Deployment Guide

Abstract
This deployment guide provides a validated procedure for deploying Red Hat OpenShift Container Platform 4.6 on Dell EMC PowerEdge servers, PowerSwitch networking, and PowerMax, PowerScale, PowerStore, and Unity XT storage arrays.
Contents

Unity XT storage ......................................................................................................................... 46
PowerMax storage .......................................................................................................................... 51
PowerScale storage ......................................................................................................................... 56
PowerStore storage ......................................................................................................................... 61
OpenShift Container storage .......................................................................................................... 65

Chapter 7 Application Deployments 69
Deploying applications ................................................................................................................. 70
Scaling applications ...................................................................................................................... 72

Chapter 8 Monitoring the Cluster 73
OpenShift monitoring overview .................................................................................................... 74
Adding storage .............................................................................................................................. 74
Enabling Grafana dashboards ....................................................................................................... 75
Viewing alerts ............................................................................................................................... 76
Viewing cluster metrics ................................................................................................................ 77

Chapter 9 Velero Backup and Restores 78
Overview ....................................................................................................................................... 79
Installing the Velero server ........................................................................................................... 79
This chapter presents the following topics:

- **Solution overview** ................................................................. 6
- **Document purpose** ............................................................... 6
- **Audience** .............................................................................. 7
- **We value your feedback** .......................................................... 7
Solution overview

Red Hat OpenShift Container Platform is an open-source application deployment platform that is based on Kubernetes container orchestration technology. Containers are stand-alone processes that run within their own environment and runtime context, independent of the underlying infrastructure. Red Hat OpenShift Container Platform helps you develop, deploy, and manage container-based applications.

**Note:** While you can rely on Red Hat Enterprise Linux security and container technologies to prevent intrusions and protect your data, some security vulnerabilities might persist. For information about security vulnerabilities in OpenShift Container Platform, see [OCP Errata](https://access.redhat.com/security/errata). For a general listing of Red Hat vulnerabilities, see the [Red Hat Security Home Page](https://access.redhat.com/security/advisories).

As part of Red Hat OpenShift Container Platform, Kubernetes manages containerized applications across a set of containers or hosts and provides mechanisms for the deployment, maintenance, and scaling of applications. The container runtime engine packages, instantiates, and runs containerized applications.

A Kubernetes cluster consists of one or more control plane nodes and a set of compute nodes. Kubernetes allocates an IP address from an internal network to each pod so that all containers within the pod behave as if they were on the same host. Giving each pod its own IP address means that pods can be treated like physical hosts or virtual machines for port allocation, networking, naming, service discovery, load balancing, application configuration, and migration. Dell Technologies recommends creating a Kubernetes service that enables your application pods to interact, rather than requiring that the pods communicate directly using their IP addresses.

A fully functioning Domain Name System (DNS) residing outside the OpenShift Container Platform is crucial in the deployment and operation of your container ecosystem. Red Hat OpenShift Container Platform has an integrated DNS so that the services can be found through DNS service entries or through the service IP/port registrations.

Dell EMC Ready Stack for Red Hat OpenShift Container Platform is a proven design to help organizations accelerate their container deployments and cloud-native adoption. Dell Technologies delivers tested, validated, and documented design guidance to help customers rapidly deploy Red Hat OpenShift on Dell EMC infrastructure by minimizing time and effort. For more information, see the [Red Hat OpenShift Container Platform 4.6 Design Guide](https://access.redhat.com/documentation/en-us/red_hat_openshift_container_platform/4.6/design_guide), which is available at [Red Hat OpenShift Container Platform](https://access.redhat.com/documentation/en-us/red_hat_openshift_container_platform/) on the Dell Technologies Info Hub.

**Note:** The guide provides links to sample configuration files in GitHub to demonstrate what values to specify in configuration procedures.

Document purpose

This deployment guide describes the infrastructure that is required for deploying and operating Red Hat OpenShift Container Platform. The guide provides a validated process for deploying a production-ready OpenShift Container Platform cluster information to facilitate readiness for Day-2 operations.

This guide describes validated steps for deploying Red Hat OpenShift Container Platform 4.6 on Dell EMC PowerEdge servers and with Dell EMC PowerSwitch switches. Dell

---

Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6
Enabled by Dell EMC PowerEdge R640 and R740xd Servers; PowerSwitch Networking; PowerMax, PowerScale, PowerStore, and Unity XT Storage
Deployment Guide
Chapter 1: Introduction

Technologies strongly recommends that you complete the validation steps that are described in this guide. Ensure that you are satisfied that your application will operate smoothly before proceeding with development or production use.

For more information about OpenShift Container Platform, see the OpenShift Container Platform 4.6 Documentation.

Note: This guide may contain language from third-party content that is not under Dell's control and is not consistent with Dell's current guidelines for Dell's own content. When such third-party content is updated by the relevant third parties, this guide will be revised accordingly.

Audience

This deployment guide is for system administrators and system architects. Some experience with Docker and Red Hat OpenShift Container Platform technologies is recommended. Review the solution design guide to familiarize yourself with the solution architecture and design before planning your deployment.

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email or provide your comments by completing our documentation survey.

Author: Umesh Sunnapu

Contributor: Aighne Kearney

Note: For links to additional documentation for this solution, see Red Hat OpenShift Container Platform on the Dell Technologies Info Hub.
This chapter presents the following topics:

- **Introduction** .................................................................................................................. 9
- **Customizing the Dell EMC switches** ............................................................................... 9
- **Configuring the Dell EMC switches** ............................................................................... 11
- **Configuring the Brocade 6510 FC Switch** .................................................................... 11
Chapter 2: Configuring Switches

Introduction

Overview

Dell Technologies has provided sample switch configuration files in GitHub. These files enable you to easily configure the switches that are used for the OpenShift Container Platform cluster. This chapter describes how to customize these configuration files.

Note: Clone the repository using `git clone https://github.com/dell-esg/openshift-bare-metal.git` and change to the examples directory.

CAUTION: If you use different hardware or require different configurations, modify the configuration files accordingly.

Typographical conventions

Configuration instructions use certain typographical conventions to designate commands and screen output.

Command syntax is in Courier New font. Information that is specific to your environment is in italics and placed inside <> symbols. For example:

- Deployment guide command reference: `OS10(config)number hostname <hostname>`
- On the top S5232F-ON switch, enter: `OS10(config)# hostname SW1`

Sample port configuration between switches

Dell EMC PowerSwitch S5232F switches are in Virtual Link Trunking (VLT) mode and are uplinked to core switches in the environment. PowerSwitch S3048 has an uplink to S5232F switches. The following table shows the port connections between switches. These connections are reflected in the sample switch configuration files.

<table>
<thead>
<tr>
<th>Switch name</th>
<th>Port</th>
<th>Description</th>
<th>S5232F-1</th>
<th>S5232F-2</th>
<th>S3048</th>
</tr>
</thead>
<tbody>
<tr>
<td>S5232F-1</td>
<td>1/1/30</td>
<td>Uplink to core switch</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>S5232F-1</td>
<td>1/1/31</td>
<td>VLTi</td>
<td>X</td>
<td>1/1/31</td>
<td>X</td>
</tr>
<tr>
<td>S5232F-1</td>
<td>1/1/32</td>
<td>VLTi</td>
<td>X</td>
<td>1/1/32</td>
<td>X</td>
</tr>
<tr>
<td>S3252F-1</td>
<td>1/1/34</td>
<td>Uplink to S3048</td>
<td>X</td>
<td>X</td>
<td>1/1/51</td>
</tr>
<tr>
<td>S5232F-2</td>
<td>1/1/30</td>
<td>Uplink to core switch</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>S5232F-2</td>
<td>1/1/34</td>
<td>Uplink to S3048</td>
<td>X</td>
<td>x</td>
<td>1/1/52</td>
</tr>
<tr>
<td>S3048</td>
<td>1/1/4</td>
<td>FC Brocade Switch</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Customizing the Dell EMC switches

The following table shows connections from each server to the switch ports with a 100 GbE NIC in PCI slot 2.

- SW1 (S5232F-1) is connected to port 1 in PCI slot 2.
- SW2 (S5232F-2) is connected to port 2 in PCI slot 2.
S3048 is connected to iDRAC.

### Table 2. Server-to-switch port connections

<table>
<thead>
<tr>
<th>Role</th>
<th>PowerSwitch model</th>
<th>iDRAC IP VLAN 34 192.168.34.0/24</th>
<th>Public IP VLAN 461 192.168.46.0/24</th>
</tr>
</thead>
<tbody>
<tr>
<td>csah</td>
<td>S5232F-1</td>
<td>192.168.34.20</td>
<td>192.168.46.20</td>
</tr>
<tr>
<td></td>
<td>S5232F-2</td>
<td>192.168.34.21</td>
<td>192.168.46.21</td>
</tr>
<tr>
<td></td>
<td>S3048</td>
<td>192.168.34.22</td>
<td>192.168.46.22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.34.23</td>
<td>192.168.46.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.34.24</td>
<td>192.168.46.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.34.25</td>
<td>192.168.46.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.34.26</td>
<td>192.168.46.26</td>
</tr>
</tbody>
</table>

The following table shows the firmware versions that are running in each switch model:

### Table 3. Switch firmware version and default credentials

<table>
<thead>
<tr>
<th>PowerSwitch model</th>
<th>Out-of-band management IP</th>
<th>Firmware version</th>
<th>Default username</th>
<th>Default password</th>
</tr>
</thead>
<tbody>
<tr>
<td>S5232F-1</td>
<td>192.168.33.44</td>
<td>10.5.1.0</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>S5232F-2</td>
<td>192.168.33.45</td>
<td>10.5.1.0</td>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>S3048</td>
<td>192.168.33.46</td>
<td>10.5.1.0</td>
<td>admin</td>
<td>admin</td>
</tr>
</tbody>
</table>

To modify the switches for your environment:

1. Download the switch configuration files from GitHub:
   
   ```bash
git clone https://github.com/dell-esg/openshift-bare-metal.git
   ```

2. Modify the sample switch configuration files in `<git clone dir>/examples` to match your VLAN and IP schemes.

The deployment uses untagged VLANs that use switchport access for nodes and tagged port channels for switch uplinks. The deployment sample uses:

- VLAN_461 configured public network
- VLAN_34 configured for management network
- Single 100 GbE Mellanox X5 DP NIC in PCI slot 2

**Notes:**

The serial-port baud rate is 115200.

This guide uses Ethernet ports ens2f0 and ens2f1 in R640 servers for Red Hat Enterprise Linux CoreOS and p2p1, p2p2 for Red Hat Enterprise Linux 7.x.
Configuring the Dell EMC switches

This section describes Dell EMC Networking OS10 initial OOB management IP setup and provides sample switch configuration directions that are copied to: running-configuration

Follow these steps:

1. Power on the switches, connect to the serial debug port, set the hostname, and configure a static IP address for management 1/1/1.

   The following code sample shows an S5232F-1 switch. Use the same process for S5232F-2 and S3048 switches.

   ```
   OS# configure terminal
   OS(config)# hostname S5232F-1
   S5232F-1(config)# interface mgmt 1/1/1
   S5232F-1(config-if-ma-1/1/1)# no shutdown
   S5232F-1(config-if-ma-1/1/1)# no ip address dhcp
   S5232F-1(config-if-ma-1/1/1)# ip address 192.168.33.44/24
   S5232F-1(config-if-ma-1/1/1)# exit
   S5232F-1(config)# management route 0.0.0.0/0 192.168.33.1
   ```

2. Copy the modified sample switch configuration to running-configuration and configure the switch:

   ```
   S5232F-1# copy scp://<user>@<hostip>/<path to downloaded S5232F config file> running-configuration
   S5232F-1# write memory
   ```

3. Repeat the preceding steps for switch S5232F-2 and S3048, ensuring that the appropriate switch configuration file is copied, and IP addresses are modified accordingly.

Configuring the Brocade 6510 FC Switch

To use CSI drivers that support the FC protocol, ensure that all compute nodes in the cluster include FC cards.

1. Use a serial cable to connect to the FC switch.

2. Use the ipaddrset command to set up the IP address in the management network for the FC switch.

3. Create aliases for each compute node and storage node that will use the FC WWN of the ports that are associated with each server or storage:

   ```
   R3FC:FID128:admin> alicreate “alias name”, “WWN 1; WWN 2; WWN 3; WWN4”
   ```

4. Create zoning to ensure that the server and storage are visible to each other:

   ```
   R3FC:FID128:admin> zonecreate “zone name”, “server alias name; storage alias name”
   ```
5. Add the zone name to the zone configuration:

   R3FC:FID128:admin> cfgadd "cfg name", "zone name"

6. Save and enable the FC configuration:

   R3FC:FID128:admin> cfgsave
   R3FC:FID128:admin> cfgenable "cfg name"

**Note:** For more information; see the sample configuration file in GitHub.
This chapter presents the following topics:

Overview........................................................................................................................................... 14
Preparing the CSAH node............................................................................................................. 14
Preparing and running the Ansible playbooks............................................................................ 17
Chapter 3: Setting Up the CSAH Node

Overview

This chapter describes the prerequisites for creating an OpenShift Container Platform cluster. Services required to create the cluster are set up in the Cluster System Admin Host (CSAH) node. The chapter provides information about installing Red Hat Enterprise Linux 7.9 in the CSAH node and running the OpenShift Container Platform cluster prerequisites.

Preparing the CSAH node

To install Red Hat Enterprise Linux 7.9 in the CSAH node:

1. Follow the guidelines in the Red Hat Enterprise Linux 7 Installation Guide.
2. In the Red Hat Enterprise Linux UI, as shown in the following figure under SOFTWARE SELECTION, ensure that Server with GUI is selected.

![Operating system installation options](image)

Figure 1. Operating system installation options

Note: Ansible playbooks that are described in this guide use packages that are installed with the Server with GUI option.
3. After the installation is complete, perform the following tasks as user root unless specified otherwise:
   
   a. Set the hostname to reflect the naming standards:
      
      ```bash
      [root@csah ~]# hostnamectl set-hostname <hostname FQDN>
      ```
   
   b. Create a bridge interface and a bond interface with bridge as the control-plane interface (in the code, "master"). Add "slaves" (secondary interfaces) to the bond and then assign an IP address to the bridge interface.
      
