Red Hat

OpenShift Container Platform 4.12

Backup and restore

Backing up and restoring your OpenShift Container Platform cluster

Last Updated: 2023-01-26

OpenShift Container Platform 4.12 Backup and restore

Backing up and restoring your OpenShift Container Platform cluster

Legal Notice

Abstract

This document provides instructions for backing up your cluster's data and for recovering from various disaster scenarios.

Table of Contents

CHAPTER 1. BACKUP AND RESTORE

1.1. CONTROL PLANE BACKUP AND RESTORE OPERATIONS

As a cluster administrator, you might need to stop an OpenShift Container Platform cluster for a period and restart it later. Some reasons for restarting a cluster are that you need to perform maintenance on a cluster or want to reduce resource costs. In OpenShift Container Platform, you can perform a graceful shutdown of a cluster so that you can easily restart the cluster later.

You must back up etcd data before shutting down a cluster; etcd is the key-value store for OpenShift Container Platform, which persists the state of all resource objects. An etcd backup plays a crucial role in disaster recovery. In OpenShift Container Platform, you can also replace an unhealthy etcd member.

When you want to get your cluster running again, restart the cluster gracefully.

NOTE

A cluster's certificates expire one year after the installation date. You can shut down a cluster and expect it to restart gracefully while the certificates are still valid. Although the cluster automatically retrieves the expired control plane certificates, you must still approve the certificate signing requests (CSRs) .

You might run into several situations where OpenShift Container Platform does not work as expected, such as:

- You have a cluster that is not functional after the restart because of unexpected conditions, such as node failure, or network connectivity issues.
- You have deleted something critical in the cluster by mistake.
- You have lost the majority of your control plane hosts, leading to etcd quorum loss.

You can always recover from a disaster situation by restoring your cluster to its previous state using the saved etcd snapshots.

1.2. APPLICATION BACKUP AND RESTORE OPERATIONS

As a cluster administrator, you can back up and restore applications running on OpenShift Container Platform by using the OpenShift API for Data Protection (OADP).

OADP backs up and restores Kubernetes resources and internal images, at the granularity of a namespace, by using Velero 1.9. OADP backs up and restores persistent volumes (PVs) by using snapshots or Restic. For details, see OADP features.

1.2.1. OADP requirements

OADP has the following requirements:

- You must be logged in as a user with a **cluster-admin** role.
- You must have object storage for storing backups, such as one of the following storage types:
  - OpenShift Data Foundation

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- S3-compatible object storage

NOTE

If you want to use CSI backup on OCP 4.11 and later, install OADP 1.1.*x*.

OADP 1.0.*x* does not support CSI backup on OCP 4.11 and later. OADP 1.0. *x* includes Velero 1.7.*x* and expects the API group **snapshot.storage.k8s.io/v1beta1**, which is not present on OCP 4.11 and later.

IMPORTANT

The **CloudStorage** API for S3 storage is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

- To back up PVs with snapshots, you must have cloud storage that has a native snapshot API or supports Container Storage Interface (CSI) snapshots, such as the following providers:
  - Amazon Web Services
  - Microsoft Azure
  - Google Cloud Platform
  - CSI snapshot-enabled cloud storage, such as Ceph RBD or Ceph FS

NOTE

If you do not want to back up PVs by using snapshots, you can use Restic, which is installed by the OADP Operator by default.

1.2.2. Backing up and restoring applications

You back up applications by creating a **Backup** custom resource (CR). You can configure the following backup options:

- Backup hooks to run commands before or after the backup operation
- Scheduled backups
- Restic backups

You restore applications by creating a **Restore** CR. You can configure restore hooks to run commands in init containers or in the application container during the restore operation.

CHAPTER 2. SHUTTING DOWN THE CLUSTER GRACEFULLY

This document describes the process to gracefully shut down your cluster. You might need to temporarily shut down your cluster for maintenance reasons, or to save on resource costs.

2.1. PREREQUISITES

- Take an etcd backup prior to shutting down the cluster.

2.2. SHUTTING DOWN THE CLUSTER

You can shut down your cluster in a graceful manner so that it can be restarted at a later date.

NOTE

You can shut down a cluster until a year from the installation date and expect it to restart gracefully. After a year from the installation date, the cluster certificates expire.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have taken an etcd backup.

IMPORTANT

It is important to take an etcd backup before performing this procedure so that your cluster can be restored if you encounter any issues when restarting the cluster.

Procedure

1. If you are shutting the cluster down for an extended period, determine the date on which certificates expire.

```
$ oc -n openshift-kube-apiserver-operator get secret kube-apiserver-to-kubelet-signer -o
jsonpath='{.metadata.annotations.auth\.openshift\.io/certificate-not-after}'
```

Example output

```
2022-08-05T14:37:50Zuser@user:~ $ (1)
```

(1) To ensure that the cluster can restart gracefully, plan to restart it on or before the specified date. As the cluster restarts, the process might require you to manually approve the pending certificate signing requests (CSRs) to recover kubelet certificates.

2. Shut down all of the nodes in the cluster. You can do this from your cloud provider's web console, or run the following loop:

```
$ for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc debug
node/${node} -- chroot /host shutdown -h 1; done (1)
```

1 **-h 1** indicates how long, in minutes, this process lasts before the control-plane nodes are shut down. For large-scale clusters with 10 nodes or more, set to 10 minutes or longer to make sure all the compute nodes have time to shut down first.

Example output

```
Starting pod/ip-10-0-130-169us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:17 UTC, use 'shutdown -c' to cancel.

Removing debug pod ...
Starting pod/ip-10-0-150-116us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:29 UTC, use 'shutdown -c' to cancel.
```

Shutting down the nodes using one of these methods allows pods to terminate gracefully, which reduces the chance for data corruption.

NOTE

Adjust the shut down time to be longer for large-scale clusters:

```
$ for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc
debug node/${node} -- chroot /host shutdown -h 10; done
```

NOTE

It is not necessary to drain control plane nodes of the standard pods that ship with OpenShift Container Platform prior to shutdown.

Cluster administrators are responsible for ensuring a clean restart of their own workloads after the cluster is restarted. If you drained control plane nodes prior to shutdown because of custom workloads, you must mark the control plane nodes as schedulable before the cluster will be functional again after restart.

3. Shut off any cluster dependencies that are no longer needed, such as external storage or an LDAP server. Be sure to consult your vendor's documentation before doing so.

IMPORTANT

If you deployed your cluster on a cloud-provider platform, do not shut down, suspend, or delete the associated cloud resources. If you delete the cloud resources of a suspended virtual machine, OpenShift Container Platform might not restore successfully.

Additional resources

- Restarting the cluster gracefully

CHAPTER 3. RESTARTING THE CLUSTER GRACEFULLY

This document describes the process to restart your cluster after a graceful shutdown.

Even though the cluster is expected to be functional after the restart, the cluster might not recover due to unexpected conditions, for example:

- etcd data corruption during shutdown
- Node failure due to hardware
- Network connectivity issues

If your cluster fails to recover, follow the steps to restore to a previous cluster state .

3.1. PREREQUISITES

- You have gracefully shut down your cluster .

3.2. RESTARTING THE CLUSTER

You can restart your cluster after it has been shut down gracefully.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- This procedure assumes that you gracefully shut down the cluster.

Procedure

1. Power on any cluster dependencies, such as external storage or an LDAP server.
2. Start all cluster machines.
   Use the appropriate method for your cloud environment to start the machines, for example, from your cloud provider's web console.

   Wait approximately 10 minutes before continuing to check the status of control plane nodes.
3. Verify that all control plane nodes are ready.

   ```
   $ oc get nodes -l node-role.kubernetes.io/master
   ```

   The control plane nodes are ready if the status is **Ready**, as shown in the following output:

   ```
   NAME                           STATUS   ROLES    AGE   VERSION
   ip-10-0-168-251.ec2.internal   Ready    master   75m   v1.25.0
   ip-10-0-170-223.ec2.internal   Ready    master   75m   v1.25.0
   ip-10-0-211-16.ec2.internal    Ready    master   75m   v1.25.0
   ```
4. If the control plane nodes are *not* ready, then check whether there are any pending certificate signing requests (CSRs) that must be approved.
   a. Get the list of current CSRs:

```
$ oc get csr
```

b. Review the details of a CSR to verify that it is valid:

```
$ oc describe csr <csr_name> (1)
```

(1) **<csr_name>** is the name of a CSR from the list of current CSRs.

c. Approve each valid CSR:

```
$ oc adm certificate approve <csr_name>
```

5. After the control plane nodes are ready, verify that all worker nodes are ready.

```
$ oc get nodes -l node-role.kubernetes.io/worker
```

The worker nodes are ready if the status is **Ready**, as shown in the following output:

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-179-95.ec2.internal    Ready    worker   64m   v1.25.0
ip-10-0-182-134.ec2.internal   Ready    worker   64m   v1.25.0
ip-10-0-250-100.ec2.internal   Ready    worker   64m   v1.25.0
```

6. If the worker nodes are *not* ready, then check whether there are any pending certificate signing requests (CSRs) that must be approved.

a. Get the list of current CSRs:

```
$ oc get csr
```

b. Review the details of a CSR to verify that it is valid:

```
$ oc describe csr <csr_name> (1)
```

(1) **<csr_name>** is the name of a CSR from the list of current CSRs.

c. Approve each valid CSR:

```
$ oc adm certificate approve <csr_name>
```

7. Verify that the cluster started properly.

a. Check that there are no degraded cluster Operators.

```
$ oc get clusteroperators
```

Check that there are no cluster Operators with the **DEGRADED** condition set to **True**.

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                              4.10.0    True        False         False      59m
```

```
cloud-credential                          4.10.0    True       False        False      85m
cluster-autoscaler                        4.10.0    True       False        False      73m
config-operator                           4.10.0    True       False        False      73m
console                                   4.10.0    True       False        False      62m
csi-snapshot-controller                   4.10.0    True       False        False      66m
dns                                       4.10.0    True       False        False      76m
etcd                                      4.10.0    True       False        False      76m
...
```

b. Check that all nodes are in the **Ready** state:

```
$ oc get nodes
```

Check that the status for all nodes is **Ready**.

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.25.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.25.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.25.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.25.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.25.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.25.0
```

If the cluster did not start properly, you might need to restore your cluster using an etcd backup.

Additional resources

- See Restoring to a previous cluster state for how to use an etcd backup to restore if your cluster failed to recover after restarting.

CHAPTER 4. APPLICATION BACKUP AND RESTORE

4.1. OADP RELEASE NOTES

The release notes for OpenShift API for Data Protection (OADP) describe new features and enhancements, deprecated features, product recommendations, known issues, and resolved issues.

4.1.1. OADP 1.1.1 release notes

The OADP 1.1.1 release notes include product recommendations and descriptions of known issues.

4.1.1.1. Product recommendations

Before you install OADP 1.1.1, it is recommended to either install VolSync 0.5.1 or to upgrade to it.

4.1.1.2. Known issues

This release has the following known issues:

- OADP currently does not support backup and restore of AWS EFS volumes using restic in Velero (**OADP-778**).
- CSI backups might fail due to a Ceph limitation of **VolumeSnapshotContent** snapshots per PVC.
  You can create many snapshots of the same persistent volume claim (PVC) but cannot schedule periodic creation of snapshots:
  - For CephFS, you can create up to 100 snapshots per PVC.
  - For RADOS Block Device (RBD), you can create up to 512 snapshots for each PVC. (**OADP-804**) and (**OADP-975**)
    For more information, see Volume Snapshots.

4.2. OADP FEATURES AND PLUGINS

OpenShift API for Data Protection (OADP) features provide options for backing up and restoring applications.

The default plugins enable Velero to integrate with certain cloud providers and to back up and restore OpenShift Container Platform resources.

4.2.1. OADP features

OpenShift API for Data Protection (OADP) supports the following features:

Backup

You can back up all resources in your cluster or you can filter the resources by type, namespace, or label.
OADP backs up Kubernetes objects and internal images by saving them as an archive file on object storage. OADP backs up persistent volumes (PVs) by creating snapshots with the native cloud snapshot API or with the Container Storage Interface (CSI). For cloud providers that do not support snapshots, OADP backs up resources and PV data with Restic.

Restore

You can restore resources and PVs from a backup. You can restore all objects in a backup or filter the restored objects by namespace, PV, or label.

Schedule

You can schedule backups at specified intervals.

Hooks

You can use hooks to run commands in a container on a pod, for example, **fsfreeze** to freeze a file system. You can configure a hook to run before or after a backup or restore. Restore hooks can run in an init container or in the application container.

4.2.2. OADP plugins

The OpenShift API for Data Protection (OADP) provides default Velero plugins that are integrated with storage providers to support backup and snapshot operations. You can create custom plugins based on the Velero plugins.

OADP also provides plugins for OpenShift Container Platform resource backups, OpenShift Virtualization resource backups, and Container Storage Interface (CSI) snapshots.

Table 4.1. OADP plugins

OADP plugin	Function	Storage location
aws	Backs up and restores Kubernetes objects.	AWS S3
	Backs up and restores volumes with snapshots.	AWS EBS
azure	Backs up and restores Kubernetes objects.	Microsoft Azure Blob storage
	Backs up and restores volumes with snapshots.	Microsoft Azure Managed Disks
gcp	Backs up and restores Kubernetes objects.	Google Cloud Storage
	Backs up and restores volumes with snapshots.	Google Compute Engine Disks
openshift	Backs up and restores OpenShift Container Platform resources. [1]	Object store
kubevirt	Backs up and restores OpenShift Virtualization resources. [2]	Object store
csi	Backs up and restores volumes with CSI snapshots. [3]	Cloud storage that supports CSI snapshots

1. Mandatory.

2. Virtual machine disks are backed up with CSI snapshots or Restic.

3. The **csi** plugin uses the Velero CSI beta snapshot API.

4.2.3. About OADP Velero plugins

You can configure two types of plugins when you install Velero:

- Default cloud provider plugins

- Custom plugins

Both types of plugin are optional, but most users configure at least one cloud provider plugin.

4.2.3.1. Default Velero cloud provider plugins

You can install any of the following default Velero cloud provider plugins when you configure the **oadp_v1alpha1_dpa.yaml** file during deployment:

- **aws** (Amazon Web Services)

- **gcp** (Google Cloud Platform)

- **azure** (Microsoft Azure)

- **openshift** (OpenShift Velero plugin)

- **csi** (Container Storage Interface)

- **kubevirt** (KubeVirt)

You specify the desired default plugins in the **oadp_v1alpha1_dpa.yaml** file during deployment.

Example file

The following **.yaml** file installs the **openshift**, **aws**, **azure**, and **gcp** plugins:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: dpa-sample
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - aws
      - azure
      - gcp
```

4.2.3.2. Custom Velero plugins

You can install a custom Velero plugin by specifying the plugin **image** and **name** when you configure the **oadp_v1alpha1_dpa.yaml** file during deployment.

You specify the desired custom plugins in the **oadp_v1alpha1_dpa.yaml** file during deployment.

Example file

The following **.yaml** file installs the default **openshift**, **azure**, and **gcp** plugins and a custom plugin that has the name **custom-plugin-example** and the image **quay.io/example-repo/custom-velero-plugin**:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: dpa-sample
spec:
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - azure
     - gcp
     customPlugins:
     - name: custom-plugin-example
       image: quay.io/example-repo/custom-velero-plugin
```

4.2.4. OADP support for IBM Power and IBM Z

OpenShift API for Data Protection (OADP) is platform neutral. The information that follows relates only to IBM Power and to IBM Z.

OADP 1.1.0 was tested successfully against OpenShift Container Platform 4.11 for both IBM Power and IBM Z. The sections that follow give testing and support information for OADP 1.1.0 in terms of backup locations for these systems.

4.2.4.1. OADP support for target backup locations using IBM Power

IBM Power running with OpenShift Container Platform 4.11 and OpenShift API for Data Protection (OADP) 1.1.0 was tested successfully against an AWS S3 backup location target. Although the test involved only an AWS S3 target, Red Hat supports running IBM Power with OpenShift Container Platform 4.11 and OADP 1.1.0 against all non-AWS S3 backup location targets as well.

4.2.4.2. OADP testing and support for target backup locations using IBM Z

IBM Z running with OpenShift Container Platform 4.11 and OpenShift API for Data Protection (OADP) 1.1.0 was tested successfully against an AWS S3 backup location target. Although the test involved only an AWS S3 target, Red Hat supports running IBM Z with OpenShift Container Platform 4.11 and OADP 1.1.0 against all non-AWS S3 backup location targets as well.

4.3. INSTALLING AND CONFIGURING OADP

4.3.1. About installing OADP

As a cluster administrator, you install the OpenShift API for Data Protection (OADP) by installing the OADP Operator. The OADP Operator installs Velero 1.9.

NOTE

Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

To back up Kubernetes resources and internal images, you must have object storage as a backup location, such as one of the following storage types:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- Multicloud Object Gateway
- S3-compatible object storage, such as Noobaa or Minio

IMPORTANT

The **CloudStorage** API, which automates the creation of a bucket for object storage, is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

You can back up persistent volumes (PVs) by using snapshots or Restic.

To back up PVs with snapshots, you must have a cloud provider that supports either a native snapshot API or Container Storage Interface (CSI) snapshots, such as one of the following cloud providers:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- CSI snapshot-enabled cloud provider, such as OpenShift Data Foundation

NOTE

If you want to use CSI backup on OCP 4.11 and later, install OADP 1.1.*x*.

OADP 1.0.*x* does not support CSI backup on OCP 4.11 and later. OADP 1.0. *x* includes Velero 1.7.*x* and expects the API group **snapshot.storage.k8s.io/v1beta1**, which is not present on OCP 4.11 and later.

If your cloud provider does not support snapshots or if your storage is NFS, you can back up applications with Restic backups on object storage.

You create a default **Secret** and then you install the Data Protection Application.

4.3.1.1. Configuring NooBaa for disaster recovery on OpenShift Data Foundation

If you use cluster storage for your NooBaa bucket **backupStorageLocation** on OpenShift Data Foundation, configure NooBaa as an external object store.

> **WARNING**
>
> Failure to configure NooBaa as an external object store might lead to backups not being available.

Procedure

- Configure NooBaa as an external object store as described in Adding storage resources for hybrid or Multicloud.

Additional resources

- Overview of backup and snapshot locations in the Velero documentation

4.3.1.2. About OADP update channels

When you install an OADP Operator, you choose an *update channel*. This channel determines which upgrades to the OADP Operator and to Velero you receive. You can switch channels at any time.

There are three update channels:

- The **stable** channel contains the latest minor updates (y-stream updates) and patches (z-stream updates) of OADP ClusterServiceVersion`. As each new release is published, the available **ClusterServiceVersion** of the OADP Operator will be appended with the latest available minor patch.

- The **stable-1.0** channel contains **oadp.v1.0.*z***, the most recent OADP 1.0 **ClusterServiceVersion**.

- The **stable-1.1** channel contains **oadp.v1.1.*z***, the most recent OADP 1.1 **ClusterServiceVersion**.

Which update channel is right for you?

- Choose the **stable** update channel to install the latest stable OADP version and to receive both minor updates and patches. If you choose this channel, you will receive all y-stream and all z-stream updates for version *x.y.z*.

- Choose the **stable-1.*y*** update channel to install OADP 1. *y* and to continue receiving patches for it. If you choose this channel, you will receive all z-stream patches for version 1.*y.z*.

When must you switch update channels?

- If you have OADP 1.*y* installed and you want to receive patches only for that y-stream, you must switch from the **stable** update channel to the **stable-1.*y*** update channel. You will then receive all z-stream patches for version 1.*y.z*.

- If you have OADP 1.0 installed, want to upgrade to OADP 1.1, and then receive patches only for OADP 1.1, you must switch from the **stable-1.0** update channel to the **stable-1.1** update channel. You will then receive all z-stream patches for version 1.1.*z*.
- If you have OADP 1.*y* installed, with *y* greater than 0, and want to switch to OADP 1.0, you must *uninstall* your OADP Operator and then reinstall it using the **stable-1.0** update channel. You will then receive all z-stream patches for version 1.0.*z*.

NOTE

You cannot switch from OADP 1.*y* to OADP 1.0 by switching update channels. You must uninstall the Operator and then reinstall it.

Additional resources

- Cluster service version

4.3.2. Installing and configuring the OpenShift API for Data Protection with Amazon Web Services

You install the OpenShift API for Data Protection (OADP) with Amazon Web Services (AWS) by installing the OADP Operator. The Operator installs Velero 1.9.

NOTE

Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

You configure AWS for Velero, create a default **Secret**, and then install the Data Protection Application.

To install the OADP Operator in a restricted network environment, you must first disable the default OperatorHub sources and mirror the Operator catalog. See Using Operator Lifecycle Manager on restricted networks for details.

4.3.2.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.12 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.9.

Prerequisites

- You must be logged in as a user with **cluster-admin** privileges.

Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field to find the **OADP Operator**.
3. Select the **OADP Operator** and click **Install**.
4. Click **Install** to install the Operator in the **openshift-adp** project.

5. Click **Operators** → **Installed Operators** to verify the installation.

4.3.2.2. Configuring Amazon Web Services

You configure Amazon Web Services (AWS) for the OpenShift API for Data Protection (OADP).

Prerequisites

- You must have the AWS CLI installed.

Procedure

1. Set the **BUCKET** variable:

```
$ BUCKET=<your_bucket>
```

2. Set the **REGION** variable:

```
$ REGION=<your_region>
```

3. Create an AWS S3 bucket:

```
$ aws s3api create-bucket \
    --bucket $BUCKET \
    --region $REGION \
    --create-bucket-configuration LocationConstraint=$REGION 1
```

1 **us-east-1** does not support a **LocationConstraint**. If your region is **us-east-1**, omit **--create-bucket-configuration LocationConstraint=$REGION**.

4. Create an IAM user:

```
$ aws iam create-user --user-name velero 1
```

1 If you want to use Velero to back up multiple clusters with multiple S3 buckets, create a unique user name for each cluster.

5. Create a **velero-policy.json** file:

```
$ cat > velero-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes",
                "ec2:DescribeSnapshots",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:CreateSnapshot",
                "ec2:DeleteSnapshot"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::${BUCKET}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::${BUCKET}"
            ]
        }
    ]
}
EOF
```

6. Attach the policies to give the **velero** user the minimum necessary permissions:

```
$ aws iam put-user-policy \
  --user-name velero \
  --policy-name velero \
  --policy-document file://velero-policy.json
```

7. Create an access key for the **velero** user:

```
$ aws iam create-access-key --user-name velero
```

Example output

```
{
  "AccessKey": {
        "UserName": "velero",
        "Status": "Active",
        "CreateDate": "2017-07-31T22:24:41.576Z",
        "SecretAccessKey": <AWS_SECRET_ACCESS_KEY>,
        "AccessKeyId": <AWS_ACCESS_KEY_ID>
  }
}
```

8. Create a **credentials-velero** file:

   ```
   $ cat << EOF > ./credentials-velero
   [default]
   aws_access_key_id=<AWS_ACCESS_KEY_ID>
   aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>
   EOF
   ```

   You use the **credentials-velero** file to create a **Secret** object for AWS before you install the Data Protection Application.

4.3.2.3. About backup and snapshot locations and their secrets

You specify backup and snapshot locations and their secrets in the **DataProtectionApplication** custom resource (CR).

Backup locations
You specify S3-compatible object storage, such as Multicloud Object Gateway, Noobaa, or Minio, as a backup location.

Velero backs up OpenShift Container Platform resources, Kubernetes objects, and internal images as an archive file on object storage.

Snapshot locations
If you use your cloud provider's native snapshot API to back up persistent volumes, you must specify the cloud provider as the snapshot location.

If you use Container Storage Interface (CSI) snapshots, you do not need to specify a snapshot location because you will create a **VolumeSnapshotClass** CR to register the CSI driver.

If you use Restic, you do not need to specify a snapshot location because Restic backs up the file system on object storage.

Secrets
If the backup and snapshot locations use the same credentials or if you do not require a snapshot location, you create a default **Secret**.

If the backup and snapshot locations use different credentials, you create two secret objects:

- Custom **Secret** for the backup location, which you specify in the **DataProtectionApplication** CR.
- Default **Secret** for the snapshot location, which is not referenced in the **DataProtectionApplication** CR.

IMPORTANT

The Data Protection Application requires a default **Secret**. Otherwise, the installation will fail.

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file.

4.3.2.3.1. Creating a default Secret

You create a default **Secret** if your backup and snapshot locations use the same credentials or if you do not require a snapshot location.

The default name of the **Secret** is **cloud-credentials**.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a default **Secret**. Otherwise, the installation will fail. If the name of the backup location **Secret** is not specified, the default name is used.

If you do not want to use the backup location credentials during the installation, you can create a **Secret** with the default name by using an empty **credentials-velero** file.

Prerequisites

- Your object storage and cloud storage, if any, must use the same credentials.
- You must configure object storage for Velero.
- You must create a **credentials-velero** file for the object storage in the appropriate format.

Procedure

- Create a **Secret** with the default name:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

The **Secret** is referenced in the **spec.backupLocations.credential** block of the **DataProtectionApplication** CR when you install the Data Protection Application.

4.3.2.3.2. Creating profiles for different credentials

If your backup and snapshot locations use different credentials, you create separate profiles in the **credentials-velero** file.

Then, you create a **Secret** object and specify the profiles in the **DataProtectionApplication** custom resource (CR).

Procedure

1. Create a **credentials-velero** file with separate profiles for the backup and snapshot locations, as in the following example:

```
[backupStorage]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>

[volumeSnapshot]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>
```

2. Create a **Secret** object with the **credentials-velero** file:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-
velero 1
```

3. Add the profiles to the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
        config:
          region: us-east-1
          profile: "backupStorage"
        credential:
          key: cloud
          name: cloud-credentials
  snapshotLocations:
    - name: default
      velero:
        provider: aws
        config:
          region: us-west-2
          profile: "volumeSnapshot"
```

4.3.2.4. Configuring the Data Protection Application

You can configure the Data Protection Application by setting Velero resource allocations or enabling self-signed CA certificates.

4.3.2.4.1. Setting Velero CPU and memory resource allocations

You set the CPU and memory resource allocations for the **Velero** pod by editing the **DataProtectionApplication** custom resource (CR) manifest.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the values in the **spec.configuration.velero.podConfig.ResourceAllocations** block of the **DataProtectionApplication** CR manifest, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
```

```
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        nodeSelector: <node selector> 1
        resourceAllocations:
          limits:
            cpu: "1"
            memory: 512Mi
          requests:
            cpu: 500m
            memory: 256Mi
```

1 1 Specify the node selector to be supplied to Velero podSpec

4.3.2.4.2. Enabling self-signed CA certificates

You must enable a self-signed CA certificate for object storage by editing the **DataProtectionApplication** custom resource (CR) manifest to prevent a **certificate signed by unknown authority** error.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the **spec.backupLocations.velero.objectStorage.caCert** parameter and **spec.backupLocations.velero.config** parameters of the **DataProtectionApplication** CR manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
        config:
          insecureSkipTLSVerify: "false" 2
...
```

1. Specify the Base46-encoded CA certificate string.

2. The **insecureSkipTLSVerify** configuration can be set to either **"true"** or **"false"**. If set to **"true"**, SSL/TLS security is disabled. If set to **"false"**, SSL/TLS security is enabled.

4.3.2.5. Installing the Data Protection Application

You install the Data Protection Application (DPA) by creating an instance of the **DataProtectionApplication** API.

Prerequisites

- You must install the OADP Operator.
- You must configure object storage as a backup location.
- If you use snapshots to back up PVs, your cloud provider must support either a native snapshot API or Container Storage Interface (CSI) snapshots.
- If the backup and snapshot locations use the same credentials, you must create a **Secret** with the default name, **cloud-credentials**.
- If the backup and snapshot locations use different credentials, you must create a **Secret** with the default name, **cloud-credentials**, which contains separate profiles for the backup and snapshot location credentials.

NOTE

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file. If there is no default **Secret**, the installation will fail.

Procedure

1. Click **Operators** → **Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.
3. Click **YAML View** and update the parameters of the **DataProtectionApplication** manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - openshift 1
        - aws
    restic:
      enable: true 2
      podConfig:
        nodeSelector: <node selector> 3
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket_name> 4
          prefix: <prefix> 5
        config:
          region: <region>
          profile: "default"
        credential:
          key: cloud
          name: cloud-credentials 6
  snapshotLocations: 7
    - name: default
      velero:
        provider: aws
        config:
          region: <region> 8
          profile: "default"
```

1. The **openshift** plugin is mandatory.
2. Set to **false** if you want to disable the Restic installation. Restic deploys a daemon set, which means that each worker node has **Restic** pods running. You configure Restic for backups by adding **spec.defaultVolumesToRestic: true** to the **Backup** CR.
3. Specify the node selector to be supplied to Restic podSpec.
4. Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix.
5. Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes.
6. Specify the name of the **Secret** object that you created. If you do not specify this value, the default name, **cloud-credentials**, is used. If you specify a custom name, the custom name is used for the backup location.
7. You do not need to specify a snapshot location if you use CSI snapshots or Restic to back up PVs.
8. The snapshot location must be in the same region as the PVs.

