



# OpenShift Container Platform 4.2

## Migration

Migrating from OpenShift Container Platform 3 to 4



# OpenShift Container Platform 4.2 Migration

---

Migrating from OpenShift Container Platform 3 to 4

## Legal Notice

Copyright © 2020 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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

This document provides instructions for migrating your OpenShift Container Platform cluster from version 3 to version 4.

## Table of Contents

<b>CHAPTER 1. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 3</b> .....	<b>6</b>
1.1. ABOUT MIGRATING OPENSIFT CONTAINER PLATFORM 3 TO 4	6
1.2. PLANNING YOUR MIGRATION	6
1.2.1. Comparing OpenShift Container Platform 3 and OpenShift Container Platform 4	6
1.2.1.1. Architecture differences	7
Immutable infrastructure	7
Operators	7
1.2.1.2. Installation and update differences	7
Installation process	7
Infrastructure options	8
Upgrading your cluster	8
1.2.2. Migration considerations	8
1.2.2.1. Storage considerations	8
Local volume persistent storage	8
FlexVolume persistent storage	8
Container Storage Interface (CSI) persistent storage	8
Red Hat OpenShift Container Storage	8
Unsupported persistent storage options	9
1.2.2.2. Networking considerations	9
Network isolation mode	9
Encrypting traffic between hosts	9
1.2.2.3. Logging considerations	9
Deploying cluster logging	9
Aggregated logging data	10
1.2.2.4. Security considerations	10
Unauthenticated access to discovery endpoints	10
Identity providers	10
1.2.2.5. Monitoring considerations	10
Alert for monitoring infrastructure availability	10
1.3. MIGRATION TOOLS AND PREREQUISITES	10
1.3.1. Migration prerequisites	11
1.3.2. About the Cluster Application Migration tool	12
1.3.3. About data copy methods	13
1.3.3.1. File system copy method	14
1.3.3.2. Snapshot copy method	14
1.3.4. About migration hooks	14
1.3.5. About the Control Plane Migration Assistant	15
1.4. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL	16
1.4.1. Installing the Cluster Application Migration Operator	16
1.4.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster	16
1.4.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 3 source cluster	17
1.4.2. Installing the Cluster Application Migration Operator in a restricted environment	18
1.4.2.1. Configuring OperatorHub for restricted networks	19
1.4.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment	24
1.4.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 3 source cluster in a restricted environment	25
1.4.3. Launching the CAM web console	27
1.5. CONFIGURING A REPLICATION REPOSITORY	27

1.5.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository	28
1.5.1.1. Installing the OpenShift Container Storage Operator	28
1.5.1.2. Creating the Multi-Cloud Object Gateway storage bucket	28
1.5.2. Configuring an AWS S3 storage bucket as a replication repository	31
1.5.3. Configuring a Google Cloud Provider storage bucket as a replication repository	33
1.5.4. Configuring a Microsoft Azure Blob storage container as a replication repository	35
1.6. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE	36
1.6.1. Creating a CA certificate bundle file	36
1.6.2. Adding a cluster to the CAM web console	37
1.6.3. Adding a replication repository to the CAM web console	38
1.6.4. Changing migration plan limits for large migrations	39
1.6.5. Creating a migration plan in the CAM web console	40
1.6.6. Running a migration plan in the CAM web console	42
1.7. MIGRATING CONTROL PLANE SETTINGS WITH THE CONTROL PLANE MIGRATION ASSISTANT (CPMA)	42
1.7.1. Installing the Control Plane Migration Assistant	43
1.7.2. Using the Control Plane Migration Assistant	43
1.8. TROUBLESHOOTING	45
1.8.1. Viewing migration Custom Resources	45
1.8.2. Downloading migration logs	50
1.8.3. Updating deprecated API GroupVersionKinds	50
1.8.4. Error messages	53
1.8.4.1. Restic timeout error message in the Velero Pod log	53
1.8.4.2. ResticVerifyErrors in the MigMigration Custom Resource	53
1.8.5. Manually rolling back a migration	55
1.8.6. Gathering data for a customer support case	56
1.8.7. Known issues	57
<b>CHAPTER 2. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 4.1</b>	<b>58</b>
2.1. MIGRATION TOOLS AND PREREQUISITES	58
2.1.1. Migration prerequisites	58
2.1.2. About the Cluster Application Migration tool	59
2.1.3. About data copy methods	60
2.1.3.1. File system copy method	61
2.1.3.2. Snapshot copy method	61
2.1.4. About migration hooks	61
2.2. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL	62
2.2.1. Installing the Cluster Application Migration Operator	62
2.2.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster	62
2.2.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.1 source cluster	63
2.2.2. Installing the Cluster Application Migration Operator in a restricted environment	64
2.2.2.1. Configuring OperatorHub for restricted networks	64
2.2.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment	69
2.2.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.1 source cluster in a restricted environment	70
2.2.3. Launching the CAM web console	70
2.3. CONFIGURING A REPLICATION REPOSITORY	71
2.3.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository	71
2.3.1.1. Installing the OpenShift Container Storage Operator	72
2.3.1.2. Creating the Multi-Cloud Object Gateway storage bucket	72
2.3.2. Configuring an AWS S3 storage bucket as a replication repository	74

2.3.3. Configuring a Google Cloud Provider storage bucket as a replication repository	77
2.3.4. Configuring a Microsoft Azure Blob storage container as a replication repository	78
2.4. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE	80
2.4.1. Creating a CA certificate bundle file	80
2.4.2. Adding a cluster to the CAM web console	80
2.4.3. Adding a replication repository to the CAM web console	81
2.4.4. Changing migration plan limits for large migrations	82
2.4.5. Creating a migration plan in the CAM web console	83
2.4.6. Running a migration plan in the CAM web console	85
2.5. TROUBLESHOOTING	86
2.5.1. Viewing migration Custom Resources	86
2.5.2. Downloading migration logs	91
2.5.3. Error messages	91
2.5.3.1. Restic timeout error message in the Velero Pod log	91
2.5.3.2. ResticVerifyErrors in the MigMigration Custom Resource	92
2.5.4. Manually rolling back a migration	93
2.5.5. Gathering data for a customer support case	95
2.5.6. Known issues	95
<b>CHAPTER 3. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 4.2</b> .....	<b>97</b>
3.1. MIGRATION TOOLS AND PREREQUISITES	97
3.1.1. Migration prerequisites	97
3.1.2. About the Cluster Application Migration tool	98
3.1.3. About data copy methods	99
3.1.3.1. File system copy method	100
3.1.3.2. Snapshot copy method	100
3.1.4. About migration hooks	100
3.2. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL	101
3.2.1. Installing the Cluster Application Migration Operator	101
3.2.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster	101
3.2.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 source cluster	102
3.2.2. Installing the Cluster Application Migration Operator in a restricted environment	103
3.2.2.1. Configuring OperatorHub for restricted networks	103
3.2.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment	108
3.2.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 source cluster in a restricted environment	109
3.2.3. Launching the CAM web console	109
3.3. CONFIGURING A REPLICATION REPOSITORY	110
3.3.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository	111
3.3.1.1. Installing the OpenShift Container Storage Operator	111
3.3.1.2. Creating the Multi-Cloud Object Gateway storage bucket	111
3.3.2. Configuring an AWS S3 storage bucket as a replication repository	113
3.3.3. Configuring a Google Cloud Provider storage bucket as a replication repository	116
3.3.4. Configuring a Microsoft Azure Blob storage container as a replication repository	117
3.4. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE	119
3.4.1. Creating a CA certificate bundle file	119
3.4.2. Adding a cluster to the CAM web console	119
3.4.3. Adding a replication repository to the CAM web console	120
3.4.4. Changing migration plan limits for large migrations	121
3.4.5. Creating a migration plan in the CAM web console	122

3.4.6. Running a migration plan in the CAM web console	124
3.5. TROUBLESHOOTING	125
3.5.1. Viewing migration Custom Resources	125
3.5.2. Downloading migration logs	130
3.5.3. Error messages	130
3.5.3.1. Restic timeout error message in the Velero Pod log	130
3.5.3.2. ResticVerifyErrors in the MigMigration Custom Resource	131
3.5.4. Manually rolling back a migration	132
3.5.5. Gathering data for a customer support case	134
3.5.6. Known issues	134





# CHAPTER 1. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 3

## 1.1. ABOUT MIGRATING OPENSIFT CONTAINER PLATFORM 3 TO 4

OpenShift Container Platform 4 includes new technologies and functionality that results in a cluster that is self-managing, flexible, and automated. The way that OpenShift Container Platform 4 clusters are deployed and managed drastically differs from OpenShift Container Platform 3.

To successfully transition from OpenShift Container Platform 3 to OpenShift Container Platform 4, it is important that you review the following information:

### Planning your transition

Learn about the differences between OpenShift Container Platform versions 3 and 4. Prior to transitioning, be sure that you have reviewed and prepared for storage, networking, logging, security, and monitoring considerations.

### Performing your migration

Learn about and use the tools to perform your migration:

- Cluster Application Migration (CAM) tool to migrate your application workloads
- Control Plane Migration Assistant (CPMA) to migrate your control plane

## 1.2. PLANNING YOUR MIGRATION

Before performing your migration to OpenShift Container Platform 4.2, it is important to take the time to properly plan for the transition. OpenShift Container Platform 4 introduces architectural changes and enhancements, so the procedures that you used to manage your OpenShift Container Platform 3 cluster might not apply for OpenShift Container Platform 4.



### NOTE

This planning document assumes that you are transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

This document provides high-level information on the most important [differences between OpenShift Container Platform 3 and OpenShift Container Platform 4](#) and the most noteworthy [migration considerations](#). For detailed information on configuring your OpenShift Container Platform 4 cluster, review the appropriate sections of the OpenShift Container Platform documentation. For detailed information on new features and other notable technical changes, review the [OpenShift Container Platform 4.2 release notes](#).

It is not possible to upgrade your existing OpenShift Container Platform 3 cluster to OpenShift Container Platform 4. You must start with a new OpenShift Container Platform 4 installation. Tools are available to assist in migrating your control plane settings and application workloads.

### 1.2.1. Comparing OpenShift Container Platform 3 and OpenShift Container Platform 4

With OpenShift Container Platform 3, administrators individually deployed Red Hat Enterprise Linux (RHEL) hosts, and then installed OpenShift Container Platform on top of these hosts to form a cluster. Administrators were responsible for properly configuring these hosts and performing updates.

OpenShift Container Platform 4 represents a significant change in the way that OpenShift Container Platform clusters are deployed and managed. OpenShift Container Platform 4 includes new technologies and functionality, such as Operators, MachineSets, and Red Hat Enterprise Linux CoreOS (RHCOS), which are core to the operation of the cluster. This technology shift enables clusters to self-manage some functions previously performed by administrators. This also ensures platform stability and consistency, and simplifies installation and scaling.

For more information, see [OpenShift Container Platform architecture](#).

### 1.2.1.1. Architecture differences

#### Immutable infrastructure

OpenShift Container Platform 4 uses Red Hat Enterprise Linux CoreOS (RHCOS), which is designed to run containerized applications, and provides efficient installation, Operator-based management, and simplified upgrades. RHCOS is an immutable container host, rather than a customizable operating system like RHEL. RHCOS enables OpenShift Container Platform 4 to manage and automate the deployment of the underlying container host. RHCOS is a part of OpenShift Container Platform, which means that everything runs inside a container and is deployed using OpenShift Container Platform.

In OpenShift Container Platform 4, control plane nodes must run RHCOS, ensuring that full-stack automation is maintained for the control plane. This makes rolling out updates and upgrades a much easier process than in OpenShift Container Platform 3.

For more information, see [Red Hat Enterprise Linux CoreOS](#).

#### Operators

Operators are a method of packaging, deploying, and managing a Kubernetes application. Operators ease the operational complexity of running another piece of software. They watch over your environment and use the current state to make decisions in real time. Advanced Operators are designed to upgrade and react to failures automatically.

For more information, see [Understanding Operators](#).

### 1.2.1.2. Installation and update differences

#### Installation process

To install OpenShift Container Platform 3.11, you prepared your Red Hat Enterprise Linux (RHEL) hosts, set all of the configuration values your cluster needed, and then ran an Ansible playbook to install and set up your cluster.

In OpenShift Container Platform 4.2, you use the OpenShift installation program to create a minimum set of resources required for a cluster. Once the cluster is running, you use Operators to further configure your cluster and to install new services. After first boot, Red Hat Enterprise Linux CoreOS (RHCOS) systems are managed by the Machine Config Operator (MCO) that runs in the OpenShift Container Platform cluster.

For more information, see [Installation process](#).

If you want to add RHEL worker machines to your OpenShift Container Platform 4.2 cluster, you use an Ansible playbook to join the RHEL worker machines after the cluster is running. For more information, see [Adding RHEL compute machines to an OpenShift Container Platform cluster](#).

### Infrastructure options

In OpenShift Container Platform 3.11, you installed your cluster on infrastructure that you prepared and maintained. In addition to providing your own infrastructure, OpenShift Container Platform 4 offers an option to deploy a cluster on infrastructure that the OpenShift Container Platform installation program provisions and the cluster maintains.

For more information, see [OpenShift Container Platform installation overview](#).

### Upgrading your cluster

In OpenShift Container Platform 3.11, you upgraded your cluster by running Ansible playbooks. In OpenShift Container Platform 4.2, The cluster manages its own updates, including updates to Red Hat Enterprise Linux CoreOS (RHCOS) on cluster nodes. You can easily upgrade your cluster by using the web console or by using the **oc adm upgrade** command from the OpenShift CLI and the Operators will automatically upgrade themselves. If your OpenShift Container Platform 4.2 cluster has Red Hat Enterprise Linux worker machines, then you will still need to run an Ansible playbook to upgrade those worker machines.

For more information, see [Updating clusters](#).

## 1.2.2. Migration considerations

Review the changes and other considerations that might affect your transition from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.

### 1.2.2.1. Storage considerations

Review the following storage changes to consider when transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

#### Local volume persistent storage

Local storage is only supported by using the Local Storage Operator in OpenShift Container Platform 4.2. It is not supported to use the local provisioner method from OpenShift Container Platform 3.11.

For more information, see [Persistent storage using local volumes](#).

#### FlexVolume persistent storage

The FlexVolume plug-in location changed from OpenShift Container Platform 3.11. The new location in OpenShift Container Platform 4.2 is **/etc/kubernetes/kubelet-plugins/volume/exec**. Attachable FlexVolume plug-ins are no longer supported.

For more information, see [Persistent storage using FlexVolume](#).

#### Container Storage Interface (CSI) persistent storage

Persistent storage using the Container Storage Interface (CSI) was [Technology Preview](#) in OpenShift Container Platform 3.11. CSI version 1.1.0 is fully supported in OpenShift Container Platform 4.2, but does not ship with any CSI drivers. You must install your own driver.

For more information, see [Persistent storage using the Container Storage Interface \(CSI\)](#).

#### Red Hat OpenShift Container Storage

Red Hat OpenShift Container Storage 3, which is available for use with OpenShift Container Platform 3.11, uses Red Hat Gluster Storage as the backing storage.

Red Hat OpenShift Container Storage 4, which is available for use with OpenShift Container Platform 4, uses Red Hat Ceph Storage as the backing storage.

For more information, see [Persistent storage using Red Hat OpenShift Container Storage](#) and the [interoperability matrix](#) article.

### Unsupported persistent storage options

Support for the following persistent storage options from OpenShift Container Platform 3.11 has changed in OpenShift Container Platform 4.2:

- GlusterFS is no longer supported.
- CephFS as a standalone product is no longer supported.
- Ceph RBD as a standalone product is no longer supported.
- iSCSI is now Technology Preview.

If you used one of these in OpenShift Container Platform 3.11, you must choose a different persistent storage option for full support in OpenShift Container Platform 4.2.

For more information, see [Understanding persistent storage](#).

### 1.2.2.2. Networking considerations

Review the following networking changes to consider when transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

#### Network isolation mode

The default network isolation mode for OpenShift Container Platform 3.11 was **ovs-subnet**, though users frequently switched to use **ovn-multitenant**. The default network isolation mode for OpenShift Container Platform 4.2 is now NetworkPolicy.

If your OpenShift Container Platform 3.11 cluster used the **ovs-subnet** or **ovs-multitenant** mode, it is recommended to switch to the NetworkPolicy mode for your OpenShift Container Platform 4.2 cluster. NetworkPolicy is supported upstream, is more flexible, and also provides the functionality that **ovs-multitenant** does. If you want to maintain the **ovs-multitenant** behavior while using NetworkPolicy in OpenShift Container Platform 4.2, follow the steps to [configure multitenant isolation using NetworkPolicy](#).

For more information, see [About network policy](#).

#### Encrypting traffic between hosts

In OpenShift Container Platform 3.11, you could use IPsec to encrypt traffic between hosts. OpenShift Container Platform 4.2 does not support IPsec. It is recommended to use Red Hat OpenShift Service Mesh to enable mutual TLS between services.

For more information, see [Understanding Red Hat OpenShift Service Mesh](#).

### 1.2.2.3. Logging considerations

Review the following logging changes to consider when transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

#### Deploying cluster logging

OpenShift Container Platform 4 provides a simple deployment mechanism for cluster logging, by using a Cluster Logging custom resource. Once deployed, the cluster logging experience is the same as it was in OpenShift Container Platform 3.11.

For more information, see [About deploying and configuring cluster logging](#).

### Aggregated logging data

You cannot transition your aggregate logging data from OpenShift Container Platform 3.11 into your new OpenShift Container Platform 4 cluster.

For more information, see [About cluster logging](#).

#### 1.2.2.4. Security considerations

Review the following security changes to consider when transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

##### Unauthenticated access to discovery endpoints

In OpenShift Container Platform 3.11, an unauthenticated user could access the discovery endpoints (for example, `/api/*` and `/apis/*`). For security reasons, unauthenticated access to the discovery endpoints is no longer allowed in OpenShift Container Platform 4.2. If you do need to allow unauthenticated access, you can configure the RBAC settings as necessary; however, be sure to consider the security implications as this can expose internal cluster components to the external network.

##### Identity providers

Configuration for identity providers has changed for OpenShift Container Platform 4, including the following notable changes:

- The request header identity provider in OpenShift Container Platform 4.2 requires mutual TLS, where in OpenShift Container Platform 3.11 it did not.
- The configuration of the OpenID Connect identity provider was simplified in OpenShift Container Platform 4.2. It now obtains data, which previously had to be specified in OpenShift Container Platform 3.11, from the provider's `/.well-known/openid-configuration` endpoint.

For more information, see [Understanding identity provider configuration](#).

#### 1.2.2.5. Monitoring considerations

Review the following monitoring changes to consider when transitioning from OpenShift Container Platform 3.11 to OpenShift Container Platform 4.2.

##### Alert for monitoring infrastructure availability

The default alert that triggers to ensure the availability of the monitoring structure was called **DeadMansSwitch** in OpenShift Container Platform 3.11. This was renamed to **Watchdog** in OpenShift Container Platform 4. If you had PagerDuty integration set up with this alert in OpenShift Container Platform 3.11, you must set up the PagerDuty integration for the **Watchdog** alert in OpenShift Container Platform 4.

For more information, see [Applying custom Alertmanager configuration](#).

## 1.3. MIGRATION TOOLS AND PREREQUISITES

You can migrate application workloads from OpenShift Container Platform 3.7, 3.9, 3.10, and 3.11 to OpenShift Container Platform 4.2 with the Cluster Application Migration (CAM) tool. The CAM tool enables you to control the migration and to minimize application downtime.

The CAM tool's web console and API, based on Kubernetes Custom Resources, enable you to migrate stateful application workloads at the granularity of a namespace.

The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.

The Control Plane Migration Assistant (CPMA) is a CLI-based tool that assists you in migrating the control plane. The CPMA processes the OpenShift Container Platform 3 configuration files and generates Custom Resource (CR) manifest files, which are consumed by OpenShift Container Platform 4.2 Operators.



### IMPORTANT

Before you begin your migration, be sure to review the information on [planning your migration](#).

#### 1.3.1. Migration prerequisites

- You must have **podman** installed.
- The source cluster must be OpenShift Container Platform 3.7, 3.9, 3.10, or 3.11.
- You must upgrade the source cluster to the latest z-stream release.
- You must have **cluster-admin** privileges on all clusters.
- The source and target clusters must have unrestricted network access to the replication repository.
- The cluster on which the Migration controller is installed must have unrestricted access to the other clusters.
- If your application uses images from the **openshift** namespace, the required versions of the images must be present on the target cluster.  
If the required images are not present, you must update the **imagestreamtags** references to use an available version that is compatible with your application. If the **imagestreamtags** cannot be updated, you can manually upload equivalent images to the application namespaces and update the applications to reference them.

The following **imagestreamtags** have been *removed* from OpenShift Container Platform 4.2:

- **dotnet:1.0, dotnet:1.1, dotnet:2.0**
- **dotnet-runtime:2.0**
- **mariadb:10.1**
- **mongodb:2.4, mongodb:2.6**
- **mysql:5.5, mysql:5.6**
- **nginx:1.8**
- **nodejs:0.10, nodejs:4, nodejs:6**
- **perl:5.16, perl:5.20**

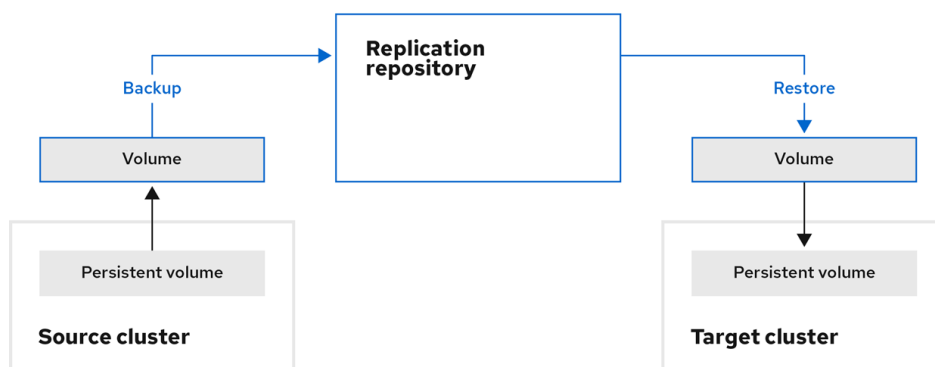
- **php:5.5, php:5.6**
- **postgresql:9.2, postgresql:9.4, postgresql:9.5**
- **python:3.3, python:3.4**
- **ruby:2.0, ruby:2.2**

### 1.3.2. About the Cluster Application Migration tool

The Cluster Application Migration (CAM) tool enables you to migrate Kubernetes resources, persistent volume data, and internal container images from an OpenShift Container Platform source cluster to an OpenShift Container Platform 4.2 target cluster, using the CAM web console or the Kubernetes API.

Migrating an application with the CAM web console involves the following steps:

1. Install the Cluster Application Migration Operator on all clusters.  
You can install the Cluster Application Migration Operator in a restricted environment with limited or no internet access. The source and target clusters must have network access to each other and to a mirror registry.
2. Configure the replication repository, an intermediate object storage that the CAM tool uses to migrate data.  
The source and target clusters must have network access to the replication repository during migration. In a restricted environment, you can use an internally hosted S3 storage repository. If you use a proxy server, you must ensure that replication repository is whitelisted.
3. Add the source cluster to the CAM web console.
4. Add the replication repository to the CAM web console.
5. Create a migration plan, with one of the following data migration options:
  - **Copy:** The CAM tool copies the data from the source cluster to the replication repository, and from the replication repository to the target cluster.



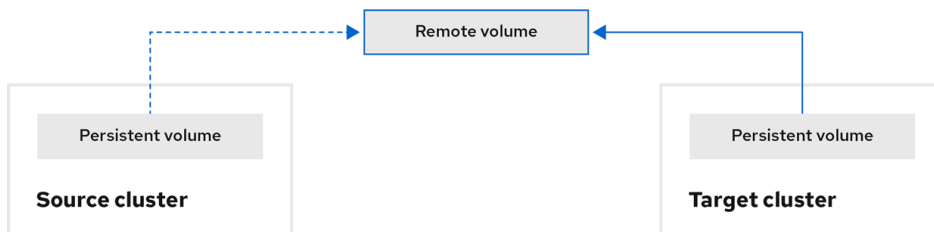
OpenShift\_45\_1019

- **Move:** The CAM tool unmounts a remote volume (for example, NFS) from the source cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.



