OpenShift Container Platform 4.6

Backup and restore

Backing up and restoring your OpenShift Container Platform cluster

Last Updated: 2022-10-04

OpenShift Container Platform 4.6 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

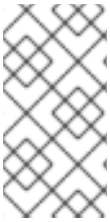
CHAPTER 1. BACKUP AND RESTORE

1.1. OVERVIEW OF BACKUP AND RESTORE OPERATIONS IN OPENSHIFT CONTAINER PLATFORM

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 1.7. 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:
  - Amazon Web Services

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

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

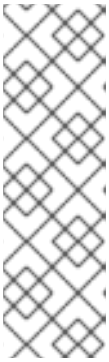
```
Starting pod/ip-10-0-130-169us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:17 UTC, use 'shutdown -c' to cancel.

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 (also known as the master 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.

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 (also known as the master 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   v1.19.0
   ip-10-0-170-223.ec2.internal   Ready    master   75m   v1.19.0
   ip-10-0-211-16.ec2.internal    Ready    master   75m   v1.19.0
   ```
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   v1.19.0
ip-10-0-182-134.ec2.internal   Ready    worker   64m   v1.19.0
ip-10-0-250-100.ec2.internal   Ready    worker   64m   v1.19.0
```

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                             4.6.0     True        False         False      59m
cloud-credential                           4.6.0     True        False         False      85m
cluster-autoscaler                         4.6.0     True        False         False      73m
config-operator                            4.6.0     True        False         False      73m
console                                    4.6.0     True        False         False      62m
csi-snapshot-controller                    4.6.0     True        False         False      66m
dns                                        4.6.0     True        False         False      76m
etcd                                       4.6.0     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   v1.19.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.19.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.19.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.19.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.19.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.19.0
```

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 FEATURES AND PLUG-INS

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

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

4.1.1. 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.

4.1.2. OADP plug-ins

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

OADP also provides plug-ins for OpenShift Container Platform resource backups and Container Storage Interface (CSI) snapshots.

Table 4.1. OADP plug-ins

OADP plug-in	Function	Storage location
aws	Backs up and restores Kubernetes objects by using object store.	AWS S3
	Backs up and restores volumes by using snapshots.	AWS EBS

OADP plug-in	Function	Storage location
azure	Backs up and restores Kubernetes objects by using object store.	Microsoft Azure Blob storage
	Backs up and restores volumes by using snapshots.	Microsoft Azure Managed Disks
gcp	Backs up and restores Kubernetes objects by using object store.	Google Cloud Storage
	Backs up and restores volumes by using snapshots.	Google Compute Engine Disks
openshift	Backs up and restores OpenShift Container Platform resources by using object store. [1]	Object store
csi	Backs up and restores volumes by using CSI snapshots. [2]	Cloud storage that supports CSI snapshots

1. Mandatory.
2. The **csi** plug-in uses the Velero CSI beta snapshot API.

4.1.3. About OADP Velero plug-ins

You can configure two types of plug-ins when you install Velero:

- Default cloud provider plug-ins
- Custom plug-ins

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

4.1.3.1. Default Velero cloud provider plug-ins

You can install any of the following default Velero cloud provider plug-ins 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 plug-in)
- **csi** (Container Storage Interface)
- **kubevirt** (KubeVirt)

You specify the desired default plug-ins in the **oadp_v1alpha1_dpa.yaml** file during deployment.

Example file

The following **.yaml** file installs the **openshift**, **aws**, **azure**, and **gcp** plug-ins:

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

4.1.3.2. Custom Velero plug-ins

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

You specify the desired custom plug-ins in the **oadp_v1alpha1_dpa.yaml** file during deployment.