4. Click **Create**.
5. Verify the installation by viewing the OADP resources:

```
$ oc get all -n openshift-adp
```

Example output

```
NAME                                                READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8    2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                    1/1     Running   0          94s
pod/restic-m4lts                                    1/1     Running   0          94s
pod/restic-pv4kr                                    1/1     Running   0          95s
pod/velero-588db7f655-n842v                         1/1     Running   0          95s

NAME                                                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc          ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager    1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                              1/1     1            1           96s

NAME                                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47    1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd   1         1         1       96s
replicaset.apps/velero-588db7f655                              1         1         1       96s
```

4.3.2.5.1. Enabling CSI in the DataProtectionApplication CR

You enable the Container Storage Interface (CSI) in the **DataProtectionApplication** custom resource (CR) in order to back up persistent volumes with CSI snapshots.

Prerequisites

- The cloud provider must support CSI snapshots.

Procedure

- Edit the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi 1
    featureFlags:
    - EnableCSI 2
```

1. Add the **csi** default plugin.
2. Add the **EnableCSI** feature flag.

4.3.3. Installing and configuring the OpenShift API for Data Protection with Microsoft Azure

You install the OpenShift API for Data Protection (OADP) with Microsoft Azure by installing the OADP Operator. The Operator installs Velero 1.9.

> **NOTE**
>
> Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

You configure Azure for Velero, create a default **Secret**, and then install the Data Protection Application.

To install the OADP Operator in a restricted network environment, you must first disable the default OperatorHub sources and mirror the Operator catalog. See Using Operator Lifecycle Manager on restricted networks for details.

4.3.3.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.12 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.9.

Prerequisites

- You must be logged in as a user with **cluster-admin** privileges.

Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field to find the **OADP Operator**.
3. Select the **OADP Operator** and click **Install**.
4. Click **Install** to install the Operator in the **openshift-adp** project.
5. Click **Operators** → **Installed Operators** to verify the installation.

4.3.3.2. Configuring Microsoft Azure

You configure a Microsoft Azure for the OpenShift API for Data Protection (OADP).

Prerequisites

- You must have the Azure CLI installed.

Procedure

1. Log in to Azure:

```
$ az login
```

2. Set the **AZURE_RESOURCE_GROUP** variable:

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

3. Create an Azure resource group:

```
$ az group create -n $AZURE_RESOURCE_GROUP --location CentralUS 1
```

   1 Specify your location.

4. Set the **AZURE_STORAGE_ACCOUNT_ID** variable:

```
$ AZURE_STORAGE_ACCOUNT_ID="velero$(uuidgen | cut -d '-' -f5 | tr '[A-Z]' '[a-z]')"
```

5. Create an Azure storage account:

```
$ az storage account create \
    --name $AZURE_STORAGE_ACCOUNT_ID \
    --resource-group $AZURE_RESOURCE_GROUP \
    --sku Standard_GRS \
    --encryption-services blob \
    --https-only true \
    --kind BlobStorage \
    --access-tier Hot
```

6. Set the **BLOB_CONTAINER** variable:

```
$ BLOB_CONTAINER=velero
```

7. Create an Azure Blob storage container:

```
$ az storage container create \
  -n $BLOB_CONTAINER \
  --public-access off \
  --account-name $AZURE_STORAGE_ACCOUNT_ID
```

8. Obtain the storage account access key:

```
$ AZURE_STORAGE_ACCOUNT_ACCESS_KEY=`az storage account keys list \
  --account-name $AZURE_STORAGE_ACCOUNT_ID \
  --query "[?keyName == 'key1'].value" -o tsv`
```

9. Create a custom role that has the minimum required permissions:

```
AZURE_ROLE=Velero
az role definition create --role-definition '{
  "Name": "'$AZURE_ROLE'",
  "Description": "Velero related permissions to perform backups, restores and deletions",
  "Actions": [
     "Microsoft.Compute/disks/read",
     "Microsoft.Compute/disks/write",
     "Microsoft.Compute/disks/endGetAccess/action",
     "Microsoft.Compute/disks/beginGetAccess/action",
     "Microsoft.Compute/snapshots/read",
     "Microsoft.Compute/snapshots/write",
     "Microsoft.Compute/snapshots/delete",
     "Microsoft.Storage/storageAccounts/listkeys/action",
     "Microsoft.Storage/storageAccounts/regeneratekey/action"
  ],
  "AssignableScopes": ["/subscriptions/'$AZURE_SUBSCRIPTION_ID'"]
  }'
```

10. Create a **credentials-velero** file:

```
$ cat << EOF > ./credentials-velero
AZURE_SUBSCRIPTION_ID=${AZURE_SUBSCRIPTION_ID}
AZURE_TENANT_ID=${AZURE_TENANT_ID}
AZURE_CLIENT_ID=${AZURE_CLIENT_ID}
AZURE_CLIENT_SECRET=${AZURE_CLIENT_SECRET}
AZURE_RESOURCE_GROUP=${AZURE_RESOURCE_GROUP}
AZURE_STORAGE_ACCOUNT_ACCESS_KEY=${AZURE_STORAGE_ACCOUNT_ACCES
S_KEY} 1
AZURE_CLOUD_NAME=AzurePublicCloud
EOF
```

1 Mandatory. You cannot back up internal images if the **credentials-velero** file contains only the service principal credentials.

You use the **credentials-velero** file to create a **Secret** object for Azure before you install the Data Protection Application.

4.3.3.3. About backup and snapshot locations and their secrets

You specify backup and snapshot locations and their secrets in the **DataProtectionApplication** custom resource (CR).

Backup locations

You specify S3-compatible object storage, such as Multicloud Object Gateway, Noobaa, or Minio, as a backup location.

Velero backs up OpenShift Container Platform resources, Kubernetes objects, and internal images as an archive file on object storage.

Snapshot locations

If you use your cloud provider's native snapshot API to back up persistent volumes, you must specify the cloud provider as the snapshot location.

If you use Container Storage Interface (CSI) snapshots, you do not need to specify a snapshot location because you will create a **VolumeSnapshotClass** CR to register the CSI driver.

If you use Restic, you do not need to specify a snapshot location because Restic backs up the file system on object storage.

Secrets

If the backup and snapshot locations use the same credentials or if you do not require a snapshot location, you create a default **Secret**.

If the backup and snapshot locations use different credentials, you create two secret objects:

- Custom **Secret** for the backup location, which you specify in the **DataProtectionApplication** CR.
- Default **Secret** for the snapshot location, which is not referenced in the **DataProtectionApplication** CR.

IMPORTANT

The Data Protection Application requires a default **Secret**. Otherwise, the installation will fail.

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file.

4.3.3.3.1. Creating a default Secret

You create a default **Secret** if your backup and snapshot locations use the same credentials or if you do not require a snapshot location.

The default name of the **Secret** is **cloud-credentials-azure**.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a default **Secret**. Otherwise, the installation will fail. If the name of the backup location **Secret** is not specified, the default name is used.

If you do not want to use the backup location credentials during the installation, you can create a **Secret** with the default name by using an empty **credentials-velero** file.

Prerequisites

- Your object storage and cloud storage, if any, must use the same credentials.
- You must configure object storage for Velero.
- You must create a **credentials-velero** file for the object storage in the appropriate format.

Procedure

- Create a **Secret** with the default name:

```
$ oc create secret generic cloud-credentials-azure -n openshift-adp --from-file cloud=credentials-velero
```

The **Secret** is referenced in the **spec.backupLocations.credential** block of the **DataProtectionApplication** CR when you install the Data Protection Application.

4.3.3.3.2. Creating secrets for different credentials

If your backup and snapshot locations use different credentials, you must create two **Secret** objects:

- Backup location **Secret** with a custom name. The custom name is specified in the **spec.backupLocations** block of the **DataProtectionApplication** custom resource (CR).
- Snapshot location **Secret** with the default name, **cloud-credentials-azure**. This **Secret** is not specified in the **DataProtectionApplication** CR.

Procedure

1. Create a **credentials-velero** file for the snapshot location in the appropriate format for your cloud provider.
2. Create a **Secret** for the snapshot location with the default name:

```
$ oc create secret generic cloud-credentials-azure -n openshift-adp --from-file
cloud=credentials-velero
```

3. Create a **credentials-velero** file for the backup location in the appropriate format for your object storage.
4. Create a **Secret** for the backup location with a custom name:

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-
velero
```

5. Add the **Secret** with the custom name to the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        config:
          resourceGroup: <azure_resource_group>
          storageAccount: <azure_storage_account_id>
          subscriptionId: <azure_subscription_id>
          storageAccountKeyEnvVar: AZURE_STORAGE_ACCOUNT_ACCESS_KEY
        credential:
          key: cloud
          name: <custom_secret> (1)
        provider: azure
        default: true
        objectStorage:
          bucket: <bucket_name>
        prefix: <prefix>
    snapshotLocations:
      - velero:
          config:
            resourceGroup: <azure_resource_group>
            subscriptionId: <azure_subscription_id>
            incremental: "true"
          name: default
          provider: azure
```

1. Backup location **Secret** with custom name.

4.3.3.4. Configuring the Data Protection Application

You can configure the Data Protection Application by setting Velero resource allocations or enabling self-signed CA certificates.

4.3.3.4.1. Setting Velero CPU and memory resource allocations

You set the CPU and memory resource allocations for the **Velero** pod by editing the **DataProtectionApplication** custom resource (CR) manifest.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the values in the **spec.configuration.velero.podConfig.ResourceAllocations** block of the **DataProtectionApplication** CR manifest, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        nodeSelector: <node selector> 1
        resourceAllocations:
          limits:
            cpu: "1"
            memory: 512Mi
          requests:
            cpu: 500m
            memory: 256Mi
```

1. Specify the node selector to be supplied to Velero podSpec

4.3.3.4.2. Enabling self-signed CA certificates

You must enable a self-signed CA certificate for object storage by editing the **DataProtectionApplication** custom resource (CR) manifest to prevent a **certificate signed by unknown authority** error.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the **spec.backupLocations.velero.objectStorage.caCert** parameter and **spec.backupLocations.velero.config** parameters of the **DataProtectionApplication** CR manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> (1)
        config:
          insecureSkipTLSVerify: "false" (2)
...
```

(1) Specify the Base46-encoded CA certificate string.

(2) The **insecureSkipTLSVerify** configuration can be set to either **"true"** or **"false"**. If set to **"true"**, SSL/TLS security is disabled. If set to **"false"**, SSL/TLS security is enabled.

4.3.3.5. Installing the Data Protection Application

You install the Data Protection Application (DPA) by creating an instance of the **DataProtectionApplication** API.

Prerequisites

- You must install the OADP Operator.
- You must configure object storage as a backup location.
- If you use snapshots to back up PVs, your cloud provider must support either a native snapshot API or Container Storage Interface (CSI) snapshots.
- If the backup and snapshot locations use the same credentials, you must create a **Secret** with the default name, **cloud-credentials-azure**.

- If the backup and snapshot locations use different credentials, you must create two **Secrets**:
  - **Secret** with a custom name for the backup location. You add this **Secret** to the **DataProtectionApplication** CR.
  - **Secret** with the default name, **cloud-credentials-azure**, for the snapshot location. This **Secret** is not referenced in the **DataProtectionApplication** CR.

NOTE

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file. If there is no default **Secret**, the installation will fail.

Procedure

1. Click **Operators** → **Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.
3. Click **YAML View** and update the parameters of the **DataProtectionApplication** manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - azure
        - openshift (1)
    restic:
      enable: true (2)
      podConfig:
        nodeSelector: <node selector> (3)
  backupLocations:
    - velero:
        config:
          resourceGroup: <azure_resource_group> (4)
          storageAccount: <azure_storage_account_id> (5)
          subscriptionId: <azure_subscription_id> (6)
          storageAccountKeyEnvVar: AZURE_STORAGE_ACCOUNT_ACCESS_KEY
        credential:
          key: cloud
          name: cloud-credentials-azure (7)
        provider: azure
        default: true
        objectStorage:
          bucket: <bucket_name> (8)
          prefix: <prefix> (9)
  snapshotLocations: (10)
    - velero:
```

```
    config:
      resourceGroup: <azure_resource_group>
      subscriptionId: <azure_subscription_id>
      incremental: "true"
    name: default
    provider: azure
```

1. The **openshift** plugin is mandatory.
2. Set to **false** if you want to disable the Restic installation. Restic deploys a daemon set, which means that each worker node has **Restic** pods running. You configure Restic for backups by adding **spec.defaultVolumesToRestic: true** to the **Backup** CR.
3. Specify the node selector to be supplied to Restic podSpec.
4. Specify the Azure resource group.
5. Specify the Azure storage account ID.
6. Specify the Azure subscription ID.
7. If you do not specify this value, the default name, **cloud-credentials-azure**, is used. If you specify a custom name, the custom name is used for the backup location.
8. Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix.
9. Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes.
10. You do not need to specify a snapshot location if you use CSI snapshots or Restic to back up PVs.

4. Click **Create**.
5. Verify the installation by viewing the OADP resources:

```
$ oc get all -n openshift-adp
```

Example output

```
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8    2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                         1/1     Running   0          94s
pod/restic-m4lts                                         1/1     Running   0          94s
pod/restic-pv4kr                                         1/1     Running   0          95s
pod/velero-588db7f655-n842v                              1/1     Running   0          95s

NAME                                                       TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s
```

```
NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3          <none>         96s

NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager   1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                         1/1     1            1           96s

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47   1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd   1         1         1       96s
replicaset.apps/velero-588db7f655                         1         1         1       96s
```

4.3.3.5.1. Enabling CSI in the DataProtectionApplication CR

You enable the Container Storage Interface (CSI) in the **DataProtectionApplication** custom resource (CR) in order to back up persistent volumes with CSI snapshots.

Prerequisites

- The cloud provider must support CSI snapshots.

Procedure

- Edit the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi 1
    featureFlags:
    - EnableCSI 2
```

1 Add the **csi** default plugin.

2 Add the **EnableCSI** feature flag.

4.3.4. Installing and configuring the OpenShift API for Data Protection with Google Cloud Platform

You install the OpenShift API for Data Protection (OADP) with Google Cloud Platform (GCP) by installing the OADP Operator. The Operator installs Velero 1.9.

NOTE

Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

You configure GCP for Velero, create a default **Secret**, and then install the Data Protection Application.

To install the OADP Operator in a restricted network environment, you must first disable the default OperatorHub sources and mirror the Operator catalog. See Using Operator Lifecycle Manager on restricted networks for details.

4.3.4.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.12 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.9.

Prerequisites

- You must be logged in as a user with **cluster-admin** privileges.

Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field to find the **OADP Operator**.
3. Select the **OADP Operator** and click **Install**.
4. Click **Install** to install the Operator in the **openshift-adp** project.
5. Click **Operators** → **Installed Operators** to verify the installation.

4.3.4.2. Configuring Google Cloud Platform

You configure Google Cloud Platform (GCP) for the OpenShift API for Data Protection (OADP).

Prerequisites

- You must have the **gcloud** and **gsutil** CLI tools installed. See the Google cloud documentation for details.

Procedure

1. Log in to GCP:

```
$ gcloud auth login
```

2. Set the **BUCKET** variable:

```
$ BUCKET=<bucket> 1
```

1 Specify your bucket name.

3. Create the storage bucket:

```
$ gsutil mb gs://$BUCKET/
```

4. Set the **PROJECT_ID** variable to your active project:

```
$ PROJECT_ID=$(gcloud config get-value project)
```

5. Create a service account:

```
$ gcloud iam service-accounts create velero \
    --display-name "Velero service account"
```

6. List your service accounts:

```
$ gcloud iam service-accounts list
```

7. Set the **SERVICE_ACCOUNT_EMAIL** variable to match its **email** value:

```
$ SERVICE_ACCOUNT_EMAIL=$(gcloud iam service-accounts list \
    --filter="displayName:Velero service account" \
    --format 'value(email)')
```

8. Attach the policies to give the **velero** user the minimum necessary permissions:

```
$ ROLE_PERMISSIONS=(
    compute.disks.get
    compute.disks.create
    compute.disks.createSnapshot
    compute.snapshots.get
    compute.snapshots.create
    compute.snapshots.useReadOnly
    compute.snapshots.delete
    compute.zones.get
)
```

9. Create the **velero.server** custom role:

```
$ gcloud iam roles create velero.server \
    --project $PROJECT_ID \
    --title "Velero Server" \
    --permissions "$(IFS=","; echo "${ROLE_PERMISSIONS[*]}")"
```

10. Add IAM policy binding to the project:

```
$ gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member serviceAccount:$SERVICE_ACCOUNT_EMAIL \
    --role projects/$PROJECT_ID/roles/velero.server
```

11. Update the IAM service account:

```
$ gsutil iam ch serviceAccount:$SERVICE_ACCOUNT_EMAIL:objectAdmin gs://${BUCKET}
```

12. Save the IAM service account keys to the **credentials-velero** file in the current directory:

    ```
    $ gcloud iam service-accounts keys create credentials-velero \
      --iam-account $SERVICE_ACCOUNT_EMAIL
    ```

    You use the **credentials-velero** file to create a **Secret** object for GCP before you install the Data Protection Application.

4.3.4.3. About backup and snapshot locations and their secrets

You specify backup and snapshot locations and their secrets in the **DataProtectionApplication** custom resource (CR).

Backup locations
You specify S3-compatible object storage, such as Multicloud Object Gateway, Noobaa, or Minio, as a backup location.

Velero backs up OpenShift Container Platform resources, Kubernetes objects, and internal images as an archive file on object storage.

Snapshot locations
If you use your cloud provider’s native snapshot API to back up persistent volumes, you must specify the cloud provider as the snapshot location.

If you use Container Storage Interface (CSI) snapshots, you do not need to specify a snapshot location because you will create a **VolumeSnapshotClass** CR to register the CSI driver.

If you use Restic, you do not need to specify a snapshot location because Restic backs up the file system on object storage.

Secrets
If the backup and snapshot locations use the same credentials or if you do not require a snapshot location, you create a default **Secret**.

If the backup and snapshot locations use different credentials, you create two secret objects:

- Custom **Secret** for the backup location, which you specify in the **DataProtectionApplication** CR.
- Default **Secret** for the snapshot location, which is not referenced in the **DataProtectionApplication** CR.

IMPORTANT

The Data Protection Application requires a default **Secret**. Otherwise, the installation will fail.

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file.

4.3.4.3.1. Creating a default Secret

You create a default **Secret** if your backup and snapshot locations use the same credentials or if you do not require a snapshot location.

The default name of the **Secret** is **cloud-credentials-gcp**.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a default **Secret**. Otherwise, the installation will fail. If the name of the backup location **Secret** is not specified, the default name is used.

If you do not want to use the backup location credentials during the installation, you can create a **Secret** with the default name by using an empty **credentials-velero** file.

Prerequisites

- Your object storage and cloud storage, if any, must use the same credentials.
- You must configure object storage for Velero.
- You must create a **credentials-velero** file for the object storage in the appropriate format.

Procedure

- Create a **Secret** with the default name:

```
$ oc create secret generic cloud-credentials-gcp -n openshift-adp --from-file
cloud=credentials-velero
```

The **Secret** is referenced in the **spec.backupLocations.credential** block of the **DataProtectionApplication** CR when you install the Data Protection Application.

4.3.4.3.2. Creating secrets for different credentials

If your backup and snapshot locations use different credentials, you must create two **Secret** objects:

- Backup location **Secret** with a custom name. The custom name is specified in the **spec.backupLocations** block of the **DataProtectionApplication** custom resource (CR).
- Snapshot location **Secret** with the default name, **cloud-credentials-gcp**. This **Secret** is not specified in the **DataProtectionApplication** CR.

Procedure

1. Create a **credentials-velero** file for the snapshot location in the appropriate format for your cloud provider.
2. Create a **Secret** for the snapshot location with the default name:

```
$ oc create secret generic cloud-credentials-gcp -n openshift-adp --from-file
cloud=credentials-velero
```

3. Create a **credentials-velero** file for the backup location in the appropriate format for your object storage.
4. Create a **Secret** for the backup location with a custom name:

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-
velero
```

5. Add the **Secret** with the custom name to the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
          name: <custom_secret> (1)
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
  snapshotLocations:
    - velero:
        provider: gcp
        default: true
        config:
          project: <project>
          snapshotLocation: us-west1
```

(1) Backup location **Secret** with custom name.

4.3.4.4. Configuring the Data Protection Application

You can configure the Data Protection Application by setting Velero resource allocations or enabling self-signed CA certificates.

4.3.4.4.1. Setting Velero CPU and memory resource allocations

You set the CPU and memory resource allocations for the **Velero** pod by editing the **DataProtectionApplication** custom resource (CR) manifest.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the values in the **spec.configuration.velero.podConfig.ResourceAllocations** block of the **DataProtectionApplication** CR manifest, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
```

```
...
 configuration:
   velero:
     podConfig:
       nodeSelector: <node selector> 1
       resourceAllocations:
         limits:
           cpu: "1"
           memory: 512Mi
         requests:
           cpu: 500m
           memory: 256Mi
```

1 Specify the node selector to be supplied to Velero podSpec

4.3.4.4.2. Enabling self-signed CA certificates

You must enable a self-signed CA certificate for object storage by editing the **DataProtectionApplication** custom resource (CR) manifest to prevent a **certificate signed by unknown authority** error.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the **spec.backupLocations.velero.objectStorage.caCert** parameter and **spec.backupLocations.velero.config** parameters of the **DataProtectionApplication** CR manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
        config:
          insecureSkipTLSVerify: "false" 2
...
```

1 Specify the Base46-encoded CA certificate string.

2 The **insecureSkipTLSVerify** configuration can be set to either **"true"** or **"false"**. If set to **"true"**, SSL/TLS security is disabled. If set to **"false"**, SSL/TLS security is enabled.

4.3.4.5. Installing the Data Protection Application

You install the Data Protection Application (DPA) by creating an instance of the **DataProtectionApplication** API.

Prerequisites

- You must install the OADP Operator.
- You must configure object storage as a backup location.
- If you use snapshots to back up PVs, your cloud provider must support either a native snapshot API or Container Storage Interface (CSI) snapshots.
- If the backup and snapshot locations use the same credentials, you must create a **Secret** with the default name, **cloud-credentials-gcp**.
- If the backup and snapshot locations use different credentials, you must create two **Secrets**:
  - **Secret** with a custom name for the backup location. You add this **Secret** to the **DataProtectionApplication** CR.
  - **Secret** with the default name, **cloud-credentials-gcp**, for the snapshot location. This **Secret** is not referenced in the **DataProtectionApplication** CR.

NOTE

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file. If there is no default **Secret**, the installation will fail.

Procedure

1. Click **Operators** → **Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.
3. Click **YAML View** and update the parameters of the **DataProtectionApplication** manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - gcp
        - openshift 1
    restic:
      enable: true 2
      podConfig:
        nodeSelector: <node selector> 3
  backupLocations:
    - velero:
      provider: gcp
      default: true
      credential:
        key: cloud
        name: cloud-credentials-gcp (4)
      objectStorage:
        bucket: <bucket_name> (5)
        prefix: <prefix> (6)
  snapshotLocations: (7)
    - velero:
      provider: gcp
      default: true
      config:
        project: <project>
        snapshotLocation: us-west1 (8)
```

1. The **openshift** plugin is mandatory.
2. Set to **false** if you want to disable the Restic installation. Restic deploys a daemon set, which means that each worker node has **Restic** pods running. You configure Restic for backups by adding **spec.defaultVolumesToRestic: true** to the **Backup** CR.
3. Specify the node selector to be supplied to Restic podSpec.
4. If you do not specify this value, the default name, **cloud-credentials-gcp**, is used. If you specify a custom name, the custom name is used for the backup location.
5. Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix.
6. Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes.
7. You do not need to specify a snapshot location if you use CSI snapshots or Restic to back up PVs.
8. The snapshot location must be in the same region as the PVs.

