



Red Hat Advanced Cluster Management for Kubernetes 2.11

Business continuity

Business continuity

Red Hat Advanced Cluster Management for Kubernetes 2.11 Business continuity

Business continuity

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Read to learn more about restoring your clusters, disaster recovery, and more.

Table of Contents

CHAPTER 1. BUSINESS CONTINUITY	4
1.1. BACKUP AND RESTORE	4
1.1.1. Backup and restore operator architecture	5
1.1.1.1. Resources that are backed up	5
1.1.1.2. Backup files created by an Red Hat Advanced Cluster Management schedule	7
1.1.1.3. Resources restored at managed clusters activation time	8
1.1.2. Configuring active-passive hub cluster	9
1.1.2.1. Active-passive configuration	9
1.1.2.2. Disaster recovery	10
1.1.2.3. Additional resources	11
1.1.3. Installing the backup and restore operator	11
1.1.3.1. Setting up hub clusters for backup and restore operators	11
1.1.3.1.1. Creating the storage location secret	12
1.1.3.1.2. Enabling the backup operator	12
1.1.3.1.2.1. Prerequisite	12
1.1.3.1.2.2. Running a restore operation on a hub cluster with a newer version	13
1.1.3.1.3. Installing the OADP operator	13
1.1.3.1.4. Installing a custom OADP version	14
1.1.3.1.5. Creating a DataProtectionApplication resource	14
1.1.3.1.6. Enabling the backup and restore component in a disconnected environment	15
1.1.3.2. Enabling the backup and restore operator	16
1.1.3.3. Additional resources	17
1.1.4. Scheduling and restoring backups	17
1.1.4.1. Extending backup data	18
1.1.4.2. Scheduling a cluster backup	19
1.1.4.2.1. Avoiding backup collisions	20
1.1.4.3. Additional resources	22
1.1.5. Restoring a backup	22
1.1.5.1. Restoring data back to the initial primary hub	24
1.1.5.2. Preparing the new hub cluster	24
1.1.5.3. Cleaning the hub cluster after restore	25
1.1.5.4. Restoring passive resources while checking for backups	26
1.1.5.5. Restoring passive resources	26
1.1.5.6. Restoring activation resources	27
1.1.5.7. Restoring managed cluster activation data	27
1.1.5.8. Restoring all resources	27
1.1.5.9. Restoring imported managed clusters	28
1.1.5.10. Using other restore samples	28
1.1.5.11. Viewing restore events	29
1.1.5.12. Additional resources	31
1.1.6. Connecting clusters automatically by using a Managed Service Account	31
1.1.6.1. Enabling automatic import	31
1.1.6.2. Automatic import considerations	33
1.1.6.3. Disabling automatic import	33
1.1.6.4. Additional resources	34
1.1.7. Validating your backup or restore configurations	34
1.1.7.1. Protecting data using server-side encryption	37
1.1.7.2. Additional resources	37
1.1.8. Backup and restore advanced configuration	37
1.1.8.1. Resource requests and limits customization	37
1.1.8.2. Additional resources	38

1.2. VOLSYNC PERSISTENT VOLUME REPLICATION SERVICE	38
1.2.1. Replicating persistent volumes with VolSync	39
1.2.1.1. Prerequisites	39
1.2.1.2. Installing VolSync on the managed clusters	39
1.2.1.2.1. Installing VolSync by using labels	40
1.2.1.2.2. Installing VolSync by using a ManagedClusterAddOn	40
1.2.1.2.3. Updating the VolSync ManagedClusterAddOn	41
1.2.1.3. Configuring an Rsync-TLS replication	41
1.2.1.3.1. Configuring Rsync-TLS replication across managed clusters	41
1.2.1.4. Configuring an Rsync replication	45
1.2.1.4.1. Configuring Rsync replication across managed clusters	46
1.2.1.5. Configuring a restic backup	50
1.2.1.5.1. Restoring a restic backup	51
1.2.1.6. Configuring an Rclone replication	52
1.2.1.7. Additional resources	54
1.2.2. Converting a replicated image to a usable persistent volume claim	55
1.2.3. Scheduling your synchronization	56
1.2.4. VolSync advanced configuration	56
1.2.4.1. Creating a secret for Rsync-TLS replication	56

CHAPTER 1. BUSINESS CONTINUITY

See the following topics for disaster recovery solutions and for the hub cluster, as well as managed clusters.

- [Backup and restore](#)
 - [Backup and restore operator architecture](#)
 - [Configuring active passive hub cluster](#)
 - [Installing the backup and restore operator](#)
 - [Scheduling and restoring backups](#)
- [Replicating persistent volumes with VolSync](#)
 - [Replicating persistent volumes with VolSync](#)
 - [Converting a replicated image to a usable persistent volume claim](#)
 - [Scheduling your synchronization](#)

1.1. BACKUP AND RESTORE

The cluster backup and restore operator runs on the hub cluster and provides disaster recovery solutions for Red Hat Advanced Cluster Management for Kubernetes hub cluster failures. When the hub cluster fails, some features, such as policy configuration-based alerting or cluster updates stop working, even if all managed clusters still work. Once the hub cluster is unavailable, you need a recovery plan to decide if recovery is possible, or if the data needs to be recovered on a newly deployed hub cluster.

The backup and restore component provides a set of policies for reporting issues with the backup and restore component, such as backup pods or backup schedules not running. You can use the policy templates to create an alert to the hub cluster administrator if the backup solution is not functioning as expected. For more information, see *Validating your backup or restore configurations*.

The cluster backup and restore operator depends on the [OADP Operator](#) to install Velero, and to create a connection from the hub cluster to the backup storage location where the data is stored. Velero is the component that runs the backup and restore operations. The cluster backup and restore operator solution provides backup and restore support for all Red Hat Advanced Cluster Management hub cluster resources, including managed clusters, applications, and policies.

The cluster backup and restore operator supports backups of any third-party resources that extend the hub cluster installation. With this backup solution, you can define cron-based backup schedules which run at specified time intervals. When the hub cluster fails, a new hub cluster can be deployed and the backed up data is moved to the new hub cluster.

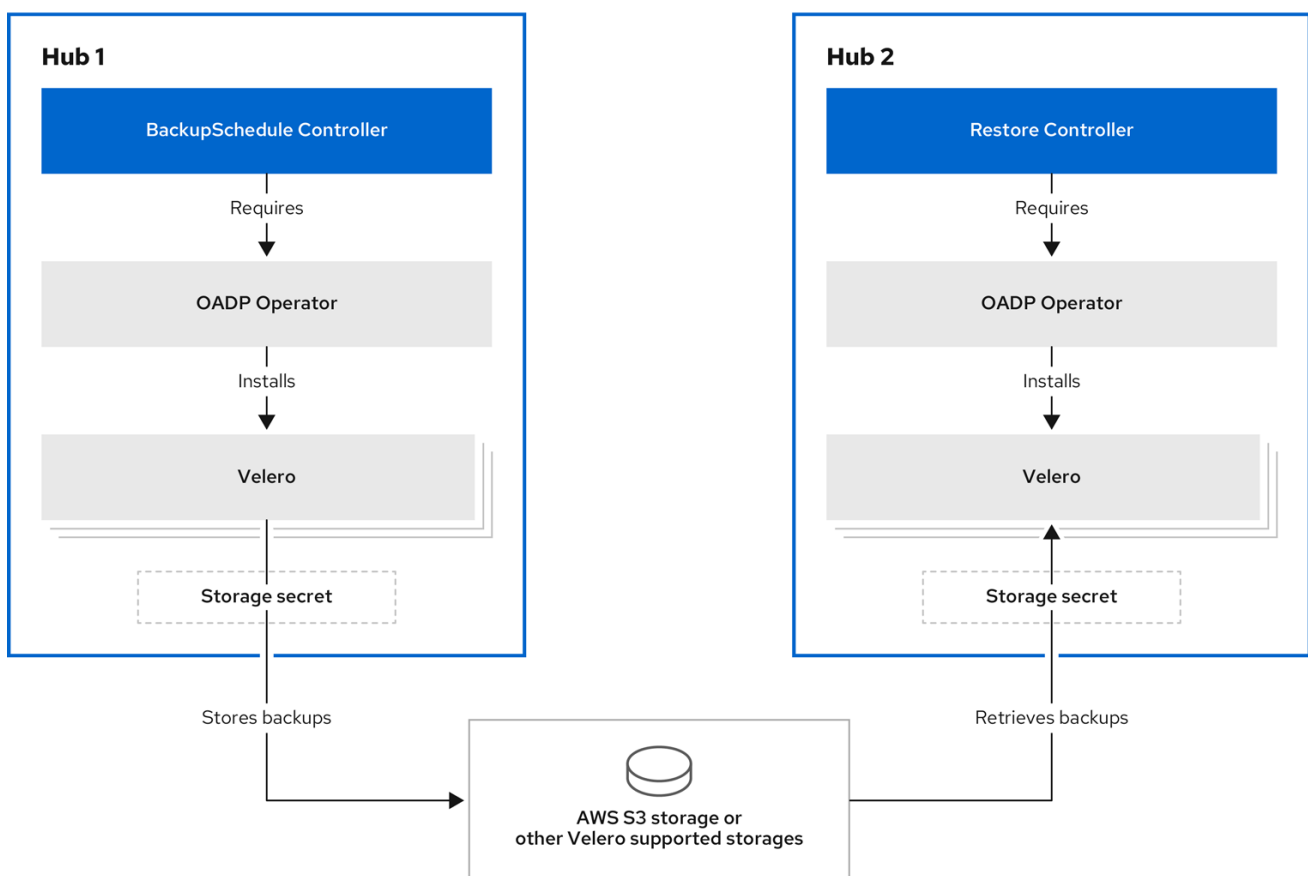
Continue reading the following topics to learn more about the backup and restore operator:

- [Backup and restore operator architecture](#)
- [Configuring active passive hub cluster](#)
- [Installing the backup and restore operator](#)
- [Scheduling and restoring backups](#)

- [Restoring a backup](#)
- [Validating your backup or restore configurations](#)
- [Connecting clusters automatically by using a Managed Service Account](#)
- [Backup and restore advanced configuration](#)

1.1.1. Backup and restore operator architecture

The operator defines the **BackupSchedule.cluster.open-cluster-management.io** resource, which is used to set up Red Hat Advanced Cluster Management backup schedules, and the **restore.cluster.open-cluster-management.io** resource, which is used to process and restore these backups. The operator creates corresponding Velero resources, and defines the options needed to back up remote clusters and any other hub cluster resources that need to be restored. View the following diagram:



235_RHACM_0422

1.1.1.1. Resources that are backed up

The cluster backup and restore operator solution provides backup and restore support for all hub cluster resources such as managed clusters, applications, and policies. You can use the solution to back up any third-party resources extending the basic hub cluster installation. With this backup solution, you can define a cron-based backup schedule, which runs at specified time intervals and continuously backs up the latest version of the hub cluster content.

When the hub cluster needs to be replaced or is in a disaster scenario when the hub cluster fails, a new hub cluster can be deployed and backed up data is moved to the new hub cluster.

View the following list of the cluster backup and restore process, and how they back up or exclude resources and groups to identify backup data:

- Exclude all resources in the **MultiClusterHub** namespace. This is to avoid backing up installation resources that are linked to the current hub cluster identity and should not be backed up.
- Back up all resources with an API version suffixed by **.open-cluster-management.io** and **.hive.openshift.io**. These suffixes indicate that all Red Hat Advanced Cluster Management resources are backed up.
- Back up all resources from the following API groups: **argoproj.io**, **app.k8s.io**, **core.observatorium.io**, **hive.openshift.io**. The resources are backed up within the **acm-resources-schedule** backup, with the exception of the resources from the **agent-install.openshift.io** API group. Resources from the **agent-install.openshift.io** API group are backed up within the **acm-managed-clusters-schedule** backup. The following resources from **hive.openshift.io** and **hiveinternal.openshift.io** API groups are also backed up by the **acm-managed-clusters-schedule** backup:
 - **clusterdeployment.hive.openshift.io**
 - **machinepool.hive.openshift.io**
 - **clusterpool.hive.openshift.io**
 - **clusterclaim.hive.openshift.io**
 - **clusterimageset.hive.openshift.io**
 - **clustersync.hiveinternal.openshift.io**
- Exclude all resources from the following API groups:
 - **internal.open-cluster-management.io**
 - **operator.open-cluster-management.io**
 - **work.open-cluster-management.io**
 - **search.open-cluster-management.io**
 - **admission.hive.openshift.io**
 - **proxy.open-cluster-management.io**
 - **action.open-cluster-management.io**
 - **view.open-cluster-management.io**
 - **clusterview.open-cluster-management.io**
 - **velero.io**
- Exclude all of the following resources that are a part of the included API groups, but are either not needed or are being recreated by owner resources that are also backed up:
 - **clustermanagementaddon.addon.open-cluster-management.io**
 - **backupschedule.cluster.open-cluster-management.io**

- **restore.cluster.open-cluster-management.io**
- **clusterclaim.cluster.open-cluster-management.io**
- **discoveredcluster.discovery.open-cluster-management.io**
- Back up secrets and ConfigMaps with one of the following labels: **cluster.open-cluster-management.io/type**, **hive.openshift.io/secret-type**, **cluster.open-cluster-management.io/backup**.
- Use the **cluster.open-cluster-management.io/backup** label for any other resources that you want to be backed up and are not included in the previously mentioned criteria or are part of the excluded API groups. See the following example:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    cluster.open-cluster-management.io/backup: ""
```

Note: Secrets used by the **hive.openshift.io.ClusterDeployment** resource need to be backed up, and are automatically annotated with the **cluster.open-cluster-management.io/backup** label only when you create the cluster by using the console. If you deploy the Hive cluster by using GitOps instead, you must manually add the **cluster.open-cluster-management.io/backup** label to the secrets used by the **ClusterDeployment** resource. Secret and config map resources with the **cluster.open-cluster-management.io/backup: cluster-activation** label are restored at cluster activation time.

- Exclude specific resources that you do not want backed up. See the following example to exclude Velero resources from the backup process:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    velero.io/exclude-from-backup: "true"
```

1.1.1.2. Backup files created by an Red Hat Advanced Cluster Management schedule

You can use an Red Hat Advanced Cluster Management schedule to backup hub resources, which are grouped in separate backup files, based on the resource type or label annotations.