Example file

The following **.yaml** file installs the default **openshift**, **azure**, and **gcp** plug-ins and a custom plug-in 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
```

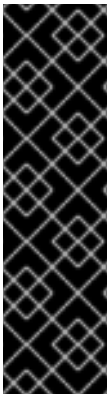
4.2. INSTALLING AND CONFIGURING OADP

4.2.1. 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 1.7.

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 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/.

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 Container Storage

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

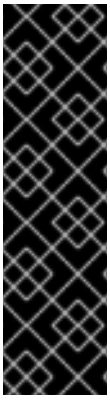
You create a **Secret** object for your storage provider credentials and then you install the Data Protection Application.

Additional resources

- Overview of backup locations and snapshot locations in the Velero documentation.

4.2.2. 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, configuring AWS for Velero, and then installing the Data Protection Application.



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 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.

4.2.2.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.6 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.7.

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.

4.2.2.2. Configuring Amazon Web Services S3

You can configure an Amazon Web Services (AWS) S3 storage bucket as a replication repository for the Migration Toolkit for Containers (MTC).

Prerequisites

- The AWS S3 storage bucket must be accessible to the source and target clusters.
- You must have the AWS CLI installed.
- If you are using the snapshot copy method:
  - You must have access to EC2 Elastic Block Storage (EBS).
  - The source and target clusters must be in the same region.

- The source and target clusters must have the same storage class.
- The storage class must be compatible with snapshots.

Procedure

1. Create an AWS S3 bucket:

```
$ aws s3api create-bucket \
    --bucket <bucket> \ 1
    --region <bucket_region> 2
```

1. Specify your S3 bucket name.
2. Specify your S3 bucket region, for example, **us-east-1**.

2. Create the IAM user **velero**:

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

3. Create an EC2 EBS snapshot policy:

```
$ cat > velero-ec2-snapshot-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeVolumes",
                "ec2:DescribeSnapshots",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:CreateSnapshot",
                "ec2:DeleteSnapshot"
            ],
            "Resource": "*"
        }
    ]
}
EOF
```

4. Create an AWS S3 access policy for one or for all S3 buckets:

```
$ cat > velero-s3-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:PutObject",
```

```
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket>/*" 1
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket>" 2
            ]
        }
    ]
}
EOF
```

1 2 To grant access to a single S3 bucket, specify the bucket name. To grant access to all AWS S3 buckets, specify * instead of a bucket name as in the following example:

Example output

```
"Resource": [
    "arn:aws:s3:::*"
```

5. Attach the EC2 EBS policy to **velero**:

```
$ aws iam put-user-policy \
  --user-name velero \
  --policy-name velero-ebs \
  --policy-document file://velero-ec2-snapshot-policy.json
```

6. Attach the AWS S3 policy to **velero**:

```
$ aws iam put-user-policy \
  --user-name velero \
  --policy-name velero-s3 \
  --policy-document file://velero-s3-policy.json
```

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

```
$ aws iam create-access-key --user-name velero
{
  "AccessKey": {
        "UserName": "velero",
        "Status": "Active",
        "CreateDate": "2017-07-31T22:24:41.576Z",
        "SecretAccessKey": <AWS_SECRET_ACCESS_KEY>, 1
```

```
        "AccessKeyId": <AWS_ACCESS_KEY_ID> 2
    }
}
```

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.

4.2.2.3. Creating a secret for backup and snapshot locations

You create a **Secret** object for the backup and snapshot locations if they use the same credentials.

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

Prerequisites

- Your object storage and cloud storage 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.

NOTE

The **DataProtectionApplication** custom resource (CR) requires a **Secret** for installation. If no **spec.backupLocations.credential.name** value is specified, the default name is used.

If you do not want to specify the backup locations or the snapshot locations, you must create a **Secret** with the default name by using an empty **credentials-velero** file.

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.

4.2.2.3.1. Configuring secrets for different backup and snapshot location 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"
```

4.2.2.4. Configuring the Data Protection Application

You can configure Velero resource allocations and enable self-signed CA certificates.

4.2.2.4.1. 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:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

(1) (2) (1) (1) Specify the value in millicpus or CPU units. Default value is **500m** or **1** CPU unit.

(2) Default value is **512Mi**.

