OpenShift Container Platform 4.8

Migration Toolkit for Containers

Migrating to OpenShift Container Platform 4

Last Updated: 2023-02-28
OpenShift Container Platform 4.8 Migration Toolkit for Containers

Migrating to OpenShift Container Platform 4
**Abstract**

This document provides instructions for migrating your stateful application workloads between OpenShift Container Platform 4 clusters.
# Table of Contents

## CHAPTER 1. ABOUT THE MIGRATION TOOLKIT FOR CONTAINERS

1.1. TERMINOLOGY ................................................. 6
1.2. MTC WORKFLOW ............................................. 7
1.3. ABOUT DATA COPY METHODS
   1.3.1. File system copy method .......................... 10
   1.3.2. Snapshot copy method ............................. 10
1.4. DIRECT VOLUME MIGRATION AND DIRECT IMAGE MIGRATION ................................. 11

## CHAPTER 2. MIGRATION TOOLKIT FOR CONTAINERS RELEASE NOTES

2.1. MIGRATION TOOLKIT FOR CONTAINERS 1.6 RELEASE NOTES .................................. 12
   2.1.1. New features and enhancements .................... 12
   2.1.2. Deprecated features .................................. 12
   2.1.3. Known issues ........................................ 12
2.2. MIGRATION TOOLKIT FOR CONTAINERS 1.5 RELEASE NOTES .................................. 12
   2.2.1. New features and enhancements .................... 13
   2.2.2. Deprecated features .................................. 13
   2.2.3. Known issues ........................................ 13
   2.2.4. Technical changes .................................... 14
2.3. MIGRATION TOOLKIT FOR CONTAINERS 1.4 RELEASE NOTES .................................. 14
   2.3.1. Known issues ........................................ 14

## CHAPTER 3. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS

3.1. COMPATIBILITY GUIDELINES .................................. 15
3.2. INSTALLING THE LEGACY MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSHIFT CONTAINER PLATFORM 4.2 TO 4.5 ........................................ 16
3.3. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSHIFT CONTAINER PLATFORM 4.8 ........................................ 17
3.4. PROXY CONFIGURATION ....................................... 18
   3.4.1. Direct volume migration ............................. 18
      3.4.1.1. TCP proxy setup for DVM ......................... 18
      3.4.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy? .................. 19
      3.4.1.3. Known issue ....................................... 19
   3.4.2. Tuning network policies for migrations ............... 19
      3.4.2.1. NetworkPolicy configuration .................... 19
         3.4.2.1.1. Egress traffic from Rsync pods ............. 19
         3.4.2.1.2. Ingress traffic to Rsync pods .............. 20
      3.4.2.2. EgressNetworkPolicy configuration .......... 20
      3.4.2.3. Configuring supplemental groups for Rsync pods ............... 21
   3.4.3. Configuring proxies .................................. 21
3.5. CONFIGURING A REPLICATION REPOSITORY .................................................. 22
   3.5.1. Prerequisites ......................................... 22
   3.5.2. Configuring Multi-Cloud Object Gateway ............. 22
      3.5.2.1. Installing the OpenShift Container Storage Operator ............. 22
      3.5.2.2. Creating the Multi-Cloud Object Gateway storage bucket ...... 23
   3.5.3. Configuring Amazon Web Services S3 .................. 25
   3.5.4. Configuring Google Cloud Platform .................. 28
   3.5.5. Configuring Microsoft Azure Blob .................... 29
   3.5.6. Additional resources ................................. 31
3.6. UNINSTALLING MTC AND DELETING RESOURCES ........................................ 31

## CHAPTER 4. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS IN A RESTRICTED NETWORK ENVIRONMENT ............................................ 33
4.1. COMPATIBILITY GUIDELINES 33
4.2. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSShift CONTAINER PLATFORM 4.8 34
4.3. INSTALLING THE LEGACY MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSShift CONTAINER PLATFORM 4.2 TO 4.5 35
4.4. PROXY CONFIGURATION 37
  4.4.1. Direct volume migration 37
  4.4.1.1. TCP proxy setup for DVM 37
  4.4.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy? 38
  4.4.1.3. Known issue 38
  4.4.2. Tuning network policies for migrations 38
  4.4.2.1. NetworkPolicy configuration 38
    4.4.2.1.1. Egress traffic from Rsync pods 38
    4.4.2.1.2. Ingress traffic to Rsync pods 39
    4.4.2.2. EgressNetworkPolicy configuration 39
  4.4.2.3. Configuring supplemental groups for Rsync pods 39
  4.4.3. Configuring proxies 40
4.5. CONFIGURING A REPLICATION REPOSITORY 41
  4.5.1. Prerequisites 41
  4.5.2. Configuring Multi-Cloud Object Gateway 41
    4.5.2.1. Installing the OpenShift Container Storage Operator 41
    4.5.2.2. Creating the Multi-Cloud Object Gateway storage bucket 41
  4.5.3. Additional resources 44
4.6. UNINSTALLING MTC AND DELETING RESOURCES 44

CHAPTER 5. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ................................. 46
  5.1. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSShift CONTAINER PLATFORM 4.8 46
  5.2. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSShift CONTAINER PLATFORM VERSIONS 4.2 TO 4.5 46
  5.3. UPGRADING MTC 1.3 TO 1.7 47

CHAPTER 6. PREMIGRATION CHECKLISTS ............................................................................. 49
  6.1. CLUSTER HEALTH CHECKLIST 49
  6.2. SOURCE CLUSTER CHECKLIST 49
  6.3. TARGET CLUSTER CHECKLIST 50

CHAPTER 7. NETWORK CONSIDERATIONS ........................................................................ 51
  7.1. DNS CONSIDERATIONS 51
    7.1.1. Isolating the DNS domain of the target cluster from the clients 51
    7.1.2. Setting up the target cluster to accept the source DNS domain 51
  7.2. NETWORK TRAFFIC REDIRECTION STRATEGIES 52

CHAPTER 8. MIGRATING YOUR APPLICATIONS ................................................................... 54
  8.1. MIGRATION PREREQUISITES 54
  8.2. MIGRATING YOUR APPLICATIONS BY USING THE MTC WEB CONSOLE 55
    8.2.1. Launching the MTC web console 55
    8.2.2. Adding a cluster to the MTC web console 56
    8.2.3. Adding a replication repository to the MTC web console 57
    8.2.4. Creating a migration plan in the MTC web console 59
      Additional resources for persistent volume copy methods 61
    8.2.5. Running a migration plan in the MTC web console 61

CHAPTER 9. ADVANCED MIGRATION OPTIONS .................................................................. 63
  9.1. TERMINOLOGY 63
9.2. MIGRATING APPLICATIONS BY USING THE COMMAND LINE
9.2.1. Migration prerequisites
9.2.2. Creating a registry route for direct image migration
9.2.3. Proxy configuration
9.2.3.1. Direct volume migration
9.2.3.1.1. TCP proxy setup for DVM
9.2.3.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy?
9.2.3.1.3. Known issue
9.2.3.2. Tuning network policies for migrations
9.2.3.2.1. NetworkPolicy configuration
9.2.3.2.1.1. Egress traffic from Rsync pods
9.2.3.2.1.2. Ingress traffic to Rsync pods
9.2.3.2.2. EgressNetworkPolicy configuration
9.2.3.2.3. Configuring supplemental groups for Rsync pods
9.2.3.3. Configuring proxies
9.2.4. Migrating an application by using the MTC API
9.2.5. State migration
Additional resources
9.3. MIGRATION HOOKS
9.3.1. Writing an Ansible playbook for a migration hook
9.3.1.1. Ansible modules
9.3.1.2. Environment variables
9.4. MIGRATION PLAN OPTIONS
9.4.1. Excluding resources
9.4.2. Mapping namespaces
9.4.3. Excluding persistent volume claims
9.4.4. Mapping persistent volume claims
9.4.5. Editing persistent volume attributes
Additional resources
9.4.6. Performing a state migration of Kubernetes objects by using the MTC API
9.5. MIGRATION CONTROLLER OPTIONS
9.5.1. Increasing limits for large migrations
9.5.2. Enabling persistent volume resizing for direct volume migration
9.5.3. Enabling cached Kubernetes clients

CHAPTER 10. TROUBLESHOOTING
10.1. MTC WORKFLOW
About MTC custom resources
10.2. MTC CUSTOM RESOURCE MANIFESTS
10.2.1. DirectImageMigration
10.2.2. DirectImageStreamMigration
10.2.3. DirectVolumeMigration
10.2.4. DirectVolumeMigrationProgress
10.2.5. MigAnalytic
10.2.6. MigCluster
10.2.7. MigHook
10.2.8. MigMigration
10.2.9. MigPlan
10.2.10. MigStorage
10.3. LOGS AND DEBUGGING TOOLS
10.3.1. Viewing migration plan resources
10.3.2. Viewing a migration plan log
10.3.3. Using the migration log reader
10.3.4. Accessing performance metrics
- 10.3.4.1. Provided metrics
  - 10.3.4.1.1. cam_app_workload_migrations
  - 10.3.4.1.2. mtc_client_request_count
  - 10.3.4.1.3. mtc_client_request_elapsed
  - 10.3.4.1.4. Useful queries

10.3.5. Using the must-gather tool
- Viewing metrics data with the Prometheus console

10.3.6. Debugging Velero resources with the Velero CLI tool
- Syntax
- Help option
- Describe command
- Logs command

10.3.7. Debugging a partial migration failure

10.3.8. Using MTC custom resources for troubleshooting

10.4. COMMON ISSUES AND CONCERNS
- 10.4.1. Direct volume migration does not complete
- 10.4.2. Error messages and resolutions
  - 10.4.2.1. CA certificate error displayed when accessing the MTC console for the first time
  - 10.4.2.2. OAuth timeout error in the MTC console
  - 10.4.2.3. Certificate signed by unknown authority error
  - 10.4.2.4. Backup storage location errors in the Velero pod log
  - 10.4.2.5. Pod volume backup timeout error in the Velero pod log
  - 10.4.2.6. Restic verification errors in the MigMigration custom resource
  - 10.4.2.7. Restic permission error when migrating from NFS storage with root_squash enabled

10.5. ROLLING BACK A MIGRATION
- 10.5.1. Rolling back a migration by using the MTC web console
- 10.5.2. Rolling back a migration from the command line interface
- 10.5.3. Rolling back a migration manually
  - Additional resources

OpenShift Container Platform 4.8 Migration Toolkit for Containers
CHAPTER 1. ABOUT THE MIGRATION TOOLKIT FOR CONTAINERS

The Migration Toolkit for Containers (MTC) enables you to migrate stateful application workloads between OpenShift Container Platform 4 clusters at the granularity of a namespace.

NOTE

If you are migrating from OpenShift Container Platform 3, see About migrating from OpenShift Container Platform 3 to 4 and Installing the legacy Migration Toolkit for Containers Operator on OpenShift Container Platform 3.

You can migrate applications within the same cluster or between clusters by using state migration.

MTC provides a web console and an API, based on Kubernetes custom resources, to help you control the migration and minimize application downtime.

The MTC console is installed on the target cluster by default. You can configure the Migration Toolkit for Containers Operator to install the console on a remote cluster.

See Advanced migration options for information about the following topics:

- Automating your migration with migration hooks and the MTC API.
- Configuring your migration plan to exclude resources, support large-scale migrations, and enable automatic PV resizing for direct volume migration.

1.1. TERMINOLOGY

Table 1.1. MTC terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source cluster</td>
<td>Cluster from which the applications are migrated.</td>
</tr>
<tr>
<td>Destination cluster[1]</td>
<td>Cluster to which the applications are migrated.</td>
</tr>
</tbody>
</table>
| Replication repository| Object storage used for copying images, volumes, and Kubernetes objects during indirect migration or for Kubernetes objects during direct volume migration or direct image migration.  
The replication repository must be accessible to all clusters. |
| Host cluster          | Cluster on which the migration-controller pod and the web console are running. The host cluster is usually the destination cluster but this is not required.  
The host cluster does not require an exposed registry route for direct image migration. |
Remote cluster

A remote cluster is usually the source cluster but this is not required.

A remote cluster requires a **Secret** custom resource that contains the **migration-controller** service account token.

A remote cluster requires an exposed secure registry route for direct image migration.

Indirect migration

Images, volumes, and Kubernetes objects are copied from the source cluster to the replication repository and then from the replication repository to the destination cluster.

Direct volume migration

Persistent volumes are copied directly from the source cluster to the destination cluster.

Direct image migration

Images are copied directly from the source cluster to the destination cluster.

Stage migration

Data is copied to the destination cluster without stopping the application.

Running a stage migration multiple times reduces the duration of the cutover migration.

Cutover migration

The application is stopped on the source cluster and its resources are migrated to the destination cluster.

State migration

Application state is migrated by copying specific persistent volume claims to the destination cluster.

Rollback migration

Rollback migration rolls back a completed migration.

---

1 Called the *target* cluster in the MTC web console.

### 1.2. MTC WORKFLOW

You can migrate Kubernetes resources, persistent volume data, and internal container images to OpenShift Container Platform 4.8 by using the Migration Toolkit for Containers (MTC) web console or the Kubernetes API.

MTC migrates the following resources:

- A namespace specified in a migration plan.

- Namespace-scoped resources: When the MTC migrates a namespace, it migrates all the objects and resources associated with that namespace, such as services or pods. Additionally, if a resource that exists in the namespace but not at the cluster level depends on a resource that exists at the cluster level, the MTC migrates both resources.
  
  For example, a security context constraint (SCC) is a resource that exists at the cluster level and a service account (SA) is a resource that exists at the namespace level. If an SA exists in a namespace that the MTC migrates, the MTC automatically locates any SCCs that are linked to...
the SA and also migrates those SCCs. Similarly, the MTC migrates persistent volume claims that are linked to the persistent volumes of the namespace.