4. Click **Create**.

5. Verify the installation by viewing the OADP resources:

```
$ oc get all -n openshift-adp
```

Example output

```
NAME                                             READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8    2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                 1/1     Running   0          94s
pod/restic-m4lts                                 1/1     Running   0          94s
pod/restic-pv4kr                                 1/1     Running   0          95s
pod/velero-588db7f655-n842v                      1/1     Running   0          95s
```

```
NAME                                                       TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic    3         3         3       3            3           <none>         96s

NAME                                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager     1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry    1/1     1            1           96s
deployment.apps/velero                               1/1     1            1           96s

NAME                                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47     1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd    1         1         1       96s
replicaset.apps/velero-588db7f655                               1         1         1       96s
```

4.3.4.5.1. Enabling CSI in the DataProtectionApplication CR

You enable the Container Storage Interface (CSI) in the **DataProtectionApplication** custom resource (CR) in order to back up persistent volumes with CSI snapshots.

Prerequisites

- The cloud provider must support CSI snapshots.

Procedure

- Edit the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) Add the **csi** default plugin.

(2) Add the **EnableCSI** feature flag.

4.3.5. Installing and configuring the OpenShift API for Data Protection with Multicloud Object Gateway

You install the OpenShift API for Data Protection (OADP) with Multicloud Object Gateway (MCG) by installing the OADP Operator. The Operator installs Velero 1.9.

NOTE

Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

You configure Multicloud Object Gateway as a backup location. MCG is a component of OpenShift Data Foundation. You configure MCG as a backup location in the **DataProtectionApplication** custom resource (CR).

IMPORTANT

The **CloudStorage** API, which automates the creation of a bucket for object storage, is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

You create a **Secret** for the backup location and then you install the Data Protection Application.

To install the OADP Operator in a restricted network environment, you must first disable the default OperatorHub sources and mirror the Operator catalog. For details, see Using Operator Lifecycle Manager on restricted networks.

4.3.5.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.12 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.9.

Prerequisites

- You must be logged in as a user with **cluster-admin** privileges.

Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field to find the **OADP Operator**.
3. Select the **OADP Operator** and click **Install**.
4. Click **Install** to install the Operator in the **openshift-adp** project.
5. Click **Operators** → **Installed Operators** to verify the installation.

4.3.5.2. Retrieving Multicloud Object Gateway credentials

You must retrieve the Multicloud Object Gateway (MCG) credentials in order to create a **Secret** custom resource (CR) for the OpenShift API for Data Protection (OADP).

MCG is a component of OpenShift Data Foundation.

Prerequisites

- You must deploy OpenShift Data Foundation by using the appropriate OpenShift Data Foundation deployment guide.

Procedure

1. Obtain the S3 endpoint, **AWS_ACCESS_KEY_ID**, and **AWS_SECRET_ACCESS_KEY** by running the **describe** command on the **NooBaa** custom resource.

2. Create a **credentials-velero** file:

```
$ cat << EOF > ./credentials-velero
[default]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>
EOF
```

   You use the **credentials-velero** file to create a **Secret** object when you install the Data Protection Application.

4.3.5.3. About backup and snapshot locations and their secrets

You specify backup and snapshot locations and their secrets in the **DataProtectionApplication** custom resource (CR).

Backup locations

You specify S3-compatible object storage, such as Multicloud Object Gateway, Noobaa, or Minio, as a backup location.

Velero backs up OpenShift Container Platform resources, Kubernetes objects, and internal images as an archive file on object storage.

Snapshot locations

If you use your cloud provider's native snapshot API to back up persistent volumes, you must specify the cloud provider as the snapshot location.

If you use Container Storage Interface (CSI) snapshots, you do not need to specify a snapshot location because you will create a **VolumeSnapshotClass** CR to register the CSI driver.

If you use Restic, you do not need to specify a snapshot location because Restic backs up the file system on object storage.

Secrets

If the backup and snapshot locations use the same credentials or if you do not require a snapshot location, you create a default **Secret**.

If the backup and snapshot locations use different credentials, you create two secret objects:

- Custom **Secret** for the backup location, which you specify in the **DataProtectionApplication** CR.

- Default **Secret** for the snapshot location, which is not referenced in the **DataProtectionApplication** CR.

IMPORTANT

The Data Protection Application requires a default **Secret**. Otherwise, the installation will fail.

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file.

4.3.5.3.1. Creating a default Secret

You create a default **Secret** if your backup and snapshot locations use the same credentials or if you do not require a snapshot location.

The default name of the **Secret** is **cloud-credentials**.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a default **Secret**. Otherwise, the installation will fail. If the name of the backup location **Secret** is not specified, the default name is used.

If you do not want to use the backup location credentials during the installation, you can create a **Secret** with the default name by using an empty **credentials-velero** file.

Prerequisites

- Your object storage and cloud storage, if any, must use the same credentials.
- You must configure object storage for Velero.
- You must create a **credentials-velero** file for the object storage in the appropriate format.

Procedure

- Create a **Secret** with the default name:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

The **Secret** is referenced in the **spec.backupLocations.credential** block of the **DataProtectionApplication** CR when you install the Data Protection Application.

4.3.5.3.2. Creating secrets for different credentials

If your backup and snapshot locations use different credentials, you must create two **Secret** objects:

- Backup location **Secret** with a custom name. The custom name is specified in the **spec.backupLocations** block of the **DataProtectionApplication** custom resource (CR).
- Snapshot location **Secret** with the default name, **cloud-credentials**. This **Secret** is not specified in the **DataProtectionApplication** CR.

Procedure

1. Create a **credentials-velero** file for the snapshot location in the appropriate format for your cloud provider.

2. Create a **Secret** for the snapshot location with the default name:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

3. Create a **credentials-velero** file for the backup location in the appropriate format for your object storage.

4. Create a **Secret** for the backup location with a custom name:

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-velero
```

5. Add the **Secret** with the custom name to the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        config:
          profile: "default"
          region: minio
          s3Url: <url>
          insecureSkipTLSVerify: "true"
          s3ForcePathStyle: "true"
        provider: aws
        default: true
        credential:
          key: cloud
          name:  <custom_secret> (1)
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
```

(1) Backup location **Secret** with custom name.

4.3.5.4. Configuring the Data Protection Application

You can configure the Data Protection Application by setting Velero resource allocations or enabling self-signed CA certificates.

4.3.5.4.1. Setting Velero CPU and memory resource allocations

You set the CPU and memory resource allocations for the **Velero** pod by editing the **DataProtectionApplication** custom resource (CR) manifest.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the values in the **spec.configuration.velero.podConfig.ResourceAllocations** block of the **DataProtectionApplication** CR manifest, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        nodeSelector: <node selector> (1)
        resourceAllocations:
          limits:
            cpu: "1"
            memory: 512Mi
          requests:
            cpu: 500m
            memory: 256Mi
```

(1) Specify the node selector to be supplied to Velero podSpec

4.3.5.4.2. Enabling self-signed CA certificates

You must enable a self-signed CA certificate for object storage by editing the **DataProtectionApplication** custom resource (CR) manifest to prevent a **certificate signed by unknown authority** error.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the **spec.backupLocations.velero.objectStorage.caCert** parameter and **spec.backupLocations.velero.config** parameters of the **DataProtectionApplication** CR manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
```

```
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> (1)
        config:
          insecureSkipTLSVerify: "false" (2)
...
```

(1) Specify the Base46-encoded CA certificate string.

(2) The **insecureSkipTLSVerify** configuration can be set to either **"true"** or **"false"**. If set to **"true"**, SSL/TLS security is disabled. If set to **"false"**, SSL/TLS security is enabled.

4.3.5.5. Installing the Data Protection Application

You install the Data Protection Application (DPA) by creating an instance of the **DataProtectionApplication** API.

Prerequisites

- You must install the OADP Operator.
- You must configure object storage as a backup location.
- If you use snapshots to back up PVs, your cloud provider must support either a native snapshot API or Container Storage Interface (CSI) snapshots.
- If the backup and snapshot locations use the same credentials, you must create a **Secret** with the default name, **cloud-credentials**.
- If the backup and snapshot locations use different credentials, you must create two **Secrets**:
  - **Secret** with a custom name for the backup location. You add this **Secret** to the **DataProtectionApplication** CR.
  - **Secret** with the default name, **cloud-credentials**, for the snapshot location. This **Secret** is not referenced in the **DataProtectionApplication** CR.

NOTE

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file. If there is no default **Secret**, the installation will fail.

Procedure

1. Click **Operators** → **Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.

3. Click **YAML View** and update the parameters of the **DataProtectionApplication** manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - aws
        - openshift (1)
    restic:
      enable: true (2)
      podConfig:
        nodeSelector: <node selector> (3)
  backupLocations:
    - velero:
        config:
          profile: "default"
          region: minio
          s3Url: <url> (4)
          insecureSkipTLSVerify: "true"
          s3ForcePathStyle: "true"
        provider: aws
        default: true
        credential:
          key: cloud
          name: cloud-credentials (5)
        objectStorage:
          bucket: <bucket_name> (6)
          prefix: <prefix> (7)
```

1. The **openshift** plugin is mandatory.
2. Set to **false** if you want to disable the Restic installation. Restic deploys a daemon set, which means that each worker node has **Restic** pods running. You configure Restic for backups by adding **spec.defaultVolumesToRestic: true** to the **Backup** CR.
3. Specify the node selector to be supplied to Restic podSpec.
4. Specify the URL of the S3 endpoint.
5. If you do not specify this value, the default name, **cloud-credentials**, is used. If you specify a custom name, the custom name is used for the backup location.
6. Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix.
7. Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes.

4. Click **Create**.

5. Verify the installation by viewing the OADP resources:

```
$ oc get all -n openshift-adp
```

Example output

```
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8    2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                         1/1     Running   0          94s
pod/restic-m4lts                                         1/1     Running   0          94s
pod/restic-pv4kr                                         1/1     Running   0          95s
pod/velero-588db7f655-n842v                              1/1     Running   0          95s

NAME                                                       TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager    1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                              1/1     1            1           96s

NAME                                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47    1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd   1         1         1       96s
replicaset.apps/velero-588db7f655                              1         1         1       96s
```

4.3.5.5.1. Enabling CSI in the DataProtectionApplication CR

You enable the Container Storage Interface (CSI) in the **DataProtectionApplication** custom resource (CR) in order to back up persistent volumes with CSI snapshots.

Prerequisites

- The cloud provider must support CSI snapshots.

Procedure

- Edit the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
```

```
    defaultPlugins:
    - openshift
    - csi (1)
  featureFlags:
  - EnableCSI (2)
```

(1) Add the **csi** default plugin.

(2) Add the **EnableCSI** feature flag.

4.3.6. Installing and configuring the OpenShift API for Data Protection with OpenShift Data Foundation

You install the OpenShift API for Data Protection (OADP) with OpenShift Data Foundation by installing the OADP Operator and configuring a backup location and a snapshot location. Then, you install the Data Protection Application.

NOTE

Starting from OADP 1.0.4, all OADP 1.0.*z* versions can only be used as a dependency of the MTC Operator and are not available as a standalone Operator.

You can configure Multicloud Object Gateway or any S3-compatible object storage as a backup location.

IMPORTANT

The **CloudStorage** API, which automates the creation of a bucket for object storage, is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

You create a **Secret** for the backup location and then you install the Data Protection Application.

To install the OADP Operator in a restricted network environment, you must first disable the default OperatorHub sources and mirror the Operator catalog. For details, see Using Operator Lifecycle Manager on restricted networks.

4.3.6.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.12 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.9.

Prerequisites

- You must be logged in as a user with **cluster-admin** privileges.

Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field to find the **OADP Operator**.
3. Select the **OADP Operator** and click **Install**.
4. Click **Install** to install the Operator in the **openshift-adp** project.
5. Click **Operators → Installed Operators** to verify the installation.

4.3.6.2. About backup and snapshot locations and their secrets

You specify backup and snapshot locations and their secrets in the **DataProtectionApplication** custom resource (CR).

Backup locations
You specify S3-compatible object storage, such as Multicloud Object Gateway, Noobaa, or Minio, as a backup location.

Velero backs up OpenShift Container Platform resources, Kubernetes objects, and internal images as an archive file on object storage.

Snapshot locations
If you use your cloud provider's native snapshot API to back up persistent volumes, you must specify the cloud provider as the snapshot location.

If you use Container Storage Interface (CSI) snapshots, you do not need to specify a snapshot location because you will create a **VolumeSnapshotClass** CR to register the CSI driver.

If you use Restic, you do not need to specify a snapshot location because Restic backs up the file system on object storage.

Secrets
If the backup and snapshot locations use the same credentials or if you do not require a snapshot location, you create a default **Secret**.

If the backup and snapshot locations use different credentials, you create two secret objects:

- Custom **Secret** for the backup location, which you specify in the **DataProtectionApplication** CR.
- Default **Secret** for the snapshot location, which is not referenced in the **DataProtectionApplication** CR.

IMPORTANT

The Data Protection Application requires a default **Secret**. Otherwise, the installation will fail.

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file.

4.3.6.2.1. Creating a default Secret

You create a default **Secret** if your backup and snapshot locations use the same credentials or if you do not require a snapshot location.

The default name of the **Secret** is **cloud-credentials**, unless your backup storage provider has a default plugin, such as **aws**, **azure**, or **gcp**. In that case, the default name is specified in the provider-specific OADP installation procedure.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a default **Secret**. Otherwise, the installation will fail. If the name of the backup location **Secret** is not specified, the default name is used.

If you do not want to use the backup location credentials during the installation, you can create a **Secret** with the default name by using an empty **credentials-velero** file.

Prerequisites

- Your object storage and cloud storage, if any, must use the same credentials.
- You must configure object storage for Velero.
- You must create a **credentials-velero** file for the object storage in the appropriate format.

Procedure

- Create a **Secret** with the default name:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

The **Secret** is referenced in the **spec.backupLocations.credential** block of the **DataProtectionApplication** CR when you install the Data Protection Application.

4.3.6.2.2. Creating secrets for different credentials

If your backup and snapshot locations use different credentials, you must create two **Secret** objects:

- Backup location **Secret** with a custom name. The custom name is specified in the **spec.backupLocations** block of the **DataProtectionApplication** custom resource (CR).
- Snapshot location **Secret** with the default name, **cloud-credentials**. This **Secret** is not specified in the **DataProtectionApplication** CR.

Procedure

1. Create a **credentials-velero** file for the snapshot location in the appropriate format for your cloud provider.
2. Create a **Secret** for the snapshot location with the default name:

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

3. Create a **credentials-velero** file for the backup location in the appropriate format for your object storage.

4. Create a **Secret** for the backup location with a custom name:

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-velero
```

5. Add the **Secret** with the custom name to the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        provider: <provider>
        default: true
        credential:
          key: cloud
          name: <custom_secret> (1)
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
```

(1) Backup location **Secret** with custom name.

4.3.6.3. Configuring the Data Protection Application

You can configure the Data Protection Application by setting Velero resource allocations or enabling self-signed CA certificates.

4.3.6.3.1. Setting Velero CPU and memory resource allocations

You set the CPU and memory resource allocations for the **Velero** pod by editing the **DataProtectionApplication** custom resource (CR) manifest.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the values in the **spec.configuration.velero.podConfig.ResourceAllocations** block of the **DataProtectionApplication** CR manifest, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
```

```
spec:
...
  configuration:
    velero:
      podConfig:
        nodeSelector: <node selector> (1)
        resourceAllocations:
          limits:
            cpu: "1"
            memory: 512Mi
          requests:
            cpu: 500m
            memory: 256Mi
```

1. Specify the node selector to be supplied to Velero podSpec

4.3.6.3.2. Enabling self-signed CA certificates

You must enable a self-signed CA certificate for object storage by editing the **DataProtectionApplication** custom resource (CR) manifest to prevent a **certificate signed by unknown authority** error.

Prerequisites

- You must have the OpenShift API for Data Protection (OADP) Operator installed.

Procedure

- Edit the **spec.backupLocations.velero.objectStorage.caCert** parameter and **spec.backupLocations.velero.config** parameters of the **DataProtectionApplication** CR manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> (1)
        config:
          insecureSkipTLSVerify: "false" (2)
...
```

1. Specify the Base46-encoded CA certificate string.
2.

The **insecureSkipTLSVerify** configuration can be set to either **"true"** or **"false"**. If set to **"true"**, SSL/TLS security is disabled. If set to **"false"**, SSL/TLS security is enabled.

4.3.6.4. Installing the Data Protection Application

You install the Data Protection Application (DPA) by creating an instance of the **DataProtectionApplication** API.

Prerequisites

- You must install the OADP Operator.
- You must configure object storage as a backup location.
- If you use snapshots to back up PVs, your cloud provider must support either a native snapshot API or Container Storage Interface (CSI) snapshots.
- If the backup and snapshot locations use the same credentials, you must create a **Secret** with the default name, **cloud-credentials**.
- If the backup and snapshot locations use different credentials, you must create two **Secrets**:
  - **Secret** with a custom name for the backup location. You add this **Secret** to the **DataProtectionApplication** CR.
  - **Secret** with the default name, **cloud-credentials**, for the snapshot location. This **Secret** is not referenced in the **DataProtectionApplication** CR.

NOTE

If you do not want to specify backup or snapshot locations during the installation, you can create a default **Secret** with an empty **credentials-velero** file. If there is no default **Secret**, the installation will fail.

Procedure

1. Click **Operators** → **Installed Operators** and select the OADP Operator.
2. Under **Provided APIs**, click **Create instance** in the **DataProtectionApplication** box.
3. Click **YAML View** and update the parameters of the **DataProtectionApplication** manifest:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - kubevirt 1
        - gcp 2
        - csi 3
      - openshift (4)
    restic:
      enable: true (5)
      podConfig:
        nodeSelector: <node selector> (6)
  backupLocations:
    - velero:
        provider: gcp (7)
        default: true
        credential:
          key: cloud
          name: <default_secret> (8)
        objectStorage:
          bucket: <bucket_name> (9)
          prefix: <prefix> (10)
```

1. Optional: The **kubevirt** plugin is used with OpenShift Virtualization.
2. Specify the default plugin for the backup provider, for example, **gcp**, if appropriate.
3. Specify the **csi** default plugin if you use CSI snapshots to back up PVs. The **csi** plugin uses the Velero CSI beta snapshot APIs. You do not need to configure a snapshot location.
4. The **openshift** plugin is mandatory.
5. Set to **false** if you want to disable the Restic installation. Restic deploys a daemon set, which means that each worker node has **Restic** pods running. You configure Restic for backups by adding **spec.defaultVolumesToRestic: true** to the **Backup** CR.
6. Specify the node selector to be supplied to Restic podSpec
7. Specify the backup provider.
8. If you use a default plugin for the backup provider, you must specify the correct default name for the **Secret**, for example, **cloud-credentials-gcp**. If you specify a custom name, the custom name is used for the backup location. If you do not specify a **Secret** name, the default name is used.
9. Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix.
10. Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes.

4. Click **Create**.

5. Verify the installation by viewing the OADP resources:

```
$ oc get all -n openshift-adp
```

Example output

```
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
```

```
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                         1/1     Running   0          94s
pod/restic-m4lts                                         1/1     Running   0          94s
pod/restic-pv4kr                                         1/1     Running   0          95s
pod/velero-588db7f655-n842v                              1/1     Running   0          95s

NAME                                                  TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc            ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager    1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                              1/1     1            1           96s

NAME                                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47     1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd    1         1         1       96s
replicaset.apps/velero-588db7f655                               1         1         1       96s
```

4.3.6.4.1. Configuring NooBaa for disaster recovery on OpenShift Data Foundation

If you use cluster storage for your NooBaa bucket **backupStorageLocation** on OpenShift Data Foundation, configure NooBaa as an external object store.

WARNING

Failure to configure NooBaa as an external object store might lead to backups not being available.

Procedure

- Configure NooBaa as an external object store as described in Adding storage resources for hybrid or Multicloud.

4.3.6.4.2. Enabling CSI in the DataProtectionApplication CR

You enable the Container Storage Interface (CSI) in the **DataProtectionApplication** custom resource (CR) in order to back up persistent volumes with CSI snapshots.

Prerequisites

- The cloud provider must support CSI snapshots.

Procedure

- Edit the **DataProtectionApplication** CR, as in the following example:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi 1
    featureFlags:
    - EnableCSI 2
```

1 Add the **csi** default plugin.

2 Add the **EnableCSI** feature flag.

4.3.7. Uninstalling the OpenShift API for Data Protection

You uninstall the OpenShift API for Data Protection (OADP) by deleting the OADP Operator. See Deleting Operators from a cluster for details.

4.4. BACKING UP AND RESTORING

4.4.1. Backing up applications

You back up applications by creating a **Backup** custom resource (CR).

The **Backup** CR creates backup files for Kubernetes resources and internal images, on S3 object storage, and snapshots for persistent volumes (PVs), if the cloud provider uses a native snapshot API or the Container Storage Interface (CSI) to create snapshots, such as OpenShift Data Foundation 4. For more information, see CSI volume snapshots.

IMPORTANT

The **CloudStorage** API for S3 storage is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

If your cloud provider has a native snapshot API or supports Container Storage Interface (CSI) snapshots, the **Backup** CR backs up persistent volumes by creating snapshots. For more information, see the Overview of CSI volume snapshots in the OpenShift Container Platform documentation.

If your cloud provider does not support snapshots or if your applications are on NFS data volumes, you can create backups by using Restic.

You can create backup hooks to run commands before or after the backup operation.

You can schedule backups by creating a **Schedule** CR instead of a **Backup** CR.

4.4.1.1. Creating a Backup CR

You back up Kubernetes images, internal images, and persistent volumes (PVs) by creating a **Backup** custom resource (CR).

Prerequisites

- You must install the OpenShift API for Data Protection (OADP) Operator.
- The **DataProtectionApplication** CR must be in a **Ready** state.
- Backup location prerequisites:
  - You must have S3 object storage configured for Velero.
  - You must have a backup location configured in the **DataProtectionApplication** CR.
- Snapshot location prerequisites:
  - Your cloud provider must have a native snapshot API or support Container Storage Interface (CSI) snapshots.
  - For CSI snapshots, you must create a **VolumeSnapshotClass** CR to register the CSI driver.
  - You must have a volume location configured in the **DataProtectionApplication** CR.

Procedure

1. Retrieve the **backupStorageLocations** CRs by entering the following command:

   ```
   $ oc get backupStorageLocations
   ```

   Example output

   ```
   NAME              PHASE       LAST VALIDATED   AGE   DEFAULT
   velero-sample-1   Available   11s              31m
   ```

2. Create a **Backup** CR, as in the following example:

   ```
   apiVersion: velero.io/v1
   kind: Backup
   metadata:
     name: <backup>
     labels:
       velero.io/storage-location: default
     namespace: openshift-adp
   spec:
     hooks: {}
   ```

```
includedNamespaces:
- <namespace> [1]
includedResources: [] [2]
excludedResources: [] [3]
storageLocation: <velero-sample-1> [4]
ttl: 720h0m0s
labelSelector: [5]
- matchLabels:
    app=<label_1>
- matchLabels:
    app=<label_2>
- matchLabels:
    app=<label_3>
orlabelSelectors: [6]
- matchLabels:
    app=<label_1>
- matchLabels:
    app=<label_2>
- matchLabels:
    app=<label_3>
```

[1] Specify an array of namespaces to back up.

[2] Optional: Specify an array of resources to include in the backup. Resources might be shortcuts (for example, 'po' for 'pods') or fully-qualified. If unspecified, all resources are included.

[3] [5] Optional: Specify an array of resources to exclude from the backup. Resources might be shortcuts (for example, 'po' for 'pods') or fully-qualified.

[4] [6] Specify the name of the **backupStorageLocations** CR.

3. Verify that the status of the **Backup** CR is **Completed**:

```
$ oc get backup -n openshift-adp <backup> -o jsonpath='{.status.phase}'
```

4.4.1.2. Backing up persistent volumes with CSI snapshots

You back up persistent volumes with Container Storage Interface (CSI) snapshots by editing the **VolumeSnapshotClass** custom resource (CR) of the cloud storage before you create the **Backup** CR.

Prerequisites

- The cloud provider must support CSI snapshots.
- You must enable CSI in the **DataProtectionApplication** CR.

Procedure

- Add the **metadata.labels.velero.io/csi-volumesnapshot-class: "true"** key-value pair to the **VolumeSnapshotClass** CR:

```
apiVersion: snapshot.storage.k8s.io/v1
```

```
kind: VolumeSnapshotClass
metadata:
  name: <volume_snapshot_class_name>
  labels:
    velero.io/csi-volumesnapshot-class: "true"
driver: <csi_driver>
deletionPolicy: Retain
```

You can now create a **Backup** CR.

4.4.1.3. Backing up applications with Restic

You back up Kubernetes resources, internal images, and persistent volumes with Restic by editing the **Backup** custom resource (CR).

You do not need to specify a snapshot location in the **DataProtectionApplication** CR.

IMPORTANT

Restic does not support backing up **hostPath** volumes. For more information, see additional Rustic limitations.

Prerequisites

- You must install the OpenShift API for Data Protection (OADP) Operator.
- You must not disable the default Restic installation by setting **spec.configuration.restic.enable** to **false** in the **DataProtectionApplication** CR.
- The **DataProtectionApplication** CR must be in a **Ready** state.