(3) Default value is **500m** or **1** CPU unit.

(4) Default value is **256Mi**.

4.2.2.4.2. 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. Must be **false** to disable SSL/TLS security.

4.2.2.5. 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 <.>
        - aws
    restic:
      enable: true <.>
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket_name> <.>
          prefix: <prefix> <.>
        config:
          region: <region>
          profile: "default"
        credential:
          key: cloud
          name: cloud-credentials <.>
  snapshotLocations: <.>
    - name: default
      velero:
        provider: aws
        config:
          region: <region> <.>
          profile: "default"
```

<.> The **openshift** plug-in is mandatory in order to back up and restore namespaces on an OpenShift Container Platform cluster. <.> 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. <.> Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix. <.> Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes. <.> 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. <.> You do not need to specify a snapshot location if you use CSI snapshots or Restic to back up PVs. <.> 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
```

4.2.2.5.1. 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:
```

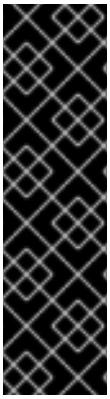
```
    defaultPlugins:
    - openshift
    - csi (1)
  featureFlags:
  - EnableCSI (2)
```

(1) Add the **csi** default plug-in.

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

4.2.3. 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, configuring Azure for Velero, and then installing the Data Protection Application.



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 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.

4.2.3.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.6 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.7.

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.

4.2.3.2. Configuring Microsoft Azure Blob

You can configure a Microsoft Azure Blob storage container as a replication repository for the Migration Toolkit for Containers (MTC).

Prerequisites

- You must have an Azure storage account.
- You must have the Azure CLI installed.
- The Azure Blob storage container must be accessible to the source and target clusters.
- If you are using the snapshot copy method:
  - The source and target clusters must be in the same region.
  - The source and target clusters must have the same storage class.
  - The storage class must be compatible with snapshots.

Procedure

1. Set the **AZURE_RESOURCE_GROUP** variable:

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. Create an Azure resource group:

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

1 Specify your location.

3. Set the **AZURE_STORAGE_ACCOUNT_ID** variable:

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. 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
```

5. Set the **BLOB_CONTAINER** variable:

```
$ BLOB_CONTAINER=velero
```

6. Create an Azure Blob storage container:

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

7. 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`
```

8. 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_ACCESS_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.

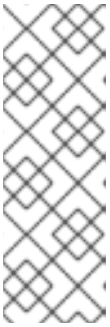
4.2.3.3. Creating a secret for backup and snapshot locations

You create a **Secret** object for the backup and snapshot locations if they use the same credentials.

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

Prerequisites

- Your object storage and cloud storage 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.



NOTE

The **DataProtectionApplication** custom resource (CR) requires a **Secret** for installation. If no **spec.backupLocations.credential.name** value is specified, the default name is used.

If you do not want to specify the backup locations or the snapshot locations, you must create a **Secret** with the default name by using an empty **credentials-velero** file.

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.

4.2.3.3.1. Configuring secrets for different backup and snapshot location credentials

If your backup and snapshot locations use different credentials, you 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.

4.2.3.4. Configuring the Data Protection Application

You can configure Velero resource allocations and enable self-signed CA certificates.

4.2.3.4.1. 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:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

1. Specify the value in millicpus or CPU units. Default value is **500m** or **1** CPU unit.
2. Default value is **512Mi**.
3. Default value is **500m** or **1** CPU unit.
4. Default value is **256Mi**.

4.2.3.4.2. 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. Must be **false** to disable SSL/TLS security.

4.2.3.5. 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 <.>
    restic:
```

```
    enable: true <.>
  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: cloud-credentials-azure  <.>
        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
```

<.> The **openshift** plug-in is mandatory in order to back up and restore namespaces on an OpenShift Container Platform cluster. <.> 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. <.> Specify the Azure resource group. <.> Specify the Azure storage account ID. <.> Specify the Azure subscription ID. <.> 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. <.> Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix. <.> Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes. <.> 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
```

4.2.3.5.1. 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:

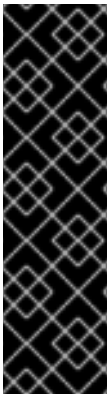
```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) Add the **csi** default plug-in.