**NOTE**

Cluster-scoped resources might have to be migrated manually, depending on the resource.

- Custom resources (CRs) and custom resource definitions (CRDs): MTC automatically migrates CRs and CRDs at the namespace level.

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

1. **Install the Migration Toolkit for Containers Operator on all clusters.**
   You can install the Migration Toolkit for Containers 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 MTC 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 Multi-Cloud Object Gateway (MCG). If you are using a proxy server, you must configure it to allow network traffic between the replication repository and the clusters.

3. **Add the source cluster to the MTC web console.**

4. **Add the replication repository to the MTC web console.**

5. **Create a migration plan, with one of the following data migration options:**
   - **Copy:** MTC copies the data from the source cluster to the replication repository, and from the replication repository to the target cluster.

**NOTE**

If you are using direct image migration or direct volume migration, the images or volumes are copied directly from the source cluster to the target cluster.

- **Move:** MTC 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 migration.

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

- **Stage** copies data to the target cluster without stopping the application. A stage migration can be run multiple times so that most of the data is copied to the target before migration. Running one or more stage migrations reduces the duration of the cutover migration.

- **Cutover** stops the application on the source cluster and moves the resources to the target cluster. Optional: You can clear the **Halt transactions on the source cluster during migration** checkbox.
1.3. ABOUT DATA COPY METHODS

The Migration Toolkit for Containers (MTC) 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.

1.3.1. File system copy method

MTC copies data files from the source cluster to the replication repository, and from there to the target cluster.

The file system copy method uses Restic for indirect migration or Rsync for direct volume migration.

**Table 1.2. File system copy method summary**

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Clusters can have different storage classes.</td>
<td>• Slower than the snapshot copy method.</td>
</tr>
<tr>
<td>• Supported for all S3 storage providers.</td>
<td>• Optional data verification significantly reduces performance.</td>
</tr>
<tr>
<td>• Optional data verification with checksum.</td>
<td></td>
</tr>
<tr>
<td>• Supports direct volume migration, which significantly increases performance.</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

The Restic and Rsync PV migration assumes that the PVs supported are only *volumeMode=filesystem*. Using *volumeMode=Block* for file system migration is *not* supported.

1.3.2. Snapshot copy method

MTC copies a snapshot of the source cluster data to the replication repository of a cloud provider. The data is restored on the target cluster.

The snapshot copy method can be used with Amazon Web Services, Google Cloud Provider, and Microsoft Azure.

**Table 1.3. Snapshot copy method summary**

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1.4. DIRECT VOLUME MIGRATION AND DIRECT IMAGE MIGRATION

You can use direct image migration (DIM) and direct volume migration (DVM) to migrate images and data directly from the source cluster to the target cluster.

If you run DVM with nodes that are in different availability zones, the migration might fail because the migrated pods cannot access the persistent volume claim.

DIM and DVM have significant performance benefits because the intermediate steps of backing up files from the source cluster to the replication repository and restoring files from the replication repository to the target cluster are skipped. The data is transferred with Rsync.

DIM and DVM have additional prerequisites.
CHAPTER 2. MIGRATION TOOLKIT FOR CONTAINERS
RELEASE NOTES

The Migration Toolkit for Containers (MTC) enables you to migrate application workloads between
OpenShift Container Platform clusters at the granularity of a namespace.

You can migrate from OpenShift Container Platform 3 to 4.8 and between OpenShift Container
Platform 4 clusters.

MTC provides a web console and an API, based on Kubernetes custom resources, to help you control the
migration and minimize application downtime.

2.1. MIGRATION TOOLKIT FOR CONTAINERS 1.6 RELEASE NOTES

The release notes for Migration Toolkit for Containers (MTC) version 1.6 describe new features and
enhancements, deprecated features, and known issues.

2.1.1. New features and enhancements

This release has the following new features and enhancements:

- State migration: You can perform repeatable, state-only migrations by selecting specific
  persistent volume claims (PVCs).

- "New operator version available" notification: The Clusters page of the MTC web console
  displays a notification when a new Migration Toolkit for Containers Operator is available.

2.1.2. Deprecated features

The following features are deprecated:

- MTC version 1.4 is no longer supported.

2.1.3. Known issues

This release has the following known issues:

- On OpenShift Container Platform 3.10, the MigrationController pod takes too long to restart.
  The Bugzilla report contains a workaround. (BZ#1986796)

- Stage pods fail during direct volume migration from a classic OpenShift Container Platform
  source cluster on IBM Cloud. The IBM block storage plugin does not allow the same volume to
  be mounted on multiple pods of the same node. As a result, the PVCs cannot be mounted on
  the Rsync pods and on the application pods simultaneously. To resolve this issue, stop the
  application pods before migration. (BZ#1887526)

- MigPlan custom resource does not display a warning when an AWS gp2 PVC has no available
  space. (BZ#1963927)

- Block storage for IBM Cloud must be in the same availability zone. See the IBM FAQ for block
  storage for virtual private cloud.

2.2. MIGRATION TOOLKIT FOR CONTAINERS 1.5 RELEASE NOTES
The release notes for Migration Toolkit for Containers (MTC) version 1.5 describe new features, enhancements, and known issues.

2.2.1. New features and enhancements
This release has the following new features and enhancements:

- The Migration resource tree on the Migration details page of the web console has been enhanced with additional resources, Kubernetes events, and live status information for monitoring and debugging migrations.

- The web console can support hundreds of migration plans.

- A source namespace can be mapped to a different target namespace in a migration plan. Previously, the source namespace was mapped to a target namespace with the same name.

- Hook phases with status information are displayed in the web console during a migration.

- The number of Rsync retry attempts is displayed in the web console during direct volume migration.

- Persistent volume (PV) resizing can be enabled for direct volume migration to ensure that the target cluster does not run out of disk space.

- The threshold that triggers PV resizing is configurable. Previously, PV resizing occurred when the disk usage exceeded 97%.

- Velero has been updated to version 1.6, which provides numerous fixes and enhancements.

- Cached Kubernetes clients can be enabled to provide improved performance.

2.2.2. Deprecated features
The following features are deprecated:

- MTC versions 1.2 and 1.3 are no longer supported.

- The procedure for updating deprecated APIs has been removed from the troubleshooting section of the documentation because the oc convert command is deprecated.

2.2.3. Known issues
This release has the following known issues:

- Microsoft Azure storage is unavailable if you create more than 400 migration plans. The MigStorage custom resource displays the following message: The request is being throttled as the limit has been reached for operation type. (BZ#1977226)

- If a migration fails, the migration plan does not retain custom persistent volume (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)

- PV resizing does not work as expected for AWS gp2 storage unless the pv_resizing_threshold is 42% or greater. (BZ#1973148)
PV resizing does not work with OpenShift Container Platform 3.7 and 3.9 source clusters in the following scenarios:

- The application was installed after MTC was installed.
- An application pod was rescheduled on a different node after MTC was installed.

OpenShift Container Platform 3.7 and 3.9 do not support the Mount Propagation feature that enables Velero to mount PVs automatically in the Restic pod. The MigAnalytic custom resource (CR) fails to collect PV data from the Restic pod and reports the resources as 0. The MigPlan CR displays a status similar to the following:

Example output

```yaml
status:
  conditions:
  - category: Warn
    lastTransitionTime: 2021-07-15T04:11:44Z
    message: Failed gathering extended PV usage information for PVs [nginx-logs nginx-html], please see MigAnalytic openshift-migration/ocp-24706-basicvolmig-migplan-1626319591-szwd6 for details
    reason: FailedRunningDf
    status: "True"
    type: ExtendedPVAnalysisFailed
```

To enable PV resizing, you can manually restart the Restic daemonset on the source cluster or restart the Restic pods on the same nodes as the application. If you do not restart Restic, you can run the direct volume migration without PV resizing. (BZ#1982729)

2.2.4. Technical changes

This release has the following technical changes:

- The legacy Migration Toolkit for Containers Operator version 1.5.1 is installed manually on OpenShift Container Platform versions 3.7 to 4.5.
- The Migration Toolkit for Containers Operator version 1.5.1 is installed on OpenShift Container Platform versions 4.6 and later by using the Operator Lifecycle Manager.

2.3. MIGRATION TOOLKIT FOR CONTAINERS 1.4 RELEASE NOTES

The release notes for Migration Toolkit for Containers (MTC) version 1.4 describe new features, enhancements, and known issues.

2.3.1. Known issues

- MTC 1.4.6 cannot be deployed on OpenShift Container Platform 3.9 (BZ#1981794) or 3.10 (BZ#1981537).
You can install the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4.

**NOTE**

To install MTC on OpenShift Container Platform 3, see *Installing the legacy Migration Toolkit for Containers Operator on OpenShift Container Platform 3*.

By default, the MTC web console and the Migration Controller pod run on the target cluster. You can configure the Migration Controller custom resource manifest to run the MTC web console and the Migration Controller pod on a remote cluster.

After you have installed MTC, you must configure an object storage to use as a replication repository.

To uninstall MTC, see *Uninstalling MTC and deleting resources*.

### 3.1. COMPATIBILITY GUIDELINES

You must install the Migration Toolkit for Containers (MTC) Operator that is compatible with your OpenShift Container Platform version.

**Definitions**

**legacy platform**
- OpenShift Container Platform 4.5 and earlier.

**modern platform**
- OpenShift Container Platform 4.6 and later.

**legacy operator**
- The MTC Operator designed for legacy platforms.

**modern operator**
- The MTC Operator designed for modern platforms.

**control cluster**
- The cluster that runs the MTC controller and GUI.

**remote cluster**
- A source or destination cluster for a migration that runs Velero. The Control Cluster communicates with Remote clusters via the Velero API to drive migrations.

<table>
<thead>
<tr>
<th>Table 3.1. MTC compatibility: Migrating from a legacy platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenShift Container Platform 4.5 or earlier</td>
</tr>
<tr>
<td>OpenShift Container Platform 4.6 or later</td>
</tr>
</tbody>
</table>
3.2. INSTALLING THE LEGACY MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSSHIFT CONTAINER PLATFORM 4.2 TO 4.5

You can install the legacy Migration Toolkit for Containers Operator manually on OpenShift Container Platform versions 4.2 to 4.5.

Prerequisites

- You must be logged in as a user with `cluster-admin` privileges on all clusters.
- You must have access to `registry.redhat.io`.
- You must have `podman` installed.

Procedure

1. Log in to `registry.redhat.io` with your Red Hat Customer Portal credentials:
   ```
   $ sudo podman login registry.redhat.io
   ```

2. Download the `operator.yml` file by entering the following command:
   ```
   -
   ```
3.3. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS
OPERATOR ON OPENSSHIFT CONTAINER PLATFORM 4.8

You install the Migration Toolkit for Containers Operator on OpenShift Container Platform 4.8 by using
the Operator Lifecycle Manager.
Prerequisites

- You must be logged in as a user with `cluster-admin` privileges on all clusters.

Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field to find the **Migration Toolkit for Containers Operator**.
3. Select the **Migration Toolkit for Containers Operator** and click **Install**.
4. Click **Install**.
   On the **Installed Operators** page, the **Migration Toolkit for Containers Operator** appears in the **openshift-migration** project with the status **Succeeded**.
5. Click **Migration Toolkit for Containers Operator**.
6. Under **Provided APIs**, locate the **Migration Controller** tile, and click **Create Instance**.
7. Click **Create**.
8. Click **Workloads → Pods** to verify that the MTC pods are running.

3.4. PROXY CONFIGURATION

For OpenShift Container Platform 4.1 and earlier versions, you must configure proxies in the **MigrationController** custom resource (CR) manifest after you install the Migration Toolkit for Containers Operator because these versions do not support a cluster-wide **proxy** object.

For OpenShift Container Platform 4.2 to 4.8, the Migration Toolkit for Containers (MTC) inherits the cluster-wide proxy settings. You can change the proxy parameters if you want to override the cluster-wide proxy settings.

3.4.1. Direct volume migration

Direct Volume Migration (DVM) was introduced in MTC 1.4.2. DVM supports only one proxy. The source cluster cannot access the route of the target cluster if the target cluster is also behind a proxy.

If you want to perform a DVM from a source cluster behind a proxy, you must configure a TCP proxy that works at the transport layer and forwards the SSL connections transparently without decrypting and re-encrypting them with their own SSL certificates. A Stunnel proxy is an example of such a proxy.