Procedure

- Edit the **Backup** CR, as in the following example:

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  labels:
    velero.io/storage-location: default
  namespace: openshift-adp
spec:
  defaultVolumesToRestic: true (1)
...
```

(1) Add **defaultVolumesToRestic: true** to the **spec** block.

4.4.1.4. Using Data Mover for CSI snapshots

IMPORTANT

Data Mover for CSI snapshots is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

The OADP 1.1.0 Data Mover enables customers to back up container storage interface (CSI) volume snapshots to a remote object store. When Data Mover is enabled, you can restore stateful applications from the store if a failure, accidental deletion, or corruption of the cluster occurs. The OADP 1.1.0 Data Mover solution uses the Restic option of VolSync.

NOTE

Data Mover supports backup and restore of CSI volume snapshots only.

Currently, Data Mover does not support Google Cloud Storage (GCS) buckets.

Prerequisites

- You have verified that the **StorageClass** and **VolumeSnapshotClass** custom resources (CRs) support CSI.
- You have verified that only one **volumeSnapshotClass** CR has the annotation **snapshot.storage.kubernetes.io/is-default-class: true**.
- You have verified that only one **storageClass** CR has the annotation **storageclass.kubernetes.io/is-default-class: true**.
- You have included the label **velero.io/csi-volumesnapshot-class: 'true'** in your **VolumeSnapshotClass** CR.
- You have installed the VolSync Operator by using the Operator Lifecycle Manager (OLM).

  NOTE

  The VolSync Operator is required only for use with the Technology Preview Data Mover. The Operator is not required for using OADP production features.

- You have installed the OADP operator by using OLM.

Procedure

1. Configure a Restic secret by creating a **.yaml** file as following:

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-adp
```

```
type: Opaque
stringData:
  RESTIC_PASSWORD: <secure_restic_password>
```

NOTE

By default, the Operator looks for a secret named **dm-credential**. If you are using a different name, you need to specify the name through a Data Protection Application (DPA) CR using **dpa.spec.features.dataMover.credentialName**.

2. Create a DPA CR similar to the following example. The default plugins include CSI.

Example Data Protection Application (DPA) CR

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero-sample
  namespace: openshift-adp
spec:
  features:
    dataMover:
      enable: true
      credentialName: <secret_name> (1)
  backupLocations:
    - velero:
        config:
          profile: default
          region: us-east-1
        credential:
          key: cloud
          name: cloud-credentials
        default: true
        objectStorage:
          bucket: <bucket_name>
          prefix: <bucket_prefix>
        provider: aws
  configuration:
    restic:
      enable: <true_or_false>
    velero:
      defaultPlugins:
        - openshift
        - aws
        - csi
```

(1) Add the Restic secret name from the previous step. If this is not done, the default secret name **dm-credential** is used.

The OADP Operator installs two custom resource definitions (CRDs), **VolumeSnapshotBackup** and **VolumeSnapshotRestore**.

Example VolumeSnapshotBackup CRD

```
apiVersion: datamover.oadp.openshift.io/v1alpha1
kind: VolumeSnapshotBackup
metadata:
  name: <vsb_name>
  namespace: <namespace_name> 1
spec:
  volumeSnapshotContent:
    name: <snapcontent_name>
  protectedNamespace: <adp_namespace>
  resticSecretRef:
    name: <restic_secret_name>
```

1 Specify the namespace where the volume snapshot exists.

Example VolumeSnapshotRestore CRD

```
apiVersion: datamover.oadp.openshift.io/v1alpha1
kind: VolumeSnapshotRestore
metadata:
  name: <vsr_name>
  namespace: <namespace_name> 1
spec:
  protectedNamespace: <protected_ns> 2
  resticSecretRef:
    name: <restic_secret_name>
  volumeSnapshotMoverBackupRef:
    sourcePVCData:
      name: <source_pvc_name>
      size: <source_pvc_size>
    resticrepository: <your_restic_repo>
    volumeSnapshotClassName: <vsclass_name>
```

1 Specify the namespace where the volume snapshot exists.

2 Specify the namespace where the Operator is installed. The default is **openshift-adp**.

3. You can back up a volume snapshot by performing the following steps:

   a. Create a backup CR:

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup_name>
  namespace: <protected_ns> 1
spec:
  includedNamespaces:
  - <app_ns>
  storageLocation: velero-sample-1
```

1 Specify the namespace where the Operator is installed. The default namespace is **openshift-adp**.

b. Wait up to 10 minutes and check whether the **VolumeSnapshotBackup** CR status is **Completed** by entering the following commands:

```
$ oc get vsb -n <app_ns>
```

```
$ oc get vsb <vsb_name> -n <app_ns> -o jsonpath="{.status.phase}"
```

A snapshot is created in the object store was configured in the DPA.

NOTE

If the status of the **VolumeSnapshotBackup** CR becomes **Failed**, refer to the Velero logs for troubleshooting.

4. You can restore a volume snapshot by performing the following steps:

   a. Delete the application namespace and the **volumeSnapshotContent** that was created by the Velero CSI plugin.

   b. Create a **Restore** CR and set **restorePVs** to **true**.

   Example Restore CR

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: <restore_name>
  namespace: <protected_ns>
spec:
  backupName: <previous_backup_name>
  restorePVs: true
```

   c. Wait up to 10 minutes and check whether the **VolumeSnapshotRestore** CR status is **Completed** by entering the following command:

```
$ oc get vsr -n <app_ns>
```

```
$ oc get vsr <vsr_name> -n <app_ns> -o jsonpath="{.status.phase}"
```

   d. Check whether your application data and resources have been restored.

NOTE

If the status of the **VolumeSnapshotRestore** CR becomes 'Failed', refer to the Velero logs for troubleshooting.

Additional resources

- Installing Operators on clusters for administrators
- Installing Operators in namespaces for non-administrators

4.4.1.5. Creating backup hooks

You create backup hooks to run commands in a container in a pod by editing the **Backup** custom resource (CR).

Pre hooks run before the pod is backed up. *Post* hooks run after the backup.

Procedure

- Add a hook to the **spec.hooks** block of the **Backup** CR, as in the following example:

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
        - <namespace> 1
        excludedNamespaces: 2
        - <namespace>
        includedResources: []
        - pods 3
        excludedResources: [] 4
        labelSelector: 5
          matchLabels:
            app: velero
            component: server
        pre: 6
          - exec:
              container: <container> 7
              command:
              - /bin/uname 8
              - -a
              onError: Fail 9
              timeout: 30s 10
        post: 11
...
```

1. Optional: You can specify namespaces to which the hook applies. If this value is not specified, the hook applies to all namespaces.
2. Optional: You can specify namespaces to which the hook does not apply.
3. Currently, pods are the only supported resource that hooks can apply to.
4. Optional: You can specify resources to which the hook does not apply.
5. Optional: This hook only applies to objects matching the label. If this value is not specified, the hook applies to all namespaces.

6. Array of hooks to run before the backup.
7. Optional: If the container is not specified, the command runs in the first container in the pod.
8. This is the entrypoint for the init container being added.
9. Allowed values for error handling are **Fail** and **Continue**. The default is **Fail**.
10. Optional: How long to wait for the commands to run. The default is **30s**.
11. This block defines an array of hooks to run after the backup, with the same parameters as the pre-backup hooks.

4.4.1.6. Scheduling backups

You schedule backups by creating a **Schedule** custom resource (CR) instead of a **Backup** CR.

> **WARNING**
>
> Leave enough time in your backup schedule for a backup to finish before another backup is created.
>
> For example, if a backup of a namespace typically takes 10 minutes, do not schedule backups more frequently than every 15 minutes.

Prerequisites

- You must install the OpenShift API for Data Protection (OADP) Operator.
- The **DataProtectionApplication** CR must be in a **Ready** state.

Procedure

1. Retrieve the **backupStorageLocations** CRs:

```
$ oc get backupStorageLocations
```

Example output

```
NAME              PHASE       LAST VALIDATED   AGE   DEFAULT
velero-sample-1   Available   11s              31m
```

2. Create a **Schedule** CR, as in the following example:

```
$ cat << EOF | oc apply -f -
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: <schedule>
  namespace: openshift-adp
spec:
  schedule: 0 7 * * * (1)
  template:
    hooks: {}
    includedNamespaces:
    - <namespace> (2)
    storageLocation: <velero-sample-1> (3)
    defaultVolumesToRestic: true (4)
    ttl: 720h0m0s
EOF
```

(1) **cron** expression to schedule the backup, for example, **0 7 * * *** to perform a backup every day at 7:00.

(2) Array of namespaces to back up.

(3) Name of the **backupStorageLocations** CR.

(4) Optional: Add the **defaultVolumesToRestic: true** key-value pair if you are backing up volumes with Restic.

3. Verify that the status of the **Schedule** CR is **Completed** after the scheduled backup runs:

```
$ oc get schedule -n openshift-adp <schedule> -o jsonpath='{.status.phase}'
```

4.4.2. Restoring applications

You restore application backups by creating a **Restore** custom resources (CRs).

You can create restore hooks to run commands in init containers, before the application container starts, or in the application container itself.

4.4.2.1. Creating a Restore CR

You restore a **Backup** custom resource (CR) by creating a **Restore** CR.

Prerequisites

- You must install the OpenShift API for Data Protection (OADP) Operator.
- The **DataProtectionApplication** CR must be in a **Ready** state.
- You must have a Velero **Backup** CR.
- Adjust the requested size so the persistent volume (PV) capacity matches the requested size at backup time.

Procedure

1. Create a **Restore** CR, as in the following example:

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  backupName: <backup> 1
  includedResources: [] 2
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  restorePVs: true
```

1 Name of the **Backup** CR.

2 Optional. Specify an array of resources to include in the restore process. Resources might be shortcuts (for example, 'po' for 'pods') or fully-qualified. If unspecified, all resources are included.

2. Verify that the status of the **Restore** CR is **Completed** by entering the following command:

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

3. Verify that the backup resources have been restored by entering the following command:

```
$ oc get all -n <namespace> 1
```

1 Namespace that you backed up.

4.4.2.2. Creating restore hooks

You create restore hooks to run commands in a container in a pod while restoring your application by editing the **Restore** custom resource (CR).

You can create two types of restore hooks:

- An **init** hook adds an init container to a pod to perform setup tasks before the application container starts.
  If you restore a Restic backup, the **restic-wait** init container is added before the restore hook init container.

- An **exec** hook runs commands or scripts in a container of a restored pod.

Procedure

- Add a hook to the **spec.hooks** block of the **Restore** CR, as in the following example:

```
apiVersion: velero.io/v1
```

```
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
        - <namespace> 1
        excludedNamespaces:
        - <namespace>
        includedResources:
        - pods 2
        excludedResources: []
        labelSelector: 3
          matchLabels:
            app: velero
            component: server
        postHooks:
        - init:
            initContainers:
            - name: restore-hook-init
              image: alpine:latest
              volumeMounts:
              - mountPath: /restores/pvc1-vm
                name: pvc1-vm
              command:
              - /bin/ash
              - -c
            timeout: 4
        - exec:
            container: <container> 5
            command:
            - /bin/bash 6
            - -c
            - "psql < /backup/backup.sql"
            waitTimeout: 5m 7
            execTimeout: 1m 8
            onError: Continue 9
```

1 Optional: Array of namespaces to which the hook applies. If this value is not specified, the hook applies to all namespaces.

2 Currently, pods are the only supported resource that hooks can apply to.

3 Optional: This hook only applies to objects matching the label selector.

4 Optional: Timeout specifies the maximum amount of time Velero waits for **initContainers** to complete.

5 Optional: If the container is not specified, the command runs in the first container in the pod.

6 This is the entrypoint for the init container being added.

7 Optional: How long to wait for a container to become ready. This should be long enough for the container to start and for any preceding hooks in the same container to complete. If not set, the restore process waits indefinitely.

8 Optional: How long to wait for the commands to run. The default is **30s**.

9 Allowed values for error handling are **Fail** and **Continue**:

- **Continue**: Only command failures are logged.
- **Fail**: No more restore hooks run in any container in any pod. The status of the **Restore** CR will be **PartiallyFailed**.

4.5. TROUBLESHOOTING

You can debug Velero custom resources (CRs) by using the OpenShift CLI tool or the Velero CLI tool. The Velero CLI tool provides more detailed logs and information.

You can check installation issues, backup and restore CR issues, and Restic issues.

You can collect logs, CR information, and Prometheus metric data by using the **must-gather** tool.

You can obtain the Velero CLI tool by:

- Downloading the Velero CLI tool
- Accessing the Velero binary in the Velero deployment in the cluster

4.5.1. Downloading the Velero CLI tool

You can download and install the Velero CLI tool by following the instructions on the Velero documentation page.

The page includes instructions for:

- macOS by using Homebrew
- GitHub
- Windows by using Chocolatey

Prerequisites

- You have access to a Kubernetes cluster, v1.16 or later, with DNS and container networking enabled.
- You have installed **kubectl** locally.

Procedure

1. Open a browser and navigate to "Install the CLI" on the Verleo website.
2. Follow the appropriate procedure for macOS, GitHub, or Windows.

3. Download the Velero version appropriate for your version of OADP, according to the table that follows:

Table 4.2. OADP-Velero version relationship

OADP version	Velero version
0.2.6	1.6.0
0.5.5	1.7.1
1.0.0	1.7.1
1.0.1	1.7.1
1.0.2	1.7.1
1.0.3	1.7.1

4.5.2. Accessing the Velero binary in the Velero deployment in the cluster

You can use a shell command to access the Velero binary in the Velero deployment in the cluster.

Prerequisites

- Your **DataProtectionApplication** custom resource has a status of **Reconcile complete**.

Procedure

- Enter the following command to set the needed alias:

```
$ alias velero='oc -n openshift-adp exec deployment/velero -c velero -it -- ./velero'
```

4.5.3. Debugging Velero resources with the OpenShift CLI tool

You can debug a failed backup or restore by checking Velero custom resources (CRs) and the **Velero** pod log with the OpenShift CLI tool.

Velero CRs
Use the **oc describe** command to retrieve a summary of warnings and errors associated with a **Backup** or **Restore** CR:

```
$ oc describe <velero_cr> <cr_name>
```

Velero pod logs
Use the **oc logs** command to retrieve the **Velero** pod logs:

```
$ oc logs pod/<velero>
```

Velero pod debug logs

You can specify the Velero log level in the **DataProtectionApplication** resource as shown in the following example.

NOTE

This option is available starting from OADP 1.0.3.

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero-sample
spec:
  configuration:
    velero:
      logLevel: warning
```

The following **logLevel** values are available:

- **trace**
- **debug**
- **info**
- **warning**
- **error**
- **fatal**
- **panic**

It is recommended to use **debug** for most logs.

4.5.4. Debugging Velero resources with the Velero CLI tool

You can debug **Backup** and **Restore** custom resources (CRs) and retrieve logs with the Velero CLI tool.

The Velero CLI tool provides more detailed information than the OpenShift CLI tool.

Syntax
Use the **oc exec** command to run a Velero CLI command:

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  <backup_restore_cr> <command> <cr_name>
```

Example

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  backup describe 0e44ae00-5dc3-11eb-9ca8-df7e5254778b-2d8ql
```

Help option
Use the **velero --help** option to list all Velero CLI commands:

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  --help
```

Describe command
Use the **velero describe** command to retrieve a summary of warnings and errors associated with a **Backup** or **Restore** CR:

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  <backup_restore_cr> describe <cr_name>
```

Example

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  backup describe 0e44ae00-5dc3-11eb-9ca8-df7e5254778b-2d8ql
```

Logs command
Use the **velero logs** command to retrieve the logs of a **Backup** or **Restore** CR:

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  <backup_restore_cr> logs <cr_name>
```

Example

```
$ oc -n openshift-adp exec deployment/velero -c velero -- ./velero \
  restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

4.5.5. Issues with Velero and admission webhooks

Velero has limited abilities to resolve admission webhook issues during a restore. If you have workloads with admission webhooks, you might need to use an additional Velero plugin or make changes to how you restore the workload.

Typically, workloads with admission webhooks require you to create a resource of a specific kind first. This is especially true if your workload has child resources because admission webhooks typically block child resources.

For example, creating or restoring a top-level object such as **service.serving.knative.dev** typically creates child resources automatically. If you do this first, you will not need to use Velero to create and restore these resources. This avoids the problem of child resources being blocked by an admission webhook that Velero might use.

4.5.5.1. Restoring workarounds for Velero backups that use admission webhooks

This section describes the additional steps required to restore resources for several types of Velero backups that use admission webhooks.

4.5.5.1.1. Restoring Knative resources

You might encounter problems using Velero to back up Knative resources that use admission webhooks.

You can avoid such problems by restoring the top level **Service** resource first whenever you back up and restore Knative resources that use admission webhooks.

Procedure

- Restore the top level **service.serving.knavtive.dev Service** resource:

```
$ velero restore <restore_name> \
  --from-backup=<backup_name> --include-resources \
  service.serving.knavtive.dev
```

4.5.5.1.2. Restoring IBM AppConnect resources

If you experience issues when you use Velero to a restore an IBM AppConnect resource that has an admission webhook, you can run the checks in this procedure.

Procedure

1. Check if you have any mutating admission plugins of **kind: MutatingWebhookConfiguration** in the cluster:

```
$ oc get mutatingwebhookconfigurations
```

2. Examine the YAML file of each **kind: MutatingWebhookConfiguration** to ensure that none of its rules block creation of the objects that are experiencing issues. For more information, see the official Kuberbetes documentation.

3. Check that any **spec.version** in **type: Configuration.appconnect.ibm.com/v1beta1** used at backup time is supported by the installed Operator.

Additional resources

- Admission plugins
- Webhook admission plugins
- Types of webhook admission plugins

4.5.6. Installation issues

You might encounter issues caused by using invalid directories or incorrect credentials when you install the Data Protection Application.

4.5.6.1. Backup storage contains invalid directories

The **Velero** pod log displays the error message, **Backup storage contains invalid top-level directories**.

Cause

The object storage contains top-level directories that are not Velero directories.

Solution

If the object storage is not dedicated to Velero, you must specify a prefix for the bucket by setting the **spec.backupLocations.velero.objectStorage.prefix** parameter in the **DataProtectionApplication** manifest.

4.5.6.2. Incorrect AWS credentials

The **oadp-aws-registry** pod log displays the error message, **InvalidAccessKeyId: The AWS Access Key Id you provided does not exist in our records.**

The **Velero** pod log displays the error message, **NoCredentialProviders: no valid providers in chain**.

Cause

The **credentials-velero** file used to create the **Secret** object is incorrectly formatted.

Solution

Ensure that the **credentials-velero** file is correctly formatted, as in the following example:

Example credentials-velero file

```
[default] (1)
aws_access_key_id=AKIAIOSFODNN7EXAMPLE (2)
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

(1) AWS default profile.

(2) Do not enclose the values with quotation marks (**"**, **'**).

4.5.7. Backup and Restore CR issues

You might encounter these common issues with **Backup** and **Restore** custom resources (CRs).

4.5.7.1. Backup CR cannot retrieve volume

The **Backup** CR displays the error message, **InvalidVolume.NotFound: The volume 'vol-xxxx' does not exist**.

Cause

The persistent volume (PV) and the snapshot locations are in different regions.

Solution

1. Edit the value of the **spec.snapshotLocations.velero.config.region** key in the **DataProtectionApplication** manifest so that the snapshot location is in the same region as the PV.

2. Create a new **Backup** CR.

4.5.7.2. Backup CR status remains in progress

The status of a **Backup** CR remains in the **InProgress** phase and does not complete.

Cause

If a backup is interrupted, it cannot be resumed.

Solution

1. Retrieve the details of the **Backup** CR:

   ```
   $ oc -n {namespace} exec deployment/velero -c velero -- ./velero \
     backup describe <backup>
   ```

2. Delete the **Backup** CR:

   ```
   $ oc delete backup <backup> -n openshift-adp
   ```

   You do not need to clean up the backup location because a **Backup** CR in progress has not uploaded files to object storage.

3. Create a new **Backup** CR.

4.5.8. Restic issues

You might encounter these issues when you back up applications with Restic.

4.5.8.1. Restic permission error for NFS data volumes with root_squash enabled

The **Restic** pod log displays the error message: **controller=pod-volume-backup error="fork/exec/usr/bin/restic: permission denied"**.

Cause

If your NFS data volumes have **root_squash** enabled, **Restic** maps to **nfsnobody** and does not have permission to create backups.

Solution

You can resolve this issue by creating a supplemental group for **Restic** and adding the group ID to the **DataProtectionApplication** manifest:

1. Create a supplemental group for **Restic** on the NFS data volume.

2. Set the **setgid** bit on the NFS directories so that group ownership is inherited.

3. Add the **spec.configuration.restic.supplementalGroups** parameter and the group ID to the **DataProtectionApplication** manifest, as in the following example:

   ```
   spec:
     configuration:
       restic:
         enable: true
         supplementalGroups:
         - <group_id> 1
   ```

   1 Specify the supplemental group ID.

4. Wait for the **Restic** pods to restart so that the changes are applied.

4.5.8.2. Restore CR of Restic backup is "PartiallyFailed", "Failed", or remains "InProgress"

The **Restore** CR of a Restic backup completes with a **PartiallyFailed** or **Failed** status or it remains **InProgress** and does not complete.

If the status is **PartiallyFailed** or **Failed**, the **Velero** pod log displays the error message, **level=error msg="unable to successfully complete restic restores of pod's volumes"**.

If the status is **InProgress**, the **Restore** CR logs are unavailable and no errors appear in the **Restic** pod logs.

Cause

The **DeploymentConfig** object redeploys the **Restore** pod, causing the **Restore** CR to fail.

Solution

1. Create a **Restore** CR that excludes the **ReplicationController**, **DeploymentConfig**, and **TemplateInstances** resources:

```
$ velero restore create --from-backup=<backup> -n openshift-adp \ (1)
  --include-namespaces <namespace> \ (2)
  --exclude-resources
replicationcontroller,deploymentconfig,templateinstances.template.openshift.io \
  --restore-volumes=true
```

(1) Specify the name of the **Backup** CR.

(2) Specify the **include-namespaces** in the **Backup** CR.

2. Verify that the status of the **Restore** CR is **Completed**:

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

3. Create a **Restore** CR that includes the **ReplicationController** and **DeploymentConfig** resources:

```
$ velero restore create --from-backup=<backup> -n openshift-adp \
  --include-namespaces <namespace> \
  --include-resources replicationcontroller,deploymentconfig \
  --restore-volumes=true
```

4. Verify that the status of the **Restore** CR is **Completed**:

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

5. Verify that the backup resources have been restored:

```
$ oc get all -n <namespace>
```

4.5.8.3. Restic Backup CR cannot be recreated after bucket is emptied

If you create a Restic **Backup** CR for a namespace, empty the object storage bucket, and then recreate the **Backup** CR for the same namespace, the recreated **Backup** CR fails.

The **velero** pod log displays the following error message: **stderr=Fatal: unable to open config file: Stat: The specified key does not exist.\nIs there a repository at the following location?**.

Cause

Velero does not recreate or update the Restic repository from the **ResticRepository** manifest if the Restic directories are deleted from object storage. See Velero issue 4421 for more information.

Solution

- Remove the related Restic repository from the namespace by running the following command:

```
$ oc delete resticrepository openshift-adp <name_of_the_restic_repository>
```

  In the following error log, **mysql-persistent** is the problematic Restic repository. The name of the repository appears in italics for clarity.

```
time="2021-12-29T18:29:14Z" level=info msg="1 errors
encountered backup up item" backup=velero/backup65
logSource="pkg/backup/backup.go:431" name=mysql-7d99fc949-qbkds
time="2021-12-29T18:29:14Z" level=error msg="Error backing up item"
backup=velero/backup65 error="pod volume backup failed: error running
restic backup, stderr=Fatal: unable to open config file: Stat: The
specified key does not exist.\nIs there a repository at the following
location?\ns3:http://minio-minio.apps.mayap-oadp-
veleo-1234.qe.devcluster.openshift.com/mayapvelerooadp2/velero1/
restic/mysql-persistent\n: exit status 1" error.file="/remote-source/
src/github.com/vmware-tanzu/velero/pkg/restic/backupper.go:184"
error.function="github.com/vmware-tanzu/velero/
pkg/restic.(*backupper).BackupPodVolumes"
logSource="pkg/backup/backup.go:435" name=mysql-7d99fc949-qbkds
```

4.5.9. Using the must-gather tool

You can collect logs, metrics, and information about OADP custom resources by using the **must-gather** tool.

The **must-gather** data must be attached to all customer cases.

Prerequisites

- You must be logged in to the OpenShift Container Platform cluster as a user with the **cluster-admin** role.
- You must have the OpenShift CLI (**oc**) installed.

Procedure

1. Navigate to the directory where you want to store the **must-gather** data.
2. Run the **oc adm must-gather** command for one of the following data collection options:

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.1
```

The data is saved as **must-gather/must-gather.tar.gz**. You can upload this file to a support case on the Red Hat Customer Portal .

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.1 \
  -- /usr/bin/gather_metrics_dump
```

This operation can take a long time. The data is saved as **must-gather/metrics/prom_data.tar.gz**.

Viewing metrics data with the Prometheus console
You can view the metrics data with the Prometheus console.

Procedure

1. Decompress the **prom_data.tar.gz** file:

```
$ tar -xvzf must-gather/metrics/prom_data.tar.gz
```

2. Create a local Prometheus instance:

```
$ make prometheus-run
```

The command outputs the Prometheus URL.

Output

```
Started Prometheus on http://localhost:9090
```

3. Launch a web browser and navigate to the URL to view the data by using the Prometheus web console.

4. After you have viewed the data, delete the Prometheus instance and data:

```
$ make prometheus-cleanup
```

4.6. APIS USED WITH OADP

The document provides information about the following APIs that you can use with OADP:

- Velero API
- OADP API

4.6.1. Velero API

Velero API documentation is maintained by Velero, not by Red Hat. It can be found at Velero API types.

4.6.2. OADP API

The following tables provide the structure of the OADP API:

Table 4.3. DataProtectionApplicationSpec

Property	Type	Description
backupLocations	[] **BackupLocation**	Defines the list of configurations to use for **BackupStorageLocations**.
snapshotLocations	[] **SnapshotLocation**	Defines the list of configurations to use for **VolumeSnapshotLocations**.
unsupportedOverrides	map [UnsupportedImageKey] string	Can be used to override the deployed dependent images for development. Options are **veleroImageFqin**, **awsPluginImageFqin**, **openshiftPluginImageFqin**, **azurePluginImageFqin**, **gcpPluginImageFqin**, **csiPluginImageFqin**, **dataMoverImageFqin**, **resticRestoreImageFqin**, **kubevirtPluginImageFqin**, and **operator-type**.
podAnnotations	map [string] string	Used to add annotations to pods deployed by Operators.
podDnsPolicy	**DNSPolicy**	Defines the configuration of the DNS of a pod.
podDnsConfig	**PodDNSConfig**	Defines the DNS parameters of a pod in addition to those generated from **DNSPolicy**.
backupImages	*bool	Used to specify whether or not you want to deploy a registry for enabling backup and restore of images.
configuration	***ApplicationConfig**	Used to define the data protection application's server configuration.
features	***Features**	Defines the configuration for the DPA to enable the Technology Preview features.