**NOTE**

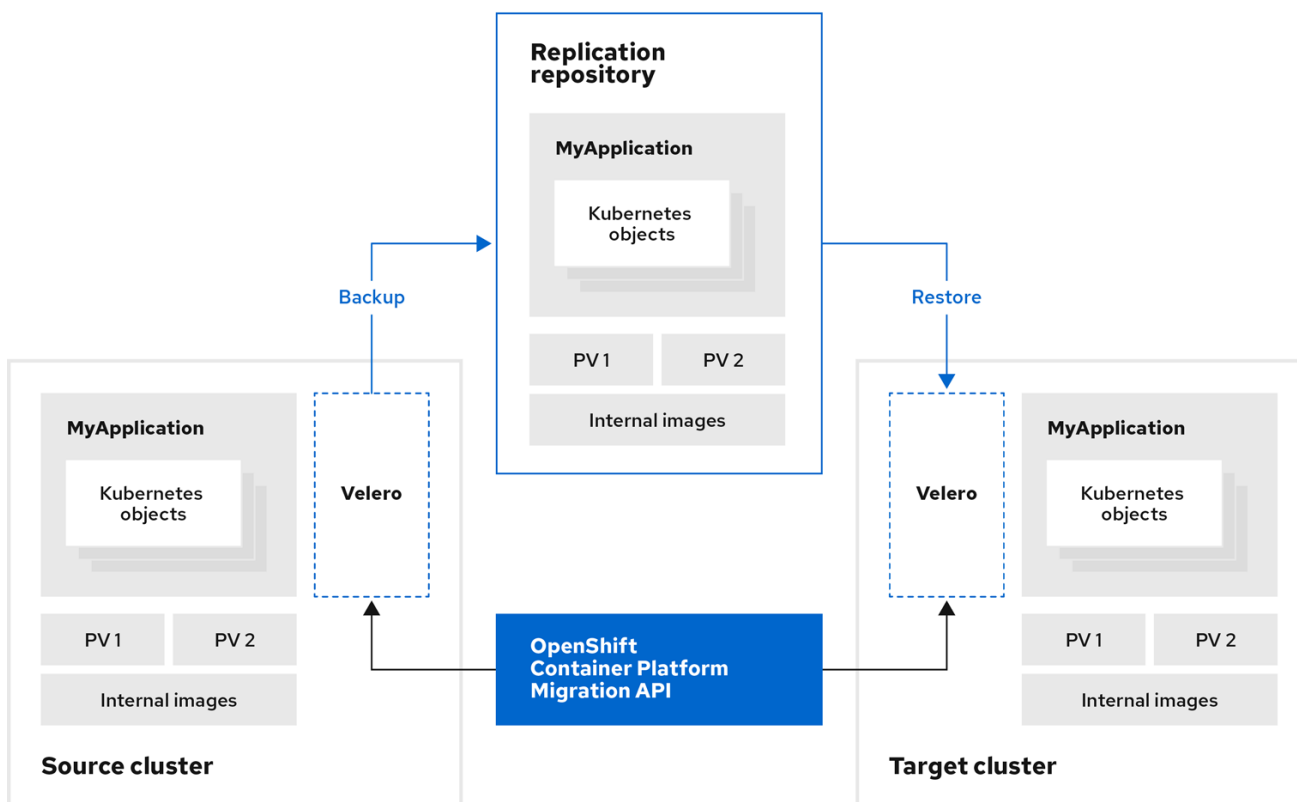
Although the replication repository does not appear in this diagram, it is required for the actual migration.



OpenShift\_45\_1019

6. Run the migration plan, with one of the following options:

- **Stage** (optional) copies data to the target cluster without stopping the application. Staging can be run multiple times so that most of the data is copied to the target before migration. This minimizes the actual migration time and application downtime.
- **Migrate** stops the application on the source cluster and recreates its resources on the target cluster. Optionally, you can migrate the workload without stopping the application.



OpenShift\_45\_1019

### 1.3.3. About data copy methods

The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

### 1.3.3.1. File system copy method

The CAM tool copies data files from the source cluster to the replication repository, and from there to the target cluster.

**Table 1.1. File system copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>● Clusters can have different storage classes</li> <li>● Supported for all S3 storage providers</li> <li>● Optional data verification with checksum</li> </ul>	<ul style="list-style-type: none"> <li>● Slower than the snapshot copy method</li> <li>● Optional data verification significantly reduces performance</li> </ul>

### 1.3.3.2. Snapshot copy method

The CAM tool copies a snapshot of the source cluster's data to a cloud provider's object storage, configured as a replication repository. The data is restored on the target cluster.

AWS, Google Cloud Provider, and Microsoft Azure support the snapshot copy method.

**Table 1.2. Snapshot copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>● Faster than the file system copy method</li> </ul>	<ul style="list-style-type: none"> <li>● Cloud provider must support snapshots.</li> <li>● Clusters must be on the same cloud provider.</li> <li>● Clusters must be in the same location or region.</li> <li>● Clusters must have the same storage class.</li> <li>● Storage class must be compatible with snapshots.</li> </ul>

### 1.3.4. About migration hooks

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.



#### NOTE

If you do not want to use Ansible playbooks, you can create a custom container image and add it to a migration plan.

Migration hooks perform tasks such as customizing application quiescence, manually migrating unsupported data types, and updating applications after migration.

A single migration hook runs on a source or target cluster at one of the following migration steps:

- **PreBackup**: Before backup tasks are started on the source cluster
  - **PostBackup**: After backup tasks are complete on the source cluster
  - **PreRestore**: Before restore tasks are started on the target cluster
  - **PostRestore**: After restore tasks are complete on the target cluster
- You can assign one hook to each migration step, up to a maximum of four hooks for a single migration plan.

The default **hook-runner** image is **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**. This image is based on Ansible Runner and includes **python-openshift** for Ansible Kubernetes resources and an updated **oc** binary. You can also create your own hook image with additional Ansible modules or tools.

The Ansible playbook is mounted on a hook container as a ConfigMap. The hook container runs as a Job on a cluster with a specified service account and namespace. The Job runs, even if the initial Pod is evicted or killed, until it reaches the default **backoffLimit (6)** or successful completion.

### 1.3.5. About the Control Plane Migration Assistant

The Control Plane Migration Assistant (CPMA) is a CLI-based tool that assists you in migrating the control plane from OpenShift Container Platform 3.7 (or later) to 4.2. The CPMA processes the OpenShift Container Platform 3 configuration files and generates Custom Resource (CR) manifest files, which are consumed by OpenShift Container Platform 4.2 Operators.

Because OpenShift Container Platform 3 and 4 have significant configuration differences, not all parameters are processed. The CPMA can generate a report that describes whether features are supported fully, partially, or not at all.

#### Configuration files

CPMA uses the Kubernetes and OpenShift Container Platform APIs to access the following configuration files on an OpenShift Container Platform 3 cluster:

- Master configuration file (default: **/etc/origin/master/master-config.yaml**)
- CRI-O configuration file (default: **/etc/crio/crio.conf**)
- etcd configuration file (default: **/etc/etcd/etcd.conf**)
- Image registries file (default: **/etc/containers/registries.conf**)
- Dependent configuration files:
  - Password files (for example, HTPasswd)
  - ConfigMaps
  - Secrets

#### CR Manifests

CPMA generates CR manifests for the following configurations:

- API server CA certificate: **100\_CPMA-cluster-config-APISecret.yaml**

**NOTE**

If you are using an unsigned API server CA certificate, you must add the certificate manually to the target cluster.

- CRI-O: **100\_CPMA-crio-config.yaml**
- Cluster resource quota: **100\_CPMA-cluster-quota-resource-x.yaml**
- Project resource quota: **100\_CPMA-resource-quota-x.yaml**
- Portable image registry (**/etc/registries/registries.conf**) and portable image policy (**/etc/origin/master/master-config.yaml**): **100\_CPMA-cluster-config-image.yaml**
- OAuth providers: **100\_CPMA-cluster-config-oauth.yaml**
- Project configuration: **100\_CPMA-cluster-config-project.yaml**
- Scheduler: **100\_CPMA-cluster-config-scheduler.yaml**
- SDN: **100\_CPMA-cluster-config-sdn.yaml**

## 1.4. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster and an OpenShift Container Platform 3 source cluster. The Cluster Application Migration Operator installs the Cluster Application Migration (CAM) tool on the target cluster by default.

**NOTE**

Optional: You can configure the Cluster Application Migration Operator to install the CAM tool [on an OpenShift Container Platform 3 cluster or on a remote cluster](#) .

In a restricted environment, you can install the Cluster Application Migration Operator from a local mirror registry.

After you have installed the Cluster Application Migration Operator on your clusters, you can launch the CAM tool.

### 1.4.1. Installing the Cluster Application Migration Operator

You can install the Cluster Application Migration Operator with the Operation Lifecycle Manager (OLM) on an OpenShift Container Platform 4.2 target cluster and manually on an OpenShift Container Platform 3 source cluster.

#### 1.4.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

## Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New** → **MigrationController**.
8. Click **Create**.
9. Click **Workloads** → **Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 1.4.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 3 source cluster

You can install the Cluster Application Migration Operator manually on an OpenShift Container Platform 3 source cluster.

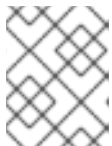
## Prerequisites

- Access to **registry.redhat.io**
- OpenShift Container Platform 3 cluster configured to pull images from **registry.redhat.io**  
To pull images, you must [create an imagestreamsecret](#) and copy it to each node in your cluster.

## Procedure

1. Log in to **registry.redhat.io** with your Red Hat Customer Portal credentials:

```
$ sudo podman login registry.redhat.io
```



### NOTE

If your system is configured for rootless Podman containers, **sudo** is not required for this procedure.

2. Download the **operator.yml** file:

```
$ sudo podman cp $(sudo podman create registry.redhat.io/rhcam-1-2/openshift-migration-rhel7-operator:v1.2):/operator.yml ./
```

- Download the **controller-3.yml** file:

```
$ sudo podman cp $(sudo podman create registry.redhat.io/rhcam-1-2/openshift-migration-rhel7-operator:v1.2):/controller-3.yml ./
```

- Log in to your OpenShift Container Platform 3 cluster.
- Verify that the cluster can authenticate with **registry.redhat.io**:

```
$ oc run test --image registry.redhat.io/ubi8 --command sleep infinity
```

- Create the Cluster Application Migration Operator CR object:

```
$ oc create -f operator.yml
```

The output resembles the following:

```
namespace/openshift-migration created
rolebinding.rbac.authorization.k8s.io/system:deployers created
serviceaccount/migration-operator created
customresourcedefinition.apiextensions.k8s.io/migrationcontrollers.migration.openshift.io
created
role.rbac.authorization.k8s.io/migration-operator created
rolebinding.rbac.authorization.k8s.io/migration-operator created
clusterrolebinding.rbac.authorization.k8s.io/migration-operator created
deployment.apps/migration-operator created
Error from server (AlreadyExists): error when creating "./operator.yml":
rolebindings.rbac.authorization.k8s.io "system:image-builders" already exists 1
Error from server (AlreadyExists): error when creating "./operator.yml":
rolebindings.rbac.authorization.k8s.io "system:image-pullers" already exists
```

- You can ignore **Error from server (AlreadyExists)** messages. They are caused by the Cluster Application Migration Operator creating resources for earlier versions of OpenShift Container Platform 3 that are provided in later releases.

- Create the Migration controller CR object:

```
$ oc create -f controller-3.yml
```

- Verify that the Velero and Restic Pods are running:

```
$ oc get pods -n openshift-migration
```

### 1.4.2. Installing the Cluster Application Migration Operator in a restricted environment

You can install the Cluster Application Migration Operator with the Operation Lifecycle Manager (OLM) on an OpenShift Container Platform 4.2 target cluster and manually on an OpenShift Container Platform 3 source cluster.

For OpenShift Container Platform 4.2, you can build a custom Operator catalog image, push it to a local mirror image registry, and configure OLM to install the Cluster Application Migration Operator from the local registry. A **mapping.txt** file is created when you run the **oc adm catalog mirror** command.

On the OpenShift Container Platform 3 cluster, you can create a manifest file based on the Operator image and edit the file to point to your local image registry. The **image** value in the manifest file uses the **sha256** value from the **mapping.txt** file. Then, you can use the local image to create the Cluster Application Migration Operator.

### 1.4.2.1. Configuring OperatorHub for restricted networks

Cluster administrators can configure OLM and OperatorHub to use local content in restricted network environments.

#### Prerequisites

- Cluster administrator access to an OpenShift Container Platform cluster and its internal registry.
- Separate workstation without network restrictions.
- If pushing images to the OpenShift Container Platform cluster's internal registry, the registry must be exposed with a route.
- **podman** version 1.4.4+

#### Procedure

##### 1. Disable the default OperatorSources.

Add **disableAllDefaultSources: true** to the spec:

```
$ oc patch OperatorHub cluster --type json \
  -p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

This disables the default OperatorSources that are configured by default during an OpenShift Container Platform installation.

##### 2. Retrieve package lists.

To get the list of packages that are available for the default OperatorSources, run the following **curl** commands from your workstation without network restrictions:

```
$ curl https://quay.io/cnr/api/v1/packages?namespace=redhat-operators > packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=community-operators >> packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=certified-operators >> packages.txt
```

Each package in the new **packages.txt** is an Operator that you could add to your restricted network catalog. From this list, you could either pull every Operator or a subset that you would like to expose to users.

##### 3. Pull Operator content.

For a given Operator in the package list, you must pull the latest released content:

```
$ curl https://quay.io/cnr/api/v1/packages/<namespace>/<operator_name>/<release>
```

This example uses the etcd Operator:

- a. Retrieve the digest:

```
$ curl https://quay.io/cnr/api/v1/packages/community-operators/etcd/0.0.12
```

- b. From that JSON, take the digest and use it to pull the gzipped archive:

```
$ curl -XGET https://quay.io/cnr/api/v1/packages/community-operators/etcd/blobs/sha256/8108475ee5e83a0187d6d0a729451ef1ce6d34c44a868a200151c36f3232822b \
-o etcd.tar.gz
```

- c. To pull the information out, you must untar the archive into a **manifests/<operator\_name>** directory with all the other Operators that you want. For example, to untar to an existing directory called **manifests/etcd/**:

```
$ mkdir -p manifests/etcd/ 1
$ tar -xf etcd.tar.gz -C manifests/etcd/
```

- 1** Create different subdirectories for each extracted archive so that files are not overwritten by subsequent extractions for other Operators.

#### 4. Break apart **bundle.yaml** content, if necessary.

In your new **manifests/<operator\_name>** directory, the goal is to get your bundle in the following directory structure:

```
manifests/
├── etcd
│   ├── 0.0.12
│   │   ├── clusterserviceversion.yaml
│   │   └── customresourcedefinition.yaml
│   └── package.yaml
```

If you see files already in this structure, you can skip this step. However, if you instead see only a single file called **bundle.yaml**, you must first break this file up to conform to the required structure.

You must separate the CSV content under **data.clusterServiceVersion** (each file in the list), the CRD content under **data.customResourceDefinition** (each file in the list), and the package content under **data.Package** into their own files.

- a. For the CSV file creation, find the following lines in the **bundle.yaml** file:

```
data:
  clusterServiceVersions: |
```

Omit those lines, but save a new file consisting of the full CSV resource content beginning with the following lines, removing the prepended - character:

#### Example **clusterserviceversion.yaml** file snippet

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
[...]
```



- b. For the CRD file creation, find the following line in the **bundle.yaml** file:

```
customResourceDefinitions: |
```

Omit this line, but save new files consisting of each, full CRD resource content beginning with the following lines, removing the prepended - character:

#### Example customresourcedefinition.yaml file snippet

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
[...]
```

- c. For the package file creation, find the following line in the **bundle.yaml** file:

```
packages: |
```

Omit this line, but save a new file consisting of the package content beginning with the following lines, removing the prepended - character, and ending with a **packageName** entry:

#### Example package.yaml file

```
channels:
- currentCSV: etcdoperator.v0.9.4
  name: singlenamespace-alpha
- currentCSV: etcdoperator.v0.9.4-clusterwide
  name: clusterwide-alpha
defaultChannel: singlenamespace-alpha
packageName: etcd
```

### 5. Identify images required by the Operators you want to use.

Inspect the CSV files of each Operator for **image:** fields to identify the pull specs for any images required by the Operator, and note them for use in a later step.

For example, in the following **deployments** spec of an etcd Operator CSV:

```
spec:
  serviceAccountName: etcd-operator
  containers:
  - name: etcd-operator
    command:
    - etcd-operator
    - --create-crd=false
    image: quay.io/coreos/etcd-
operator@sha256:bd944a211eaf8f31da5e6d69e8541e7cada8f16a9f7a5a570b224789978199
43 1
  env:
  - name: MY_POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: MY_POD_NAME
```

```
valueFrom:
  fieldRef:
    fieldPath: metadata.name
```

- 1 Image required by Operator.

## 6. Create an Operator catalog image.

- a. Save the following to a Dockerfile, for example named **custom-registry.Dockerfile**:

```
FROM registry.redhat.io/openshift4/ose-operator-registry:v4.2.24 AS builder

COPY manifests manifests

RUN /bin/initializer -o ./bundles.db

FROM registry.access.redhat.com/ubi7/ubi

COPY --from=builder /registry/bundles.db /bundles.db
COPY --from=builder /usr/bin/registry-server /registry-server
COPY --from=builder /bin/grpc_health_probe /bin/grpc_health_probe

EXPOSE 50051

ENTRYPOINT ["/registry-server"]

CMD ["--database", "bundles.db"]
```

- b. Use the **podman** command to create and tag the container image from the Dockerfile:

```
$ podman build -f custom-registry.Dockerfile \
  -t <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-
  registry 1
```

- 1 Tag the image for the internal registry of the restricted network OpenShift Container Platform cluster and any namespace.

## 7. Push the Operator catalog image to a registry.

Your new Operator catalog image must be pushed to a registry that the restricted network OpenShift Container Platform cluster can access. This can be the internal registry of the cluster itself or another registry that the cluster has network access to, such as an on-premise Quay Enterprise registry.

For this example, login and push the image to the internal registry OpenShift Container Platform cluster:

```
$ podman push <local_registry_host_name>:
  <local_registry_host_port>/<namespace>/custom-registry
```

## 8. Create a CatalogSource pointing to the new Operator catalog image.

- a. Save the following to a file, for example **my-operator-catalog.yaml**:

```

apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: My Operator Catalog
  sourceType: grpc
  image: <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-registry:latest

```

- b. Create the CatalogSource resource:

```
$ oc create -f my-operator-catalog.yaml
```

- c. Verify the CatalogSource and package manifest are created successfully:

```

# oc get pods -n openshift-marketplace
NAME READY STATUS RESTARTS AGE
my-operator-catalog-6njx6 1/1 Running 0 28s
marketplace-operator-d9f549946-96sgr 1/1 Running 0 26h

# oc get catalogsource -n openshift-marketplace
NAME DISPLAY TYPE PUBLISHER AGE
my-operator-catalog My Operator Catalog grpc 5s

# oc get packagemanifest -n openshift-marketplace
NAME CATALOG AGE
etcd My Operator Catalog 34s

```

You should also be able to view them from the **OperatorHub** page in the web console.

## 9. Mirror the images required by the Operators you want to use.

- a. Determine the images defined by the Operator(s) that you are expecting. This example uses the etcd Operator, requiring the **quay.io/coreos/etcd-operator** image.



### IMPORTANT

This procedure only shows mirroring Operator images themselves and not Operand images, which are the components that an Operator manages. Operand images must be mirrored as well; see each Operator's documentation to identify the required Operand images.

- b. To use mirrored images, you must first create an ImageContentSourcePolicy for each image to change the source location of the Operator catalog image. For example:

```

apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: etcd-operator
spec:
  repositoryDigestMirrors:

```

```
- mirrors:  
- <local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator  
source: quay.io/coreos/etcd-operator
```

- c. Use the **oc image mirror** command from your workstation without network restrictions to pull the image from the source registry and push to the internal registry without being stored locally:

```
$ oc image mirror quay.io/coreos/etcd-operator \  
<local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator
```

You can now install the Operator from the OperatorHub on your restricted network OpenShift Container Platform cluster.

### 1.4.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

#### Prerequisites

- You created a custom Operator catalog and pushed it to a mirror registry.
- You configured OLM to install the Cluster Application Migration Operator from the mirror registry.

#### Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more...**
7. Click **Create New → MigrationController**.
8. Click **Create**.
9. Click **Workloads → Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 1.4.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 3 source cluster in a restricted environment

You can create a manifest file based on the Cluster Application Migration Operator image and edit the manifest to point to your local image registry. Then, you can use the local image to create the Cluster Application Migration Operator on an OpenShift Container Platform 3 source cluster.

#### Prerequisites

- Access to **registry.redhat.io**
- Linux workstation with unrestricted network access
- Mirror registry that supports [Docker v2-2](#)
- Custom Operator catalog pushed to a mirror registry

#### Procedure

1. On the workstation with unrestricted network access, log in to **registry.redhat.io** with your Red Hat Customer Portal credentials:

```
$ sudo podman login registry.redhat.io
```



#### NOTE

If your system is configured for rootless Podman containers, **sudo** is not required for this procedure.

2. Download the **operator.yml** file:

```
$ sudo podman cp $(sudo podman create registry.redhat.io/rhcam-1-2/openshift-migration-rhel7-operator:v1.2):/operator.yml ./
```

3. Download the **controller-3.yml** file:

```
$ sudo podman cp $(sudo podman create registry.redhat.io/rhcam-1-2/openshift-migration-rhel7-operator:v1.2):/controller-3.yml ./
```

4. Obtain the Operator image value from the **mapping.txt** file that was created when you ran the **oc adm catalog mirror** on the OpenShift Container Platform 4 cluster:

```
$ grep openshift-migration-rhel7-operator ./mapping.txt | grep rhcam-1-2
```

The output shows the mapping between the **registry.redhat.io** image and your mirror registry image:

```
registry.redhat.io/rhcam-1-2/openshift-migration-rhel7-operator@sha256:468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a=<registry.apps.example.com>/rhcam-1-2/openshift-migration-rhel7-operator
```

5. Update the **image** and **REGISTRY** values in the **operator.yml** file:

```

containers:
  - name: ansible
    image: <registry.apps.example.com>/rhcam-1-2/openshift-migration-rhel7-
operator@sha256:
<468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a> 1
  ...
  - name: operator
    image: <registry.apps.example.com>/rhcam-1-2/openshift-migration-rhel7-
operator@sha256:
<468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a> 2
  ...
  env:
  - name: REGISTRY
    value: <registry.apps.example.com> 3

```

1 2 Specify your mirror registry and the **sha256** value of the Operator image in the **mapping.txt** file.

3 Specify your mirror registry.

6. Log in to your OpenShift Container Platform 3 cluster.

7. Create the Cluster Application Migration Operator CR object:

```
$ oc create -f operator.yml
```

The output resembles the following:

```

namespace/openshift-migration created
rolebinding.rbac.authorization.k8s.io/system:deployers created
serviceaccount/migration-operator created
customresourcedefinition.apiextensions.k8s.io/migrationcontrollers.migration.openshift.io
created
role.rbac.authorization.k8s.io/migration-operator created
rolebinding.rbac.authorization.k8s.io/migration-operator created
clusterrolebinding.rbac.authorization.k8s.io/migration-operator created
deployment.apps/migration-operator created
Error from server (AlreadyExists): error when creating "./operator.yml":
rolebindings.rbac.authorization.k8s.io "system:image-builders" already exists 1
Error from server (AlreadyExists): error when creating "./operator.yml":
rolebindings.rbac.authorization.k8s.io "system:image-pullers" already exists

```

1 You can ignore **Error from server (AlreadyExists)** messages. They are caused by the Cluster Application Migration Operator creating resources for earlier versions of OpenShift Container Platform 3 that are provided in later releases.

8. Create the Migration controller CR object:

```
$ oc create -f controller-3.yml
```

9. Verify that the Velero and Restic Pods are running:

```
$ oc get pods -n openshift-migration
```

### 1.4.3. Launching the CAM web console

You can launch the CAM web console in a browser.

#### Procedure

1. Log in to the OpenShift Container Platform cluster on which you have installed the CAM tool.
2. Obtain the CAM web console URL by entering the following command:

```
$ oc get -n openshift-migration route/migration -o go-template='https://{{ .spec.host }}'
```

The output resembles the following: **https://migration-openshift-migration.apps.cluster.openshift.com**.

3. Launch a browser and navigate to the CAM web console.



#### NOTE

If you try to access the CAM web console immediately after installing the Cluster Application Migration Operator, the console may not load because the Operator is still configuring the cluster. Wait a few minutes and retry.

4. If you are using self-signed CA certificates, you will be prompted to accept the CA certificate of the source cluster's API server. The web page guides you through the process of accepting the remaining certificates.
5. Log in with your OpenShift Container Platform **username** and **password**.

## 1.5. CONFIGURING A REPLICATION REPOSITORY

You must configure an object storage to use as a replication repository. The Cluster Application Migration tool copies data from the source cluster to the replication repository, and then from the replication repository to the target cluster.

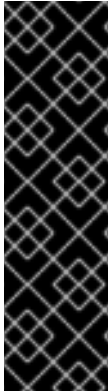
The CAM tool supports the [file system and snapshot data copy methods](#) for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

The following storage providers are supported:

- [Multi-Cloud Object Gateway \(MCG\)](#)
- [Amazon Web Services \(AWS\) S3](#)
- [Google Cloud Provider \(GCP\)](#)
- [Microsoft Azure](#)
- Generic S3 object storage, for example, Minio or Ceph S3

The source and target clusters must have network access to the replication repository during migration.

In a restricted environment, you can create an internally hosted replication repository. If you use a proxy server, you must ensure that your replication repository is whitelisted.



## IMPORTANT

Configuring Multi-Cloud Object Gateway as a replication repository for migration is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

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

### 1.5.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository

You can install the OpenShift Container Storage Operator and configure a Multi-Cloud Object Gateway (MCG) storage bucket as a replication repository.

#### 1.5.1.1. Installing the OpenShift Container Storage Operator

You can install the OpenShift Container Storage Operator from OperatorHub.

##### Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use **Filter by keyword** (in this case, **OCS**) to find the **OpenShift Container Storage Operator**.
3. Select the **OpenShift Container Storage Operator** and click **Install**.
4. Select an **Update Channel**, **Installation Mode**, and **Approval Strategy**.
5. Click **Subscribe**.

On the **Installed Operators** page, the **OpenShift Container Storage Operator** appears in the **openshift-storage** project with the status **Succeeded**.

#### 1.5.1.2. Creating the Multi-Cloud Object Gateway storage bucket

You can create the Multi-Cloud Object Gateway (MCG) storage bucket's Custom Resources (CRs).

##### Procedure

1. Log in to the OpenShift Container Platform cluster:

```
$ oc login
```

2. Create the **NooBaa** CR configuration file, **noobaa.yml**, with the following content:

```
apiVersion: noobaa.io/v1alpha1
kind: NooBaa
metadata:
```



```

name: noobaa
namespace: openshift-storage
spec:
  dbResources:
    requests:
      cpu: 0.5 1
      memory: 1Gi
  coreResources:
    requests:
      cpu: 0.5 2
      memory: 1Gi

```

**1** **2** For a very small cluster, you can change the **cpu** value to **0.1**.

3. Create the **NooBaa** object:

```
$ oc create -f noobaa.yml
```

4. Create the **BackingStore** CR configuration file, **bs.yml**, with the following content:

```

apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
    - noobaa.io/finalizer
  labels:
    app: noobaa
    name: mcg-pv-pool-bs
    namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 1
    resources:
      requests:
        storage: 50Gi 2
    storageClass: gp2 3
  type: pv-pool

```

**1** Specify the number of volumes in the PV pool.

**2** Specify the size of the volumes.

**3** Specify the storage class.

5. Create the **BackingStore** object:

```
$ oc create -f bs.yml
```

6. Create the **BucketClass** CR configuration file, **bc.yml**, with the following content:

```

apiVersion: noobaa.io/v1alpha1
kind: BucketClass

```

```

metadata:
  labels:
    app: noobaa
    name: mcg-pv-pool-bc
    namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
      - backingStores:
          - mcg-pv-pool-bs
        placement: Spread

```

7. Create the **BucketClass** object:

```
$ oc create -f bc.yml
```

8. Create the **ObjectBucketClaim** CR configuration file, **obc.yml**, with the following content:

```

apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: migstorage
  namespace: openshift-storage
spec:
  bucketName: migstorage 1
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: mcg-pv-pool-bc

```

- 1** Record the bucket name for adding the replication repository to the CAM web console.

9. Create the **ObjectBucketClaim** object:

```
$ oc create -f obc.yml
```

10. Watch the resource creation process to verify that the **ObjectBucketClaim** status is **Bound**:

```
$ watch -n 30 'oc get -n openshift-storage objectbucketclaim migstorage -o yaml'
```

This process can take five to ten minutes.

11. Obtain and record the following values, which are required when you add the replication repository to the CAM web console:

- S3 endpoint:

```
$ oc get route -n openshift-storage s3
```

- S3 provider access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_ACCESS_KEY_ID }}' | base64 -d
```

- S3 provider secret access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_SECRET_ACCESS_KEY }}' | base64 -d
```

## 1.5.2. Configuring an AWS S3 storage bucket as a replication repository

You can configure an AWS S3 storage bucket as a replication repository.

### Prerequisites

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

### Procedure

1. Create an AWS S3 bucket:

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

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

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

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

3. Create an EC2 EBS snapshot policy:

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

```

        "ec2:CreateVolume",
        "ec2:CreateSnapshot",
        "ec2>DeleteSnapshot"
    ],
    "Resource": "*"
}
]
}
EOF

```

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

```

$ cat > velero-s3-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3>DeleteObject",
        "s3:PutObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket_name>/*" 1
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": [
        "arn:aws:s3:::<bucket_name>" 2
      ]
    }
  ]
}
EOF

```

- 1** **2** To grant access to a single S3 bucket, specify the bucket name. To grant access to all AWS S3 buckets, specify \* instead of a bucket name:

```

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

```

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

```

$ aws iam put-user-policy \
  --user-name velero \

```

```
--policy-name velero-ebs \
--policy-document file://velero-ec2-snapshot-policy.json
```

- Attach the AWS S3 policy to **velero**:

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

- Create an access key for **velero**:

```
$ aws iam create-access-key --user-name velero
{
  "AccessKey": {
    "UserName": "velero",
    "Status": "Active",
    "CreateDate": "2017-07-31T22:24:41.576Z",
    "SecretAccessKey": <AWS_SECRET_ACCESS_KEY>, 1
    "AccessKeyId": <AWS_ACCESS_KEY_ID> 2
  }
}
```

- 1** Record the **AWS\_SECRET\_ACCESS\_KEY** and the **AWS\_ACCESS\_KEY\_ID** for adding the AWS repository to the CAM web console.

### 1.5.3. Configuring a Google Cloud Provider storage bucket as a replication repository

You can configure a Google Cloud Provider (GCP) storage bucket as a replication repository.

#### Prerequisites

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

#### Procedure

- Run **gsutil init** to log in:

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

```
Your current configuration has been set to: [default]
```

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

2. Set the **BUCKET** variable:

```
$ BUCKET=<bucket_name> 1
```

- 1 Specify your bucket name.

3. Create a storage bucket:

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

4. Set the **PROJECT\_ID** variable to your active project:

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

5. Create a **velero** service account:

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

6. Set the **SERVICE\_ACCOUNT\_EMAIL** variable to the service account's email address:

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

7. Grant permissions to the service account:

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

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

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

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

- Save the service account's keys to the **credentials-velero** file in the current directory:

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

### 1.5.4. Configuring a Microsoft Azure Blob storage container as a replication repository

You can configure a Microsoft Azure Blob storage container as a replication repository.

#### Prerequisites

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

#### Procedure

- Set the **AZURE\_RESOURCE\_GROUP** variable:

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

- Create an Azure resource group:

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

- Specify your location.

- Set the **AZURE\_STORAGE\_ACCOUNT\_ID** variable:

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

- Create an Azure storage account:

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

- Set the **BLOB\_CONTAINER** variable:

```
$ BLOB_CONTAINER=velero
```

- Create an Azure Blob storage container:

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

- Create a service principal and credentials for **velero**:

```
$ AZURE_SUBSCRIPTION_ID=`az account list --query '[?isDefault].id' -o tsv`
$ AZURE_TENANT_ID=`az account list --query '[?isDefault].tenantId' -o tsv`
$ AZURE_CLIENT_SECRET=`az ad sp create-for-rbac --name "velero" --role "Contributor" --
query 'password' -o tsv`
$ AZURE_CLIENT_ID=`az ad sp list --display-name "velero" --query '[0].appId' -o tsv`
```

- Save the service principal's credentials in the **credentials-velero** file:

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

## 1.6. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE

You can migrate application workloads by adding your clusters and replication repository to the CAM web console. Then, you can create and run a migration plan.

If your cluster or replication repository are secured with self-signed certificates, you can create a CA certificate bundle file or disable SSL verification.

### 1.6.1. Creating a CA certificate bundle file

If you use a self-signed certificate to secure a cluster or a replication repository, certificate verification might fail with the following error message: **Certificate signed by unknown authority**.

You can create a custom CA certificate bundle file and upload it in the CAM web console when you add a cluster or a replication repository.

#### Procedure

Download a CA certificate from a remote endpoint and save it as a CA bundle file:

```
$ echo -n | openssl s_client -connect <host_FQDN>:<port> \ 1
| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > <ca_bundle.cert> 2
```





6. Click **Add cluster**.

The cluster appears in the **Clusters** section.

### 1.6.3. Adding a replication repository to the CAM web console

You can add an object storage bucket as a replication repository to the CAM web console.

#### Prerequisites

- You must configure an object storage bucket for migrating the data.

#### Procedure

1. Log in to the CAM web console.
2. In the **Replication repositories** section, click **Add repository**.
3. Select a **Storage provider type** and fill in the following fields:
  - **AWS** for AWS S3, MCG, and generic S3 providers:
    - **Replication repository name**: Specify the replication repository name in the CAM web console.
    - **S3 bucket name**: Specify the name of the S3 bucket you created.
    - **S3 bucket region**: Specify the S3 bucket region. **Required** for AWS S3. **Optional** for other S3 providers.
    - **S3 endpoint**: Specify the URL of the S3 service, not the bucket, for example, **https://<s3-storage.apps.cluster.com>**. **Required** for a generic S3 provider. You must use the **https://** prefix.
    - **S3 provider access key**: Specify the **<AWS\_SECRET\_ACCESS\_KEY>** for AWS or the S3 provider access key for MCG.
    - **S3 provider secret access key**: Specify the **<AWS\_ACCESS\_KEY\_ID>** for AWS or the S3 provider secret access key for MCG.
    - **Require SSL verification**: Clear this check box if you are using a generic S3 provider.
    - If you use a custom CA bundle, click **Browse** and browse to the Base64-encoded CA bundle file.
  - **GCP**:
    - **Replication repository name**: Specify the replication repository name in the CAM web console.
    - **GCP bucket name**: Specify the name of the GCP bucket.
    - **GCP credential JSON blob**: Specify the string in the **credentials-velero** file.
  - **Azure**:
    - **Replication repository name**: Specify the replication repository name in the CAM web console.

- **Azure resource group:** Specify the resource group of the Azure Blob storage.
  - **Azure storage account name:** Specify the Azure Blob storage account name.
  - **Azure credentials - INI file contents:** Specify the string in the **credentials-velero** file.
4. Click **Add repository** and wait for connection validation.
  5. Click **Close**.  
The new repository appears in the **Replication repositories** section.

### 1.6.4. Changing migration plan limits for large migrations

You can change the migration plan limits for large migrations.



#### IMPORTANT

Changes should first be tested in your environment to avoid a failed migration.

A single migration plan has the following default limits:

- 10 namespaces  
If this limit is exceeded, the CAM web console displays a **Namespace limit exceeded** error and you cannot create a migration plan.
- 100 Pods  
If the Pod limit is exceeded, the CAM web console displays a warning message similar to the following example: **Plan has been validated with warning condition(s). See warning message. Pod limit: 100 exceeded, found: 104.**
- 100 persistent volumes  
If the persistent volume limit is exceeded, the CAM web console displays a similar warning message.

#### Procedure

1. Edit the Migration controller CR:

```
$ oc get migrationcontroller -n openshift-migration
NAME AGE
migration-controller 5d19h

$ oc edit migrationcontroller -n openshift-migration
```

2. Update the following parameters:

```
...
migration_controller: true

# This configuration is loaded into mig-controller, and should be set on the
# cluster where `migration_controller: true`
mig_pv_limit: 100
```

```
mig_pod_limit: 100  
mig_namespace_limit: 10  
...
```

## 1.6.5. Creating a migration plan in the CAM web console

You can create a migration plan in the CAM web console.

### Prerequisites

- The CAM web console must contain the following:
  - Source cluster
  - Target cluster, which is added automatically during the CAM tool installation
  - Replication repository
- The source and target clusters must have network access to each other and to the replication repository.
- If you use snapshots to copy data, the source and target clusters must run on the same cloud provider (AWS, GCP, or Azure) and in the same region.

### Procedure

1. Log in to the CAM web console.
2. In the **Plans** section, click **Add plan**.
3. Enter the **Plan name** and click **Next**.  
The **Plan name** can contain up to 253 lower-case alphanumeric characters (**a-z, 0-9**). It must not contain spaces or underscores (\_).
4. Select a **Source cluster**.
5. Select a **Target cluster**.
6. Select a **Replication repository**.
7. Select the projects to be migrated and click **Next**.
8. Select **Copy** or **Move** for the PVs:
  - **Copy** copies the data in a source cluster's PV to the replication repository and then restores it on a newly created PV, with similar characteristics, in the target cluster.  
Optional: You can verify data copied with the filesystem method by selecting **Verify copy**. This option, which generates a checksum for each source file and checks it after restoration, significantly reduces performance.
  - **Move** unmounts a remote volume (for example, NFS) from the source cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.

9. Click **Next**.
10. Select a **Copy method** for the PVs:
  - **Snapshot** backs up and restores the disk using the cloud provider's snapshot functionality. It is significantly faster than **Filesystem**.

**NOTE**

The storage and clusters must be in the same region and the storage class must be compatible.

- **Filesystem** copies the data files from the source disk to a newly created target disk.
11. Select a **Storage class** for the PVs.  
If you selected the **Filesystem** copy method, you can change the storage class during migration, for example, from Red Hat Gluster Storage or NFS storage to Red Hat Ceph Storage.
  12. Click **Next**.
  13. If you want to add a migration hook, click **Add Hook** and perform the following steps:
    - a. Specify the name of the hook.
    - b. Select **Ansible playbook** to use your own playbook or **Custom container image** for a hook written in another language.
    - c. Click **Browse** to upload the playbook.
    - d. Optional: If you are not using the default Ansible runtime image, specify your custom Ansible image.
    - e. Specify the cluster on which you want the hook to run.
    - f. Specify the service account name.
    - g. Specify the namespace.
    - h. Select the migration step at which you want the hook to run:
      - PreBackup: Before backup tasks are started on the source cluster
      - PostBackup: After backup tasks are complete on the source cluster
      - PreRestore: Before restore tasks are started on the target cluster
      - PostRestore: After restore tasks are complete on the target cluster
  14. Click **Add**.  
You can add up to four hooks to a migration plan, assigning each hook to a different migration step.
  15. Click **Finish**.
  16. Click **Close**.  
The migration plan appears in the **Plans** section.

## 1.6.6. Running a migration plan in the CAM web console


You can stage or migrate applications and data with the migration plan you created in the CAM web console.

### Prerequisites

The CAM web console must contain the following:

- Source cluster
- Target cluster, which is added automatically during the CAM tool installation
- Replication repository
- Valid migration plan

### Procedure

1. Log in to the CAM web console on the target cluster.
2. Select a migration plan.
3. Click **Stage** to copy data from the source cluster to the target cluster without stopping the application.  
You can run **Stage** multiple times to reduce the actual migration time.
4. When you are ready to migrate the application workload, click **Migrate**.  
**Migrate** stops the application workload on the source cluster and recreates its resources on the target cluster.
5. Optional: In the **Migrate** window, you can select **Do not stop applications on the source cluster during migration**.
6. Click **Migrate**.
7. Optional: To stop a migration in progress, click the Options menu  and select **Cancel**.
8. When the migration is complete, verify that the application migrated successfully in the OpenShift Container Platform web console:
  - a. Click **Home** → **Projects**.
  - b. Click the migrated project to view its status.
  - c. In the **Routes** section, click **Location** to verify that the application is functioning, if applicable.
  - d. Click **Workloads** → **Pods** to verify that the Pods are running in the migrated namespace.
  - e. Click **Storage** → **Persistent volumes** to verify that the migrated persistent volume is correctly provisioned.

## 1.7. MIGRATING CONTROL PLANE SETTINGS WITH THE CONTROL PLANE MIGRATION ASSISTANT (CPMA)

The Control Plane Migration Assistant (CPMA) is a CLI-based tool that assists you in migrating the control plane from OpenShift Container Platform 3.7 (or later) to 4.2. The CPMA processes the OpenShift Container Platform 3 configuration files and generates Custom Resource (CR) manifest files, which are consumed by OpenShift Container Platform 4.2 Operators.

### 1.7.1. Installing the Control Plane Migration Assistant

You can download the Control Plane Migration Assistant (CPMA) binary file from the Red Hat Customer Portal and install it on Linux, MacOSX, or Windows operating systems.

#### Procedure

1. In the [Red Hat Customer Portal](#), navigate to **Downloads** → **Red Hat OpenShift Container Platform**.
2. On the **Download Red Hat OpenShift Container Platform** page, select **Red Hat OpenShift Container Platform** from the **Product Variant** list.
3. Select **CPMA 1.0 for RHEL 7** from the **Version** list. This binary works on RHEL 7 and RHEL 8.
4. Click **Download Now** to download **cpma** for Linux or MacOSX or **cpma.exe** for Windows.
5. Save the file in a directory defined as **\$PATH** for Linux or MacOSX or **%PATH%** for Windows.
6. For Linux, make the file executable:

```
$ sudo chmod +x cpma
```

### 1.7.2. Using the Control Plane Migration Assistant

The Control Plane Migration Assistant (CPMA) generates CR manifests, which are consumed by OpenShift Container Platform 4.2 Operators, and a report that indicates which OpenShift Container Platform 3 features are supported fully, partially, or not at all.

The CPMA can run in remote mode, retrieving the configuration files from the source cluster using SSH, or in local mode, using local copies of the source cluster's configuration files.

#### Prerequisites

- The source cluster must be OpenShift Container Platform 3.7 or later.
- The source cluster must be updated to the latest synchronous release.
- An environment health check must be run on the source cluster to confirm that there are no diagnostic errors or warnings.
- The CPMA binary must be executable.
- You must have **cluster-admin** privileges for the source cluster.

#### Procedure

1. Log in to the OpenShift Container Platform 3 cluster:

```
$ oc login https://<master1.example.com> 1
```

- 1. OpenShift Container Platform 3 master node. You must be logged in to the cluster to receive a token for the Kubernetes and OpenShift Container Platform APIs.
2. Run the CPMA. Each prompt requires you to provide input, as in the following example:

```

$ cpma --manifests=false 1
? Do you wish to save configuration for future use? true
? What will be the source for OCP3 config files? Remote host 2
? Path to crio config file /etc/crio/crio.conf
? Path to etcd config file /etc/etcd/etcd.conf
? Path to master config file /etc/origin/master/master-config.yaml
? Path to node config file /etc/origin/node/node-config.yaml
? Path to registries config file /etc/containers/registries.conf
? Do wish to find source cluster using KUBECONFIG or prompt it? KUBECONFIG
? Select cluster obtained from KUBECONFIG contexts master1-example-com:443
? Select master node master1.example.com
? SSH login root 3
? SSH Port 22
? Path to private SSH key /home/user/.ssh/openshift_key
? Path to application data, skip to use current directory .
INFO[29 Aug 19 00:07 UTC] Starting manifest and report generation
INFO[29 Aug 19 00:07 UTC] Transform:Starting for - API
INFO[29 Aug 19 00:07 UTC] APITransform::Extract
INFO[29 Aug 19 00:07 UTC] APITransform::Transform:Reports
INFO[29 Aug 19 00:07 UTC] Transform:Starting for - Cluster
INFO[29 Aug 19 00:08 UTC] ClusterTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportQuotas
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportPVs
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportNamespaces
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportNodes
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportRBAC
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportStorageClasses
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Crio
INFO[29 Aug 19 00:08 UTC] CrioTransform::Extract
WARN[29 Aug 19 00:08 UTC] Skipping Crio: No configuration file available
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Docker
INFO[29 Aug 19 00:08 UTC] DockerTransform::Extract
INFO[29 Aug 19 00:08 UTC] DockerTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - ETCD
INFO[29 Aug 19 00:08 UTC] ETCDTransform::Extract
INFO[29 Aug 19 00:08 UTC] ETCDTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - OAuth
INFO[29 Aug 19 00:08 UTC] OAuthTransform::Extract
INFO[29 Aug 19 00:08 UTC] OAuthTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - SDN
INFO[29 Aug 19 00:08 UTC] SDNTransform::Extract
INFO[29 Aug 19 00:08 UTC] SDNTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Image
INFO[29 Aug 19 00:08 UTC] ImageTransform::Extract
INFO[29 Aug 19 00:08 UTC] ImageTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Project
INFO[29 Aug 19 00:08 UTC] ProjectTransform::Extract
INFO[29 Aug 19 00:08 UTC] ProjectTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Flushing reports to disk

```



```
INFO[29 Aug 19 00:08 UTC] Report:Added: report.json
INFO[29 Aug 19 00:08 UTC] Report:Added: report.html
INFO[29 Aug 19 00:08 UTC] Successfully finished transformations
```

- 1 **--manifests=false**: Without generating CR manifests
- 2 **Remote host**: Remote mode
- 3 **SSH login**: The SSH user must have **sudo** permissions on the OpenShift Container Platform 3 cluster in order to access the configuration files.

The CPMA creates the following files and directory in the current directory if you did not specify an output directory:

- **cpma.yaml** file: Configuration options that you provided when you ran the CPMA
  - **master1.example.com/**: Configuration files from the master node
  - **report.json**: JSON-encoded report
  - **report.html**: HTML-encoded report
3. Open the **report.html** file in a browser to view the CPMA report.
  4. If you generate CR manifests, apply the CR manifests to the OpenShift Container Platform 4.2 cluster, as in the following example:

```
$ oc apply -f 100_CPMA-cluster-config-secret-htpasswd-secret.yaml
```

## 1.8. TROUBLESHOOTING

You can view the migration Custom Resources (CRs) and download logs to troubleshoot a failed migration.

If the application was stopped during the failed migration, you must roll it back manually in order to prevent data corruption.

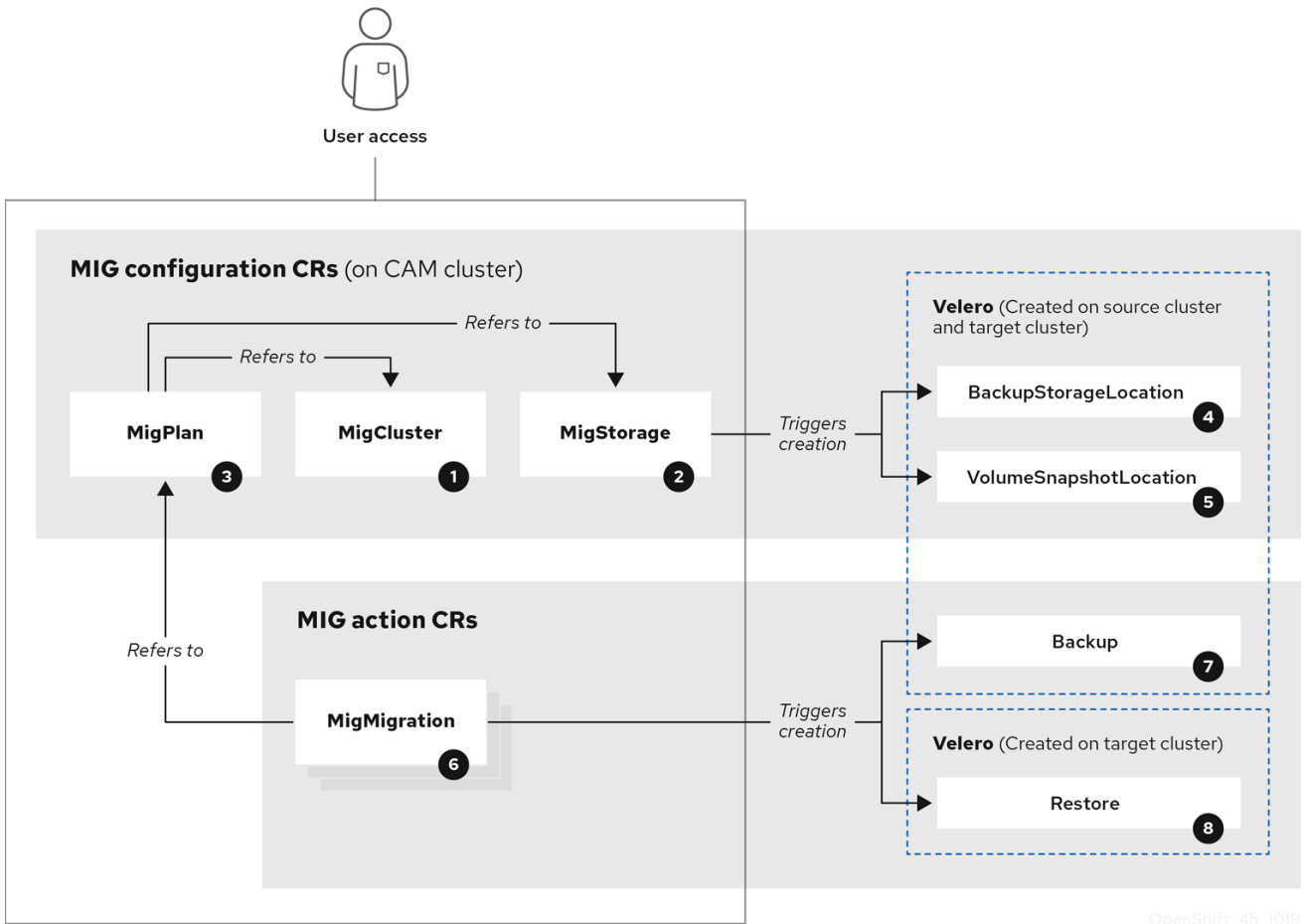


### NOTE

Manual rollback is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

### 1.8.1. Viewing migration Custom Resources

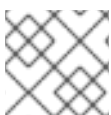
The Cluster Application Migration (CAM) tool creates the following Custom Resources (CRs):



OpenShift\_45\_1019

- 1 MigCluster (configuration, CAM cluster): Cluster definition
- 2 MigStorage (configuration, CAM cluster): Storage definition
- 3 MigPlan (configuration, CAM cluster): Migration plan

The MigPlan CR describes the source and target clusters, repository, and namespace(s) being migrated. It is associated with 0, 1, or many MigMigration CRs.



**NOTE**

Deleting a MigPlan CR deletes the associated MigMigration CRs.

- 4 BackupStorageLocation (configuration, CAM cluster): Location of Velero backup objects
- 5 VolumeSnapshotLocation (configuration, CAM cluster): Location of Velero volume snapshots
- 6 MigMigration (action, CAM cluster): Migration, created during migration

A MigMigration CR is created every time you stage or migrate data. Each MigMigration CR is associated with a MigPlan CR.

**7 Backup** (action, source cluster): When you run a migration plan, the MigMigration CR creates two Velero backup CRs on each source cluster:

- Backup CR #1 for Kubernetes objects
- Backup CR #2 for PV data

**8 Restore** (action, target cluster): When you run a migration plan, the MigMigration CR creates two Velero restore CRs on the target cluster:

- Restore CR #1 (using Backup CR #2) for PV data
- Restore CR #2 (using Backup CR #1) for Kubernetes objects

## Procedure

1. Get the CR name:

```
$ oc get <migration_cr> -n openshift-migration 1
```

- 1** Specify the migration CR, for example, **migmigration**.

The output is similar to the following:

```
NAME                                AGE
88435fe0-c9f8-11e9-85e6-5d593ce65e10 6m42s
```

2. View the CR:

```
$ oc describe <migration_cr> <88435fe0-c9f8-11e9-85e6-5d593ce65e10> -n openshift-migration
```

The output is similar to the following examples.

## MigMigration example

```
name:      88435fe0-c9f8-11e9-85e6-5d593ce65e10
namespace: openshift-migration
labels:    <none>
annotations: touch: 3b48b543-b53e-4e44-9d34-33563f0f8147
apiVersion: migration.openshift.io/v1alpha1
kind:      MigMigration
metadata:
  creationTimestamp: 2019-08-29T01:01:29Z
  generation:       20
  resourceVersion:  88179
  selfLink:         /apis/migration.openshift.io/v1alpha1/namespaces/openshift-
migration/migmigrations/88435fe0-c9f8-11e9-85e6-5d593ce65e10
  uid:              8886de4c-c9f8-11e9-95ad-0205fe66cbb6
spec:
  migPlanRef:
    name:      socks-shop-mig-plan
    namespace: openshift-migration
```

```

quiescePods: true
stage:      false
status:
conditions:
  category:      Advisory
  durable:       True
  lastTransitionTime: 2019-08-29T01:03:40Z
  message:       The migration has completed successfully.
  reason:        Completed
  status:        True
  type:          Succeeded
phase:        Completed
startTimestamp: 2019-08-29T01:01:29Z
events:       <none>

```

## Velero backup CR #2 example (PV data)

```

apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.105.179:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-44dd3bd5-c9f8-11e9-95ad-0205fe66cbb6
  creationTimestamp: "2019-08-29T01:03:15Z"
  generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
  generation: 1
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    velero.io/storage-location: myrepo-vpzq9
  name: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
  namespace: openshift-migration
  resourceVersion: "87313"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/backups/88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
  uid: c80dbbc0-c9f8-11e9-95ad-0205fe66cbb6
spec:
  excludedNamespaces: []
  excludedResources: []
  hooks:
    resources: []
  includeClusterResources: null
  includedNamespaces:
  - sock-shop
  includedResources:
  - persistentvolumes
  - persistentvolumeclaims
  - namespaces
  - imagestreams
  - imagestreamtags
  - secrets
  - configmaps

```

```

- pods
labelSelector:
  matchLabels:
    migration-included-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
storageLocation: myrepo-vpzq9
ttl: 720h0m0s
volumeSnapshotLocations:
- myrepo-wv6fx
status:
  completionTimestamp: "2019-08-29T01:02:36Z"
  errors: 0
  expiration: "2019-09-28T01:02:35Z"
  phase: Completed
  startTimestamp: "2019-08-29T01:02:35Z"
  validationErrors: null
  version: 1
  volumeSnapshotsAttempted: 0
  volumeSnapshotsCompleted: 0
  warnings: 0

```

## Velero restore CR #2 example (Kubernetes resources)

```

apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.90.187:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-36f54ca7-c925-11e9-825a-06fa9fb68c88
  creationTimestamp: "2019-08-28T00:09:49Z"
  generateName: e13a1b60-c927-11e9-9555-d129df7f3b96-
  generation: 3
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: e18252c9-c927-11e9-825a-06fa9fb68c88
    migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
  name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  namespace: openshift-migration
  resourceVersion: "82329"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  includedNamespaces: null
  includedResources: null

```

```


namespaceMapping: null
restorePVs: true
status:
  errors: 0
  failureReason: ""
  phase: Completed
  validationErrors: null
  warnings: 15

```

## 1.8.2. Downloading migration logs

You can download the Velero, Restic, and Migration controller logs in the CAM web console to troubleshoot a failed migration.

### Procedure

1. Log in to the CAM console.
2. Click **Plans** to view the list of migration plans.
3. Click the **Options** menu  of a specific migration plan and select **Logs**.
4. Click **Download Logs** to download the logs of the Migration controller, Velero, and Restic for all clusters.
5. To download a specific log:
  - a. Specify the log options:
    - **Cluster:** Select the source, target, or CAM host cluster.
    - **Log source:** Select **Velero**, **Restic**, or **Controller**.
    - **Pod source:** Select the Pod name, for example, **controller-manager-78c469849c-v6wcf**  
The selected log is displayed.

You can clear the log selection settings by changing your selection.

- b. Click **Download Selected** to download the selected log.

Optionally, you can access the logs by using the CLI, as in the following example:

```

$ oc get pods -n openshift-migration | grep controller
controller-manager-78c469849c-v6wcf      1/1   Running   0      4h49m

$ oc logs controller-manager-78c469849c-v6wcf -f -n openshift-migration

```

## 1.8.3. Updating deprecated API GroupVersionKinds

In OpenShift Container Platform 4.2, some API **GroupVersionKinds** (GVKs) that are used by OpenShift Container Platform 3.x are [deprecated](#).

If your source cluster uses deprecated GVKs, the following warning is displayed when you create a migration plan: **Some namespaces contain GVKs incompatible with destination cluster**. You can click **See details** to view the namespace and the incompatible GVKs.



## NOTE

This warning does not block the migration.

During migration, the deprecated GVKs are saved in the Velero Backup Custom Resource (CR) #1 for Kubernetes objects. You can download the Backup CR, extract the deprecated GVK **yaml** files, and update them with the **oc convert** command. Then you create the updated GVKs on the target cluster.

## Procedure

1. Run the migration plan.
2. View the MigPlan CR:

```
$ oc describe migplan <migplan_name> -n openshift-migration 1
```

- 1** Specify the name of the migration plan.

The output is similar to the following:

```
metadata:
  ...
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a 1
status:
  ...
conditions:
- category: Warn
  lastTransitionTime: 2020-04-30T17:16:23Z
  message: 'Some namespaces contain GVKs incompatible with destination cluster.
  See: `incompatibleNamespaces` for details'
  status: "True"
  type: GVKsIncompatible
incompatibleNamespaces:
- gvks:
  - group: batch
    kind: cronjobs 2
    version: v2alpha1
  - group: batch
    kind: scheduledjobs 3
    version: v2alpha1
```

- 1** Record the MigPlan UID.
- 2** **3** Record the deprecated GVKs.

3. Get the MigMigration name associated with the MigPlan UID:

```
$ oc get migmigration -o json | jq -r '.items[] | select(.metadata.ownerReferences[].uid=="<migplan_uid>") | .metadata.name' 1
```

- 1 Specify the MigPlan UID.

4. Get the MigMigration UID associated with the MigMigration name:

```
$ oc get migmigration <migmigration_name> -o jsonpath='{.metadata.uid}' 1
```

- 1 Specify the MigMigration name.

5. Get the Velero Backup name associated with the MigMigration UID:

```
$ oc get backup.velero.io --selector migration-initial-backup="<migmigration_uid>" -o jsonpath='{.items[*].metadata.name}' 1
```

- 1 Specify the MigMigration UID.

6. Download the contents of the Velero Backup to your local machine:

- For AWS S3:

```
$ aws s3 cp s3://<bucket_name>/velero/backups/<backup_name> <backup_local_dir> --recursive 1
```

- 1 Specify the bucket, backup name, and your local backup directory name.

- For GCP:

```
$ gsutil cp gs://<bucket_name>/velero/backups/<backup_name> <backup_local_dir> --recursive 1
```

- 1 Specify the bucket, backup name, and your local backup directory name.

- For Azure:

```
$ azcopy copy 'https://velerobackups.blob.core.windows.net/velero/backups/<backup_name>' '<backup_local_dir>' --recursive 1
```

- 1 Specify the backup name and your local backup directory name.

7. Extract the Velero Backup archive file:

```
$ tar -xvf <backup_local_dir>/<backup_name>.tar.gz -C <backup_local_dir>
```

8. Run **oc convert** in offline mode on each deprecated GVK:



```
$ oc convert <backup_local_dir>/resources/<gvk>.yaml 1
```

- 1 Specify the deprecated GVK.

9. Create the converted GVK on the target cluster:

```
$ oc create -f <gvk>.yaml 1
```

- 1 Specify the converted GVK.

## 1.8.4. Error messages

### 1.8.4.1. Restic timeout error message in the Velero Pod log

If a migration fails because Restic times out, the following error appears in the Velero Pod log:

```
level=error msg="Error backing up item" backup=velero/monitoring error="timed out waiting for all
PodVolumeBackups to complete"
error.file="/go/src/github.com/heptio/velero/pkg/restic/backupper.go:165"
error.function="github.com/heptio/velero/pkg/restic.(*backupper).BackupPodVolumes" group=v1
```

The default value of **restic\_timeout** is one hour. You can increase this parameter for large migrations, keeping in mind that a higher value may delay the return of error messages.

#### Procedure

1. In the OpenShift Container Platform web console, navigate to **Operators** → **Installed Operators**.
2. Click **Cluster Application Migration Operator**.
3. In the **MigrationController** tab, click **migration-controller**.
4. In the **YAML** tab, update the following parameter value:

```
spec:
  restic_timeout: 1h 1
```

- 1 Valid units are **h** (hours), **m** (minutes), and **s** (seconds), for example, **3h30m15s**.

5. Click **Save**.

### 1.8.4.2. ResticVerifyErrors in the MigMigration Custom Resource

If data verification fails when migrating a PV with the filesystem data copy method, the following error appears in the MigMigration Custom Resource (CR):

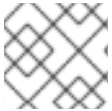
```
status:
  conditions:
  - category: Warn
```

```

durable: true
lastTransitionTime: 2020-04-16T20:35:16Z
message: There were verify errors found in 1 Restic volume restores. See restore `<registry-
example-migration-rvwcm>`
  for details 1
status: "True"
type: ResticVerifyErrors 2

```

- 1** The error message identifies the Restore CR name.
- 2** **ResticErrors** also appears. **ResticErrors** is a general error warning that includes verification errors.



## NOTE

A data verification error does not cause the migration process to fail.

You can check the target cluster's Restore CR to identify the source of the data verification error.

## Procedure

1. Log in to the target cluster.
2. View the Restore CR:

```
$ oc describe <registry-example-migration-rvwcm> -n openshift-migration
```

The output identifies the PV with **PodVolumeRestore** errors:

```

status:
  phase: Completed
  podVolumeRestoreErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rvwcm-98t49>
    namespace: openshift-migration
  podVolumeRestoreResticErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rvwcm-98t49>
    namespace: openshift-migration

```

3. View the **PodVolumeRestore** CR:

```
$ oc describe <migration-example-rvwcm-98t49>
```

The output identifies the Restic Pod that logged the errors:

```

completionTimestamp: 2020-05-01T20:49:12Z
errors: 1
resticErrors: 1
...
resticPod: <restic-nr2v5>

```

4. View the Restic Pod log:

```
$ oc logs -f restic-nr2v5
```

### 1.8.5. Manually rolling back a migration

If your application was stopped during a failed migration, you must roll it back manually in order to prevent data corruption in the PV.

This procedure is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

#### Procedure

1. On the target cluster, switch to the migrated project:

```
$ oc project <project>
```

2. Get the deployed resources:

```
$ oc get all
```

3. Delete the deployed resources to ensure that the application is not running on the target cluster and accessing data on the PVC:

```
$ oc delete <resource_type>
```

4. To stop a DaemonSet without deleting it, update the **nodeSelector** in the YAML file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: hello-daemonset
spec:
  selector:
    matchLabels:
      name: hello-daemonset
  template:
    metadata:
      labels:
        name: hello-daemonset
    spec:
      nodeSelector:
        role: worker 1
```

- 1 Specify a **nodeSelector** value that does not exist on any node.

5. Update each PV's reclaim policy so that unnecessary data is removed. During migration, the reclaim policy for bound PVs is **Retain**, to ensure that data is not lost when an application is removed from the source cluster. You can remove these PVs during rollback.

```
apiVersion: v1
kind: PersistentVolume
metadata:
```

```

name: pv0001
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain 1
  ...
status:
  ...

```

**1** Specify **Recycle** or **Delete**.

6. On the source cluster, switch to your migrated project:

```
$ oc project <project_name>
```

7. Obtain the project's deployed resources:

```
$ oc get all
```

8. Start one or more replicas of each deployed resource:

```
$ oc scale --replicas=1 <resource_type>/<resource_name>
```

9. Update the **nodeSelector** of a DaemonSet to its original value, if you changed it during the procedure.

### 1.8.6. Gathering data for a customer support case

If you open a customer support case, you can run the **must-gather** tool with the **openshift-migration-must-gather-rhel8** image to collect information about your cluster and upload it to the [Red Hat Customer Portal](#).

The **openshift-migration-must-gather-rhel8** image collects logs and Custom Resource data that are not collected by the default **must-gather** image.

#### Procedure

1. Navigate to the directory where you want to store the **must-gather** data.
2. Run the **oc adm must-gather** command:

```
$ oc adm must-gather --image=registry.redhat.io/rhcam-1-2/openshift-migration-must-gather-rhel8
```

The **must-gather** tool collects the cluster information and stores it in a **must-gather.local.<uid>** directory.

3. Remove authentication keys and other sensitive information from the **must-gather** data.
4. Create an archive file containing the contents of the **must-gather.local.<uid>** directory:

```
$ tar cvaf must-gather.tar.gz must-gather.local.<uid>/
```

You can attach the compressed file to your customer support case on the [Red Hat Customer Portal](#).

### 1.8.7. Known issues

This release has the following known issues:

- During migration, the Cluster Application Migration (CAM) tool preserves the following namespace annotations:
  - **openshift.io/sa.scc.mcs**
  - **openshift.io/sa.scc.supplemental-groups**
  - **openshift.io/sa.scc.uid-range**

These annotations preserve the UID range, ensuring that the containers retain their file system permissions on the target cluster. There is a risk that the migrated UIDs could duplicate UIDs within an existing or future namespace on the target cluster. ([BZ#1748440](#))
- If an AWS bucket is added to the CAM web console and then deleted, its status remains **True** because the MigStorage CR is not updated. ([BZ#1738564](#))
- Most cluster-scoped resources are not yet handled by the CAM tool. If your applications require cluster-scoped resources, you may have to create them manually on the target cluster.
- If a migration fails, the migration plan does not retain custom PV settings for quiesced pods. You must manually roll back the migration, delete the migration plan, and create a new migration plan with your PV settings. ([BZ#1784899](#))
- If a large migration fails because Restic times out, you can increase the **restic\_timeout** parameter value (default: **1h**) in the Migration controller CR.
- If you select the data verification option for PVs that are migrated with the filesystem copy method, performance is significantly slower. Velero generates a checksum for each file and checks it when the file is restored.
- In the current release (CAM 1.2), you cannot migrate from OpenShift Container Platform 3.7 to 4.4 because certain API **GroupVersionKinds** (GVKs) that are used by the source cluster are deprecated. You can manually update the GVKs after migration. ([BZ#1817251](#))
- If you cannot install CAM 1.2 on an OpenShift Container Platform 3 cluster, download the current **operator.yml** file, which fixes this problem. ([BZ#1843059](#))

## CHAPTER 2. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 4.1

### 2.1. MIGRATION TOOLS AND PREREQUISITES

You can migrate application workloads from OpenShift Container Platform 4.1 to 4.2 with the Cluster Application Migration (CAM) tool. The CAM tool enables you to control the migration and to minimize application downtime.



#### NOTE

You can migrate between OpenShift Container Platform clusters of the same version, for example, from 4.1 to 4.1, as long as the source and target clusters are configured correctly.

The CAM tool's web console and API, based on Kubernetes Custom Resources, enable you to migrate stateful and stateless application workloads at the granularity of a namespace.

The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.

#### 2.1.1. Migration prerequisites

- You must upgrade the source cluster to the latest z-stream release.
- You must have **cluster-admin** privileges on all clusters.
- The source and target clusters must have unrestricted network access to the replication repository.
- The cluster on which the Migration controller is installed must have unrestricted access to the other clusters.
- If your application uses images from the **openshift** namespace, the required versions of the images must be present on the target cluster.  
If the required images are not present, you must update the **imagestreamtags** references to use an available version that is compatible with your application. If the **imagestreamtags** cannot be updated, you can manually upload equivalent images to the application namespaces and update the applications to reference them.

The following **imagestreamtags** have been *removed* from OpenShift Container Platform 4.2:

- **dotnet:1.0, dotnet:1.1, dotnet:2.0**
- **dotnet-runtime:2.0**
- **mariadb:10.1**
- **mongodb:2.4, mongodb:2.6**

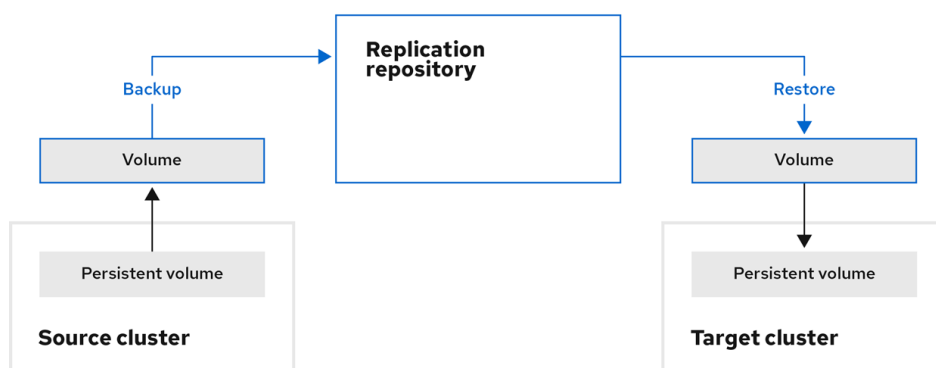
- `mysql:5.5`, `mysql:5.6`
- `nginx:1.8`
- `nodejs:0.10`, `nodejs:4`, `nodejs:6`
- `perl:5.16`, `perl:5.20`
- `php:5.5`, `php:5.6`
- `postgresql:9.2`, `postgresql:9.4`, `postgresql:9.5`
- `python:3.3`, `python:3.4`
- `ruby:2.0`, `ruby:2.2`

## 2.1.2. About the Cluster Application Migration tool

The Cluster Application Migration (CAM) tool enables you to migrate Kubernetes resources, persistent volume data, and internal container images from an OpenShift Container Platform source cluster to an OpenShift Container Platform 4.2 target cluster, using the CAM web console or the Kubernetes API.

Migrating an application with the CAM web console involves the following steps:

1. Install the Cluster Application Migration Operator on all clusters.  
You can install the Cluster Application Migration Operator in a restricted environment with limited or no internet access. The source and target clusters must have network access to each other and to a mirror registry.
2. Configure the replication repository, an intermediate object storage that the CAM tool uses to migrate data.  
The source and target clusters must have network access to the replication repository during migration. In a restricted environment, you can use an internally hosted S3 storage repository. If you use a proxy server, you must ensure that replication repository is whitelisted.
3. Add the source cluster to the CAM web console.
4. Add the replication repository to the CAM web console.
5. Create a migration plan, with one of the following data migration options:
  - **Copy:** The CAM tool copies the data from the source cluster to the replication repository, and from the replication repository to the target cluster.



OpenShift\_45\_1019

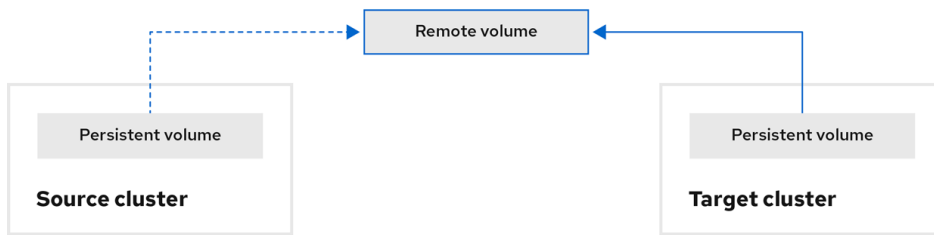
- **Move:** The CAM tool unmounts a remote volume (for example, NFS) from the source

cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.



**NOTE**

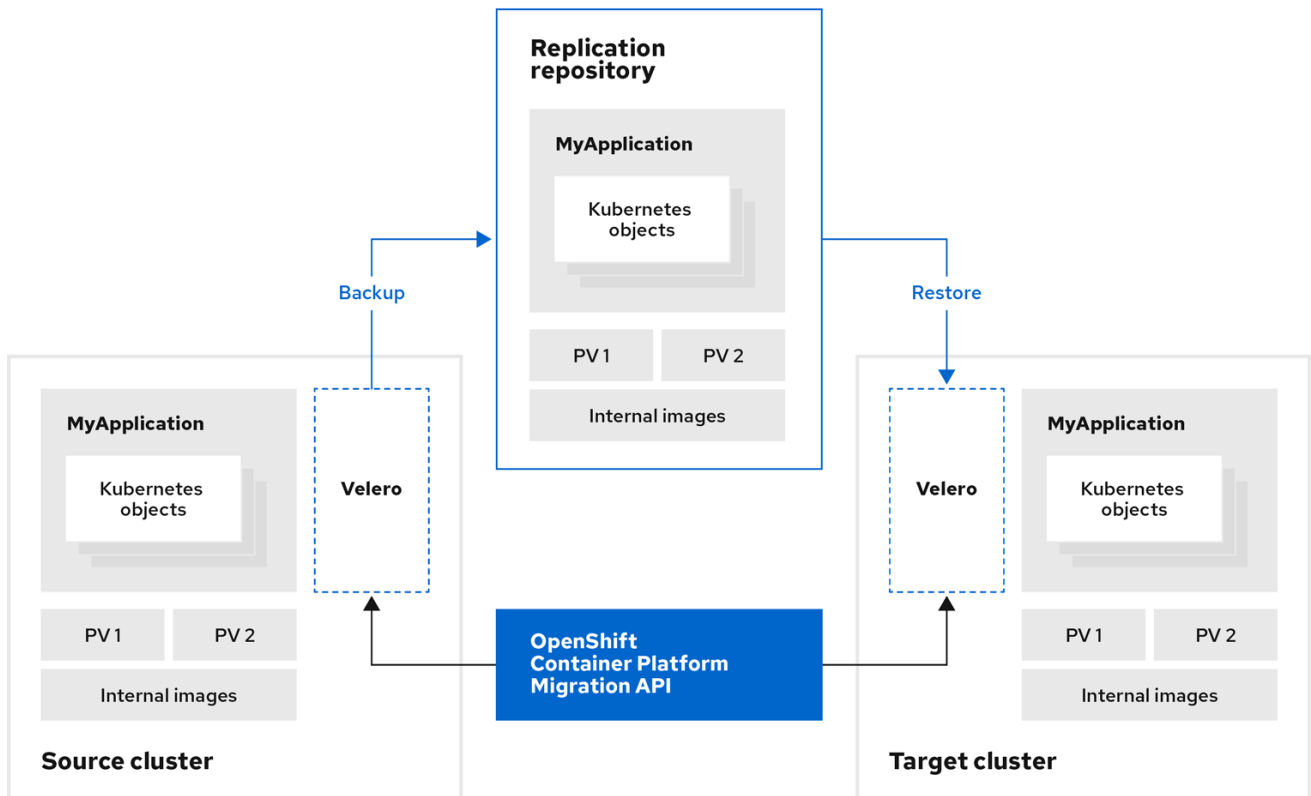
Although the replication repository does not appear in this diagram, it is required for the actual migration.



OpenShift\_45\_1019

6. Run the migration plan, with one of the following options:

- **Stage** (optional) copies data to the target cluster without stopping the application. Staging can be run multiple times so that most of the data is copied to the target before migration. This minimizes the actual migration time and application downtime.
- **Migrate** stops the application on the source cluster and recreates its resources on the target cluster. Optionally, you can migrate the workload without stopping the application.



OpenShift\_45\_1019

### 2.1.3. About data copy methods



The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

### 2.1.3.1. File system copy method

The CAM tool copies data files from the source cluster to the replication repository, and from there to the target cluster.

**Table 2.1. File system copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>• Clusters can have different storage classes</li> <li>• Supported for all S3 storage providers</li> <li>• Optional data verification with checksum</li> </ul>	<ul style="list-style-type: none"> <li>• Slower than the snapshot copy method</li> <li>• Optional data verification significantly reduces performance</li> </ul>

### 2.1.3.2. Snapshot copy method

The CAM tool copies a snapshot of the source cluster's data to a cloud provider's object storage, configured as a replication repository. The data is restored on the target cluster.

AWS, Google Cloud Provider, and Microsoft Azure support the snapshot copy method.

**Table 2.2. Snapshot copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>• Faster than the file system copy method</li> </ul>	<ul style="list-style-type: none"> <li>• Cloud provider must support snapshots.</li> <li>• Clusters must be on the same cloud provider.</li> <li>• Clusters must be in the same location or region.</li> <li>• Clusters must have the same storage class.</li> <li>• Storage class must be compatible with snapshots.</li> </ul>

### 2.1.4. About migration hooks

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.



