)
2. Configure the persistent storage for the local registry using the NFS server running on the utility server.

Use a new folder named registry within the NFS /exports folder on the utility server. The folder permissions must be 777.

* 1. Create the /exports/registry folder with 777 permissions.
  2. [lab@utility ~]$ **mkdir /exports/registry**

[lab@utility ~]$ **sudo chmod 777 /exports/registry/**

* 1. Create the persistent volume file (PV) pv.yaml.
  2. [lab@utility ~]$ **vi pv.yaml**
  3. apiVersion: v1
  4. kind: PersistentVolume
  5. metadata:
  6. name: registry-pv
  7. spec:
  8. capacity:
  9. storage: 5Gi
  10. accessModes:
  11. - ReadWriteMany
  12. nfs:
  13. path: /exports/registry
  14. server: 192.168.50.254

persistentVolumeReclaimPolicy: Recycle

### NOTE

If not done previously, you must configure the KUBECONFIG environment variable to interact with the cluster.

[lab@utility ~]$ **export KUBECONFIG=~/ocp4upi/auth/kubeconfig**

[lab@utility ~]$ **oc create -f pv.yaml**

persistentvolume/registry-pv created

* 1. Create the persistent volume claim (PVC) file pvc.yaml.
  2. [lab@utility ~]$ **vi pvc.yaml**
  3. kind: PersistentVolumeClaim
  4. apiVersion: v1
  5. metadata:
  6. name: registry-claim
  7. namespace: openshift-image-registry
  8. spec:
  9. accessModes:
  10. - ReadWriteMany
  11. resources:
  12. requests:

storage: 5Gi

[lab@utility ~]$ **oc create -f pvc.yaml**

persistentvolumeclaim/registry-claim created

* 1. Set the image registry operator in a "Managed" state. Edit the configuration of the cluster image registry to add the PVC. Also, configure the image registry to have two pod replicas.
  2. [lab@utility ~]$ **oc edit configs.imageregistry/cluster**
  3. ...output omitted...
  4. spec:
  5. ...output omitted...
  6. managementState: **Managed**
  7. ...output omitted...
  8. proxy: {}
  9. replicas: **2**
  10. requests:
  11. ...output omitted...
  12. rolloutStrategy: RollingUpdate
  13. **storage:**
  14. **pvc:**
  15. **claim: registry-claim**
  16. ...output omitted...

config.imageregistry.operator.openshift.io/cluster edited

* 1. Verify the configuration of the registry.
  2. [lab@utility ~]$ **oc get pv -A**
  3. NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

registry-pv 5Gi RWX Recycle **Bound** openshift-image-registry/registry-claim 106s

[lab@utility ~]$ **oc get pvc -A**

NAMESPACE NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

openshift-image-registry registry-claim **Bound** registry-pv 5Gi RWX 107s

[lab@utility ~]$ **oc get pods -n openshift-image-registry \**

> **-o wide | grep ^image-registry**

image-registry-84dbc74f75-bd9zk 1/1 Running 0 32s 10.128.0.34 master03

image-registry-84dbc74f75-bxdgm 1/1 Running 0 25s 10.129.0.34 master02

### NOTE

You might see other pods in the Terminating status while the image-registry pods are configured.

[Hide Solution](https://rol.redhat.com/rol/app/)

1. Verify the cluster health.
   1. Verify the NTP configuration.
   2. [lab@utility ~]$ **oc debug node/master01**
   3. Creating debug namespace/openshift-debug-node-pnwv9 ...
   4. Starting pod/master01-debug ...
   5. To use host binaries, run **chroot /host**
   6. Pod IP: 192.168.50.10
   7. If you don't see a command prompt, try pressing enter.
   8. sh-4.4# **chroot /host**
   9. sh-4.4# **cat /etc/chrony.conf**
   10. # Use public servers from the pool.ntp.org project.
   11. # Please consider joining the pool (http://www.pool.ntp.org/join.html).
   12. **pool 2.rhel.pool.ntp.org iburst**
   13. ...output omitted...
   14. sh-4.4# **sudo chronyc tracking**
   15. Reference ID : 2D4F6F72 (chl.la)
   16. Stratum : 3
   17. Ref time (UTC) : Thu Feb 04 12:52:11 2021
   18. System time : 0.000170327 seconds slow of NTP time
   19. Last offset : -0.000190245 seconds
   20. RMS offset : 0.000336431 seconds
   21. Frequency : 16.969 ppm slow
   22. Residual freq : -0.005 ppm
   23. Skew : 0.184 ppm
   24. Root delay : 0.041434124 seconds
   25. Root dispersion : 0.002719518 seconds
   26. Update interval : 1031.9 seconds
   27. Leap status : **Normal**
   28. sh-4.4# **exit**
   29. exit
   30. sh-4.4# **exit**