3.4.1.1. TCP proxy setup for DVM

You can set up a direct connection between the source and the target cluster through a TCP proxy and configure the **stunnel_tcp_proxy** variable in the **MigrationController** CR to use the proxy:

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
```
Direct volume migration (DVM) supports only basic authentication for the proxy. Moreover, DVM works only from behind proxies that can tunnel a TCP connection transparently. HTTP/HTTPS proxies in man-in-the-middle mode do not work. The existing cluster-wide proxies might not support this behavior. As a result, the proxy settings for DVM are intentionally kept different from the usual proxy configuration in MTC.

### 3.4.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy?

You can enable DVM by running Rsync between the source and the target cluster over an OpenShift route. Traffic is encrypted using Stunnel, a TCP proxy. The Stunnel running on the source cluster initiates a TLS connection with the target Stunnel and transfers data over an encrypted channel.

Cluster-wide HTTP/HTTPS proxies in OpenShift are usually configured in man-in-the-middle mode where they negotiate their own TLS session with the outside servers. However, this does not work with Stunnel. Stunnel requires that its TLS session be untouched by the proxy, essentially making the proxy a transparent tunnel which simply forwards the TCP connection as-is. Therefore, you must use a TCP proxy.

### 3.4.1.3. Known issue

**Migration fails with error Upgrade request required**

The migration Controller uses the SPDY protocol to execute commands within remote pods. If the remote cluster is behind a proxy or a firewall that does not support the SPDY protocol, the migration controller fails to execute remote commands. The migration fails with the error message *Upgrade request required*. Workaround: Use a proxy that supports the SPDY protocol.

In addition to supporting the SPDY protocol, the proxy or firewall also must pass the *Upgrade* HTTP header to the API server. The client uses this header to open a websocket connection with the API server. If the *Upgrade* header is blocked by the proxy or firewall, the migration fails with the error message *Upgrade request required*. Workaround: Ensure that the proxy forwards the *Upgrade* header.

### 3.4.2. Tuning network policies for migrations

OpenShift supports restricting traffic to or from pods using *NetworkPolicy* or *EgressFirewalls* based on the network plugin used by the cluster. If any of the source namespaces involved in a migration use such mechanisms to restrict network traffic to pods, the restrictions might inadvertently stop traffic to Rsync pods during migration.

Rsync pods running on both the source and the target clusters must connect to each other over an OpenShift Route. Existing *NetworkPolicy* or *EgressNetworkPolicy* objects can be configured to automatically exempt Rsync pods from these traffic restrictions.

### 3.4.2.1. NetworkPolicy configuration

#### 3.4.2.1.1. Egress traffic from Rsync pods

You can use the unique labels of Rsync pods to allow egress traffic to pass from them if the *NetworkPolicy* configuration in the source or destination namespaces blocks this type of traffic. The following policy allows all egress traffic from Rsync pods in the namespace:

```yaml
spec:
  [...]
  stunnel_tcp_proxy: http://username:password@ip:port
```

CHAPTER 3. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS
3.4.2.1.2. Ingress traffic to Rsync pods

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-from-rsync-pods
spec:
podSelector:
  matchLabels:
    owner: directvolumemigration
    app: directvolumemigration-rsync-transfer
  egress:
  - {}
policyTypes:
  - Egress
```

3.4.2.2. EgressNetworkPolicy configuration

The **EgressNetworkPolicy** object or **Egress Firewalls** are OpenShift constructs designed to block egress traffic leaving the cluster.

Unlike the **NetworkPolicy** object, the Egress Firewall works at a project level because it applies to all pods in the namespace. Therefore, the unique labels of Rsync pods do not exempt only Rsync pods from the restrictions. However, you can add the CIDR ranges of the source or target cluster to the Allow rule of the policy so that a direct connection can be setup between two clusters.

Based on which cluster the Egress Firewall is present in, you can add the CIDR range of the other cluster to allow egress traffic between the two:

```yaml
apiVersion: network.openshift.io/v1
kind: EgressNetworkPolicy
metadata:
  name: test-egress-policy
  namespace: <namespace>
spec:
egress:
  - to:
          cidrSelector: <cidr_of_source_or_target_cluster>
type: Deny
```
3.4.2.3. Configuring supplemental groups for Rsync pods

When your PVCs use a shared storage, you can configure the access to that storage by adding supplemental groups to Rsync pod definitions in order for the pods to allow access:

Table 3.2. Supplementary groups for Rsync pods

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for source Rsync pods</td>
</tr>
<tr>
<td>target_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for target Rsync pods</td>
</tr>
</tbody>
</table>

Example usage

The MigrationController CR can be updated to set values for these supplemental groups:

```yaml
spec:
  src_supplemental_groups: "1000,2000"
  target_supplemental_groups: "2000,3000"
```

3.4.3. Configuring proxies

Prerequisites

- You must be logged in as a user with `cluster-admin` privileges on all clusters.

Procedure

1. Get the MigrationController CR manifest:

   ```bash
   $ oc get migrationcontroller <migration_controller> -n openshift-migration
   ```

2. Update the proxy parameters:

   ```yaml
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: <migration_controller>
     namespace: openshift-migration
   ...
   spec:
     stunnel_tcp_proxy: http://<username>:<password>@<ip>:<port> ①
     noProxy: example.com ②
   ```

   ① Stunnel proxy URL for direct volume migration.
   ②
Comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying.

Preface a domain with . to match subdomains only. For example, .y.com matches x.y.com, but not y.com. Use * to bypass proxy for all destinations. If you scale up workers that are not included in the network defined by the networking.machineNetwork[].cidr field from the installation configuration, you must add them to this list to prevent connection issues.

This field is ignored if neither the httpProxy nor the httpsProxy field is set.

3. Save the manifest as migration-controller.yaml.

4. Apply the updated manifest:

```
$ oc replace -f migration-controller.yaml -n openshift-migration
```

For more information, see Configuring the cluster-wide proxy.

### 3.5. CONFIGURING A REPLICATION REPOSITORY

You must configure an object storage to use as a replication repository. The Migration Toolkit for Containers (MTC) copies data from the source cluster to the replication repository, and then from the replication repository to the target cluster.

MTC supports the file system and snapshot data copy methods for migrating data from the source cluster to the target cluster. Select a method that is suited for your environment and is supported by your storage provider.

MTC supports the following storage providers:

- Multi-Cloud Object Gateway (MCG)
- Amazon Web Services (AWS) S3
- Google Cloud Platform (GCP)
- Microsoft Azure Blob
- Generic S3 object storage, for example, Minio or Ceph S3

#### 3.5.1. Prerequisites

- All clusters must have uninterrupted network access to the replication repository.
- If you use a proxy server with an internally hosted replication repository, you must ensure that the proxy allows access to the replication repository.

#### 3.5.2. Configuring Multi-Cloud Object Gateway

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

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

3.5.2.2. 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:
   ```bash
   $ oc login -u <username>
   ```
2. Create the NooBaa CR configuration file, noobaa.yml, with the following content:
   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: NooBaa
   metadata:
     name: <noobaa>
     namespace: openshift-storage
   spec:
     dbResources:
       requests:
         cpu: 0.5
         memory: 1Gi
     coreResources:
       requests:
         cpu: 0.5
         memory: 1Gi
   ```
   1 2 For a very small cluster, you can change the value to 0.1.
3. Create the NooBaa object:
   ```bash
   $ oc create -f noobaa.yml
   ```
4. Create the BackingStore CR configuration file, bs.yml, with the following content:
   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: BackingStore
   ```
Specify the number of volumes in the persistent volume pool.

Specify the size of the volumes, for example, **50Gi**.

Specify the storage class, for example, **gp2**.

5. Create the **BackingStore** object:

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

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

   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: BucketClass
   metadata:
     labels:
       app: noobaa
       name: <mcg_bucket_class>
     namespace: openshift-storage
   spec:
     placementPolicy:
     tiers:
     - backingStores:
       - <mcg_backing_store>
     placement: Spread
   ```

7. Create the **BucketClass** object:

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

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

   ```yaml
   apiVersion: objectbucket.io/v1alpha1
   kind: ObjectBucketClaim
   metadata:
     name: <bucket>
     namespace: openshift-storage
   ```
Record the bucket name for adding the replication repository to the MTC 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 MTC 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 --decode

    - S3 provider secret access key:
      
      $ oc get secret -n openshift-storage migstorage \  
      -o go-template="{{ .data.AWS_SECRET_ACCESS_KEY }}" | base64 --decode

**3.5.3. Configuring Amazon Web Services S3**

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

**Prerequisites**

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

**Procedure**

1. Create an AWS S3 bucket:
   
   ```bash
   $ aws s3api create-bucket \
   --bucket <bucket-name> \\
   --region <bucket-region>
   ```
   
   1. Specify your S3 bucket name.
   2. Specify your S3 bucket region, for example, **us-east-1**.

2. Create the IAM user **velero**:
   
   ```bash
   $ aws iam create-user --user-name velero
   ```

3. Create an EC2 EBS snapshot policy:
   
   ```json
   $ 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": "*"
   }
   ]
   "Resource": "*"
   }
   EOF
   ```

4. Create an AWS S3 access policy for one or for all S3 buckets:
   
   ```json
   $ 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": "*"
   }
   EOF
   ```
To grant access to a single S3 bucket, specify the bucket name. To grant access to all AWS S3 buckets, specify * instead of a bucket name as in the following example:

```
"Resource": [  
  "arn:aws:s3:::<bucket>/*"  
],


```

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

```
"Resource": [  
  "arn:aws:s3:::<bucket>/*"
]
```

(2) Example output

```
"Resource": [  
  "arn:aws:s3:::<bucket>"
]
```

(1) Define the policy for EC2 EBS

(2) Define the policy for AWS S3
Record the `AWS_SECRET_ACCESS_KEY` and the `AWS_ACCESS_KEY_ID`. You use the credentials to add AWS as a replication repository.

3.5.4. Configuring Google Cloud Platform

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

Prerequisites

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

Procedure

1. Log in to `gsutil`:

   ```bash
   $ gsutil init
   ```

   **Example output**

   Welcome! This command will take you through the configuration of gcloud.
   Your current configuration has been set to: [default]
   To continue, you must login. Would you like to login (Y/n)?

2. Set the `BUCKET` variable:

   ```bash
   $ BUCKET=<bucket>
   ```

   Specify your bucket name.

3. Create a storage bucket:

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

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

   ```bash
   $ PROJECT_ID=`gcloud config get-value project`
   ```
5. Create a `velero` IAM service account:

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

6. Create the `SERVICE_ACCOUNT_EMAIL` variable:

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

7. Create the `ROLE_PERMISSIONS` variable:

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

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

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

9. Add IAM policy binding to the project:

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

10. Update the IAM service account:

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

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

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

You use the `credentials-velero` file to add GCP as a replication repository.

### 3.5.5. Configuring Microsoft Azure Blob

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

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

Procedure

1. Set the `AZURE_RESOURCE_GROUP` variable:

   ```bash
   $ AZURE_RESOURCE_GROUP=Velero_Backups
   ```

2. Create an Azure resource group:

   ```bash
   $ az group create -n $AZURE_RESOURCE_GROUP --location <CentralUS>  
   Specify your location.
   ```

3. Set the `AZURE_STORAGE_ACCOUNT_ID` variable:

   ```bash
   $ AZURE_STORAGE_ACCOUNT_ID=velerobackups
   ```

4. Create an Azure storage account:

   ```bash
   $ 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:

   ```bash
   $ BLOB_CONTAINER=velero
   ```

6. Create an Azure Blob storage container:

   ```bash
   $ 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 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
```

You use the credentials-velero file to add Azure as a replication repository.

3.5.6. Additional resources

- MTC workflow
- About data copy methods
- Adding a replication repository to the MTC web console

3.6. UNINSTALLING MTC AND DELETING RESOURCES

You can uninstall the Migration Toolkit for Containers (MTC) and delete its resources to clean up the cluster.

**NOTE**

Deleting the velero CRDs removes Velero from the cluster.

Prerequisites

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

Procedure

1. Delete the MigrationController custom resource (CR) on all clusters:

```
$ oc delete migrationcontroller <migration_controller>
```

2. Uninstall the Migration Toolkit for Containers Operator on OpenShift Container Platform 4 by using the Operator Lifecycle Manager.

3. Delete cluster-scoped resources on all clusters by running the following commands:
- **migration** custom resource definitions (CRDs):
  
  ```bash
  $ oc delete $(oc get crds -o name | grep 'migration.openshift.io')
  ```

- **velero** CRDs:
  
  ```bash
  $ oc delete $(oc get crds -o name | grep 'velero')
  ```

- **migration** cluster roles:
  
  ```bash
  $ oc delete $(oc get clusterroles -o name | grep 'migration.openshift.io')
  ```

- **migration-operator** cluster role:
  
  ```bash
  $ oc delete clusterrole migration-operator
  ```

- **velero** cluster roles:
  
  ```bash
  $ oc delete $(oc get clusterroles -o name | grep 'velero')
  ```

- **migration** cluster role bindings:
  
  ```bash
  $ oc delete $(oc get clusterrolebindings -o name | grep 'migration.openshift.io')
  ```

- **migration-operator** cluster role bindings:
  
  ```bash
  $ oc delete clusterrolebindings migration-operator
  ```

- **velero** cluster role bindings:
  
  ```bash
  $ oc delete $(oc get clusterrolebindings -o name | grep 'velero')
  ```
CHAPTER 4. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS IN A RESTRICTED NETWORK ENVIRONMENT

You can install the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4 in a restricted network environment by performing the following procedures:

1. Create a mirrored Operator catalog. This process creates a `mapping.txt` file, which contains the mapping between the `registry.redhat.io` image and your mirror registry image. The `mapping.txt` file is required for installing the *legacy* Migration Toolkit for Containers Operator on an OpenShift Container Platform 4.2 to 4.5 source cluster.

2. Install the Migration Toolkit for Containers Operator on the OpenShift Container Platform 4.8 target cluster by using Operator Lifecycle Manager. By default, the MTC web console and the *Migration Controller* pod run on the target cluster. You can configure the *Migration Controller* custom resource manifest to run the MTC web console and the *Migration Controller* pod on a remote cluster.

3. Install the Migration Toolkit for Containers Operator on the source cluster:
   - OpenShift Container Platform 4.6 or later: Install the Migration Toolkit for Containers Operator by using Operator Lifecycle Manager.
   - OpenShift Container Platform 4.2 to 4.5: Install the legacy Migration Toolkit for Containers Operator from the command line interface.

4. Configure object storage to use as a replication repository.

**NOTE**

To install MTC on OpenShift Container Platform 3, see Installing the legacy Migration Toolkit for Containers Operator on OpenShift Container Platform 3.

To uninstall MTC, see Uninstalling MTC and deleting resources.

4.1. COMPATIBILITY GUIDELINES

You must install the Migration Toolkit for Containers (MTC) Operator that is compatible with your OpenShift Container Platform version.

**Definitions**

- **legacy platform**
  - OpenShift Container Platform 4.5 and earlier.

- **modern platform**
  - OpenShift Container Platform 4.6 and later.

- **legacy operator**
  - The MTC Operator designed for legacy platforms.

- **modern operator**
  - The MTC Operator designed for modern platforms.