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

4.2.4. 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, configuring GCP for Velero, and then installing the Data Protection Application.



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 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.

4.2.4.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.6 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.7.

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.

4.2.4.2. Configuring Google Cloud Platform

You can configure a Google Cloud Platform (GCP) storage bucket as a replication repository for the Migration Toolkit for Containers (MTC).

Prerequisites

- The GCP storage bucket must be accessible to the source and target clusters.
- You must have **gsutil** installed.
- If you are using the snapshot copy method:

- The source and target clusters must be in the same region.
- The source and target clusters must have the same storage class.
- The storage class must be compatible with snapshots.

Procedure

1. Log in to **gsutil**:

```
$ gsutil init
```

Example output

```
Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

To continue, you must login. Would you like to login (Y/n)?
```

2. Set the **BUCKET** variable:

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

1 Specify your bucket name.

3. Create a 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 **velero** IAM service account:

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

6. Create the **SERVICE_ACCOUNT_EMAIL** variable:

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

7. Create the **ROLE_PERMISSIONS** variable:

```
$ 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
)
```

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

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

9. 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
```

10. Update the IAM service account:

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

11. 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.

4.2.4.3. Creating a secret for backup and snapshot locations

You create a **Secret** object for the backup and snapshot locations if they use the same credentials.

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

Prerequisites

- Your object storage and cloud storage 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.

4.2.4.3.1. Configuring secrets for different backup and snapshot location credentials

If your backup and snapshot locations use different credentials, you 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.

4.2.4.4. Configuring the Data Protection Application

You can configure Velero resource allocations and enable self-signed CA certificates.

4.2.4.4.1. 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:
        resourceAllocations:
          limits:
            cpu: "1" 1
            memory: 512Mi 2
          requests:
            cpu: 500m 3
            memory: 256Mi 4
```

1 Specify the value in millicpus or CPU units. Default value is **500m** or **1** CPU unit.

2 Default value is **512Mi**.

3 Default value is **500m** or **1** CPU unit.

4 Default value is **256Mi**.

4.2.4.4.2. 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. Must be **false** to disable SSL/TLS security.

4.2.4.5. 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 <.>
    restic:
      enable: true <.>
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
          name: cloud-credentials-gcp <.>
        objectStorage:
          bucket: <bucket_name> <.>
          prefix: <prefix> <.>
  snapshotLocations: <.>
    - velero:
        provider: gcp
        default: true
        config:
          project: <project>
          snapshotLocation: us-west1 <.>
```

<.> The **openshift** plug-in is mandatory in order to back up and restore namespaces on an OpenShift Container Platform cluster. <.> 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. <.> 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. <.> Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix. <.> Specify a prefix for Velero backups, for example, **velero**, if the bucket is used for multiple purposes. <.> You do not need to specify a snapshot location if you use CSI snapshots or Restic to back up PVs. <.> 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
```

4.2.4.5.1. 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 plug-in.

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

4.2.5. 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, creating a **Secret** object, and then installing the Data Protection Application.

MCG is a component of OpenShift Container Storage (OCS). You configure MCG as a backup location in the **DataProtectionApplication** custom resource (CR).

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, configure a snapshot location. If your cloud provider does not support snapshots or if your storage is NFS, you can create backups with Restic.

You do not need to specify a snapshot location in the **DataProtectionApplication** CR for Restic or Container Storage Interface (CSI) snapshots.

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.

4.2.5.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.6 by using Operator Lifecycle Manager (OLM).

The OADP Operator installs Velero 1.7.

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.

4.2.5.2. Configuring the Multicloud Object Gateway

You can configure the Multicloud Object Gateway (MCG) as a replication repository for the Migration Toolkit for Containers (MTC). MCG is a component of OpenShift Container Storage.