The **BackupSchedule.cluster.open-cluster-management.io** resource creates a set of four **schedule.velero.io** resources. These **schedule.velero.io** resources generate the backup files, which are also called resources.

To view the list of scheduled backup files, run the following command: **oc get schedules -A | grep acm**.

The scheduled backup files are **backup.velero.io**. See the following table to view the descriptions of these scheduled backup files:

Table 1.1. Scheduled backup table

Scheduled backup	Description
<i>Credentials backup</i>	Stores the following: Hive credentials, Red Hat Advanced Cluster Management, user-created credentials, and ConfigMaps . The name of this backup file is acm-credentials-schedule- <timestamp> .
<i>Resources backup</i>	Contains one backup for the Red Hat Advanced Cluster Management resources, acm-resources-schedule- <timestamp> backup and one for generic resources, acm-resources-generic-schedule- <timestamp> . Any resources annotated with the backup label, cluster.open-cluster-management.io/backup , are stored under the backup, acm-resources-generic-schedule- <timestamp> backup . The exceptions are Secrets or ConfigMap resources which are stored under the backup acm-credentials-schedule- <timestamp> .
<i>Managed clusters backup</i>	Contains only resources that activate the managed cluster connection to the hub cluster, where the backup is restored. The name of this backup file is acm-managed-clusters-schedule- <timestamp> .

1.1.1.3. Resources restored at managed clusters activation time

When you add the **cluster.open-cluster-management.io/backup** label to a resource, the resource is automatically backed up in the **acm-resources-generic-schedule** backup. You must set the label value to **cluster-activation** if any of the resources need to be restored only after the managed clusters are moved to the new hub cluster and when the **veleroManagedClustersBackupName:latest** is used on the restored resource. This ensures that the resource is not restored unless the managed cluster activation is called. View the following example:

```
apiVersion: my.group/v1alpha1
kind: MyResource
metadata:
  labels:
    cluster.open-cluster-management.io/backup: cluster-activation
```

Note: For any managed cluster namespace, or any resource in it, you must restore either one at the cluster activation step. Therefore, if you need to add to the backup resource created in the managed cluster namespace, then use the **cluster-activation** value for the **cluster.open-cluster-management.io/backup** label. To understand the restore process, see the following information:

- If you restore the namespace, then the **managedcluster-import-controller** deletes the namespace.
- If you restore the **managedCluster** custom resource, then the **cluster-manager-registration-controller** creates the namespace.

Aside from the activation data resources that are identified by using the **cluster.open-cluster-management.io/backup: cluster-activation** label and stored by the **acm-resources-generic-schedule** backup, the cluster backup and restore operator includes a few resources in the activation set by default. The following resources are backed up by the **acm-managed-clusters-schedule** backup:

- **managedcluster.cluster.open-cluster-management.io**
- **klusterletaddonconfig.agent.open-cluster-management.io**
- **managedclusteraddon.addon.open-cluster-management.io**
- **managedclusterset.cluster.open-cluster-management.io**
- **managedclusterset.clusterview.open-cluster-management.io**
- **managedclustersetbinding.cluster.open-cluster-management.io**
- **clusterpool.hive.openshift.io**
- **clusterclaim.hive.openshift.io**
- **clustercurator.cluster.open-cluster-management.io**

1.1.2. Configuring active-passive hub cluster

Learn how to configure an active-passive hub cluster configuration, where the initial hub cluster backs up data and one or more passive hub clusters are on stand-by to control the managed clusters when the active cluster becomes unavailable.

1.1.2.1. Active-passive configuration

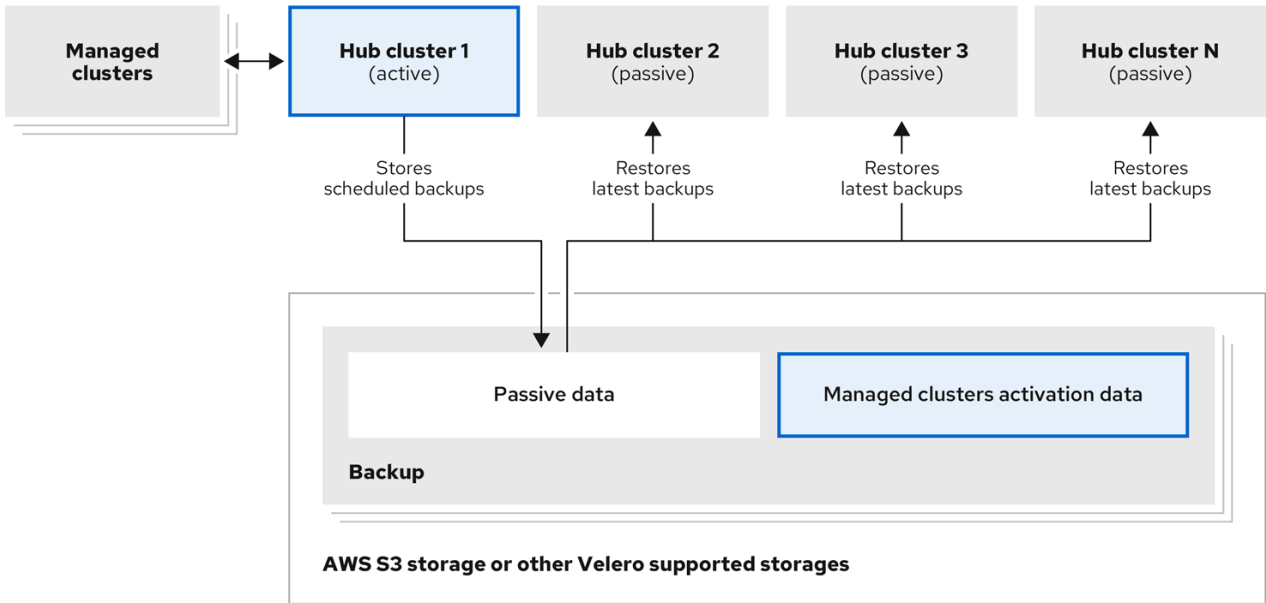
In an active-passive configuration, there is one active hub cluster and passive hub clusters. An active hub cluster is also considered the primary hub cluster, which manages clusters and backs up resources at defined time intervals, using the **BackupSchedule.cluster.open-cluster-management.io** resource.

Note: To backup the primary hub cluster data, you do not need an **active-passive** configuration. You can simply backup and store the hub cluster data. This way, if there is an issue or failure, you can deploy a new hub cluster and restore your primary hub cluster data on this new hub cluster. To reduce the time to recover the primary hub cluster data, you can use an **active-passive** configuration; however, this is not necessary.

Passive hub clusters continuously retrieve the latest backups and restore the passive data. Passive hubs use the **Restore.cluster.open-cluster-management.io** resource to restore passive data from the primary hub cluster when new backup data is available. These hub clusters are on standby to become a primary hub when the primary hub cluster fails.

Active and passive hub clusters are connected to the same storage location, where the primary hub cluster backs up data for passive hub clusters to access the primary hub cluster backups. For more details on how to set up this automatic restore configuration, see *Restoring passive resources while checking for backups*.

In the following diagram, the active hub cluster manages the local clusters and backs up the hub cluster data at regular intervals:

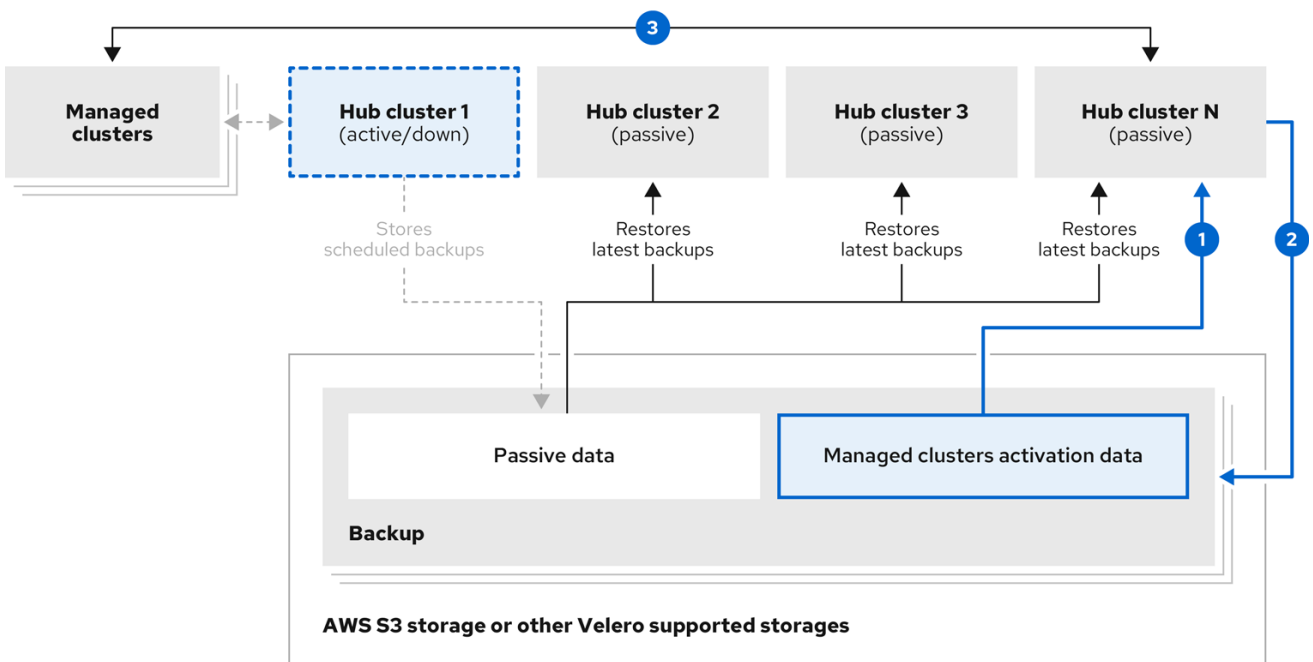


235_RHACM_0422

The passive hub cluster restores this data, except for the managed cluster activation data, which moves the managed clusters to the passive hub cluster. The passive hub clusters can restore the passive data continuously. Passive hub clusters can restore passive data as a one-time operation. See *Restoring passive resources* for more details.

1.1.2.2. Disaster recovery

When the primary hub cluster fails, as a hub administrator, you can select a passive hub cluster to take over the managed clusters. In the following *Disaster recovery diagram*, see how you can use *Hub cluster N* as the new primary hub cluster:



- 1 Activates hub cluster N Restores managed clusters activation data
- 2 Becomes active Stores scheduled backups
- 3 Managed clusters connect to new hub N

235_RHACM_0422

Hub cluster N restores the managed cluster activation data. The managed clusters connect to *Hub cluster N*. You activate a backup on the new primary hub cluster, *Hub cluster N*, by creating a **BackupSchedule.cluster.open-cluster-management.io** resource, and storing the backups at the same storage location as the initial primary hub cluster.

All other passive hub clusters now restore passive data by using the backup data that is created by the new primary hub cluster. *Hub N* is now the primary hub cluster, managing clusters and backing up data.

Important:

- The first process in the earlier *Disaster recovery diagram* is not automated because of the following reasons:
 - You must decide if the primary hub cluster has failed and needs to be replaced, or if there is a network communication error between the hub cluster and the managed clusters.
 - You must decide which passive hub cluster becomes the primary hub cluster. The policy integration with Red Hat Ansible Automation Platform jobs can help you automate this step by making a job run when the backup policy reports backup errors.
- The second process in the earlier *Disaster recovery diagram* is manual. If you did not create a backup schedule on the new primary hub cluster, the **backup-restore-enabled** policy shows a violation by using the **backup-schedule-cron-enabled** policy template. In this second process, you can do the following actions:
 - Use the **backup-schedule-cron-enabled** policy template to validate if the new primary hub cluster has backups running as a cron job.
 - Use the policy integration with **Ansible** and define an **Ansible** job that can run when the **backup-schedule-cron-enabled** policy template reports violations.
- For more details about the **backup-restore-enabled** policy templates, see [Validating your backup or restore configurations](#).

1.1.2.3. Additional resources

- See [Restoring passive resources while checking for backups](#).
- See [Restoring passive resources](#).

1.1.3. Installing the backup and restore operator

The cluster backup and restore operator is not installed automatically. Continue reading to learn how to install and enable the operator.

1.1.3.1. Setting up hub clusters for backup and restore operators

To use the backup and restore operator, you must set up hub clusters. To set up hub clusters, complete the following sections in order:

- [Creating the storage location secret](#)
- [Enabling the backup operator](#)
- [Installing the OADP operator](#)
- [Creating a **DataProtectionApplication** resource](#)

- [Enabling the backup and restore component in a disconnected environment](#)

1.1.3.1.1. Creating the storage location secret

To create a storage location secret, complete the following steps:

1. Complete the steps for [Creating a default Secret](#) for the cloud storage where the backups are saved.
2. Create the secret resource in the OADP Operator namespace, which is located in the backup component namespace.

1.1.3.1.2. Enabling the backup operator

Enable the backup operator to use the disaster recovery solutions for Red Hat Advanced Cluster Management for Kubernetes.

1.1.3.1.2.1. Prerequisite

- From your Red Hat OpenShift Container Platform cluster, install the Red Hat Advanced Cluster Management operator version 2.11. The **MultiClusterHub** resource is automatically created when you install Red Hat Advanced Cluster Management, and displays the following status: **Running**. ===== Enabling backup operators for your hub clusters

To enable the backup operator for your active and passive hub clusters, complete the following steps:

1. Enable the cluster backup and restore operator, **cluster-backup**, by completing the following steps:
 - a. Update the **MultiClusterHub** resource by setting the **cluster-backup** parameter to **true**. If this hub cluster does not have the AWS Security Token Service (STS) option enabled, the OADP operator is also installed in the same namespace with the backup component.
 - b. For hub clusters where the STS option is enabled, you must manually install the OADP operator.
2. Ensure that the restore hub cluster uses the same Red Hat Advanced Cluster Management version that the backup hub cluster uses. You cannot restore a backup on a hub cluster with a version earlier than the one used by the backup hub cluster.
3. Manually configure the restore hub cluster by completing the following steps:
 - a. Install all operators that are installed on the active hub cluster and in the same namespace as the active hub cluster.
 - b. Verify that the new hub cluster is configured the same way as the backup hub cluster.
 - c. Use the same namespace name as the backup hub cluster when you install the backup and restore operator and any operators that are configured on the backup hub cluster.
4. Create the **DataProtectionApplication** resource on the passive hub cluster by completing the following steps:
 - a. Create the **DataProtectionApplication** resource on both the active and passive hub cluster.