#### NOTE

If you do not want to use Ansible playbooks, you can create a custom container image and add it to a migration plan.

Migration hooks perform tasks such as customizing application quiescence, manually migrating unsupported data types, and updating applications after migration.

A single migration hook runs on a source or target cluster at one of the following migration steps:

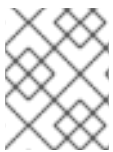
- **PreBackup:** Before backup tasks are started on the source cluster
- **PostBackup:** After backup tasks are complete on the source cluster
- **PreRestore:** Before restore tasks are started on the target cluster
- **PostRestore:** After restore tasks are complete on the target cluster  
You can assign one hook to each migration step, up to a maximum of four hooks for a single migration plan.

The default **hook-runner** image is **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**. This image is based on Ansible Runner and includes **python-openshift** for Ansible Kubernetes resources and an updated **oc** binary. You can also create your own hook image with additional Ansible modules or tools.

The Ansible playbook is mounted on a hook container as a ConfigMap. The hook container runs as a Job on a cluster with a specified service account and namespace. The Job runs, even if the initial Pod is evicted or killed, until it reaches the default **backoffLimit (6)** or successful completion.

## 2.2. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL

You can install the Cluster Application Migration Operator on your OpenShift Container Platform 4.2 target cluster and 4.1 source cluster. The Cluster Application Migration Operator installs the Cluster Application Migration (CAM) tool on the target cluster by default.



### NOTE

Optional: You can configure the Cluster Application Migration Operator to install the CAM tool [on an OpenShift Container Platform 3 cluster or on a remote cluster](#).

In a restricted environment, you can install the Cluster Application Migration Operator from a local mirror registry.

After you have installed the Cluster Application Migration Operator on your clusters, you can launch the CAM tool.

### 2.2.1. Installing the Cluster Application Migration Operator

You can install the Cluster Application Migration Operator with the Operation Lifecycle Manager (OLM) on an OpenShift Container Platform 4.2 target cluster and on an OpenShift Container Platform 4.1 source cluster.

#### 2.2.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

## Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New** → **MigrationController**.
8. Click **Create**.
9. Click **Workloads** → **Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 2.2.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.1 source cluster

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4 source cluster with the Operation Lifecycle Manager (OLM).

## Procedure

1. In the OpenShift Container Platform web console, click **Catalog** → **OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New** → **MigrationController**.
8. Update the **migration\_controller** and **migration\_ui** parameters and add the **deprecated\_cors\_configuration** parameter to the **spec** stanza:

```
spec:
  ...
  migration_controller: false
```

```
migration_ui: false
...
deprecated_cors_configuration: true
```

9. Click **Create**.

10. Click **Workloads** → **Pods** to verify that the Restic and Velero Pods are running.

## 2.2.2. Installing the Cluster Application Migration Operator in a restricted environment

You can build a custom Operator catalog image for OpenShift Container Platform 4, push it to a local mirror image registry, and configure the Operation Lifecycle Manager to install the Cluster Application Migration Operator from the local registry.

### 2.2.2.1. Configuring OperatorHub for restricted networks

Cluster administrators can configure OLM and OperatorHub to use local content in restricted network environments.

#### Prerequisites

- Cluster administrator access to an OpenShift Container Platform cluster and its internal registry.
- Separate workstation without network restrictions.
- If pushing images to the OpenShift Container Platform cluster's internal registry, the registry must be exposed with a route.
- **podman** version 1.4.4+

#### Procedure

##### 1. Disable the default OperatorSources.

Add **disableAllDefaultSources: true** to the spec:

```
$ oc patch OperatorHub cluster --type json \
  -p [{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]
```

This disables the default OperatorSources that are configured by default during an OpenShift Container Platform installation.

##### 2. Retrieve package lists.

To get the list of packages that are available for the default OperatorSources, run the following **curl** commands from your workstation without network restrictions:

```
$ curl https://quay.io/cnr/api/v1/packages?namespace=redhat-operators > packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=community-operators >> packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=certified-operators >> packages.txt
```

Each package in the new **packages.txt** is an Operator that you could add to your restricted network catalog. From this list, you could either pull every Operator or a subset that you would like to expose to users.

### 3. Pull Operator content.

For a given Operator in the package list, you must pull the latest released content:

```
$ curl https://quay.io/cnr/api/v1/packages/<namespace>/<operator_name>/<release>
```

This example uses the etcd Operator:

- a. Retrieve the digest:

```
$ curl https://quay.io/cnr/api/v1/packages/community-operators/etcd/0.0.12
```

- b. From that JSON, take the digest and use it to pull the gzipped archive:

```
$ curl -XGET https://quay.io/cnr/api/v1/packages/community-
operators/etcd/blobs/sha256/8108475ee5e83a0187d6d0a729451ef1ce6d34c44a868a2001
51c36f3232822b \
-o etcd.tar.gz
```

- c. To pull the information out, you must untar the archive into a **manifests/<operator\_name>/** directory with all the other Operators that you want. For example, to untar to an existing directory called **manifests/etcd/**:

```
$ mkdir -p manifests/etcd/ 1
$ tar -xf etcd.tar.gz -C manifests/etcd/
```

- 1 Create different subdirectories for each extracted archive so that files are not overwritten by subsequent extractions for other Operators.

### 4. Break apart **bundle.yaml** content, if necessary.

In your new **manifests/<operator\_name>** directory, the goal is to get your bundle in the following directory structure:

```
manifests/
├── etcd
│   ├── 0.0.12
│   │   ├── clusterserviceversion.yaml
│   │   └── customresourcedefinition.yaml
│   └── package.yaml
```

If you see files already in this structure, you can skip this step. However, if you instead see only a single file called **bundle.yaml**, you must first break this file up to conform to the required structure.

You must separate the CSV content under **data.clusterServiceVersion** (each file in the list), the CRD content under **data.customResourceDefinition** (each file in the list), and the package content under **data.Package** into their own files.

- a. For the CSV file creation, find the following lines in the **bundle.yaml** file:

```
data:
  clusterServiceVersions: |
```

Omit those lines, but save a new file consisting of the full CSV resource content beginning with the following lines, removing the prepended `-` character:

#### Example `clusterserviceversion.yaml` file snippet

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
[...]
```

- b. For the CRD file creation, find the following line in the `bundle.yaml` file:

```
customResourceDefinitions: |
```

Omit this line, but save new files consisting of each, full CRD resource content beginning with the following lines, removing the prepended `-` character:

#### Example `customresourcedefinition.yaml` file snippet

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
[...]
```

- c. For the package file creation, find the following line in the `bundle.yaml` file:

```
packages: |
```

Omit this line, but save a new file consisting of the package content beginning with the following lines, removing the prepended `-` character, and ending with a `packageName` entry:

#### Example `package.yaml` file

```
channels:
- currentCSV: etcdoperator.v0.9.4
  name: singlenamespace-alpha
- currentCSV: etcdoperator.v0.9.4-clusterwide
  name: clusterwide-alpha
defaultChannel: singlenamespace-alpha
packageName: etcd
```

### 5. Identify images required by the Operators you want to use.

Inspect the CSV files of each Operator for `image:` fields to identify the pull specs for any images required by the Operator, and note them for use in a later step.

For example, in the following `deployments` spec of an etcd Operator CSV:

```
spec:
  serviceAccountName: etcd-operator
  containers:
  - name: etcd-operator
    command:
    - etcd-operator
    - --create-crd=false
    image: quay.io/coreos/etcd-
```

```
operator@sha256:bd944a211eaf8f31da5e6d69e8541e7cada8f16a9f7a5a570b224789978199
43 1
  env:
  - name: MY_POD_NAMESPACE
    valueFrom:
      fieldRef:
        fieldPath: metadata.namespace
  - name: MY_POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
```

- 1** Image required by Operator.

## 6. Create an Operator catalog image.

- a. Save the following to a Dockerfile, for example named **custom-registry.Dockerfile**:

```
FROM registry.redhat.io/openshift4/ose-operator-registry:v4.2.24 AS builder

COPY manifests manifests

RUN /bin/initializer -o ./bundles.db

FROM registry.access.redhat.com/ubi7/ubi

COPY --from=builder /registry/bundles.db /bundles.db
COPY --from=builder /usr/bin/registry-server /registry-server
COPY --from=builder /bin/grpc_health_probe /bin/grpc_health_probe

EXPOSE 50051

ENTRYPOINT ["/registry-server"]

CMD ["--database", "bundles.db"]
```

- b. Use the **podman** command to create and tag the container image from the Dockerfile:

```
$ podman build -f custom-registry.Dockerfile \
  -t <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-
  registry 1
```

- 1** Tag the image for the internal registry of the restricted network OpenShift Container Platform cluster and any namespace.

## 7. Push the Operator catalog image to a registry.

Your new Operator catalog image must be pushed to a registry that the restricted network OpenShift Container Platform cluster can access. This can be the internal registry of the cluster itself or another registry that the cluster has network access to, such as an on-premise Quay Enterprise registry.

For this example, login and push the image to the internal registry OpenShift Container Platform cluster:

-

```
$ podman push <local_registry_host_name>:
<local_registry_host_port>/<namespace>/custom-registry
```

## 8. Create a CatalogSource pointing to the new Operator catalog image.

- a. Save the following to a file, for example **my-operator-catalog.yaml**:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: My Operator Catalog
  sourceType: grpc
  image: <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-
registry:latest
```

- b. Create the CatalogSource resource:

```
$ oc create -f my-operator-catalog.yaml
```

- c. Verify the CatalogSource and package manifest are created successfully:

```
# oc get pods -n openshift-marketplace
NAME READY STATUS RESTARTS AGE
my-operator-catalog-6njx6 1/1 Running 0 28s
marketplace-operator-d9f549946-96sgr 1/1 Running 0 26h

# oc get catalogsource -n openshift-marketplace
NAME DISPLAY TYPE PUBLISHER AGE
my-operator-catalog My Operator Catalog grpc 5s

# oc get packagemanifest -n openshift-marketplace
NAME CATALOG AGE
etcd My Operator Catalog 34s
```

You should also be able to view them from the **OperatorHub** page in the web console.

## 9. Mirror the images required by the Operators you want to use.

- a. Determine the images defined by the Operator(s) that you are expecting. This example uses the etcd Operator, requiring the **quay.io/coreos/etcd-operator** image.



### IMPORTANT

This procedure only shows mirroring Operator images themselves and not Operand images, which are the components that an Operator manages. Operand images must be mirrored as well; see each Operator's documentation to identify the required Operand images.

- b. To use mirrored images, you must first create an ImageContentSourcePolicy for each image to change the source location of the Operator catalog image. For example:



```

apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: etcd-operator
spec:
  repositoryDigestMirrors:
  - mirrors:
    - <local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator
    source: quay.io/coreos/etcd-operator

```

- c. Use the **oc image mirror** command from your workstation without network restrictions to pull the image from the source registry and push to the internal registry without being stored locally:

```

$ oc image mirror quay.io/coreos/etcd-operator \
  <local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator

```

You can now install the Operator from the OperatorHub on your restricted network OpenShift Container Platform cluster.

### 2.2.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

#### Prerequisites

- You created a custom Operator catalog and pushed it to a mirror registry.
- You configured OLM to install the Cluster Application Migration Operator from the mirror registry.

#### Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more...**
7. Click **Create New → MigrationController**.

8. Click **Create**.
9. Click **Workloads** → **Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 2.2.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.1 source cluster in a restricted environment

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4 source cluster with the Operation Lifecycle Manager (OLM).

#### Prerequisites

- You created a custom Operator catalog and pushed it to a mirror registry.
- You configured OLM to install the Cluster Application Migration Operator from the mirror registry.

#### Procedure

1. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
2. Select the **Cluster Application Migration Operator** and click **Install**.
3. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
4. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
5. Under **Provided APIs**, click **View 12 more...**
6. Click **Create New** → **MigrationController**.
7. Click **Create**.

### 2.2.3. Launching the CAM web console

You can launch the CAM web console in a browser.

#### Procedure

1. Log in to the OpenShift Container Platform cluster on which you have installed the CAM tool.
2. Obtain the CAM web console URL by entering the following command:

```
$ oc get -n openshift-migration route/migration -o go-template='https://{ .spec.host }'
```

The output resembles the following: **https://migration-openshift-migration.apps.cluster.openshift.com**.

3. Launch a browser and navigate to the CAM web console.

**NOTE**

If you try to access the CAM web console immediately after installing the Cluster Application Migration Operator, the console may not load because the Operator is still configuring the cluster. Wait a few minutes and retry.

4. If you are using self-signed CA certificates, you will be prompted to accept the CA certificate of the source cluster's API server. The web page guides you through the process of accepting the remaining certificates.
5. Log in with your OpenShift Container Platform **username** and **password**.

## 2.3. CONFIGURING A REPLICATION REPOSITORY

You must configure an object storage to use as a replication repository. The Cluster Application Migration tool copies data from the source cluster to the replication repository, and then from the replication repository to the target cluster.

The CAM tool supports the [file system and snapshot data copy methods](#) for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

The following storage providers are supported:

- [Multi-Cloud Object Gateway \(MCG\)](#)
- [Amazon Web Services \(AWS\) S3](#)
- [Google Cloud Provider \(GCP\)](#)
- [Microsoft Azure](#)
- Generic S3 object storage, for example, Minio or Ceph S3

The source and target clusters must have network access to the replication repository during migration.

In a restricted environment, you can create an internally hosted replication repository. If you use a proxy server, you must ensure that your replication repository is whitelisted.

**IMPORTANT**

Configuring Multi-Cloud Object Gateway as a replication repository for migration is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

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

### 2.3.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository

You can install the OpenShift Container Storage Operator and configure a Multi-Cloud Object Gateway (MCG) storage bucket as a replication repository.

### 2.3.1.1. Installing the OpenShift Container Storage Operator

You can install the OpenShift Container Storage Operator from OperatorHub.

#### Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use **Filter by keyword** (in this case, **OCS**) to find the **OpenShift Container Storage Operator**.
3. Select the **OpenShift Container Storage Operator** and click **Install**.
4. Select an **Update Channel**, **Installation Mode**, and **Approval Strategy**.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **OpenShift Container Storage Operator** appears in the **openshift-storage** project with the status **Succeeded**.

### 2.3.1.2. Creating the Multi-Cloud Object Gateway storage bucket

You can create the Multi-Cloud Object Gateway (MCG) storage bucket's Custom Resources (CRs).

#### Procedure

1. Log in to the OpenShift Container Platform cluster:

```
$ oc login
```

2. Create the **NooBaa** CR configuration file, **noobaa.yml**, with the following content:

```
apiVersion: noobaa.io/v1alpha1
kind: NooBaa
metadata:
  name: noobaa
  namespace: openshift-storage
spec:
  dbResources:
    requests:
      cpu: 0.5 1
      memory: 1Gi
  coreResources:
    requests:
      cpu: 0.5 2
      memory: 1Gi
```

**1 2** For a very small cluster, you can change the **cpu** value to **0.1**.

3. Create the **NooBaa** object:

```
$ oc create -f noobaa.yml
```

4. Create the **BackingStore** CR configuration file, **bs.yml**, with the following content:

```

apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
    - noobaa.io/finalizer
  labels:
    app: noobaa
  name: mcg-pv-pool-bs
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 1
    resources:
      requests:
        storage: 50Gi 2
    storageClass: gp2 3
  type: pv-pool

```

- 1** Specify the number of volumes in the PV pool.
- 2** Specify the size of the volumes.
- 3** Specify the storage class.

5. Create the **BackingStore** object:

```
$ oc create -f bs.yml
```

6. Create the **BucketClass** CR configuration file, **bc.yml**, with the following content:

```

apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: mcg-pv-pool-bc
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
      - backingStores:
          - mcg-pv-pool-bs
    placement: Spread

```

7. Create the **BucketClass** object:

```
$ oc create -f bc.yml
```

8. Create the **ObjectBucketClaim** CR configuration file, **obc.yml**, with the following content:

```

apiVersion: objectbucket.io/v1alpha1

```

```

kind: ObjectBucketClaim
metadata:
  name: migstorage
  namespace: openshift-storage
spec:
  bucketName: migstorage ❶
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: mcg-pv-pool-bc

```

- ❶ Record the bucket name for adding the replication repository to the CAM web console.

9. Create the **ObjectBucketClaim** object:

```
$ oc create -f obc.yml
```

10. Watch the resource creation process to verify that the **ObjectBucketClaim** status is **Bound**:

```
$ watch -n 30 'oc get -n openshift-storage objectbucketclaim migstorage -o yaml'
```

This process can take five to ten minutes.

11. Obtain and record the following values, which are required when you add the replication repository to the CAM web console:

- S3 endpoint:

```
$ oc get route -n openshift-storage s3
```

- S3 provider access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_ACCESS_KEY_ID }}' | base64 -d
```

- S3 provider secret access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_SECRET_ACCESS_KEY }}' | base64 -d
```

### 2.3.2. Configuring an AWS S3 storage bucket as a replication repository

You can configure an AWS S3 storage bucket as a replication repository.

#### Prerequisites

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

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

## Procedure

1. Create an AWS S3 bucket:

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

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

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

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

3. Create an EC2 EBS snapshot policy:

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

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

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

```

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

```

- 1 2 To grant access to a single S3 bucket, specify the bucket name. To grant access to all AWS S3 buckets, specify \* instead of a bucket name:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```



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

- 1 2 Record the **AWS\_SECRET\_ACCESS\_KEY** and the **AWS\_ACCESS\_KEY\_ID** for adding the AWS repository to the CAM web console.

### 2.3.3. Configuring a Google Cloud Provider storage bucket as a replication repository

You can configure a Google Cloud Provider (GCP) storage bucket as a replication repository.

#### Prerequisites

- The GCP storage bucket must be accessible to the source and target clusters.
- You must have [gsutil](#) installed.
- If you are using the snapshot copy method:
  - The source and target clusters must be in the same region.
  - The source and target clusters must have the same storage class.
  - The storage class must be compatible with snapshots.

#### Procedure

1. Run **gsutil init** to log in:

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

Your current configuration has been set to: [default]

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

2. Set the **BUCKET** variable:

```
$ BUCKET=<bucket_name> 1
```

- 1 Specify your bucket name.

3. Create a storage bucket:

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

4. Set the **PROJECT\_ID** variable to your active project:

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

5. Create a **velero** service account:

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

- Set the **SERVICE\_ACCOUNT\_EMAIL** variable to the service account's email address:

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

- Grant permissions to the service account:

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

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

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

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

- Save the service account's keys to the **credentials-velero** file in the current directory:

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

### 2.3.4. Configuring a Microsoft Azure Blob storage container as a replication repository

You can configure a Microsoft Azure Blob storage container as a replication repository.

#### Prerequisites

- You must have an [Azure storage account](#).
- You must have the [Azure CLI](#) installed.
- The Azure Blob storage container must be accessible to the source and target clusters.
- If you are using the snapshot copy method:
  - The source and target clusters must be in the same region.

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

## Procedure

1. Set the **AZURE\_RESOURCE\_GROUP** variable:

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. Create an Azure resource group:

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

- 1 Specify your location.

3. Set the **AZURE\_STORAGE\_ACCOUNT\_ID** variable:

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. Create an Azure storage account:

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

5. Set the **BLOB\_CONTAINER** variable:

```
$ BLOB_CONTAINER=velero
```

6. Create an Azure Blob storage container:

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

7. Create a service principal and credentials for **velero**:

```
$ AZURE_SUBSCRIPTION_ID=`az account list --query '[?isDefault].id' -o tsv`
$ AZURE_TENANT_ID=`az account list --query '[?isDefault].tenantId' -o tsv`
$ AZURE_CLIENT_SECRET=`az ad sp create-for-rbac --name "velero" --role "Contributor" --
query 'password' -o tsv`
$ AZURE_CLIENT_ID=`az ad sp list --display-name "velero" --query '[0].appId' -o tsv`
```

8. Save the service principal's credentials in the **credentials-velero** file:

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

## 2.4. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE

You can migrate application workloads by adding your clusters and replication repository to the CAM web console. Then, you can create and run a migration plan.

If your cluster or replication repository are secured with self-signed certificates, you can create a CA certificate bundle file or disable SSL verification.

### 2.4.1. Creating a CA certificate bundle file

If you use a self-signed certificate to secure a cluster or a replication repository, certificate verification might fail with the following error message: **Certificate signed by unknown authority**.

You can create a custom CA certificate bundle file and upload it in the CAM web console when you add a cluster or a replication repository.

#### Procedure

Download a CA certificate from a remote endpoint and save it as a CA bundle file:

```
$ echo -n | openssl s_client -connect <host_FQDN>:<port> \ 1
| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > <ca_bundle.cert> 2
```

- 1 Specify the host FQDN and port of the endpoint, for example, **api.my-cluster.example.com:6443**.
- 2 Specify the name of the CA bundle file.

### 2.4.2. Adding a cluster to the CAM web console

You can add a cluster to the CAM web console.

#### Prerequisites

If you are using Azure snapshots to copy data:

- You must provide the Azure resource group name when you add the source cluster.
- The source and target clusters must be in the same Azure resource group and in the same location.

#### Procedure

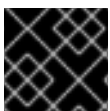
1. Log in to the cluster.
2. Obtain the service account token:



- **Replication repository name:** Specify the replication repository name in the CAM web console.
  - **S3 bucket name:** Specify the name of the S3 bucket you created.
  - **S3 bucket region:** Specify the S3 bucket region. **Required** for AWS S3. **Optional** for other S3 providers.
  - **S3 endpoint:** Specify the URL of the S3 service, not the bucket, for example, **https://<s3-storage.apps.cluster.com>**. **Required** for a generic S3 provider. You must use the **https://** prefix.
  - **S3 provider access key:** Specify the **<AWS\_SECRET\_ACCESS\_KEY>** for AWS or the S3 provider access key for MCG.
  - **S3 provider secret access key:** Specify the **<AWS\_ACCESS\_KEY\_ID>** for AWS or the S3 provider secret access key for MCG.
  - **Require SSL verification:** Clear this check box if you are using a generic S3 provider.
  - If you use a custom CA bundle, click **Browse** and browse to the Base64-encoded CA bundle file.
- **GCP:**
    - **Replication repository name:** Specify the replication repository name in the CAM web console.
    - **GCP bucket name:** Specify the name of the GCP bucket.
    - **GCP credential JSON blob:** Specify the string in the **credentials-velero** file.
  - **Azure:**
    - **Replication repository name:** Specify the replication repository name in the CAM web console.
    - **Azure resource group:** Specify the resource group of the Azure Blob storage.
    - **Azure storage account name:** Specify the Azure Blob storage account name.
    - **Azure credentials - INI file contents:** Specify the string in the **credentials-velero** file.
4. Click **Add repository** and wait for connection validation.
  5. Click **Close**.  
The new repository appears in the **Replication repositories** section.

#### 2.4.4. Changing migration plan limits for large migrations

You can change the migration plan limits for large migrations.



#### IMPORTANT

Changes should first be tested in your environment to avoid a failed migration.

A single migration plan has the following default limits:

- 10 namespaces  
If this limit is exceeded, the CAM web console displays a **Namespace limit exceeded** error and you cannot create a migration plan.
- 100 Pods  
If the Pod limit is exceeded, the CAM web console displays a warning message similar to the following example: **Plan has been validated with warning condition(s). See warning message. Pod limit: 100 exceeded, found: 104.**
- 100 persistent volumes  
If the persistent volume limit is exceeded, the CAM web console displays a similar warning message.

## Procedure

1. Edit the Migration controller CR:

```
$ oc get migrationcontroller -n openshift-migration
NAME AGE
migration-controller 5d19h

$ oc edit migrationcontroller -n openshift-migration
```

2. Update the following parameters:

```
...
migration_controller: true

# This configuration is loaded into mig-controller, and should be set on the
# cluster where `migration_controller: true`
mig_pv_limit: 100
mig_pod_limit: 100
mig_namespace_limit: 10
...
```


### 2.4.5. Creating a migration plan in the CAM web console

You can create a migration plan in the CAM web console.

#### Prerequisites

- The CAM web console must contain the following:
  - Source cluster
  - Target cluster, which is added automatically during the CAM tool installation
  - Replication repository
- The source and target clusters must have network access to each other and to the replication repository.
- If you use snapshots to copy data, the source and target clusters must run on the same cloud provider (AWS, GCP, or Azure) and in the same region.

## Procedure

1. Log in to the CAM web console.
  2. In the **Plans** section, click **Add plan**.
  3. Enter the **Plan name** and click **Next**.  
The **Plan name** can contain up to 253 lower-case alphanumeric characters (**a-z, 0-9**). It must not contain spaces or underscores (**\_**).
  4. Select a **Source cluster**.
  5. Select a **Target cluster**.
  6. Select a **Replication repository**.
  7. Select the projects to be migrated and click **Next**.
  8. Select **Copy** or **Move** for the PVs:
    - **Copy** copies the data in a source cluster's PV to the replication repository and then restores it on a newly created PV, with similar characteristics, in the target cluster.  
Optional: You can verify data copied with the filesystem method by selecting **Verify copy**. This option, which generates a checksum for each source file and checks it after restoration, significantly reduces performance.
    - **Move** unmounts a remote volume (for example, NFS) from the source cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.
  9. Click **Next**.
  10. Select a **Copy method** for the PVs:
    - **Snapshot** backs up and restores the disk using the cloud provider's snapshot functionality. It is significantly faster than **Filesystem**.
- 

**NOTE**

The storage and clusters must be in the same region and the storage class must be compatible.
- **Filesystem** copies the data files from the source disk to a newly created target disk.
  11. Select a **Storage class** for the PVs.  
If you selected the **Filesystem** copy method, you can change the storage class during migration, for example, from Red Hat Gluster Storage or NFS storage to Red Hat Ceph Storage.
  12. Click **Next**.
  13. If you want to add a migration hook, click **Add Hook** and perform the following steps:
    - a. Specify the name of the hook.
    - b. Select **Available playbooks** to use your own playbook or **Custom container image** for a hook.



- d. Select **Ansible playbook** to use your own playbook or **Custom container image** for a hook written in another language.
  - c. Click **Browse** to upload the playbook.
  - d. Optional: If you are not using the default Ansible runtime image, specify your custom Ansible image.
  - e. Specify the cluster on which you want the hook to run.
  - f. Specify the service account name.
  - g. Specify the namespace.
  - h. Select the migration step at which you want the hook to run:
    - PreBackup: Before backup tasks are started on the source cluster
    - PostBackup: After backup tasks are complete on the source cluster
    - PreRestore: Before restore tasks are started on the target cluster
    - PostRestore: After restore tasks are complete on the target cluster
14. Click **Add**.  
You can add up to four hooks to a migration plan, assigning each hook to a different migration step.
15. Click **Finish**.
16. Click **Close**.  
The migration plan appears in the **Plans** section.

### 2.4.6. Running a migration plan in the CAM web console

You can stage or migrate applications and data with the migration plan you created in the CAM web console.

#### Prerequisites

The CAM web console must contain the following:


- Source cluster
- Target cluster, which is added automatically during the CAM tool installation
- Replication repository
- Valid migration plan

#### Procedure

1. Log in to the CAM web console on the target cluster.
2. Select a migration plan.
3. Click **Stage** to copy data from the source cluster to the target cluster without stopping the application.

You can run **Stage** multiple times to reduce the actual migration time.

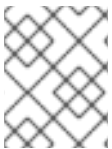
4. When you are ready to migrate the application workload, click **Migrate**. **Migrate** stops the application workload on the source cluster and recreates its resources on the target cluster.
5. Optional: In the **Migrate** window, you can select **Do not stop applications on the source cluster during migration**.
6. Click **Migrate**.

7. Optional: To stop a migration in progress, click the Options menu  and select **Cancel**.
8. When the migration is complete, verify that the application migrated successfully in the OpenShift Container Platform web console:
  - a. Click **Home** → **Projects**.
  - b. Click the migrated project to view its status.
  - c. In the **Routes** section, click **Location** to verify that the application is functioning, if applicable.
  - d. Click **Workloads** → **Pods** to verify that the Pods are running in the migrated namespace.
  - e. Click **Storage** → **Persistent volumes** to verify that the migrated persistent volume is correctly provisioned.

## 2.5. TROUBLESHOOTING

You can view the migration Custom Resources (CRs) and download logs to troubleshoot a failed migration.

If the application was stopped during the failed migration, you must roll it back manually in order to prevent data corruption.

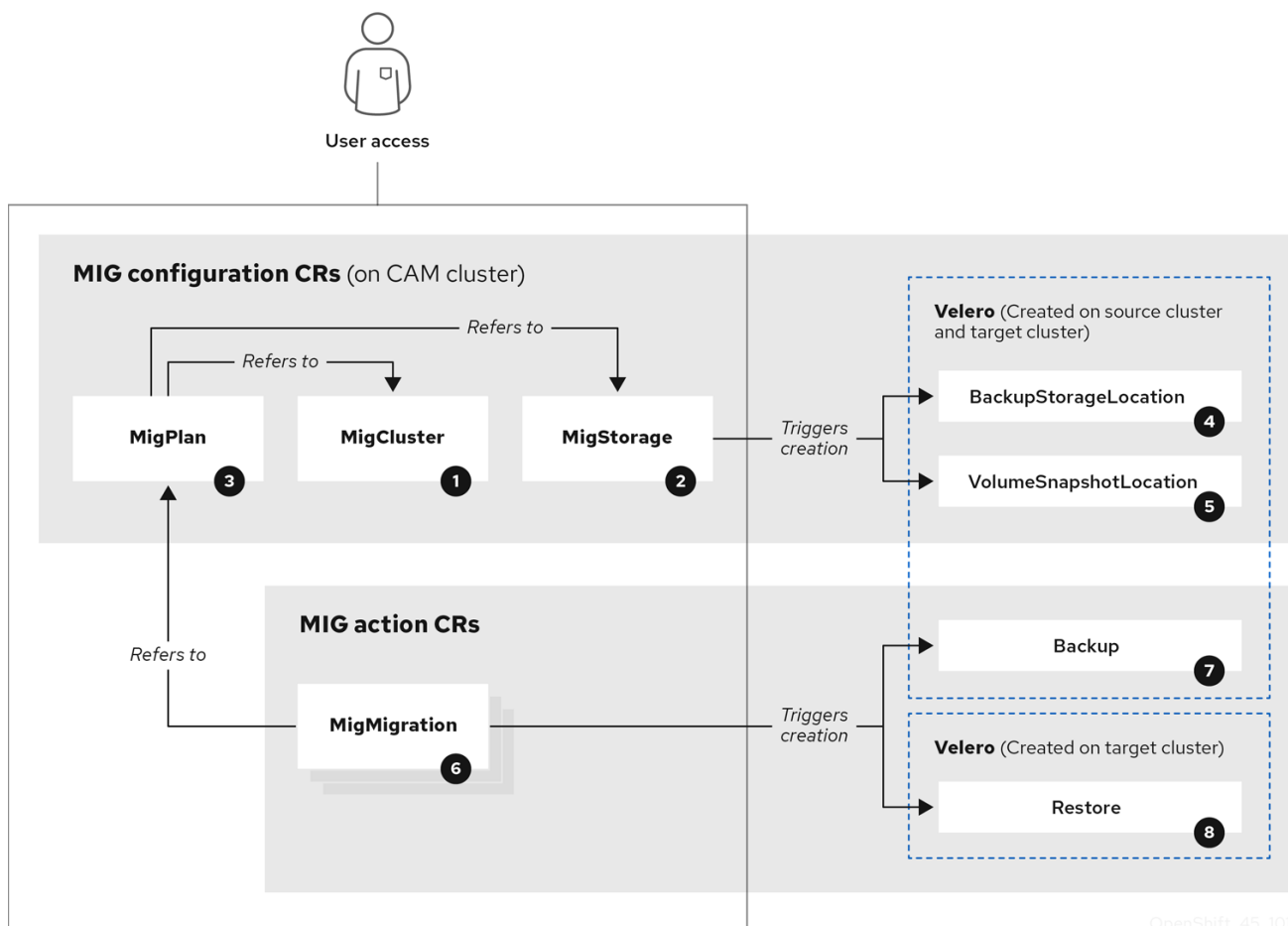


### NOTE

Manual rollback is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

### 2.5.1. Viewing migration Custom Resources

The Cluster Application Migration (CAM) tool creates the following Custom Resources (CRs):



- 1 **MigCluster** (configuration, CAM cluster): Cluster definition
- 2 **MigStorage** (configuration, CAM cluster): Storage definition
- 3 **MigPlan** (configuration, CAM cluster): Migration plan

The MigPlan CR describes the source and target clusters, repository, and namespace(s) being migrated. It is associated with 0, 1, or many MigMigration CRs.



#### NOTE

Deleting a MigPlan CR deletes the associated MigMigration CRs.

- 4 **BackupStorageLocation** (configuration, CAM cluster): Location of Velero backup objects
- 5 **VolumeSnapshotLocation** (configuration, CAM cluster): Location of Velero volume snapshots
- 6 **MigMigration** (action, CAM cluster): Migration, created during migration

A MigMigration CR is created every time you stage or migrate data. Each MigMigration CR is associated with a MigPlan CR.

**7** **Backup** (action, source cluster): When you run a migration plan, the MigMigration CR creates two Velero backup CRs on each source cluster:

- Backup CR #1 for Kubernetes objects
- Backup CR #2 for PV data

**8** **Restore** (action, target cluster): When you run a migration plan, the MigMigration CR creates two Velero restore CRs on the target cluster:

- Restore CR #1 (using Backup CR #2) for PV data
- Restore CR #2 (using Backup CR #1) for Kubernetes objects

## Procedure

1. Get the CR name:

```
$ oc get <migration_cr> -n openshift-migration 1
```

- 1** Specify the migration CR, for example, **migmigration**.

The output is similar to the following:

```
NAME                                AGE
88435fe0-c9f8-11e9-85e6-5d593ce65e10 6m42s
```

2. View the CR:

```
$ oc describe <migration_cr> <88435fe0-c9f8-11e9-85e6-5d593ce65e10> -n openshift-migration
```

The output is similar to the following examples.

## MigMigration example

```
name:      88435fe0-c9f8-11e9-85e6-5d593ce65e10
namespace: openshift-migration
labels:    <none>
annotations: touch: 3b48b543-b53e-4e44-9d34-33563f0f8147
apiVersion: migration.openshift.io/v1alpha1
kind:      MigMigration
metadata:
  creationTimestamp: 2019-08-29T01:01:29Z
  generation:       20
  resourceVersion:  88179
  selfLink:         /apis/migration.openshift.io/v1alpha1/namespaces/openshift-
migration/migmigrations/88435fe0-c9f8-11e9-85e6-5d593ce65e10
  uid:              8886de4c-c9f8-11e9-95ad-0205fe66cbb6
spec:
  migPlanRef:
    name:      socks-shop-mig-plan
    namespace: openshift-migration
```

```

quiescePods: true
stage:      false
status:
conditions:
  category:      Advisory
  durable:       True
  lastTransitionTime: 2019-08-29T01:03:40Z
  message:       The migration has completed successfully.
  reason:        Completed
  status:        True
  type:          Succeeded
phase:        Completed
startTimestamp: 2019-08-29T01:01:29Z
events:       <none>

```

## Velero backup CR #2 example (PV data)

```

apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.105.179:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-44dd3bd5-c9f8-11e9-95ad-0205fe66cbb6
  creationTimestamp: "2019-08-29T01:03:15Z"
  generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
  generation: 1
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    velero.io/storage-location: myrepo-vpzq9
  name: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
  namespace: openshift-migration
  resourceVersion: "87313"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/backups/88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
  uid: c80dbbc0-c9f8-11e9-95ad-0205fe66cbb6
spec:
  excludedNamespaces: []
  excludedResources: []
  hooks:
    resources: []
  includeClusterResources: null
  includedNamespaces:
  - sock-shop
  includedResources:
  - persistentvolumes
  - persistentvolumeclaims
  - namespaces
  - imagestreams
  - imagestreamtags
  - secrets
  - configmaps

```

```

- pods
labelSelector:
  matchLabels:
    migration-included-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
storageLocation: myrepo-vpzq9
ttl: 720h0m0s
volumeSnapshotLocations:
- myrepo-wv6fx
status:
  completionTimestamp: "2019-08-29T01:02:36Z"
  errors: 0
  expiration: "2019-09-28T01:02:35Z"
  phase: Completed
  startTimestamp: "2019-08-29T01:02:35Z"
  validationErrors: null
  version: 1
  volumeSnapshotsAttempted: 0
  volumeSnapshotsCompleted: 0
  warnings: 0

```

## Velero restore CR #2 example (Kubernetes resources)

```

apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.90.187:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-36f54ca7-c925-11e9-825a-06fa9fb68c88
  creationTimestamp: "2019-08-28T00:09:49Z"
  generateName: e13a1b60-c927-11e9-9555-d129df7f3b96-
  generation: 3
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: e18252c9-c927-11e9-825a-06fa9fb68c88
    migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
  name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  namespace: openshift-migration
  resourceVersion: "82329"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources:
    - nodes
    - events
    - events.events.k8s.io
    - backups.velero.io
    - restores.velero.io
    - resticrepositories.velero.io
  includedNamespaces: null
  includedResources: null

```

```


namespaceMapping: null
restorePVs: true
status:
  errors: 0
  failureReason: ""
  phase: Completed
  validationErrors: null
  warnings: 15

```

## 2.5.2. Downloading migration logs

You can download the Velero, Restic, and Migration controller logs in the CAM web console to troubleshoot a failed migration.

### Procedure

1. Log in to the CAM console.
2. Click **Plans** to view the list of migration plans.
3. Click the **Options** menu  of a specific migration plan and select **Logs**.
4. Click **Download Logs** to download the logs of the Migration controller, Velero, and Restic for all clusters.
5. To download a specific log:
  - a. Specify the log options:
    - **Cluster:** Select the source, target, or CAM host cluster.
    - **Log source:** Select **Velero**, **Restic**, or **Controller**.
    - **Pod source:** Select the Pod name, for example, **controller-manager-78c469849c-v6wcf**  
The selected log is displayed.

You can clear the log selection settings by changing your selection.

- b. Click **Download Selected** to download the selected log.

Optionally, you can access the logs by using the CLI, as in the following example:

```

$ oc get pods -n openshift-migration | grep controller
controller-manager-78c469849c-v6wcf      1/1   Running   0      4h49m

$ oc logs controller-manager-78c469849c-v6wcf -f -n openshift-migration

```

## 2.5.3. Error messages

### 2.5.3.1. Restic timeout error message in the Velero Pod log

If a migration fails because Restic times out, the following error appears in the Velero Pod log:

```
level=error msg="Error backing up item" backup=velero/monitoring error="timed out waiting for all
PodVolumeBackups to complete"
error.file="/go/src/github.com/heptio/velero/pkg/restic/backupper.go:165"
error.function="github.com/heptio/velero/pkg/restic.(*backupper).BackupPodVolumes" group=v1
```

The default value of **restic\_timeout** is one hour. You can increase this parameter for large migrations, keeping in mind that a higher value may delay the return of error messages.

### Procedure

1. In the OpenShift Container Platform web console, navigate to **Operators** → **Installed Operators**.
2. Click **Cluster Application Migration Operator**.
3. In the **MigrationController** tab, click **migration-controller**.
4. In the **YAML** tab, update the following parameter value:

```
spec:
  restic_timeout: 1h 1
```

- 1** Valid units are **h** (hours), **m** (minutes), and **s** (seconds), for example, **3h30m15s**.

5. Click **Save**.

### 2.5.3.2. ResticVerifyErrors in the MigMigration Custom Resource

If data verification fails when migrating a PV with the filesystem data copy method, the following error appears in the MigMigration Custom Resource (CR):

```
status:
  conditions:
  - category: Warn
    durable: true
    lastTransitionTime: 2020-04-16T20:35:16Z
    message: There were verify errors found in 1 Restic volume restores. See restore `<registry-
example-migration-rvwcm>`
    for details 1
    status: "True"
    type: ResticVerifyErrors 2
```

- 1** The error message identifies the Restore CR name.

- 2** **ResticErrors** also appears. **ResticErrors** is a general error warning that includes verification errors.



#### NOTE

A data verification error does not cause the migration process to fail.

You can check the target cluster's Restore CR to identify the source of the data verification error.



## Procedure

1. Log in to the target cluster.
2. View the Restore CR:

```
$ oc describe <registry-example-migration-rwcm> -n openshift-migration
```

The output identifies the PV with **PodVolumeRestore** errors:

```
status:
  phase: Completed
  podVolumeRestoreErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rwcm-98t49>
    namespace: openshift-migration
  podVolumeRestoreResticErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rwcm-98t49>
    namespace: openshift-migration
```

3. View the **PodVolumeRestore** CR:

```
$ oc describe <migration-example-rwcm-98t49>
```

The output identifies the Restic Pod that logged the errors:

```
completionTimestamp: 2020-05-01T20:49:12Z
errors: 1
resticErrors: 1
...
resticPod: <restic-nr2v5>
```

4. View the Restic Pod log:

```
$ oc logs -f restic-nr2v5
```

### 2.5.4. Manually rolling back a migration

If your application was stopped during a failed migration, you must roll it back manually in order to prevent data corruption in the PV.

This procedure is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

## Procedure

1. On the target cluster, switch to the migrated project:

```
$ oc project <project>
```

2. Get the deployed resources:

```
$ oc get all
```

3. Delete the deployed resources to ensure that the application is not running on the target cluster and accessing data on the PVC:

```
$ oc delete <resource_type>
```

4. To stop a DaemonSet without deleting it, update the **nodeSelector** in the YAML file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: hello-daemonset
spec:
  selector:
    matchLabels:
      name: hello-daemonset
  template:
    metadata:
      labels:
        name: hello-daemonset
    spec:
      nodeSelector:
        role: worker 1
```

- 1 Specify a **nodeSelector** value that does not exist on any node.

5. Update each PV's reclaim policy so that unnecessary data is removed. During migration, the reclaim policy for bound PVs is **Retain**, to ensure that data is not lost when an application is removed from the source cluster. You can remove these PVs during rollback.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain 1
  ...
status:
  ...
```

- 1 Specify **Recycle** or **Delete**.

6. On the source cluster, switch to your migrated project:

```
$ oc project <project_name>
```

7. Obtain the project's deployed resources:

```
$ oc get all
```

- Start one or more replicas of each deployed resource:

```
$ oc scale --replicas=1 <resource_type>/<resource_name>
```

- Update the **nodeSelector** of a DaemonSet to its original value, if you changed it during the procedure.

### 2.5.5. Gathering data for a customer support case

If you open a customer support case, you can run the **must-gather** tool with the **openshift-migration-must-gather-rhel8** image to collect information about your cluster and upload it to the [Red Hat Customer Portal](#).

The **openshift-migration-must-gather-rhel8** image collects logs and Custom Resource data that are not collected by the default **must-gather** image.

#### Procedure

- Navigate to the directory where you want to store the **must-gather** data.
- Run the **oc adm must-gather** command:

```
$ oc adm must-gather --image=registry.redhat.io/rhcam-1-2/openshift-migration-must-gather-rhel8
```

The **must-gather** tool collects the cluster information and stores it in a **must-gather.local.<uid>** directory.

- Remove authentication keys and other sensitive information from the **must-gather** data.
- Create an archive file containing the contents of the **must-gather.local.<uid>** directory:

```
$ tar cvaf must-gather.tar.gz must-gather.local.<uid>/
```

You can attach the compressed file to your customer support case on the [Red Hat Customer Portal](#).

### 2.5.6. Known issues

This release has the following known issues:

- During migration, the Cluster Application Migration (CAM) tool preserves the following namespace annotations:
  - openshift.io/sa.scc.mcs**
  - openshift.io/sa.scc.supplemental-groups**
  - openshift.io/sa.scc.uid-range**  
These annotations preserve the UID range, ensuring that the containers retain their file system permissions on the target cluster. There is a risk that the migrated UIDs could duplicate UIDs within an existing or future namespace on the target cluster. ([BZ#1748440](#))

- If an AWS bucket is added to the CAM web console and then deleted, its status remains **True** because the MigStorage CR is not updated. ([BZ#1738564](#))
- Most cluster-scoped resources are not yet handled by the CAM tool. If your applications require cluster-scoped resources, you may have to create them manually on the target cluster.
- If a migration fails, the migration plan does not retain custom PV settings for quiesced pods. You must manually roll back the migration, delete the migration plan, and create a new migration plan with your PV settings. ([BZ#1784899](#))
- If a large migration fails because Restic times out, you can increase the **restic\_timeout** parameter value (default: **1h**) in the Migration controller CR.
- If you select the data verification option for PVs that are migrated with the filesystem copy method, performance is significantly slower. Velero generates a checksum for each file and checks it when the file is restored.

## CHAPTER 3. MIGRATING FROM OPENSIFT CONTAINER PLATFORM 4.2

### 3.1. MIGRATION TOOLS AND PREREQUISITES

You can migrate application workloads from OpenShift Container Platform 4.2 to 4.2 with the Cluster Application Migration (CAM) tool. The CAM tool enables you to control the migration and to minimize application downtime.



#### NOTE

You can migrate between OpenShift Container Platform clusters of the same version, for example, from 4.2 to 4.2, as long as the source and target clusters are configured correctly.

The CAM tool's web console and API, based on Kubernetes Custom Resources, enable you to migrate stateful and stateless application workloads at the granularity of a namespace.

The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.

#### 3.1.1. Migration prerequisites

- You must upgrade the source cluster to the latest z-stream release.
- You must have **cluster-admin** privileges on all clusters.
- The source and target clusters must have unrestricted network access to the replication repository.
- The cluster on which the Migration controller is installed must have unrestricted access to the other clusters.
- If your application uses images from the **openshift** namespace, the required versions of the images must be present on the target cluster.  
If the required images are not present, you must update the **imagestreamtags** references to use an available version that is compatible with your application. If the **imagestreamtags** cannot be updated, you can manually upload equivalent images to the application namespaces and update the applications to reference them.

The following **imagestreamtags** have been *removed* from OpenShift Container Platform 4.2:

- **dotnet:1.0, dotnet:1.1, dotnet:2.0**
- **dotnet-runtime:2.0**
- **mariadb:10.1**
- **mongodb:2.4, mongodb:2.6**

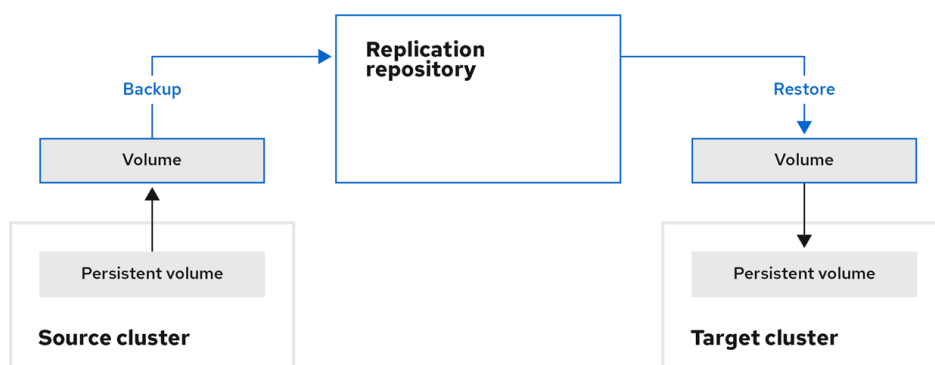
- **mysql:5.5, mysql:5.6**
- **nginx:1.8**
- **nodejs:0.10, nodejs:4, nodejs:6**
- **perl:5.16, perl:5.20**
- **php:5.5, php:5.6**
- **postgresql:9.2, postgresql:9.4, postgresql:9.5**
- **python:3.3, python:3.4**
- **ruby:2.0, ruby:2.2**

### 3.1.2. About the Cluster Application Migration tool

The Cluster Application Migration (CAM) tool enables you to migrate Kubernetes resources, persistent volume data, and internal container images from an OpenShift Container Platform source cluster to an OpenShift Container Platform 4.2 target cluster, using the CAM web console or the Kubernetes API.

Migrating an application with the CAM web console involves the following steps:

1. Install the Cluster Application Migration Operator on all clusters.  
You can install the Cluster Application Migration Operator in a restricted environment with limited or no internet access. The source and target clusters must have network access to each other and to a mirror registry.
2. Configure the replication repository, an intermediate object storage that the CAM tool uses to migrate data.  
The source and target clusters must have network access to the replication repository during migration. In a restricted environment, you can use an internally hosted S3 storage repository. If you use a proxy server, you must ensure that replication repository is whitelisted.
3. Add the source cluster to the CAM web console.
4. Add the replication repository to the CAM web console.
5. Create a migration plan, with one of the following data migration options:
  - **Copy:** The CAM tool copies the data from the source cluster to the replication repository, and from the replication repository to the target cluster.



OpenShift\_45\_1019

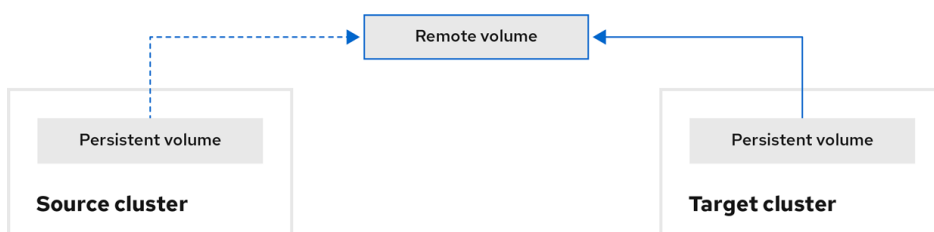
- **Move:** The CAM tool unmounts a remote volume (for example, NFS) from the source

cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.



## NOTE

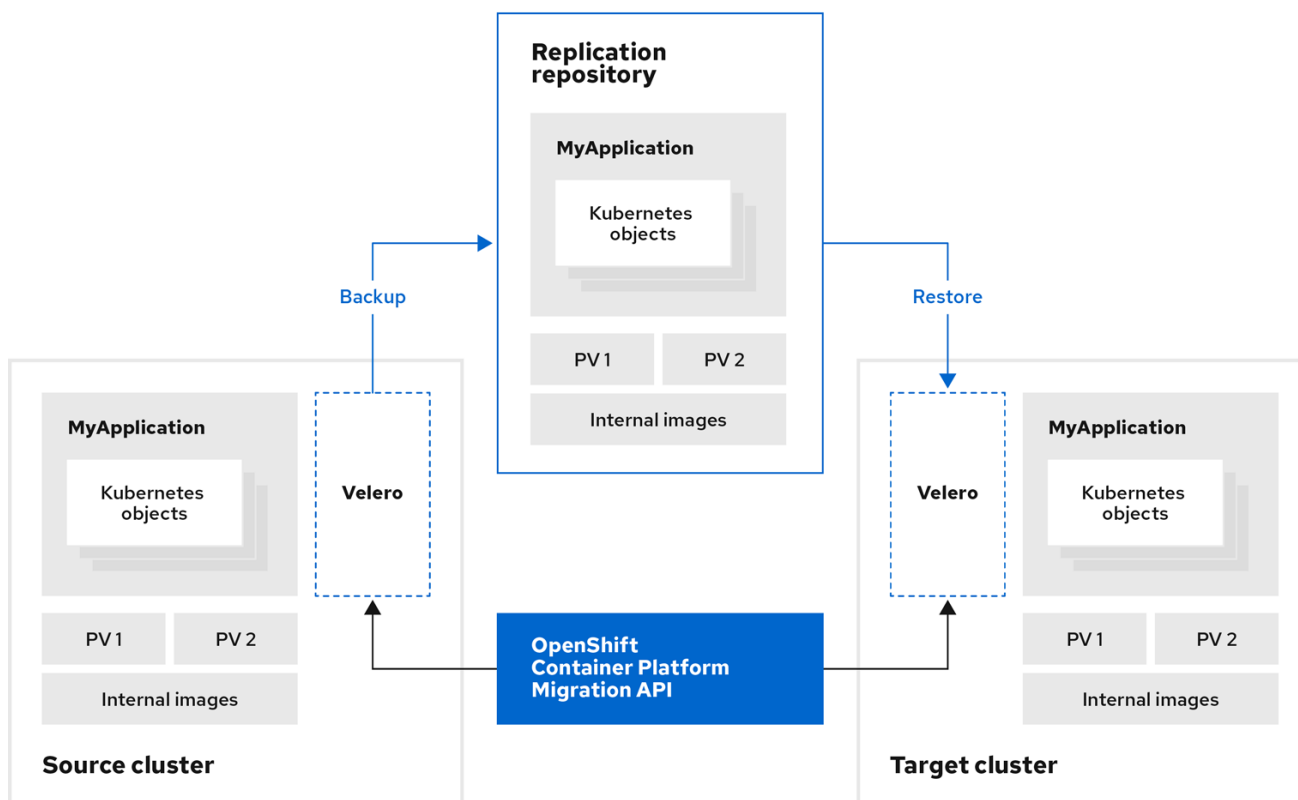
Although the replication repository does not appear in this diagram, it is required for the actual migration.



OpenShift\_45\_1019

6. Run the migration plan, with one of the following options:

- **Stage** (optional) copies data to the target cluster without stopping the application. Staging can be run multiple times so that most of the data is copied to the target before migration. This minimizes the actual migration time and application downtime.
- **Migrate** stops the application on the source cluster and recreates its resources on the target cluster. Optionally, you can migrate the workload without stopping the application.



OpenShift\_45\_1019

### 3.1.3. About data copy methods

The CAM tool supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

### 3.1.3.1. File system copy method

The CAM tool copies data files from the source cluster to the replication repository, and from there to the target cluster.

**Table 3.1. File system copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>• Clusters can have different storage classes</li> <li>• Supported for all S3 storage providers</li> <li>• Optional data verification with checksum</li> </ul>	<ul style="list-style-type: none"> <li>• Slower than the snapshot copy method</li> <li>• Optional data verification significantly reduces performance</li> </ul>

### 3.1.3.2. Snapshot copy method

The CAM tool copies a snapshot of the source cluster's data to a cloud provider's object storage, configured as a replication repository. The data is restored on the target cluster.

AWS, Google Cloud Provider, and Microsoft Azure support the snapshot copy method.

**Table 3.2. Snapshot copy method summary**

Benefits	Limitations
<ul style="list-style-type: none"> <li>• Faster than the file system copy method</li> </ul>	<ul style="list-style-type: none"> <li>• Cloud provider must support snapshots.</li> <li>• Clusters must be on the same cloud provider.</li> <li>• Clusters must be in the same location or region.</li> <li>• Clusters must have the same storage class.</li> <li>• Storage class must be compatible with snapshots.</li> </ul>

### 3.1.4. About migration hooks

You can use migration hooks to run Ansible playbooks at certain points during the migration. The hooks are added when you create a migration plan.



#### NOTE

If you do not want to use Ansible playbooks, you can create a custom container image and add it to a migration plan.



Migration hooks perform tasks such as customizing application quiescence, manually migrating unsupported data types, and updating applications after migration.

A single migration hook runs on a source or target cluster at one of the following migration steps:

- **PreBackup:** Before backup tasks are started on the source cluster
- **PostBackup:** After backup tasks are complete on the source cluster
- **PreRestore:** Before restore tasks are started on the target cluster
- **PostRestore:** After restore tasks are complete on the target cluster  
You can assign one hook to each migration step, up to a maximum of four hooks for a single migration plan.

The default **hook-runner** image is **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**. This image is based on Ansible Runner and includes **python-openshift** for Ansible Kubernetes resources and an updated **oc** binary. You can also create your own hook image with additional Ansible modules or tools.

The Ansible playbook is mounted on a hook container as a ConfigMap. The hook container runs as a Job on a cluster with a specified service account and namespace. The Job runs, even if the initial Pod is evicted or killed, until it reaches the default **backoffLimit (6)** or successful completion.

## 3.2. DEPLOYING THE CLUSTER APPLICATION MIGRATION TOOL

You can install the Cluster Application Migration Operator on your OpenShift Container Platform 4.2 target cluster and 4.2 source cluster. The Cluster Application Migration Operator installs the Cluster Application Migration (CAM) tool on the target cluster by default.



### NOTE

Optional: You can configure the Cluster Application Migration Operator to install the CAM tool [on an OpenShift Container Platform 3 cluster or on a remote cluster](#).

In a restricted environment, you can install the Cluster Application Migration Operator from a local mirror registry.

After you have installed the Cluster Application Migration Operator on your clusters, you can launch the CAM tool.

### 3.2.1. Installing the Cluster Application Migration Operator

You can install the Cluster Application Migration Operator with the Operation Lifecycle Manager (OLM) on an OpenShift Container Platform 4.2 target cluster and on an OpenShift Container Platform 4.1 source cluster.

#### 3.2.1.1. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

## Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New → MigrationController**.
8. Click **Create**.
9. Click **Workloads → Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 3.2.1.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 source cluster

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4 source cluster with the Operation Lifecycle Manager (OLM).

## Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New → MigrationController**.
8. Update the **migration\_controller** and **migration\_ui** parameters in the **spec** stanza:

```
spec:  
  ...  
  migration_controller: false
```

```
migration_ui: false
```

```
...
```

9. Click **Create**.
10. Click **Workloads** → **Pods** to verify that the Restic and Velero Pods are running.

### 3.2.2. Installing the Cluster Application Migration Operator in a restricted environment

You can build a custom Operator catalog image for OpenShift Container Platform 4, push it to a local mirror image registry, and configure OLM to install the Operator from the local registry.

#### 3.2.2.1. Configuring OperatorHub for restricted networks

Cluster administrators can configure OLM and OperatorHub to use local content in restricted network environments.

##### Prerequisites

- Cluster administrator access to an OpenShift Container Platform cluster and its internal registry.
- Separate workstation without network restrictions.
- If pushing images to the OpenShift Container Platform cluster's internal registry, the registry must be exposed with a route.
- **podman** version 1.4.4+

##### Procedure

1. **Disable the default OperatorSources.**

Add **disableAllDefaultSources: true** to the spec:

```
$ oc patch OperatorHub cluster --type json \
  -p [{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]
```

This disables the default OperatorSources that are configured by default during an OpenShift Container Platform installation.

2. **Retrieve package lists.**

To get the list of packages that are available for the default OperatorSources, run the following **curl** commands from your workstation without network restrictions:

```
$ curl https://quay.io/cnr/api/v1/packages?namespace=redhat-operators > packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=community-operators >> packages.txt
$ curl https://quay.io/cnr/api/v1/packages?namespace=certified-operators >> packages.txt
```

Each package in the new **packages.txt** is an Operator that you could add to your restricted network catalog. From this list, you could either pull every Operator or a subset that you would like to expose to users.

3. **Pull Operator content.**

For a given Operator in the package list, you must pull the latest released content:

```
$ curl https://quay.io/cnr/api/v1/packages/<namespace>/<operator_name>/<release>
```

This example uses the etcd Operator:

- a. Retrieve the digest:

```
$ curl https://quay.io/cnr/api/v1/packages/community-operators/etcd/0.0.12
```

- b. From that JSON, take the digest and use it to pull the gzipped archive:

```
$ curl -XGET https://quay.io/cnr/api/v1/packages/community-operators/etcd/blobs/sha256/8108475ee5e83a0187d6d0a729451ef1ce6d34c44a868a200151c36f3232822b \
-o etcd.tar.gz
```

- c. To pull the information out, you must untar the archive into a **manifests/<operator\_name>/** directory with all the other Operators that you want. For example, to untar to an existing directory called **manifests/etcd/**:

```
$ mkdir -p manifests/etcd/ 1
$ tar -xf etcd.tar.gz -C manifests/etcd/
```

- 1 Create different subdirectories for each extracted archive so that files are not overwritten by subsequent extractions for other Operators.

#### 4. Break apart **bundle.yaml** content, if necessary.

In your new **manifests/<operator\_name>** directory, the goal is to get your bundle in the following directory structure:

```
manifests/
├── etcd
│   ├── 0.0.12
│   │   ├── clusterserviceversion.yaml
│   │   └── customresourcedefinition.yaml
│   └── package.yaml
```

If you see files already in this structure, you can skip this step. However, if you instead see only a single file called **bundle.yaml**, you must first break this file up to conform to the required structure.

You must separate the CSV content under **data.clusterServiceVersion** (each file in the list), the CRD content under **data.customResourceDefinition** (each file in the list), and the package content under **data.Package** into their own files.

- a. For the CSV file creation, find the following lines in the **bundle.yaml** file:

```
data:
  clusterServiceVersions: |
```

Omit those lines, but save a new file consisting of the full CSV resource content beginning with the following lines, removing the prepended - character:

**Example clusterserviceversion.yaml file snippet**

```
apiVersion: operators.coreos.com/v1alpha1
kind: ClusterServiceVersion
[...]
```

- b. For the CRD file creation, find the following line in the **bundle.yaml** file:

```
customResourceDefinitions: |
```

Omit this line, but save new files consisting of each, full CRD resource content beginning with the following lines, removing the prepended - character:

**Example customresourcedefinition.yaml file snippet**

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
[...]
```

- c. For the package file creation, find the following line in the **bundle.yaml** file:

```
packages: |
```

Omit this line, but save a new file consisting of the package content beginning with the following lines, removing the prepended - character, and ending with a **packageName** entry:

**Example package.yaml file**

```
channels:
- currentCSV: etcdoperator.v0.9.4
  name: singlenamespace-alpha
- currentCSV: etcdoperator.v0.9.4-clusterwide
  name: clusterwide-alpha
defaultChannel: singlenamespace-alpha
packageName: etcd
```

**5. Identify images required by the Operators you want to use.**

Inspect the CSV files of each Operator for **image:** fields to identify the pull specs for any images required by the Operator, and note them for use in a later step.

For example, in the following **deployments** spec of an etcd Operator CSV:

```
spec:
  serviceAccountName: etcd-operator
  containers:
  - name: etcd-operator
    command:
    - etcd-operator
    - --create-crd=false
    image: quay.io/coreos/etcd-
operator@sha256:bd944a211eaf8f31da5e6d69e8541e7cada8f16a9f7a5a570b224789978199
43 1
```

```

env:
- name: MY_POD_NAMESPACE
  valueFrom:
    fieldRef:
      fieldPath: metadata.namespace
- name: MY_POD_NAME
  valueFrom:
    fieldRef:
      fieldPath: metadata.name

```

- 1 Image required by Operator.

## 6. Create an Operator catalog image.

- a. Save the following to a Dockerfile, for example named **custom-registry.Dockerfile**:

```

FROM registry.redhat.io/openshift4/ose-operator-registry:v4.2.24 AS builder

COPY manifests manifests

RUN /bin/initializer -o ./bundles.db

FROM registry.access.redhat.com/ubi7/ubi

COPY --from=builder /registry/bundles.db /bundles.db
COPY --from=builder /usr/bin/registry-server /registry-server
COPY --from=builder /bin/grpc_health_probe /bin/grpc_health_probe

EXPOSE 50051

ENTRYPOINT ["/registry-server"]

CMD ["--database", "bundles.db"]

```

- b. Use the **podman** command to create and tag the container image from the Dockerfile:

```

$ podman build -f custom-registry.Dockerfile \
  -t <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-
  registry 1

```

- 1 Tag the image for the internal registry of the restricted network OpenShift Container Platform cluster and any namespace.

## 7. Push the Operator catalog image to a registry.

Your new Operator catalog image must be pushed to a registry that the restricted network OpenShift Container Platform cluster can access. This can be the internal registry of the cluster itself or another registry that the cluster has network access to, such as an on-premise Quay Enterprise registry.

For this example, login and push the image to the internal registry OpenShift Container Platform cluster:

```
$ podman push <local_registry_host_name>:
<local_registry_host_port>/<namespace>/custom-registry
```

## 8. Create a CatalogSource pointing to the new Operator catalog image.

- a. Save the following to a file, for example **my-operator-catalog.yaml**:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  displayName: My Operator Catalog
  sourceType: grpc
  image: <local_registry_host_name>:<local_registry_host_port>/<namespace>/custom-
registry:latest
```

- b. Create the CatalogSource resource:

```
$ oc create -f my-operator-catalog.yaml
```

- c. Verify the CatalogSource and package manifest are created successfully:

```
# oc get pods -n openshift-marketplace
NAME READY STATUS RESTARTS AGE
my-operator-catalog-6njx6 1/1 Running 0 28s
marketplace-operator-d9f549946-96sgr 1/1 Running 0 26h

# oc get catalogsource -n openshift-marketplace
NAME DISPLAY TYPE PUBLISHER AGE
my-operator-catalog My Operator Catalog grpc 5s

# oc get packagemanifest -n openshift-marketplace
NAME CATALOG AGE
etcd My Operator Catalog 34s
```

You should also be able to view them from the **OperatorHub** page in the web console.

## 9. Mirror the images required by the Operators you want to use.

- a. Determine the images defined by the Operator(s) that you are expecting. This example uses the etcd Operator, requiring the **quay.io/coreos/etcd-operator** image.



### IMPORTANT

This procedure only shows mirroring Operator images themselves and not Operand images, which are the components that an Operator manages. Operand images must be mirrored as well; see each Operator's documentation to identify the required Operand images.

- b. To use mirrored images, you must first create an ImageContentSourcePolicy for each image to change the source location of the Operator catalog image. For example:

```

apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: etcd-operator
spec:
  repositoryDigestMirrors:
  - mirrors:
    - <local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator
    source: quay.io/coreos/etcd-operator

```

- c. Use the **oc image mirror** command from your workstation without network restrictions to pull the image from the source registry and push to the internal registry without being stored locally:

```

$ oc image mirror quay.io/coreos/etcd-operator \
  <local_registry_host_name>:<local_registry_host_port>/coreos/etcd-operator

```

You can now install the Operator from the OperatorHub on your restricted network OpenShift Container Platform cluster.

### 3.2.2.2. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster in a restricted environment

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 target cluster with the Operation Lifecycle Manager (OLM).

The Cluster Application Migration Operator installs the Cluster Application Migration tool on the target cluster by default.

#### Prerequisites

- You created a custom Operator catalog and pushed it to a mirror registry.
- You configured OLM to install the Cluster Application Migration Operator from the mirror registry.

#### Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more...**
7. Click **Create New → MigrationController**.



8. Click **Create**.
9. Click **Workloads** → **Pods** to verify that the Controller Manager, Migration UI, Restic, and Velero Pods are running.

### 3.2.2.3. Installing the Cluster Application Migration Operator on an OpenShift Container Platform 4.2 source cluster in a restricted environment

You can install the Cluster Application Migration Operator on an OpenShift Container Platform 4 source cluster with the Operation Lifecycle Manager (OLM).

#### Prerequisites

- You created a custom Operator catalog and pushed it to a mirror registry.
- You configured OLM to install the Cluster Application Migration Operator from the mirror registry.

#### Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use the **Filter by keyword** field (in this case, **Migration**) to find the **Cluster Application Migration Operator**.
3. Select the **Cluster Application Migration Operator** and click **Install**.
4. On the **Create Operator Subscription** page, select the **openshift-migration** namespace, and specify an approval strategy.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **Cluster Application Migration Operator** appears in the **openshift-migration** project with the status **InstallSucceeded**.
6. Under **Provided APIs**, click **View 12 more....**
7. Click **Create New** → **MigrationController**.
8. Click **Create**.

### 3.2.3. Launching the CAM web console

You can launch the CAM web console in a browser.

#### Procedure

1. Log in to the OpenShift Container Platform cluster on which you have installed the CAM tool.
2. Obtain the CAM web console URL by entering the following command:

```
$ oc get -n openshift-migration route/migration -o go-template='https://{ .spec.host }'
```

The output resembles the following: **https://migration-openshift-migration.apps.cluster.openshift.com**.

3. Launch a browser and navigate to the CAM web console.



#### NOTE

If you try to access the CAM web console immediately after installing the Cluster Application Migration Operator, the console may not load because the Operator is still configuring the cluster. Wait a few minutes and retry.

4. If you are using self-signed CA certificates, you will be prompted to accept the CA certificate of the source cluster's API server. The web page guides you through the process of accepting the remaining certificates.
5. Log in with your OpenShift Container Platform **username** and **password**.

### 3.3. CONFIGURING A REPLICATION REPOSITORY

You must configure an object storage to use as a replication repository. The Cluster Application Migration tool copies data from the source cluster to the replication repository, and then from the replication repository to the target cluster.

The CAM tool supports the [file system and snapshot data copy methods](#) for migrating data from the source cluster to the target cluster. You can select a method that is suited for your environment and is supported by your storage provider.

The following storage providers are supported:

- [Multi-Cloud Object Gateway \(MCG\)](#)
- [Amazon Web Services \(AWS\) S3](#)
- [Google Cloud Provider \(GCP\)](#)
- [Microsoft Azure](#)
- Generic S3 object storage, for example, Minio or Ceph S3

The source and target clusters must have network access to the replication repository during migration.

In a restricted environment, you can create an internally hosted replication repository. If you use a proxy server, you must ensure that your replication repository is whitelisted.



#### IMPORTANT

Configuring Multi-Cloud Object Gateway as a replication repository for migration is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

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

### 3.3.1. Configuring a Multi-Cloud Object Gateway storage bucket as a replication repository

You can install the OpenShift Container Storage Operator and configure a Multi-Cloud Object Gateway (MCG) storage bucket as a replication repository.

#### 3.3.1.1. Installing the OpenShift Container Storage Operator

You can install the OpenShift Container Storage Operator from OperatorHub.

##### Procedure

1. In the OpenShift Container Platform web console, click **Operators** → **OperatorHub**.
2. Use **Filter by keyword** (in this case, **OCS**) to find the **OpenShift Container Storage Operator**.
3. Select the **OpenShift Container Storage Operator** and click **Install**.
4. Select an **Update Channel**, **Installation Mode**, and **Approval Strategy**.
5. Click **Subscribe**.  
On the **Installed Operators** page, the **OpenShift Container Storage Operator** appears in the **openshift-storage** project with the status **Succeeded**.

#### 3.3.1.2. Creating the Multi-Cloud Object Gateway storage bucket

You can create the Multi-Cloud Object Gateway (MCG) storage bucket's Custom Resources (CRs).

##### Procedure

1. Log in to the OpenShift Container Platform cluster:

```
$ oc login
```

2. Create the **NooBaa** CR configuration file, **noobaa.yml**, with the following content:

```
apiVersion: noobaa.io/v1alpha1
kind: NooBaa
metadata:
  name: noobaa
  namespace: openshift-storage
spec:
  dbResources:
    requests:
      cpu: 0.5 1
      memory: 1Gi
  coreResources:
    requests:
      cpu: 0.5 2
      memory: 1Gi
```

**1** **2** For a very small cluster, you can change the **cpu** value to **0.1**.

3. Create the **NooBaa** object:

```
$ oc create -f noobaa.yml
```

4. Create the **BackingStore** CR configuration file, **bs.yml**, with the following content:

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
    name: mcg-pv-pool-bs
    namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 1
    resources:
      requests:
        storage: 50Gi 2
        storageClass: gp2 3
    type: pv-pool
```

- 1** Specify the number of volumes in the PV pool.
- 2** Specify the size of the volumes.
- 3** Specify the storage class.

5. Create the **BackingStore** object:

```
$ oc create -f bs.yml
```

6. Create the **BucketClass** CR configuration file, **bc.yml**, with the following content:

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
    name: mcg-pv-pool-bc
    namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
    - backingStores:
      - mcg-pv-pool-bs
    placement: Spread
```

7. Create the **BucketClass** object:

```
$ oc create -f bc.yml
```

8. Create the **ObjectBucketClaim** CR configuration file, **obc.yml**, with the following content:

```

apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: migstorage
  namespace: openshift-storage
spec:
  bucketName: migstorage 1
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: mcg-pv-pool-bc

```

- 1** Record the bucket name for adding the replication repository to the CAM web console.

9. Create the **ObjectBucketClaim** object:

```
$ oc create -f obc.yml
```

10. Watch the resource creation process to verify that the **ObjectBucketClaim** status is **Bound**:

```
$ watch -n 30 'oc get -n openshift-storage objectbucketclaim migstorage -o yaml'
```

This process can take five to ten minutes.

11. Obtain and record the following values, which are required when you add the replication repository to the CAM web console:

- S3 endpoint:

```
$ oc get route -n openshift-storage s3
```

- S3 provider access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_ACCESS_KEY_ID }}' | base64 -d
```

- S3 provider secret access key:

```
$ oc get secret -n openshift-storage migstorage -o go-template='{{
.data.AWS_SECRET_ACCESS_KEY }}' | base64 -d
```

### 3.3.2. Configuring an AWS S3 storage bucket as a replication repository

You can configure an AWS S3 storage bucket as a replication repository.

#### Prerequisites

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

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

## Procedure

1. Create an AWS S3 bucket:

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

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

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

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

3. Create an EC2 EBS snapshot policy:

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

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

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

```

        "s3:DeleteObject",
        "s3:PutObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": [
        "arn:aws:s3:::<bucket_name>/*" ❶
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation",
        "s3:ListBucketMultipartUploads"
    ],
    "Resource": [
        "arn:aws:s3:::<bucket_name>" ❷
    ]
}
]
}
EOF

```

- ❶ ❷ To grant access to a single S3 bucket, specify the bucket name. To grant access to all AWS S3 buckets, specify \* instead of a bucket name:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

- 1 2 Record the **AWS\_SECRET\_ACCESS\_KEY** and the **AWS\_ACCESS\_KEY\_ID** for adding the AWS repository to the CAM web console.

### 3.3.3. Configuring a Google Cloud Provider storage bucket as a replication repository

You can configure a Google Cloud Provider (GCP) storage bucket as a replication repository.

#### Prerequisites

- The GCP storage bucket must be accessible to the source and target clusters.
- You must have [gsutil](#) installed.
- If you are using the snapshot copy method:
  - The source and target clusters must be in the same region.
  - The source and target clusters must have the same storage class.
  - The storage class must be compatible with snapshots.

#### Procedure

1. Run **gsutil init** to log in:

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

Your current configuration has been set to: [default]

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

2. Set the **BUCKET** variable:

```
$ BUCKET=<bucket_name> 1
```

- 1 Specify your bucket name.

3. Create a storage bucket:

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

4. Set the **PROJECT\_ID** variable to your active project:

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

5. Create a **velero** service account:



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

6. Set the **SERVICE\_ACCOUNT\_EMAIL** variable to the service account's email address:

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

7. Grant permissions to the service account:

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

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

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

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

8. Save the service account's keys to the **credentials-velero** file in the current directory:

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

### 3.3.4. Configuring a Microsoft Azure Blob storage container as a replication repository

You can configure a Microsoft Azure Blob storage container as a replication repository.

#### Prerequisites

- You must have an [Azure storage account](#).
- You must have the [Azure CLI](#) installed.
- The Azure Blob storage container must be accessible to the source and target clusters.
- If you are using the snapshot copy method:
  - The source and target clusters must be in the same region.

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

## Procedure

1. Set the **AZURE\_RESOURCE\_GROUP** variable:

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. Create an Azure resource group:

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

- 1** Specify your location.

3. Set the **AZURE\_STORAGE\_ACCOUNT\_ID** variable:

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. Create an Azure storage account:

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

5. Set the **BLOB\_CONTAINER** variable:

```
$ BLOB_CONTAINER=velero
```

6. Create an Azure Blob storage container:

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

7. Create a service principal and credentials for **velero**:

```
$ AZURE_SUBSCRIPTION_ID=`az account list --query '[?isDefault].id' -o tsv`
$ AZURE_TENANT_ID=`az account list --query '[?isDefault].tenantId' -o tsv`
$ AZURE_CLIENT_SECRET=`az ad sp create-for-rbac --name "velero" --role "Contributor" --
query 'password' -o tsv`
$ AZURE_CLIENT_ID=`az ad sp list --display-name "velero" --query '[0].appId' -o tsv`
```

8. Save the service principal's credentials in the **credentials-velero** file:

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

## 3.4. MIGRATING APPLICATIONS WITH THE CAM WEB CONSOLE

You can migrate application workloads by adding your clusters and replication repository to the CAM web console. Then, you can create and run a migration plan.