      As part of our validation, we created bond0 using 100 Gb NIC ports in slot 2 (p2p1 and p2p2):
      
      ```bash
      # Create bridge interface
      [root@csah ~]# nmcli connection add type bridge ifname br0 con-name bridge-br0
      # Create bond interface with bridge bridge-br0 as master
      nmcli connection add type bond con-name bond0 ifname bond0 bond.options "lacp_rate=1,miimon=100,mode=802.3ad,xmit_hash_policy=layer3+4" master bridge-br0
      # Add slaves to bond0 interface
      [root@csah ~]# nmcli con add type ethernet ifname p2p1 master bond0
      [root@csah ~]# nmcli con add type ethernet ifname p2p2 master bond0
      # Set IP Address to bridge-br0 interface
      [root@csah ~]# nmcli connection modify bridge-br0 ipv4.method manual ipv4.addresses 192.168.46.20/26 connection.autoconnect yes ipv4.gateway 192.168.46.1 ipv4.dns 192.168.32.10 ipv4.dns-search example.com
      # Bring up bridge-br0 interface
      [root@csah ~]# nmcli connection up bridge-br0
      [root@csah ~]# systemctl restart NetworkManager
      ```
   
   c. Enable the rhel-7-server-ansible-2.9-rpms, rhel-server-rhsc1-7-rpms, and rhel-7-server-ose-4.6-rpms repositories by running:
      
      ```bash
      [root@csah ~]# subscription-manager register --username <subscription.user> --password <subscription.password> --force
      [root@csah ~]# subscription-manager attach --pool=<pool id>
      ```
   
   **Notes:**
   
   Check the /proc/net/bonding/bond0 file and ensure that it lists the slave interfaces (p2p1 and p2p2) along with the bonding configuration.
   
   The assigned IP address must be able to reach the Internet, and the DNS must be able to resolve subscription.rhsm.redhat.com.
Chapter 3: Setting Up the CSAH Node

[root@csah ~]# subscription-manager repos --enable=rhel-7-server-ansible-2.9-rpms --enable=rhel-server-rhscl-7-rpms --enable=rhel-7-server-ose-4.6-rpms
d. Install the following RPMs:

[root@csah ~]# yum install -y git ansible python-netaddr rh-python38-python rh-python38-python-pyyaml rh-python38-python-requests

Package git-1.8.3.1-23.el7_8.x86_64
Package ansible-2.9.14-1.el7ae.noarch
Package python-netaddr-0.7.5-9.el7.noarch
Package rh-python38-python-3.8.6-1.el7.x86_64
Package rh-python38-python-pyyaml-5.3.1-1.el7.x86_64
Package rh-python38-python-requests-2.22.0-10.el7.noarch

4. Create a user to run the playbooks.

**Note:** Do not use the username core. User core is part of the OpenShift Container Platform cluster configuration and is a predefined user in CoreOS.

[root@csah ~]# useradd ansible
[root@csah ~]# passwd ansible
Changing password for user ansible.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

**Note:** The remainder of this guide assumes that user ansible runs the playbooks.

5. As user root, provide sudoers permissions to user ansible by running:

[root@csah sudoers.d]# pwd
/etc/sudoers.d
[root@csah sudoers.d]# cat ansible
ansible ALL=(ALL) NOPASSWD: ALL

6. As user ansible, set up passwordless access to the CSAH FQDN:

[ansible@csah ~]$ ssh-keygen (press enter and go by defaults for the next set of questions)
[ansible@csah ~]$ cat .ssh/id_rsa.pub > .ssh/authorized_keys
[ansible@csah ~]$ chmod 600 .ssh/authorized_keys

7. Download the Ansible playbooks from GitHub:

[ansible@csah ~]$ git clone https://github.com/dell-esg/openshift-bare-metal.git

Preparing and running the Ansible playbooks

As user ansible (unless otherwise specified), prepare and run the Ansible playbooks.

**Note:** Ensure that the CSAH node can reach the iDRAC network IPs. If there is no connectivity, manually create the inventory file by following the steps in the sample file in GitHub.

1. Update a YAML file containing information about bootstrap, control-plane, and compute nodes.

   **Note:** Ensure that only values in the YAML file are modified. Keys must always remain the same.

   - For bootstrap, which is created as a kernel-based virtual machine (KVM), specify only the operating system IP address.
   - For control plane nodes, specify both the operating system and iDRAC IP address.
   - For compute nodes, specify the operating system also. This information is necessary because compute nodes support Red Hat Enterprise Linux 7.9 and RHCOS 4.6. The supported value for the ‘os’ key is either rhcos or rhel.

2. Run the Python scripts in the <git clone directory>/python directory to create an inventory file automatically for Ansible playbooks:

   ```bash
   [ansible@csah python]$ python3 generate_inventory_file.py
   usage: generate_inventory_file.py [-h] [--run | --add] --ver {4.6} --nodes NODES [--id_user ID_USER] [--id_pass ID_PASS] [--debug]
   ```

   Generate Inventory

   optional arguments:
   -h, --help         show this help message and exit
   --run             generate inventory file
   --add             number of compute nodes
   --ver {4.6}       specify OpenShift version
   --nodes NODES     nodes inventory file
   --id_user ID_USER specify idrac user
   --id_pass ID_PASS specify idrac user
   --debug           specify debug logs

   **Note:** If the iDRAC user and password are the same across all control-plane and compute nodes, run the program with arguments --id_user and --id_pass.

3. Run the program:

   ```bash
   [ansible@csah python]$ python3 generate_inventory_file.py
   --run --id_user <user> --id_pass <password> --ver 4.6 --nodes nodes.yaml
   ```
Chapter 3: Setting Up the CSAH Node

Note: In the argument that is passed, --ver 4.6 specifies the OpenShift version. Currently, the script accepts only one value: 4.6. The nodes.yaml file that you updated in Step 1 includes information about the bootstrap, control-plane, and compute nodes.

A list of numbered tasks is displayed, as shown in the following figure:

Figure 2. Inventory file generation input tasks menu

4. Select the number of each task in turn and provide the requested input.

Note: If you are unsure about what value to enter for an option, accept the default value if it is provided.

a. For option 1, specify the directory to which to download the files:

provide complete path of directory to download OCP 4.6 software bits
default [/home/ansible/files]:

Option 1 downloads OpenShift Container Platform 4.6 software from Red Hat into a directory for which user ansible has permissions. This guide assumes that the directory is specified as /home/ansible/files.

b. For option 2:

   i. Enter the cluster installation options by selecting 3 node or 6+ node:

   task choice for necessary inputs: 2
   supported cluster install options:
   1. 3 node (control/compute in control nodes)
   2. 6+ node (3 control and 3+ compute)
   enter cluster install option: 2

Note: OpenShift 4.6 supports the 3 node and 6+ node cluster options. The following example shows the steps to follow if you select a 6+ node cluster installation. If you select the 3 node installation option, you are not prompted for information about compute nodes.
ii Specify the bootstrap node name and the IP address to be assigned to the bootstrap node:

```
enter the bootstrap node name
default [bootstrap]:
ip address for os in bootstrap node: 192.168.46.19
```

**Note:** Leave the IP address 192.168.42.19 that you specified in the preceding step unassigned. The bootstrap node is created as a KVM using `virt-install`.

iii Specify the number of control-plane nodes in the cluster and provide additional details as appropriate.

**Note:** The following example assumes that three control-plane nodes are set up in the cluster. NIC.Slot.2-1-1 is used for DHCP, and PXE boot is enabled in the interface. Bonding is performed through two interfaces: NIC.Slot.2-1-1 and NIC.Slot.2-2-1. If only one interface is available, specify NO.

Do you want to perform bonding (y/NO): y
ip address for os in etcd-0 node: 192.168.46.21
ip address for idrac in etcd-0 node: 192.168.34.21
1 -> NIC.Integrated.1-1-1
2 -> NIC.Integrated.1-2-1
3 -> NIC.Slot.2-1-1
4 -> NIC.Slot.2-2-1
Select the interface used by DHCP: 3
selected interface is: NIC.Slot.2-1-1
device NIC.Slot.2-1-1 mac address is B8:59:9F:C0:36:46
1 -> NIC.Integrated.1-1-1
2 -> NIC.Integrated.1-2-1
3 -> NIC.Slot.2-1-1
4 -> NIC.Slot.2-2-1
Select the interface used by etcd-0 active bond interface: 3
selected interface is: NIC.Slot.2-1-1
1 -> NIC.Integrated.1-1-1
2 -> NIC.Integrated.1-2-1
3 -> NIC.Slot.2-1-1
4 -> NIC.Slot.2-2-1
Select the interface used by etcd-0 backup bond interface: 4
selected interface is: NIC.Slot.2-2-1

**Note:** The selected network interface determines the calculation for network enumeration `ens2fo`. This network enumeration logic is tested in PowerEdge R640 servers. Select two interfaces, one for each “slave” interface in bond.

c. Repeat the preceding step for the remaining control-plane nodes.
d. After you have entered the control-plane node information, provide the compute node information by entering the default number of compute nodes.  

**Note:**  
This step is not necessary if you selected 3 node in Step 4, substep b.i. The compute node supports either Red Hat Enterprise Linux 7.9 or RHCOS 4.6 as the operating system.

Specify information relating to bonding and the interfaces that bonding uses for each compute node (see Step 4, substep b.iii for control-plane nodes).

e. For option 3, provide details about the disks that are used in control-plane and compute nodes:

```
ensure disknames are absolutely available. Otherwise OpenShift install fails
specify the control plane device that will be installed
default [nvme0n1]:
specify the compute node device that will be installed
default [nvme0n1]:
```

**Note:** This guide assumes that the NVMe drive in the first slot is used for the OpenShift installation.

f. For option 4, provide the cluster name and the DNS zone file name:

```
specify cluster name
default [ocp]:
specify zone file
default [/var/named/ocp.zones]:
```

g. For option 5, provide details for the HTTP web server setup and directory names that are created under /var/www/html:

```
enter http port
default [8080]:
specify dir where ignition files will be placed
directory will be created under /var/www/html
default [ignition]:
```

h. For option 6, provide details about the default user that is used to install the OpenShift Container Platform cluster, the service network CIDR, pod network CIDR, and other information to be added in the install-config.yaml file:

```
enter the user used to install openshift
DONOT CHANGE THIS VALUE
default [core]:
enter the directory where openshift installs
directory will be created under /home/core
default [openshift]:
enter the pod network cidr
default [10.128.0.0/14]:
pod network cidr: 10.128.0.0/14
```
Chapter 3: Setting Up the CSAH Node

specify cidr notation for number of ips in each node:
cidr number should be an integer and less than 32
default [23]:
specify the service network cidr
default [172.30.0.0/16]:

**Note:** Do not change the user value from `core`. Only the `core` user can connect into cluster nodes by using SSH. The CNI options are the specified information.

i. Select option 7 to print the inputs that you have provided.
   To modify any values, run the related option again and correct the values.

j. Select option 8 to perform a YAML dump of all the displayed contents into the `generated_inventory` file in the current directory (see this [sample file](https://github.com) in GitHub for guidance).

5. Download Red Hat Enterprise Linux 7.9 nodes from the [Red Hat Customer Portal](https://www.redhat.com) to install on the compute nodes. (Red Hat account credentials are required.)

6. Log in to Red Hat to download the `pullsecret` file (Red Hat account credentials are required). Copy the file contents into the `pullsecret` file in the directory containing the OpenShift Container Platform 4.6 software bits.

   **Note:** This guide uses the `/home/ansible/files` directory containing the software bits.

7. In the `generated_inventory` file, add the following content under the `software_src` key, and then save the file:

   ```yaml
   vars:
     software_src: /home/ansible/files
     pull_secret_file: pullsecret
     rhel_os: rhel-server-7.9-x86_64-dvd.iso
   ``

   **Note:** Copy the `generated_inventory` file from the `<git clone dir>/python` directory to `<git clone dir>/ansible` directory.

8. As user `ansible`, run the playbooks:

   ```bash
   [ansible@csah ansible] $ pwd
   /home/ansible/openshift-bare-metal/ansible
   [ansible@csah ansible] $ ansible-playbook -i generated_inventory ocp.yaml
   ```

   The CSAH node is installed and configured with HTTP, HAProxy, DHCP, DNS, and PXE services. Also, the `install-config.yaml` file is generated, and the ignition config files are created and made available over HTTP.

   **Note:** If any errors occur while the program is running, see the `inventory.log` file under the `<git clone dir>/python` directory to find out what went wrong and how to resolve the issue.
Chapter 4  Deploying OpenShift 4.6

This chapter presents the following topics:

- Introduction ............................................................................................................. 23
- Creating a bootstrap KVM ....................................................................................... 23
- Installing control-plane nodes ............................................................................... 25
- Completing the bootstrap setup ............................................................................... 27
- Installing compute nodes ....................................................................................... 27
- Completing the cluster setup ................................................................................ 30
- Removing the bootstrap node ............................................................................... 31
- Accessing the OpenShift web console ................................................................. 32
- OpenShift license and support ............................................................................. 32
- Configuring authentication .................................................................................... 34
- Adding a compute node ....................................................................................... 36
Introduction

To create an OpenShift Container Platform cluster, first create a bootstrap KVM, then create the control-plane nodes, and finally create the compute nodes.

Notes:
This guide assumes that NIC in Slot 2 Port 1 is used for PXE installation. If necessary, replace the interface to suit your environment.

All nodes must run in UEFI mode so that the playbooks running in the CSAH node work effectively.

Creating a bootstrap KVM

Start the cluster installation by creating a bootstrap KVM. The bootstrap KVM creates the persistent control plane that the control-plane nodes manage. The bootstrap KVM is created as a VM by using a QEMU emulator in the CSAH node.

1. As user root, run the command that is specified in the bootstrap_command file:

   [root@csah ~]# systemctl get-default graphical.target

2. Ensure that DNS is updated for the bridge interface, which is necessary because the Ansible playbooks configured a DNS setup in the CSAH node:

   [root@csah sudoers.d]# nmcli connection modify bridge-br0 ipv4.dns 192.168.46.20
   [root@csah sudoers.d]# systemctl restart NetworkManager
   [root@csah sudoers.d]# cat /etc/resolv.conf
   # Generated by NetworkManager
   search example.com
   nameserver 192.168.46.20

Note: For KVM to be created using virt-install, a command is generated by the Ansible playbooks and placed under /home/ansible/files directory in a file called bootstrap_command

3. Create the bootstrap VM:

   [root@csah ~] virt-install --name bootstrapkvm --ram 20480 --vcpu 8 --disk path=/home/bootstrapvm-disk.qcow2,format=qcow2,size=200 --os-variant generic --network=bridge=br0,model=virtio,mac=52:54:00:15:91:64 --pxe --boot hd, network &
Chapter 4: Deploying OpenShift 4.6

Note:
Do not change the Mac address. This address is auto generated and added in the dhcpd.conf file by the Ansible playbooks. Adding & at the end ensures that the command is run in the background.

Ensure that the partition used to save the disk has enough size. This example uses /home and allocates 200 G to the qcow2 image used by bootstrap KVM. Configure the graphical display to ensure that the PXE menu is displayed.

The following figure shows the PXE menu. If no graphic menu is set, connect to the virtual console in iDRAC and run the command.

Ensure that PXE is enabled through a bridge interface. Following Red Hat Bugzilla – Bug 533684, as user root, run brctl stp br0 off and brctl setfd br0 2 if there are any issues with PXE boot of KVM.

The bootstrap KVM menu is displayed, as shown in the following figure:

![PXE Menu]

Figure 3. Bootstrap KVM BIOS PXE menu

4. Press Enter to start installing the bootstrap KVM.

5. As user core in CSAH, run ssh bootstrap to ensure that the proper IP address is assigned to bond0.

When the installation process is complete, KVM reboots and boots into the hard disk, as shown in the following figure:
6. As user `core` in CSAH, run `ssh bootstrap` to ensure that the proper IP address is assigned to bond0.

7. From the CSAH node, as user `core`, SSH to the bootstrap node and verify that ports 6443 and 22623 are listening.

Allow approximately 30 minutes for the ports to show up as listening. If the ports are not up and listening after 30 minutes, reinstall the bootstrap by repeating the preceding steps:

```bash
[core@csah ~]$ ssh bootstrap sudo ss -tulpn | grep -E '^6443|22623|2379'
tcp    LISTEN   0       128 *:22623             tcp    LISTEN   0       128 *:6443
*:      users:("machine-config",pid=6972,fd=8))
tcp    LISTEN   0       128 *:6443             tcp    LISTEN   0       128 *:2379
*:      users:("kube-apiserver",pid=7998,fd=8))
tcp    LISTEN   0       128 *:2379             tcp    LISTEN   0       128 *:6036
*:      users:("etcd",pid=6036,fd=5))
```

## Installing control-plane nodes

To install the control-plane nodes:

1. Connect to the iDRAC of a control-plane node and open the virtual console.

2. In the iDRAC UI, click **Configuration** and select **BIOS Settings**.
   a. Expand **Network Settings**.
   b. Set **PXE Device1** to **Enabled**.
   c. Expand **PXE Device1 Settings**.
   d. Set **NIC in Slot 2 Port 1 Partition 1** as the interface.
   e. Scroll to the bottom of the **Network Settings** section and select **Apply**.

The system boots automatically into the PXE network and displays the PXE menu, as shown in the following figure:

![iDRAC console PXE menu](image)
3. Select etcd-0 (the first node), and, after the installation is complete but before the node reboots into the PXE, ensure that the hard disk is placed above the PXE interface in the boot order, as follows:

   a. Press F2 to enter **System Setup**.
   b. Select **System BIOS > Boot Settings > UEFI Boot Settings > UEFI Boot Sequence**.
   c. Select **PXE Device 1** and click `-`
   d. Repeat the preceding step until **PXE Device 1** is at the bottom of the boot menu.
   e. Click **OK** and then click **Back**.
   f. Click **Finish** and save the changes.

4. Let the node boot into the hard drive where the operating system is installed.

5. After the node comes up, ensure that the hostname is displayed as **etcd-0** in the iDRAC console, as shown in the following figure:

   ![Control-plane node (etcd-0) iDRAC console](image)

   **Figure 6. Control-plane node (etcd-0) iDRAC console**

6. Repeat the preceding steps for the remaining two control-plane nodes, selecting **etcd-1** for the second control-plane node and **etcd-2** for the third control-plane node.

7. After all three control-plane (**etcd-**) nodes are installed and running, from the CSAH node, log in to the bootstrap node as user **core** and check the status of the bootkube service:

   ```
   [core@bootstrap ~]$ journalctl -b -f -u release-image.service -u bootkube.service
   Dec 05 08:35:46 bootstrap.example.com bootkube.sh[31257]:
   Sending bootstrap-finished event. Tearing down temporary bootstrapt control plane...
   Dec 05 08:35:46 bootstrap.example.com bootkube.sh[31257]:
   Waiting for CEO to finish...
   Dec 05 08:35:46 bootstrap.example.com bootkube.sh[31257]:
   I1205 08:35:46.764292       1 waitforceo.go:64] Cluster etcd
   operator bootstrapped successfully
   Dec 05 08:35:46 bootstrap.example.com bootkube.sh[31257]:
   I1205 08:35:46.765279       1 waitforceo.go:58] cluster-
   etc-operator bootstrap etcd
   Dec 05 08:35:46 bootstrap.example.com bootkube.sh[31257]:
   bootkube.service complete
   ```
8. Ensure that the output of the `bootkube.service` is complete.

## Completing the bootstrap setup

On the CSAH node:

1. As user `core`, run the following command in `/home/core` to complete the bootstrap process:

   ```
   [core@csah ~]$ ./openshift-install --dir=openshift wait-for bootstrap-complete --log-level debug
   DEBUG OpenShift Installer 4.6.9
   DEBUG Built from commit a48ad4a15b42102d1747d2f5f3b635deffb950b5
   INFO Waiting up to 20m0s for the Kubernetes API at https://api.ocp.example.com:6443...
   INFO API v1.19.0+7070803 up
   INFO Waiting up to 30m0s for bootstrapping to complete...
   DEBUG Bootstrap status: complete
   INFO It is now safe to remove the bootstrap resources
   INFO Time elapsed: 0s
   ```

2. Validate the status of the control-plane nodes:

   ```
   [core@csah ~]$ oc get nodes
   NAME                 STATUS   ROLES    AGE     VERSION
   etcd-0.example.com   Ready    master   83m     v1.19.0+7070803
   etcd-1.example.com   Ready    master   71m     v1.19.0+7070803
   etcd-2.example.com   Ready    master   9m30s   v1.19.0+7070803
   ```

   **Note:** In a 3 node cluster, each `etcd-*` node has an additional ROLE `worker` along with the `master` node.

3. Run the `oc get co` command to view the Cluster Operator status.

   **Note:** In a 6+ node cluster, compute nodes must be in the Ready state before the Cluster Operator AVAILABLE state is displayed as True.

## Installing compute nodes

**Note:** Ignore this section if the cluster is a 3-node setup.

Follow these steps:

1. Connect to the iDRAC of a compute node and open the virtual console.

2. In the iDRAC UI, click Configuration and select BIOS Settings.
   a. Expand Network Settings.
   b. Set PXE Device1 to Enabled.
   c. Expand PXE Device1 Settings.
d. Select **NIC in Slot 2 Port 1 Partition 1** as the interface.

e. Scroll to the bottom of the **Network Settings** section and click **Apply**.

The system automatically boots into the PXE network and displays the PXE menu, as shown in the following figure:

![PXE menu](image)

**Figure 7. iDRAC console PXE menu**

3. Select **compute-1** and let the system reboot after the installation. Before the node reboots into the PXE, ensure that the hard disk is placed above the PXE interface in the boot order:

a. Press F2 to enter System Setup.

b. Select **System BIOS > Boot Settings > UEFI Boot Settings > UEFI Boot Sequence.**

c. Select **PXE Device 1** and click `-`.

d. Repeat step c until **PXE Device 1** is at the bottom of the boot menu.

e. Click **OK** and then click **Back.**

f. Click **Finish** and save the changes.

4. Let the node boot into the hard drive where the operating system is installed, as shown in the following figure:

![RHEL console](image)

**Figure 8. iDRAC console: compute-1**

5. Repeat the preceding steps for the remaining compute nodes. Then:

- Skip steps 6 through 13 if RHCOS is the compute node operating system.
- Continue with steps 6 through 13 if the control node operating system is Red Hat Enterprise Linux 7.9.
6. For Red Hat Enterprise Linux compute nodes, ensure that the default users who are specified in the kickstart file exist, as shown in the following figure:

![iDRAC console: compute-3](image)

**Figure 9. iDRAC console: compute-3**

7. In the CSAH node, as user root, run `ssh compute-3` to ensure that the correct IP address is assigned to bond0.

The default password that is used in the kickstart files is `password`. This default password is also used for user `user` and `ansible`.

8. From the CSAH node and as user `ansible`, copy the ssh keys to the compute node by running:

```
[ansible@csah ~]$ ssh-copy-id compute-3.example.com
```

9. Modify the values of subscription user, password, and pool id in the Ansible `rhel_inv_file` inventory file that is available in the `<git clone dir>/ansible/inventory` directory.

**Note:** Ensure that the hostname for `hosts` key in the `rhel_inv_file` file is correct and that the FQDN is specified (for example, `compute-3.example.com`).

10. Validate the Ansible role `compute.yaml` file that is available in the `<git clone dir>/ansible` directory.

11. On the CSAH node, run the Ansible playbook. The playbook installs all prerequisites that are required to set up the compute node to join the cluster.

```
[ansible@csah ansible]$ pwd
/home/ansible/openshift-bare-metal/ansible

[ansible@csah ~]$ ansible-playbook -i inventory/rhel_inv_file compute.yaml
```

12. Modify the `rh_rhel_worker inventory` file in the `<git clone dir>/ansible/inventory` directory with the appropriate values, as shown in the following example:

```
[all:vars]
ansible_user=ansible
ansible_become=True
openshift_kubeconfig_path="/home/ansible/kubeconfig"

[new_workers]
compute-3.example.com
```
13. On the CSAH node, as user ansible run the playbook that Red Hat has provided to add the compute node to the existing cluster:

```
[ansible@csah openshift-ansible]$ pwd
/usr/share/ansible/openshift-ansible
[ansible@csah openshift-ansible]$ ansible-playbook -i <git clone dir>/ansible/inventory/rh_rhel_worker_playbooks/scaleup.yml
```

The playbook runs in approximately 10 minutes. During this time, the compute node reboots and joins the existing cluster by auto approving the Certificate Signing Request (csr).

14. As user core in CSAH node, approve the csr to ensure RHCOS-based compute nodes are added into the cluster.

```
[core@csah ~]$ oc get csr -o name | xargs oc adm certificate approve
```

15. Check the status of the compute nodes and verify that all compute nodes are listed and their status is READY:

```
[core@csah ~]$ oc get nodes
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>ROLES</th>
<th>AGE</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute-1.example.com</td>
<td>Ready</td>
<td>worker</td>
<td>6d7h</td>
<td>v1.19.0+43983cd</td>
</tr>
<tr>
<td>compute-2.example.com</td>
<td>Ready</td>
<td>worker</td>
<td>6d7h</td>
<td>v1.19.0+43983cd</td>
</tr>
<tr>
<td>compute-3.example.com</td>
<td>Ready</td>
<td>worker</td>
<td>81s</td>
<td>v1.19.0+7070803</td>
</tr>
<tr>
<td>etcd-0.example.com</td>
<td>Ready</td>
<td>master</td>
<td>6d12h</td>
<td>v1.19.0+43983cd</td>
</tr>
<tr>
<td>etcd-1.example.com</td>
<td>Ready</td>
<td>master</td>
<td>6d12h</td>
<td>v1.19.0+43983cd</td>
</tr>
<tr>
<td>etcd-2.example.com</td>
<td>Ready</td>
<td>master</td>
<td>6d12h</td>
<td>v1.19.0+43983cd</td>
</tr>
</tbody>
</table>

**Completing the cluster setup**

*Note: This section uses openshift for the install_dir variable. See the inventory file for the value specified for the install_dir variable. By default, the inventory file is placed under <git clone dir>/python/generated_inventory.*

After the bootstrap, control-plane, and compute nodes are installed, complete the cluster setup:

1. Switch to the home directory of core.
2. Run the following command:

   
   ```
   [core@csah ~]$ cd /home/core
   [core@csah ~]$ oc get clusteroperators
   ```

3. Ensure that all the operators are set to True in the AVAILABLE column. After the verification is complete, run:

   ```
   [core@csah ~]$ ./openshift-install --dir=openshift wait-for install-complete --log-level debug
   DEBUG OpenShift Installer 4.6.9
   DEBUG Built from commit db0f93089a64c5fd459d226fc224a2584e8c3be7e
   DEBUG Loading Install Config...
   ```
Removing the bootstrap node

We created a bootstrap node as part of the deployment procedure. Now that the OpenShift Container Platform cluster is running, remove this node.

To remove the bootstrap node:

1. Remove the bootstrap node entries along with the names, IP addresses, and MAC addresses. For example, in the following sample entry for bootstrap_node in the inventory file, remove all entries along with the bootstrap_node: line:

   bootstrap_node:
   - name: bootstrap
     ip: 192.168.46.26
     mac: B8:59:9F:C0:35:86

2. On the CSASH node, run the playbooks as user ansible:

   [ansible@csah ansible] $ sudo virsh list

3. If bootstrap KVM is listed, delete it by running:

   [ansible@csah ansible] $ sudo virsh destroy bootstrapkvm
   [ansible@csah ansible] $ sudo virsh undefine bootstrapkvm
4. Delete the disk that was created under /home for KVM:

```
[ansible@csah ansible]$ sudo rm -rf /home/bootstrapvm-disk.qcow2
```

**Note:** Replace the location of the qcow2 image as appropriate.

### Accessing the OpenShift web console

The OpenShift web console provides access to all cluster functionality, including pod creation and application deployment.

To access OpenShift through a web browser:

1. Obtain the console URL of the routes.
2. Observe the existing routes in all namespaces:

```
[core@csah ~]$ oc get routes --all-namespaces | grep -i console
console-console         console       console
console-console.apps.ocp.example.com
console                  https              reencrypt/Redirect  None
```

**Note:** The URL in the openshift-console namespace is `console-openshift-console.apps.ocp.example.com`.

3. Open a web browser and paste in the URL.
4. Log in as kubeadmin, using the password that was saved in `/home/core/<install dir>/auth/kubeadmin-password`.

### OpenShift license and support

1. Log in to the OpenShift console using the credentials kubeadmin and the password that is provided in the `/home/core/openshift/auth/kubeadmin-password` directory.
2. Under Cluster ID, select **Home > Dashboard > OpenShift Cluster Manager**. The following page opens:
3. Click OpenShift Cluster Manager.

4. Log in using the RH support account.

5. From the Actions drop-down menu, as shown in the following figure, select Edit subscription settings:

Figure 10. OpenShift Cluster Manager

Figure 11. Subscription options
Chapter 4: Deploying OpenShift 4.6

6. Select the support type and other options that you require:

   ![Subscription settings](image)

   Figure 12. Subscription settings

7. Click Save settings.

Configuring authentication

Introduction

OpenShift supports different authentication methods based on the identity provider. For more information, see Understanding authentication in the OpenShift Container Platform documentation.

This section describes how to configure identity providers by using htpasswd.

Creating an admin user

Unless otherwise specified, run the commands in this section on the CSAH node as user core.