exit

The system is using the NTP pool 2.rhel.pool.ntp.org. It is in sync with the NTP server chl.la of the stratum 3, and the status is Normal.

Repeat the commands above for the other control plane nodes to check the NTP status.

* 1. Verify the cluster resources usage metrics.
  2. [lab@utility ~]$ **oc adm top node**
  3. NAME CPU(cores) CPU% MEMORY(bytes) MEMORY%
  4. master01 841m 24% 5803Mi 38%
  5. master02 744m 21% 6282Mi 42%

master03 622m 17% 4431Mi 29%

* 1. Verify that there are not pods in Failed status.
  2. [lab@utility ~]$ **oc get pods --all-namespaces | grep -v -E 'Running|Completed'**

NAMESPACE NAME READY STATUS RESTARTS AGE

### NOTE

It is normal to see pods from the openshift-apiserver namespace for a short time after the installation completes.

After a few minutes, the preceding command will not show any output.

* 1. Check the health of the etcd cluster.
  2. [lab@utility ~]$ **oc rsh -n openshift-etcd etcd-master01**
  3. Defaulting container name to etcdctl.
  4. Use 'oc describe pod/etcd-master01 -n openshift-etcd' to see all of the containers in this pod.
  5. sh-4.4# **etcdctl endpoint health --cluster**
  6. https://192.168.50.10:2379 is healthy: successfully committed proposal: took = 8.470285ms
  7. https://192.168.50.12:2379 is healthy: successfully committed proposal: took = 9.379322ms
  8. https://192.168.50.11:2379 is healthy: successfully committed proposal: took = 9.99598ms
  9. sh-4.4# **exit**

exit

* 1. Retrieve the API version to check the status of the cluster API.
  2. [lab@utility ~]$ **curl -k https://api.ocp4.example.com:6443/version**
  3. {
  4. "major": "1",
  5. "minor": "19",
  6. "gitVersion": "v1.19.0+9f84db3",
  7. "gitCommit": "9f84db336d1a77cba52684ecb51bfb197e9b4533",
  8. "gitTreeState": "clean",
  9. "buildDate": "2020-10-30T09:33:51Z",
  10. "goVersion": "go1.15.2",
  11. "compiler": "gc",

"platform": "linux/amd64"

* 1. Check the availability of the OpenShift console.
  2. [lab@utility ~]$ **curl -kIs \**
  3. > **https://console-openshift-console.apps.ocp4.example.com**
  4. **HTTP/1.1 200 OK**

...output omitted...

* 1. Verify the availability of the image registry from the cluster nodes.
  2. [lab@utility ~]$ **oc debug node/master01**
  3. Creating debug namespace/openshift-debug-node-zjb2h ...
  4. Starting pod/master01-debug ...
  5. To use host binaries, run **chroot /host**
  6. Pod IP: 192.168.50.10
  7. If you don't see a command prompt, try pressing enter.
  8. sh-4.4# **chroot /host**
  9. sh-4.4# **curl -kIs https://image-registry.openshift-image-registry.svc:5000/healthz**
  10. **HTTP/2 200**
  11. cache-control: no-cache
  12. date: Tue, 02 Feb 2021 12:01:49 GMT
  13. sh-4.4# **exit**
  14. exit
  15. sh-4.4# **exit**
  16. exit
  17. Removing debug pod ...

Removing debug namespace/openshift-debug-node-zjb2h ...

Repeat this step for the master02 and master03 control plane nodes.