- b. For the restore hub cluster, when you create the **DataProtectionApplication** resource, use the same storage location as the backup hub.

1.1.3.1.2.2. Running a restore operation on a hub cluster with a newer version

If you want to run the restore operation on a hub cluster that uses a newer Red Hat Advanced Cluster Management version than the hub cluster where the backup was created, you can still run a restore operation by completing the following steps:

1. On the restore hub cluster, install the operator at the same version as the hub cluster where the backup was created.
2. Restore the backup data on the restore hub cluster.
3. On the restore hub cluster, use the upgrade operation to upgrade to the version you want to use.

1.1.3.1.3. Installing the OADP operator

For hub clusters where you enabled the AWS Security Token Service (STS) option, the OADP operator is not installed when you installed the cluster backup and restore operator on the hub cluster. To pass the STS **role_arn** to the OADP operator, you must manually install the OADP operator.

For hub clusters where you did not enable the STS option, the OADP operator is automatically installed when you install the cluster backup and restore operator on the hub cluster.

When you enable the backup option from the **MultiClusterHub** resource, the backup option remains in the **Pending** state until you install the OADP operator in the **open-cluster-management-backup** namespace. If you have not enabled the AWS STS option on your hub cluster, then the OADP operator is automatically installed with the backup chart. Otherwise, you must manually install the OADP operator.

When the cluster is in the STS mode, the backup chart creates a config map named **acm-redhat-oadp-operator-subscription** in the **open-cluster-management-backup** namespace. This config map also contains the OADP channel and environment information. Use this config map to get the OADP version that you need to install and to get any environment options to be set on the OADP subscription.

To learn more about installing the OADP operator with the STS option, see [Installing OADP on ROSA with STS](#).

Important:

When you install the OADP operator, see the following notes:

- Custom resource definitions are cluster-scoped, so you cannot have two different versions of OADP or Velero installed on the same cluster. If you have two different versions, one version runs with the wrong custom resource definitions.
- When you create the **MultiClusterHub** resource, the Velero custom resource definitions (CRDs) do not automatically install on the hub cluster. When you install the OADP operator, the Velero CRDs are installed on the hub cluster. As a result, the Velero CRD resources are no longer reconciled and updated by the **MultiClusterHub** resource.
- The backup component works with the OADP Operator that is installed in the component namespace.

- If you manually install the OADP operator, the custom resource definition version of the OADP operator and Velero must exactly match. If these versions do not exactly match one another, you get problems.
- Velero is installed with the OADP Operator on the Red Hat Advanced Cluster Management for Kubernetes hub cluster. It is used to backup and restore Red Hat Advanced Cluster Management hub cluster resources.
- For a list of supported storage providers for Velero, see [About installing OADP](#).

1.1.3.1.4. Installing a custom OADP version

Setting an annotation on your **MultiClusterHub** resource allows you to customize the OADP version for your specific needs. Complete the following steps:

1. To override the OADP version installed by the backup and restore operator, use the following annotation on the **MultiClusterHub** resource:

```
installer.open-cluster-management.io/oadp-subscription-spec: '{"channel": "stable-1.4"}'
```

Important: If you use this option to get a different OADP version than the one installed by default by the backup and restore operator, make sure this version is supported on the Red Hat OpenShift Container Platform version used by the hub cluster. Use this override option with caution because it can result in unsupported configurations.

2. For example, install the OADP 1.5 version by setting your annotation. Your **MultiClusterHub** resource might resemble the following example:

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterHub
metadata:
  annotations:
    installer.open-cluster-management.io/oadp-subscription-spec: '{"channel": "stable-1.5","installPlanApproval": "Automatic","name": "redhat-oadp-operator","source": "redhat-operators","sourceNamespace": "openshift-marketplace"}'
  name: multiclusterhub
spec: {}
```

3. Enable the **cluster-backup** option on the **MultiClusterHub** resource by setting **cluster-back** to **true**.

1.1.3.1.5. Creating a DataProtectionApplication resource

To create an instance of the **DataProtectionApplication** resource for your active and passive hub clusters, complete the following steps:

1. From the Red Hat OpenShift Container Platform console, select **Operators > Installed Operators**.
2. Click **Create instance** under DataProtectionApplication.
3. Create the Velero instance by selecting configurations using the OpenShift Container Platform console or by using a YAML file as mentioned in the **DataProtectionApplication** example.
4. Set the **DataProtectionApplication** namespace to **open-cluster-management-backup**.

- Set the specification (**spec:**) values appropriately for the **DataProtectionApplication** resource. Then click **Create**.

If you intend on using the default backup storage location, set the following value, **default: true** in the **backupStorageLocations** section. View the following **DataProtectionApplication** resource sample:

```

apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: dpa-sample
spec:
  configuration:
    velero:
      defaultPlugins:
        - openshift
        - aws
    restic:
      enable: true
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: my-bucket
          prefix: my-prefix
        config:
          region: us-east-1
          profile: "default"
        credential:
          name: cloud-credentials
          key: cloud
  snapshotLocations:
    - name: default
      velero:
        provider: aws
        config:
          region: us-west-2
          profile: "default"

```

1.1.3.1.6. Enabling the backup and restore component in a disconnected environment

To enable the backup and restore component with Red Hat OpenShift Container Platform in a disconnected environment, complete the following steps:

- Update the **MultiClusterHub** resource with the following annotation to override the source from which the OADP operator is installed. Create the annotation before the **cluster-backup** component is enabled on the **MultiClusterHub** resource:

```

apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterHub
metadata:
  annotations:
    installer.open-cluster-management.io/oadp-subscription-spec: '{"source": "redhat-operator-index"}'

```

- The **redhat-operator-index** is a custom name and represents the name of the **CatalogSource** resource that you define and use to access Red Hat OpenShift Operators in the disconnected environment. Run the following command to retrieve the **catalogsource**:

```
oc get catalogsource -A
```

The output might resemble the following:

NAMESPACE	NAME	DISPLAY	TYPE	PUBLISHER	AGE
openshift-marketplace	acm-custom-registry	Advanced Cluster Management	grpc	Red Hat	42h
openshift-marketplace	multiclusterengine-catalog	MultiCluster Engine	grpc	Red Hat	42h
openshift-marketplace	redhat-operator-index		grpc		42h

1.1.3.2. Enabling the backup and restore operator

The cluster backup and restore operator can be enabled when the **MultiClusterHub** resource is created for the first time. The **cluster-backup** parameter is set to **true**. When the operator is enabled, the operator resources are installed.

If the **MultiClusterHub** resource is already created, you can install or uninstall the cluster backup operator by editing the **MultiClusterHub** resource. Set **cluster-backup** to **false**, if you want to uninstall the cluster backup operator.

When the backup and restore operator is enabled, your **MultiClusterHub** resource might resemble the following YAML file:

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterHub
metadata:
  name: multiclusterhub
  namespace: open-cluster-management
spec:
  availabilityConfig: High
  enableClusterBackup: false
  imagePullSecret: multiclusterhub-operator-pull-secret
  ingress:
    sslCiphers:
      - ECDHE-ECDSA-AES256-GCM-SHA384
      - ECDHE-RSA-AES256-GCM-SHA384
      - ECDHE-ECDSA-AES128-GCM-SHA256
      - ECDHE-RSA-AES128-GCM-SHA256
  overrides:
    components:
      - enabled: true
        name: multiclusterhub-repo
      - enabled: true
        name: search
      - enabled: true
        name: management-ingress
      - enabled: true
        name: console
      - enabled: true
```

```

name: insights
- enabled: true
name: grc
- enabled: true
name: cluster-lifecycle
- enabled: true
name: volsync
- enabled: true
name: multicluster-engine
- enabled: true
name: cluster-backup
separateCertificateManagement: false

```

1.1.3.3. Additional resources

- See [Velero](#).
- See [AWS S3 compatible backup storage providers](#) in the OpenShift Container Platform documentation for a list of supported Velero storage providers.
- Learn more about the [DataProtectionApplication](#) resource.

1.1.4. Scheduling and restoring backups

Complete the following steps to schedule and restore backups:

1. Use the backup and restore operator, **backupschedule.cluster.open-cluster-management.io**, to create a backup schedule and use the **restore.cluster.open-cluster-management.io** resources to restore a backup.
2. Run the following command to create a **backupschedule.cluster.open-cluster-management.io** resource:

```
oc create -f cluster_v1beta1_backupschedule.yaml
```

Your **cluster_v1beta1_backupschedule.yaml** resource might resemble the following file:

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm
  namespace: open-cluster-management-backup
spec:
  veleroSchedule: 0 */2 * * * 1
  veleroTtl: 120h 2

```

- 1** Create a backup every 2 hours
- 2** **Optional:** Deletes scheduled backups after 120h. If not specified, the maximum Velero default value of 720h is used.

View the following descriptions of the **backupschedule.cluster.open-cluster-management.io spec** properties:

- **veleroSchedule** is a required property and defines a cron job for scheduling the backups.
 - **veleroTtl** is an optional property and defines the expiration time for a scheduled backup resource. If not specified, the maximum default value set by Velero is used, which is **720h**.
3. Check the status of your **backupschedule.cluster.open-cluster-management.io** resource, which displays the definition for the three **schedule.velero.io** resources. Run the following command:

```
oc get BackupSchedule -n open-cluster-management-backup
```

4. As a reminder, the restore operation is run on a different hub cluster for restore scenarios. To initiate a restore operation, create a **restore.cluster.open-cluster-management.io** resource on the hub cluster where you want to restore backups.

Note: When you restore a backup on a new hub cluster, make sure that you shut down the previous hub cluster where the backup was created. If it is running, the previous hub cluster tries to reimport the managed clusters as soon as the managed cluster reconciliation finds that the managed clusters are no longer available.

You can use the cluster backup and restore operator, **backupschedule.cluster.open-cluster-management.io** and **restore.cluster.open-cluster-management.io** resources, to create a backup or restore resource. See the *cluster-backup-operator* samples.

5. Run the following command to create a **restore.cluster.open-cluster-management.io** resource:

```
oc create -f cluster_v1beta1_backupschedule.yaml
```

Your resource might resemble the following file:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

6. View the Velero **Restore** resource by running the following command:

```
oc get restore.velero.io -n open-cluster-management-backup
```

7. View the Red Hat Advanced Cluster Management **Restore** events by running the following command:

```
oc describe restore.cluster.open-cluster-management.io -n open-cluster-management-backup
```

For descriptions of the parameters and samples of **Restore** YAML resources, see the *Restoring a backup* section.

1.1.4.1. Extending backup data

You can backup third-party resources with cluster backup and restore by adding the **cluster.open-cluster-management.io/backup** label to the resources. The value of the label can be any string, including an empty string. Use a value that can help you identify the component that you are backing up. For example, use the **cluster.open-cluster-management.io/backup: idp** label if the components are provided by an IDP solution.

Note: Use the **cluster-activation** value for the **cluster.open-cluster-management.io/backup** label if you want the resources to be restored when the managed clusters activation resources are restored. Restoring the managed clusters activation resources result in managed clusters being actively managed by the hub cluster, where the restore was started.

1.1.4.2. Scheduling a cluster backup

A backup schedule is activated when you create the **backupschedule.cluster.open-cluster-management.io** resource. View the following **backupschedule.cluster.open-cluster-management.io** sample:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm
  namespace: open-cluster-management-backup
spec:
  veleroSchedule: 0 */2 * * *
  veleroTtl: 120h
```

After you create a **backupschedule.cluster.open-cluster-management.io** resource, run the following command to get the status of the scheduled cluster backups:

```
oc get BackupSchedule -n open-cluster-management-backup
```

The **backupschedule.cluster.open-cluster-management.io** resource creates six **schedule.velero.io** resources, which are used to generate backups. Run the following command to view the list of the backups that are scheduled:

```
oc get schedules -A | grep acm
```

Resources are separately backed up in the groups as seen in the following table:

Table 1.2. Resource groups table

Resource	Description
Credentials backup	Backup file that stores Hive credentials, Red Hat Advanced Cluster Management, and user-created credentials and ConfigMaps.
Resources backup	Contains one backup for the Red Hat Advanced Cluster Management resources and one for generic resources. These resources use the cluster.open-cluster-management.io/backup label.

Resource	Description
ManagedClusters backup	Contains only resources that activate the managed cluster connection to the hub cluster, where the backup is restored.

Note: The *resources backup* file contains managed cluster-specific resources, but does not contain the subset of resources that connect managed clusters to the hub cluster. The resources that connect managed clusters are called activation resources and are contained in the managed clusters backup. When you restore backups only for the *credentials* and *resources* backup on a new hub cluster, the new hub cluster shows all managed clusters that are created by using the Hive API in a detached state. The managed clusters that are imported on the primary hub cluster by using the import operation appear only when the activation data is restored on the passive hub cluster. The managed clusters are still connected to the original hub cluster that created the backup files.

When the activation data is restored, only managed clusters created by using the Hive API are automatically connected with the new hub cluster. All other managed clusters appear in a *Pending* state. You need to manually reattach them to the new cluster.

For descriptions of the various **BackupSchedule** statuses, see the following table:

Table 1.3. BackupSchedule status table

BackupSchedule status	Description
Enabled	The BackupSchedule is running and generating backups.
FailedValidation	An error prevents the BackupSchedule from running. As a result, the BackupSchedule is not generating backups, but instead, it is reconciling and waiting for the error to be fixed. Look at the BackupSchedule status section for the reason why the resource is not valid. After the error is addressed, the BackupSchedule status changes to Enabled , and the resource starts generating backups.
BackupCollision	The BackupSchedule is not generating backups. Look at the BackupSchedule status section for the reason why the resource status is BackupCollision . To start creating backups, delete this resource and create a new one.

1.1.4.2.1. Avoiding backup collisions

Backup collisions might occur if the hub cluster changes status from passive to primary hub cluster, or from primary to passive, and different hub clusters back up data at the same storage location.