To create an admin user:

1. Create an htpasswd file on the CSAH node:

   ```bash
   [core@csah ~]$ cd /home/core/openshift
   [core@csah openshift]$ htpasswd -c -B -b htpasswd ocpadmin Password1
   [core@csah openshift]$ htpasswd -b htpasswd ocpuser Password2
   ```
2. **Create a secret for htpasswd:**

   ```bash
   [core@csah openshift]$ oc create secret generic htpass-secret --from-file=htpasswd=/home/core/openshift/htpasswd -n openshift-config
   ```

3. **Create a custom resource (CR), and save the following contents in a file:**

   ```yaml
   apiVersion: config.openshift.io/v1
   kind: OAuth
   metadata:
     name: cluster
   spec:
     identityProviders:
     - name: htpasswd
       mappingMethod: claim
       type: HTPasswd
       htpasswd:
         fileData:
           name: htpass-secret
   ```

4. **Apply the CR:**

   ```bash
   [core@csah ~]$ oc apply -f <file name>
   ```

5. **Log in as a user that you created with htpasswd:**

   ```bash
   [core@csah ~]$ oc login -u <username>
   Authentication required for https://api.ocp.example.com:6443 (openshift)
   Username: <username>
   Password: <password>
   Login successful.
   You don't have any projects. You can try to create a new project, by running `oc new-project <projectname>`

---

**Assigning a cluster-admin role**

To assign a cluster-admin role to the admin user:

1. **Log in as kubeadmin to assign cluster-admin access:**

   ```bash
   [core@csah openshift]$ oc login -u kubeadmin -p xxxxx-xxxxxx-xxxxxx-xxxxxx-xxxxxx-xxxxxx-xxxxxx-xxxxxx
   Login successful.
   You have access to 53 projects, the list has been suppressed. You can list all projects with 'oc projects'
   Using project "default".
   ```

2. **Run the following command and ensure that the user is listed:**

   ```bash
   [core@csah ~]$ oc get users
   NAME   UID                                    FULL NAME
   IDENTITIES
   ocpadmin 273ccf25-9b32-4b4d-aad4-503c5aa27eee
   htpasswd:ocpadmin
   ```

3. **Obtain a list of all the available cluster roles:**
oc get clusterrole --all-namespaces

4. **Assign the cluster-admin role to the user ocpadmin by running:**

   [core@csah ~]$ oc adm policy add-cluster-role-to-user
cluster-admin ocpadmin
clusterrole.rbac.authorization.k8s.io/cluster-admin added: "ocpadmin"

---

**Adding a compute node**

Compute nodes in OpenShift Container Platform 4.6 can run Red Hat CoreOS or Red Hat Enterprise Linux 7.9.

**Note:** Red Hat does not support running Red Hat Enterprise Linux 8.x on a compute node.

**Modify the nodes YAML file**

**To modify the file:**

1. **Add a new_compute_nodes key in nodes.yaml file.** Under the new_compute_nodes key, add information manually by specifying values such as hostname, IP address for ip_os, IP address for ip_idrac, and supported os.

   **Note:** Supported values for the operating system are rhel and rhcos. The following example uses a Red Hat Enterprise Linux compute node. To find out which values to specify, see this sample file in GitHub.

   ```yaml
   new_compute_nodes:
     - name: compute-3
       ip_os: 192.168.46.27
       ip_idrac: 192.168.34.27
       os: rhel
   ```

   **Note:** On the CSAH node, run the commands as user ansible unless otherwise specified.

2. **Execute the python script to add the new compute nodes into inventory file**

   [ansible@csah python]$ pwd
   /home/ansible/openshift-bare-metal/python

   [ansible@csah python]$ python3 generate_inventory_file.py --add --ver 4.6 --nodes nodes.yaml --id_user <idrac user> --id_pass <idrac_password>

   Enter complete path to existing inventory file:
   /home/ansible/openshift-bare-metal/ansible/generated_inventory

   Do you want to perform bonding for 'new_compute_nodes' (y/NO): y

   **Note:** Specify the bonding information as appropriate. An updated inventory file is created with new compute nodes information added.
3. Run the playbook to create kickstart files, update DNS entries, and set up PXE for the new compute node:

   [ansible@csah ansible]$ pwd
   /home/ansible/openShift-bare-metal/ansible
   [ansible@csah ansible]$ ansible-playbook -i <updated inventory file> ocp.yaml

4. Follow the steps that are described in Installing compute nodes.
This chapter presents the following topics:

Overview ........................................................................................................................................ 39
Defining the CNI .......................................................................................................................... 39
Installing the SR-IOV .................................................................................................................. 39
Overview

This chapter describes the Container Network Interface (CNI) and Single Root Input Output Virtualization (SR-IOV).

A CNI is defined to ensure that all pods and services get an IP address within the OpenShift Cluster. SR-IOV enables us to split the physical network interface into multiple virtual functions (VFs). A VF can then be assigned to the pod as a network interface.

Defining the CNI

By default, the network options are in the install-config.yaml file. See Step 4, substep h in Preparing and running the Ansible playbooks.

The following sample code shows CNI information in the install-config.yaml file:

```yaml
networking:
  clusterNetworks:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
      networkType: OpenShiftSDN
      serviceNetwork:
        - 172.30.0.0/16
```

Where:

- **Cidr**: Range of IP addresses for pods running in the OpenShift cluster.
- **hostPrefix**: Number of IP addresses assigned in each compute node. The value 23 means that there is a limit of 512 IP addresses in each compute node.
- **networkType**: Default CNI driver for OpenShift Cluster, which is `OpenShiftSDN` in this example.
- **serviceNetwork**: Range of IP addresses for services created in the OpenShift Cluster.

Ignition configuration files are created from the information that you define in the install-config.yaml file by using the openshift-install binary file. Creating ignition files is described as a play in the Ansible playbooks. For the specific steps to perform the task, see this GitHub file.

Installing the SR-IOV

The following steps show how to create multiple VFs by using a single physical function (PF), which is a single network device present in the compute node:

1. **Install the SR-IOV network operator** in an `openshift-sriov-network-operator` namespace. To create the namespace, run:
   ```bash
   [core@csah ~]$ oc create namespace openshift-sriov-network-operator
   ```

2. **Log in to the OpenShift console** by following the steps that are described in Accessing the OpenShift web console and install the node feature discovery.
operator. Select **Operators > OperatorHub**, search for **Node Feature Discovery**, and click **Install**.

3. Keep the defaults under the **Install Operator** options and click **Install**. Verify that the Node Feature Discovery POD is in the Running state:

   ```bash
   [core@csah ~]$ oc get pods
   NAME                            READY   STATUS    RESTARTS AGE
   NAME                            READY   STATUS    RESTARTS AGE
   nfd-operator-55487fd584-4rwj9   1/1     Running   0          41s
   ```

4. In the OpenShift console, install the SR-IOV network operator.

5. Select **Operators > OperatorHub**, search for **SR-IOV**, and click **Install**.  

   **Note:** Ensure that the project is set to openshift-sriov-network-operator and select **Install**.

6. Verify that the pods are created in the openshift-sriov-network-operator project and are in the Running state:

   ```bash
   [core@csah ~]$ oc get pods -n openshift-sriov-network-operator
   NAME                            READY   STATUS    RESTARTS AGE
   NAME                            READY   STATUS    RESTARTS AGE
   ```

7. Verify that all the compute nodes are listed by running:

   ```bash
   oc get SriovNetworkNodeState
   ```

   **Note:** The output lists all the compute nodes that are part of the cluster. This guide shows examples for only one compute node. Use the same steps on other compute nodes also.

8. Gather information about the network device card in each compute node by running:

   ```bash
   [core@csah sriov]$ oc get SriovNetworkNodeState compute-1.example.com -o yaml -n openshift-sriov-network-operator
   ```

   **Note:** The command output includes information about the deviceID, pciAddress, and vendor for each NIC. For a list of supported network devices for SR-IOV, see [About Single Root I/O Virtualization (SR-IOV) hardware networks](#).

9. Create a network node policy by following the steps in this sample file in GitHub. Include information about pfNames and rootDevices that you obtained in the preceding step:

   ```bash
   [core@csah sriov]$ oc create -f <YAML file>
   ```

   **Note:** In the sample YAML file, five virtual functions are created in each interface eno1 and eno2.

10. Verify that the network node policy has been created successfully:

    ```bash
    [core@csah sriov]$ oc get sriovnetworknodepolicies compute-1-network-node-policy
    NAME                            AGE
    compute-1-network-node-policy   33s
    ```

    **Note:** The compute-1.example.com node reboots to apply the network node policy.
11. After the node reboots, run `lspci | grep -i virtual` as user core in the compute node to validate that the virtual functions are created successfully.

To test the functionality, the NICs that are used as part of network node policy are connected to a separate VLAN 150 created in S5232F-1, S5232F-2 with IP address 192.168.150.1/24 and 192.168.150.2/24.

12. Create a SRIoVNetwork and assign a static IP address to the virtual function by following the steps in this sample file in GitHub. Verify that the network is created:

```
[core@csah sriov]$ oc create -f <YAML file>
[core@csah sriov]$ oc get sriovnetwork
NAME          AGE       
compute-1-vf0-sriov-network   8s
```

The IP address 192.168.150.51/24 is assigned to the network device.

13. Create a pod and attach the SRIoVNetwork that you created by following the steps in this sample file in GitHub:

```
[core@csah sriov]$ oc create -f <YAML file>
[core@csah sriov]$ oc get pod compute-1-pod --o wide
NAME           READY STATUS     RESTARTS AGE IP NODE
compute-1-pod   1/1      Running 0 5m21s 10.131.2.8 compute-1.example.com <none> <none>
```

14. Verify the interfaces that you assigned in the pod:

```
[core@csah sriov]$ oc exec -it compute-1-pod -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
3: eth0@if41: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc mq state UP group default qlen 1000
    link/ether 0a:58:0a:83:02:08 brd ff:ff:ff:ff:ff:ff inet 10.131.2.8/23 brd 10.131.3.255 scope global eth0
      valid_lft forever preferred_lft forever inet6 fe80::8c0b:d0ff:fe54:d04d/64 scope link
      valid_lft forever preferred_lft forever
37: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether da:b0:ca:69:be:b2 brd ff:ff:ff:ff:ff:ff inet 192.168.150.51/24 brd 192.168.150.255 scope global
      valid_lft forever preferred_lft forever inet6 fe80::d8b0:caff:fe69:beb2/64 scope link
      valid_lft forever preferred_lft forever
```
15. Ping the VLAN IPs that you configured in the S5232F-1 and S5232F-2 switches:

```bash
[core@csah sriov]$ oc exec -it compute-1-pod -- ping 192.168.150.1
PING 192.168.150.1 (192.168.150.1) 56(84) bytes of data.
64 bytes from 192.168.150.1: icmp_seq=1 ttl=64 time=0.308 ms
64 bytes from 192.168.150.1: icmp_seq=2 ttl=64 time=0.337 ms
64 bytes from 192.168.150.1: icmp_seq=3 ttl=64 time=0.307 ms
^C
--- 192.168.150.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2078ms
rtt min/avg/max/mdev = 0.307/0.317/0.337/0.020 ms

[core@csah sriov]$ oc exec -it compute-1-pod -- ping 192.168.150.2
PING 192.168.150.2 (192.168.150.2) 56(84) bytes of data.
64 bytes from 192.168.150.2: icmp_seq=1 ttl=64 time=0.480 ms
64 bytes from 192.168.150.2: icmp_seq=2 ttl=64 time=0.340 ms
64 bytes from 192.168.150.2: icmp_seq=3 ttl=64 time=0.304 ms
64 bytes from 192.168.150.2: icmp_seq=4 ttl=64 time=0.387 ms
^C
--- 192.168.150.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3101ms
rtt min/avg/max/mdev = 0.304/0.377/0.480/0.070 ms
```
Chapter 6 Provisioning Storage

This chapter presents the following topics:

- Introduction ................................................................. 44
- Prerequisites .................................................................. 44
- Unity XT storage ......................................................... 46
- PowerMax storage ........................................................ 51
- PowerScale storage ....................................................... 56
- PowerStore storage ....................................................... 61
- OpenShift Container storage ....................................... 65
Chapter 6: Provisioning Storage

Introduction

OpenShift Container Platform cluster administrators can map storage to containers. For a list of supported PV plug-ins, see Types of PVs in the OpenShift Container Platform documentation.

This chapter describes how to use Dell CSI drivers to configure iSCSI and FC storage for PowerMax, PowerScale (formerly Isilon), and Unity storage units. Topics that are discussed are:

- Installing the CSI Operator
- Specifying prerequisites for installing CSI drivers
- Installing CSI drivers for PowerMax, Isilon, and Unity with support for FC, iSCSI or NFS storage protocols
- Creating static and dynamic PVs by using CSI drivers

Prerequisites

Ensure that:

- OpenShift cluster 4.6 is running with multiple compute nodes that are running RHCOS or Red Hat Enterprise Linux 7.9.

  **Note:** Red Hat does not support running Red Hat Enterprise Linux 8.x on a compute node.

- Dell EMC PowerMax, PowerScale, and Unity storage systems are properly configured.
- FC switches are configured with proper zoning and compute nodes, and PowerMax or Unity storage systems are accessible to each other.
- PowerMax is configured for iSCSI or FC; Unity is configured for iSCSI, FC, and NFS; PowerScale storage systems are configured for NFS.

Follow these steps:

1. **Obtain base64 content of the multipath.conf file:**

   ```bash
   [core@csah multipathd]$ echo 'defaults {
   user_friendly_names yes
   find_multipaths yes
   }