- **control cluster**
The cluster that runs the MTC controller and GUI.

**remote cluster**

A source or destination cluster for a migration that runs Velero. The Control Cluster communicates with Remote clusters via the Velero API to drive migrations.

### Table 4.1. MTC compatibility: Migrating from a legacy platform

<table>
<thead>
<tr>
<th></th>
<th>OpenShift Container Platform 4.5 or earlier</th>
<th>OpenShift Container Platform 4.6 or later</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable MTC version</td>
<td>MTC 1.7. z</td>
<td>MTC 1.7. z</td>
</tr>
<tr>
<td></td>
<td>Legacy 1.7 operator: Install manually with the <strong>operator.yml</strong> file.</td>
<td>Install with OLM, release channel <strong>release-v1.7</strong></td>
</tr>
</tbody>
</table>

**IMPORTANT**

This cluster cannot be the control cluster.

**NOTE**

Edge cases exist in which network restrictions prevent modern clusters from connecting to other clusters involved in the migration. For example, when migrating from an OpenShift Container Platform 3.11 cluster on premises to a modern OpenShift Container Platform cluster in the cloud, where the modern cluster cannot connect to the OpenShift Container Platform 3.11 cluster.

With MTC 1.7, if one of the remote clusters is unable to communicate with the control cluster because of network restrictions, use the **crane tunnel-api** command.

With the stable MTC release, although you should always designate the most modern cluster as the control cluster, in this specific case it is possible to designate the legacy cluster as the control cluster and push workloads to the remote cluster.

### 4.2. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSSHIFT CONTAINER PLATFORM 4.8

You install the Migration Toolkit for Containers Operator on OpenShift Container Platform 4.8 by using the Operator Lifecycle Manager.

**Prerequisites**

- You must be logged in as a user with **cluster-admin** privileges on all clusters.
- You must create an Operator catalog from a mirror image in a local registry.

**Procedure**

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.
2. Use the **Filter by keyword** field to find the **Migration Toolkit for Containers Operator**.
3. Select the Migration Toolkit for Containers Operator and click Install.

4. Click Install.
   On the Installed Operators page, the Migration Toolkit for Containers Operator appears in the openshift-migration project with the status Succeeded.

5. Click Migration Toolkit for Containers Operator.

6. Under Provided APIs, locate the Migration Controller tile, and click Create Instance.

7. Click Create.

8. Click Workloads → Pods to verify that the MTC pods are running.

**4.3. INSTALLING THE LEGACY MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENShift CONTAINER PLATFORM 4.2 TO 4.5**

You can install the legacy Migration Toolkit for Containers Operator manually on OpenShift Container Platform versions 4.2 to 4.5.

**Prerequisites**

- You must be logged in as a user with cluster-admin privileges on all clusters.
- You must have access to registry.redhat.io.
- You must have podman installed.
- You must have a Linux workstation with network access in order to download files from registry.redhat.io.
- You must create a mirror image of the Operator catalog.
- You must install the Migration Toolkit for Containers Operator from the mirrored Operator catalog on OpenShift Container Platform 4.8.

**Procedure**

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

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

2. Download the operator.yml file by entering the following command:

   ```bash
   $ sudo podman cp $(sudo podman create \ registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-operator:v1.7):/operator.yml ./
   ```

3. Download the controller.yml file by entering the following command:

   ```bash
   $ sudo podman cp $(sudo podman create \ registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-operator:v1.7):/controller.yml ./
   ```

4. Obtain the Operator image mapping by running the following command:
The **mapping.txt** file was created when you mirrored the Operator catalog. The output shows the mapping between the **registry.redhat.io** image and your mirror registry image.

### Example output

```
registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-operator@sha256:468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a =<registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-operator
```

5. Update the **image** values for the **ansible** and **operator** containers and the **REGISTRY** value in the **operator.yml** file:

```
containers:
  - name: ansible
    image: <registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-operator@sha256:
      <468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a>
    ...
  - name: operator
    image: <registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-operator@sha256:
      <468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a>
    ...
    env:
      - name: REGISTRY
        value: <registry.apps.example.com>
```

1. Specify your mirror registry and the **sha256** value of the Operator image.
2. Specify your mirror registry.
3. 

6. Log in to your source cluster.

7. Create the Migration Toolkit for Containers Operator object:

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

### Example output

```
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":
```
You can ignore Error from server (AlreadyExists) messages. They are caused by the Migration Toolkit for Containers Operator creating resources for earlier versions of OpenShift Container Platform 4 that are provided in later releases.

8. Create the MigrationController object:

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

9. Verify that the MTC pods are running:

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

### 4.4. PROXY CONFIGURATION

For OpenShift Container Platform 4.1 and earlier versions, you must configure proxies in the MigrationController custom resource (CR) manifest after you install the Migration Toolkit for Containers Operator because these versions do not support a cluster-wide proxy object.

For OpenShift Container Platform 4.2 to 4.8, the Migration Toolkit for Containers (MTC) inherits the cluster-wide proxy settings. You can change the proxy parameters if you want to override the cluster-wide proxy settings.

#### 4.4.1. Direct volume migration

Direct Volume Migration (DVM) was introduced in MTC 1.4.2. DVM supports only one proxy. The source cluster cannot access the route of the target cluster if the target cluster is also behind a proxy.

If you want to perform a DVM from a source cluster behind a proxy, you must configure a TCP proxy that works at the transport layer and forwards the SSL connections transparently without decrypting and re-encrypting them with their own SSL certificates. A Stunnel proxy is an example of such a proxy.

#### 4.4.1.1. TCP proxy setup for DVM

You can set up a direct connection between the source and the target cluster through a TCP proxy and configure the stunnel_tcp_proxy variable in the MigrationController CR to use the proxy:

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  ...
  stunnel_tcp_proxy: http://username:password@ip:port
```

Direct volume migration (DVM) supports only basic authentication for the proxy. Moreover, DVM works only from behind proxies that can tunnel a TCP connection transparently. HTTP/HTTPS proxies in man-in-the-middle mode do not work. The existing cluster-wide proxies might not support this behavior. As a
result, the proxy settings for DVM are intentionally kept different from the usual proxy configuration in MTC.

4.4.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy?

You can enable DVM by running Rsync between the source and the target cluster over an OpenShift route. Traffic is encrypted using Stunnel, a TCP proxy. The Stunnel running on the source cluster initiates a TLS connection with the target Stunnel and transfers data over an encrypted channel.

Cluster-wide HTTP/HTTPS proxies in OpenShift are usually configured in man-in-the-middle mode where they negotiate their own TLS session with the outside servers. However, this does not work with Stunnel. Stunnel requires that its TLS session be untouched by the proxy, essentially making the proxy a transparent tunnel which simply forwards the TCP connection as-is. Therefore, you must use a TCP proxy.

4.4.1.3. Known issue

Migration fails with error Upgrade request required

The migration Controller uses the SPDY protocol to execute commands within remote pods. If the remote cluster is behind a proxy or a firewall that does not support the SPDY protocol, the migration controller fails to execute remote commands. The migration fails with the error message Upgrade request required. Workaround: Use a proxy that supports the SPDY protocol.

In addition to supporting the SPDY protocol, the proxy or firewall also must pass the Upgrade HTTP header to the API server. The client uses this header to open a websocket connection with the API server. If the Upgrade header is blocked by the proxy or firewall, the migration fails with the error message Upgrade request required. Workaround: Ensure that the proxy forwards the Upgrade header.

4.4.2. Tuning network policies for migrations

OpenShift supports restricting traffic to or from pods using NetworkPolicy or EgressNetworkPolicy based on the network plugin used by the cluster. If any of the source namespaces involved in a migration use such mechanisms to restrict network traffic to pods, the restrictions might inadvertently stop traffic to Rsync pods during migration.

Rsync pods running on both the source and the target clusters must connect to each other over an OpenShift Route. Existing NetworkPolicy or EgressNetworkPolicy objects can be configured to automatically exempt Rsync pods from these traffic restrictions.

4.4.2.1. NetworkPolicy configuration

4.4.2.1.1. Egress traffic from Rsync pods

You can use the unique labels of Rsync pods to allow egress traffic to pass from them if the NetworkPolicy configuration in the source or destination namespaces blocks this type of traffic. The following policy allows all egress traffic from Rsync pods in the namespace:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-from-rsync-pods
spec:
  podSelector:
    matchLabels:
```
4.4.2.1.2. Ingress traffic to Rsync pods

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-from-rsync-pods
spec:
  podSelector:
    matchLabels:
      owner: directvolumemigration
      app: directvolumemigration-rsync-transfer
ingress:
  - {}
policyTypes:
  - Ingress

4.4.2.2. EgressNetworkPolicy configuration

The EgressNetworkPolicy object or Egress Firewalls are OpenShift constructs designed to block egress traffic leaving the cluster.

Unlike the NetworkPolicy object, the Egress Firewall works at a project level because it applies to all pods in the namespace. Therefore, the unique labels of Rsync pods do not exempt only Rsync pods from the restrictions. However, you can add the CIDR ranges of the source or target cluster to the Allow rule of the policy so that a direct connection can be setup between two clusters.

Based on which cluster the Egress Firewall is present in, you can add the CIDR range of the other cluster to allow egress traffic between the two:

apiVersion: network.openshift.io/v1
kind: EgressNetworkPolicy
metadata:
  name: test-egress-policy
  namespace: <namespace>
spec:
  egress:
  - to:
    cidrSelector: <cidr_of_source_or_target_cluster>
type: Deny

4.4.2.3. Configuring supplemental groups for Rsync pods

When your PVCs use a shared storage, you can configure the access to that storage by adding supplemental groups to Rsync pod definitions in order for the pods to allow access:

Table 4.2. Supplementary groups for Rsync pods
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for source Rsync pods</td>
</tr>
<tr>
<td>target_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for target Rsync pods</td>
</tr>
</tbody>
</table>

**Example usage**

The **MigrationController** CR can be updated to set values for these supplemental groups:

```yaml
spec:
  src_supplemental_groups: "1000,2000"
  target_supplemental_groups: "2000,3000"
```

**4.4.3. Configuring proxies**

**Prerequisites**

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

**Procedure**

1. Get the **MigrationController** CR manifest:

   ```bash
   $ oc get migrationcontroller <migration_controller> -n openshift-migration
   ```

2. Update the proxy parameters:

   ```yaml
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: <migration_controller>
     namespace: openshift-migration
   ...
   spec:
     stunnel_tcp_proxy: http://<username>:<password>@<ip>:<port> ¹
     noProxy: example.com ²
   ```

1. Stunnel proxy URL for direct volume migration.

2. Comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying.

Preface a domain with . to match subdomains only. For example, .y.com matches x.y.com, but not y.com. Use * to bypass proxy for all destinations. If you scale up workers that are not included in the network defined by the networking.machineNetwork[].cidr field from the installation configuration, you must add them to this list to prevent connection issues.
This field is ignored if neither the `httpProxy` nor the `httpsProxy` field is set.

3. Save the manifest as `migration-controller.yaml`.

4. Apply the updated manifest:

   ```bash
   $ oc replace -f migration-controller.yaml -n openshift-migration
   ```

For more information, see Configuring the cluster-wide proxy.

### 4.5. Configuring a Replication Repository

You must configure an object storage to use as a replication repository. The Migration Toolkit for Containers (MTC) copies data from the source cluster to the replication repository, and then from the replication repository to the target cluster. Multi-Cloud Object Gateway (MCG) is the only supported option for a restricted network environment.

MTC 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.

#### 4.5.1. Prerequisites

- All clusters must have uninterrupted network access to the replication repository.

- If you use a proxy server with an internally hosted replication repository, you must ensure that the proxy allows access to the replication repository.

#### 4.5.2. Configuring Multi-Cloud Object Gateway

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

##### 4.5.2.1. 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 **Install**.

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

##### 4.5.2.2. 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:

   ```bash
   $ oc login -u <username>
   ```

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

   ```yaml
   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 value to **0.1**.

3. Create the **NooBaa** object:

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

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

   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: BackingStore
   metadata:
     finalizers:
     - noobaa.io/finalizer
     labels:
       app: noobaa
     name: <mcg_backing_store>
     namespace: openshift-storage
   spec:
     pvPool:
       numVolumes: 3 1
       resources:
       requests:
       storage: <volume_size> 2
       storageClass: <storage_class> 3
       type: pv-pool
   ```

   1 Specify the number of volumes in the persistent volume pool.

   2 Specify the size of the volumes, for example, **50Gi**.
Specify the storage class, for example, gp2.

5. Create the **BackingStore** object:

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

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

   ```yaml
   apiVersion: noobaa.io/v1alpha1
   kind: BucketClass
   metadata:
     app: noobaa
     name: <mcg_bucket_class>
     namespace: openshift-storage
   spec:
     placementPolicy:
       tiers:
         - backingStores:
             - <mcg_backing_store>
     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:

   ```yaml
   apiVersion: objectbucket.io/v1alpha1
   kind: ObjectBucketClaim
   metadata:
     name: <bucket>
     namespace: openshift-storage
   spec:
     bucketName: <bucket>  
     storageClassName: <storage_class>
     additionalConfig:
       bucketclass: <mcg_bucket_class>
   ```

   1. Record the bucket name for adding the replication repository to the MTC 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.
II. Obtain and record the following values, which are required when you add the replication repository to the MTC 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 --decode
  ```

- S3 provider secret access key:
  ```
  $ oc get secret -n openshift-storage migstorage \
  -o go-template='{{ .data.AWS_SECRET_ACCESS_KEY }}' | base64 --decode
  ```

4.5.3. Additional resources

- **Disconnected environment** in the Red Hat OpenShift Container Storage documentation.
- **MTC workflow**
- **About data copy methods**
- **Adding a replication repository to the MTC web console**

### 4.6. UNINSTALLING MTC AND DELETING RESOURCES

You can uninstall the Migration Toolkit for Containers (MTC) and delete its resources to clean up the cluster.