As a result, the latest backups are generated by a hub cluster that is no longer set as the primary hub cluster. This hub cluster still produces backups because the **BackupSchedule.cluster.open-cluster-management.io** resource is still enabled.

See the following list to learn about two scenarios that might cause a backup collision:

1. The primary hub cluster fails unexpectedly, which is caused by the following conditions:
 - Communication from the primary hub cluster to the initial hub cluster fails.
 - The initial hub cluster backup data is restored on a secondary hub cluster, called secondary hub cluster.
 - The administrator creates the **BackupSchedule.cluster.open-cluster-management.io** resource on the secondary hub cluster, which is now the primary hub cluster and generates backup data to the common storage location.
 - The initial hub cluster unexpectedly starts working again.
Since the **BackupSchedule.cluster.open-cluster-management.io** resource is still enabled on the initial hub cluster, the initial hub cluster resumes writing backups to the same storage location as the secondary hub cluster. Both hub clusters are now writing backup data at the same storage location. Any hub cluster restoring the latest backups from this storage location might use the initial hub cluster data instead of the secondary hub cluster data.
2. The administrator tests a disaster scenario by making the secondary hub cluster a primary hub cluster, which is caused by the following conditions:
 - The initial hub cluster is stopped.
 - The initial hub cluster backup data is restored on the secondary hub cluster.
 - The administrator creates the **BackupSchedule.cluster.open-cluster-management.io** resource on the secondary hub cluster, which is now the primary hub cluster and generates backup data to the common storage location.
 - After the disaster test is completed, the administrator reverts to the earlier state and makes the initial hub cluster the primary hub cluster again.
 - The initial hub cluster starts while the secondary hub cluster is still active.
Since the **BackupSchedule.cluster.open-cluster-management.io** resource is still enabled on the secondary hub cluster, it writes backups at the same storage location that corrupts the backup data. Any hub cluster restoring the latest backups from this location might use the secondary hub cluster data instead of the initial hub cluster data. In this scenario, stopping the secondary hub cluster first or deleting the **BackupSchedule.cluster.open-cluster-management.io** resource on the secondary hub cluster before starting the initial hub cluster fixes the backup collision issue.
3. The administrator tests a disaster scenario by making the secondary hub cluster a primary hub cluster, without stopping the initial hub cluster first, causing the following conditions:
 - The initial hub cluster is still running.
 - The initial hub cluster backup data is restored on the secondary hub cluster, including managed clusters backup. Therefore, the secondary hub cluster is now the active cluster.
 - Since the **BackupSchedule.cluster.open-cluster-management.io** resource is still enabled on the initial hub cluster, it writes backups at the same storage location which corrupts the backup data. For example, any hub cluster restoring the latest backups from this location

might use the initial hub cluster data instead of the secondary hub cluster data. To avoid data corruption, the initial hub cluster **BackupSchedule** resource status automatically changes to **BackupCollision**. In this scenario, to avoid getting into this backup collision state, stop the initial hub cluster first or delete the **BackupSchedule.cluster.open-cluster-management.io** resource on the initial hub cluster before restoring managed clusters data on the secondary hub cluster.

To avoid and report backup collisions, a **BackupCollision** state exists for the **BackupSchedule.cluster.open-cluster-management.io** resource. The controller checks regularly if the latest backup in the storage location has been generated from the current hub cluster. If not, a different hub cluster has recently written backup data to the storage location, indicating that the hub cluster is colliding with a different hub cluster.

In this case, the current hub cluster **BackupSchedule.cluster.open-cluster-management.io** resource status is set to **BackupCollision** and the **Schedule.velero.io** resources created by this resource are deleted to avoid data corruption. The **BackupCollision** is reported by the backup policy. The administrator verifies which hub cluster writes to the storage location, before removing the **BackupSchedule.cluster.open-cluster-management.io** resource from the invalid hub cluster and creating a new **BackupSchedule.cluster.open-cluster-management.io** resource on the valid primary hub cluster, to resume the backup.

Run the following command to check if there is a backup collision:

```
oc get backupschedule -A
```

If there is a backup collision, the output might resemble the following example:

```

NAMESPACE   NAME           PHASE           MESSAGE
openshift-adp schedule-hub-1 BackupCollision Backup acm-resources-schedule-
20220301234625, from cluster with id [be97a9eb-60b8-4511-805c-298e7c0898b3] is using the same
storage location. This is a backup collision with current cluster [1f30bfe5-0588-441c-889e-
eaf0ae55f941] backup. Review and resolve the collision then create a new BackupSchedule resource
to resume backups from this cluster.
```

1.1.4.3. Additional resources

- See the [Restoring a backup](#) section for descriptions of the parameters and samples of **Restore** YAML resources.
- For more information, see [Backup and restore](#).

1.1.5. Restoring a backup

In a typical restore scenario, the hub cluster where the backups run becomes unavailable, and the backed up data needs to be moved to a new hub cluster. This is done by running the cluster restore operation on the new hub cluster. In this case, the restore operation runs on a different hub cluster than where the backup is created.

There are also cases where you want to restore the data on the same hub cluster where the backup was collected so that you can recover the data from a previous snapshot. In this case, both restore and backup operations run on the same hub cluster.

After you create a **restore.cluster.open-cluster-management.io** resource on the hub cluster, you can run the following command to get the status of the restore operation:

```
oc get restore -n open-cluster-management-backup
```

You can also verify that the backed up resources that are contained by the backup file are created.

Note: The **restore.cluster.open-cluster-management.io** resource runs once unless you use the **syncRestoreWithNewBackups** option and set it to **true**, as mentioned in the [Restore passive resources](#) section. If you want to run the same restore operation again after the restore operation is complete, you must create a new **restore.cluster.open-cluster-management.io** resource with the same **spec** options.

The restore operation is used to restore all 3 backup types that are created by the backup operation. You can choose to install only a certain type of backup, such as only managed clusters, only user credentials, or only hub cluster resources.

The restore defines the following 3 required **spec** properties, where the restore logic is defined for the types of backed up files:

- **veleroManagedClustersBackupName** is used to define the restore option for the managed clusters activation resources.
- **veleroCredentialsBackupName** is used to define the restore option for the user credentials.
- **veleroResourcesBackupName** is used to define the restore option for the hub cluster resources (**Applications**, **Policy**, and other hub cluster resources like managed cluster passive data).

The valid options for the previously mentioned properties are following values:

- **latest** - This property restores the last available backup file for this type of backup.
- **skip** - This property does not attempt to restore this type of backup with the current restore operation.
- **<backup_name>** - This property restores the specified backup pointing to it by name.

The name of the **restore.velero.io** resources that are created by the **restore.cluster.open-cluster-management.io** is generated using the following template rule, **<restore.cluster.open-cluster-management.io name>-<velero-backup-resource-name>**. View the following descriptions:

- **restore.cluster.open-cluster-management.io name** is the name of the current **restore.cluster.open-cluster-management.io** resource, which initiates the restore.
- **velero-backup-resource-name** is the name of the Velero backup file that is used for restoring the data. For example, the **restore.cluster.open-cluster-management.io** resource named **restore-acm** creates **restore.velero.io** restore resources. View the following examples for the format:
 - **restore-acm-acm-managed-clusters-schedule-20210902205438** is used for restoring managed cluster activation data backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-clusters-schedule-20210902205438**.
 - **restore-acm-acm-credentials-schedule-20210902206789** is used for restoring credential backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-clusters-schedule-20210902206789**.
 - **restore-acm-acm-resources-schedule-20210902201234** is used for restoring application, policy, and other hub cluster resources like managed cluster passive data backups. In this sample, the **backup.velero.io** backup name used to restore the resource is **acm-managed-**

clusters-schedule-20210902201234.

Note: If **skip** is used for a backup type, **restore.velero.io** is not created.

View the following YAML sample of the cluster **Restore** resource. In this sample, all three types of backed up files are being restored, using the latest available backed up files:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

Note: Only managed clusters created by the Hive API are automatically connected with the new hub cluster when the **acm-managed-clusters** backup from the managed clusters backup is restored on another hub cluster. All other managed clusters remain in the **Pending Import** state and must be imported back onto the new hub cluster. For more information, see [Restoring imported managed clusters](#).

1.1.5.1. Restoring data back to the initial primary hub

When you need to restore backup data on a cluster, create a new cluster instead of using the primary hub cluster or a hub cluster where user resources were created before the restore operation. During the hub cluster restore operation, you can configure the hub cluster backup restore to clean up the existing resources, if these resources are not part of the backup data being restored. The restore cleans up resources created by an earlier backup but does not clean up user resources. As a result, resources created by the user on this hub cluster are not cleaned up, so the data on this hub cluster is not reflective of the data available with the restored resource.

A disaster recovery test, where you only validate the restore operation on a passive hub, then move back to use the primary hub cluster, is one situation where you can use the primary hub cluster as the passive cluster. In this recovery test scenario, you are only testing the hub backup scenario. You do not use the primary hub cluster to create new resources. Instead, the backup data has temporarily moved from the primary hub cluster to the passive hub cluster. You can make the initial primary hub cluster a primary hub cluster again if you restore all resources to this hub cluster. The operation runs the post restore operation, which moves the managed clusters back to this hub cluster.

1.1.5.2. Preparing the new hub cluster

When you need to restore backup data on a hub cluster, create a new hub cluster instead of using the primary hub cluster or a hub cluster where an earlier user created resources before the restore operation.

In a disaster recovery scenario, when you restore the backup to the passive hub cluster, you want a hub cluster identical to the hub cluster that failed during the disaster. You do not want any other user resources, such as: applications, policies, and extra managed clusters.

When you restore the backup on this hub cluster, if you have managed clusters managed by this hub cluster, then the restore operation might detach them and delete their cluster resources. The restore operation does this to clean up the resources that are not part of the restored backup, making the

content in this hub cluster identical with the initial hub cluster. In this case, you might lose user data because the managed cluster workloads get deleted by detach action.

Before running the restore operation on a new hub cluster, you need to manually configure the hub cluster and install the same operators as on the initial hub cluster. You must install the Red Hat Advanced Cluster Management operator in the same namespace as the initial hub cluster, create the *DataProtectionApplication* resource, and then connect to the same storage location where the initial hub cluster previously backed up data.

Use the same configuration as on the initial hub cluster for the **MultiClusterHub** resource created by the Red Hat Advanced Cluster Management operator, including any changes to the **MultiClusterEngine** resource.

For example, if the initial hub cluster has any other operators installed, such as Ansible Automation Platform, Red Hat OpenShift GitOps, **cert-manager**, you have to install them before running the restore operation. This ensures that the new hub cluster is configured in the same way as the initial hub cluster.

1.1.5.3. Cleaning the hub cluster after restore

Velero updates existing resources if they have changed with the currently restored backup. Velero does not clean up delta resources, which are resources created by a previous restore and not part of the currently restored backup. This limits the scenarios you can use when restoring hub cluster data on a new hub cluster. Unless the restore is applied only once, you cannot reliably use the new hub cluster as a passive configuration. The data on the hub cluster does not reflect the data available with the restored resources.

To address this limitation, when a **Restore.cluster.open-cluster-management.io** resource is created, the backup operator runs a post restore operation that cleans up the hub cluster. The operation removes any resources created by a previous Red Hat Advanced Cluster Management restore that are not part of the currently restored backup.

The post restore cleanup uses the **cleanupBeforeRestore** property to identify the subset of objects to clean up. You can use the following options for the post restore cleanup:

- **None**: No clean up necessary, just begin Velero restore. Use **None** on a brand new hub cluster.
- **CleanupRestored**: Clean up all resources created by a previous Red Hat Advanced Cluster Management restore that are not part of the currently restored backup.
- **CleanupAll**: Clean up all resources on the hub cluster that might be part of a Red Hat Advanced Cluster Management backup, even if they were not created as a result of a restore operation. This is to be used when extra content is created on a hub cluster before the restore operation starts.
Best Practice: Avoid using the **CleanupAll** option. Only use it as a last resort with extreme caution. **CleanupAll** also cleans up resources on the hub cluster created by the user, in addition to resources created by a previously restored backup. Instead, use the **CleanupRestored** option to prevent updating the hub cluster content when the hub cluster is designated as a passive candidate for a disaster scenario. Use a clean hub cluster as a passive cluster.

Notes:

- Velero sets the status, **PartiallyFailed**, for a velero restore resource if the restored backup has no resources. This means that a **restore.cluster.open-cluster-management.io** resource can be in **PartiallyFailed** status if any of the created **restore.velero.io** resources do not restore any resources because the corresponding backup is empty.

- The **restore.cluster.open-cluster-management.io** resource is run once, unless you use the **syncRestoreWithNewBackups:true** to keep restoring passive data when new backups are available. For this case, follow the restore passive with sync sample. See [Restoring passive resources while checking for backups](#). After the restore operation is complete and you want to run another restore operation on the same hub cluster, you have to create a new **restore.cluster.open-cluster-management.io** resource.
- Although you can create multiple **restore.cluster.open-cluster-management.io** resources, only one can be active at any moment in time.

1.1.5.4. Restoring passive resources while checking for backups

Use the **restore-passive-sync** sample to restore passive data, while continuing to check if new backups are available and restore them automatically. To automatically restore new backups, you must set the **syncRestoreWithNewBackups** parameter to **true**. You must also only restore the latest passive data. You can find the sample example at the end of this section.

Set the **VeleroResourcesBackupName** and **VeleroCredentialsBackupName** parameters to **latest**, and the **VeleroManagedClustersBackupName** parameter to **skip**. Immediately after the **VeleroManagedClustersBackupName** is set to **latest**, the managed clusters are activated on the new hub cluster and is now the primary hub cluster.

When the activated managed cluster becomes the primary hub cluster, the restore resource is set to **Finished** and the **syncRestoreWithNewBackups** is ignored, even if set to **true**.

By default, the controller checks for new backups every 30 minutes when the **syncRestoreWithNewBackups** is set to **true**. If new backups are found, it restores the backed up resources. You can change the duration of the check by updating the **restoreSyncInterval** parameter.

For example, see the following resource that checks for backups every 10 minutes:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm-passive-sync
  namespace: open-cluster-management-backup
spec:
  syncRestoreWithNewBackups: true # restore again when new backups are available
  restoreSyncInterval: 10m # check for new backups every 10 minutes
  cleanupBeforeRestore: CleanupRestored
  veleroManagedClustersBackupName: skip
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

1.1.5.5. Restoring passive resources

Use the **restore-acm-passive** sample to restore hub cluster resources in a passive configuration. Passive data is backup data such as secrets, ConfigMaps, applications, policies, and all the managed cluster custom resources, which do not activate a connection between managed clusters and hub clusters. The backup resources are restored on the hub cluster by the credentials backup and restore resources.