Procedure

1. Deploy OpenShift Container Storage by using the appropriate OpenShift Container Storage deployment guide.
2. Obtain the S3 endpoint, **AWS_ACCESS_KEY_ID**, and **AWS_SECRET_ACCESS_KEY** by running the **describe** command on the **NooBaa** custom resource.
   These values are required in order to add MCG as a replication repository to the MTC web console.

4.2.5.3. Creating a secret for backup and snapshot locations

You create a **Secret** object for the backup and snapshot locations if they use the same credentials.

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

Prerequisites

- Your object storage and cloud storage 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.

4.2.5.3.1. Configuring secrets for different backup and snapshot location credentials

If your backup and snapshot locations use different credentials, you 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:
  configuration:
    velero:
      defaultPlugins:
        - aws
        - openshift
    restic:
      enable: true
  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.

4.2.5.4. Configuring the Data Protection Application

You can configure Velero resource allocations and enable self-signed CA certificates.

4.2.5.4.1. 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:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

(1) Specify the value in millicpus or CPU units. Default value is **500m** or **1** CPU unit.

(2) Default value is **512Mi**.

3 Default value is **500m** or **1** CPU unit.

4 Default value is **256Mi**.

4.2.5.4.2. 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 Must be **false** to disable SSL/TLS security.

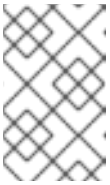
4.2.5.5. Installing the Data Protection Application

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 <.>
    restic:
      enable: true <.>
  backupLocations:
    - velero:
        config:
          profile: "default"
          region: minio
          s3Url: <url> <.>
          insecureSkipTLSVerify: "true"
          s3ForcePathStyle: "true"
        provider: aws
        default: true
        credential:
          key: cloud
          name: cloud-credentials <.>
```

```
    objectStorage:
      bucket: <bucket_name> <.>
      prefix: <prefix> <.>
```

<.> The **openshift** plug-in is mandatory in order to back up and restore namespaces on an OpenShift Container Platform cluster. <.> 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. <.> Specify the URL of the S3 endpoint. <.> 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. <.> Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix. <.> 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
```

4.2.5.5.1. 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 plug-in.

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

4.2.6. Installing and configuring the OpenShift API for Data Protection with OpenShift Container Storage

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

You can configure Multicloud Object Gateway or any S3-compatible object storage as a backup location in the **DataProtectionApplication** custom resource (CR).

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 the cloud provider has a native snapshot API, you can configure cloud storage as a snapshot location in the **DataProtectionApplication** CR. You do not need to specify a snapshot location for Restic or Container Storage Interface (CSI) snapshots.

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.

4.2.6.1. Installing the OADP Operator

You install the OpenShift API for Data Protection (OADP) Operator on OpenShift Container Platform 4.6 by using Operator Lifecycle Manager (OLM).

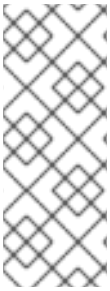
The OADP Operator installs Velero 1.7.

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.



NOTE

After you install the OADP Operator, you configure object storage as a backup location and cloud storage as a snapshot location, if the cloud provider supports a native snapshot API.

If the cloud provider does not support snapshots or if your storage is NFS, you can create backups with Restic. Restic does not require a snapshot location.

4.2.6.2. Creating a secret for backup and snapshot locations

You create a **Secret** object for the backup and snapshot locations if they use the same credentials.

The default name of the **Secret** is **cloud-credentials**, unless you specify a default plug-in for the backup storage provider.

Prerequisites

- Your object storage and cloud storage 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.