**NOTE**

Deleting the **velero** CRDs removes Velero from the cluster.

**Prerequisites**

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

**Procedure**

1. Delete the **MigrationController** custom resource (CR) on all clusters:
   ```
   $ oc delete migrationcontroller <migration_controller>
   ```

2. Uninstall the Migration Toolkit for Containers Operator on OpenShift Container Platform 4 by using the Operator Lifecycle Manager.

3. Delete cluster-scoped resources on all clusters by running the following commands:
   ```
   $ oc delete $(oc get crds -o name | grep 'migration.openshift.io')
   ```
- velero CRDs:
  $ oc delete $(oc get crds -o name | grep 'velero')

- migration cluster roles:
  $ oc delete $(oc get clusterroles -o name | grep 'migration.openshift.io')

- migration-operator cluster role:
  $ oc delete clusterrole migration-operator

- velero cluster roles:
  $ oc delete $(oc get clusterroles -o name | grep 'velero')

- migration cluster role bindings:
  $ oc delete $(oc get clusterrolebindings -o name | grep 'migration.openshift.io')

- migration-operator cluster role bindings:
  $ oc delete clusterrolebindings migration-operator

- velero cluster role bindings:
  $ oc delete $(oc get clusterrolebindings -o name | grep 'velero')
CHAPTER 5. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS

You can upgrade the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4.8 by using Operator Lifecycle Manager.

You can upgrade MTC on OpenShift Container Platform 4.5, and earlier versions, by reinstalling the legacy Migration Toolkit for Containers Operator.

IMPORTANT

If you are upgrading from MTC version 1.3, you must perform an additional procedure to update the MigPlan custom resource (CR).

5.1. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSSHIFT CONTAINER PLATFORM 4.8

You can upgrade the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4.8 by using the Operator Lifecycle Manager.

Prerequisites

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

Procedure

1. In the OpenShift Container Platform console, navigate to Operators → Installed Operators. Operators that have a pending upgrade display an Upgrade available status.

2. Click Migration Toolkit for Containers Operator.

3. Click the Subscription tab. Any upgrades requiring approval are displayed next to Upgrade Status. For example, it might display 1 requires approval.

4. Click 1 requires approval, then click Preview Install Plan.

5. Review the resources that are listed as available for upgrade and click Approve.

6. Navigate back to the Operators → Installed Operators page to monitor the progress of the upgrade. When complete, the status changes to Succeeded and Up to date.

7. Click Workloads → Pods to verify that the MTC pods are running.

5.2. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSHEET CONTAINER PLATFORM VERSIONS 4.2 TO 4.5

You can upgrade Migration Toolkit for Containers (MTC) on OpenShift Container Platform versions 4.2 to 4.5 by manually installing the legacy Migration Toolkit for Containers Operator.

Prerequisites

- You must be logged in as a user with cluster-admin privileges.
You must have access to registry.redhat.io.

You must have podman installed.

Procedure

1. Log in to registry.redhat.io with your Red Hat Customer Portal credentials by entering the following command:
   
   $ sudo podman login registry.redhat.io

   Download the operator.yml file by entering the following command:

   $ sudo podman cp $(sudo podman create \
   registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-operator:v1.7):/operator.yml ./

   1. Replace the Migration Toolkit for Containers Operator by entering the following command:

   $ oc replace --force -f operator.yml

   2. Scale the migration-operator deployment to 0 to stop the deployment by entering the following command:

   $ oc scale -n openshift-migration --replicas=0 deployment/migration-operator

   3. Scale the migration-operator deployment to 1 to start the deployment and apply the changes by entering the following command:

   $ oc scale -n openshift-migration --replicas=1 deployment/migration-operator

   4. Verify that the migration-operator was upgraded by entering the following command:

   $ oc -o yaml -n openshift-migration get deployment/migration-operator | grep image: | awk -F ":" '{ print $NF }'

   5. Download the controller.yml file by entering the following command:

   $ sudo podman cp $(sudo podman create \ 
   registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-operator:v1.7):/controller.yml ./

   6. Create the migration-controller object by entering the following command:

   $ oc create -f controller.yml

   7. Verify that the MTC pods are running by entering the following command:

   $ oc get pods -n openshift-migration

5.3. UPGRADING MTC 1.3 TO 1.7
If you are upgrading Migration Toolkit for Containers (MTC) version 1.3.x to 1.7, you must update the MigPlan custom resource (CR) manifest on the cluster on which the MigrationController pod is running.

Because the indirectImageMigration and indirectVolumeMigration parameters do not exist in MTC 1.3, their default value in version 1.4 is false, which means that direct image migration and direct volume migration are enabled. Because the direct migration requirements are not fulfilled, the migration plan cannot reach a Ready state unless these parameter values are changed to true.

Prerequisites

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

Procedure

1. Log in to the cluster on which the MigrationController pod is running.

2. Get the MigPlan CR manifest:

   ```bash
   $ oc get migplan <migplan> -o yaml -n openshift-migration
   ...
   spec:
   indirectImageMigration: true
   indirectVolumeMigration: true
   ```

3. Update the following parameter values and save the file as migplan.yaml:

4. Replace the MigPlan CR manifest to apply the changes:

   ```bash
   $ oc replace -f migplan.yaml -n openshift-migration
   ...
   spec:
   indirectImageMigration: true
   indirectVolumeMigration: true
   ```

5. Get the updated MigPlan CR manifest to verify the changes:

   ```bash
   $ oc get migplan <migplan> -o yaml -n openshift-migration
   ```
CHAPTER 6. PREMIGRATION CHECKLISTS

Before you migrate your application workloads with the Migration Toolkit for Containers (MTC), review the following checklists.

6.1. CLUSTER HEALTH CHECKLIST

☐ The clusters meet the minimum hardware requirements for the specific platform and installation method, for example, on bare metal.

☐ All MTC prerequisites are met.

☐ All nodes have an active OpenShift Container Platform subscription.

☐ You have verified node health.

☐ The identity provider is working.

☐ The migration network has a minimum throughput of 10 Gbps.

☐ The clusters have sufficient resources for migration.

**NOTE**

Clusters require additional memory, CPUs, and storage in order to run a migration on top of normal workloads. Actual resource requirements depend on the number of Kubernetes resources being migrated in a single migration plan. You must test migrations in a non-production environment in order to estimate the resource requirements.

☐ The etcd disk performance of the clusters has been checked with fio.

6.2. SOURCE CLUSTER CHECKLIST

☐ You have checked for persistent volumes (PVs) with abnormal configurations stuck in a Terminating state by running the following command:

```
$ oc get pv
```

☐ You have checked for pods whose status is other than Running or Completed by running the following command:

```
$ oc get pods --all-namespaces | egrep -v 'Running | Completed'
```

☐ You have checked for pods with a high restart count by running the following command:

```
$ oc get pods --all-namespaces --field-selector=status.phase=Running \ 
-0 json | jq '.items[]|select(any(.status.containerStatuses[]; \ 
.restartCount > 3))|.metadata.name'
```

Even if the pods are in a Running state, a high restart count might indicate underlying problems.

☐ The cluster certificates are valid for the duration of the migration process.
☐ You have checked for pending certificate-signing requests by running the following command:

```
$ oc get csr -A | grep pending -i
```

☐ The registry uses a recommended storage type.

☐ You can read and write images to the registry.

☐ The etcd cluster is healthy.

☐ The average API server response time on the source cluster is less than 50 ms.

### 6.3. TARGET CLUSTER CHECKLIST

☐ The cluster has the correct network configuration and permissions to access external services, for example, databases, source code repositories, container image registries, and CI/CD tools.

☐ External applications and services that use services provided by the cluster have the correct network configuration and permissions to access the cluster.

☐ Internal container image dependencies are met.

☐ The target cluster and the replication repository have sufficient storage space.
CHAPTER 7. NETWORK CONSIDERATIONS

Review the strategies for redirecting your application network traffic after migration.

7.1. DNS CONSIDERATIONS

The DNS domain of the target cluster is different from the domain of the source cluster. By default, applications get FQDNs of the target cluster after migration.

To preserve the source DNS domain of migrated applications, select one of the two options described below.

7.1.1. Isolating the DNS domain of the target cluster from the clients

You can allow the clients' requests sent to the DNS domain of the source cluster to reach the DNS domain of the target cluster without exposing the target cluster to the clients.

Procedure

1. Place an exterior network component, such as an application load balancer or a reverse proxy, between the clients and the target cluster.
2. Update the application FQDN on the source cluster in the DNS server to return the IP address of the exterior network component.
3. Configure the network component to send requests received for the application in the source domain to the load balancer in the target cluster domain.
4. Create a wildcard DNS record for the *.apps.source.example.com domain that points to the IP address of the load balancer of the source cluster.
5. Create a DNS record for each application that points to the IP address of the exterior network component in front of the target cluster. A specific DNS record has higher priority than a wildcard record, so no conflict arises when the application FQDN is resolved.

NOTE

- The exterior network component must terminate all secure TLS connections. If the connections pass through to the target cluster load balancer, the FQDN of the target application is exposed to the client and certificate errors occur.
- The applications must not return links referencing the target cluster domain to the clients. Otherwise, parts of the application might not load or work properly.

7.1.2. Setting up the target cluster to accept the source DNS domain

You can set up the target cluster to accept requests for a migrated application in the DNS domain of the source cluster.

Procedure

For both non-secure HTTP access and secure HTTPS access, perform the following steps:

1. Create a route in the target cluster’s project that is configured to accept requests addressed to the application’s FQDN in the source cluster:
$ oc expose svc <app1-svc> --hostname <app1.apps.source.example.com> -n <app1-namespace>

With this new route in place, the server accepts any request for that FQDN and sends it to the corresponding application pods. In addition, when you migrate the application, another route is created in the target cluster domain. Requests reach the migrated application using either of these hostnames.

2. Create a DNS record with your DNS provider that points the application’s FQDN in the source cluster to the IP address of the default load balancer of the target cluster. This will redirect traffic away from your source cluster to your target cluster.

The FQDN of the application resolves to the load balancer of the target cluster. The default ingress controller router accept requests for that FQDN because a route for that hostname is exposed.

For secure HTTPS access, perform the following additional step:

1. Replace the x509 certificate of the default ingress controller created during the installation process with a custom certificate.

2. Configure this certificate to include the wildcard DNS domains for both the source and target clusters in the subjectAltName field.

The new certificate is valid for securing connections made using either DNS domain.

Additional resources

- See Replacing the default ingress certificate for more information.

7.2. NETWORK TRAFFIC REDIRECTION STRATEGIES

After a successful migration, you must redirect network traffic of your stateless applications from the source cluster to the target cluster.

The strategies for redirecting network traffic are based on the following assumptions:

- The application pods are running on both the source and target clusters.
- Each application has a route that contains the source cluster hostname.
- The route with the source cluster hostname contains a CA certificate.
- For HTTPS, the target router CA certificate contains a Subject Alternative Name for the wildcard DNS record of the source cluster.

Consider the following strategies and select the one that meets your objectives.

- Redirecting all network traffic for all applications at the same time
  Change the wildcard DNS record of the source cluster to point to the target cluster router’s virtual IP address (VIP).
  
  This strategy is suitable for simple applications or small migrations.

- Redirecting network traffic for individual applications
  Create a DNS record for each application with the source cluster hostname pointing to the target cluster router’s VIP. This DNS record takes precedence over the source cluster wildcard DNS record.
• Redirecting network traffic gradually for individual applications
  1. Create a proxy that can direct traffic to both the source cluster router’s VIP and the target cluster router’s VIP, for each application.
  2. Create a DNS record for each application with the source cluster hostname pointing to the proxy.
  3. Configure the proxy entry for the application to route a percentage of the traffic to the target cluster router’s VIP and the rest of the traffic to the source cluster router’s VIP.
  4. Gradually increase the percentage of traffic that you route to the target cluster router’s VIP until all the network traffic is redirected.

• User-based redirection of traffic for individual applications
  Using this strategy, you can filter TCP/IP headers of user requests to redirect network traffic for predefined groups of users. This allows you to test the redirection process on specific populations of users before redirecting the entire network traffic.
  1. Create a proxy that can direct traffic to both the source cluster router’s VIP and the target cluster router’s VIP, for each application.
  2. Create a DNS record for each application with the source cluster hostname pointing to the proxy.
  3. Configure the proxy entry for the application to route traffic matching a given header pattern, such as *test customers*, to the target cluster router’s VIP and the rest of the traffic to the source cluster router’s VIP.
  4. Redirect traffic to the target cluster router’s VIP in stages until all the traffic is on the target cluster router’s VIP.
CHAPTER 8. MIGRATING YOUR APPLICATIONS

You can migrate your applications by using the Migration Toolkit for Containers (MTC) web console or the command line.

Most cluster-scoped resources are not yet handled by MTC. If your applications require cluster-scoped resources, you might have to create them manually on the target cluster.

You can use stage migration and cutover migration to migrate an application between clusters:

- Stage migration copies data from the source cluster to the target cluster without stopping the application. You can run a stage migration multiple times to reduce the duration of the cutover migration.

- Cutover migration stops the transactions on the source cluster and moves the resources to the target cluster.

You can use state migration to migrate an application’s state:

- State migration copies selected persistent volume claims (PVCs).

- You can use state migration to migrate a namespace within the same cluster.

During migration, the Migration Toolkit for Containers (MTC) 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.

8.1. MIGRATION PREREQUISITES

- You must be logged in as a user with `cluster-admin` privileges on all clusters.

Direct image migration

- You must ensure that the secure internal registry of the source cluster is exposed.

- You must create a route to the exposed registry.

Direct volume migration

- If your clusters use proxies, you must configure an Stunnel TCP proxy.