See the following sample:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
```

```

kind: Restore
metadata:
  name: restore-acm-passive
  namespace: open-cluster-management-backup
spec:
  cleanupBeforeRestore: CleanupRestored
  veleroManagedClustersBackupName: skip
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest

```

1.1.5.6. Restoring activation resources

Before you restore the activation data on the passive hub cluster, shut down the previous hub cluster where the backup was created. If the primary hub cluster is still running, it attempts to reconnect with the managed clusters that are no longer available, based on the reconciliation procedure running on this hub cluster.

Use the **restore-acm-passive-activate** sample when you want the hub cluster to manage the clusters. In this case it is assumed that the other data has been restored already on the hub cluster that using the passive resource.

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm-passive-activate
  namespace: open-cluster-management-backup
spec:
  cleanupBeforeRestore: CleanupRestored
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: skip
  veleroResourcesBackupName: skip

```

You have some options to restore activation resources, depending on how you restored the passive resources:

- If you used the **restore-acm-passive-sync cluster.open-cluster-management.io** resource as documented in the *Restore passive resources while checking for backups to restore passive data* section, update the **veleroManagedClustersBackupName** value to **latest** on this resource. As a result, the managed cluster resources and the **restore-acm-passive-sync** resource are restored.
- If you restored the passive resources as a one time operation, or did not restore any resources yet, choose to restore all resources as specified in the *Restoring all resources* section.

1.1.5.7. Restoring managed cluster activation data

Managed cluster activation data or other activation data resources are stored by the managed clusters backup and by the resource-generic backups, when you use the **cluster.open-cluster-management.io/backup: cluster-activation** label. When the activation data is restored on a new hub cluster, managed clusters are being actively managed by the hub cluster where the restore is run. See *Scheduling and restoring backups* to learn how you can use the operator.

1.1.5.8. Restoring all resources

Use the **restore-acm** sample if you want to restore all data at once and make the hub cluster manage the managed clusters in one step. After you create a **restore.cluster.open-cluster-management.io** resource on the hub cluster, run the following command to get the status of the restore operation:

```
oc get restore -n open-cluster-management-backup
```

Your sample might resemble the following resource:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  cleanupBeforeRestore: CleanupRestored
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: latest
  veleroResourcesBackupName: latest
```

From your hub cluster, verify that the backed up resources contained by the backup file are created.

1.1.5.9. Restoring imported managed clusters

Only managed clusters connected with the primary hub cluster using the Hive API are automatically connected with the new hub cluster, where the activation data is restored. These clusters have been created on the primary hub cluster by the Hive API, using the **Create cluster** button in the **Clusters** tab or through the CLI. Managed clusters connected with the initial hub cluster using the **Import cluster** button appear as **Pending Import** when the activation data is restored, and must be imported back on the new hub cluster.

The Hive managed clusters can be connected with the new hub cluster because Hive stores the managed cluster **kubeconfig** in the managed cluster namespace on the hub cluster. This is backed up and restored on the new hub cluster. The import controller then updates the bootstrap **kubeconfig** on the managed cluster using the restored configuration, which is only available for managed clusters created using the Hive API. It is not available for imported clusters.

To reconnect imported clusters on the new hub cluster, manually create the **auto-import-secret** resource after you start the restore operation. See *Importing the cluster with the auto import secret* for more details.

Create the **auto-import-secret** resource in the managed cluster namespace for each cluster in **Pending Import** state. Use a **kubeconfig** or token with enough permissions for the import component to start the automatic import on the new hub cluster. You must have access for each managed cluster by using a token to connect with the managed cluster. The token must have a **klusterlet** role binding or a role with the same permissions.

1.1.5.10. Using other restore samples

View the following Restore section to view the YAML examples to restore different types of backed up files.

- Restore all three types of backed up resources:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
```



```

metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  veleroManagedClustersBackupSchedule: latest
  veleroCredentialsBackupSchedule: latest
  veleroResourcesBackupSchedule: latest

```

- Restore only managed cluster resources:

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  veleroManagedClustersBackupName: latest
  veleroCredentialsBackupName: skip
  veleroResourcesBackupName: skip

```

- Restore the resources for managed clusters only, using the **acm-managed-clusters-schedule-20210902205438** backup:

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Restore
metadata:
  name: restore-acm
  namespace: open-cluster-management-backup
spec:
  veleroManagedClustersBackupName: acm-managed-clusters-schedule-20210902205438
  veleroCredentialsBackupName: skip
  veleroResourcesBackupName: skip

```

Notes:

- The **restore.cluster.open-cluster-management.io** resource is run once. After the restore operation is completed, you can optionally run another restore operation on the same hub cluster. You must create a new **restore.cluster.open-cluster-management.io** resource to run a new restore operation.
- You can create multiple **restore.cluster.open-cluster-management.io**, however only one can be run at any moment.

1.1.5.11. Viewing restore events

Use the following command to get information about restore events:

```
oc describe -n open-cluster-management-backup <restore-name>
```

Your list of events might resemble the following sample:

```

Spec:
  Cleanup Before Restore:      CleanupRestored
  Restore Sync Interval:      4m

```

```

Sync Restore With New Backups:    true
Velero Credentials Backup Name:   latest
Velero Managed Clusters Backup Name: skip
Velero Resources Backup Name:    latest
Status:
  Last Message:                   Velero restores have run to completion, restore will continue to sync
  with new backups
  Phase:                           Enabled
  Velero Credentials Restore Name: example-acm-credentials-schedule-20220406171919
  Velero Resources Restore Name:  example-acm-resources-schedule-20220406171920
Events:
  Type   Reason          Age From           Message
  ----   -
Normal Prepare to restore: 76m Restore controller Cleaning up resources for backup acm-
credentials-hive-schedule-20220406155817
Normal Prepare to restore: 76m Restore controller Cleaning up resources for backup acm-
credentials-cluster-schedule-20220406155817
Normal Prepare to restore: 76m Restore controller Cleaning up resources for backup acm-
credentials-schedule-20220406155817
Normal Prepare to restore: 76m Restore controller Cleaning up resources for backup acm-
resources-generic-schedule-20220406155817
Normal Prepare to restore: 76m Restore controller Cleaning up resources for backup acm-
resources-schedule-20220406155817
Normal Velero restore created: 74m Restore controller example-acm-credentials-schedule-
20220406155817
Normal Velero restore created: 74m Restore controller example-acm-resources-generic-
schedule-20220406155817
Normal Velero restore created: 74m Restore controller example-acm-resources-schedule-
20220406155817
Normal Velero restore created: 74m Restore controller example-acm-credentials-cluster-
schedule-20220406155817
Normal Velero restore created: 74m Restore controller example-acm-credentials-hive-schedule-
20220406155817
Normal Prepare to restore: 64m Restore controller Cleaning up resources for backup acm-
resources-schedule-20220406165328
Normal Prepare to restore: 62m Restore controller Cleaning up resources for backup acm-
credentials-hive-schedule-20220406165328
Normal Prepare to restore: 62m Restore controller Cleaning up resources for backup acm-
credentials-cluster-schedule-20220406165328
Normal Prepare to restore: 62m Restore controller Cleaning up resources for backup acm-
credentials-schedule-20220406165328
Normal Prepare to restore: 62m Restore controller Cleaning up resources for backup acm-
resources-generic-schedule-20220406165328
Normal Velero restore created: 61m Restore controller example-acm-credentials-cluster-
schedule-20220406165328
Normal Velero restore created: 61m Restore controller example-acm-credentials-schedule-
20220406165328
Normal Velero restore created: 61m Restore controller example-acm-resources-generic-
schedule-20220406165328
Normal Velero restore created: 61m Restore controller example-acm-resources-schedule-
20220406165328
Normal Velero restore created: 61m Restore controller example-acm-credentials-hive-schedule-
20220406165328
Normal Prepare to restore: 38m Restore controller Cleaning up resources for backup acm-
resources-generic-schedule-20220406171920
Normal Prepare to restore: 38m Restore controller Cleaning up resources for backup acm-

```

resources-schedule-20220406171920

Normal Prepare to restore: 36m Restore controller Cleaning up resources for backup acm-credentials-hive-schedule-20220406171919

Normal Prepare to restore: 36m Restore controller Cleaning up resources for backup acm-credentials-cluster-schedule-20220406171919

Normal Prepare to restore: 36m Restore controller Cleaning up resources for backup acm-credentials-schedule-20220406171919

Normal Velero restore created: 36m Restore controller example-acm-credentials-cluster-schedule-20220406171919

Normal Velero restore created: 36m Restore controller example-acm-credentials-schedule-20220406171919

Normal Velero restore created: 36m Restore controller example-acm-resources-generic-schedule-20220406171920

Normal Velero restore created: 36m Restore controller example-acm-resources-schedule-20220406171920

Normal Velero restore created: 36m Restore controller example-acm-credentials-hive-schedule-20220406171919

1.1.5.12. Additional resources

- See [DataProtectionApplication](#).
- See [Importing the cluster with the auto import secret](#).
- See [Scheduling and restoring backups](#).

1.1.6. Connecting clusters automatically by using a Managed Service Account

The backup controller automatically connects imported clusters to the new hub cluster by using the Managed Service Account component. The Managed Service Account creates a token that is backed up for each imported cluster in each managed cluster namespace. The token uses a **klusterlet-bootstrap-kubeconfig ClusterRole** binding, which allows the token to be used by an automatic import operation. The **klusterlet-bootstrap-kubeconfig ClusterRole** can only get or update the **bootstrap-hub-kubeconfig** secret. To learn more about the Managed Service Account component, see *What is Managed Service Account?*

When the activation data is restored on the new hub cluster, the restore controller runs a post restore operation and looks for all managed clusters in the **Pending Import** state. If a valid token generated by the Managed Service Account is found, the controller creates an **auto-import-secret** using the token. As a result, the import component tries to reconnect the managed cluster. If the cluster is accessible, the operation is successful.

1.1.6.1. Enabling automatic import

The automatic import feature using the Managed Service Account component is disabled by default. To enable the automatic import feature, complete the following steps:

1. Enable the Managed Service Account component by setting the **managedserviceaccount enabled** parameter to **true** in the **MultiClusterEngine** resource. See the following example:

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterhub
spec:
```

```

overrides:
  components:
    - enabled: true
      name: managedserviceaccount

```

2. Enable the automatic import feature for the **BackupSchedule.cluster.open-cluster-management.io** resource by setting the **useManagedServiceAccount** parameter to **true**. See the following example:

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm-msa
  namespace: open-cluster-management-backup
spec:
  veleroSchedule:
  veleroTtl: 120h
  useManagedServiceAccount: true

```

The default token validity duration is set to twice the value of **veleroTtl** to increase the chance of the token being valid for all backups storing the token for their entire lifecycle. In some cases, you might need to control how long a token is valid by setting a value for the optional **managedServiceAccountTTL** property.

Use **managedServiceAccountTTL** with caution if you need to update the default token expiration time for the generated tokens. Changing the token expiration time from the default value might result in producing backups with tokens set to expire during the lifecycle of the backup. As a result, the import feature does not work for the managed clusters.

Important: Do not use **managedServiceAccountTTL** unless you need to control how long the token is valid.

See the following example for using the **managedServiceAccountTTL** property:

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm-msa
  namespace: open-cluster-management-backup
spec:
  veleroSchedule:
  veleroTtl: 120h
  useManagedServiceAccount: true
  managedServiceAccountTTL: 300h

```

After you enable the automatic import feature, the backup component starts processing imported managed clusters by creating the following:

- A **ManagedServiceAddon** named **managed-serviceaccount**.
- A **ManagedServiceAccount** named **auto-import-account**.
- A **ManifestWork** for each **ManagedServiceAccount** to set up a **klusterlet-bootstrap-kubeconfig RoleBinding** for the **ManagedServiceAccount** token on the managed cluster.

The token is only created if the managed cluster is accessible when you create the Managed Service Account, otherwise it is created later once the managed cluster becomes available.

1.1.6.2. Automatic import considerations

The following scenarios can prevent the managed cluster from being automatically imported when moving to a new hub cluster:

- When running a hub backup without a **ManagedServiceAccount** token, for example when you create the **ManagedServiceAccount** resource while the managed cluster is not accessible, the backup does not contain a token to auto import the managed cluster.
- The auto import operation fails if the **auto-import-account** secret token is valid and is backed up but the restore operation is run when the token available with the backup has already expired. The **restore.cluster.open-cluster-management.io** resource reports invalid token issues for each managed cluster.
- Since the **auto-import-secret** created on restore uses the **ManagedServiceAccount** token to connect to the managed cluster, the managed cluster must also provide the kube **apiserver** information. The **apiserver** must be set on the **ManagedCluster** resource. See the following example:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: managed-cluster-name
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
  managedClusterClientConfigs:
    url: <apiserver>
```

When a cluster is imported on the hub cluster, the **apiserver** is only set up automatically on OpenShift Container Platform clusters. You must set the **apiserver** manually on other types of managed clusters, such as EKS clusters, otherwise the automatic import feature ignores the clusters. As a result, the clusters remain in the **Pending Import** state when you move them to the restore hub cluster.

- It is possible that a **ManagedServiceAccount** secret might not be included in a backup if the backup schedule runs before the backup label is set on the **ManagedServiceAccount** secret. **ManagedServiceAccount** secrets don't have the cluster **open-cluster-management.io/backup** label set on creation. For this reason, the backup controller regularly searches for **ManagedServiceAccount** secrets under the managed cluster's namespaces, and adds the backup label if not found.