Complete schema definitions for the OADP API .

Table 4.4. BackupLocation

Property	Type	Description
velero	*velero.BackupStorageLocationSpec	Location to store volume snapshots, as described in Backup Storage Location.
bucket	*CloudStorageLocation	[Technology Preview] Automates creation of a bucket at some cloud storage providers for use as a backup storage location.

IMPORTANT

The **bucket** parameter is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see Technology Preview.

Complete schema definitions for the type **BackupLocation**.

Table 4.5. SnapshotLocation

Property	Type	Description
velero	*VolumeSnapshotLocationSpec	Location to store volume snapshots, as described in Volume Snapshot Location.

Complete schema definitions for the type **SnapshotLocation**.

Table 4.6. ApplicationConfig

Property	Type	Description
velero	*VeleroConfig	Defines the configuration for the Velero server.
restic	*ResticConfig	Defines the configuration for the Restic server.

Complete schema definitions for the type **ApplicationConfig**.

Table 4.7. VeleroConfig

Property	Type	Description
featureFlags	[] string	Defines the list of features to enable for the Velero instance.
defaultPlugins	[] string	The following types of default Velero plugins can be installed: **aws**,**azure**, **csi**, **gcp**, **kubevirt**, and **openshift**.
customPlugins	[]CustomPlugin	Used for installation of custom Velero plugins. Default and custom plugins are described in OADP plugins
restoreResourcesVersionPriority	string	Represents a config map that is created if defined for use in conjunction with the **EnableAPIGroupVersions** feature flag. Defining this field automatically adds **EnableAPIGroupVersions** to the Velero server feature flag.
noDefaultBackupLocation	bool	To install Velero without a default backup storage location, you must set the **noDefaultBackupLocation** flag in order to confirm installation.
podConfig	***PodConfig**	Defines the configuration of the **Velero** pod.
logLevel	string	Velero server's log level (use **debug** for the most granular logging, leave unset for Velero default). Valid options are **trace**, **debug**, **info**, **warning**, **error**, **fatal**, and **panic**.

Complete schema definitions for the type **VeleroConfig**.

Table 4.8. CustomPlugin

Property	Type	Description
name	string	Name of custom plugin.
image	string	Image of custom plugin.

Complete schema definitions for the type **CustomPlugin**.

Table 4.9. ResticConfig

Property	Type	Description
enable	*bool	If set to **true**, enables backup and restore using Restic. If set to **false**, snapshots are needed.
supplementalGroups	[]int64	Defines the Linux groups to be applied to the **Restic** pod.
timeout	string	A user-supplied duration string that defines the Restic timeout. Default value is **1hr** (1 hour). A duration string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as **300ms**, -1.5h` or **2h45m**. Valid time units are**ns**, **us** (or **µs**), **ms**, **s**, **m**, and **h**.
podConfig	***PodConfig**	Defines the configuration of the **Restic** pod.

Complete schema definitions for the type **ResticConfig**.

Table 4.10. PodConfig

Property	Type	Description
nodeSelector	map [string] string	Defines the **nodeSelector** to be supplied to a **Velero podSpec** or a **Restic podSpec**.
tolerations	[]Toleration	Defines the list of tolerations to be applied to a Velero deployment or a Restic **daemonset**.
resourceAllocations	ResourceRequirements	Set specific resource **limits** and **requests** for a **Velero** pod or a **Restic** pod as described in Setting Velero CPU and memory resource allocations.
labels	map [string] string	Labels to add to pods.

Complete schema definitions for the type **PodConfig**.

Table 4.11. Features

Property	Type	Description
dataMover	*DataMover	Defines the configuration of the Data Mover.

Complete schema definitions for the type **Features**.

Table 4.12. DataMover

Property	Type	Description
enable	bool	If set to **true**, deploys the volume snapshot mover controller and a modified CSI Data Mover plugin. If set to **false**, these are not deployed.
credentialName	string	User-supplied Restic **Secret** name for Data Mover.
timeout	string	A user-supplied duration string for **VolumeSnapshotBackup** and **VolumeSnapshotRestore** to complete. Default is **10m** (10 minutes). A duration string is a possibly signed sequence of decimal numbers, each with optional fraction and a unit suffix, such as **300ms**, -1.5h` or **2h45m**. Valid time units are **ns**, **us** (or **µs**), **ms**, **s**, **m**, and **h**.

The OADP API is more fully detailed in OADP Operator.

4.7. ADVANCED OADP FEATURES AND FUNCTIONALITIES

This document provides information on advanced features and functionalities of OpenShift API for Data Protection (OADP).

4.7.1. Working with different Kubernetes API versions on the same cluster

4.7.1.1. Listing the Kubernetes API group versions on a cluster

A source cluster might offer multiple versions of an API, where one of these versions is the preferred API version. For example, a source cluster with an API named **Example** might be available in the **example.com/v1** and **example.com/v1beta2** API groups.

If you use Velero to back up and restore such a source cluster, Velero backs up only the version of that resource that uses the preferred version of its Kubernetes API.

To return to the above example, if **example.com/v1** is the preferred API, then Velero only backs up the version of a resource that uses **example.com/v1**. Moreover, the target cluster needs to have **example.com/v1** registered in its set of available API resources in order for Velero to restore the resource on the target cluster.

Therefore, you need to generate a list of the Kubernetes API group versions on your target cluster to be sure the prefered API version is registered in its set of available API resources.

Procedure

- Enter the following command:

```
$ oc api-resources
```

4.7.1.2. About Enable API Group Versions

By default, Velero only backs up resources that use the preferred version of the Kubernetes API. However, Velero also includes a feature, Enable API Group Versions , that overcomes this limitation. When enabled on the source cluster, this feature causes Velero to back up *all* Kubernetes API group versions that are supported on the cluster, not only the preferred one. After the versions are stored in the backup .tar file, they are available to be restored on the destination cluster.

For example, a source cluster with an API named **Example** might be available in the **example.com/v1** and **example.com/v1beta2** API groups, with **example.com/v1** being the preferred API.

Without the Enable API Group Versions feature enabled, Velero backs up only the preferred API group version for **Example**, which is **example.com/v1**. With the feature enabled, Velero also backs up **example.com/v1beta2**.

When the Enable API Group Versions feature is enabled on the destination cluster, Velero selects the version to restore on the basis of the order of priority of API group versions.

NOTE

Enable API Group Versions is still in beta.

Velero uses the following algorithm to assign priorities to API versions, with **1** as the top priority:

1. Preferred version of the *destination* cluster

2. Preferred version of the source_ cluster

3. Common non-preferred supported version with the highest Kubernetes version priority

Additional resources

- Enable API Group Versions Feature

4.7.1.3. Using Enable API Group Versions

You can use Velero’s Enable API Group Versions feature to back up *all* Kubernetes API group versions that are supported on a cluster, not only the preferred one.

NOTE

Enable API Group Versions is still in beta.

Procedure

- Configure the **EnableAPIGroupVersions** feature flag:

```
apiVersion: oadp.openshift.io/vialpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      featureFlags:
      - EnableAPIGroupVersions
```

Additional resources

- Enable API Group Versions Feature

CHAPTER 5. CONTROL PLANE BACKUP AND RESTORE

5.1. BACKING UP ETCD

etcd is the key-value store for OpenShift Container Platform, which persists the state of all resource objects.

Back up your cluster's etcd data regularly and store in a secure location ideally outside the OpenShift Container Platform environment. Do not take an etcd backup before the first certificate rotation completes, which occurs 24 hours after installation, otherwise the backup will contain expired certificates. It is also recommended to take etcd backups during non-peak usage hours because the etcd snapshot has a high I/O cost.

Be sure to take an etcd backup after you upgrade your cluster. This is important because when you restore your cluster, you must use an etcd backup that was taken from the same z-stream release. For example, an OpenShift Container Platform 4.y.z cluster must use an etcd backup that was taken from 4.y.z.

IMPORTANT

Back up your cluster's etcd data by performing a single invocation of the backup script on a control plane host. Do not take a backup for each control plane host.

After you have an etcd backup, you can restore to a previous cluster state .

5.1.1. Backing up etcd data

Follow these steps to back up etcd data by creating an etcd snapshot and backing up the resources for the static pods. This backup can be saved and used at a later time if you need to restore etcd.

IMPORTANT

Only save a backup from a single control plane host. Do not take a backup from each control plane host in the cluster.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have checked whether the cluster-wide proxy is enabled.

  TIP

  You can check whether the proxy is enabled by reviewing the output of **oc get proxy cluster -o yaml**. The proxy is enabled if the **httpProxy**, **httpsProxy**, and **noProxy** fields have values set.

Procedure

1. Start a debug session for a control plane node:

```
$ oc debug node/<node_name>
```

2. Change your root directory to **/host**:

```
sh-4.2# chroot /host
```

3. If the cluster-wide proxy is enabled, be sure that you have exported the **NO_PROXY**, **HTTP_PROXY**, and **HTTPS_PROXY** environment variables.

4. Run the **cluster-backup.sh** script and pass in the location to save the backup to.

   TIP

   The **cluster-backup.sh** script is maintained as a component of the etcd Cluster Operator and is a wrapper around the **etcdctl snapshot save** command.

```
sh-4.4# /usr/local/bin/cluster-backup.sh /home/core/assets/backup
```

   Example script output

```
found latest kube-apiserver: /etc/kubernetes/static-pod-resources/kube-apiserver-pod-6
found latest kube-controller-manager: /etc/kubernetes/static-pod-resources/kube-controller-
manager-pod-7
found latest kube-scheduler: /etc/kubernetes/static-pod-resources/kube-scheduler-pod-6
found latest etcd: /etc/kubernetes/static-pod-resources/etcd-pod-3
ede95fe6b88b87ba86a03c15e669fb4aa5bf0991c180d3c6895ce72eaade54a1
etcdctl version: 3.4.14
API version: 3.4
{"level":"info","ts":1624647639.0188997,"caller":"snapshot/v3_snapshot.go:119","msg":"created
temporary db file","path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db.part"}
{"level":"info","ts":"2021-06-
25T19:00:39.030Z","caller":"clientv3/maintenance.go:200","msg":"opened snapshot stream;
downloading"}
{"level":"info","ts":1624647639.0301006,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching
snapshot","endpoint":"https://10.0.0.5:2379"}
{"level":"info","ts":"2021-06-
25T19:00:40.215Z","caller":"clientv3/maintenance.go:208","msg":"completed snapshot read;
closing"}
{"level":"info","ts":1624647640.6032252,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched
snapshot","endpoint":"https://10.0.0.5:2379","size":"114 MB","took":1.584090459}
{"level":"info","ts":1624647640.6047094,"caller":"snapshot/v3_snapshot.go:152","msg":"saved",
"path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db"}
Snapshot saved at /home/core/assets/backup/snapshot_2021-06-25_190035.db
{"hash":3866667823,"revision":31407,"totalKey":12828,"totalSize":114446336}
snapshot db and kube resources are successfully saved to /home/core/assets/backup
```

   In this example, two files are created in the **/home/core/assets/backup/** directory on the control plane host:

   - **snapshot_<datetimestamp>.db**: This file is the etcd snapshot. The **cluster-backup.sh** script confirms its validity.
   - **static_kuberesources_<datetimestamp>.tar.gz**: This file contains the resources for the static pods. If etcd encryption is enabled, it also contains the encryption keys for the etcd snapshot.

NOTE

If etcd encryption is enabled, it is recommended to store this second file separately from the etcd snapshot for security reasons. However, this file is required to restore from the etcd snapshot.

Keep in mind that etcd encryption only encrypts values, not keys. This means that resource types, namespaces, and object names are unencrypted.

5.2. REPLACING AN UNHEALTHY ETCD MEMBER

This document describes the process to replace a single unhealthy etcd member.

This process depends on whether the etcd member is unhealthy because the machine is not running or the node is not ready, or whether it is unhealthy because the etcd pod is crashlooping.

NOTE

If you have lost the majority of your control plane hosts, leading to etcd quorum loss, then you must follow the disaster recovery procedure to restore to a previous cluster state instead of this procedure.

If the control plane certificates are not valid on the member being replaced, then you must follow the procedure to recover from expired control plane certificates instead of this procedure.

If a control plane node is lost and a new one is created, the etcd cluster Operator handles generating the new TLS certificates and adding the node as an etcd member.

5.2.1. Prerequisites

- Take an etcd backup prior to replacing an unhealthy etcd member.

5.2.2. Identifying an unhealthy etcd member

You can identify if your cluster has an unhealthy etcd member.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.

Procedure

1. Check the status of the **EtcdMembersAvailable** status condition using the following command:

```
$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="EtcdMembersAvailable")]}{.message}{"\n"}'
```

2. Review the output:

```
2 of 3 members are available, ip-10-0-131-183.ec2.internal is unhealthy
```

This example output shows that the **ip-10-0-131-183.ec2.internal** etcd member is unhealthy.

5.2.3. Determining the state of the unhealthy etcd member

The steps to replace an unhealthy etcd member depend on which of the following states your etcd member is in:

- The machine is not running or the node is not ready
- The etcd pod is crashlooping

This procedure determines which state your etcd member is in. This enables you to know which procedure to follow to replace the unhealthy etcd member.

> **NOTE**
>
> If you are aware that the machine is not running or the node is not ready, but you expect it to return to a healthy state soon, then you do not need to perform a procedure to replace the etcd member. The etcd cluster Operator will automatically sync when the machine or node returns to a healthy state.

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have identified an unhealthy etcd member.

Procedure

1. Determine if the **machine is not running**:

   ```
   $ oc get machines -A -ojsonpath='{range .items[*]}{@.status.nodeRef.name}{"\t"}
   {@.status.providerStatus.instanceState}{"\n"}' | grep -v running
   ```

   Example output

   ```
   ip-10-0-131-183.ec2.internal  stopped (1)
   ```

   (1) This output lists the node and the status of the node's machine. If the status is anything other than **running**, then the **machine is not running**.

   If the **machine is not running**, then follow the *Replacing an unhealthy etcd member whose machine is not running or whose node is not ready* procedure.

2. Determine if the **node is not ready**.
   If either of the following scenarios are true, then the **node is not ready**.

   - If the machine is running, then check whether the node is unreachable:

     ```
     $ oc get nodes -o jsonpath='{range .items[*]}{"\n"}{.metadata.name}{"\t"}{range
     .spec.taints[*]}{.key}{" "}' | grep unreachable
     ```

     Example output

```
ip-10-0-131-183.ec2.internal node-role.kubernetes.io/master
node.kubernetes.io/unreachable node.kubernetes.io/unreachable 1
```

1 If the node is listed with an **unreachable** taint, then the **node is not ready**.

- If the node is still reachable, then check whether the node is listed as **NotReady**:

  ```
  $ oc get nodes -l node-role.kubernetes.io/master | grep "NotReady"
  ```

  Example output

  ```
  ip-10-0-131-183.ec2.internal   NotReady   master   122m   v1.25.0 1
  ```

  1 If the node is listed as **NotReady**, then the **node is not ready**.

If the **node is not ready**, then follow the *Replacing an unhealthy etcd member whose machine is not running or whose node is not ready* procedure.

3. Determine if the **etcd pod is crashlooping**
   If the machine is running and the node is ready, then check whether the etcd pod is crashlooping.

   a. Verify that all control plane nodes are listed as **Ready**:

   ```
   $ oc get nodes -l node-role.kubernetes.io/master
   ```

   Example output

   ```
   NAME                           STATUS   ROLES    AGE     VERSION
   ip-10-0-131-183.ec2.internal   Ready    master   6h13m   v1.25.0
   ip-10-0-164-97.ec2.internal    Ready    master   6h13m   v1.25.0
   ip-10-0-154-204.ec2.internal   Ready    master   6h13m   v1.25.0
   ```

   b. Check whether the status of an etcd pod is either **Error** or **CrashloopBackoff**:

   ```
   $ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
   ```

   Example output

   ```
   etcd-ip-10-0-131-183.ec2.internal                2/3     Error       7          6h9m 1
   etcd-ip-10-0-164-97.ec2.internal                 3/3     Running     0          6h6m
   etcd-ip-10-0-154-204.ec2.internal                3/3     Running     0          6h6m
   ```

   1 Since this status of this pod is **Error**, then the **etcd pod is crashlooping**

If the **etcd pod is crashlooping** then follow the *Replacing an unhealthy etcd member whose etcd pod is crashlooping* procedure.

5.2.4. Replacing the unhealthy etcd member

Depending on the state of your unhealthy etcd member, use one of the following procedures:

- Replacing an unhealthy etcd member whose machine is not running or whose node is not ready
- Replacing an unhealthy etcd member whose etcd pod is crashlooping
- Replacing an unhealthy stopped baremetal etcd member

5.2.4.1. Replacing an unhealthy etcd member whose machine is not running or whose node is not ready

This procedure details the steps to replace an etcd member that is unhealthy either because the machine is not running or because the node is not ready.

NOTE

If your cluster uses a control plane machine set, see "Troubleshooting the control plane machine set" for a more simple etcd recovery procedure.

Prerequisites

- You have identified the unhealthy etcd member.
- You have verified that either the machine is not running or the node is not ready.
- You have access to the cluster as a user with the **cluster-admin** role.
- You have taken an etcd backup.

  IMPORTANT

  It is important to take an etcd backup before performing this procedure so that your cluster can be restored if you encounter any issues.

Procedure

1. Remove the unhealthy member.

   a. Choose a pod that is *not* on the affected node:
      In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

      ```
      $ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
      ```

      Example output

      ```
      etcd-ip-10-0-131-183.ec2.internal                3/3     Running     0          123m
      etcd-ip-10-0-164-97.ec2.internal                 3/3     Running     0          123m
      etcd-ip-10-0-154-204.ec2.internal                3/3     Running     0          124m
      ```

   b. Connect to the running etcd container, passing in the name of a pod that is not on the affected node:
      In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

c. View the member list:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
|        ID        | STATUS  |             NAME             |        PEER ADDRS         |        CLIENT
ADDRS        |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
| 6fc1e7c9db35841d | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 |
https://10.0.131.183:2379 |
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
```

Take note of the ID and the name of the unhealthy etcd member, because these values are needed later in the procedure. The **$ etcdctl endpoint health** command will list the removed member until the procedure of replacement is finished and a new member is added.

d. Remove the unhealthy etcd member by providing the ID to the **etcdctl member remove** command:

```
sh-4.2# etcdctl member remove 6fc1e7c9db35841d
```

Example output

```
Member 6fc1e7c9db35841d removed from cluster ead669ce1fbfb346
```

e. View the member list again and verify that the member was removed:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
|        ID        | STATUS  |             NAME             |        PEER ADDRS         |        CLIENT
ADDRS        |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
```

You can now exit the node shell.

IMPORTANT

After you remove the member, the cluster might be unreachable for a short time while the remaining etcd instances reboot.

2. Turn off the quorum guard by entering the following command:

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides":
{"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
```

This command ensures that you can successfully re-create secrets and roll out the static pods.

3. Remove the old secrets for the unhealthy etcd member that was removed.

a. List the secrets for the unhealthy etcd member that was removed.

```
$ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal (1)
```

(1) Pass in the name of the unhealthy etcd member that you took note of earlier in this procedure.

There is a peer, serving, and metrics secret as shown in the following output:

Example output

```
etcd-peer-ip-10-0-131-183.ec2.internal            kubernetes.io/tls                2      47m
etcd-serving-ip-10-0-131-183.ec2.internal          kubernetes.io/tls                2      47m
etcd-serving-metrics-ip-10-0-131-183.ec2.internal   kubernetes.io/tls                2
47m
```

b. Delete the secrets for the unhealthy etcd member that was removed.

i. Delete the peer secret:

```
$ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
```

ii. Delete the serving secret:

```
$ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
```

iii. Delete the metrics secret:

```
$ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-
183.ec2.internal
```

4. Delete and recreate the control plane machine. After this machine is recreated, a new revision is forced and etcd scales up automatically.
   If you are running installer-provisioned infrastructure, or you used the Machine API to create your machines, follow these steps. Otherwise, you must create the new master using the same method that was used to originally create it.

   a. Obtain the machine for the unhealthy member.
      In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

      ```
      $ oc get machines -n openshift-machine-api -o wide
      ```

      Example output

      ```
      NAME                                        PHASE     TYPE        REGION      ZONE         AGE
      NODE                          PROVIDERID                            STATE
      clustername-8qw5l-master-0                  Running   m4.xlarge   us-east-1   us-east-1a
      3h37m   ip-10-0-131-183.ec2.internal   aws:///us-east-1a/i-0ec2782f8287dfb7e   stopped
      1
      clustername-8qw5l-master-1                  Running   m4.xlarge   us-east-1   us-east-1b
      3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
      clustername-8qw5l-master-2                  Running   m4.xlarge   us-east-1   us-east-1c
      3h37m   ip-10-0-164-97.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
      clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
      1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
      running
      clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
      3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
      clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
      1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
      running
      ```

      1 This is the control plane machine for the unhealthy node, **ip-10-0-131-183.ec2.internal**.

   b. Save the machine configuration to a file on your file system:

      ```
      $ oc get machine clustername-8qw5l-master-0 \ 1
          -n openshift-machine-api \
          -o yaml \
          > new-master-machine.yaml
      ```

      1 Specify the name of the control plane machine for the unhealthy node.

   c. Edit the **new-master-machine.yaml** file that was created in the previous step to assign a new name and remove unnecessary fields.

      i. Remove the entire **status** section:

         ```
         status:
           addresses:
           - address: 10.0.131.183
             type: InternalIP
           - address: ip-10-0-131-183.ec2.internal
         ```

```
    type: InternalDNS
  - address: ip-10-0-131-183.ec2.internal
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Node
    name: ip-10-0-131-183.ec2.internal
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2020-04-20T16:53:50Z"
      lastTransitionTime: "2020-04-20T16:53:50Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: AWSMachineProviderStatus
```

ii. Change the **metadata.name** field to a new name.
It is recommended to keep the same base name as the old machine and change the ending number to the next available number. In this example, **clustername-8qw5l-master-0** is changed to **clustername-8qw5l-master-3**.

For example:

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

iii. Remove the **spec.providerID** field:

```
  providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

d. Delete the machine of the unhealthy member:

```
$ oc delete machine -n openshift-machine-api clustername-8qw5l-master-0 1
```

1 Specify the name of the control plane machine for the unhealthy node.

e. Verify that the machine was deleted:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output

```
NAME                                PHASE    TYPE      REGION     ZONE       AGE
```

```
NODE                         PROVIDERID                              STATE
clustername-8qw5l-master-1                Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2                Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-164-97.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

f. Create the new machine using the **new-master-machine.yaml** file:

```
$ oc apply -f new-master-machine.yaml
```

g. Verify that the new machine has been created:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output

```
NAME                                   PHASE        TYPE        REGION      ZONE        AGE
NODE                         PROVIDERID                              STATE
clustername-8qw5l-master-1                Running       m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2                Running       m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-164-97.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-master-3                Provisioning   m4.xlarge   us-east-1   us-east-1a
85s     ip-10-0-133-53.ec2.internal    aws:///us-east-1a/i-015b0888fe17bc2c8   running
1
clustername-8qw5l-worker-us-east-1a-wbtgd   Running       m4.large    us-east-1   us-
east-1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running       m4.large    us-east-1   us-east-
1b   3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b
running
clustername-8qw5l-worker-us-east-1c-pkg26   Running       m4.large    us-east-1   us-
east-1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

1 The new machine, **clustername-8qw5l-master-3** is being created and is ready once the phase changes from **Provisioning** to **Running**.

It might take a few minutes for the new machine to be created. The etcd cluster Operator will automatically sync when the machine or node returns to a healthy state.

5. Turn the quorum guard back on by entering the following command:

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides": null}}
```

6. You can verify that the **unsupportedConfigOverrides** section is removed from the object by entering this command:

```
$ oc get etcd/cluster -oyaml
```

7. If you are using single-node OpenShift, restart the node. Otherwise, you might encounter the following error in the etcd cluster Operator:

Example output

```
EtcdCertSignerControllerDegraded: [Operation cannot be fulfilled on secrets "etcd-peer-sno-0": the object has been modified; please apply your changes to the latest version and try again, Operation cannot be fulfilled on secrets "etcd-serving-sno-0": the object has been modified; please apply your changes to the latest version and try again, Operation cannot be fulfilled on secrets "etcd-serving-metrics-sno-0": the object has been modified; please apply your changes to the latest version and try again]
```

Verification

1. Verify that all etcd pods are running properly.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

Example output

```
etcd-ip-10-0-133-53.ec2.internal          3/3   Running   0    7m49s
etcd-ip-10-0-164-97.ec2.internal          3/3   Running   0    123m
etcd-ip-10-0-154-204.ec2.internal          3/3   Running   0    124m
```

If the output from the previous command only lists two pods, you can manually force an etcd redeployment. In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge 1
```

1 The **forceRedeploymentReason** value must be unique, which is why a timestamp is appended.

2. Verify that there are exactly three etcd members.

a. Connect to the running etcd container, passing in the name of a pod that was not on the affected node:
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

b. View the member list:

```
sh-4.2# etcdctl member list -w table
```

■

Example output

```
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
|       ID       | STATUS  |          NAME          |       PEER ADDRS       |      CLIENT
ADDRS       |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
| 5eb0d6b8ca24730c | started |  ip-10-0-133-53.ec2.internal |  https://10.0.133.53:2380 |
https://10.0.133.53:2379 |
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+------------------+---------+------------------------------+---------------------------+----------------
-----------+
```

If the output from the previous command lists more than three etcd members, you must carefully remove the unwanted member.

WARNING

Be sure to remove the correct etcd member; removing a good etcd member might lead to quorum loss.

5.2.4.2. Replacing an unhealthy etcd member whose etcd pod is crashlooping

This procedure details the steps to replace an etcd member that is unhealthy because the etcd pod is crashlooping.

Prerequisites

- You have identified the unhealthy etcd member.
- You have verified that the etcd pod is crashlooping.
- You have access to the cluster as a user with the **cluster-admin** role.
- You have taken an etcd backup.

IMPORTANT

It is important to take an etcd backup before performing this procedure so that your cluster can be restored if you encounter any issues.