   blacklist {
   }' | base64 -w0
   ```

2. **Create a machine config YAML file and specify the base64 contents in the file by following the steps in this sample file in GitHub. As user core in the CSAH node, run:**

   ```bash
   [core@csah multipathd]$ oc create -f <machine config YAML>
   ```
Chapter 6: Provisioning Storage

**Note:** The preceding machine configuration applies only to compute nodes. After the machine configuration file is created, every compute node is rebooted one at a time after the configuration is applied.

---

### Installing the CSI Operator

The Dell CSI Operator for OpenShift is available in the operator hub.

**Note:** Install the Dell CSI Operator before you install any of the CSI drivers for the installed storage system.

To install the CSI Operator:

1. Log in to the console using the `kubeadmin` username and password that are provided in the `/home/core/openshift/auth/kubeadmin-password` directory.

2. Create a project in the CLI or from the OpenShift Console.

   ```bash
   [core@csah ~]$ oc new-project dell-csi-operators
   ```

3. Go to **Operators > OperatorHub** and type Dell in the **Filter by keyword** search option.

4. Select **DELL EMC CSI Operator** and click **Install**, as shown in the following figure:

   ![CSI Operator Installation](image)

   **Figure 13. CSI Operator Installation**

5. Specify the namespace and select “A specific namespace on the cluster” under **Installation Mode**. Click **Install**.

   The Subscription page opens, as shown in the following figure:
Chapter 6: Provisioning Storage

Figure 14. CSI Operator Subscription

6. Validate the CSI Operator installation by running:

   $ oc get pods -n dell-openshift-operators

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>csi-operator-7bfc7fd59c-8q4hs</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>4m3s</td>
</tr>
</tbody>
</table>

Unity XT storage

Dell EMC Unity is a midrange storage platform that is designed for performance and efficiency. For more information, see [Dell EMC Unity](#).

Ensure that:

- Dell CSI Operator is installed. See [Installing the CSI Operator](#).
- The Dell EMC Unity storage system is configured properly.
- Storage pools have been created along with FC ports, iSCSI interfaces are configured, and the NFS configured as necessary.

To provision Dell EMC Unity storage:

1. Create the namespace:

   $ oc new-project unity

2. Create an empty secret (see this [sample file](#) in GitHub),

3. Apply the empty secret:

   $ oc create -f <secret YAML file>

4. Create a JSON configuration file containing the IP address and Unity array ID. Enter the username and password that you used to log in to Unity through Unisphere. Ensure that the username and password are specified in base64 format (see this [sample file](#) in GitHub).

5. Create a secret by using the JSON file:
Configure the Unity CSI driver

**Note:** You can use the Unity CSI driver to create volumes using either the iSCSI, FC, or NFS protocols.

Create the CSI Unity driver file by following the steps in this [sample file](#) in GitHub.

**Note:** Obtain the arrayID, storagePool, and nasServer information from the UniSphere UI.

1. Create the Unity setup using the driver file that you created in step 1:

   ```
   [core@csah unity]$ oc create -f <YAML file>
   ```

2. Ensure that the Unity pods are running in the unity namespace:

   ```
   [core@csah unity]$ oc get pods -n unity -o wide
   ```

   **Note:** There are two Unity controller pods and one Unity node pod running for every compute node that is running. The Unity pod in each compute node automatically adds any FC or iSCSI Qualified Name into the Unisphere Unity.

3. Verify that the storage classes for Unity are created. Because the CSI driver is the same for the iSCSI, FC, and NFS protocols, three storage classes are created:

   ```
   [core@csah unity]$ oc get sc
   ```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROVISIONER</th>
<th>RECLAIMPOLICY</th>
<th>VOLUMEBINDINGMODE</th>
<th>ALLOWVOLUMEEXPANSION</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>unity-iscsi</td>
<td>csi-unity.dellemc.com</td>
<td>Delete</td>
<td>True</td>
<td>2m30s</td>
<td></td>
</tr>
<tr>
<td>unity-nfs</td>
<td>csi-unity.dellemc.com</td>
<td>Delete</td>
<td>True</td>
<td>2m30s</td>
<td></td>
</tr>
<tr>
<td>unity-fc</td>
<td>csi-unity.dellemc.com</td>
<td>Delete</td>
<td>True</td>
<td>2m30s</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

Configure iSCSI for Unity Create dynamic iSCSI volumes

**Note:** For compute nodes running RHCOS, the `/etc/iscsi/initiatorname.iscsi` file is missing with OCP version 4.6.8 or earlier. For more information, see [Red Hat Bug908847 - RHCOS 4.6 - Missing Initiatorname](#).

Create a YAML file with which to create a PVC by following the steps in this [sample file](#) in GitHub.

1. Create a PVC using the YAML file:

   ```
   [core@csah ~]$ oc create -f <YAML file>
   ```

   **Note:** Ensure that the storage class name designates an iSCSI configuration. This guide uses `unity-iscsi` as the iSCSI storage class name.

2. Verify that the volume is created

   ```
   [core@csah ~]$ oc get pvc
   ```
Chapter 6: Provisioning Storage

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
<th>VOLUME</th>
<th>CAPACITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS MODES</td>
<td>STORAGECLASS</td>
<td>AGE</td>
<td></td>
</tr>
<tr>
<td>dynamic-iscsi-unity-pvc</td>
<td>Bound</td>
<td>csiunity-dd5f2628cb</td>
<td>7Gi</td>
</tr>
<tr>
<td>RWO</td>
<td>unit-iscsi</td>
<td>21m</td>
<td></td>
</tr>
</tbody>
</table>

**Attach a pod to the iSCSI volume**

Create a YAML file with which to create a pod, and mount a volume using the PVC you created in step 2 of *Create dynamic iSCSI volumes*. Follow the steps in this sample file in GitHub.

1. Create a pod:

   [core@csah unity] $ oc create -f <YAML file>

2. Verify that the pod is created and the Unity volume is mounted:

   [core@csah unity]$ oc get pods dynamic-iscsi-unity-pod -o wide

   NAME                      STATUS    RESTARTS   AGE   IP
   NODE                      NOMINATED NODE   READINESS GATES
   dynamic-iscsi-unity-pod   1/1         Running   28s   10.131.0.168   compute-2.example.com

   [core@csah unity]$ oc exec -it dynamic-iscsi-unity-pod -- df -h /usr/share/nginx/html

   Filesystem          Size  Used Avail Use% Mounted on
   /dev/mapper/mpathc  6.9G   32M  6.5G   1% /usr/share/nginx/html

**Configure FC Unity**

**Create dynamic FC volumes**

It is not necessary to create a volume manually when you use a dynamic volume. The PV is created automatically and is attached to the PVC through a YAML file.

Create a YAML file to create the PVC by following the steps in this sample file in GitHub.

1. Use the YAML file that you created in step 1 to create the dynamic PV:

   [core@csah unity] $ oc create -f <YAML file>

   **Note:** Ensure that the storage class name designates an FC configuration. This guide uses unity-fc as the FC storage class name.

2. Verify that the volume is created:

   [core@csah unity]$ oc get pvc

   NAME                      STATUS    VOLUME            CAPACITY   ACCESS MODES   STORAGECLASS | AGE
   dynamic-fc-unity-pvc      Bound   csiunity-64e67b27c4   6Gi       RWO          unity-fc       24m

**Attach a pod to the FC volume**

Create a YAML file with which to create a pod, and mount a volume using the PVC you created in step 2 of *Create dynamic FC volumes*. Follow the steps in this sample file in GitHub.

1. Create a pod:
[core@csah unity] $ oc create -f <YAML file>

2. Verify that the pod is created and the PowerMax volume is mounted:

   [core@csah unity]$ oc get pods dynamic-fc-unity-pod -o wide
   NAME        READY STATUS    RESTARTS AGE IP
   NODE       NOMINATED NODE READINESS GATES
   dynamic-fc-unity-pod 1/1 Running 0 2m59s
   10.131.0.167 compute-2.example.com <none> <none>

   [core@csah unity]$ oc exec -it dynamic-fc-unity-pod -- df -h
   /usr/share/nginx/html

   Filesystem Size Used Avail Use% Mounted on
   /dev/mapper/mpathb 5.9G 24M 5.6G 1% /usr/share/nginx/html

**NFS Unity setup**

*Create dynamic NFS volumes*

Create a YAML file with which to create the PVC by following the steps in this sample file in GitHub.

1. Use the YAML file that you created in step 1 to create the dynamic PV:

   [core@csah unity] $ oc create -f <YAML file>

   **Note:** Ensure that the storage class name designates an NFS configuration. This guide uses *unity-nfs* as the NFS storage class name.

2. Verify that the volume is created:

   core@csah unity]$ oc get pvc
   NAME       STATUS VOLUME
   CAPACITY ACCESS MODES STORAGECLASS AGE
   dynamic-nfs-unity-pvc Bound csiunity-f81650fca8 4Gi
   RWX unity-nfs 20m

**Attach a pod to the NFS volume**

Create a YAML file with which to create a pod, and mount a volume using the PVC you created in step 2 of *Create dynamic NFS volumes*. For guidance, see this sample file in GitHub.

1. Create a pod:

   [core@csah unity] $ oc create -f <YAML file>

2. Verify that the pod is created and the PowerMax volume is mounted:

   [core@csah unity]$ oc get pods dynamic-nfs-unity-pod -o wide
   NAME        READY STATUS    RESTARTS AGE IP
   NODE       NOMINATED NODE READINESS GATES
   dynamic-nfs-unity-pod 1/1 Running 0 2m59s
   10.131.0.12 compute-2.example.com <none> <none>

   [core@csah unity]$ oc exec -it dynamic-nfs-unity-pod -- df -h
   /usr/share/nginx/html
Creating Unity snapshots

Dell Unity CSI drivers support snapshots of volumes. Create snapshots as a means of backing up either the iSCSI, FC, or NFS volume that you created, and then use the snapshots to create a volume and attach it to the pod.

**Note:** You can take Unity volume snapshots for all the volumes that are created using FC, iSCSI, or NFS protocols. The following example shows snapshots that were taken using Unity snapshot volume class `unity-snap` and the iSCSI volume.

1. **Verify that the volume snapshot class is created:**
   ```
   [core@csah unity]$ oc get volumesnapshotclass
   NAME      DRIVER                  DELETIONPOLICY   AGE
   unity-snap csi-unity.dellemc.com Delete 2m53s
   ```

2. **Add content to the iSCSI pod that is mounted with the iSCSI volume that you created previously. Perform this step to validate successful volumes created using snapshots with the same content:**
   ```
   [core@csah unity]$ oc exec -it dynamic-iscsi-unity-pod /bin/bash
   kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
   root@dynamic-iscsi-unity-pod:/# touch /usr/share/nginx/html/sample
   root@dynamic-iscsi-unity-pod:/# echo 'powermax backup' > /usr/share/nginx/html/sample
   root@dynamic-iscsi-unity-pod:/# cat /usr/share/nginx/html/sample
   unity backup
   ```

3. **Create a snapshot (see this sample file in GitHub) using the YAML file:**
   ```
   [core@csah unity]$ oc create -f <YAML file>
   ```

4. **Verify that the volume snapshots are created:**
   ```
   [core@csah unity]$ oc get volumesnapshot
   NAME     READYTOUSE   SOURCEPVC        SOURCESNAPSHOTCONTENT   RESTORESIZE   SNAPSHOTCLASS    SNAPSHOTCONTENT     CREATIONTIME   AGE
   unity-snapshot true dynamic-iscsi-unity-pvc snapcontent-ccf83239-03bf-4207-80e9-41d02bd5e77d 7Gi 104s
   ```

5. **Create a volume using the snapshot (see this sample file in GitHub), and then verify that the restored volume is created:**

```
Create a pod and attach the volume you created in the preceding step. Verify that the pod is created, that the volume is attached, and that the content you added in step 2 exists (see this sample file in GitHub):

```
[core@csah unity]$ oc get pods restore-pod
```

```
NAME         READY   STATUS    RESTARTS AGE
restore-pod   1/1     Running   0        9m58s
```

```
[core@csah unity]$ oc exec -it restore-pod -- cat /usr/share/nginx/html/sample
```

```
unity backup
```

**PowerMax storage**

Dell EMC PowerMax delivers high levels of performance and efficiency with an integrated machine learning engine. For more information, see Dell EMC PowerMax.

Prerequisites include:

- The Dell EMC PowerMax storage system is configured properly.
- Storage pools have been created along with FC ports, and iSCSI interfaces are configured.
- Dell EMC Unisphere version 9.1 or later is installed to enable use of CSI drivers.

To provision Dell EMC PowerMax storage:

1. Create the namespace:
   
   ```
   [core@csah ~]$ oc new-project powermax
   ```

2. Create the secret to include the username and password for PowerMax by following the steps in this sample file in GitHub.

   **Note:** Specify the secret name as powermax-creds and ensure that the username and password are in base64 format, as shown in the sample file.

3. Apply the secret:
   
   ```
   [core@csah pmax]$ oc create -f <secret YAML file>
   ```
**ISCSI PowerMax Setup**

*Note:* PowerMax supports either the FC or iSCSI protocol using Dell CSI drivers. However, it does not support FC and iSCSI storage provisioning simultaneously. If a CSI driver for PowerMax is already installed, delete it from the OpenShift console: select **Installed Operators > CSI Operator > CSI Driver > PowerMax Instance** and click **Delete.**

Create the iSCSI PowerMax driver file by following the steps in this sample file in GitHub.

1. Create the PowerMax iSCSI setup using the driver:
   
   ```bash
   [core@csah powermax]$ oc create -f <secret YAML file>
   ```

2. Ensure that the PowerMax pods are running in the `powermax` namespace:
   
   ```bash
   [core@csah powermax]$ oc get pods -n powermax -o wide
   ```
   *Note:* Two powermax-controller pods and one powermax-node pod are running for every compute node that is running.

3. Validate that the iSCSI storage class for PowerMax is created:
   
   ```bash
   [core@csah pmax]$ oc get sc
   NAME                      PROVISIONER                AGE
   powermax-iscsi       csi-powermax.dellemc.com   67s
   ```

**Create dynamic iSCSI volumes**

4. Create a YAML file with which to create a PVC by following the steps in this sample file in GitHub.

5. Use the YAML file that you created in the preceding step to create a PVC:
   
   ```bash
   [core@csah ~]$ oc create -f <YAML file>
   ```
   *Note:* Ensure that the storage class name designates an FC configuration. This guide uses `powermax-iscsi` as the iSCSI storage class name.

6. Verify that the volume is created:
   
   ```bash
   [core@csah powermax]$ oc get pvc
   NAME                  STATUS   VOLUME
   dynamic-iscsi-powermax-pvc Bound   pmax-21balb251f   7Gi
   powermax-iscsi        RWO      powermax-iscsi   47s
   ```

**Attach pod to iscsi Volume**

1. Create a YAML file with which to create a pod, and then mount a volume using the PVC you created in step 2 of **Create dynamic iSCSI volumes.** For guidance, see this sample file in GitHub.

2. Create a pod by running:
   
   ```bash
   [core@csah powermax] $ oc create -f <YAML file>
   ```

3. Verify that the pod is created and the PowerMax volume is mounted:
core@csah powermax]$ oc get pods dynamic-iscsi-powermax-pod -0 wide
NAME                             READY STATUS    RESTARTS AGE
IP NODE NOMINATED NODE
READINESS GATES
dynamic-iscsi-powermax-pod 1/1 Running 0 66s
10.129.2.117 compute-3.example.com <none> <none>
core@csah powermax]$ oc exec -it dynamic-iscsi-powermax-pod --

Filesystem Size Used Avail Use% Mounted on
/dev/mapper/mpathb 5.9G 24M 5.6G 1% /usr/share/nginx/html

Configure FC PowerMax

1. Create a PowerMax FC driver file by following the steps in this sample file in GitHub.

   **Note:** Obtain required information such as the Unisphere management IP address and specify additional parameters for the storage class as needed. PowerMax supports either the FC or iSCSI protocol using Dell CSI drivers. It does not support both FC and iSCSI storage provisioning simultaneously.

2. To create the PowerMax FC setup using the driver file, run:

   [core@csah powermax]$ oc create -f <secret YAML file>

3. Ensure that the PowerMax pods are running in the powermax namespace:

   [core@csah powermax]$ oc get pods -n powermax -o wide

   **Note:** Two powermax-controller pods and one powermax-node pod are running for every compute node that is running.

4. Verify that the PowerMax storage class is created:

   [core@csah powermax]$ oc get sc

   **Create dynamic FC volumes**

   It is not necessary to create a volume manually when you use a dynamic volume. The PV is created automatically and is attached to the PVC through a YAML file.

1. Create a YAML file to create PVC by following the steps in this sample file in GitHub.

2. Use the YAML file that you created in step 1 to create the dynamic PV:

   [core@csah powermax] $ oc create -f <YAML file>

   **Note:** Ensure that the storage class name designates an FC configuration. This guide uses powermax-bronze as the FC storage class name.
3. Verify that the volume is created:

```
[c当地@csah powermax]$ oc get pvc
NAME       STATUS    VOLUME                          CAPACITY
ACCESS MODES STORAGECLASS    AGE
dynamic-fc-powermax-pvc   Bound    pmax-0ffe2c3c3a1   7Gi
RWO                  powermax-bronze  10s
```

### Attach a pod to the FC Volume

1. Create a YAML file with which to create a pod, and then mount a volume using the PVC you created in step 2 of *Create dynamic FC volumes*. For guidance, see this [sample file in GitHub](#).