4.2.6.2.1. Configuring secrets for different backup and snapshot location credentials

If your backup and snapshot locations use different credentials, you 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:
  configuration:
    velero:
      defaultPlugins:
        - csi
        - openshift
    featureFlags:
    - EnableCSI
    restic:
      enable: true
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
```

```
      name: <custom_secret> (1)
    objectStorage:
      bucket: <bucket_name>
      prefix: <prefix>
```

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

4.2.6.3. Configuring the Data Protection Application

You can configure Velero resource allocations and enable self-signed CA certificates.

4.2.6.3.1. 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:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

(1) Specify the value in millicpus or CPU units. Default value is **500m** or **1** CPU unit.

(2) Default value is **512Mi**.

(3) Default value is **500m** or **1** CPU unit.

(4) Default value is **256Mi**.

4.2.6.3.2. 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. Must be **false** to disable SSL/TLS security.

4.2.6.4. 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:
        - gcp <.>
        - csi <.>
        - openshift <.>
    restic:
      enable: true <.>
  backupLocations:
    - velero:
        provider: gcp <.>
        default: true
        credential:
          key: cloud
          name: <default_secret> <.>
        objectStorage:
          bucket: <bucket_name> <.>
          prefix: <prefix> <.>
```

<.> Specify the default plug-in for the backup provider, for example, **gcp**, if appropriate. <.> Specify the **csi** default plug-in if you use CSI snapshots to back up PVs. The **csi** plug-in uses the Velero CSI beta snapshot APIs. You do not need to configure a snapshot location. <.> The **openshift** plug-in is mandatory in order to back up and restore namespaces on an OpenShift Container Platform cluster. <.> 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. <.> Specify the backup provider. <.> If you use a default plug-in 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. <.> Specify a bucket as the backup storage location. If the bucket is not a dedicated bucket for Velero backups, you must specify a prefix. <.> 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
```

4.2.6.4.1. 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 plug-in.

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

4.2.7. Uninstalling the OpenShift API for Data Protection

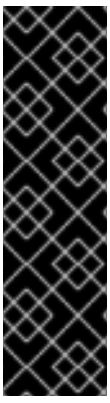
You uninstall the OpenShift API for Data Protection (OADP) by deleting the OADP Operator. See Deleting Operators from a cluster for details.

4.3. BACKING UP AND RESTORING

4.3.1. 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 Container Storage 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.

4.3.1.1. 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:

```
$ 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
    storageLocation: <velero-sample-1> 2
    ttl: 720h0m0s
```

1 Specify an array of namespaces to back up.

2 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}'
```

4.3.1.2. Backing up persistent volumes with CSI snapshots

You back up persistent volumes with Container Storage Interface (CSI) snapshots by creating a **VolumeSnapshotClass** custom resource (CR) to register the CSI driver before you create the **Backup** CR.

Prerequisites

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

Procedure

- Create a **VolumeSnapshotClass** CR, as in the following examples:

  Ceph RBD

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
deletionPolicy: Retain
metadata:
  name: <volume_snapshot_class_name>
  labels:
    velero.io/csi-volumesnapshot-class: "true"
    snapshotter: openshift-storage.rbd.csi.ceph.com
driver: openshift-storage.rbd.csi.ceph.com
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/snapshotter-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
```

  Ceph FS

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: <volume_snapshot_class_name>
  labels:
    velero.io/csi-volumesnapshot-class: "true"
driver: openshift-storage.cephfs.csi.ceph.com
```

```
deletionPolicy: Retain
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/snapshotter-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
```

Other cloud providers

```
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.

4.3.1.3. 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.

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.

4.3.1.4. 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:
        - <namespace>
        includedResources:
        - pods 2
        excludedResources: []
        labelSelector: 3
          matchLabels:
            app: velero
            component: server
        pre: 4
          - exec:
              container: <container> 5
              command:
              - /bin/uname 6
              - -a
              onError: Fail 7
              timeout: 30s 8
        post: 9
...
```

1. 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.
3. Optional: This hook only applies to objects matching the label selector.
4. Array of hooks to run before the backup.
5. Optional: If the container is not specified, the command runs in the first container in the pod.
6. Array of commands that the hook runs.
7. Allowed values for error handling are **Fail** and **Continue**. The default is **Fail**.