Procedure

1. Stop the crashlooping etcd pod.

a. Debug the node that is crashlooping.
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc debug node/ip-10-0-131-183.ec2.internal 1
```

1 Replace this with the name of the unhealthy node.

b. Change your root directory to /**host**:

```
sh-4.2# chroot /host
```

c. Move the existing etcd pod file out of the kubelet manifest directory:

```
sh-4.2# mkdir /var/lib/etcd-backup
```

```
sh-4.2# mv /etc/kubernetes/manifests/etcd-pod.yaml /var/lib/etcd-backup/
```

d. Move the etcd data directory to a different location:

```
sh-4.2# mv /var/lib/etcd/ /tmp
```

You can now exit the node shell.

2. Remove the unhealthy member.

a. Choose a pod that is *not* on the affected node.
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

Example output

```
etcd-ip-10-0-131-183.ec2.internal        2/3    Error      7        6h9m
etcd-ip-10-0-164-97.ec2.internal         3/3    Running    0        6h6m
etcd-ip-10-0-154-204.ec2.internal        3/3    Running    0        6h6m
```

b. Connect to the running etcd container, passing in the name of a pod that is not on the affected node.
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

c. View the member list:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+------------------------------+---------------------------+---------------------------+
|        ID        | STATUS  |             NAME             |        PEER ADDRS         |       CLIENT ADDRS        |
+------------------+---------+------------------------------+---------------------------+---------------------------+
| 62bcf33650a7170a | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 | https://10.0.131.183:2379 |
| b78e2856655bc2eb | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |  https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 | https://10.0.154.204:2379 |
+------------------+---------+------------------------------+---------------------------+---------------------------+
```

Take note of the ID and the name of the unhealthy etcd member, because these values are needed later in the procedure.

d. Remove the unhealthy etcd member by providing the ID to the **etcdctl member remove** command:

```
sh-4.2# etcdctl member remove 62bcf33650a7170a
```

Example output

```
Member 62bcf33650a7170a removed from cluster ead669ce1fbfb346
```

e. View the member list again and verify that the member was removed:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+------------------------------+---------------------------+---------------------------+
|        ID        | STATUS  |             NAME             |        PEER ADDRS         |       CLIENT ADDRS        |
+------------------+---------+------------------------------+---------------------------+---------------------------+
| b78e2856655bc2eb | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |  https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 | https://10.0.154.204:2379 |
+------------------+---------+------------------------------+---------------------------+---------------------------+
```

You can now exit the node shell.

3. Turn off the quorum guard by entering the following command:

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides": {"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
```

This command ensures that you can successfully re-create secrets and roll out the static pods.

4. Remove the old secrets for the unhealthy etcd member that was removed.

   a. List the secrets for the unhealthy etcd member that was removed.

   ```
   $ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal (1)
   ```

   (1) Pass in the name of the unhealthy etcd member that you took note of earlier in this procedure.

   There is a peer, serving, and metrics secret as shown in the following output:

   Example output

   ```
   etcd-peer-ip-10-0-131-183.ec2.internal          kubernetes.io/tls                2      47m
   etcd-serving-ip-10-0-131-183.ec2.internal        kubernetes.io/tls                2      47m
   etcd-serving-metrics-ip-10-0-131-183.ec2.internal   kubernetes.io/tls                2
   47m
   ```

   b. Delete the secrets for the unhealthy etcd member that was removed.

      i. Delete the peer secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
      ```

      ii. Delete the serving secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
      ```

      iii. Delete the metrics secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-
      183.ec2.internal
      ```

5. Force etcd redeployment.
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

   ```
   $ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "single-master-recovery-
   '"$( date --rfc-3339=ns )"'"}}' --type=merge (1)
   ```

   (1) The **forceRedeploymentReason** value must be unique, which is why a timestamp is appended.

   When the etcd cluster Operator performs a redeployment, it ensures that all control plane nodes have a functioning etcd pod.

6. Turn the quorum guard back on by entering the following command:

   ```
   $ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides": null}}'
   ```

7. You can verify that the **unsupportedConfigOverrides** section is removed from the object by entering this command:

```
$ oc get etcd/cluster -oyaml
```

8. If you are using single-node OpenShift, restart the node. Otherwise, you might encounter the following error in the etcd cluster Operator:

Example output

```
EtcdCertSignerControllerDegraded: [Operation cannot be fulfilled on secrets "etcd-peer-sno-0": the object has been modified; please apply your changes to the latest version and try again, Operation cannot be fulfilled on secrets "etcd-serving-sno-0": the object has been modified; please apply your changes to the latest version and try again, Operation cannot be fulfilled on secrets "etcd-serving-metrics-sno-0": the object has been modified; please apply your changes to the latest version and try again]
```

Verification

- Verify that the new member is available and healthy.

  a. Connect to the running etcd container again.
  In a terminal that has access to the cluster as a cluster-admin user, run the following command:

  ```
  $ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
  ```

  b. Verify that all members are healthy:

  ```
  sh-4.2# etcdctl endpoint health
  ```

  Example output

  ```
  https://10.0.131.183:2379 is healthy: successfully committed proposal: took = 16.671434ms
  https://10.0.154.204:2379 is healthy: successfully committed proposal: took = 16.698331ms
  https://10.0.164.97:2379 is healthy: successfully committed proposal: took = 16.621645ms
  ```

5.2.4.3. Replacing an unhealthy bare metal etcd member whose machine is not running or whose node is not ready

This procedure details the steps to replace a bare metal etcd member that is unhealthy either because the machine is not running or because the node is not ready.

If you are running installer-provisioned infrastructure or you used the Machine API to create your machines, follow these steps. Otherwise you must create the new control plane node using the same method that was used to originally create it.

Prerequisites

- You have identified the unhealthy bare metal etcd member.

- You have verified that either the machine is not running or the node is not ready.
- You have access to the cluster as a user with the **cluster-admin** role.
- You have taken an etcd backup.

IMPORTANT

You must take an etcd backup before performing this procedure so that your cluster can be restored if you encounter any issues.

Procedure

1. Verify and remove the unhealthy member.

   a. Choose a pod that is *not* on the affected node:
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

   ```
   $ oc get pods -n openshift-etcd -o wide | grep etcd | grep -v guard
   ```

   Example output

   ```
   etcd-openshift-control-plane-0   5/5   Running   11   3h56m   192.168.10.9   openshift-
   control-plane-0  <none>           <none>
   etcd-openshift-control-plane-1   5/5   Running   0    3h54m   192.168.10.10   openshift-
   control-plane-1   <none>           <none>
   etcd-openshift-control-plane-2   5/5   Running   0    3h58m   192.168.10.11   openshift-
   control-plane-2   <none>           <none>
   ```

   b. Connect to the running etcd container, passing in the name of a pod that is not on the affected node:
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

   ```
   $ oc rsh -n openshift-etcd etcd-openshift-control-plane-0
   ```

   c. View the member list:

   ```
   sh-4.2# etcdctl member list -w table
   ```

   Example output

   ```
   +------------------+---------+--------------------+---------------------------+--------------------------
   -+---------------------+
   | ID               | STATUS  | NAME                 | PEER ADDRS                | CLIENT
   ADDRS              | IS LEARNER |
   +------------------+---------+--------------------+---------------------------+--------------------------
   -+---------------------+
   | 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380/ |
   https://192.168.10.11:2379/ | false |
   | 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380/ |
   https://192.168.10.10:2379/ | false |
   ```

```
| cc3830a72fc357f9 | started | openshift-control-plane-0 | https://192.168.10.9:2380/ |
https://192.168.10.9:2379/   | false |
+------------------+---------+--------------------+---------------------------+---------------------------
-+---------------------+
```

Take note of the ID and the name of the unhealthy etcd member, because these values are required later in the procedure. The **etcdctl endpoint health** command will list the removed member until the replacement procedure is completed and the new member is added.

d. Remove the unhealthy etcd member by providing the ID to the **etcdctl member remove** command:

> **WARNING**
>
> Be sure to remove the correct etcd member; removing a good etcd member might lead to quorum loss.

```
sh-4.2# etcdctl member remove 7a8197040a5126c8
```

Example output

```
Member 7a8197040a5126c8 removed from cluster b23536c33f2cdd1b
```

e. View the member list again and verify that the member was removed:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+--------------------+---------------------------+---------------------------
-+-------------------------+
| ID               | STATUS  | NAME                      | PEER ADDRS                  | CLIENT
ADDRS                | IS LEARNER |
+------------------+---------+--------------------+---------------------------+---------------------------
-+-------------------------+
| 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380/ |
https://192.168.10.11:2379/ | false |
| 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380/ |
https://192.168.10.10:2379/ | false |
+------------------+---------+--------------------+---------------------------+---------------------------
-+-------------------------+
```

You can now exit the node shell.

> **IMPORTANT**
>
> After you remove the member, the cluster might be unreachable for a short time while the remaining etcd instances reboot.

2. Turn off the quorum guard by entering the following command:

   ```
   $ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides":
   {"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
   ```

   This command ensures that you can successfully re-create secrets and roll out the static pods.

3. Remove the old secrets for the unhealthy etcd member that was removed by running the following commands.

   a. List the secrets for the unhealthy etcd member that was removed.

      ```
      $ oc get secrets -n openshift-etcd | grep openshift-control-plane-2
      ```

      Pass in the name of the unhealthy etcd member that you took note of earlier in this procedure.

      There is a peer, serving, and metrics secret as shown in the following output:

      ```
      etcd-peer-openshift-control-plane-2            kubernetes.io/tls   2   134m
      etcd-serving-metrics-openshift-control-plane-2  kubernetes.io/tls   2   134m
      etcd-serving-openshift-control-plane-2         kubernetes.io/tls   2   134m
      ```

   b. Delete the secrets for the unhealthy etcd member that was removed.

      i. Delete the peer secret:

         ```
         $ oc delete secret etcd-peer-openshift-control-plane-2 -n openshift-etcd

         secret "etcd-peer-openshift-control-plane-2" deleted
         ```

      ii. Delete the serving secret:

         ```
         $ oc delete secret etcd-serving-metrics-openshift-control-plane-2 -n openshift-etcd

         secret "etcd-serving-metrics-openshift-control-plane-2" deleted
         ```

      iii. Delete the metrics secret:

         ```
         $ oc delete secret etcd-serving-openshift-control-plane-2 -n openshift-etcd

         secret "etcd-serving-openshift-control-plane-2" deleted
         ```

4. Delete the control plane machine.
   If you are running installer-provisioned infrastructure, or you used the Machine API to create your machines, follow these steps. Otherwise, you must create the new control plane node using the same method that was used to originally create it.

   a. Obtain the machine for the unhealthy member.
      In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

      ```
      $ oc get machines -n openshift-machine-api -o wide
      ```

Example output

```
NAME                          PHASE    TYPE   REGION   ZONE   AGE    NODE
PROVIDERID                                                              STATE
examplecluster-control-plane-0    Running                    3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned (1)
examplecluster-control-plane-1    Running                    3h11m   openshift-control-
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-control-plane-2    Running                    3h11m   openshift-control-
plane-2   baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-
61d8-410f-be5b-6a395b056135   externally provisioned
examplecluster-compute-0         Running                    165m    openshift-compute-0
baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-4bb3-80ec-
13a31858241f        provisioned
examplecluster-compute-1         Running                    165m    openshift-compute-1
baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-4241-91dc-
e7ea72ab13b9        provisioned
```

(1) This is the control plane machine for the unhealthy node, **examplecluster-control-plane-2**.

b. Save the machine configuration to a file on your file system:

```
$ oc get machine examplecluster-control-plane-2 \ (1)
    -n openshift-machine-api \
    -o yaml \
    > new-master-machine.yaml
```

(1) Specify the name of the control plane machine for the unhealthy node.

c. Edit the **new-master-machine.yaml** file that was created in the previous step to assign a new name and remove unnecessary fields.

i. Remove the entire **status** section:

```
status:
  addresses:
  - address: ""
    type: InternalIP
  - address: fe80::4adf:37ff:feb0:8aa1%ens1f1.373
    type: InternalDNS
  - address: fe80::4adf:37ff:feb0:8aa1%ens1f1.371
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Machine
    name: fe80::4adf:37ff:feb0:8aa1%ens1f1.372
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: machine.openshift.io/v1beta1
```

```
conditions:
- lastProbeTime: "2020-04-20T16:53:50Z"
  lastTransitionTime: "2020-04-20T16:53:50Z"
  message: machine successfully created
  reason: MachineCreationSucceeded
  status: "True"
  type: MachineCreation
instanceId: i-0fdb85790d76d0c3f
instanceState: stopped
kind: Machine
```

5. Change the **metadata.name** field to a new name.
   It is recommended to keep the same base name as the old machine and change the ending number to the next available number. In this example, **examplecluster-control-plane-2** is changed to **examplecluster-control-plane-3**.

   For example:

   ```
   apiVersion: machine.openshift.io/v1beta1
   kind: Machine
   metadata:
     ...
     name: examplecluster-control-plane-3
     ...
   ```

   a. Remove the **spec.providerID** field:

   ```
     providerID: baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-
   61d8-410f-be5b-6a395b056135
   ```

   b. Remove the **metadata.annotations** and **metadata.generation** fields:

   ```
     annotations:
       machine.openshift.io/instance-state: externally provisioned
     ...
     generation: 2
   ```

   c. Remove the **spec.conditions**, **spec.lastUpdated**, **spec.nodeRef** and **spec.phase** fields:

   ```
     lastTransitionTime: "2022-08-03T08:40:36Z"
   message: 'Drain operation currently blocked by: [{Name:EtcdQuorumOperator
   Owner:clusteroperator/etcd}]'
   reason: HookPresent
   severity: Warning
   status: "False"

   type: Drainable
   lastTransitionTime: "2022-08-03T08:39:55Z"
   status: "True"
   type: InstanceExists

   lastTransitionTime: "2022-08-03T08:36:37Z"
   status: "True"
   type: Terminable
   ```

```
lastUpdated: "2022-08-03T08:40:36Z"
nodeRef:
kind: Node
name: openshift-control-plane-2
uid: 788df282-6507-4ea2-9a43-24f237ccbc3c
phase: Running
```

6. Ensure that the Bare Metal Operator is available by running the following command:

```
$ oc get clusteroperator baremetal
```

Example output

```
NAME       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
baremetal  4.11.3    True        False         False      3d15h
```

7. Delete the machine of the unhealthy member using this command:

```
$ oc delete machine -n openshift-machine-api examplecluster-control-plane-2
```

If deletion of the machine is delayed for any reason or the command is obstructed and delayed, you can force deletion by removing the machine object finalizer field.

IMPORTANT

Do not interrupt machine deletion by pressing **Ctrl+c**. You must allow the command to proceed to completion. Open a new terminal window to edit and delete the finalizer fields.

```
$ oc edit machine -n openshift-machine-api examplecluster-control-plane-2
```

a. Find and delete the fields:

```
finalizers:
- machine.machine.openshift.io
```

Save your changes:

```
machine.machine.openshift.io/examplecluster-control-plane-2 edited
```

b. Verify the machine was deleted by running the following command:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output

```
NAME                            PHASE     TYPE   REGION   ZONE   AGE     NODE
PROVIDERID                                                               STATE
examplecluster-control-plane-0   Running                         3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned
examplecluster-control-plane-1   Running                         3h11m   openshift-control-
```

```
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-compute-0          Running                          165m    openshift-compute-0
baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-4bb3-80ec-
13a31858241f         provisioned
examplecluster-compute-1          Running                          165m    openshift-compute-1
baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-4241-91dc-
e7ea72ab13b9         provisioned
```

8. Remove the old **BareMetalHost** object with this command:

```
$ oc delete bmh openshift-control-plane-2 -n openshift-machine-api
```

Example output

```
baremetalhost.metal3.io "openshift-control-plane-2" deleted
```

After you remove the **BareMetalHost** and **Machine** objects, then the **Machine** controller automatically deletes the **Node** object.

If, after deletion of the **BareMetalHost** object, the machine node requires excessive time for deletion, the machine node can be deleted using:

```
$ oc delete node openshift-control-plane-2

node "openshift-control-plane-2" deleted
```

Verify the node has been deleted:

```
$ oc get nodes

NAME                    STATUS ROLES   AGE   VERSION
openshift-control-plane-0 Ready master 3h24m v1.25.0
openshift-control-plane-1 Ready master 3h24m v1.25.0
openshift-compute-0       Ready worker 176m v1.25.0
openshift-compute-1       Ready worker 176m v1.25.0
```

9. Create the new **BareMetalHost** object and the secret to store the BMC credentials:

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: openshift-control-plane-2-bmc-secret
  namespace: openshift-machine-api
data:
  password: <password>
  username: <username>
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-control-plane-2
  namespace: openshift-machine-api
spec:
  automatedCleaningMode: disabled
  bmc:
    address: redfish://10.46.61.18:443/redfish/v1/Systems/1
    credentialsName: openshift-control-plane-2-bmc-secret
    disableCertificateVerification: true
  bootMACAddress: 48:df:37:b0:8a:a0
  bootMode: UEFI
  externallyProvisioned: false
  online: true
  rootDeviceHints:
    deviceName: /dev/sda
  userData:
    name: master-user-data-managed
    namespace: openshift-machine-api
EOF
```

NOTE

The username and password can be found from the other bare metal host’s secrets. The protocol to use in **bmc:address** can be taken from other bmh objects.

IMPORTANT

If you reuse the **BareMetalHost** object definition from an existing control plane host, do not leave the **externallyProvisioned** field set to **true**.

Existing control plane **BareMetalHost** objects may have the **externallyProvisioned** flag set to **true** if they were provisioned by the OpenShift Container Platform installation program.

After the inspection is complete, the **BareMetalHost** object is created and available to be provisioned.

10. Verify the creation process using available **BareMetalHost** objects:

```
$ oc get bmh -n openshift-machine-api

NAME                      STATE                    CONSUMER                        ONLINE ERROR   AGE
openshift-control-plane-0 externally provisioned examplecluster-control-plane-0 true
4h48m
openshift-control-plane-1 externally provisioned examplecluster-control-plane-1 true
4h48m
openshift-control-plane-2 available              examplecluster-control-plane-3 true        47m
openshift-compute-0       provisioned            examplecluster-compute-0       true        4h48m
openshift-compute-1       provisioned            examplecluster-compute-1       true        4h48m
```

a. Create the new control plane machine using the **new-master-machine.yaml** file:

```
$ oc apply -f new-master-machine.yaml
```

b. Verify that the new machine has been created:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output

```
NAME                            PHASE     TYPE   REGION   ZONE   AGE    NODE
PROVIDERID                                                              STATE
examplecluster-control-plane-0        Running                           3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned (1)
examplecluster-control-plane-1        Running                           3h11m   openshift-control-
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-control-plane-2        Running                           3h11m   openshift-control-
plane-2   baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-
61d8-410f-be5b-6a395b056135   externally provisioned
examplecluster-compute-0              Running                           165m    openshift-compute-
0        baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-
4bb3-80ec-13a31858241f        provisioned
examplecluster-compute-1              Running                           165m    openshift-compute-
1        baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-
4241-91dc-e7ea72ab13b9        provisioned
```

1. The new machine, **clustername-8qw5l-master-3** is being created and is ready after the phase changes from **Provisioning** to **Running**.

It should take a few minutes for the new machine to be created. The etcd cluster Operator will automatically sync when the machine or node returns to a healthy state.

c. Verify that the bare metal host becomes provisioned and no error reported by running the following command:

```
$ oc get bmh -n openshift-machine-api
```

Example output

```
$ oc get bmh -n openshift-machine-api
NAME                      STATE                CONSUMER                          ONLINE ERROR AGE
openshift-control-plane-0 externally provisioned examplecluster-control-plane-0 true
4h48m
openshift-control-plane-1 externally provisioned examplecluster-control-plane-1 true
4h48m
openshift-control-plane-2 provisioned          examplecluster-control-plane-3 true
47m
openshift-compute-0       provisioned          examplecluster-compute-0       true
4h48m
openshift-compute-1       provisioned          examplecluster-compute-1       true
4h48m
```

d. Verify that the new node is added and in a ready state by running this command:

```
$ oc get nodes
```

Example output

```
$ oc get nodes
NAME                   STATUS ROLES   AGE   VERSION
openshift-control-plane-0 Ready master 4h26m v1.25.0
openshift-control-plane-1 Ready master 4h26m v1.25.0
openshift-control-plane-2 Ready master 12m   v1.25.0
openshift-compute-0       Ready worker 3h58m v1.25.0
openshift-compute-1       Ready worker 3h58m v1.25.0
```

11. Turn the quorum guard back on by entering the following command:

```
$ oc patch etcd/cluster --type=merge -p '\{"spec": {"unsupportedConfigOverrides": null}}
```

12. You can verify that the **unsupportedConfigOverrides** section is removed from the object by entering this command:

```
$ oc get etcd/cluster -oyaml
```

13. If you are using single-node OpenShift, restart the node. Otherwise, you might encounter the following error in the etcd cluster Operator:

Example output

```
EtcdCertSignerControllerDegraded: [Operation cannot be fulfilled on secrets "etcd-peer-sno-
0": the object has been modified; please apply your changes to the latest version and try
again, Operation cannot be fulfilled on secrets "etcd-serving-sno-0": the object has been
modified; please apply your changes to the latest version and try again, Operation cannot be
fulfilled on secrets "etcd-serving-metrics-sno-0": the object has been modified; please apply
your changes to the latest version and try again]
```

Verification

1. Verify that all etcd pods are running properly.
   In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc get pods -n openshift-etcd -o wide | grep etcd | grep -v guard
```

Example output

```
etcd-openshift-control-plane-0    5/5    Running    0    105m
etcd-openshift-control-plane-1    5/5    Running    0    107m
etcd-openshift-control-plane-2    5/5    Running    0    103m
```

If the output from the previous command only lists two pods, you can manually force an etcd redeployment. In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-
3339=ns )"'"}}' --type=merge (1)
```

(1) The **forceRedeploymentReason** value must be unique, which is why a timestamp is appended.

To verify there are exactly three etcd members, connect to the running etcd container, passing in the name of a pod that was not on the affected node. In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc rsh -n openshift-etcd etcd-openshift-control-plane-0
```

2. View the member list:

```
sh-4.2# etcdctl member list -w table
```

Example output

```
+------------------+---------+--------------------+---------------------------+---------------------------+---
--------------+
|       ID       | STATUS  |       NAME       |       PEER ADDRS       |       CLIENT ADDRS
|    IS LEARNER    |
+------------------+---------+--------------------+---------------------------+---------------------------+---
--------------+
| 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380 |
https://192.168.10.11:2379 |   false |
| 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380 |
https://192.168.10.10:2379 |   false |
| cc3830a72fc357f9 | started | openshift-control-plane-0 | https://192.168.10.9:2380 |
https://192.168.10.9:2379 |     false |
+------------------+---------+--------------------+---------------------------+---------------------------+---
--------------+
```

NOTE

If the output from the previous command lists more than three etcd members, you must carefully remove the unwanted member.

3. Verify that all etcd members are healthy by running the following command:

```
# etcdctl endpoint health --cluster
```

Example output

```
https://192.168.10.10:2379 is healthy: successfully committed proposal: took = 8.973065ms
https://192.168.10.9:2379 is healthy: successfully committed proposal: took = 11.559829ms
https://192.168.10.11:2379 is healthy: successfully committed proposal: took = 11.665203ms
```

4. Validate that all nodes are at the latest revision by running the following command:

```
$ oc get etcd -o=jsonpath='{range.items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

```
AllNodesAtLatestRevision
```

5.3. BACKING UP AND RESTORING ETCD ON A HOSTED CLUSTER

If you use hosted control planes on OpenShift Container Platform, the process to back up and restore etcd is different from the usual etcd backup process.

IMPORTANT

Hosted control planes is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

5.3.1. Taking a snapshot of etcd on a hosted cluster

As part of the process to back up etcd for a hosted cluster, you take a snapshot of etcd. After you take the snapshot, you can restore it, for example, as part of a disaster recovery operation.

IMPORTANT

This procedure requires API downtime.

Procedure

1. Pause reconciliation of the hosted cluster by entering this command:

```
$ oc patch -n clusters hostedclusters/${CLUSTER_NAME} -p '{"spec":
{"pausedUntil":"'${PAUSED_UNTIL}'"}}' --type=merge
```

2. Stop all etcd-writer deployments by entering this command:

```
$ oc scale deployment -n ${HOSTED_CLUSTER_NAMESPACE} --replicas=0 kube-
apiserver openshift-apiserver openshift-oauth-apiserver
```

3. Take an etcd snapshot by using the **exec** command in each etcd container:

```
$ oc exec -it etcd-0 -n ${HOSTED_CLUSTER_NAMESPACE} -- env ETCDCTL_API=3
/usr/bin/etcdctl --cacert /etc/etcd/tls/client/etcd-client-ca.crt --cert /etc/etcd/tls/client/etcd-
client.crt --key /etc/etcd/tls/client/etcd-client.key --endpoints=localhost:2379 snapshot save
/var/lib/data/snapshot.db
$ oc exec -it etcd-0 -n ${HOSTED_CLUSTER_NAMESPACE} -- env ETCDCTL_API=3
/usr/bin/etcdctl -w table snapshot status /var/lib/data/snapshot.db
```

4. Copy the snapshot data to a location where you can retrieve it later, such as an S3 bucket, as shown in the following example.

NOTE

The following example uses signature version 2. If you are in a region that supports signature version 4, such as the us-east-2 region, use signature version 4. Otherwise, if you use signature version 2 to copy the snapshot to an S3 bucket, the upload fails and signature version 2 is deprecated.

Example

```
BUCKET_NAME=somebucket
FILEPATH="/${BUCKET_NAME}/${CLUSTER_NAME}-snapshot.db"
CONTENT_TYPE="application/x-compressed-tar"
DATE_VALUE=`date -R`
SIGNATURE_STRING="PUT\n\n${CONTENT_TYPE}\n${DATE_VALUE}\n${FILEPATH}"
ACCESS_KEY=accesskey
SECRET_KEY=secret
SIGNATURE_HASH=`echo -en ${SIGNATURE_STRING} | openssl sha1 -hmac
${SECRET_KEY} -binary | base64`

oc exec -it etcd-0 -n ${HOSTED_CLUSTER_NAMESPACE} -- curl -X PUT -T
"/var/lib/data/snapshot.db" \
  -H "Host: ${BUCKET_NAME}.s3.amazonaws.com" \
  -H "Date: ${DATE_VALUE}" \
  -H "Content-Type: ${CONTENT_TYPE}" \
  -H "Authorization: AWS ${ACCESS_KEY}:${SIGNATURE_HASH}" \
  https://${BUCKET_NAME}.s3.amazonaws.com/${CLUSTER_NAME}-snapshot.db
```

5. If you want to be able to restore the snapshot on a new cluster later, save the encryption secret that the hosted cluster references, as shown in this example:

Example

```
oc get hostedcluster $CLUSTER_NAME -o=jsonpath='{.spec.secretEncryption.aescbc}'
{"activeKey":{"name":"CLUSTER_NAME-etcd-encryption-key"}}

# Save this secret, or the key it contains so the etcd data can later be decrypted
oc get secret ${CLUSTER_NAME}-etcd-encryption-key -o=jsonpath='{.data.key}'
```

Next steps

Restore the etcd snapshot.