If your cluster or replication repository are secured with self-signed certificates, you can create a CA certificate bundle file or disable SSL verification.

### 3.4.1. Creating a CA certificate bundle file

If you use a self-signed certificate to secure a cluster or a replication repository, certificate verification might fail with the following error message: **Certificate signed by unknown authority**.

You can create a custom CA certificate bundle file and upload it in the CAM web console when you add a cluster or a replication repository.

#### Procedure

Download a CA certificate from a remote endpoint and save it as a CA bundle file:

```
$ echo -n | openssl s_client -connect <host_FQDN>:<port> \ 1
| sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > <ca_bundle.cert> 2
```

- 1 Specify the host FQDN and port of the endpoint, for example, **api.my-cluster.example.com:6443**.
- 2 Specify the name of the CA bundle file.

### 3.4.2. Adding a cluster to the CAM web console

You can add a cluster to the CAM web console.

#### Prerequisites

If you are using Azure snapshots to copy data:

- You must provide the Azure resource group name when you add the source cluster.
- The source and target clusters must be in the same Azure resource group and in the same location.

#### Procedure

1. Log in to the cluster.
2. Obtain the service account token:



- **Replication repository name** Specify the replication repository name in the CAM web console.
  - **S3 bucket name:** Specify the name of the S3 bucket you created.
  - **S3 bucket region:** Specify the S3 bucket region. **Required** for AWS S3. **Optional** for other S3 providers.
  - **S3 endpoint:** Specify the URL of the S3 service, not the bucket, for example, **https://<s3-storage.apps.cluster.com>**. **Required** for a generic S3 provider. You must use the **https://** prefix.
  - **S3 provider access key:** Specify the **<AWS\_SECRET\_ACCESS\_KEY>** for AWS or the S3 provider access key for MCG.
  - **S3 provider secret access key:** Specify the **<AWS\_ACCESS\_KEY\_ID>** for AWS or the S3 provider secret access key for MCG.
  - **Require SSL verification:** Clear this check box if you are using a generic S3 provider.
  - If you use a custom CA bundle, click **Browse** and browse to the Base64-encoded CA bundle file.
- **GCP:**
    - **Replication repository name** Specify the replication repository name in the CAM web console.
    - **GCP bucket name:** Specify the name of the GCP bucket.
    - **GCP credential JSON blob:** Specify the string in the **credentials-velero** file.
  - **Azure:**
    - **Replication repository name** Specify the replication repository name in the CAM web console.
    - **Azure resource group:** Specify the resource group of the Azure Blob storage.
    - **Azure storage account name:** Specify the Azure Blob storage account name.
    - **Azure credentials - INI file contents** Specify the string in the **credentials-velero** file.
4. Click **Add repository** and wait for connection validation.
  5. Click **Close**.  
The new repository appears in the **Replication repositories** section.

### 3.4.4. Changing migration plan limits for large migrations

You can change the migration plan limits for large migrations.



#### IMPORTANT

Changes should first be tested in your environment to avoid a failed migration.

A single migration plan has the following default limits:

- 10 namespaces  
If this limit is exceeded, the CAM web console displays a **Namespace limit exceeded** error and you cannot create a migration plan.
- 100 Pods  
If the Pod limit is exceeded, the CAM web console displays a warning message similar to the following example: **Plan has been validated with warning condition(s). See warning message. Pod limit: 100 exceeded, found: 104.**
- 100 persistent volumes  
If the persistent volume limit is exceeded, the CAM web console displays a similar warning message.

## Procedure

1. Edit the Migration controller CR:

```
$ oc get migrationcontroller -n openshift-migration
NAME AGE
migration-controller 5d19h

$ oc edit migrationcontroller -n openshift-migration
```

2. Update the following parameters:

```
...
migration_controller: true

# This configuration is loaded into mig-controller, and should be set on the
# cluster where `migration_controller: true`
mig_pv_limit: 100
mig_pod_limit: 100
mig_namespace_limit: 10
...
```

### 3.4.5. Creating a migration plan in the CAM web console

You can create a migration plan in the CAM web console.

#### Prerequisites

- The CAM web console must contain the following:
  - Source cluster
  - Target cluster, which is added automatically during the CAM tool installation
  - Replication repository
- The source and target clusters must have network access to each other and to the replication repository.
- If you use snapshots to copy data, the source and target clusters must run on the same cloud provider (AWS, GCP, or Azure) and in the same region.

## Procedure

1. Log in to the CAM web console.
2. In the **Plans** section, click **Add plan**.
3. Enter the **Plan name** and click **Next**.  
The **Plan name** can contain up to 253 lower-case alphanumeric characters (**a-z, 0-9**). It must not contain spaces or underscores (**\_**).
4. Select a **Source cluster**.
5. Select a **Target cluster**.
6. Select a **Replication repository**.
7. Select the projects to be migrated and click **Next**.
8. Select **Copy** or **Move** for the PVs:
  - **Copy** copies the data in a source cluster's PV to the replication repository and then restores it on a newly created PV, with similar characteristics, in the target cluster.  
Optional: You can verify data copied with the filesystem method by selecting **Verify copy**. This option, which generates a checksum for each source file and checks it after restoration, significantly reduces performance.
  - **Move** unmounts a remote volume (for example, NFS) from the source cluster, creates a PV resource on the target cluster pointing to the remote volume, and then mounts the remote volume on the target cluster. Applications running on the target cluster use the same remote volume that the source cluster was using. The remote volume must be accessible to the source and target clusters.
9. Click **Next**.
10. Select a **Copy method** for the PVs:
  - **Snapshot** backs up and restores the disk using the cloud provider's snapshot functionality. It is significantly faster than **Filesystem**.



### NOTE

The storage and clusters must be in the same region and the storage class must be compatible.

- **Filesystem** copies the data files from the source disk to a newly created target disk.
11. Select a **Storage class** for the PVs.  
If you selected the **Filesystem** copy method, you can change the storage class during migration, for example, from Red Hat Gluster Storage or NFS storage to Red Hat Ceph Storage.
  12. Click **Next**.
  13. If you want to add a migration hook, click **Add Hook** and perform the following steps:
    - a. Specify the name of the hook.
    - b. Select **Available playbooks** to use your own playbook or **Custom container image for a hook**.

- d. Select **ANSIBLE PLAYBOOK** to use your own playbook or **CUSTOM CONTAINER IMAGE** for a hook written in another language.
  - c. Click **Browse** to upload the playbook.
  - d. Optional: If you are not using the default Ansible runtime image, specify your custom Ansible image.
  - e. Specify the cluster on which you want the hook to run.
  - f. Specify the service account name.
  - g. Specify the namespace.
  - h. Select the migration step at which you want the hook to run:
    - PreBackup: Before backup tasks are started on the source cluster
    - PostBackup: After backup tasks are complete on the source cluster
    - PreRestore: Before restore tasks are started on the target cluster
    - PostRestore: After restore tasks are complete on the target cluster
14. Click **Add**.  
You can add up to four hooks to a migration plan, assigning each hook to a different migration step.
15. Click **Finish**.
16. Click **Close**.  
The migration plan appears in the **Plans** section.

### 3.4.6. Running a migration plan in the CAM web console

You can stage or migrate applications and data with the migration plan you created in the CAM web console.

#### Prerequisites

The CAM web console must contain the following:


- Source cluster
- Target cluster, which is added automatically during the CAM tool installation
- Replication repository
- Valid migration plan

#### Procedure

1. Log in to the CAM web console on the target cluster.
2. Select a migration plan.
3. Click **Stage** to copy data from the source cluster to the target cluster without stopping the application.



You can run **Stage** multiple times to reduce the actual migration time.

4. When you are ready to migrate the application workload, click **Migrate**. **Migrate** stops the application workload on the source cluster and recreates its resources on the target cluster.
5. Optional: In the **Migrate** window, you can select **Do not stop applications on the source cluster during migration**.
6. Click **Migrate**.
7. Optional: To stop a migration in progress, click the Options menu  and select **Cancel**.
8. When the migration is complete, verify that the application migrated successfully in the OpenShift Container Platform web console:
  - a. Click **Home** → **Projects**.
  - b. Click the migrated project to view its status.
  - c. In the **Routes** section, click **Location** to verify that the application is functioning, if applicable.
  - d. Click **Workloads** → **Pods** to verify that the Pods are running in the migrated namespace.
  - e. Click **Storage** → **Persistent volumes** to verify that the migrated persistent volume is correctly provisioned.

## 3.5. TROUBLESHOOTING

You can view the migration Custom Resources (CRs) and download logs to troubleshoot a failed migration.

If the application was stopped during the failed migration, you must roll it back manually in order to prevent data corruption.

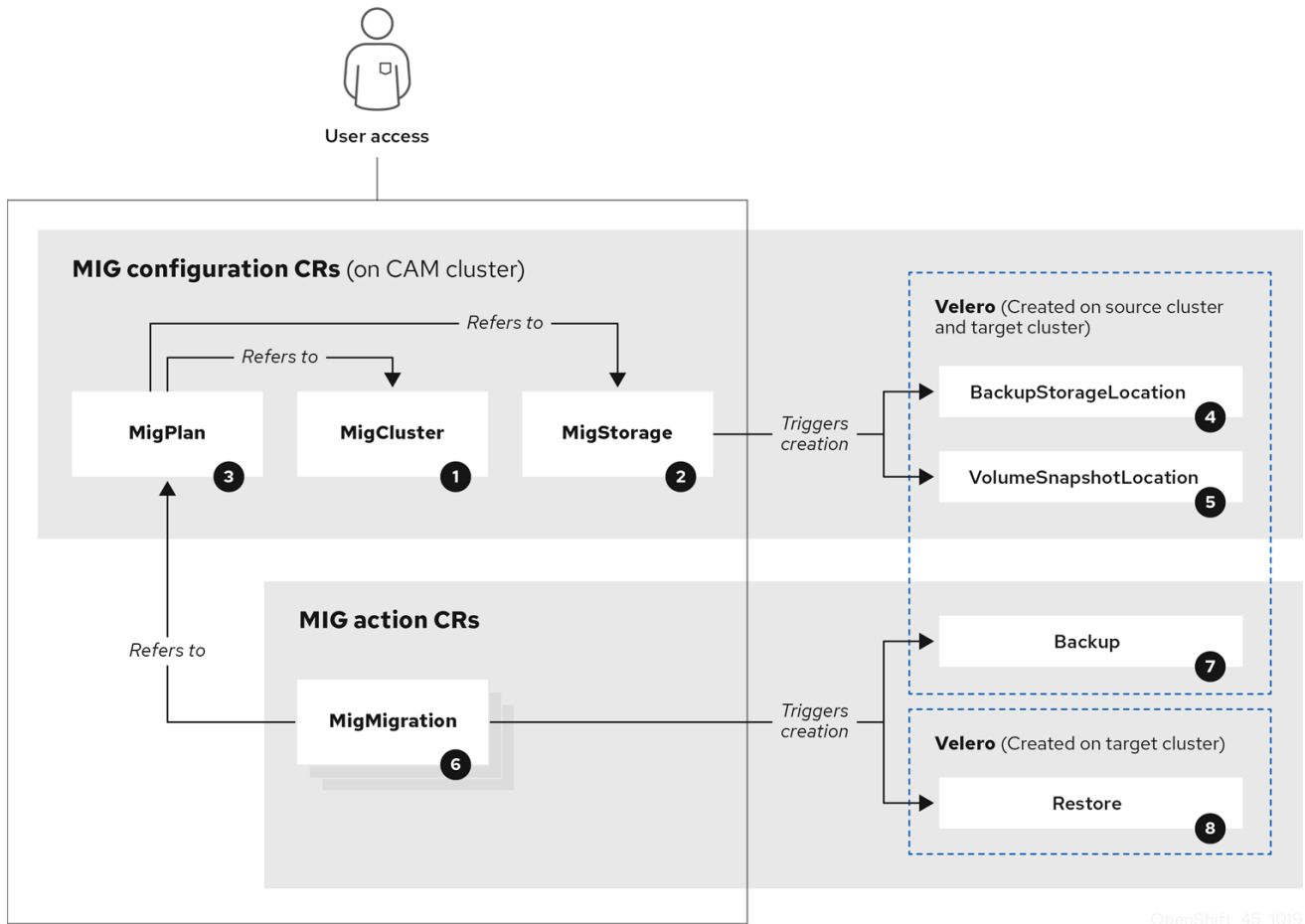


### NOTE

Manual rollback is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

### 3.5.1. Viewing migration Custom Resources

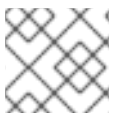
The Cluster Application Migration (CAM) tool creates the following Custom Resources (CRs):



OpenShift\_45\_1019

- 1 **MigCluster** (configuration, CAM cluster): Cluster definition
- 2 **MigStorage** (configuration, CAM cluster): Storage definition
- 3 **MigPlan** (configuration, CAM cluster): Migration plan

The MigPlan CR describes the source and target clusters, repository, and namespace(s) being migrated. It is associated with 0, 1, or many MigMigration CRs.



#### NOTE

Deleting a MigPlan CR deletes the associated MigMigration CRs.

- 4 **BackupStorageLocation** (configuration, CAM cluster): Location of Velero backup objects
- 5 **VolumeSnapshotLocation** (configuration, CAM cluster): Location of Velero volume snapshots
- 6 **MigMigration** (action, CAM cluster): Migration, created during migration

A MigMigration CR is created every time you stage or migrate data. Each MigMigration CR is associated with a MigPlan CR.

**7** **Backup** (action, source cluster): When you run a migration plan, the MigMigration CR creates two Velero backup CRs on each source cluster:

- Backup CR #1 for Kubernetes objects
- Backup CR #2 for PV data

**8** **Restore** (action, target cluster): When you run a migration plan, the MigMigration CR creates two Velero restore CRs on the target cluster:

- Restore CR #1 (using Backup CR #2) for PV data
- Restore CR #2 (using Backup CR #1) for Kubernetes objects

## Procedure

1. Get the CR name:

```
$ oc get <migration_cr> -n openshift-migration 1
```

- 1** Specify the migration CR, for example, **migmigration**.

The output is similar to the following:

```
NAME                                AGE
88435fe0-c9f8-11e9-85e6-5d593ce65e10 6m42s
```

2. View the CR:

```
$ oc describe <migration_cr> <88435fe0-c9f8-11e9-85e6-5d593ce65e10> -n openshift-migration
```

The output is similar to the following examples.

## MigMigration example

```
name:      88435fe0-c9f8-11e9-85e6-5d593ce65e10
namespace: openshift-migration
labels:    <none>
annotations: touch: 3b48b543-b53e-4e44-9d34-33563f0f8147
apiVersion: migration.openshift.io/v1alpha1
kind:      MigMigration
metadata:
  creationTimestamp: 2019-08-29T01:01:29Z
  generation:       20
  resourceVersion:  88179
  selfLink:         /apis/migration.openshift.io/v1alpha1/namespaces/openshift-
migration/migmigrations/88435fe0-c9f8-11e9-85e6-5d593ce65e10
  uid:              8886de4c-c9f8-11e9-95ad-0205fe66cbb6
spec:
  migPlanRef:
    name:      socks-shop-mig-plan
    namespace: openshift-migration
```

```

  quiescePods: true
  stage:      false
status:
  conditions:
    category:      Advisory
    durable:       True
    lastTransitionTime: 2019-08-29T01:03:40Z
    message:       The migration has completed successfully.
    reason:        Completed
    status:        True
    type:          Succeeded
  phase:        Completed
  startTimestamp: 2019-08-29T01:01:29Z
  events:       <none>

```

## Velero backup CR #2 example (PV data)

```

apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.105.179:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-44dd3bd5-c9f8-11e9-95ad-
0205fe66cbb6
  creationTimestamp: "2019-08-29T01:03:15Z"
  generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
  generation: 1
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    velero.io/storage-location: myrepo-vpzq9
  name: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
  namespace: openshift-migration
  resourceVersion: "87313"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/backups/88435fe0-c9f8-11e9-85e6-
5d593ce65e10-59gb7
  uid: c80dbbc0-c9f8-11e9-95ad-0205fe66cbb6
spec:
  excludedNamespaces: []
  excludedResources: []
  hooks:
    resources: []
  includeClusterResources: null
  includedNamespaces:
  - sock-shop
  includedResources:
  - persistentvolumes
  - persistentvolumeclaims
  - namespaces
  - imagestreams
  - imagestreamtags
  - secrets
  - configmaps

```

```

- pods
labelSelector:
  matchLabels:
    migration-included-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
storageLocation: myrepo-vpzq9
ttl: 720h0m0s
volumeSnapshotLocations:
- myrepo-wv6fx
status:
  completionTimestamp: "2019-08-29T01:02:36Z"
  errors: 0
  expiration: "2019-09-28T01:02:35Z"
  phase: Completed
  startTimestamp: "2019-08-29T01:02:35Z"
  validationErrors: null
  version: 1
  volumeSnapshotsAttempted: 0
  volumeSnapshotsCompleted: 0
  warnings: 0

```

### Velero restore CR #2 example (Kubernetes resources)

```

apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.90.187:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-36f54ca7-c925-11e9-825a-06fa9fb68c88
  creationTimestamp: "2019-08-28T00:09:49Z"
  generateName: e13a1b60-c927-11e9-9555-d129df7f3b96-
  generation: 3
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: e18252c9-c927-11e9-825a-06fa9fb68c88
    migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
  name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  namespace: openshift-migration
  resourceVersion: "82329"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  includedNamespaces: null
  includedResources: null

```

```


namespaceMapping: null
restorePVs: true
status:
  errors: 0
  failureReason: ""
  phase: Completed
  validationErrors: null
  warnings: 15

```

### 3.5.2. Downloading migration logs

You can download the Velero, Restic, and Migration controller logs in the CAM web console to troubleshoot a failed migration.

#### Procedure

1. Log in to the CAM console.
2. Click **Plans** to view the list of migration plans.
3. Click the **Options** menu  of a specific migration plan and select **Logs**.
4. Click **Download Logs** to download the logs of the Migration controller, Velero, and Restic for all clusters.
5. To download a specific log:
  - a. Specify the log options:
    - **Cluster:** Select the source, target, or CAM host cluster.
    - **Log source:** Select **Velero**, **Restic**, or **Controller**.
    - **Pod source:** Select the Pod name, for example, **controller-manager-78c469849c-v6wcf**  
The selected log is displayed.

You can clear the log selection settings by changing your selection.

- a. Click **Download Selected** to download the selected log.

Optionally, you can access the logs by using the CLI, as in the following example:

```

$ oc get pods -n openshift-migration | grep controller
controller-manager-78c469849c-v6wcf      1/1   Running   0      4h49m

$ oc logs controller-manager-78c469849c-v6wcf -f -n openshift-migration

```

### 3.5.3. Error messages

#### 3.5.3.1. Restic timeout error message in the Velero Pod log

If a migration fails because Restic times out, the following error appears in the Velero Pod log:

```
level=error msg="Error backing up item" backup=velero/monitoring error="timed out waiting for all
PodVolumeBackups to complete"
error.file="/go/src/github.com/heptio/velero/pkg/restic/backupper.go:165"
error.function="github.com/heptio/velero/pkg/restic.(*backupper).BackupPodVolumes" group=v1
```

The default value of **restic\_timeout** is one hour. You can increase this parameter for large migrations, keeping in mind that a higher value may delay the return of error messages.

### Procedure

1. In the OpenShift Container Platform web console, navigate to **Operators** → **Installed Operators**.
2. Click **Cluster Application Migration Operator**.
3. In the **MigrationController** tab, click **migration-controller**.
4. In the **YAML** tab, update the following parameter value:

```
spec:
  restic_timeout: 1h 1
```

- 1** Valid units are **h** (hours), **m** (minutes), and **s** (seconds), for example, **3h30m15s**.

5. Click **Save**.

### 3.5.3.2. ResticVerifyErrors in the MigMigration Custom Resource

If data verification fails when migrating a PV with the filesystem data copy method, the following error appears in the MigMigration Custom Resource (CR):

```
status:
  conditions:
  - category: Warn
    durable: true
    lastTransitionTime: 2020-04-16T20:35:16Z
    message: There were verify errors found in 1 Restic volume restores. See restore `<registry-example-migration-rvwcm>`
      for details 1
    status: "True"
    type: ResticVerifyErrors 2
```

- 1** The error message identifies the Restore CR name.
- 2** **ResticErrors** also appears. **ResticErrors** is a general error warning that includes verification errors.



#### NOTE

A data verification error does not cause the migration process to fail.

You can check the target cluster's Restore CR to identify the source of the data verification error.

## Procedure

1. Log in to the target cluster.
2. View the Restore CR:

```
$ oc describe <registry-example-migration-rwcm> -n openshift-migration
```

The output identifies the PV with **PodVolumeRestore** errors:

```
status:
  phase: Completed
  podVolumeRestoreErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rwcm-98t49>
    namespace: openshift-migration
  podVolumeRestoreResticErrors:
  - kind: PodVolumeRestore
    name: <registry-example-migration-rwcm-98t49>
    namespace: openshift-migration
```

3. View the **PodVolumeRestore** CR:

```
$ oc describe <migration-example-rwcm-98t49>
```

The output identifies the Restic Pod that logged the errors:

```
completionTimestamp: 2020-05-01T20:49:12Z
errors: 1
resticErrors: 1
...
resticPod: <restic-nr2v5>
```

4. View the Restic Pod log:

```
$ oc logs -f restic-nr2v5
```

### 3.5.4. Manually rolling back a migration

If your application was stopped during a failed migration, you must roll it back manually in order to prevent data corruption in the PV.

This procedure is not required if the application was not stopped during migration, because the original application is still running on the source cluster.

## Procedure

1. On the target cluster, switch to the migrated project:

```
$ oc project <project>
```

2. Get the deployed resources:



```
$ oc get all
```

3. Delete the deployed resources to ensure that the application is not running on the target cluster and accessing data on the PVC:

```
$ oc delete <resource_type>
```

4. To stop a DaemonSet without deleting it, update the **nodeSelector** in the YAML file:

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: hello-daemonset
spec:
  selector:
    matchLabels:
      name: hello-daemonset
  template:
    metadata:
      labels:
        name: hello-daemonset
    spec:
      nodeSelector:
        role: worker ❶
```

- ❶ Specify a **nodeSelector** value that does not exist on any node.

5. Update each PV's reclaim policy so that unnecessary data is removed. During migration, the reclaim policy for bound PVs is **Retain**, to ensure that data is not lost when an application is removed from the source cluster. You can remove these PVs during rollback.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain ❶
  ...
status:
  ...
```

- ❶ Specify **Recycle** or **Delete**.

6. On the source cluster, switch to your migrated project:

```
$ oc project <project_name>
```

7. Obtain the project's deployed resources:

```
$ oc get all
```

8. Start one or more replicas of each deployed resource:

```
$ oc scale --replicas=1 <resource_type>/<resource_name>
```

9. Update the **nodeSelector** of a DaemonSet to its original value, if you changed it during the procedure.

### 3.5.5. Gathering data for a customer support case

If you open a customer support case, you can run the **must-gather** tool with the **openshift-migration-must-gather-rhel8** image to collect information about your cluster and upload it to the [Red Hat Customer Portal](#).

The **openshift-migration-must-gather-rhel8** image collects logs and Custom Resource data that are not collected by the default **must-gather** image.

#### Procedure

1. Navigate to the directory where you want to store the **must-gather** data.
2. Run the **oc adm must-gather** command:

```
$ oc adm must-gather --image=registry.redhat.io/rhcam-1-2/openshift-migration-must-gather-rhel8
```

The **must-gather** tool collects the cluster information and stores it in a **must-gather.local.<uid>** directory.

3. Remove authentication keys and other sensitive information from the **must-gather** data.
4. Create an archive file containing the contents of the **must-gather.local.<uid>** directory:

```
$ tar cvaf must-gather.tar.gz must-gather.local.<uid>/
```

You can attach the compressed file to your customer support case on the [Red Hat Customer Portal](#).

### 3.5.6. Known issues

This release has the following known issues:

- During migration, the Cluster Application Migration (CAM) tool preserves the following namespace annotations:
  - **openshift.io/sa.scc.mcs**
  - **openshift.io/sa.scc.supplemental-groups**
  - **openshift.io/sa.scc.uid-range**  
These annotations preserve the UID range, ensuring that the containers retain their file system permissions on the target cluster. There is a risk that the migrated UIDs could duplicate UIDs within an existing or future namespace on the target cluster. ([BZ#1748440](#))

- If an AWS bucket is added to the CAM web console and then deleted, its status remains **True** because the MigStorage CR is not updated. ([BZ#1738564](#))
- Most cluster-scoped resources are not yet handled by the CAM tool. If your applications require cluster-scoped resources, you may have to create them manually on the target cluster.
- If a migration fails, the migration plan does not retain custom PV settings for quiesced pods. You must manually roll back the migration, delete the migration plan, and create a new migration plan with your PV settings. ([BZ#1784899](#))
- If a large migration fails because Restic times out, you can increase the **restic\_timeout** parameter value (default: **1h**) in the Migration controller CR.
- If you select the data verification option for PVs that are migrated with the filesystem copy method, performance is significantly slower. Velero generates a checksum for each file and checks it when the file is restored.