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

9 This block defines an array of hooks to run after the backup, with the same parameters as the pre-backup hooks.

4.3.1.5. 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}'
```

4.3.2. 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.

4.3.2.1. 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
  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. Verify that the status of the **Restore** CR is **Completed**:

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

3. Verify that the backup resources have been restored:

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

1 Namespace that you backed up.

4.3.2.2. 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
    - exec:
        container: <container> 4
        command:
        - /bin/bash 5
        - -c
        - "psql < /backup/backup.sql"
        waitTimeout: 5m 6
        execTimeout: 1m 7
        onError: Continue 8
```

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.
3. Optional: This hook only applies to objects matching the label selector.
4. Optional: If the container is not specified, the command runs in the first container in the pod.
5. Array of commands that the hook runs.
6. Optional: If the **waitTimeout** is not specified, the restore waits indefinitely. You can specify how long to wait for a container to start and for preceding hooks in the container to complete. The wait timeout starts when the container is restored and might require time for the container to pull the image and mount the volumes.
7. Optional: How long to wait for the commands to run. The default is **30s**.
8. 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**.

4.4. 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

4.4.1. 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

OADP version	Velero version
1.0.2	1.7.1
1.0.3	1.7.1

4.4.2. 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'
```

4.4.3. 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.

4.4.4. 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
```

4.4.5. Installation issues

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

4.4.5.1. 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.

4.4.5.2. 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 (**''**, **'**).

4.4.6. Backup and Restore CR issues

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

4.4.6.1. 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.

4.4.6.2. 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.

4.4.7. Restic issues

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

4.4.7.1. 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.

4.4.7.2. 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>
```

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

If you create a Restic **Backup** CR for a namespace, empty the S3 bucket, and then recreate the **Backup** CR for the same namespace, the recreated **Backup** CR fails.

The **velero** pod log displays the error message, **msg="Error checking repository for stale locks"**.

Cause

Velero does not create the Restic repository from the **ResticRepository** manifest if the Restic directories are deleted on object storage. See (Velero issue 4421) for details.

4.4.8. 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.

You can run the **must-gather** tool with the following data collection options:

- Full **must-gather** data collection collects Prometheus metrics, pod logs, and Velero CR information for all namespaces where the OADP Operator is installed.
- Essential **must-gather** data collection collects pod logs and Velero CR information for a specific duration of time, for example, one hour or 24 hours. Prometheus metrics and duplicate logs are not included.

- **must-gather** data collection with timeout. Data collection can take a long time if there are many failed **Backup** CRs. You can improve performance by setting a timeout value.
- Prometheus metrics data dump downloads an archive file containing the metrics data collected by Prometheus.

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 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:
   - Full **must-gather** data collection, including Prometheus metrics:

     ```
     $ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0
     ```

     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 .
   - Essential **must-gather** data collection, without Prometheus metrics, for a specific time duration:

     ```
     $ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
       -- /usr/bin/gather_<time>_essential 1
     ```

     1 Specify the time in hours. Allowed values are **1h**, **6h**, **24h**, **72h**, or **all**, for example, **gather_1h_essential** or **gather_all_essential**.
   - **must-gather** data collection with timeout:

     ```
     $ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
       -- /usr/bin/gather_with_timeout <timeout> 1
     ```

     1 Specify a timeout value in seconds.
   - Prometheus metrics data dump:

     ```
     $ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
       -- /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
```

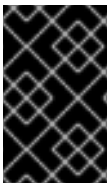
CHAPTER 5. CONTROL PLANE BACKUP AND RESTORE

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 4.6.2 cluster must use an etcd backup that was taken from 4.6.2.



IMPORTANT

Back up your cluster's etcd data by performing a single invocation of the backup script on a control plane host (also known as the master 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 (also known as the master 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 the 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

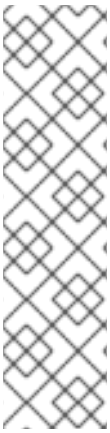
If etcd encryption is enabled, it is recommended to store this second file separately from the etcd snapshot for security reasons. However, this file is required in order to restore from the etcd snapshot.