Clusters

- The source cluster must be upgraded to the latest MTC z-stream release.

- The MTC version must be the same on all clusters.

Network
The clusters have unrestricted network access to each other and to the replication repository.

If you copy the persistent volumes with move, the clusters must have unrestricted network access to the remote volumes.

You must enable the following ports on an OpenShift Container Platform 4 cluster:
- 6443 (API server)
- 443 (routes)
- 53 (DNS)

You must enable port 443 on the replication repository if you are using TLS.

Persistent volumes (PVs)
- The PVs must be valid.
- The PVs must be bound to persistent volume claims.
- If you use snapshots to copy the PVs, the following additional prerequisites apply:
  - The cloud provider must support snapshots.
  - The PVs must have the same cloud provider.
  - The PVs must be located in the same geographic region.
  - The PVs must have the same storage class.

8.2. MIGRATING YOUR APPLICATIONS BY USING THE MTC WEB CONSOLE

You can configure clusters and a replication repository by using the MTC web console. Then, you can create and run a migration plan.

8.2.1. Launching the MTC web console

You can launch the Migration Toolkit for Containers (MTC) web console in a browser.

Prerequisites
- The MTC web console must have network access to the OpenShift Container Platform web console.
- The MTC web console must have network access to the OAuth authorization server.

Procedure
1. Log in to the OpenShift Container Platform cluster on which you have installed MTC.
2. Obtain the MTC web console URL by entering the following command:
3. Launch a browser and navigate to the MTC web console.

NOTE
If you try to access the MTC web console immediately after installing the Migration Toolkit for Containers Operator, the console might 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 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**.

8.2.2. Adding a cluster to the MTC web console
You can add a cluster to the Migration Toolkit for Containers (MTC) web console.

**Prerequisites**
- If you are using Azure snapshots to copy data:
  - You must specify the Azure resource group name for the cluster.
  - The clusters must be in the same Azure resource group.
  - The clusters must be in the same geographic location.
- If you are using direct image migration, you must expose a route to

**Procedure**
1. Log in to the cluster.

2. Obtain the **migration-controller** service account token:

   ```
   $ oc sa get-token migration-controller -n openshift-migration
   ```

**Example output**

```bash
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJwYXV0b2d8c2VydmljZS5uYW1lIjoiY2F0YWJsZSIiLCJzdWIiOiJjcmVhdmluZ3MiLCJpYXQiOjE1NjEyNzY3MDU4fQ.SZ_sU5OJtYDw3H_q1C-0IDcA1v-wTzWfPd-iQJ-ZHJK
```

OpenShift Container Platform 4.8 Migration Toolkit for Containers

56
3. In the MTC web console, click `Clusters`.

4. Click `Add cluster`.

5. Fill in the following fields:
   - **Cluster name**: The cluster name can contain lower-case letters (a-z) and numbers (0-9). It must not contain spaces or international characters.
   - **URL**: Specify the API server URL, for example, `https://<www.example.com>:8443`.
   - **Service account token**: Paste the `migration-controller` service account token.
   - **Exposed route host to image registry**: If you are using direct image migration, specify the exposed route to the image registry of the source cluster. To create the route, run the following command:
     - For OpenShift Container Platform 3:
       ```bash
       $ oc create route passthrough --service=docker-registry --port=5000 -n default
     ```
     - For OpenShift Container Platform 4:
       ```bash
       $ oc create route passthrough --service=image-registry --port=5000 -n openshift-image-registry
     ```
   - **Azure cluster**: You must select this option if you use Azure snapshots to copy your data.
   - **Azure resource group**: This field is displayed if Azure cluster is selected. Specify the Azure resource group.
   - **Require SSL verification**: Optional: Select this option to verify SSL connections to the cluster.
   - **CA bundle file**: This field is displayed if Require SSL verification is selected. If you created a custom CA certificate bundle file for self-signed certificates, click `Browse`, select the CA bundle file, and upload it.

6. Click `Add cluster`.
   The cluster appears in the `Clusters` list.

### 8.2.3. Adding a replication repository to the MTC web console

You can add an object storage as a replication repository to the Migration Toolkit for Containers (MTC) web console.

MTC supports the following storage providers:

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

**Prerequisites**

- You must configure the object storage as a replication repository.

**Procedure**

1. In the MTC web console, click *Replication repositories*.
2. Click *Add repository*.
3. Select a *Storage provider type* and fill in the following fields:
   - **AWS** for S3 providers, including AWS and MCG:
     - *Replication repository name*: Specify the replication repository name in the MTC web console.
     - *S3 bucket name*: Specify the name of the S3 bucket.
     - *S3 bucket region*: Specify the S3 bucket region. **Required** for AWS S3. **Optional** for some S3 providers. Check the product documentation of your S3 provider for expected values.
     - *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 and other S3 providers.
     - *S3 provider secret access key*: Specify the `<AWS_ACCESS_KEY_ID>` for AWS or the S3 provider secret access key for MCG and other S3 providers.
     - *require SSL verification*: Clear this checkbox if you are using a generic S3 provider.
     - If you created a custom CA certificate bundle for self-signed certificates, click *Browse* and browse to the Base64-encoded file.
   - **GCP**:
     - *Replication repository name*: Specify the replication repository name in the MTC 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 MTC 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** list.

### 8.2.4. Creating a migration plan in the MTC web console

You can create a migration plan in the Migration Toolkit for Containers (MTC) web console.

**Prerequisites**

- You must be logged in as a user with **cluster-admin** privileges on all clusters.
- You must ensure that the same MTC version is installed on all clusters.
- You must add the clusters and the replication repository to the MTC web console.
- If you want to use the **move** data copy method to migrate a persistent volume (PV), the source and target clusters must have uninterrupted network access to the remote volume.
- If you want to use direct image migration, you must specify the exposed route to the image registry of the source cluster. This can be done by using the MTC web console or by updating the **MigCluster** custom resource manifest.

**Procedure**

1. In the MTC web console, click **Migration plans**.

2. Click **Add migration plan**.

3. Enter the **Plan name**. The migration plan name must not exceed 253 lower-case alphanumeric characters (a-z, 0-9) and must not contain spaces or underscores (_).

4. Select a **Source cluster**, a **Target cluster**, and a **Repository**.

5. Click **Next**.

6. Select the projects for migration.

7. Optional: Click the edit icon beside a project to change the target namespace.

8. Click **Next**.

9. Select a **Migration type** for each PV:
   - **Copy** option copies the data from the PV of a source cluster to the replication repository and then restores the data on a newly created PV, with similar characteristics, in the target cluster.
   - **Move** option 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.

10. Click **Next**.

11. Select a **Copy method** for each PV:

   - **Snapshot copy** backs up and restores data using the cloud provider’s snapshot functionality. It is significantly faster than **Filesystem copy**.
   
   - **Filesystem copy** backs up the files on the source cluster and restores them on the target cluster.
     The file system copy method is required for direct volume migration.

12. You can select **Verify copy** to verify data migrated with **Filesystem copy**. Data is verified by generating a checksum for each source file and checking the checksum after restoration. Data verification significantly reduces performance.

13. Select a **Target storage class**.
   If you selected **Filesystem copy**, you can change the target storage class.

14. Click **Next**.

15. On the **Migration options** page, the **Direct image migration** option is selected if you specified an exposed image registry route for the source cluster. The **Direct PV migration** option is selected if you are migrating data with **Filesystem copy**.
   The direct migration options copy images and files directly from the source cluster to the target cluster. This option is much faster than copying images and files from the source cluster to the replication repository and then from the replication repository to the target cluster.

16. Click **Next**.

17. Optional: Click **Add Hook** to add a hook to the migration plan.
   A hook runs custom code. You can add up to four hooks to a single migration plan. Each hook runs during a different migration step.

   a. Enter the name of the hook to display in the web console.

   b. If the hook is an Ansible playbook, select **Ansible playbook** and click **Browse** to upload the playbook or paste the contents of the playbook in the field.

   c. Optional: Specify an Ansible runtime image if you are not using the default hook image.

   d. If the hook is not an Ansible playbook, select **Custom container image** and specify the image name and path.
      A custom container image can include Ansible playbooks.

   e. Select **Source cluster** or **Target cluster**.

   f. Enter the **Service account name** and the **Service account namespace**

   g. Select the migration step for the hook:
      
      - **preBackup**: Before the application workload is backed up on the source cluster
      - **postBackup**: After the application workload is backed up on the source cluster
• **preRestore**: Before the application workload is restored on the target cluster

• **postRestore**: After the application workload is restored on the target cluster

h. Click **Add**.

18. Click **Finish**.
   The migration plan is displayed in the **Migration plans** list.

**Additional resources for persistent volume copy methods**

- **MTC file system copy method**
- **MTC snapshot copy method**

**8.2.5. Running a migration plan in the MTC web console**

You can migrate applications and data with the migration plan you created in the Migration Toolkit for Containers (MTC) web console.

**NOTE**

During migration, MTC sets the reclaim policy of migrated persistent volumes (PVs) to **Retain** on the target cluster.

The **Backup** custom resource contains a **PVOriginalReclaimPolicy** annotation that indicates the original reclaim policy. You can manually restore the reclaim policy of the migrated PVs.

**Prerequisites**

The MTC web console must contain the following:

- Source cluster in a **Ready** state
- Target cluster in a **Ready** state
- Replication repository
- Valid migration plan

**Procedure**

1. Log in to the MTC web console and click **Migration plans**.

2. Click the Options menu next to a migration plan and select one of the following options under **Migration**:

   - **Stage** copies data from the source cluster to the target cluster without stopping the application.
   - **Cutover** stops the transactions on the source cluster and moves the resources to the target cluster.
Optional: In the Cutover migration dialog, you can clear the Halt transactions on the source cluster during migration checkbox.

- **State** copies selected persistent volume claims (PVCs).

**IMPORTANT**

Do not use state migration to migrate a namespace between clusters. Use stage or cutover migration instead.

- Select one or more PVCs in the State migration dialog and click **Migrate**.

3. 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 volumes are correctly provisioned.
CHAPTER 9. ADVANCED MIGRATION OPTIONS

You can automate your migrations and modify the MigPlan and MigrationController custom resources in order to perform large-scale migrations and to improve performance.

9.1. TERMINOLOGY

Table 9.1. MTC terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source cluster</td>
<td>Cluster from which the applications are migrated.</td>
</tr>
<tr>
<td>Destination cluster[1]</td>
<td>Cluster to which the applications are migrated.</td>
</tr>
</tbody>
</table>
| Replication repository| Object storage used for copying images, volumes, and Kubernetes objects during indirect migration or for Kubernetes objects during direct volume migration or direct image migration.  
                          | The replication repository must be accessible to all clusters.                                                                                 |
| Host cluster          | Cluster on which the migration-controller pod and the web console are running. The host cluster is usually the destination cluster but this is not required.  
                          | The host cluster does not require an exposed registry route for direct image migration.                                                       |
| Remote cluster        | A remote cluster is usually the source cluster but this is not required.  
                          | A remote cluster requires a Secret custom resource that contains the migration-controller service account token.  
                          | A remote cluster requires an exposed secure registry route for direct image migration.                                                        |
| Indirect migration    | Images, volumes, and Kubernetes objects are copied from the source cluster to the replication repository and then from the replication repository to the destination cluster. |
| Direct volume migration| Persistent volumes are copied directly from the source cluster to the destination cluster.                                                  |
| Direct image migration | Images are copied directly from the source cluster to the destination cluster.                                                              |
| Stage migration       | Data is copied to the destination cluster without stopping the application.  
<pre><code>                      | Running a stage migration multiple times reduces the duration of the cutover migration.                                                      |
</code></pre>
<p>| Cutover migration     | The application is stopped on the source cluster and its resources are migrated to the destination cluster.                                  |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>State migration</td>
<td>Application state is migrated by copying specific persistent volume claims to the destination cluster.</td>
</tr>
<tr>
<td>Rollback migration</td>
<td>Rollback migration rolls back a completed migration.</td>
</tr>
</tbody>
</table>

1 Called the *target* cluster in the MTC web console.

### 9.2. MIGRATING APPLICATIONS BY USING THE COMMAND LINE

You can migrate applications with the MTC API by using the command line interface (CLI) in order to automate the migration.

#### 9.2.1. Migration prerequisites

- You must be logged in as a user with `cluster-admin` privileges on all clusters.

**Direct image migration**

- You must ensure that the secure internal registry of the source cluster is exposed.
- You must create a route to the exposed registry.

**Direct volume migration**

- If your clusters use proxies, you must configure an Stunnel TCP proxy.

**Clusters**

- The source cluster must be upgraded to the latest MTC z-stream release.
- The MTC version must be the same on all clusters.

**Network**

- The clusters have unrestricted network access to each other and to the replication repository.
- If you copy the persistent volumes with `move`, the clusters must have unrestricted network access to the remote volumes.
- You must enable the following ports on an OpenShift Container Platform 4 cluster:
  - 6443 (API server)
  - 443 (routes)
  - 53 (DNS)
- You must enable port 443 on the replication repository if you are using TLS.

**Persistent volumes (PVs)**
- The PVs must be valid.
- The PVs must be bound to persistent volume claims.
- If you use snapshots to copy the PVs, the following additional prerequisites apply:
  - The cloud provider must support snapshots.
  - The PVs must have the same cloud provider.
  - The PVs must be located in the same geographic region.
  - The PVs must have the same storage class.

### 9.2.2. Creating a registry route for direct image migration

For direct image migration, you must create a route to the exposed internal registry on all remote clusters.

**Prerequisites**

- The internal registry must be exposed to external traffic on all remote clusters. The OpenShift Container Platform 4 registry is exposed by default.