1.1.6.3. Disabling automatic import

You can disable the automatic import cluster feature by setting the **useManagedServiceAccount** parameter to **false** in the **BackupSchedule** resource. See the following example:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: BackupSchedule
metadata:
  name: schedule-acm-msa
  namespace: open-cluster-management-backup
```

```
spec:
  veleroSchedule:
  veleroTtl: 120h
  useManagedServiceAccount: false
```

The default value is **false**. After setting the value to **false**, the backup operator removes all created resources, including **ManagedServiceAddon**, **ManagedServiceAccount**, and **ManifestWork**. Removing the resources deletes the automatic import token on the hub cluster and managed cluster.

1.1.6.4. Additional resources

- See [What is Managed Service Account?](#) to learn more about the Managed Service Account component.
- Return to [Automatically connecting clusters by using a Managed Service Account](#) .

1.1.7. Validating your backup or restore configurations

When you set the **cluster-backup** option to **true** on the **MultiClusterHub** resource, multicluster engine operator installs the cluster backup and restore operator Helm chart that is named the **cluster-backup-chart**. This chart then installs the **backup-restore-enabled** and **backup-restore-auto-import** policies. Use these policies to view information about issues with your backup and restore components.

Note: A hub cluster is automatically imported and self-managed by as the **local-cluster**. If you disable self-management by setting **disableHubSelfManagement** to **true** on the **MultiClusterHub** resource, the **backup-restore-enabled** policy is not placed on the hub cluster and the policy templates do not produce any reports.

If a hub cluster is managed by a global hub cluster, or if it is installed on a managed cluster instance, the self-management option might get disabled by setting the **disableHubSelfManagement** to **true**. In this instance, you can still enable the **backup-restore-enabled** policy on the hub cluster. Set the **is-hub=true** label on the **ManagedCluster** resource that represents the local cluster.

The **backup-restore-enabled** policy includes a set of templates that check for the following constraints:

- **OADP channel validation**
 - When you enable the backup component on the **MultiClusterHub**, the cluster backup and restore operator Helm chart can automatically install the OADP operator in the **open-cluster-management-backup** namespace. You can also manually install the OADP operator in the namespace.
 - The **OADP-channel** that you selected for manual installation must match or exceed the version set by the Red Hat Advanced Cluster Management backup and restore operator Helm chart.
 - Since the OADP Operator and Velero Custom Resource Definitions (CRDs) are **cluster-scoped**, you cannot have multiple versions on the same cluster. You must install the same version in the **open-cluster-management-backup** namespace and any other namespaces.
 - Use the following templates to check for availability and validate the OADP installation:
 - **oadp-operator-exists:** Use this template to verify if the OADP operator is installed in the **open-cluster-management-backup** namespace.

- **oadp-channel-validation**: Use this template to ensure the OADP operator version in the **open-cluster-management-backup** namespace matches or exceeds the version set by the Red Hat Advanced Cluster Management backup and restore operator.
 - **custom-oadp-channel-validation**: Use this template to check if OADP operators in other namespaces match the version in the **open-cluster-management-backup** namespace.
- **Pod validation**

The following templates check the pod status for the backup component and dependencies:

 - **acm-backup-pod-running** template checks if the backup and restore operator pod is running.
 - **oadp-pod-running** template checks if the OADP operator pod is running.
 - **velero-pod-running** template checks if the Velero pod is running.
- **Data Protection Application validation**
 - **data-protection-application-available** template checks if a **DataProtectionApplication.oadp.openshift.io** resource is created. This OADP resource sets up Velero configurations.
- **Backup storage validation**
 - **backup-storage-location-available** template checks if a **BackupStorageLocation.velero.io** resource is created and if the status value is **Available**. This implies that the connection to the backup storage is valid.
- **BackupSchedule collision validation**
 - **acm-backup-clusters-collision-report** template verifies that the status is not **BackupCollision**, if a **BackupSchedule.cluster.open-cluster-management.io** exists on the current hub cluster. This verifies that the current hub cluster is not in collision with any other hub cluster when you write backup data to the storage location. For a definition of the **BackupCollision**, see [Avoiding backup collisions](#).
- **BackupSchedule and restore status validation**
 - **acm-backup-phase-validation** template checks that the status is not in **Failed**, or **Empty** state, if a **BackupSchedule.cluster.open-cluster-management.io** exists on the current cluster. This ensures that if this cluster is the primary hub cluster and is generating backups, the **BackupSchedule.cluster.open-cluster-management.io** status is healthy.
 - The same template checks that the status is not in a **Failed**, or **Empty** state, if a **Restore.cluster.open-cluster-management.io** exists on the current cluster. This ensures that if this cluster is the secondary hub cluster and is restoring backups, the **Restore.cluster.open-cluster-management.io** status is healthy.
- **Backups exist validation**
 - **acm-managed-clusters-schedule-backups-available** template checks if **Backup.velero.io** resources are available at the location specified by the **BackupStorageLocation.velero.io**, and if the backups are created by a **BackupSchedule.cluster.open-cluster-management.io** resource. This validates that the backups have been run at least once, using the backup and restore operator.

- **Backups for completion**
 - An **acm-backup-in-progress-report** template checks if **Backup.velero.io** resources are stuck in the **InProgress** state. This validation is added because with a large number of resources, the velero pod restarts as the backup runs, and the backup stays in progress without proceeding to completion. During a normal backup, the backup resources are in progress at some point when it is run, but are not stuck and run to completion. It is normal to see the **acm-backup-in-progress-report** template report a warning during the time the schedule is running and backups are in progress.
- **Backup schedules generate backups on the primary hub cluster**
 - The **backup-schedule-cron-enabled** policy template notifies the hub cluster administrator that the primary hub cluster is not creating new backups. The policy shows that the **BackupSchedule.cluster.open-cluster-management.io** resource is disabled on the primary hub cluster or is not able to create scheduled backups.
 - The **backup-schedule-cron-enabled** policy template is in violation if the backup schedule on the primary hub cluster did not generate new backups when the **BackupSchedule** cron job definition expects new backups. This might happen if the **BackupSchedule** does not exist, is paused, or the **BackupSchedule** cron job property updated and the time to run the backups increased. The template is in violation until a new set of backups is created.
 - The template validation works in the following ways:
 - A **BackupSchedule.cluster.open-cluster-management.io** actively runs and saves new backups at the storage location. The **backup-schedule-cron-enabled** policy template completes this validation. The template checks that there is a **Backup.velero.io** with a **velero.io/schedule-name: acm-validation-policy-schedule** label at the storage location.
 - The **acm-validation-policy-schedule** backups are set to expire after you set a time for the backups cron schedule. If no cron job is running to create backups, the old **acm-validation-policy-schedule** backup is deleted because it expired and a new one is not created. As a result, if no **acm-validation-policy-schedule** backups exists, there are no active cron jobs generating backups.

The **backup-restore-auto-import** policy includes a set of templates that check for the following constraints:

- **Auto import secret validation**
 - The **auto-import-account-secret** template checks whether a **ManagedServiceAccount** secret is created in the managed cluster namespaces other than the **local-cluster**. The backup controller regularly scans for imported managed clusters. As soon as a managed cluster is discovered, the backup controller creates the **ManagedServiceAccount** resource in the managed cluster namespace. This process initiates token creation on the managed cluster. However, if the managed cluster is not accessible at the time of this operation, the **ManagedServiceAccount** is unable to create the token. For example, if the managed cluster is hibernating, it is unable to create the token. So, if a hub backup is executed during this period, the backup then lacks a token for auto-importing the managed cluster.
- **Auto import backup label validation**
 - The **auto-import-backup-label** template verifies the existence of a **ManagedServiceAccount** secret in the managed cluster namespaces other than the **local-cluster**. If the template finds the **ManagedServiceAccount** secret, then the template

enforces the **cluster.open-cluster-management.io/backup** label on the secret. This label is crucial for including the **ManagedServiceAccount** secrets in Red Hat Advanced Cluster Management backups.

1.1.7.1. Protecting data using server-side encryption

Server-side encryption is data encryption for the application or service that receives the data at the storage location. The backup mechanism itself does not encrypt data while in-transit (as it travels to and from backup storage location), or at rest (while it is stored on disks at backup storage location). Instead it relies on the native mechanisms in the object and snapshot systems.

Best practice: Encrypt the data at the destination using the available backup storage server-side encryption. The backup contains resources, such as credentials and configuration files that need to be encrypted when stored outside of the hub cluster.

You can use **serverSideEncryption** and **kmsKeyId** parameters to enable encryption for the backups stored in Amazon S3. For more details, see the *Backup Storage Location YAML*. The following sample specifies an AWS KMS key ID when setting up the **DataProtectionApplication** resource:

```
spec:
  backupLocations:
    - velero:
      config:
        kmsKeyId: 502b409c-4da1-419f-a16e-eif453b3i49f
        profile: default
        region: us-east-1
```

Refer to *Velero supported storage providers* to find out about all of the configurable parameters of other storage providers.

1.1.7.2. Additional resources

- See the [Backup Storage Location YAML](#).
- See [Velero supported storage providers](#).
- Return to [Validating your backup or restore configurations](#).

1.1.8. Backup and restore advanced configuration

You can further configure backup and restore by viewing the following sections:

1.1.8.1. Resource requests and limits customization

When Velero is initially installed, Velero pod is set to the default CPU and memory limits as defined in the following sample:

```
resources:
  limits:
    cpu: "1"
    memory: 256Mi
  requests:
    cpu: 500m
    memory: 128Mi
```

The limits from the previous sample work well with some scenarios, but might need to be updated when your cluster backs up a large number of resources. For instance, when backup is run on a hub cluster that manages 2000 clusters, then the Velero pod fails due to the out-of-memory error (OOM). The following configuration allows for the backup to complete for this scenario:

```
limits:
  cpu: "2"
  memory: 1Gi
requests:
  cpu: 500m
  memory: 256Mi
```

To update the limits and requests for the Velero pod resource, you need to update the **DataProtectionApplication** resource and insert the **resourceAllocation** template for the Velero pod. View the following sample:

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero
  namespace: open-cluster-management-backup
spec:
  ...
  configuration:
  ...
  velero:
    podConfig:
      resourceAllocations:
        limits:
          cpu: "2"
          memory: 1Gi
        requests:
          cpu: 500m
          memory: 256Mi
```

1.1.8.2. Additional resources

- Refer to the [Default Velero cloud provider plugins](#) topic in the Red Hat OpenShift Container Platform documentation to find out more about the **DataProtectionApplication** parameters.
- Refer to the [CPU and memory requirement for configurations](#) topic in the OpenShift Container Platform documentation for more details about the backup and restore CPU and memory requirements based on cluster usage.

1.2. VOLSYNC PERSISTENT VOLUME REPLICATION SERVICE

VolSync is a Kubernetes operator that enables asynchronous replication of persistent volumes within a cluster, or across clusters with storage types that are not otherwise compatible for replication. It uses the Container Storage Interface (CSI) to overcome the compatibility limitation. After deploying the VolSync operator in your environment, you can leverage it to create and maintain copies of your persistent data. VolSync can only replicate persistent volume claims on Red Hat OpenShift Container Platform clusters that are at version 4.14 or later.

Important: VolSync only supports replicating persistent volume claims with the **volumeMode** of **Filesystem**. If you do not select the **volumeMode**, it defaults to **Filesystem**.

- [Replicating persistent volumes with VolSync](#)
 - [Installing VolSync on the managed clusters](#)
 - [Configuring an Rsync-TLS replication](#)
 - [Configuring an Rsync replication](#)
 - [Configuring a restic backup](#)
 - [Configuring an Rclone replication](#)
- [Converting a replicated image to a usable persistent volume claim](#)
- [Scheduling your synchronization](#)

1.2.1. Replicating persistent volumes with VolSync

You can use three supported methods to replicate persistent volumes with VolSync, which depend on the number of synchronization locations that you have: `rsync`, `rsync-tls`, `restic`, or `Rclone`.

1.2.1.1. Prerequisites

Before installing VolSync on your clusters, you must have the following requirements:

- A configured Red Hat OpenShift Container Platform environment on a supported Red Hat Advanced Cluster Management hub cluster
- At least two managed clusters for the same Red Hat Advanced Cluster Management hub cluster
- Network connectivity between the clusters that you are configuring with VolSync. If the clusters are not on the same network, you can configure the [Submariner multicluster networking and service discovery](#) and use the **ClusterIP** value for **ServiceType** to network the clusters, or use a load balancer with the **LoadBalancer** value for **ServiceType**.
- The storage driver that you use for your source persistent volume must be CSI-compatible and able to support snapshots.

1.2.1.2. Installing VolSync on the managed clusters

To enable VolSync to replicate the persistent volume claim on one cluster to the persistent volume claim of another cluster, you must install VolSync on both the source and the target managed clusters.

VolSync does not create its own namespace, so it is in the same namespace as other OpenShift Container Platform all-namespace operators. Any changes that you make to the operator settings for VolSync also affects the other operators in the same namespace, such as if you change to manual approval for channel updates.

You can use either of two methods to install VolSync on two clusters in your environment. You can either add a label to each of the managed clusters in the hub cluster, or you can manually create and apply a **ManagedClusterAddOn**, as they are described in the following sections:

1.2.1.2.1. Installing VolSync by using labels

To install VolSync on the managed cluster by adding a label.

- Complete the following steps from the Red Hat Advanced Cluster Management console:
 1. Select one of the managed clusters from the **Clusters** page in the hub cluster console to view its details.
 2. In the **Labels** field, add the following label:

```
addons.open-cluster-management.io/volsync=true
```

The VolSync service pod is installed on the managed cluster.

3. Add the same label the other managed cluster.
4. Run the following command on each managed cluster to confirm that the VolSync operator is installed:

```
oc get csv -n openshift-operators
```

There is an operator listed for VolSync when it is installed.

- Complete the following steps from the command-line interface:
 1. Start a command-line session on the hub cluster.
 2. Enter the following command to add the label to the first cluster:

```
oc label managedcluster <managed-cluster-1> "addons.open-cluster-management.io/volsync"="true"
```

Replace **managed-cluster-1** with the name of one of your managed clusters.

3. Enter the following command to add the label to the second cluster:

```
oc label managedcluster <managed-cluster-2> "addons.open-cluster-management.io/volsync"="true"
```

Replace **managed-cluster-2** with the name of your other managed cluster.

A **ManagedClusterAddOn** resource should be created automatically on your hub cluster in the namespace of each corresponding managed cluster.