5.3.2. Restoring an etcd snapshot on a hosted cluster

If you have a snapshot of etcd from your hosted cluster, you can restore it. Currently, you can restore an etcd snapshot only during cluster creation.

To restore an etcd snapshot, you modify the output from the **create cluster --render** command and define a **restoreSnapshotURL** value in the etcd section of the **HostedCluster** specification.

Prerequisites

You took an etcd snapshot on a hosted cluster.

Procedure

1. On the **aws** command-line interface (CLI), create a pre-signed URL so that you can download your etcd snapshot from S3 without passing credentials to the etcd deployment:

```
ETCD_SNAPSHOT=${ETCD_SNAPSHOT:-"s3://${BUCKET_NAME}/${CLUSTER_NAME}-
snapshot.db"}
ETCD_SNAPSHOT_URL=$(aws s3 presign ${ETCD_SNAPSHOT})
```

2. Modify the **HostedCluster** specification to refer to the URL:

```
spec:
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 4Gi
        type: PersistentVolume
        restoreSnapshotURL:
        - "${ETCD_SNAPSHOT_URL}"
    managementType: Managed
```

3. Ensure that the secret that you referenced from the **spec.secretEncryption.aescbc** value contains the same AES key that you saved in the previous steps.

5.3.3. Additional resources

- Disaster recovery for a hosted cluster within an AWS region

5.4. DISASTER RECOVERY

5.4.1. About disaster recovery

The disaster recovery documentation provides information for administrators on how to recover from several disaster situations that might occur with their OpenShift Container Platform cluster. As an administrator, you might need to follow one or more of the following procedures to return your cluster to a working state.

IMPORTANT

Disaster recovery requires you to have at least one healthy control plane host.

Restoring to a previous cluster state

This solution handles situations where you want to restore your cluster to a previous state, for example, if an administrator deletes something critical. This also includes situations where you have lost the majority of your control plane hosts, leading to etcd quorum loss and the cluster going offline. As long as you have taken an etcd backup, you can follow this procedure to restore your cluster to a previous state.
If applicable, you might also need to recover from expired control plane certificates .

WARNING

Restoring to a previous cluster state is a destructive and destablizing action to take on a running cluster. This procedure should only be used as a last resort.

Prior to performing a restore, see About restoring cluster state for more information on the impact to the cluster.

NOTE

If you have a majority of your masters still available and have an etcd quorum, then follow the procedure to replace a single unhealthy etcd member.

Recovering from expired control plane certificates
: This solution handles situations where your control plane certificates have expired. For example, if you shut down your cluster before the first certificate rotation, which occurs 24 hours after installation, your certificates will not be rotated and will expire. You can follow this procedure to recover from expired control plane certificates.

5.4.2. Restoring to a previous cluster state

To restore the cluster to a previous state, you must have previously backed up etcd data by creating a snapshot. You will use this snapshot to restore the cluster state.

5.4.2.1. About restoring cluster state

You can use an etcd backup to restore your cluster to a previous state. This can be used to recover from the following situations:

- The cluster has lost the majority of control plane hosts (quorum loss).
- An administrator has deleted something critical and must restore to recover the cluster.

WARNING

Restoring to a previous cluster state is a destructive and destablizing action to take on a running cluster. This should only be used as a last resort.

If you are able to retrieve data using the Kubernetes API server, then etcd is available and you should not restore using an etcd backup.

Restoring etcd effectively takes a cluster back in time and all clients will experience a conflicting, parallel history. This can impact the behavior of watching components like kubelets, Kubernetes controller managers, SDN controllers, and persistent volume controllers.

It can cause Operator churn when the content in etcd does not match the actual content on disk, causing Operators for the Kubernetes API server, Kubernetes controller manager, Kubernetes scheduler, and etcd to get stuck when files on disk conflict with content in etcd. This can require manual actions to resolve the issues.

In extreme cases, the cluster can lose track of persistent volumes, delete critical workloads that no longer exist, reimage machines, and rewrite CA bundles with expired certificates.

5.4.2.2. Restoring to a previous cluster state

You can use a saved etcd backup to restore a previous cluster state or restore a cluster that has lost the majority of control plane hosts.

NOTE

If your cluster uses a control plane machine set, see "Troubleshooting the control plane machine set" for a more simple etcd recovery procedure.

IMPORTANT

When you restore your cluster, you must use an etcd backup that was taken from the same z-stream release. For example, an OpenShift Container Platform 4.7.2 cluster must use an etcd backup that was taken from 4.7.2.

Prerequisites

- Access to the cluster as a user with the **cluster-admin** role.
- A healthy control plane host to use as the recovery host.
- SSH access to control plane hosts.
- A backup directory containing both the etcd snapshot and the resources for the static pods, which were from the same backup. The file names in the directory must be in the following formats: **snapshot_<datetimestamp>.db** and **static_kuberesources_<datetimestamp>.tar.gz**.

IMPORTANT

For non-recovery control plane nodes, it is not required to establish SSH connectivity or to stop the static pods. You can delete and recreate other non-recovery, control plane machines, one by one.

Procedure

1. Select a control plane host to use as the recovery host. This is the host that you will run the restore operation on.
2. Establish SSH connectivity to each of the control plane nodes, including the recovery host.
   The Kubernetes API server becomes inaccessible after the restore process starts, so you cannot access the control plane nodes. For this reason, it is recommended to establish SSH connectivity to each control plane host in a separate terminal.

   IMPORTANT

   If you do not complete this step, you will not be able to access the control plane hosts to complete the restore procedure, and you will be unable to recover your cluster from this state.

3. Copy the etcd backup directory to the recovery control plane host.
   This procedure assumes that you copied the **backup** directory containing the etcd snapshot and the resources for the static pods to the /**home**/**core**/ directory of your recovery control plane host.
4. Stop the static pods on any other control plane nodes.

> **NOTE**
>
> It is not required to manually stop the pods on the recovery host. The recovery script will stop the pods on the recovery host.

a. Access a control plane host that is not the recovery host.

b. Move the existing etcd pod file out of the kubelet manifest directory:

```
$ sudo mv /etc/kubernetes/manifests/etcd-pod.yaml /tmp
```

c. Verify that the etcd pods are stopped.

```
$ sudo crictl ps | grep etcd | grep -v operator
```

The output of this command should be empty. If it is not empty, wait a few minutes and check again.

d. Move the existing Kubernetes API server pod file out of the kubelet manifest directory:

```
$ sudo mv /etc/kubernetes/manifests/kube-apiserver-pod.yaml /tmp
```

e. Verify that the Kubernetes API server pods are stopped.

```
$ sudo crictl ps | grep kube-apiserver | grep -v operator
```

The output of this command should be empty. If it is not empty, wait a few minutes and check again.

f. Move the etcd data directory to a different location:

```
$ sudo mv /var/lib/etcd/ /tmp
```

g. Repeat this step on each of the other control plane hosts that is not the recovery host.

5. Access the recovery control plane host.

6. If the cluster-wide proxy is enabled, be sure that you have exported the **NO_PROXY**, **HTTP_PROXY**, and **HTTPS_PROXY** environment variables.

   TIP

   You can check whether the proxy is enabled by reviewing the output of **oc get proxy cluster -o yaml**. The proxy is enabled if the **httpProxy**, **httpsProxy**, and **noProxy** fields have values set.

7. Run the restore script on the recovery control plane host and pass in the path to the etcd backup directory:

```
$ sudo -E /usr/local/bin/cluster-restore.sh /home/core/backup
```

Example script output

```
...stopping kube-scheduler-pod.yaml
```

```
...stopping kube-controller-manager-pod.yaml
...stopping etcd-pod.yaml
...stopping kube-apiserver-pod.yaml
Waiting for container etcd to stop
.complete
Waiting for container etcdctl to stop
.............................complete
Waiting for container etcd-metrics to stop
complete
Waiting for container kube-controller-manager to stop
complete
Waiting for container kube-apiserver to stop
..........................................................................................complete
Waiting for container kube-scheduler to stop
complete
Moving etcd data-dir /var/lib/etcd/member to /var/lib/etcd-backup
starting restore-etcd static pod
starting kube-apiserver-pod.yaml
static-pod-resources/kube-apiserver-pod-7/kube-apiserver-pod.yaml
starting kube-controller-manager-pod.yaml
static-pod-resources/kube-controller-manager-pod-7/kube-controller-manager-pod.yaml
starting kube-scheduler-pod.yaml
static-pod-resources/kube-scheduler-pod-8/kube-scheduler-pod.yaml
```

NOTE

The restore process can cause nodes to enter the **NotReady** state if the node certificates were updated after the last etcd backup.

8. Check the nodes to ensure they are in the **Ready** state.

   a. Run the following command:

   ```
   $ oc get nodes -w
   ```

   Sample output

   ```
   NAME                STATUS  ROLES          AGE     VERSION
   host-172-25-75-28   Ready   master         3d20h   v1.25.0
   host-172-25-75-38   Ready   infra,worker   3d20h   v1.25.0
   host-172-25-75-40   Ready   master         3d20h   v1.25.0
   host-172-25-75-65   Ready   master         3d20h   v1.25.0
   host-172-25-75-74   Ready   infra,worker   3d20h   v1.25.0
   host-172-25-75-79   Ready   worker         3d20h   v1.25.0
   host-172-25-75-86   Ready   worker         3d20h   v1.25.0
   host-172-25-75-98   Ready   infra,worker   3d20h   v1.25.0
   ```

   It can take several minutes for all nodes to report their state.

   b. If any nodes are in the **NotReady** state, log in to the nodes and remove all of the PEM files from the **/var/lib/kubelet/pki** directory on each node. You can SSH into the nodes or use the terminal window in the web console.

   ```
   $  ssh -i <ssh-key-path> core@<master-hostname>
   ```

Sample pki directory

```
sh-4.4# pwd
/var/lib/kubelet/pki
sh-4.4# ls
kubelet-client-2022-04-28-11-24-09.pem  kubelet-server-2022-04-28-11-24-15.pem
kubelet-client-current.pem              kubelet-server-current.pem
```

9. Restart the kubelet service on all control plane hosts.

   a. From the recovery host, run the following command:

   ```
   $ sudo systemctl restart kubelet.service
   ```

   b. Repeat this step on all other control plane hosts.

10. Approve the pending CSRs:

    a. Get the list of current CSRs:

    ```
    $ oc get csr
    ```

    Example output

    ```
    NAME        AGE    SIGNERNAME                                    REQUESTOR
    CONDITION
    csr-2s94x   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
    Pending 1
    csr-4bd6t   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
    Pending 2
    csr-4hl85   13m    kubernetes.io/kube-apiserver-client-kubelet
    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
    3
    csr-zhhhp   3m8s   kubernetes.io/kube-apiserver-client-kubelet
    system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
    4
    ...
    ```

    1 2 A pending kubelet service CSR (for user-provisioned installations).

    3 4 A pending **node-bootstrapper** CSR.

    b. Review the details of a CSR to verify that it is valid:

    ```
    $ oc describe csr <csr_name> 1
    ```

    1 **<csr_name>** is the name of a CSR from the list of current CSRs.

    c. Approve each valid **node-bootstrapper** CSR:

    ```
    $ oc adm certificate approve <csr_name>
    ```

d. For user-provisioned installations, approve each valid kubelet service CSR:

```
$ oc adm certificate approve <csr_name>
```

11. Verify that the single member control plane has started successfully.

a. From the recovery host, verify that the etcd container is running.

```
$ sudo crictl ps | grep etcd | grep -v operator
```

Example output

```
3ad41b7908e32
36f86e2eeaaffe662df0d21041eb22b8198e0e58abeeae8c743c3e6e977e8009
About a minute ago   Running           etcd                                    0
7c05f8af362f0
```

b. From the recovery host, verify that the etcd pod is running.

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

NOTE

If you attempt to run **oc login** prior to running this command and receive the following error, wait a few moments for the authentication controllers to start and try again.

```
Unable to connect to the server: EOF
```

Example output

```
NAME                                        READY   STATUS      RESTARTS   AGE
etcd-ip-10-0-143-125.ec2.internal                1/1     Running     1          2m47s
```

If the status is **Pending**, or the output lists more than one running etcd pod, wait a few minutes and check again.

c. Repeat this step for each lost control plane host that is not the recovery host.

NOTE

Perform the following step only if you are using **OVNKubernetes** network plugin.

12. Restart the Open Virtual Network (OVN) Kubernetes pods on all the hosts.

a. Remove the northbound database (nbdb) and southbound database (sbdb). Access the recovery host and the remaining control plane nodes by using Secure Shell (SSH) and run the following command:

```
$ sudo rm -f /var/lib/ovn/etc/*.db
```

b. Delete all OVN-Kubernetes control plane pods by running the following command:

```
$ oc delete pods -l app=ovnkube-master -n openshift-ovn-kubernetes
```

c. Ensure that all the OVN-Kubernetes control plane pods are deployed again and are in a **Running** state by running the following command:

```
$ oc get pods -l app=ovnkube-master -n openshift-ovn-kubernetes
```

Example output

```
NAME                   READY   STATUS    RESTARTS   AGE
ovnkube-master-nb24h   4/4     Running   0          48s
ovnkube-master-rm8kw   4/4     Running   0          47s
ovnkube-master-zbqnh   4/4     Running   0          56s
```

d. Delete all **ovnkube-node** pods by running the following command:

```
$ oc get pods -n openshift-ovn-kubernetes -o name | grep ovnkube-node | while read p ;
do oc delete $p -n openshift-ovn-kubernetes ; done
```

e. Ensure that all the **ovnkube-node** pods are deployed again and are in a **Running** state by running the following command:

```
$ oc get  pods -n openshift-ovn-kubernetes | grep ovnkube-node
```

13. Delete and recreate other non-recovery, control plane machines, one by one. After these machines are recreated, a new revision is forced and etcd scales up automatically.

WARNING

For bare metal installations on installer-provisioned infrastructure, control plane machines are not recreated. For more information, see "Replacing a bare-metal control plane node".

If you are running installer-provisioned infrastructure, or you used the Machine API to create your machines, follow these steps. Otherwise, you must create the new control plane node using the same method that was used to originally create it.

WARNING

Do not delete and recreate the machine for the recovery host.

a. Obtain the machine for one of the lost control plane hosts.
In a terminal that has access to the cluster as a cluster-admin user, run the following command:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output:

```
NAME                                        PHASE     TYPE        REGION      ZONE         AGE
NODE                          PROVIDERID                                 STATE
clustername-8qw5l-master-0                  Running   m4.xlarge   us-east-1   us-east-1a
3h37m   ip-10-0-131-183.ec2.internal   aws:///us-east-1a/i-0ec2782f8287dfb7e   stopped
(1)
clustername-8qw5l-master-1                  Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2                  Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba  running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large    us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large    us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large    us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

1 This is the control plane machine for the lost control plane host, **ip-10-0-131-183.ec2.internal**.

b. Save the machine configuration to a file on your file system:

```
$ oc get machine clustername-8qw5l-master-0 \ (1)
    -n openshift-machine-api \
    -o yaml \
    > new-master-machine.yaml
```

1 Specify the name of the control plane machine for the lost control plane host.

c. Edit the **new-master-machine.yaml** file that was created in the previous step to assign a new name and remove unnecessary fields.

i. Remove the entire **status** section:

```
status:
  addresses:
  - address: 10.0.131.183
    type: InternalIP
  - address: ip-10-0-131-183.ec2.internal
    type: InternalDNS
  - address: ip-10-0-131-183.ec2.internal
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Node
    name: ip-10-0-131-183.ec2.internal
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2020-04-20T16:53:50Z"
      lastTransitionTime: "2020-04-20T16:53:50Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: AWSMachineProviderStatus
```

ii. Change the **metadata.name** field to a new name.
It is recommended to keep the same base name as the old machine and change the ending number to the next available number. In this example, **clustername-8qw5l-master-0** is changed to **clustername-8qw5l-master-3**:

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

iii. Remove the **spec.providerID** field:

```
providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

iv. Remove the **metadata.annotations** and **metadata.generation** fields:

```
annotations:
  machine.openshift.io/instance-state: running
...
generation: 2
```

v. Remove the **metadata.resourceVersion** and **metadata.uid** fields:

```
resourceVersion: "13291"
uid: a282eb70-40a2-4e89-8009-d05dd420d31a
```

d. Delete the machine of the lost control plane host:

```
$ oc delete machine -n openshift-machine-api clustername-8qw5l-master-0 1
```

1 Specify the name of the control plane machine for the lost control plane host.

e. Verify that the machine was deleted:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output:

```
NAME                                PHASE    TYPE      REGION     ZONE        AGE
NODE                       PROVIDERID                          STATE
clustername-8qw5l-master-1               Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2               Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba  running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

f. Create the new machine using the **new-master-machine.yaml** file:

```
$ oc apply -f new-master-machine.yaml
```

g. Verify that the new machine has been created:

```
$ oc get machines -n openshift-machine-api -o wide
```

Example output:

```
NAME                                PHASE       TYPE      REGION     ZONE        AGE
NODE                       PROVIDERID                          STATE
clustername-8qw5l-master-1               Running       m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2               Running       m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba  running
clustername-8qw5l-master-3               Provisioning   m4.xlarge   us-east-1   us-east-1a
85s     ip-10-0-173-171.ec2.internal    aws:///us-east-1a/i-015b0888fe17bc2c8  running
(1)
clustername-8qw5l-worker-us-east-1a-wbtgd   Running       m4.large    us-east-1   us-
east-1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running       m4.large    us-east-1   us-east-
1b   3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b
running
clustername-8qw5l-worker-us-east-1c-pkg26   Running       m4.large    us-east-1   us-
east-1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

(1) The new machine, **clustername-8qw5l-master-3** is being created and is ready after the phase changes from **Provisioning** to **Running**.

It might take a few minutes for the new machine to be created. The etcd cluster Operator will automatically sync when the machine or node returns to a healthy state.

h. Repeat these steps for each lost control plane host that is not the recovery host.

14. In a separate terminal window, log in to the cluster as a user with the **cluster-admin** role by using the following command:

```
$ oc login -u <cluster_admin> (1)
```

(1) For **<cluster_admin>**, specify a user name with the **cluster-admin** role.

15. Force etcd redeployment.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-
3339=ns )"'"}}' --type=merge (1)
```

(1) The **forceRedeploymentReason** value must be unique, which is why a timestamp is appended.

When the etcd cluster Operator performs a redeployment, the existing nodes are started with new pods similar to the initial bootstrap scale up.

16. Verify all nodes are updated to the latest revision.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

Review the **NodeInstallerProgressing** status condition for etcd to verify that all nodes are at the latest revision. The output shows **AllNodesAtLatestRevision** upon successful update:

```
AllNodesAtLatestRevision
3 nodes are at revision 7 (1)
```

(1) In this example, the latest revision number is **7**.

If the output includes multiple revision numbers, such as **2 nodes are at revision 6; 1 nodes are at revision 7**, this means that the update is still in progress. Wait a few minutes and try again.

17. After etcd is redeployed, force new rollouts for the control plane. The Kubernetes API server will reinstall itself on the other nodes because the kubelet is connected to API servers using an internal load balancer.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following commands.

a. Force a new rollout for the Kubernetes API server:

```
$ oc patch kubeapiserver cluster -p='{"spec": {"forceRedeploymentReason": "recovery-
'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

Verify all nodes are updated to the latest revision.

```
$ oc get kubeapiserver -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

Review the **NodeInstallerProgressing** status condition to verify that all nodes are at the latest revision. The output shows **AllNodesAtLatestRevision** upon successful update:

```
AllNodesAtLatestRevision
3 nodes are at revision 7 1
```

1 In this example, the latest revision number is **7**.

If the output includes multiple revision numbers, such as **2 nodes are at revision 6; 1 nodes are at revision 7**, this means that the update is still in progress. Wait a few minutes and try again.

b. Force a new rollout for the Kubernetes controller manager:

```
$ oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

Verify all nodes are updated to the latest revision.

```
$ oc get kubecontrollermanager -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

Review the **NodeInstallerProgressing** status condition to verify that all nodes are at the latest revision. The output shows **AllNodesAtLatestRevision** upon successful update:

```
AllNodesAtLatestRevision
3 nodes are at revision 7 1
```

1 In this example, the latest revision number is **7**.

If the output includes multiple revision numbers, such as **2 nodes are at revision 6; 1 nodes are at revision 7**, this means that the update is still in progress. Wait a few minutes and try again.

c. Force a new rollout for the Kubernetes scheduler:

```
$ oc patch kubescheduler cluster -p='{"spec": {"forceRedeploymentReason": "recovery-
'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

Verify all nodes are updated to the latest revision.

```
$ oc get kubescheduler -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

Review the **NodeInstallerProgressing** status condition to verify that all nodes are at the latest revision. The output shows **AllNodesAtLatestRevision** upon successful update:

```
AllNodesAtLatestRevision
3 nodes are at revision 7 1
```

1 In this example, the latest revision number is **7**.

If the output includes multiple revision numbers, such as **2 nodes are at revision 6; 1 nodes are at revision 7**, this means that the update is still in progress. Wait a few minutes and try again.

18. Verify that all control plane hosts have started and joined the cluster.
In a terminal that has access to the cluster as a **cluster-admin** user, run the following command:

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

Example output

```
etcd-ip-10-0-143-125.ec2.internal         2/2   Running   0      9h
etcd-ip-10-0-154-194.ec2.internal         2/2   Running   0      9h
etcd-ip-10-0-173-171.ec2.internal         2/2   Running   0      9h
```

To ensure that all workloads return to normal operation following a recovery procedure, restart each pod that stores Kubernetes API information. This includes OpenShift Container Platform components such as routers, Operators, and third-party components.

Note that it might take several minutes after completing this procedure for all services to be restored. For example, authentication by using **oc login** might not immediately work until the OAuth server pods are restarted.

5.4.2.3. Issues and workarounds for restoring a persistent storage state

If your OpenShift Container Platform cluster uses persistent storage of any form, a state of the cluster is typically stored outside etcd. It might be an Elasticsearch cluster running in a pod or a database running in a **StatefulSet** object. When you restore from an etcd backup, the status of the workloads in OpenShift Container Platform is also restored. However, if the etcd snapshot is old, the status might be invalid or outdated.

IMPORTANT

The contents of persistent volumes (PVs) are never part of the etcd snapshot. When you restore an OpenShift Container Platform cluster from an etcd snapshot, non-critical workloads might gain access to critical data, or vice-versa.

The following are some example scenarios that produce an out-of-date status:

- MySQL database is running in a pod backed up by a PV object. Restoring OpenShift Container Platform from an etcd snapshot does not bring back the volume on the storage provider, and does not produce a running MySQL pod, despite the pod repeatedly attempting to start. You must manually restore this pod by restoring the volume on the storage provider, and then editing the PV to point to the new volume.

- Pod P1 is using volume A, which is attached to node X. If the etcd snapshot is taken while another pod uses the same volume on node Y, then when the etcd restore is performed, pod P1 might not be able to start correctly due to the volume still being attached to node Y. OpenShift

Container Platform is not aware of the attachment, and does not automatically detach it. When this occurs, the volume must be manually detached from node Y so that the volume can attach on node X, and then pod P1 can start.

- Cloud provider or storage provider credentials were updated after the etcd snapshot was taken. This causes any CSI drivers or Operators that depend on the those credentials to not work. You might have to manually update the credentials required by those drivers or Operators.

- A device is removed or renamed from OpenShift Container Platform nodes after the etcd snapshot is taken. The Local Storage Operator creates symlinks for each PV that it manages from **/dev/disk/by-id** or **/dev** directories. This situation might cause the local PVs to refer to devices that no longer exist.
  To fix this problem, an administrator must:

  1. Manually remove the PVs with invalid devices.

  2. Remove symlinks from respective nodes.

  3. Delete **LocalVolume** or **LocalVolumeSet** objects (see *Storage* → *Configuring persistent storage* → *Persistent storage using local volumes* → *Deleting the Local Storage Operator Resources*).

5.4.2.4. Additional resources

- Creating a bastion host to access OpenShift Container Platform instances and the control plane nodes with SSH.

- Replacing a bare-metal control plane node .

5.4.3. Recovering from expired control plane certificates

5.4.3.1. Recovering from expired control plane certificates

The cluster can automatically recover from expired control plane certificates.

However, you must manually approve the pending **node-bootstrapper** certificate signing requests (CSRs) to recover kubelet certificates. For user-provisioned installations, you might also need to approve pending kubelet serving CSRs.

Use the following steps to approve the pending CSRs:

Procedure

1. Get the list of current CSRs:

```
$ oc get csr
```

Example output

```
NAME        AGE    SIGNERNAME                                  REQUESTOR
CONDITION
csr-2s94x   8m3s   kubernetes.io/kubelet-serving               system:node:<node_name>
Pending (1)
csr-4bd6t   8m3s   kubernetes.io/kubelet-serving               system:node:<node_name>
```

```
Pending (2)
csr-4hl85   13m     kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending (3)
csr-zhhhp   3m8s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending (4)
...
```

(1) (2) A pending kubelet service CSR (for user-provisioned installations).

(3) (4) A pending **node-bootstrapper** CSR.

2. Review the details of a CSR to verify that it is valid:

```
$ oc describe csr <csr_name> (1)
```

(1) **<csr_name>** is the name of a CSR from the list of current CSRs.

3. Approve each valid **node-bootstrapper** CSR:

```
$ oc adm certificate approve <csr_name>
```

4. For user-provisioned installations, approve each valid kubelet serving CSR:

```
$ oc adm certificate approve <csr_name>
```

5.4.4. Disaster recovery for a hosted cluster within an AWS region

In a situation where you need disaster recovery (DR) for a hosted cluster, you can recover a hosted cluster to the same region within AWS. For example, you need DR when the upgrade of a management cluster fails and the hosted cluster is in a read-only state.

IMPORTANT

Hosted control planes is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview/.

The DR process involves three main steps:

1. Backing up the hosted cluster on the source management cluster
2. Restoring the hosted cluster on a destination management cluster
3. Deleting the hosted cluster from the source management cluster

Your workloads remain running during the process. The Cluster API might be unavailable for a period, but that will not affect the services that are running on the worker nodes.

IMPORTANT

Both the source management cluster and the destination management cluster must have the **--external-dns** flags to maintain the API server URL, as shown in this example:

Example: External DNS flags

```
--external-dns-provider=aws \
--external-dns-credentials=<AWS Credentials location> \
--external-dns-domain-filter=<DNS Base Domain>
```

That way, the server URL ends with **https://api-sample-hosted.sample-hosted.aws.openshift.com**.

If you do not include the **--external-dns** flags to maintain the API server URL, the hosted cluster cannot be migrated.

5.4.4.1. Example environment and context

Consider an scenario where you have three clusters to restore. Two are management clusters, and one is a hosted cluster. You can restore either the control plane only or the control plane and the nodes. Before you begin, you need the following information:

- Source MGMT Namespace: The source management namespace
- Source MGMT ClusterName: The source management cluster name
- Source MGMT Kubeconfig: The source management **kubeconfig** file
- Destination MGMT Kubeconfig: The destination management **kubeconfig** file
- HC Kubeconfig: The hosted cluster **kubeconfig** file
- SSH key file: The SSH public key
- Pull secret: The pull secret file to access the release images
- AWS credentials
- AWS region
- Base domain: The DNS base domain to use as an external DNS
- S3 bucket name: The bucket in the AWS region where you plan to upload the etcd backup

This information is shown in the following example environment variables.

Example environment variables

```
SSH_KEY_FILE=${HOME}/.ssh/id_rsa.pub
BASE_PATH=${HOME}/hypershift
BASE_DOMAIN="aws.sample.com"
PULL_SECRET_FILE="${HOME}/pull_secret.json"
AWS_CREDS="${HOME}/.aws/credentials"
AWS_ZONE_ID="Z02718293M33QHDEQBROL"
```

```
CONTROL_PLANE_AVAILABILITY_POLICY=SingleReplica
HYPERSHIFT_PATH=${BASE_PATH}/src/hypershift
HYPERSHIFT_CLI=${HYPERSHIFT_PATH}/bin/hypershift
HYPERSHIFT_IMAGE=${HYPERSHIFT_IMAGE:-"quay.io/${USER}/hypershift:latest"}
NODE_POOL_REPLICAS=${NODE_POOL_REPLICAS:-2}

# MGMT Context
MGMT_REGION=us-west-1
MGMT_CLUSTER_NAME="${USER}-dev"
MGMT_CLUSTER_NS=${USER}
MGMT_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${MGMT_CLUSTER_NS}-
${MGMT_CLUSTER_NAME}"
MGMT_KUBECONFIG="${MGMT_CLUSTER_DIR}/kubeconfig"

# MGMT2 Context
MGMT2_CLUSTER_NAME="${USER}-dest"
MGMT2_CLUSTER_NS=${USER}
MGMT2_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${MGMT2_CLUSTER_NS}-
${MGMT2_CLUSTER_NAME}"
MGMT2_KUBECONFIG="${MGMT2_CLUSTER_DIR}/kubeconfig"

# Hosted Cluster Context
HC_CLUSTER_NS=clusters
HC_REGION=us-west-1
HC_CLUSTER_NAME="${USER}-hosted"
HC_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}"
HC_KUBECONFIG="${HC_CLUSTER_DIR}/kubeconfig"
BACKUP_DIR=${HC_CLUSTER_DIR}/backup

BUCKET_NAME="${USER}-hosted-${MGMT_REGION}"

# DNS
AWS_ZONE_ID="Z07342811SH9AA102K1AC"
EXTERNAL_DNS_DOMAIN="hc.jpdv.aws.kerbeross.com"
```

5.4.4.2. Overview of the backup and restore process

The backup and restore process works as follows:

1. On management cluster 1, which you can think of as the source management cluster, the control plane and workers interact by using the ExternalDNS API.

2. You take a snapshot of the hosted cluster, which includes etcd, the control plane, and the worker nodes. The worker nodes are moved to the external DNS, the control plane is saved in a local manifest file, and etcd is backed up to an S3 bucket.

3. On management cluster 2, which you can think of as the destination management cluster, you restore etcd from the S3 bucket and restore the control plane from the local manifest file.

4. By using the External DNS API, the worker nodes are restored to management cluster 2.

5. On management cluster 2, the control plane and worker nodes interact by using the ExternalDNS API.

You can manually back up and restore your hosted cluster, or you can run a script to complete the process. For more information about the script, see "Running a script to back up and restore a hosted cluster".

5.4.4.3. Backing up a hosted cluster

To recover your hosted cluster in your target management cluster, you first need to back up all of the relevant data.

Procedure

1. Create a configmap file to declare the source management cluster by entering this command:

```
$ oc create configmap mgmt-parent-cluster -n default --from-
literal=from=${MGMT_CLUSTER_NAME}
```

2. Shut down the reconciliation in the hosted cluster and in the node pools by entering these commands:

```
PAUSED_UNTIL="true"
oc patch -n ${HC_CLUSTER_NS} hostedclusters/${HC_CLUSTER_NAME} -p '{"spec":
{"pausedUntil":"'${PAUSED_UNTIL}'"}}' --type=merge
oc scale deployment -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --replicas=0 kube-
apiserver openshift-apiserver openshift-oauth-apiserver control-plane-operator
```

```
PAUSED_UNTIL="true"
oc patch -n ${HC_CLUSTER_NS} hostedclusters/${HC_CLUSTER_NAME} -p '{"spec":
{"pausedUntil":"'${PAUSED_UNTIL}'"}}' --type=merge
oc patch -n ${HC_CLUSTER_NS} nodepools/${NODEPOOLS} -p '{"spec":
{"pausedUntil":"'${PAUSED_UNTIL}'"}}' --type=merge
oc scale deployment -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --replicas=0 kube-
apiserver openshift-apiserver openshift-oauth-apiserver control-plane-operator
```

3. Back up etcd and upload the data to an S3 bucket by running this bash script:

TIP

Wrap this script in a function and call it from the main function.

```
# ETCD Backup
ETCD_PODS="etcd-0"
if [ "${CONTROL_PLANE_AVAILABILITY_POLICY}" = "HighlyAvailable" ]; then
  ETCD_PODS="etcd-0 etcd-1 etcd-2"
fi

for POD in ${ETCD_PODS}; do
  # Create an etcd snapshot
  oc exec -it ${POD} -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -- env
ETCDCTL_API=3 /usr/bin/etcdctl --cacert /etc/etcd/tls/client/etcd-client-ca.crt --cert
/etc/etcd/tls/client/etcd-client.crt --key /etc/etcd/tls/client/etcd-client.key --
endpoints=localhost:2379 snapshot save /var/lib/data/snapshot.db
  oc exec -it ${POD} -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -- env
ETCDCTL_API=3 /usr/bin/etcdctl -w table snapshot status /var/lib/data/snapshot.db
```

```
  FILEPATH="/${BUCKET_NAME}/${HC_CLUSTER_NAME}-${POD}-snapshot.db"
  CONTENT_TYPE="application/x-compressed-tar"
  DATE_VALUE=`date -R`
  SIGNATURE_STRING="PUT\n\n${CONTENT_TYPE}\n${DATE_VALUE}\n${FILEPATH}"

  set +x
  ACCESS_KEY=$(grep aws_access_key_id ${AWS_CREDS} | head -n1 | cut -d= -f2 | sed
"s/ //g")
  SECRET_KEY=$(grep aws_secret_access_key ${AWS_CREDS} | head -n1 | cut -d= -f2 |
sed "s/ //g")
  SIGNATURE_HASH=$(echo -en ${SIGNATURE_STRING} | openssl sha1 -hmac
"${SECRET_KEY}" -binary | base64)
  set -x

  # FIXME: this is pushing to the OIDC bucket
  oc exec -it etcd-0 -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -- curl -X PUT -T
"/var/lib/data/snapshot.db" \
    -H "Host: ${BUCKET_NAME}.s3.amazonaws.com" \
    -H "Date: ${DATE_VALUE}" \
    -H "Content-Type: ${CONTENT_TYPE}" \
    -H "Authorization: AWS ${ACCESS_KEY}:${SIGNATURE_HASH}" \
    https://${BUCKET_NAME}.s3.amazonaws.com/${HC_CLUSTER_NAME}-${POD}-
snapshot.db
done
```

For more information about backing up etcd, see "Backing up and restoring etcd on a hosted cluster".

4. Back up Kubernetes and OpenShift Container Platform objects by entering the following commands. You need to back up the following objects:

   - **HostedCluster** and **NodePool** objects from the HostedCluster namespace
   - **HostedCluster** secrets from the HostedCluster namespace
   - **HostedControlPlane** from the Hosted Control Plane namespace
   - **Cluster** from the Hosted Control Plane namespace
   - **AWSCluster**, **AWSMachineTemplate**, and **AWSMachine** from the Hosted Control Plane namespace
   - **MachineDeployments**, **MachineSets**, and **Machines** from the Hosted Control Plane namespace
   - **ControlPlane** secrets from the Hosted Control Plane namespace

```
mkdir -p ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
chmod 700 ${BACKUP_DIR}/namespaces/

# HostedCluster
echo "Backing Up HostedCluster Objects:"
oc get hc ${HC_CLUSTER_NAME} -n ${HC_CLUSTER_NS} -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/hc-${HC_CLUSTER_NAME}.yaml
echo "--> HostedCluster"
```

```
sed -i '' -e '/^status:$/,$d' ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/hc-
${HC_CLUSTER_NAME}.yaml

# NodePool
oc get np ${NODEPOOLS} -n ${HC_CLUSTER_NS} -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/np-${NODEPOOLS}.yaml
echo "--> NodePool"
sed -i '' -e '/^status:$/,$ d' ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/np-
${NODEPOOLS}.yaml

# Secrets in the HC Namespace
echo "--> HostedCluster Secrets:"
for s in $(oc get secret -n ${HC_CLUSTER_NS} | grep "^${HC_CLUSTER_NAME}" |
awk '{print $1}'); do
    oc get secret -n ${HC_CLUSTER_NS} $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/secret-${s}.yaml
done

# Secrets in the HC Control Plane Namespace
echo "--> HostedCluster ControlPlane Secrets:"
for s in $(oc get secret -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} | egrep -v
"docker|service-account-token|oauth-openshift|NAME|token-${HC_CLUSTER_NAME}" |
awk '{print $1}'); do
    oc get secret -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}/secret-
${s}.yaml
done

# Hosted Control Plane
echo "--> HostedControlPlane:"
oc get hcp ${HC_CLUSTER_NAME} -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
-o yaml > ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/hcp-${HC_CLUSTER_NAME}.yaml

# Cluster
echo "--> Cluster:"
CL_NAME=$(oc get hcp ${HC_CLUSTER_NAME} -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} -o jsonpath={.metadata.labels.\*} | grep
${HC_CLUSTER_NAME})
oc get cluster ${CL_NAME} -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -o yaml
> ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}/cl-
${HC_CLUSTER_NAME}.yaml

# AWS Cluster
echo "--> AWS Cluster:"
oc get awscluster ${HC_CLUSTER_NAME} -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}/awscl-
${HC_CLUSTER_NAME}.yaml

# AWS MachineTemplate
echo "--> AWS Machine Template:"
oc get awsmachinetemplate ${NODEPOOLS} -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}/awsmt-
${HC_CLUSTER_NAME}.yaml
```

```
# AWS Machines
echo "--> AWS Machine:"
CL_NAME=$(oc get hcp ${HC_CLUSTER_NAME} -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} -o jsonpath={.metadata.labels.\*} | grep
${HC_CLUSTER_NAME})
for s in $(oc get awsmachines -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --no-
headers | grep ${CL_NAME} | cut -f1 -d\ ); do
    oc get -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} awsmachines $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}/awsm-
${s}.yaml
done

# MachineDeployments
echo "--> HostedCluster MachineDeployments:"
for s in $(oc get machinedeployment -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
-o name); do
    mdp_name=$(echo ${s} | cut -f 2 -d /)
    oc get -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machinedeployment-${mdp_name}.yaml
done

# MachineSets
echo "--> HostedCluster MachineSets:"
for s in $(oc get machineset -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -o
name); do
    ms_name=$(echo ${s} | cut -f 2 -d /)
    oc get -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machineset-${ms_name}.yaml
done

# Machines
echo "--> HostedCluster Machine:"
for s in $(oc get machine -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -o name);
do
    m_name=$(echo ${s} | cut -f 2 -d /)
    oc get -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} $s -o yaml >
${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machine-${m_name}.yaml
done
```

5. Clean up the **ControlPlane** routes by entering this command:

   ```
   $ oc delete routes -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --all
   ```

   By entering that command, you enable the ExternalDNS Operator to delete the Route53 entries.

6. Verify that the Route53 entries are clean by running this script:

   ```
   function clean_routes() {

       if [[ -z "${1}" ]];then
           echo "Give me the NS where to clean the routes"
   ```

```
        exit 1
    fi

    # Constants
    if [[ -z "${2}" ]];then
        echo "Give me the Route53 zone ID"
        exit 1
    fi

    ZONE_ID=${2}
    ROUTES=10
    timeout=40
    count=0

    # This allows us to remove the ownership in the AWS for the API route
    oc delete route -n ${1} --all

    while [ ${ROUTES} -gt 2 ]
    do
        echo "Waiting for ExternalDNS Operator to clean the DNS Records in AWS Route53
where the zone id is: ${ZONE_ID}..."
        echo "Try: (${count}/${timeout})"
        sleep 10
        if [[ $count -eq timeout ]];then
            echo "Timeout waiting for cleaning the Route53 DNS records"
            exit 1
        fi
        count=$((count+1))
        ROUTES=$(aws route53 list-resource-record-sets --hosted-zone-id ${ZONE_ID} --max-
items 10000 --output json | grep -c ${EXTERNAL_DNS_DOMAIN})
    done
}

# SAMPLE: clean_routes "<HC ControlPlane Namespace>" "<AWS_ZONE_ID>"
clean_routes "${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}" "${AWS_ZONE_ID}"
```

Verification

Check all of the OpenShift Container Platform objects and the S3 bucket to verify that everything looks as expected.

Next steps

Restore your hosted cluster.

5.4.4.4. Restoring a hosted cluster

Gather all of the objects that you backed up and restore them in your destination management cluster.

Prerequisites

You backed up the data from your source management cluster.

TIP

Ensure that the **kubeconfig** file of the destination management cluster is placed as it is set in the **KUBECONFIG** variable or, if you use the script, in the **MGMT2_KUBECONFIG** variable. Use **export KUBECONFIG=<Kubeconfig FilePath>** or, if you use the script, use **export KUBECONFIG=${MGMT2_KUBECONFIG}**.

Procedure

1. Verify that the new management cluster does not contain any namespaces from the cluster that you are restoring by entering these commands:

```
# Just in case
export KUBECONFIG=${MGMT2_KUBECONFIG}
BACKUP_DIR=${HC_CLUSTER_DIR}/backup

# Namespace deletion in the destination Management cluster
$ oc delete ns ${HC_CLUSTER_NS} || true
$ oc delete ns ${HC_CLUSTER_NS}-{HC_CLUSTER_NAME} || true
```

2. Re-create the deleted namespaces by entering these commands:

```
# Namespace creation
$ oc new-project ${HC_CLUSTER_NS}
$ oc new-project ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
```

3. Restore the secrets in the HC namespace by entering this command:

```
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/secret-*
```

4. Restore the objects in the **HostedCluster** control plane namespace by entering these commands:

```
# Secrets
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/secret-*

# Cluster
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/hcp-*
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/cl-*
```

5. If you are recovering the nodes and the node pool to reuse AWS instances, restore the objects in the HC control plane namespace by entering these commands:

```
# AWS
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/awscl-*
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/awsmt-*
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/awsm-*
```

```
# Machines
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machinedeployment-*
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machineset-*
$ oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}/machine-*
```

6. Restore the etcd data and the hosted cluster by running this bash script:

```
ETCD_PODS="etcd-0"
if [ "${CONTROL_PLANE_AVAILABILITY_POLICY}" = "HighlyAvailable" ]; then
  ETCD_PODS="etcd-0 etcd-1 etcd-2"
fi

HC_RESTORE_FILE=${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/hc-
${HC_CLUSTER_NAME}-restore.yaml
HC_BACKUP_FILE=${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/hc-
${HC_CLUSTER_NAME}.yaml
HC_NEW_FILE=${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/hc-
${HC_CLUSTER_NAME}-new.yaml
cat ${HC_BACKUP_FILE} > ${HC_NEW_FILE}
cat > ${HC_RESTORE_FILE} <<EOF
    restoreSnapshotURL:
EOF

for POD in ${ETCD_PODS}; do
  # Create a pre-signed URL for the etcd snapshot
  ETCD_SNAPSHOT="s3://${BUCKET_NAME}/${HC_CLUSTER_NAME}-${POD}-
snapshot.db"
  ETCD_SNAPSHOT_URL=$(AWS_DEFAULT_REGION=${MGMT2_REGION} aws s3
presign ${ETCD_SNAPSHOT})

  # FIXME no CLI support for restoreSnapshotURL yet
  cat >> ${HC_RESTORE_FILE} <<EOF
    - "${ETCD_SNAPSHOT_URL}"
EOF
done

cat ${HC_RESTORE_FILE}

if ! grep ${HC_CLUSTER_NAME}-snapshot.db ${HC_NEW_FILE}; then
  sed -i '' -e "/type: PersistentVolume/r ${HC_RESTORE_FILE}" ${HC_NEW_FILE}
  sed -i '' -e '/pausedUntil:/d' ${HC_NEW_FILE}
fi

HC=$(oc get hc -n ${HC_CLUSTER_NS} ${HC_CLUSTER_NAME} -o name || true)
if [[ ${HC} == "" ]];then
    echo "Deploying HC Cluster: ${HC_CLUSTER_NAME} in ${HC_CLUSTER_NS}
namespace"
    oc apply -f ${HC_NEW_FILE}
else
    echo "HC Cluster ${HC_CLUSTER_NAME} already exists, avoiding step"
fi
```

7. If you are recovering the nodes and the node pool to reuse AWS instances, restore the node pool by entering this command:

```
oc apply -f ${BACKUP_DIR}/namespaces/${HC_CLUSTER_NS}/np-*
```

Verification

- To verify that the nodes are fully restored, use this function:

```
timeout=40
count=0
NODE_STATUS=$(oc get nodes --kubeconfig=${HC_KUBECONFIG} | grep -v NotReady |
grep -c "worker") || NODE_STATUS=0

while [ ${NODE_POOL_REPLICAS} != ${NODE_STATUS} ]
do
   echo "Waiting for Nodes to be Ready in the destination MGMT Cluster:
${MGMT2_CLUSTER_NAME}"
   echo "Try: (${count}/${timeout})"
   sleep 30
   if [[ $count -eq timeout ]];then
      echo "Timeout waiting for Nodes in the destination MGMT Cluster"
      exit 1
   fi
   count=$((count+1))
   NODE_STATUS=$(oc get nodes --kubeconfig=${HC_KUBECONFIG} | grep -v NotReady |
grep -c "worker") || NODE_STATUS=0
done
```

Next steps

Shut down and delete your cluster.

5.4.4.5. Deleting a hosted cluster from your source management cluster

After you back up your hosted cluster and restore it to your destination management cluster, you shut down and delete the hosted cluster on your source management cluster.

Prerequisites

You backed up your data and restored it to your source management cluster.

TIP

Ensure that the **kubeconfig** file of the destination management cluster is placed as it is set in the **KUBECONFIG** variable or, if you use the script, in the **MGMT_KUBECONFIG** variable. Use **export KUBECONFIG=<Kubeconfig FilePath>** or, if you use the script, use **export KUBECONFIG=${MGMT_KUBECONFIG}**.

Procedure

1. Scale the **deployment** and **statefulset** objects by entering these commands:

```
# Just in case
export KUBECONFIG=${MGMT_KUBECONFIG}
```

```
# Scale down deployments
oc scale deployment -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --replicas=0 --all
oc scale statefulset.apps -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --replicas=0 --
all
sleep 15
```

2. Delete the **NodePool** objects by entering these commands:

```
NODEPOOLS=$(oc get nodepools -n ${HC_CLUSTER_NS} -o=jsonpath='{.items[?
(@.spec.clusterName=="'${HC_CLUSTER_NAME}'")].metadata.name}')
if [[ ! -z "${NODEPOOLS}" ]];then
   oc patch -n "${HC_CLUSTER_NS}" nodepool ${NODEPOOLS} --type=json --patch='[ {
"op":"remove", "path": "/metadata/finalizers" }]'
   oc delete np -n ${HC_CLUSTER_NS} ${NODEPOOLS}
fi
```

3. Delete the **machine** and **machineset** objects by entering these commands:

```
# Machines
for m in $(oc get machines -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -o name); do
   oc patch -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} ${m} --type=json --patch='[ {
"op":"remove", "path": "/metadata/finalizers" }]' || true
   oc delete -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} ${m} || true
done

oc delete machineset -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --all || true
```

4. Delete the cluster object by entering these commands:

```
# Cluster
C_NAME=$(oc get cluster -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} -o name)
oc patch -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} ${C_NAME} --type=json --
patch='[ { "op":"remove", "path": "/metadata/finalizers" }]'
oc delete cluster.cluster.x-k8s.io -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} --all
```

5. Delete the AWS machines (Kubernetes objects) by entering these commands. Do not worry about deleting the real AWS machines. The cloud instances will not be affected.

```
# AWS Machines
for m in $(oc get awsmachine.infrastructure.cluster.x-k8s.io -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} -o name)
do
   oc patch -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} ${m} --type=json --patch='[ {
"op":"remove", "path": "/metadata/finalizers" }]' || true
   oc delete -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} ${m} || true
done
```

6. Delete the **HostedControlPlane** and **ControlPlane** HC namespace objects by entering these commands:

```
# Delete HCP and ControlPlane HC NS
oc patch -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
hostedcontrolplane.hypershift.openshift.io ${HC_CLUSTER_NAME} --type=json --patch='[ {
```

```
"op":"remove", "path": "/metadata/finalizers" }]'
oc delete hostedcontrolplane.hypershift.openshift.io -n ${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME} --all
oc delete ns ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME} || true
```

7. Delete the **HostedCluster** and HC namespace objects by entering these commands:

```
# Delete HC and HC Namespace
oc -n ${HC_CLUSTER_NS} patch hostedclusters ${HC_CLUSTER_NAME} -p '{"metadata":
{"finalizers":null}}' --type merge || true
oc delete hc -n ${HC_CLUSTER_NS} ${HC_CLUSTER_NAME}  || true
oc delete ns ${HC_CLUSTER_NS} || true
```

Verification

- To verify that everything works, enter these commands:

```
# Validations
export KUBECONFIG=${MGMT2_KUBECONFIG}

oc get hc -n ${HC_CLUSTER_NS}
oc get np -n ${HC_CLUSTER_NS}
oc get pod -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}
oc get machines -n ${HC_CLUSTER_NS}-${HC_CLUSTER_NAME}

# Inside the HostedCluster
export KUBECONFIG=${HC_KUBECONFIG}
oc get clusterversion
oc get nodes
```

Next steps

Delete the OVN pods in the hosted cluster so that you can connect to the new OVN control plane that runs in the new management cluster:

1. Load the **KUBECONFIG** environment variable with the hosted cluster's kubeconfig path.

2. Enter this command:

```
$ oc delete pod -n openshift-ovn-kubernetes --all
```

5.4.4.6. Running a script to back up and restore a hosted cluster

To expedite the process to back up a hosted cluster and restore it within the same region on AWS, you can modify and run a script.

Procedure

1. Replace the variables in the following script with your information:

```
# Fill the Common variables to fit your environment, this is just a sample
SSH_KEY_FILE=${HOME}/.ssh/id_rsa.pub
BASE_PATH=${HOME}/hypershift
BASE_DOMAIN="aws.sample.com"
```

```
PULL_SECRET_FILE="${HOME}/pull_secret.json"
AWS_CREDS="${HOME}/.aws/credentials"
CONTROL_PLANE_AVAILABILITY_POLICY=SingleReplica
HYPERSHIFT_PATH=${BASE_PATH}/src/hypershift
HYPERSHIFT_CLI=${HYPERSHIFT_PATH}/bin/hypershift
HYPERSHIFT_IMAGE=${HYPERSHIFT_IMAGE:-"quay.io/${USER}/hypershift:latest"}
NODE_POOL_REPLICAS=${NODE_POOL_REPLICAS:-2}

# MGMT Context
MGMT_REGION=us-west-1
MGMT_CLUSTER_NAME="${USER}-dev"
MGMT_CLUSTER_NS=${USER}
MGMT_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${MGMT_CLUSTER_NS}-
${MGMT_CLUSTER_NAME}"
MGMT_KUBECONFIG="${MGMT_CLUSTER_DIR}/kubeconfig"

# MGMT2 Context
MGMT2_CLUSTER_NAME="${USER}-dest"
MGMT2_CLUSTER_NS=${USER}
MGMT2_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${MGMT2_CLUSTER_NS}-
${MGMT2_CLUSTER_NAME}"
MGMT2_KUBECONFIG="${MGMT2_CLUSTER_DIR}/kubeconfig"

# Hosted Cluster Context
HC_CLUSTER_NS=clusters
HC_REGION=us-west-1
HC_CLUSTER_NAME="${USER}-hosted"
HC_CLUSTER_DIR="${BASE_PATH}/hosted_clusters/${HC_CLUSTER_NS}-
${HC_CLUSTER_NAME}"
HC_KUBECONFIG="${HC_CLUSTER_DIR}/kubeconfig"
BACKUP_DIR=${HC_CLUSTER_DIR}/backup

BUCKET_NAME="${USER}-hosted-${MGMT_REGION}"

# DNS
AWS_ZONE_ID="Z026552815SS3YPH9H6MG"
EXTERNAL_DNS_DOMAIN="guest.jpdv.aws.kerbeross.com"
```

2. Save the script to your local file system.

3. Run the script by entering the following command:

```
source <env_file>
```

where: **env_file** is the name of the file where you saved the script.