**Procedure**

- To create a route to an OpenShift Container Platform 4 registry, run the following command:

```
$ oc create route passthrough --service=image-registry -n openshift-image-registry
```

### 9.2.3. Proxy configuration

For OpenShift Container Platform 4.1 and earlier versions, you must configure proxies in the MigrationController custom resource (CR) manifest after you install the Migration Toolkit for Containers Operator because these versions do not support a cluster-wide `proxy` object.

For OpenShift Container Platform 4.2 to 4.8, the Migration Toolkit for Containers (MTC) inherits the cluster-wide proxy settings. You can change the proxy parameters if you want to override the cluster-wide proxy settings.

#### 9.2.3.1. Direct volume migration

Direct Volume Migration (DVM) was introduced in MTC 1.4.2. DVM supports only one proxy. The source cluster cannot access the route of the target cluster if the target cluster is also behind a proxy.

If you want to perform a DVM from a source cluster behind a proxy, you must configure a TCP proxy that works at the transport layer and forwards the SSL connections transparently without decrypting and re-encrypting them with their own SSL certificates. A Stunnel proxy is an example of such a proxy.

#### 9.2.3.1.1. TCP proxy setup for DVM

You can set up a direct connection between the source and the target cluster through a TCP proxy and configure the `stunnel_tcp_proxy` variable in the MigrationController CR to use the proxy:
Direct volume migration (DVM) supports only basic authentication for the proxy. Moreover, DVM works only from behind proxies that can tunnel a TCP connection transparently. HTTP/HTTPS proxies in man-in-the-middle mode do not work. The existing cluster-wide proxies might not support this behavior. As a result, the proxy settings for DVM are intentionally kept different from the usual proxy configuration in MTC.

9.2.3.1.2. Why use a TCP proxy instead of an HTTP/HTTPS proxy?

You can enable DVM by running Rsync between the source and the target cluster over an OpenShift route. Traffic is encrypted using Stunnel, a TCP proxy. The Stunnel running on the source cluster initiates a TLS connection with the target Stunnel and transfers data over an encrypted channel.

Cluster-wide HTTP/HTTPS proxies in OpenShift are usually configured in man-in-the-middle mode where they negotiate their own TLS session with the outside servers. However, this does not work with Stunnel. Stunnel requires that its TLS session be untouched by the proxy, essentially making the proxy a transparent tunnel which simply forwards the TCP connection as-is. Therefore, you must use a TCP proxy.

9.2.3.1.3. Known issue

Migration fails with error Upgrade request required

The migration Controller uses the SPDY protocol to execute commands within remote pods. If the remote cluster is behind a proxy or a firewall that does not support the SPDY protocol, the migration controller fails to execute remote commands. The migration fails with the error message Upgrade request required. Workaround: Use a proxy that supports the SPDY protocol.

In addition to supporting the SPDY protocol, the proxy or firewall also must pass the Upgrade HTTP header to the API server. The client uses this header to open a websocket connection with the API server. If the Upgrade header is blocked by the proxy or firewall, the migration fails with the error message Upgrade request required. Workaround: Ensure that the proxy forwards the Upgrade header.

9.2.3.2. Tuning network policies for migrations

OpenShift supports restricting traffic to or from pods using NetworkPolicy or EgressFirewalls based on the network plugin used by the cluster. If any of the source namespaces involved in a migration use such mechanisms to restrict network traffic to pods, the restrictions might inadvertently stop traffic to Rsync pods during migration.

Rsync pods running on both the source and the target clusters must connect to each other over an OpenShift Route. Existing NetworkPolicy or EgressNetworkPolicy objects can be configured to automatically exempt Rsync pods from these traffic restrictions.

9.2.3.2.1. NetworkPolicy configuration

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  ...
stunnel_tcp_proxy: http://username:password@ip:port
```
9.2.3.2.1.1. Egress traffic from Rsync pods

You can use the unique labels of Rsync pods to allow egress traffic to pass from them if the 
NetworkPolicy configuration in the source or destination namespaces blocks this type of traffic. The 
following policy allows all egress traffic from Rsync pods in the namespace:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-from-rsync-pods
spec:
  podSelector:
    matchLabels:
      owner: directvolumemigration
      app: directvolumemigration-rsync-transfer
  egress:
    - {}
  policyTypes:
    - Egress
```

9.2.3.2.1.2. Ingress traffic to Rsync pods

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-egress-from-rsync-pods
spec:
  podSelector:
    matchLabels:
      owner: directvolumemigration
      app: directvolumemigration-rsync-transfer
  ingress:
    - {}
  policyTypes:
    - Ingress
```

9.2.3.2.2. EgressNetworkPolicy configuration

The EgressNetworkPolicy object or Egress Firewalls are OpenShift constructs designed to block 
egress traffic leaving the cluster.

Unlike the NetworkPolicy object, the Egress Firewall works at a project level because it applies to all 
pods in the namespace. Therefore, the unique labels of Rsync pods do not exempt only Rsync pods from 
the restrictions. However, you can add the CIDR ranges of the source or target cluster to the Allow rule 
of the policy so that a direct connection can be setup between two clusters.

Based on which cluster the Egress Firewall is present in, you can add the CIDR range of the other cluster 
to allow egress traffic between the two:

```yaml
apiVersion: network.openshift.io/v1
kind: EgressNetworkPolicy
metadata:
  name: test-egress-policy
namespace: <namespace>
```
9.2.3.2.3. Configuring supplemental groups for Rsync pods

When your PVCs use a shared storage, you can configure the access to that storage by adding supplemental groups to Rsync pod definitions in order for the pods to allow access:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for source Rsync pods</td>
</tr>
<tr>
<td>target_supplemental_groups</td>
<td>string</td>
<td>Not set</td>
<td>Comma-separated list of supplemental groups for target Rsync pods</td>
</tr>
</tbody>
</table>

**Example usage**

The **MigrationController** CR can be updated to set values for these supplemental groups:

```yaml
spec:
  egress:
    - to:
      cidrSelector: <cidr_of_source_or_target_cluster>
      type: Deny
```

9.2.3.3. Configuring proxies

**Prerequisites**

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

**Procedure**

1. Get the **MigrationController** CR manifest:

   ```bash
   $ oc get migrationcontroller <migration_controller> -n openshift-migration
   ```

2. Update the proxy parameters:

   ```yaml
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: <migration_controller>
     namespace: openshift-migration
   ```
spec:
  stunnel_tcp_proxy: http://<username>:<password>@<ip>:<port>  
noProxy: example.com

1. Stunnel proxy URL for direct volume migration.
2. Comma-separated list of destination domain names, domains, IP addresses, or other network CIDRs to exclude proxying.

Preface a domain with . to match subdomains only. For example, .y.com matches x.y.com, but not y.com. Use * to bypass proxy for all destinations. If you scale up workers that are not included in the network defined by the networking.machineNetwork[].cidr field from the installation configuration, you must add them to this list to prevent connection issues.

This field is ignored if neither the httpProxy nor the httpsProxy field is set.

3. Save the manifest as migration-controller.yaml.

4. Apply the updated manifest:

   $ oc replace -f migration-controller.yaml -n openshift-migration

9.2.4. Migrating an application by using the MTC API

You can migrate an application from the command line by using the Migration Toolkit for Containers (MTC) API.

Procedure

1. Create a MigCluster CR manifest for the host cluster:

   $ cat << EOF | oc apply -f -
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigCluster
   metadata:
     name: <host_cluster>
     namespace: openshift-migration
   spec:
     isHostCluster: true
   EOF

2. Create a Secret CR manifest for each remote cluster:

   $ cat << EOF | oc apply -f -
   apiVersion: v1
   kind: Secret
   metadata:
     name: <cluster_secret>
     namespace: openshift-config
   type: Opaque
   data:
     saToken: <sa_token>  
   EOF
Specify the base64-encoded migration-controller service account (SA) token of the remote cluster. You can obtain the token by running the following command:

```
$ oc sa get-token migration-controller -n openshift-migration | base64 -w 0
```

3. Create a **MigCluster** CR manifest for each remote cluster:

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  name: <remote_cluster>  
  namespace: openshift-migration
spec:
  exposedRegistryPath: <exposed_registry_route>   
  insecure: false
  isHostCluster: false
  serviceAccountSecretRef:
    name: <remote_cluster_secret>   
    namespace: openshift-config
  url: <remote_cluster_url>
EOF
```

1. Specify the **Cluster** CR of the remote cluster.

2. Optional: For direct image migration, specify the exposed registry route.

3. SSL verification is enabled if **false**. CA certificates are not required or checked if **true**.

4. Specify the **Secret** CR object of the remote cluster.

5. Specify the URL of the remote cluster.

4. Verify that all clusters are in a **Ready** state:

```
$ oc describe cluster <cluster>
```

5. Create a **Secret** CR manifest for the replication repository:

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  namespace: openshift-config
  name: <migstorage_creds>
  type: Opaque
data:
  aws-access-key-id: <key_id_base64>   
  aws-secret-access-key: <secret_key_base64>
EOF
```

1. Specify the key ID in base64 format.
Specify the secret key in base64 format.

AWS credentials are base64-encoded by default. For other storage providers, you must encode your credentials by running the following command with each key:

```
$ echo -n "<key>" | base64 -w 0
```

Specify the key ID or the secret key. Both keys must be base64-encoded.

6. Create a **MigStorage** CR manifest for the replication repository:

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigStorage
metadata:
  name: <migstorage>
  namespace: openshift-migration
spec:
  backupStorageConfig:
    awsBucketName: <bucket>  
    credsSecretRef:
      name: <storage_secret>
      namespace: openshift-config
  backupStorageProvider: <storage_provider>
  volumeSnapshotConfig:
    credsSecretRef:
      name: <storage_secret>
      namespace: openshift-config
  volumeSnapshotProvider: <storage_provider>
EOF
```

1. Specify the bucket name.

2. Specify the **Secrets** CR of the object storage. You must ensure that the credentials stored in the **Secrets** CR of the object storage are correct.

3. Specify the storage provider.

4. Optional: If you are copying data by using snapshots, specify the **Secrets** CR of the object storage. You must ensure that the credentials stored in the **Secrets** CR of the object storage are correct.

5. Optional: If you are copying data by using snapshots, specify the storage provider.

7. Verify that the **MigStorage** CR is in a **Ready** state:

```
$ oc describe migstorage <migstorage>
```

8. Create a **MigPlan** CR manifest:

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  migStorageRef:
    name: <migstorage>
    namespace: openshift-migration
EOF
```
kind: MigPlan
metadata:
  name: <migplan>
namespace: openshift-migration
spec:
destMigClusterRef:
  name: <host_cluster>
namespace: openshift-migration
indirectImageMigration: true
indirectVolumeMigration: true
migStorageRef:
  name: <migstorage>
namespace: openshift-migration
namespaces:
  - <source_namespace_1>
  - <source_namespace_2>
  - <source_namespace_3>:<destination_namespace>
srcMigClusterRef:
  name: <remote_cluster>
namespace: openshift-migration
EOF

1. Direct image migration is enabled if false.
2. Direct volume migration is enabled if false.
3. Specify the name of the MigStorage CR instance.
4. Specify one or more source namespaces. By default, the destination namespace has the same name.
5. Specify a destination namespace if it is different from the source namespace.
6. Specify the name of the source cluster MigCluster instance.

9. Verify that the MigPlan instance is in a Ready state:

   $ oc describe migplan <migplan> -n openshift-migration

10. Create a MigMigration CR manifest to start the migration defined in the MigPlan instance:

   $ cat << EOF | oc apply -f -
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigMigration
   metadata:
     name: <migmigration>
     namespace: openshift-migration
   spec:
     migPlanRef:
       name: <migplan>
       namespace: openshift-migration
     quiescePods: true
   EOF
CHAPTER 9. ADVANCED MIGRATION OPTIONS

Specify the **MigPlan** CR name.

The pods on the source cluster are stopped before migration if **true**.

A stage migration, which copies most of the data without stopping the application, is performed if **true**.

A completed migration is rolled back if **true**.

11. Verify the migration by watching the **MigMigration** CR progress:

```
$ oc watch migmigration <migmigration> -n openshift-migration
```

The output resembles the following:

**Example output**