2. Create a pod:

```
[c当地@csah powermax] $ oc create -f <YAML file>
```

3. Verify that the pod is created and the PowerMax volume is mounted:

```
[c当地@csah powermax]$ oc get pods dynamic-fc-powermax-pod -o wide
NAME                      READY   STATUS    RESTARTS   AGE   IP
NO DE                    NOMINATED NODE   READINESS GATES
dynamic-fc-powermax-pod   1/1     Running 0          2d
10.131.0.208   compute-2.example.com   <none>           <none>
```

```
[c当地@csah powermax]$ oc exec -it dynamic-fc-powermax-pod -- df -h /usr/share/nginx/html
Filesystem          Size  Used Avail Use% Mounted on
/dev/mapper/mpathc  5.9G   24M  5.6G   1% /usr/share/nginx/html
```

---

**PowerMax snapshots**

PowerMax CSI drivers support snapshots of volumes. Create snapshots to back up either the iSCSI or the FC volume you created. Then, use the snapshots to create a volume and attach to the pod.

---

**Note:** The following example shows snapshots that were taken using the PowerMax FC volumes and pod that you created. See *Create dynamic FC volumes*. The name of the volume snapshot is provided in the FC driver file. For guidance, see this [sample FC driver](#) in GitHub.

4. Verify that the volume snapshot class is created:

```
[c当地@csah powermax]$ oc get volumesnapshotclass
NAME                 DRIVER
DELETIONPOLICY   AGE
powermax-snapclass   csi-powermax.dellemc.com Delete 3d14h
```

5. Add content to the FC pod that you created previously to validate successful volumes that are created using snapshots with the same content:

```
[c当地@csah powermax]$ oc exec -it dynamic-fc-powermax-pod /bin/bash
```
7. Create a snapshot using a YAML file (see this sample file in GitHub):

   [core@csah powermax]$ oc create -f <YAML file>

8. Verify that the volume snapshots are created:

   [core@csah powermax]$ oc get volumesnapshot
   NAME          READYTOUSE   SOURCEPVC
   SOURCESNAPSHOTCONTENT   RESTORESIZE   SNAPSHOTCLASS
   CREATIONTIME   AGE
   powermax-snapshot   true         dynamic-fc-powermax-pvc
   0 powermax-snapclass snapcontent-fb2c314f-0b09-4d89-885d-49b8dd9360a0 14s 17s

9. Create a volume using the snapshot (see this sample file in GitHub). Verify that the restore volume is created:

   [core@csah powermax]$ oc create -f restore.yaml persistentvolumeclaim/powermax-restore created
   [core@csah powermax]$ oc get pvc powermax-restore
   NAME               STATUS   VOLUME            CAPACITY
   ACCESS MODES   STORAGECLASS      AGE
   powermax-restore Bound pmax-c4e2f6340a 7Gi RWO
   powermax-bronze 31s

10. Create a pod and attach the volume that you created in the preceding step. Verify that the pod is created, the volume is attached, and the content you added in step 2 exists:

    [core@csah powermax]$ oc get pods restore-pod --o wide
    NAME          READY   STATUS    RESTARTS   AGE
    NODE                    NOMINATED NODE   READINESS GATES
    restore-pod   1/1     Running   0          43s
    10.131.0.144   compute-2.example.com   <none>
    <none>
    [core@csah powermax]$ oc exec --it restore-pod -- df -h
    /usr/share/nginx/html
    Filesystem         Size  Used Avail Use% Mounted on
    /dev/mapper/mpathf  5.9G   24M  5.6G   1%
PowerScale storage

Note: The Dell EMC Isilon storage array has been renamed to PowerScale.

The PowerScale storage array delivers high levels of performance and efficiency with an integrated machine learning engine. For more information, see Dell EMC PowerScale Family.

Prerequisites include:

- Dell CSI Operator is installed. See Installing the CSI Operator.
- The PowerScale storage system is configured correctly.
- Storage pools have been created along with the NFS setup as necessary

The following steps assume that there is a running Dell EMC PowerScale system:

1. Create the namespace:

   ```
   [core@csah ~]$ oc new-project Isilon
   ```

2. Create a YAML file for the secret key (see this sample file in GitHub):

   ```
   Note: The secret name must be isilon-creds.
   ```

   ```
   [core@csah ~]$ oc create -f isilon-creds.yaml
   ```

3. Create a CSI PowerScale driver YAML file (see this sample file in GitHub):

   ```
   [core@csah ~]$ oc create -f <YAML file>
   ```

4. Validate that the PowerScale pods are running:

   ```
   [core@csah isilon]$ oc get pods -o wide
   NAME                     READY   STATUS              RESTARTS  AGE       IP                NODE
   NOMINATED NODE           READY   STATUS              RESTARTS  AGE       IP                NODE
   NOMINATED NODE           READY   STATUS              RESTARTS  AGE       IP                NODE
   isilon-controller-68bdf67d46-9f9x5  5/5   Running             0       19s       10.129.2.161 compute-3.example.com <none>
   <none>
   isilon-controller-68bdf67d46-nw4v2  5/5   Running             0       19s       10.131.0.190 compute-2.example.com <none>
   <none>
   isilon-node-2vhvn         2/2    Running             0       19s       10.131.0.190 compute-2.example.com <none>
   <none>
   isilon-node-k4rs5         2/2    Running             0       19s       192.168.46.24  compute-1.example.com <none>
   <none>
   isilon-node-k4rs5         2/2    Running             0       19s       192.168.46.27  compute-3.example.com <none>
   <none>
   ```
5. Verify that the storage class is created for using PowerScale storage:

```
[core@csah isilon]$ oc get sc
NAME           PROVISIONER               RECLAIMPOLICY
               VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE
isilon-nfs    csi-isilon.dellemc.com   Delete    WaitForFirstConsumer   true                   32s
```

Provision the image registry storage

To perform application builds, configure the image registry for the OpenShift cluster.

To provision storage for the image registry, create a PVC and assign the storage to the image registry configuration. Run the commands as user `core` on the CSAH node unless otherwise specified.

1. Create a YAML file with which to create a PVC (see this sample file in GitHub).

   **Note:** Ensure that the namespace is `openshift-image-registry`. Any other namespace causes an error. The recommended size is greater than 100 G. Assign a volume that supports `ReadWriteMany` access mode.

2. Use the YAML file that you created in step 1 to create the PVC:

   ```
   [core@csah ~]$ oc create -f dynfsreg.yaml
   ```

3. Edit the registry configuration to use the PV:

   ```
   [core@csah ~]$ oc edit configs.imageregistry.operator.openshift.io
   ```

4. In the YAML output, under `spec`, do the following:

   a. Change the value for `managementState` from the Removed default value to Managed.

   b. Add the PVC name as the `storage` key:

      ```
      spec:
          managementState: Managed
          storage:
              pvc:
                  claim: isilon-nfs-image-registry
      ```

5. Type `:wq` to save the changes.

   **Note:** For operators such as `image-registry`, `apiserver`, and so on, the PROGRESSING column displays the True state for a few minutes before the AVAILABLE column displays the True state.
Validate the image registry

1. Verify that the AVAILABLE column displays True for all the cluster operators:
   
   ```
   oc get co
   ```

   **Note:** While the image-registry cluster operator status is being verified, the status of other cluster operators such as operator-lifecycle-manager and kube-apiserver might change. We recommend that you check all cluster operators before continuing.

2. Ensure that the image registry pods are all in the Running state, as shown in the following output:
   
   ```
   [core@csah ~]$ oc get pods -n openshift-image-registry
   NAME                                                      READY   STATUS      RESTARTS   AGE
   cluster-image-registry-operator-6895dd787f-knqcp           1/1      Running     0          19d
   image-pruner-1609804800-jk6g2                              0/1      Completed   0          21h
   image-registry-77bdfcb58b-jjgp2                            1/1      Running     0          9d
   node-ca-526vx                                             1/1      Running     0          19d
   node-ca-5hsks                                             1/1      Running     0          19d
   node-ca-7tjxb                                             1/1      Running     0          19d
   node-ca-99zx7                                             1/1      Running     0          19d
   node-ca-9gjxx                                             1/1      Running     0          19d
   node-ca-bgw6k                                             1/1      Running     0          19d
   ```

3. Verify that the registry storage ClaimName that is used for the image registry pod in the preceding output matches the PVC name:
   
   ```
   [core@csah ~]$ oc describe pod image-registry-77bdfcb58b-jjgp2 -n openshift-image-registry | grep -i volumes
   Volumes:
   registry-storage:
   Type: PersistentVolumeClaim (a reference to PersistentVolumeClaim in the same namespace)
   ClaimName: isilon-nfs-image-registry
   ReadOnly: false
   ```

4. As user core in the CSAH node, connect to any control-plane or compute node in the OpenShift Container Platform cluster:
   
   ```
   [core@csah ~]$ oc debug nodes/etcd-0.example.com
   Starting pod/etcd-0examplecom-debug ...
   ```
To use host binaries, run `chroot /host`
Pod IP: 192.168.46.21
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host

5. Log in as kubeadmin:
sh-4.4# oc login -u kubeadm -p xxxxx-xxxxxx-xxxxxx-xxxxxx-xxxxx
Login successful.
You have access to 53 projects, the list has been
suppressed. You can list all projects with 'oc projects'
Using project "default".

6. Test the connection to the image registry service listening in port 5000:
sh-4.4# podman login -u kubeadmin -p $(oc whoami -t) image-registry.openshift-image-registry.svc:5000
Login Succeeded!

Create dynamic NFS volumes
1. Create a YAML file with which to create a PVC (see the sample file in GitHub):
   [core@csah isilon]$ oc create -f dynfspvc.yaml
2. Ensure that the PV and the PVC for Isilon are created appropriately:
   [core@csah isilon]$ oc get pvc
   NAME                     STATUS   VOLUME     CAPACITY ACCESS MODES   STORAGECLASS   AGE
   dynamic-nfs-isilon-pvc   Pending  /ifs/csipscale     12s
dynamic-nfs-isilon-pvc   Pending  /ifs/csipscale     12s
   isilon-nfs               Pending  /ifs/csipscale     12s
   Note: The volume is in a pending state. Create a pod and attach the PVC to change the
   status to BOUND.

Attach the volume to the pod
1. Create a YAML file for the pod (see this sample file in GitHub):
2. Create the pod by running the following command:
   [core@csah ~]$ oc create -f dynfspod.yaml
3. Validate that the storage is attached to the pod:
   [core@csah isilon]$ oc exec -it dynamic-nfs-isilon-pod -- df
   /usr/share/nginx/html
   Filesystem Size Used Avail Use% Mounted on
   192.168.182.85:/ifs/csipscale-d68e2b6415 4.0G 0 4.0G 0%

PowerScale snapshots
1. Verify that the PowerScale volume snapshot is created:
2. Create a file in the volume attached to the pod that you created in Attach the volume to the pod.

3. Create a snapshot of the volume and validate snapshot creation (see this sample file in GitHub):

4. Create a PVC by using the snapshot (see this sample file in GitHub).

5. Create a pod and attach the PVC (see this sample file in GitHub). To verify that the pod is created with the PVC you created from the snapshot, check the contents of the mountpoint:
PowerStore storage

The Dell EMC PowerStore storage system delivers high levels of performance and efficiency with an integrated machine learning (ML) engine. For more information, see Dell EMC PowerStore.

Prerequisites include:

- The PowerStore storage system is configured correctly.
- FC, iSCSI, and NFS are configured in the PowerStore system.

To provision PowerStore storage:

1. Create the namespace by running:
   ```
   [core@csah ~]$ oc new-project powerstore
   ```

2. Create the secret to include the username and password for PowerStore by following the steps in this sample file in GitHub.
   
   **Note:** Specify the secret name as `powerstore-creds` and ensure that the `username` and `password` are in base64 format, as shown in the sample file. Ensure that the values for `chapuser` and `chapsecret` are specified. Also, ensure that the `chapuser` and `chapsecret` key exists. Set the key to empty if chap authentication is not used (see the sample file for guidance).

3. Apply the secret:
   ```
   [core@csah powerstore]$ oc create -f <secret YAML file>
   ```

PowerStore driver setup

PowerStore supports FC, iSCSI and NFS protocols using Dell CSI drivers. Setting the driver `X-CSI_TRANSPORT_PROTOCOL` to `auto` ensures that FC is used if FC ports are available in the compute node and that iSCSI is used otherwise.

Create the PowerStore driver file by following the steps in this sample file in GitHub.

   **Note:** Modify the IP address and NAS server name appropriately.

1. Create the PowerStore setup using the driver:
   ```
   [core@csah powerstore]$ oc create -f <secret YAML file>
   ```

2. Ensure that the PowerStore pods are running in the `powerstore` namespace:
   ```
   [core@csah powerstore]$ oc get pods -n powerstore -o wide
   ```

   **Note:** There are two `powerstore-controller` pods and one `powerstore-node` pod running for every compute node that is running.

3. Validate that the storage class for PowerStore is created:
Applying the sample file creates two storage classes: the `Powerstore-nfs` storage class is for NFS, while `powerstore-xfs` is for all volumes created using either FC or iSCSI depending on the node setup.

**Create dynamic volumes**

**Note:** The setup used for this cluster has FC ports available, ensuring that only the FC protocol is used for all compute nodes.

1. Create a YAML file with which to create a PVC by following the steps in this [sample file](#) in GitHub.

2. Use the YAML file that you created in the preceding step to create a PVC:

   ```
   [core@csah ~]$ oc create -f <YAML file>
   ```

   **Note:** Ensure that the storage class name is modified as needed. This guide uses `powerstore-xfs` as the storage class name.

3. Verify that the volume is created:

   ```
   [core@csah powerstore]$ oc get pvc
   NAME       STATUS     VOLUME   CAPACITY   ACCESS MODES
   STORAGECLASS AGE
   fc-powerstore-pvc   Pending
   powerstore-xfs   9s
   ```

**Attach a pod to the volume**

To attach a pod to a volume, create a YAML file with which to create a pod, and then mount a volume using the PVC you created in Step 2 of **Create dynamic volumes** (see this [sample file](#) for guidance).