1. [Hide Solution](https://rol.redhat.com/rol/app/)
2. Deploy a rails-postgresql-example application in the test project to perform a functional test of the cluster.
   1. Deploy a test application to verify the proper function of the build process and the registry storage.
   2. [lab@utility ~]$ **oc new-project test**
   3. Now using project "test" on server "https://api.ocp4.example.com:6443".
   4. You can add applications to this project with the 'new-app' command. For example, use the following command to build a new example application in Ruby.
   5. **oc new-app rails-postgresql-example**
   6. Or, you can use kubectl to deploy a simple Kubernetes application.

kubectl create deployment hello-node --image=k8s.gcr.io/serve\_hostname

[lab@utility ~]$ **oc new-app rails-postgresql-example**

--> Deploying template "openshift/rails-postgresql-example" to project test

Rails + PostgreSQL (Ephemeral)

---------

An example Rails application with a PostgreSQL database. For more information about using this template, including OpenShift considerations, see https://github.com/sclorg/rails-ex/blob/master/README.md.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.

The following service(s) have been created in your project: rails-postgresql-example, postgresql.

For more information about using this template, including OpenShift considerations, see https://github.com/sclorg/rails-ex/blob/master/README.md.

\* With parameters:

...output omitted...

--> Creating resources ...

secret "rails-postgresql-example" created

service "rails-postgresql-example" created

route.route.openshift.io "rails-postgresql-example" created

imagestream.image.openshift.io "rails-postgresql-example" created

buildconfig.build.openshift.io "rails-postgresql-example" created

deploymentconfig.apps.openshift.io "rails-postgresql-example" created

service "postgresql" created

deploymentconfig.apps.openshift.io "postgresql" created

--> **Success**

Access your application via route 'rails-postgresql-example-test.apps.ocp4.example.com'

Build scheduled, use 'oc logs -f buildconfig/rails-postgresql-example' to track its progress.

Run 'oc status' to view your app.

* 1. Watch the build process logs.
  2. [lab@utility ~]$ **oc logs -f buildconfig/rails-postgresql-example**
  3. Cloning "https://github.com/sclorg/rails-ex.git" ...
  4. ...output omitted...
  5. **Successfully pushed image-registry.openshift-image-registry.svc:5000/test/rails-postgresql-example@sha256:b97b...ff82**

**Push successful**

The image is built and pushed successfully to the internal registry, using the persistent storage.

* 1. Verify that all the pods in the namespace test are in the Running or Completed status.
  2. [lab@utility ~]$ **oc get pods -n test**
  3. NAME READY STATUS RESTARTS AGE
  4. postgresql-1-deploy 0/1 Completed 0 2m50s
  5. postgresql-1-pxg5x 1/1 Running 0 2m47s
  6. rails-postgresql-example-1-build 0/1 Completed 0 2m50s
  7. rails-postgresql-example-1-c4ljh 1/1 Running 0 31s
  8. rails-postgresql-example-1-deploy 0/1 Completed 0 52s

rails-postgresql-example-1-hook-pre 0/1 Completed 0 47s

* 1. Get the route for the rails-postgres-example test application.
  2. [lab@utility ~]$ **oc get routes -n test**
  3. NAME HOST/PORT PATH SERVICES PORT TERMINATION WILDCARD

rails-postgresql-example **rails-postgresql-example-test.apps.ocp4.example.com** rails-postgresql-example <all> None

* 1. On the workstation machine, open the Chromium web browser and navigate to the route rails-postgresql-example-test.apps.ocp4.example.com.

A screenshot of a computer

Description automatically generated

1. [Hide Solution](https://rol.redhat.com/rol/app/)

**Evaluation**

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

[lab@utility ~]$ **exit**

...output omitted...

[student@workstation ~]$ **lab comprehensive-review grade**

**Finish**

As the student user on the workstation machine, use the lab command to complete this exercise.

[student@workstation ~]$ **lab comprehensive-review finish**

### NOTE

The lab comprehensive-review finish command does not destroy the cluster. Running the lab comprehensive-review grade script after the lab comprehensive-review finish command shows that the three-node cluster exists.

If you wish to start over, reset the bootstrap, master01, master02, and master03 machines, and then start again at the step that instructs you to install RHCOS in the bootstrap, master01, master02, and master03 machines from the PXE menu.