1.2.1.2.2. Installing VolSync by using a ManagedClusterAddOn

To install VolSync on the managed cluster by adding a **ManagedClusterAddOn** manually, complete the following steps:

1. On the hub cluster, create a YAML file called **volsync-mcao.yaml** that contains content that is similar to the following example:

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
```

```
name: volsync
namespace: <managed-cluster-1-namespace>
spec: {}
```

Replace **managed-cluster-1-namespace** with the namespace of one of your managed clusters. This namespace is the same as the name of the managed cluster.

Note: The name must be **volsync**.

2. Apply the file to your configuration by entering a command similar to the following example:

```
oc apply -f volsync-mcao.yaml
```

3. Repeat the procedure for the other managed cluster.
A **ManagedClusterAddOn** resource should be created automatically on your hub cluster in the namespace of each corresponding managed cluster.

1.2.1.2.3. Updating the *VolSyncManagedClusterAddOn*

Depending on what version of Red Hat Advanced Cluster Management you use, you might need to update your VolSync version. To update VolSync **ManagedClusterAddOn** resource, complete the following steps:

1. Add the following annotation to your **ManagedClusterAddOn** resource:

```
annotations:
  operator-subscription-channel: stable-0.9
```

2. Define the **operator-subscription-channel** where you want Volsync deployed from.
3. Verify that you updated your Volsync version by going to your **ManagedClusterAddOn** resource and confirming that your chosen **operator-subscription-channel** is included.

1.2.1.3. Configuring an Rsync-TLS replication

You can create a 1:1 asynchronous replication of persistent volumes by using an Rsync-TLS replication. You can use Rsync-TLS-based replication for disaster recovery or sending data to a remote site. When using Rsync-TLS, VolSync synchronizes data by using Rsync across a TLS-protected tunnel provided by stunnel. See the [stunnel documentation](#) for more information.

The following example shows how to configure by using the Rsync-TLS method. For additional information about Rsync-TLS, see [Usage](#) in the VolSync documentation.

1.2.1.3.1. Configuring Rsync-TLS replication across managed clusters

For Rsync-TLS-based replication, configure custom resources on the source and destination clusters. The custom resources use the **address** value to connect the source to the destination, and a TLS-protected tunnel provided by stunnel to ensure that the transferred data is secure.

See the following information and examples to configure an Rsync-TLS replication from a persistent volume claim on the **source** cluster in the **source-ns** namespace to a persistent volume claim on a **destination** cluster in the **destination-ns** namespace. Replace the values where necessary:

1. Configure your destination cluster.

- a. Run the following command on the destination cluster to create the namespace:

```
oc create ns <destination-ns>
```

Replace **destination-ns** with the namespace where your replication destination is located.

- b. Create a new YAML file called **replication_destination** and copy the following content:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
  namespace: <destination-ns>
spec:
  rsyncTLS:
    serviceType: LoadBalancer 1
    copyMethod: Snapshot
    capacity: 2Gi 2
    accessModes: [ReadWriteOnce]
    storageClassName: gp2-csi
    volumeSnapshotClassName: csi-aws-vsc
```

- 1 For this example, the **ServiceType** value of **LoadBalancer** is used. The load balancer service is created by the source cluster to enable your source managed cluster to transfer information to a different destination managed cluster. You can use **ClusterIP** as the service type if your source and destinations are on the same cluster, or if you have Submariner network service configured. Note the address and the name of the secret to refer to when you configure the source cluster. Make sure that the **capacity** value matches the capacity of the persistent volume claim that is being replicated.
- 2 Make sure that the **capacity** value matches the capacity of the persistent volume claim that is being replicated.

Optional: Specify the values of the **storageClassName** and **volumeSnapshotClassName** parameters if you are using a storage class and volume snapshot class name that are different than the default values for your environment.

- c. Run the following command on the destination cluster to create the **replicationdestination** resource:

```
oc create -n <destination-ns> -f replication_destination.yaml
```

Replace **destination-ns** with the name of the namespace where your destination is located.

After the **replicationdestination** resource is created, the following parameters and values are added to the resource:

Parameter	Value
.status.rsyncTLS.address	IP address of the destination cluster that is used to enable the source and destination clusters to communicate.

Parameter	Value
.status.rsyncTLS.keySecret	Name of the secret containing the TLS key that authenticates the connection with the source cluster.

- d. Run the following command to copy the value of **.status.rsyncTLS.address** to use on the source cluster. Replace **destination** with the name of your replication destination custom resource. Replace **destination-ns** with the name of the namespace where your destination is located:

```
ADDRESS=`oc get replicationdestination <destination> -n <destination-ns> --template={{.status.rsyncTLS.address}}`
echo $ADDRESS
```

The output appears similar to the following, which is for an Amazon Web Services environment:

```
a831264645yhrjrjyer6f9e4a02eb2-5592c0b3d94dd376.elb.us-east-1.amazonaws.com
```

- e. Run the following command to copy the name of the secret:

```
KEYSECRET=`oc get replicationdestination <destination> -n <destination-ns> --template={{.status.rsyncTLS.keySecret}}`
echo $KEYSECRET
```

Replace **destination** with the name of your replication destination custom resource.

Replace **destination-ns** with the name of the namespace where your destination is located.

You will have to enter it on the source cluster when you configure the source. The output should be the name of your SSH keys secret file, which might resemble the following name:

```
volsync-rsync-tls-destination-name
```

- f. Copy the key secret from the destination cluster by entering the following command against the destination cluster:

```
oc get secret -n <destination-ns> $KEYSECRET -o yaml > /tmp/secret.yaml
```

Replace **destination-ns** with the namespace where your replication destination is located.

- g. Open the secret file in the **vi** editor by entering the following command:

```
vi /tmp/secret.yaml
```

- h. In the open secret file on the destination cluster, make the following changes:
- Change the namespace to the namespace of your source cluster. For this example, it is **source-ns**.
 - Remove the owner references (**.metadata.ownerReferences**).

- i. On the source cluster, create the secret file by entering the following command on the source cluster:

```
oc create -f /tmp/secret.yaml
```

2. Identify the source persistent volume claim that you want to replicate.

Note: The source persistent volume claim must be on a CSI storage class.

3. Create the **ReplicationSource** items.

- a. Create a new YAML file called **replication_source** on the source cluster and copy the following content:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source> 1
  namespace: <source-ns> 2
spec:
  sourcePVC: <persistent_volume_claim> 3
  trigger:
    schedule: "*/3 * * * *" #/*
  rsyncTLS:
    keySecret: <mykeysecret> 4
    address: <my.host.com> 5
    copyMethod: Snapshot
    storageClassName: gp2-csi
    volumeSnapshotClassName: csi-aws-vsc
```

- 1 Replace **source** with the name for your replication source custom resource. See step 3-*vi* of this procedure for instructions on how to replace this automatically.
- 2 Replace **source-ns** with the namespace of the persistent volume claim where your source is located. See step 3-*vi* of this procedure for instructions on how to replace this automatically.
- 3 Replace **persistent_volume_claim** with the name of your source persistent volume claim.
- 4 Replace **mykeysecret** with the name of the secret that you copied from the destination cluster to the source cluster (the value of **\$KEYSECRET**).
- 5 Replace **my.host.com** with the host address that you copied from the **.status.rsyncTLS.address** field of the **ReplicationDestination** when you configured it. You can find examples of **sed** commands in the next step.

If your storage driver supports cloning, using **Clone** as the value for **copyMethod** might be a more streamlined process for the replication.

Optional: Specify the values of the **storageClassName** and **volumeSnapshotClassName** parameters if you are using a storage class and volume snapshot class name that are different than the default values for your environment.

You can now set up the synchronization method of the persistent volume.

- b. On the source cluster, modify the **replication_source.yaml** file by replacing the value of the **address** and **keySecret** in the **ReplicationSource** object with the values that you noted from the destination cluster by entering the following commands:

```
sed -i "s/<my.host.com>/$ADDRESS/g" replication_source.yaml
sed -i "s/<mykeysecret>/$KEYSECRET/g" replication_source.yaml
oc create -n <source> -f replication_source.yaml
```

Replace **my.host.com** with the host address that you copied from the **.status.rsyncTLS.address** field of the **ReplicationDestination** when you configured it.

Replace **keySecret** with the keys that you copied from the **.status.rsyncTLS.keySecret** field of the **ReplicationDestination** when you configured it.

Replace **source** with the name of the persistent volume claim where your source is located.

Note: You must create the file in the same namespace as the persistent volume claim that you want to replicate.

- c. Verify that the replication completed by running the following command on the **ReplicationSource** object:

```
oc describe ReplicationSource -n <source-ns> <source>
```

Replace **source-ns** with the namespace of the persistent volume claim where your source is located.

Replace **source** with the name of your replication source custom resource.

If the replication was successful, the output should be similar to the following example:

```
Status:
Conditions:
  Last Transition Time: 2021-10-14T20:48:00Z
  Message:             Synchronization in-progress
  Reason:              SyncInProgress
  Status:              True
  Type:                Synchronizing
  Last Transition Time: 2021-10-14T20:41:41Z
  Message:             Reconcile complete
  Reason:              ReconcileComplete
  Status:              True
  Type:                Reconciled
Last Sync Duration:   5m20.764642395s
Last Sync Time:      2021-10-14T20:47:01Z
Next Sync Time:      2021-10-14T20:48:00Z
```

If the **Last Sync Time** has no time listed, then the replication is not complete.

You have a replica of your original persistent volume claim.

1.2.1.4. Configuring an Rsync replication

Important: Use Rsync-TLS instead of Rsync for enhanced security. By using Rsync-TLS, you can avoid using elevated user permissions that are not required for replicating persistent volumes.

You can create a 1:1 asynchronous replication of persistent volumes by using an Rsync replication. You can use Rsync-based replication for disaster recovery or sending data to a remote site.

The following example shows how to configure by using the Rsync method.

1.2.1.4.1. Configuring Rsync replication across managed clusters

For Rsync-based replication, configure custom resources on the source and destination clusters. The custom resources use the **address** value to connect the source to the destination, and the **sshKeys** to ensure that the transferred data is secure.

Note: You must copy the values for **address** and **sshKeys** from the destination to the source, so configure the destination before you configure the source.

This example provides the steps to configure an Rsync replication from a persistent volume claim on the **source** cluster in the **source-ns** namespace to a persistent volume claim on a **destination** cluster in the **destination-ns** namespace. You can replace those values with other values, if necessary.

1. Configure your destination cluster.
 - a. Run the following command on the destination cluster to create the namespace:

```
oc create ns <destination-ns>
```

Replace **destination-ns** with a name for the namespace that will contain your destination persistent volume claim.

- b. Copy the following YAML content to create a new file called **replication_destination.yaml**:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
  namespace: <destination-ns>
spec:
  rsync:
    serviceType: LoadBalancer
    copyMethod: Snapshot
    capacity: 2Gi
    accessModes: [ReadWriteOnce]
    storageClassName: gp2-csi
    volumeSnapshotClassName: csi-aws-vsc
```

Note: The **capacity** value should match the capacity of the persistent volume claim that is being replicated.

Replace **destination** with the name of your replication destination CR.

Replace **destination-ns** with the name of the namespace where your destination is located.

For this example, the **ServiceType** value of **LoadBalancer** is used. The load balancer service is created by the source cluster to enable your source managed cluster to transfer information to a different destination managed cluster. You can use **ClusterIP** as the service type if your source and destinations are on the same cluster, or if you have Submariner network service configured. Note the address and the name of the secret to refer to when you configure the source cluster.

The **storageClassName** and **volumeSnapshotClassName** are optional parameters. Specify the values for your environment, particularly if you are using a storage class and volume snapshot class name that are different than the default values for your environment.

- c. Run the following command on the destination cluster to create the **replicationdestination** resource:

```
oc create -n <destination-ns> -f replication_destination.yaml
```

Replace **destination-ns** with the name of the namespace where your destination is located.

After the **replicationdestination** resource is created, following parameters and values are added to the resource:

Parameter	Value
.status.rsync.address	IP address of the destination cluster that is used to enable the source and destination clusters to communicate.
.status.rsync.sshKeys	Name of the SSH key file that enables secure data transfer from the source cluster to the destination cluster.

- d. Run the following command to copy the value of **.status.rsync.address** to use on the source cluster:

```
ADDRESS=`oc get replicationdestination <destination> -n <destination-ns> --template={{.status.rsync.address}}`
echo $ADDRESS
```

Replace **destination** with the name of your replication destination custom resource.

Replace **destination-ns** with the name of the namespace where your destination is located.

The output should appear similar to the following output, which is for an Amazon Web Services environment:

```
a831264645yhrjrjyer6f9e4a02eb2-5592c0b3d94dd376.elb.us-east-1.amazonaws.com
```

- e. Run the following command to copy the name of the secret:

```
SSHKEYS=`oc get replicationdestination <destination> -n <destination-ns> --template={{.status.rsync.sshKeys}}`
echo $SSHKEYS
```

Replace **destination** with the name of your replication destination custom resource.

Replace **destination-ns** with the name of the namespace where your destination is located.

You will have to enter it on the source cluster when you configure the source. The output should be the name of your SSH keys secret file, which might resemble the following name:

```
volsync-rsync-dst-src-destination-name
```

- f. Copy the SSH secret from the destination cluster by entering the following command against the destination cluster:

```
oc get secret -n <destination-ns> $SSHKEYS -o yaml > /tmp/secret.yaml
```

Replace **destination-ns** with the namespace of the persistent volume claim where your destination is located.

- g. Open the secret file in the **vi** editor by entering the following command:

```
vi /tmp/secret.yaml
```

- h. In the open secret file on the destination cluster, make the following changes:

- Change the namespace to the namespace of your source cluster. For this example, it is **source-ns**.
- Remove the owner references (**.metadata.ownerReferences**).

- i. On the source cluster, create the secret file by entering the following command on the source cluster:

```
oc create -f /tmp/secret.yaml
```

2. Identify the source persistent volume claim that you want to replicate.

Note: The source persistent volume claim must be on a CSI storage class.