1. Create a pod by running:

   ```
   [core@csah powerstore] $ oc create -f <YAML file>
   ```

2. Verify that the pod is created and the PowerStore volume is mounted:

   ```
   [core@csah powerstore]$ oc get pods -o wide fc-powerstore-pod
   NAME               READY STATUS     RESTARTS AGE   IP NODE                   NOMINATED NODE   READINESS GATES
   fc-powerstore-pod  1/1   Running   0       38s   10.128.2.72 compute-2.example.com <none>           <none>
   ```

   ```
   [core@csah powerstore]$ oc exec -it fc-powerstore-pod -- df -h /usr/share/nginx/html
   Filesystem     Size  Used  Avail Use% Mounted on
   /var/lib/kubelet/plugins/kubernetes.io/csi/pv/csi-pstore-b40e754ac3/globalmount/3ba27897-e19c-4ec7-baf2-a23a1fbb0cb3 4.0G  33M  4.0G   1% /usr/share/nginx/html
   ```
Create **NFS volumes**

1. Create a YAML file with which to create a PVC (see this [sample file](https://github.com) in GitHub for guidance).

2. Use the YAML file that you created to create a PVC:
   ```sh
   [core@csah ~]$ oc create -f <YAML file>
   ```
   **Note**: Ensure that the storage class name is modified as needed. This guide uses `powerstore-nfs` as the storage class name.

3. Verify that the volume is created by running:
   ```sh
   [core@csah powerstore]$ oc get pvc
   NAME                 STATUS    VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS     AGE
   nfs-powerstore-pvc   Pending   powerstore-nfs 9s
   ```

**Attach a pod to the NFS volume**

Create a YAML file with which to create a pod, and then mount a volume using the PVC (see this [sample file](https://github.com) in GitHub for guidance).

1. Create a pod by running:
   ```sh
   [core@csah powerstore] $ oc create -f <YAML file>
   ```

2. Verify that the pod is created and the PowerStore volume is mounted:
   ```sh
   [core@csah powerstore]$ oc get pods nfs-powerstore-pod
   NAME                 READY   STATUS    RESTARTS   AGE
   nfs-powerstore-pod   1/1     Running   0          2m53s
   
   [core@csah-sec powerstore]$ oc exec -it nfs-powerstore-pod -- df -h /usr/share/nginx/html
   Filesystem                Size  Used  Avail Use% Mounted on
   <NFS Server>:/csi-pstore-46d1177e8b 5.5G  1.6G  4.0G  28% /usr/share/nginx/html
   ```

**PowerStore snapshots**

PowerStore CSI drivers support snapshots of volumes. Create snapshots to back up either the iSCSI, FC, or NFS volume that you created, and then use the snapshots to create a volume and attach it to the pod.

1. Verify that the volume snapshot class is created:
   ```sh
   [core@csah powerstore]$ oc get volumesnapshotclass
   NAME                DRIVER                                        DELETIONPOLICY   AGE
   powerstore-snapshot  csi-powerstore.dell EMC.com       Delete     4m52s
   ```

2. Add content to the FC pod that you created previously to validate successful volumes that are created using snapshots with the same content:
3. Create a snapshot using a YAML file (see this sample file in GitHub for guidance):
   [core@csah powerstore]$ oc create -f <YAML file>

4. Verify that the volume snapshots are created:
   [core@csah powerstore]$ oc get volumesnapshot
   NAME                  READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT RESTORESIZE SNAPSHOTCLASS SNAPSHOTCONTENT CREATIONTIME AGE
   powerstore-snapshot   true         nfs-powerstore-pvc 4Gi powerstore-snapshot snapcontent-1672bca7-de9c-4207-9ee4a0df3fedb5 1s 4s

5. Create a volume using the snapshot (see this sample file in GitHub for guidance).
   Verify that the restore volume is created:
   [core@csah powerstore]$ oc create -f restore.yaml persistentvolumeclaim/powerstore-restore created
   [core@csah powerstore]$ oc get pvc
   NAME                  STATUS    VOLUME CAPACITY   ACCESS MODES   STORAGECLASS     AGE
   powerstore-restore    Pending   powerstore-nfs 4s

6. Create a pod and attach the volume that you created in the preceding step to it (see this sample file in GitHub for guidance). Before taking the snapshot, verify that the content you added exists in the new pod by running:
   [core@csah powerstore]$ oc get pods restore-pod
   NAME          READY   STATUS    RESTARTS   AGE
   restore-pod   1/1     Running   0          87s
   [core@csah-powerstore]$ oc exec -it restore-pod -- cat /usr/share/nginx/html/backup
   test backup
OpenShift Container storage

Red Hat provides OpenShift container storage as a method to provision storage to pods using local devices in compute nodes.

Prerequisites include:
- Minimum of three compute nodes in the OpenShift cluster.
- Disk size is the same across all compute nodes.
- No partitions are configured.
- Additional steps for Red Hat Enterprise Linux-based compute nodes, as described in the following section.

Perform these steps in all Red Hat Enterprise Linux-based compute nodes.

1. Ensure that the rhel-7-server-rpms and rhel-7-server-extras-rpms repositories are enabled. To enable Red Hat subscription, see Step 3 substep c in Preparing the CSAH node in this guide:

   [root@compute-3 ~]# subscription-manager repos --enable=rhel-7-server-rpms
   Repository 'rhel-7-server-rpms' is enabled for this system.

   [root@compute-3 ~]# subscription-manager repos --enable=rhel-7-server-extras-rpms
   Repository 'rhel-7-server-extras-rpms' is enabled for this system.

2. Install RPMs:

   [root@compute-3 ~]# yum install -y policycoreutils container-selinux

3. Set selinux permissions:

   [root@compute-3 ~]# setsebool -P container_use_cephfs on

Install OpenShift Container Storage Operator

1. From the OpenShift console, select Operators > OperatorHub and search for OpenShift Container Storage.

2. Ensure the version that is displayed is 4.6, and then click the Install option.

   **Note:** Project openshift-storage is created automatically. Installed operators are listed under Operators > Installed Operators.

3. Verify that the OCS operator pods are running:

   [core@csah ocs]$ oc get pods -n openshift-storage -o wide

   NAME                              READY   STATUS     RESTARTS   AGE          IP                     NODE
   noobaa-operator-55c779bc76-tb4dr   1/1     Running    0          3m34s       10.131.1.232 compute-2.example.com <none>
Chapter 6: Provisioning Storage

Install the Local Storage Operator

1. Log in to the OpenShift console. See Accessing the OpenShift web console for instructions. Select Operators > OperatorHub.

2. Search for Local Storage and select it.

3. Ensure that the version is displayed as 4.6, and then click the Install option.

Note: Project openshift-local-storage is created automatically. Installed operators are listed under Operators -> Installed Operators.

4. Verify that the local storage operator pod is running:

   ```
   [core@csah ocs]$ oc get pods -n openshift-local-storage -owide
   NAME                                     READY   STATUS
   RESTARTS   AGE     IP             NODE
   NOMINATED NODE   READINESS GATES
   local-storage-operator-967d7f899-g12qw   1/1     Running   0
   6m33s   10.128.3.159   compute
   <none>
   ```

Create the OpenShift Container Storage cluster

1. Log in to the OpenShift console. See Accessing the OpenShift web console for instructions. Select Operators > Installed Operators > OpenShift Container Storage.

2. Under Details > Storage Cluster, click Create Instance.

3. Ensure that Project is set to openshift-storage. Under Select Mode, select Internal – Attached Devices, then select the appropriate number of compute nodes (a minimum of three), and click Next.

   A volume set is created by default by adding all available disks under all compute nodes.

4. Specify a name for the volume set. By default, a storage class with a similar name is created.

5. Select the compute nodes, click Advanced, specify a size limit under Disk Size, and click Next.

Note: In our deployment, we used only SSD/NVMe drives by setting the size to 700 G to 900 G.
6. Click the drop-down menu in **Storage Class** and select the storage class that was created automatically in step 4. Click **Create** after the appropriate nodes are displayed under **Nodes**.

7. Verify that the storage class is created and all pods in the openshift-storage and openshift-local-storage projects are either in a **Running** or **Completed** state:

   [core@csah ocs]$ oc get sc | grep <storage class name>

   **Note:** The `ocs-storagecluster-ceph-rbd`, `ocs-storagecluster-ceph-rgw`, `ocs-storagecluster-cephfs` storage classes are created by default.

   [core@csah ocs]$ oc get pods -n openshift-storage
   [core@csah ocs]$ oc get pods -n openshift-local-storage

**Create Cephfs PVC**

The following steps create a PVC by using `ocs-storagecluster-cephfs` and `ocs-storagecluster-ceph-rbd`:

1. Create a YAML file to create a PVC using the `ocs-storagecluster-cephfs` storage class (for guidance, see this [sample file](https://github.com) in GitHub).

   [core@csah ~]$ oc create -f <YAML file>

2. Verify that the volume is created:

   [core@csah ocs]$ oc get pvc -n ocs
   NAME       STATUS   VOLUME                  CAPACITY   ACCESS MODES   STORAGECLASS                AGE
   ocsfspvc   Bound    pvc-44d3a805-0e6b-48b1-a8d6-0ab6c56e1975 10Gi        RWX            ocs-storagecluster-cephfs   29s

3. Create a pod and attach the volume (see this [sample file](https://github.com) in GitHub).

   [core@csah ~]$ oc create -f <YAML file>

4. Validate that the pod is created and the volume is attached:

   [core@csah ocs]$ oc exec -it -n ocs ocsfspod -- df -h /usr/share/nginx/html

   Filesystem Size Used Avail Use% Mounted on
   172.30.64.242:6789,172.30.152.143:6789,172.30.10.82:6789:/volumes/csi/csi-vol-5799b028-4e0f-11eb-b2ab-0a580a8003a8/1021f0b6-5b98-4d33-81e6-cdced668345 10G 0 10G 0% /usr/share/nginx/html
Create Ceph rbd PVC

1. Create a YAML file with which to create a PVC by using the `ocs-storagecluster-ceph-rbd` storage class (see this sample file in GitHub).

   ```bash
   [core@csah ~]$ oc create -f <YAML file>
   ```

2. Verify that the PVC is created successfully:

   ```bash
   [core@csah ocs]$ oc get pvc ocsrbdpvc -n ocs
   NAME        STATUS   VOLUME                                      CAPACITY   ACCESS MODES   STORAGECLASS                  AGE
   ocsrbdpvc   Bound    pvc-7742925d-8ecb-4d49-8b4b-00313d87c85  10Gi       RWO            ocs-storagecluster-ceph-rbd   27s
   ```

3. Create a pod and attach the volume (see this sample file in GitHub).

   ```bash
   [core@csah ~]$ oc create -f <YAML file>
   ```

4. Verify that the pod is created and the volume is attached:

   ```bash
   [core@csah ocs]$ oc exec -it ocsrbdpod -n ocs -- df -h
   /usr/share/nginx/html
   Filesystem      Size  Used Avail Use% Mounted on
   /dev/rbd1       9.8G   37M  9.8G   1% /usr/share/nginx/html
   ```
Chapter 7  Application Deployments

This chapter presents the following topics:

Deploying applications ................................................................. 70
Scaling applications ........................................................................ 72
Deploying applications

Introduction

You can use multiple methods to deploy applications in an OpenShift cluster. This guide provides just some examples. For more information, see Creating applications using the Developer perspective in the OpenShift Container Platform documentation.

Note: To build configurations, ensure that the image registry is configured. For the steps for setting up the image registry using a PowerScale volume, see Provision the image registry storage.

Deploy application images

OpenShift supports application deployment using an image that is stored in an external image registry. Images have the necessary packages and program tools to run the applications by default.

To deploy an application that is already part of an image, complete the following steps. Unless specified otherwise, run all the commands as user core in CSAH.

1. Log in to the OpenShift cluster:
   
   ```
   [core@csah ~]$ oc login -u <user name>
   ```

2. Create a project:

   ```
   [core@csah ~]$ oc new-project <project name>
   ```

3. Create an application:

   ```
   [core@csah ~]$ oc new-app <image-name>
   ```

   This guide uses openshift/hello-openshift for the image name that is being tested.

4. After the image is deployed, identify all the objects that are created as part of the deployment by running the oc get all command.

Deploy S2I

OpenShift supports application deployment by using a source from GitHub and specifying an image. An application example is Source-to-Image (S2I), which is a toolkit and workflow for building reproducible container images from source code. A build configuration is generated for the S2I deployment in a new pod called Build Pod. In the build configuration, configure the triggers that are required to automate the new build process every time a condition meets the specifications that you defined. After the deployment is complete, a new image with injected source code is created automatically.

Follow these steps to deploy an application using a source from GitHub. The source in the sample deployment is at httpd-ex.

1. Log in to the OpenShift cluster:

   ```
   [core@csah ~]$ oc login -u <user name>
   ```

2. Create a project:

   ```
   [core@csah ~]$ oc new-project <project name>
   ```

3. Create an application by using the GitHub source and specifying the image of which the application will be a part:
Chapter 7: Application Deployments

Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6
Enabled by Dell EMC PowerEdge R640 and R740xd Servers; PowerSwitch Networking; PowerMax, PowerScale, PowerStore, and Unity XT Storage
Deployment Guide

[core@csah ~]$ oc new-app centos/httpd-24-centos7~https://github.com/sclorg/httpd-ex.git

Note: The image is centos/httpd-24-centos7. The GitHub source is https://github.com/sclorg/httpd-ex.git. You can obtain build logs by running oc logs -f bc/httpd-ex for this example.

4. After the image is deployed, identify all the objects that were created as part of the deployment:
   oc get all

5. Obtain triggers for this deployment by checking the YAML template of the build configuration:
   [core@csah ~]$ oc get buildconfig httpd-ex -o yaml

To access applications that are deployed within the OpenShift cluster using images or source code from GitHub, use the service IP address that is associated with the deployments. External access to the applications is not available by default.

To enable access to the applications from an external network:

1. Log in to the OpenShift cluster:
   [core@csah ~]$ oc login -u <user name>

2. Switch to the project under which the application is running:
   [core@csah ~]$ oc project sample
   Now using project "sample" on server "https://api.ocp.example.com:6443".

3. Identify the service that is associated with the application.
   Note: Typically, the name of the service is the same as the name of the deployment.

   [core@csah ~]$ oc get svc
   NAME              TYPE        CLUSTER-IP      EXTERNAL-IP
   PORT(S)             AGE
   hello-openshift    ClusterIP   172.30.93.229   <none>
   8080/TCP,8888/TCP   23m

4. Expose the route for service of your application:
   [core@csah yaml]$ oc expose svc/hello-openshift
   route.route.openshift.io/hello-openshift exposed

5. Obtain the routes that were created:
   [core@csah ~]$ oc get routes
   NAME              HOST/PORT
   PATH    SERVICES    PORT    TERMINATION    WILDCARD
   hello-openshift  hello-openshift-
   sample.apps.ocp.example.com  hello-openshift  8080-
tcp
Scaling applications

Applications are designed and created to meet the demands of customers and can be scaled up or down based on business needs.

To scale an application, follow these steps.