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
```

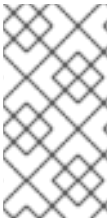
This example output shows that the **ip-10-0-131-183.ec2.internal** etcd member is unhealthy.

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   v1.19.0 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 (also known as the master 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   v1.19.0
ip-10-0-164-97.ec2.internal    Ready    master   6h13m   v1.19.0
ip-10-0-154-204.ec2.internal   Ready    master   6h13m   v1.19.0
```

   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:

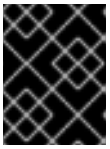
- Replacing an unhealthy etcd member whose machine is not running or whose node is not ready
- Replacing an unhealthy etcd member whose etcd pod is crashlooping

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.

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. 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
      ```

3. Delete and recreate the control plane machine (also known as the master 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. Update the **metadata.selfLink** field to use the new machine name from the previous step.

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  selfLink: /apis/machine.openshift.io/v1beta1/namespaces/openshift-machine-
api/machines/clustername-8qw5l-master-3
  ...
```

iv. Remove the **spec.providerID** field:

```
  providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

v. Remove the **metadata.annotations** and **metadata.generation** fields:

```
  annotations:
    machine.openshift.io/instance-state: running
  ...
  generation: 2
```

vi. Remove the **metadata.resourceVersion** and **metadata.uid** fields:

```
  resourceVersion: "13291"
  uid: a282eb70-40a2-4e89-8009-d05dd420d31a
```

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.

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 the 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. 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. 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 (also known as the master nodes) have a functioning etcd pod.

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 --cluster
```

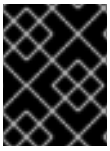
Example output

```
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
```

5.3. DISASTER RECOVERY

5.3.1. 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 in order to return your cluster to a working state.



IMPORTANT

Disaster recovery requires you to have at least one healthy control plane host (also known as the master 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.

5.3.2. 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.

5.3.2.1. 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.

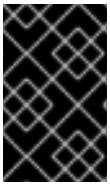
Restoring etcd effectively takes a cluster back in time and all clients will experience a conflicting, parallel history. This can impact the behavior of watching components like kubelets, Kubernetes controller managers, SDN controllers, and persistent volume controllers.

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.

5.3.2.2. 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 (also known as the master hosts).



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 4.6.2 cluster must use an etcd backup that was taken from 4.6.2.

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**.

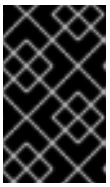
IMPORTANT

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.

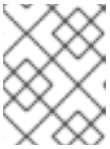
   

   IMPORTANT

   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   v1.23.3+e419edf
   host-172-25-75-38   Ready   infra,worker  3d20h   v1.23.3+e419edf
   host-172-25-75-40   Ready   master        3d20h   v1.23.3+e419edf
   host-172-25-75-65   Ready   master        3d20h   v1.23.3+e419edf
   host-172-25-75-74   Ready   infra,worker  3d20h   v1.23.3+e419edf
   host-172-25-75-79   Ready   worker        3d20h   v1.23.3+e419edf
   host-172-25-75-86   Ready   worker        3d20h   v1.23.3+e419edf
   host-172-25-75-98   Ready   infra,worker  3d20h   v1.23.3+e419edf
   ```

   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.

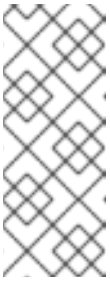
```
$ sudo crictl ps | grep etcd | grep -v operator
```

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.

12. 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.

13. 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.

14. 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.

15. 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.

5.3.2.3. 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*).

Additional resources

- See Accessing the hosts for how to create a bastion host to access OpenShift Container Platform instances and the control plane nodes with SSH.

5.3.3. Recovering from expired control plane certificates

5.3.3.1. 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>
```