3. Create the **ReplicationSource** items.

- a. Copy the following YAML content to create a new file called **replication_source.yaml** on the source cluster:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
spec:
  sourcePVC: <persistent_volume_claim>
  trigger:
    schedule: "*/3 * * * *" #/*
  rsync:
    sshKeys: <mysshkeys>
    address: <my.host.com>
    copyMethod: Snapshot
    storageClassName: gp2-csi
    volumeSnapshotClassName: csi-aws-vsc
```

Replace **source** with the name for your replication source custom resource. See step 3-*vi* of this procedure for instructions on how to replace this automatically.

Replace **source-ns** with the namespace of the persistent volume claim where your source is located. See step 3-*vi* of this procedure for instructions on how to replace this automatically.

Replace **persistent_volume_claim** with the name of your source persistent volume claim.

Replace **mysshkeys** with the keys that you copied from the **.status.rsync.sshKeys** field of the **ReplicationDestination** when you configured it.

Replace **my.host.com** with the host address that you copied from the **.status.rsync.address** field of the **ReplicationDestination** when you configured it.

If your storage driver supports cloning, using **Clone** as the value for **copyMethod** might be a more streamlined process for the replication.

StorageClassName and **volumeSnapshotClassName** are optional parameters. If you are using a storage class and volume snapshot class name that are different than the defaults for your environment, specify those values.

You can now set up the synchronization method of the persistent volume.

- b. On the source cluster, modify the **replication_source.yaml** file by replacing the value of the **address** and **sshKeys** in the **ReplicationSource** object with the values that you noted from the destination cluster by entering the following commands:

```
sed -i "s/<my.host.com>/$ADDRESS/g" replication_source.yaml
sed -i "s/<mysshkeys>/$SSHKEYS/g" replication_source.yaml
oc create -n <source> -f replication_source.yaml
```

Replace **my.host.com** with the host address that you copied from the **.status.rsync.address** field of the **ReplicationDestination** when you configured it.

Replace **mysshkeys** with the keys that you copied from the **.status.rsync.sshKeys** field of the **ReplicationDestination** when you configured it.

Replace **source** with the name of the persistent volume claim where your source is located.

Note: You must create the file in the same namespace as the persistent volume claim that you want to replicate.

- c. Verify that the replication completed by running the following command on the **ReplicationSource** object:

```
oc describe ReplicationSource -n <source-ns> <source>
```

Replace **source-ns** with the namespace of the persistent volume claim where your source is located.

Replace **source** with the name of your replication source custom resource.

If the replication was successful, the output should be similar to the following example:

```
Status:
Conditions:
  Last Transition Time: 2021-10-14T20:48:00Z
  Message:             Synchronization in-progress
```

```

Reason:      SynclnProgress
Status:      True
Type:        Synchronizing
Last Transition Time: 2021-10-14T20:41:41Z
Message:     Reconcile complete
Reason:      ReconcileComplete
Status:      True
Type:        Reconciled
Last Sync Duration: 5m20.764642395s
Last Sync Time:  2021-10-14T20:47:01Z
Next Sync Time:  2021-10-14T20:48:00Z

```

If the **Last Sync Time** has no time listed, then the replication is not complete.

You have a replica of your original persistent volume claim.

1.2.1.5. Configuring a restic backup

A restic-based backup copies a restic-based backup copy of the persistent volume to a location that is specified in your **restic-config.yaml** secret file. A restic backup does not synchronize data between the clusters, but provides data backup.

Complete the following steps to configure a restic-based backup:

1. Specify a repository where your backup images are stored by creating a secret that resembles the following YAML content:

```

apiVersion: v1
kind: Secret
metadata:
  name: restic-config
type: Opaque
stringData:
  RESTIC_REPOSITORY: <my-restic-repository>
  RESTIC_PASSWORD: <my-restic-password>
  AWS_ACCESS_KEY_ID: access
  AWS_SECRET_ACCESS_KEY: password

```

Replace **my-restic-repository** with the location of the S3 bucket repository where you want to store your backup files.

Replace **my-restic-password** with the encryption key that is required to access the repository.

Replace **access** and **password** with the credentials for your provider, if required.

If you need to prepare a new repository, see [Preparing a new repository](#) for the procedure. If you use that procedure, skip the step that requires you to run the **restic init** command to initialize the repository. VolSync automatically initializes the repository during the first backup.

Important: When backing up multiple persistent volume claims to the same S3 bucket, the path to the bucket must be unique for each persistent volume claim. Each persistent volume claim is backed up with a separate **ReplicationSource**, and each requires a separate restic-config secret.

By sharing the same S3 bucket, each **ReplicationSource** has write access to the entire S3 bucket.

2. Configure your backup policy by creating a **ReplicationSource** object that resembles the following YAML content:

```

apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: mydata-backup
spec:
  sourcePVC: <source>
  trigger:
    schedule: "*/30 * * * *" #\*
  restic:
    pruneIntervalDays: 14
    repository: <restic-config>
  retain:
    hourly: 6
    daily: 5
    weekly: 4
    monthly: 2
    yearly: 1
  copyMethod: Clone
# The StorageClass to use when creating the PiT copy (same as source PVC if omitted)
#storageClassName: my-sc-name
# The VSC to use if the copy method is Snapshot (default if omitted)
#volumeSnapshotClassName: my-vsc-name

```

Replace **source** with the persistent volume claim that you are backing up.

Replace the value for **schedule** with how often to run the backup. This example has the schedule for every 30 minutes. See [Scheduling your synchronization](#) for more information about setting up your schedule.

Replace the value of **PruneIntervalDays** to the number of days that elapse between instances of repacking the data to save space. The prune operation can generate significant I/O traffic while it is running.

Replace **restic-config** with the name of the secret that you created in step 1.

Set the values for **retain** to your retention policy for the backed up images.

Best practice: Use **Clone** for the value of **CopyMethod** to ensure that a point-in-time image is saved.

Note: Restic movers run without root permissions by default. If you want to run restic movers as root, run the following command to add the elevated permissions annotation to your namespace.

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

Replace **<namespace>** with the name of your namespace.

1.2.1.5.1. Restoring a restic backup

You can restore the copied data from a restic backup into a new persistent volume claim. **Best practice:** Restore only one backup into a new persistent volume claim. To restore the restic backup, complete the following steps:

1. Create a new persistent volume claim to contain the new data similar to the following example:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Replace **pvc-name** with the name of the new persistent volume claim.

2. Create a **ReplicationDestination** custom resource that resembles the following example to specify where to restore the data:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
spec:
  trigger:
    manual: restore-once
  restic:
    repository: <restic-repo>
    destinationPVC: <pvc-name>
    copyMethod: Direct
```

Replace **destination** with the name of your replication destination CR.

Replace **restic-repo** with the path to your repository where the source is stored.

Replace **pvc-name** with the name of the new persistent volume claim where you want to restore the data. Use an existing persistent volume claim for this, rather than provisioning a new one.

The restore process only needs to be completed once, and this example restores the most recent backup. For more information about restore options, see [Restore options](#) in the VolSync documentation.

1.2.1.6. Configuring an Rclone replication

An Rclone backup copies a single persistent volume to multiple locations by using Rclone through an intermediate object storage location, like AWS S3. It can be helpful when distributing data to multiple locations.

Complete the following steps to configure an Rclone replication:

1. Create a **ReplicationSource** custom resource that resembles the following example:

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
```



```

spec:
  sourcePVC: <source-pvc>
  trigger:
    schedule: "*" / 6 * * * * * #\*
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    storageClassName: <my-sc-name>
    volumeSnapshotClassName: <my-vsc>

```

Replace **source-pvc** with the name for your replication source custom resource.

Replace **source-ns** with the namespace of the persistent volume claim where your source is located.

Replace **source** with the persistent volume claim that you are replicating.

Replace the value of **schedule** with how often to run the replication. This example has the schedule for every 6 minutes. This value must be within quotation marks. See [Scheduling your synchronization](#) for more information.

Replace **intermediate-s3-bucket** with the path to the configuration section of the Rclone configuration file.

Replace **destination-bucket** with the path to the object bucket where you want your replicated files copied.

Replace **rclone-secret** with the name of the secret that contains your Rclone configuration information.

Set the value for **copyMethod** as **Clone**, **Direct**, or **Snapshot**. This value specifies whether the point-in-time copy is generated, and if so, what method is used for generating it.

Replace **my-sc-name** with the name of the storage class that you want to use for your point-in-time copy. If not specified, the storage class of the source volume is used.

Replace **my-vsc** with the name of the **VolumeSnapshotClass** to use, if you specified **Snapshot** as your **copyMethod**. This is not required for other types of **copyMethod**.

2. Create a **ReplicationDestination** custom resource that resembles the following example:

```

apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: database-destination
  namespace: dest
spec:
  trigger:
    schedule: "3,9,15,21,27,33,39,45,51,57 * * * * * #/*"
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    accessModes: [ReadWriteOnce]

```

```
capacity: 10Gi
storageClassName: <my-sc>
volumeSnapshotClassName: <my-vsc>
```

Replace the value for **schedule** with how often to move the replication to the destination. The schedules for the source and destination must be offset to allow the data to finish replicating before it is pulled from the destination. This example has the schedule for every 6 minutes, offset by 3 minutes. This value must be within quotation marks. See [Scheduling your synchronization](#) for more information about scheduling.

Replace **intermediate-s3-bucket** with the path to the configuration section of the Rclone configuration file.

Replace **destination-bucket** with the path to the object bucket where you want your replicated files copied.

Replace **rclone-secret** with the name of the secret that contains your Rclone configuration information.

Set the value for **copyMethod** as **Clone**, **Direct**, or **Snapshot**. This value specifies whether the point-in-time copy is generated, and if so, which method is used for generating it.

The value for **accessModes** specifies the access modes for the persistent volume claim. Valid values are **ReadWriteOnce** or **ReadWriteMany**.

The **capacity** specifies the size of the destination volume, which must be large enough to contain the incoming data.

Replace **my-sc** with the name of the storage class that you want to use as the destination for your point-in-time copy. If not specified, the system storage class is used.

Replace **my-vsc** with the name of the **VolumeSnapshotClass** to use, if you specified **Snapshot** as your **copyMethod**. This is not required for other types of **copyMethod**. If not included, the system default **VolumeSnapshotClass** is used.

Note: Rclone movers run without root permissions by default. If you want to run Rclone movers as root, run the following command to add the elevated permissions annotation to your namespace.

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

Replace **<namespace>** with the name of your namespace.

1.2.1.7. Additional resources

See the following topics for more information:

- See [Creating a secret for Rsync-TLS replication](#) to learn how to create your own secret for an Rsync-TLS replication.
- For additional information about Rsync, see [Usage](#) in the VolSync documentation.
- For additional information about restic options, see [Backup options](#) in the VolSync documentation.
- Go back to [Installing VolSync on the managed clusters](#)

1.2.2. Converting a replicated image to a usable persistent volume claim

You might need to convert the replicated image to a persistent volume claim to recover data.

When you replicate or restore a persistent volume claim from a **ReplicationDestination** location by using a **VolumeSnapshot**, a **VolumeSnapshot** is created. The **VolumeSnapshot** contains the **latestImage** from the last successful synchronization. The copy of the image must be converted to a persistent volume claim before it can be used. The VolSync **ReplicationDestination** volume populator can be used to convert a copy of the image to a usable persistent volume claim.

1. Create a persistent volume claim with a **dataSourceRef** that points to the **ReplicationDestination** where you want to restore a persistent volume claim. This persistent volume claim is populated with the contents of the **VolumeSnapshot** that is specified in the **status.latestImage** setting of the **ReplicationDestination** custom resource definition. The following YAML content shows a sample persistent volume claim that might be used:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <pvc-name>
  namespace: <destination-ns>
spec:
  accessModes:
    - ReadWriteOnce
  dataSourceRef:
    kind: ReplicationDestination
    apiGroup: volsync.backube
    name: <replicationdestination_to_replace>
  resources:
    requests:
      storage: 2Gi
```

Replace **pvc-name** with a name for your new persistent volume claim.

Replace **destination-ns** with the namespace where the persistent volume claim and **ReplicationDestination** are located.

Replace **replicationdestination_to_replace** with the **ReplicationDestination** name.

Best practice: You can update **resources.requests.storage** with a different value when the value is at least the same size as the initial source persistent volume claim.

2. Validate that your persistent volume claim is running in the environment by entering the following command:

```
$ kubectl get pvc -n <destination-ns>
```

Note:

If no **latestImage** exists, the persistent volume claim remains in a pending state until the **ReplicationDestination** completes and a snapshot is available. You can create a **ReplicationDestination** and a persistent volume controller that use the **ReplicationDestination** at the same time. The persistent volume claim only starts the volume population process after the **ReplicationDestination** completed a replication and a snapshot is available. You can find the snapshot in **.status.latestImage**.

Additionally, if the storage class that is used has a **volumeBindingMode** value of **WaitForFirstConsumer**, the volume populator waits until there is a consumer of the persistent volume claim before it is populated. When a consumer requires access, such as a pod that wants to mount the persistent volume claim, then the volume is populated. The VolSync volume populator controller uses the **latestImage** from the **ReplicationDestination**. The **latestImage** is updated each time a replication completes after the persistent volume control was created.

1.2.3. Scheduling your synchronization

Select from three options when determining how you start your replications: always running, on a schedule, or manually. Scheduling your replications is an option that is often selected.

The **Schedule** option runs replications at scheduled times. A schedule is defined by a **cronspec**, so the schedule can be configured as intervals of time or as specific times. The order of the schedule values are:

"minute (0-59) hour (0-23) day-of-month (1-31) month (1-12) day-of-week (0-6)"

The replication starts when the scheduled time occurs. Your setting for this replication option might resemble the following content:

```
spec:
  trigger:
    schedule: "*/6 * * * *"
```

After enabling one of these methods, your synchronization schedule runs according to the method that you configured.

See the [VolSync](#) documentation for additional information and options.

1.2.4. VolSync advanced configuration

You can further configure VolSync when replicating persistent volumes, such as creating your own secret.

1.2.4.1. Creating a secret for Rsync-TLS replication

The source and destination must have access to the shared key for the TLS connection. You can find the key location in the **keySecret** field. If you do not provide a secret name in **.spec.rsyncTLS.keySecret**, the secret name is automatically generated and added to **.status.rsyncTLS.keySecret**.

To create your own secret, complete the following steps:

1. Use the following format for the secret: **<id>:<at_least_32_hex_digits>**
See the following example: **1:23b7395fafc3e842bd8ac0fe142e6ad1**
2. See the following **secret.yaml** example which corresponds to the previous example:

```
apiVersion: v1
data:
  # echo -n 1:23b7395fafc3e842bd8ac0fe142e6ad1 | base64
  psk.txt: MTToyM2l3Mzk1ZmFmYzNIODQyYmQ4YWMwZmUxNDJINmFkMQ==
kind: Secret
```

metadata:
name: tls-key-secret
type: Opaque