**Note:** This example uses hello-openshift.

1. Log in to the OpenShift cluster:
   ```bash
   [core@csah ~]$ oc login -u <username>
   ```

2. Switch to the project under which the application is running:
   ```bash
   [core@csah ~]$ oc project sample
   Now using project "sample" on server "https://api.ocp.example.com:6443".
   ```

3. Identify the deployment configuration that is associated with the application:
   ```bash
   [core@csah ~]$ oc get dc
   NAME              REVISION   DESIRED   CURRENT   TRIGGERED
   BY
   hello-openshift   1           3         3         config,image(hello-openshift:latest)
   ```

4. Increase the DESIRED count to 3:
   ```bash
   [core@csah ~]$ oc scale --replicas=3 dc/hello-openshift
   deploymentconfig.apps.openshift.io/hello-openshift scaled
   ```

   ```bash
   [core@csah ~]$ oc get dc
   NAME              REVISION   DESIRED   CURRENT   TRIGGERED
   BY
   Hello-openshift   1           3         3         config,image(hello-openshift:latest)
   ```

**Note:** OpenShift supports the autoscaling of pods if cluster metrics are installed. Run `oc autoscale dc/hello-openshift -min=1 --min=10 --cpu-percent=80` under Technology Preview. For more information about enabling cluster metrics for pods, see Exposing custom application metrics for autoscaling in the OpenShift Container Platform documentation.
This chapter presents the following topics:

OpenShift monitoring overview ................................................................. 74
Adding storage .................................................................................................. 74
Enabling Grafana dashboards ......................................................................... 75
Viewing alerts .................................................................................................... 76
Viewing cluster metrics .................................................................................... 77
OpenShift monitoring overview

By default, OpenShift Container Platform includes a monitoring cluster operator that is based on the Prometheus open-source project. Multiple pods run in the cluster to monitor the state of the cluster and immediately raise any alerts in the OpenShift web console. Grafana dashboards provide cluster metrics.

For more information, see Understanding the monitoring stack in the OpenShift Container Platform documentation.

Adding storage

By default, alert and metrics data are stored in an empty directory. When you delete the pods, the data is also deleted. We recommend that you save the data in persistent volumes.

Add PV storage to Prometheus and Alert Manager pods:

1. Create a YAML file to create a config map (see this sample file in GitHub):
   ```bash
   [core@csah ~] oc create -f <YAML file>
   ```
2. Edit the config map and specify storage options by using `volumeClaimTemplate` for both Prometheus and Alert Manager pods:

   ```yaml
   kind: ConfigMap
   apiVersion: v1
   data:
     config.yaml: |
     prometheusk8s:
       volumeClaimTemplate:
         metadata:
           name: prometheusdb
         spec:
           storageClassName: powermax-bronze
           resources:
             requests:
               storage: 40Gi
     alertmanagerMain:
       volumeClaimTemplate:
         metadata:
           name: alertmanager
         spec:
           storageClassName: powermax-bronze
           resources:
             requests:
               storage: 40Gi
   ```
metadata:
  creationTimestamp: "2020-12-10T19:06:05Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "4150896"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: 4796aecd-e188-40a6-bfd0-6d069c5d01e5

3. Save the configmap using `:wq`.

**Note:** This step terminates existing Prometheus and Alert Manager pods. New pods with storage added are created and are in the Running state, as shown in the following example.

```
[core@csah pvcs]$ oc get pods -n openshift-monitoring | grep -i -E "alertmanager|prometheus-k8s"
alertmanager-main-0 3/3 Running 0 4m
alertmanager-main-1 3/3 Running 0 4m
alertmanager-main-2 3/3 Running 0 4m
prometheus-k8s-0 7/7 Running 1 3m50s
prometheus-k8s-1 7/7 Running 1 3m50s
```

### Enabling the Grafana dashboards

To view cluster metrics in the OpenShift web console, enable the Grafana dashboards by following these steps. Unless specified otherwise, run all the commands as user `core`.

1. Log in to the CSAH node.
2. Obtain the Grafana route:
   
   ```
   [core@csah ~]$ oc get routes --all-namespaces | grep -i grafana
   openshift-monitoring grafana grafana-openshift-monitoring.apps.ocp.example.lab
   grafana https reencrypt/Redirect None
   
   Open a web browser and paste in the URL (grafana-openshift-monitoring.apps.ocp.example.com from the preceding output example).
   ```
3. Log in as `kubeadmin` or as a user with cluster admin privileges.
   
   A list of available components in the cluster is displayed.
4. Click `etcd`.  

---

**Chapter 8: Monitoring the Cluster**

Dell EMC Ready Stack for Red Hat OpenShift Container Platform 4.6

Enabled by Dell EMC PowerEdge R640 and R740xd Servers; PowerSwitch Networking; PowerMax, PowerScale, PowerStore, and Unity XT Storage Deployment Guide
The dashboard shows the active streams, the number of **etcd** nodes that are up, and other details, as shown in the following figure:

![Sample Grafana dashboard](image)

**Figure 15. Sample Grafana dashboard**

**Viewing alerts**

To view the alerts in the OpenShift web console:

1. Log in to the CSAH node.
2. Obtain the Alert Manager route:
   ```
   [core@csah ~]$ oc get routes --all-namespaces | grep -i alertmanager
   openshift-monitoring    alertmanager-main    alertmanager-main-openshift-monitoring.apps.ocp.example.com
   alertmanager-main      web              reencrypt/Redirect None
   ```
3. Open a web browser and paste in the URL (in the preceding output example, it is `alertmanager-main-openshift-monitoring.apps.ocp.example.com`).
4. Log in as **kubeadmin** or as a **cluster admin** user.

**Note:** To temporarily mute the notification of alerts, see [Silencing Alerts](#) in the OpenShift Container Platform documentation.
Viewing cluster metrics

To view cluster metrics in the OpenShift web console:

1. Log in to the CSAH node
2. Obtain the cluster metrics route:

   ```bash
   [core@csah auth]$ oc get routes --all-namespaces | grep -i prometheus
   openshift-monitoring  prometheus-k8s  prometheus-k8s-prometheus-k8s-openshift-monitoring.apps.ocp.example.com
   prometheus-k8s  web  reencrypt/Redirect  None
   ```
3. Open a web browser and paste in the URL (in the preceding output example, it is prometheus-k8s-openshift-monitoring.apps.ocp.example.com).
4. Log in as kubeadmin or as a Microsoft Active Directory user.
5. From the Execute menu, select one of the available queries and click Execute. A graph for the selected query is displayed.
This chapter presents the following topics:

Overview ................................................................................................................................. 79
Installing the Velero server ..................................................................................................... 79
Overview

Velero provides tools to back up and restore your Kubernetes cluster resources and persistent volumes. This chapter provides information about backing up persistent volumes and restoring a specified volume.

Installing the Velero server

Install the Velero server in the CSAH node that is running Red Hat Enterprise Linux 7.9. Follow these steps:

1. To download Velero from GitHub, run the following command:

   ```
   [core@csah velero]$ wget https://github.com/vmware-tanzu/velero/releases/download/v1.5.2/velero-v1.5.2-linux-amd64.tar.gz
   ```

2. Extract the Velero .gz file and copy the Velero binary file to /usr/bin:

   ```
   [root@csah ~]# tar -czvf /home/core/velero/velero-v1.5.2-linux-amd64.tar.gz
   [root@csah ~]# cd velero-v1.5.2-linux-amd64
   [root@csah velero-v1.5.2-linux-amd64]# pwd
   /home/core/velero/velero-v1.5.2-linux-amd64
   [root@csah velero-v1.5.2-linux-amd64]# cp velero /usr/bin/
   ```

3. Download MinIO from GitHub. For more information about MinIO, see this MinIO Quickstart Guide.

   ```
   [core@csah velero]$ wget https://dl.min.io/server/minio/release/linux-amd64/minio
   ```

4. Update the /home/core/.bash_profile with the following MinIO credentials:

   ```
   export MINIO_ACCESS_KEY=minio
   export MINIO_SECRET_KEY=minio123
   ```

5. Create a bucket using the minio file that you downloaded in step 1:

   ```
   [core@csah minio]$ ./minio server bucket &
   [1] 272822
   [core@csah minio]$ ./minio server bucket &
   [1] 272822
   [core@csah minio]$ Attempting encryption of all config, IAM users and policies on MinIO backend
   Endpoint:  http://192.168.46.20:9000
   AccessKey: minio
   SecretKey: minio123

   Browser Access:
   http://127.0.0.1:9000

   Command-line Access: https://docs.min.io/docs/minio-client-quickstart-guide
   ```
$ mc alias set myminio http://192.168.46.20:9000 minio
miniol23

Object API (Amazon S3 compatible):
  Go:       https://docs.min.io/docs/golang-client-
quickstart-guide
  Java:     https://docs.min.io/docs/java-client-
quickstart-guide
  Python:   https://docs.min.io/docs/python-client-
quickstart-guide
  JavaScript: https://docs.min.io/docs/javascript-client-
quickstart-guide
  .NET:     https://docs.min.io/docs/dotnet-client-
quickstart-guide

6. Download the MinIO client from GitHub by running:

   [core@csah minio]$ wget https://dl.min.io/client/mc/release/linux-amd64/mc

7. Set the firewall permissions for port 9000 to be used by MinIO:

   [root@csah ~]$ firewall-cmd --permanent --add-port=9000/tcp
   success
   [root@csah ~]$ firewall-cmd --reload
   Success

8. Set up object storage to create a backup location for storing OpenShift
   Kubernetes objects:

   [core@csah minio]$ ./mc config host add velero
   http://192.168.46.20:9000 $MINIO_ACCESS_KEY
   $MINIO_SECRET_KEY
   mc: Configuration written to `/home/core/.mc/config.json`. Please update your access credentials.
   mc: Successfully created `/home/core/.mc/share`.
   Added `velero` successfully.

9. Create a bucket:

   [core@csah minio]$ ./mc mb -p velero/velero
   Bucket created successfully `velero/velero`.

10. Create a credentials-velero file containing the following information:

    [core@csah minio]$ cat credentials-velero
    [default]
    aws_access_key_id = minio
    aws_secret_access_key = miniol23
11. Install Velero by passing the following arguments:

velero install \
    --features=EnableCSI \
    --provider aws \
    --plugins velero/velero-plugin-for-aws:v1.0.0,velero/velero-plugin-for-csi:v0.1.0 \
    --bucket velero \
    --secret-file ./credentials \
    --use-volume-snapshots=true \
    --backup-location-config
    region=minio,s3ForcePathStyle="true",s3Url=http://192.168.46.20:9000 \
    --snapshot-location-config region="default"

12. Verify that the Velero installation is successful:

[core@csah minio]$ oc get pods -o wide
NAME         READY STATUS    RESTARTS AGE
IP            NODE                   NOMINATED NODE
READINESS GATES
velero-f65d7d8cb-h8826 1/1   Running   0        40s
10.131.1.193 compute-2.example.com <none>
<none>

Follow these steps:

1. Create a new project in which to run the backup by running:

[core@csah velero]$ oc new-project test-velero

2. Create a PVC. The volume is created with storage class unity-fc (see the steps in this sample file in GitHub).

[core@csah velero]$ oc create -f <YAML file>

3. Create a pod and attach the PVC you created in step 2 (see this sample file in GitHub).

4. Add content to the volume mounted in the pod:

[core@csah velero-v1.5.2-linux-amd64]$ oc exec -it velero-bkpone-pod /bin/bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@velero-bkpone-pod:/# cd /usr/share/nginx/html/
root@velero-bkpone-pod:/usr/share/nginx/html# touch sample
root@velero-bkpone-pod:/usr/share/nginx/html# echo 'before backup' > sample
5. **Create a backup using Velero:**

   ```bash
   [core@csah velero]$ velero backup create csi-velero-backup - -include-namespaces test-velero
   Backup request "csi-velero-backup" submitted successfully. Run 'velero backup describe csi-velero-backup' or 'velero backup logs csi-velero-backup' for more details.
   ```

6. **Verify that the backup is created successfully:**

   ```bash
   [core@csah velero-v1.5.2-linux-amd64]$ velero get backup
   NAME                     STATUS      ERRORS   WARNINGS   CREATED      EXPIRES   STORAGE LOCATION   SELECTOR
   csi-velero-backup        Completed   0        1          2020-12-26 20:24:13 -0500 EST 29d default <none>
   ```

   A snapshot is created when there is a PVC in the namespace that is backed up in the preceding step:

   ```bash
   [core@csah velero]$ oc get volumesnapshot
   NAME                             READYTOUSE   SOURCEPVC   SOURCESNAPSHOTCONTENT   RESTORESIZE   SNAPSHOTCLASS   SNAPSHOTCONTENT   CREATIONTIME   AGE
   velero-velero-bkpone-pvc-q7z9c   true         velero-velero-bkpone-pvc   2Gi           unity-snap   snapshotcontent-f6b88526-e499-4428-9195-24fd4f40521d   2m26s 59s
   [core@csah velero]$ oc get volumesnapshotcontents.snapshot.storage.k8s.io
   NAME                     READYTOUSE   RESTORESIZE   DELETIONPOLICY   DRIVER            VOLUMESNAPSHOTCLASS   VOLUMESNAPSHOT   AGE
   snapshotcontent-f6b88526-e499-4428-9195-24fd4f40521d   true   2147483648    Delete           csi-unity.dellemc.com   unity-snap   velero-velero-bkpone-pvc-q7z9c   9m16s
   ```

   To restore the PVC:

   1. **Delete the pod and PVC that you created in Creating a backup:**

      ```bash
      [core@csah velero-v1.5.2-linux-amd64]$ oc delete -f <YAML file>
      persistentvolumeclaim "velero-bkpone-pvc" deleted
      [core@csah velero-v1.5.2-linux-amd64]$ oc delete -f <YAML file>
      pod "velero-bkpone-pod" deleted
      ```
2. **Using the backup that you created in Creating a backup, restore the deleted PVC and pod:**

   ```bash
   [core@csah velero-v1.5.2-linux-amd64]$ velero create restore --from-backup csi-velero-backup
   Restore request "csi-velero-backup-20201226211645" submitted successfully.
   Run `velero restore describe csi-velero-backup-20201226211645` or `velero restore logs csi-velero-backup-20201226211645` for more details.
   
   **Note:** `csi-velero-backup` is the name of the backup that is created in Step 5 in Creating a backup.

3. **Verify that the PVC and pod are restored by using the backed-up snapshot content:**

   ```bash
   [core@csah velero-v1.5.2-linux-amd64]$ oc get pvc
   NAME                STATUS   VOLUME                CAPACITY
   ACCESS MODES   STORAGECLASS   AGE
   velero-bkpone-pvc   Bound    csiunity-83cc00c497   2Gi
   RWO            unity-fc       42s
   [core@csah velero-v1.5.2-linux-amd64]$ oc get pods -o wide
   NAME                READY   STATUS    RESTARTS   AGE    IP
   NODE                    NO
   NOMINATED NODE   READINESS GATES
   velero-bkpone-pod   1/1     Running   0          106s
   10.131.0.175   compute-2.example.com   <none>
   <none>
   
   4. **Validate the content in the PVC to ensure that the content that was added before the backups were taken is present:**

   ```bash
   [core@csah velero]$ oc exec -it velero-bkpone-pod -- cat /usr/share/nginx/html/sample
   before backup