```
Name:         c8b034c0-6567-11eb-9a4f-0bc004db0fbc
Namespace:    openshift-migration
Labels:       migration.openshift.io/migplan-name=djingo
Annotations:  openshift.io/touch: e99f9083-6567-11eb-8420-0a580a81020c
API Version:  migration.openshift.io/v1alpha1
Kind:         MigMigration
...
Spec:
  Mig Plan Ref:
    Name:       migplan
    Namespace:  openshift-migration
    Stage:      false
Status:
  Conditions:
    Category:      Advisory
    Last Transition Time:  2021-02-02T15:04:09Z
    Message:       Step: 19/47
    Reason:        InitialBackupCreated
    Status:        True
    Type:          Running
    Category:      Required
    Last Transition Time:  2021-02-02T15:03:19Z
    Message:       The migration is ready.
    Status:        True
    Type:          Ready
    Category:      Required
    Durable:       true
    Last Transition Time:  2021-02-02T15:04:05Z
    Message:       The migration registries are healthy.
    Status:        True
    Type:          RegistriesHealthy
    Itinerary:     Final
  Observed Digest: 7fae9d21f15979c71ddc7dd075cb97061895caac5b936d92fae967019ab616d5
```
9.2.5. State migration

You can perform repeatable, state-only migrations by using Migration Toolkit for Containers (MTC) to migrate persistent volume claims (PVCs) that constitute an application’s state. You migrate specified PVCs by excluding other PVCs from the migration plan. You can map the PVCs to ensure that the source and the target PVCs are synchronized. Persistent volume (PV) data is copied to the target cluster. The PV references are not moved, and the application pods continue to run on the source cluster.

State migration is specifically designed to be used in conjunction with external CD mechanisms, such as OpenShift Gitops. You can migrate application manifests using GitOps while migrating the state using MTC.

If you have a CI/CD pipeline, you can migrate stateless components by deploying them on the target cluster. Then you can migrate stateful components by using MTC.
You can perform a state migration between clusters or within the same cluster.

**IMPORTANT**
State migration migrates only the components that constitute an application’s state. If you want to migrate an entire namespace, use stage or cutover migration.

**Prerequisites**
- The state of the application on the source cluster is persisted in **PersistentVolumes** provisioned through **PersistentVolumeClaims**.
- The manifests of the application are available in a central repository that is accessible from both the source and the target clusters.

**Procedure**
1. Migrate persistent volume data from the source to the target cluster. You can perform this step as many times as needed. The source application continues running.
2. Quiesce the source application. You can do this by setting the replicas of workload resources to 0, either directly on the source cluster or by updating the manifests in GitHub and re-syncing the Argo CD application.
3. Clone application manifests to the target cluster. You can use Argo CD to clone the application manifests to the target cluster.
4. Migrate the remaining volume data from the source to the target cluster. Migrate any new data created by the application during the state migration process by performing a final data migration.
5. If the cloned application is in a quiesced state, unquiesce it.
6. Switch the DNS record to the target cluster to re-direct user traffic to the migrated application.

**NOTE**
MTC 1.6 cannot quiesce applications automatically when performing state migration. It can only migrate PV data. Therefore, you must use your CD mechanisms for quiescing or unquiescing applications.

MTC 1.7 introduces explicit Stage and Cutover flows. You can use staging to perform initial data transfers as many times as needed. Then you can perform a cutover, in which the source applications are quiesced automatically.

**Prerequisites**
- The state of the application on the source cluster is persisted in **PersistentVolumes** provisioned through **PersistentVolumeClaims**.
- The manifests of the application are available in a central repository that is accessible from both the source and the target clusters.

**Procedure**
1. Migrate persistent volume data from the source to the target cluster.
You can perform this step as many times as needed. The source application continues running.

2. Quiesce the source application.
You can do this by setting the replicas of workload resources to 0, either directly on the source
cluster or by updating the manifests in GitHub and re-syncing the Argo CD application.

3. Clone application manifests to the target cluster.
You can use Argo CD to clone the application manifests to the target cluster.

4. Migrate the remaining volume data from the source to the target cluster.
Migrate any new data created by the application during the state migration process by
performing a final data migration.

5. If the cloned application is in a quiesced state, unquiesce it.

6. Switch the DNS record to the target cluster to re-direct user traffic to the migrated application.

**NOTE**
MTC 1.6 cannot quiesce applications automatically when performing state migration. It
can only migrate PV data. Therefore, you must use your CD mechanisms for quiescing or
unquiescing applications.

MTC 1.7 introduces explicit Stage and Cutover flows. You can use staging to perform
initial data transfers as many times as needed. Then you can perform a cutover, in which
the source applications are quiesced automatically.

**Additional resources**
- See Excluding PVCs from migration to select PVCs for state migration.
- See Mapping PVCs to migrate source PV data to provisioned PVCs on the destination cluster.
- See Migrating Kubernetes objects to migrate the Kubernetes objects that constitute an
  application’s state.

**9.3. MIGRATION HOOKS**
You can add up to four migration hooks to a single migration plan, with each hook running at a different
phase of the migration. Migration hooks perform tasks such as customizing application quiescence,
manually migrating unsupported data types, and updating applications after migration.

A migration hook runs on a source or a target cluster at one of the following migration steps:
- **PreBackup**: Before resources are backed up on the source cluster.
- **PostBackup**: After resources are backed up on the source cluster.
- **PreRestore**: Before resources are restored on the target cluster.
- **PostRestore**: After resources are restored on the target cluster.

You can create a hook by creating an Ansible playbook that runs with the default Ansible image or with a
custom hook container.
Ansible playbook

The Ansible playbook is mounted on a hook container as a config map. The hook container runs as a job, using the cluster, service account, and namespace specified in the MigPlan custom resource. The job continues to run until it reaches the default limit of 6 retries or a successful completion. This continues even if the initial pod is evicted or killed.

The default Ansible runtime image is registry.redhat.io/rhmtc/openshift-migration-hook-runner-rhel7:1.7. This image is based on the Ansible Runner image and includes python-openshift for Ansible Kubernetes resources and an updated oc binary.

Custom hook container

You can use a custom hook container instead of the default Ansible image.

9.3.1. Writing an Ansible playbook for a migration hook

You can write an Ansible playbook to use as a migration hook. The hook is added to a migration plan by using the MTC web console or by specifying values for the spec.hooks parameters in the MigPlan custom resource (CR) manifest.

The Ansible playbook is mounted onto a hook container as a config map. The hook container runs as a job, using the cluster, service account, and namespace specified in the MigPlan CR. The hook container uses a specified service account token so that the tasks do not require authentication before they run in the cluster.

9.3.1.1. Ansible modules

You can use the Ansible shell module to run oc commands.

Example shell module

```yaml
- hosts: localhost
gather_facts: false
tasks:
- name: get pod name
  shell: oc get po --all-namespaces
```

You can use kubernetes.core modules, such as k8s_info, to interact with Kubernetes resources.

Example k8s_facts module

```yaml
- hosts: localhost
gather_facts: false
tasks:
- name: Get pod
  k8s_info:
    kind: pods
    api: v1
    namespace: openshift-migration
    name: "{{ lookup('env', 'HOSTNAME') }}"
  register: pods
```
You can use the `fail` module to produce a non-zero exit status in cases where a non-zero exit status would not normally be produced, ensuring that the success or failure of a hook is detected. Hooks run as jobs and the success or failure status of a hook is based on the exit status of the job container.

**Example fail module**

```yaml
- hosts: localhost
gather_facts: false
tasks:
  - name: Set a boolean
    set_fact:
      do_fail: true

  - name: "fail"
    fail:
      msg: "Cause a failure"
      when: do_fail
```

### 9.3.1.2. Environment variables

The `MigPlan` CR name and migration namespaces are passed as environment variables to the hook container. These variables are accessed by using the `lookup` plugin.

**Example environment variables**

```yaml
- hosts: localhost
  gather_facts: false
tasks:
  - set_fact:
      namespaces: "{{ (lookup('env', 'MIGRATION_NAMESPACES')).split(',') }}"

  - debug:
      msg: "{{ item }}"
      with_items: "{{ namespaces }}"

  - debug:
      msg: "{{ lookup('env', 'MIGRATION_PLAN_NAME') }}"
```

### 9.4. MIGRATION PLAN OPTIONS

You can exclude, edit, and map components in the `MigPlan` custom resource (CR).

#### 9.4.1. Excluding resources

You can exclude resources, for example, image streams, persistent volumes (PVs), or subscriptions, from a Migration Toolkit for Containers (MTC) migration plan to reduce the resource load for migration or to migrate images or PVs with a different tool.
By default, the MTC excludes service catalog resources and Operator Lifecycle Manager (OLM) resources from migration. These resources are parts of the service catalog API group and the OLM API group, neither of which is supported for migration at this time.

Procedure

1. Edit the MigrationController custom resource manifest:

   ```bash
   $ oc edit migrationcontroller <migration_controller> -n openshift-migration
   ```

2. Update the spec section by adding a parameter to exclude specific resources or by adding a resource to the excluded_resources parameter if it does not have its own exclusion parameter:

   ```yaml
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: migration-controller
     namespace: openshift-migration
   spec:
     disable_image_migration: true
     disable_pv_migration: true
     ... 
     excluded_resources:
       - imagetags
       - templateinstances
       - clusterserviceversions
       - packagemanifests
       - subscriptions
       - servicebrokers
       - servicebindings
       - serviceclasses
       - serviceinstances
       - operatorgroups
       - events
       - events.events.k8s.io
   ```

   **1** Add `disable_image_migration: true` to exclude image streams from the migration. Do not edit the excluded_resources parameter. `imagetags` is added to excluded_resources when the MigrationController pod restarts.

   **2** Add `disable_pv_migration: true` to exclude PVs from the migration plan. Do not edit the excluded_resources parameter. `persistentvolumes` and `persistentvolumeclaims` are added to excluded_resources when the MigrationController pod restarts. Disabling PV migration also disables PV discovery when you create the migration plan.

   **3** You can add OpenShift Container Platform resources to the excluded_resources list. Do not delete the default excluded resources. These resources are problematic to migrate and must be excluded.

3. Wait two minutes for the MigrationController pod to restart so that the changes are applied.

4. Verify that the resource is excluded:
9.4.2. Mapping namespaces

If you map namespaces in the **MigPlan** custom resource (CR), you must ensure that the namespaces are not duplicated on the source or the destination clusters because the UID and GID ranges of the namespaces are copied during migration.

### Two source namespaces mapped to the same destination namespace

```yaml
spec:
  namespaces:
    - namespace_2
    - namespace_1:namespace_2
```

If you want the source namespace to be mapped to a namespace of the same name, you do not need to create a mapping. By default, a source namespace and a target namespace have the same name.

### Incorrect namespace mapping

```yaml
spec:
  namespaces:
    - namespace_1:namespace_1
```

### Correct namespace reference

```yaml
spec:
  namespaces:
    - namespace_1
```

9.4.3. Excluding persistent volume claims

You select persistent volume claims (PVCs) for state migration by excluding the PVCs that you do not want to migrate. You exclude PVCs by setting the `spec.persistentVolumes.pvc.selection.action` parameter of the **MigPlan** custom resource (CR) after the persistent volumes (PVs) have been discovered.

**Prerequisites**

- **MigPlan** CR is in a **Ready** state.
9.4.4. Mapping persistent volume claims

You can migrate persistent volume (PV) data from the source cluster to persistent volume claims (PVCs) that are already provisioned in the destination cluster in the MigPlan CR by mapping the PVCs. This mapping ensures that the destination PVCs of migrated applications are synchronized with the source PVCs.

You map PVCs by updating the spec.persistentVolumes.pvc.name parameter in the MigPlan custom resource (CR) after the PVs have been discovered.

Prerequisites

- MigPlan CR is in a Ready state.

Procedure

- Update the spec.persistentVolumes.pvc.name parameter in the MigPlan CR:

```
apiVersion: migration.openshift.io/v1alpha1
class: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  ...
  persistentVolumes:
  - capacity: 10Gi
    name: <pv_name>
    pvc:
      name: <source_pvc>:<destination_pvc>
```

1 Specify the PVC on the source cluster and the PVC on the destination cluster. If the destination PVC does not exist, it will be created. You can use this mapping to change the PVC name during migration.
9.4.5. Editing persistent volume attributes

After you create a MigPlan custom resource (CR), the MigrationController CR discovers the persistent volumes (PVs). The spec.persistentVolumes block and the status.destStorageClasses block are added to the MigPlan CR.

You can edit the values in the spec.persistentVolumes.selection block. If you change values outside the spec.persistentVolumes.selection block, the values are overwritten when the MigPlan CR is reconciled by the MigrationController CR.

**NOTE**

The default value for the spec.persistentVolumes.selection.storageClass parameter is determined by the following logic:

1. If the source cluster PV is Gluster or NFS, the default is either cephfs, for accessMode: ReadWriteMany, or cephrbd, for accessMode: ReadWriteOnce.
2. If the PV is neither Gluster nor NFS or if cephfs or cephrbd are not available, the default is a storage class for the same provisioner.
3. If a storage class for the same provisioner is not available, the default is the default storage class of the destination cluster.

You can change the storageClass value to the value of any name parameter in the status.destStorageClasses block of the MigPlan CR.

If the storageClass value is empty, the PV will have no storage class after migration. This option is appropriate if, for example, you want to move the PV to an NFS volume on the destination cluster.

**Prerequisites**

- MigPlan CR is in a Ready state.

**Procedure**

- Edit the spec.persistentVolumes.selection values in the MigPlan CR:

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
persistentVolumes:
- capacity: 10Gi
  name: pvc-095a6559-b27f-11eb-b27f-021bddcaf6e4
  proposedCapacity: 10Gi
  pvc:
    accessModes:
    - ReadWriteMany
    hasReference: true
    name: mysql
    namespace: mysql-persistent
    selection:
```
**9.4.6. Performing a state migration of Kubernetes objects by using the MTC API**

After you migrate all the PV data, you can use the Migration Toolkit for Containers (MTC) API to perform a one-time state migration of Kubernetes objects that constitute an application.

You do this by configuring **MigPlan** custom resource (CR) fields to provide a list of Kubernetes resources with an additional label selector to further filter those resources, and then performing a migration by creating a **MigMigration** CR. The **MigPlan** resource is closed after the migration.

**NOTE**

Selecting Kubernetes resources is an API-only feature. You must update the **MigPlan** CR and create a **MigMigration** CR for it by using the CLI. The MTC web console does not support migrating Kubernetes objects.

**NOTE**

After migration, the **closed** parameter of the **MigPlan** CR is set to **true**. You cannot create another **MigMigration** CR for this **MigPlan** CR.

You add Kubernetes objects to the **MigPlan** CR by using one of the following options:

```yaml
action: <copy>  
copyMethod: <filesystem>  
verify: true  
storageClass: <gp2>  
accessMode: <ReadWriteMany>  
storageClass: cephfs
```

1. Allowed values are **move**, **copy**, and **skip**. If only one action is supported, the default value is the supported action. If multiple actions are supported, the default value is **copy**.

2. Allowed values are **snapshot** and **filesystem**. Default value is **filesystem**.

3. The **verify** parameter is displayed if you select the verification option for file system copy in the MTC web console. You can set it to **false**.

4. You can change the default value to the value of any **name** parameter in the **status.destStorageClasses** block of the **MigPlan** CR. If no value is specified, the PV will have no storage class after migration.

5. Allowed values are **ReadWriteOnce** and **ReadWriteMany**. If this value is not specified, the default is the access mode of the source cluster PVC. You can only edit the access mode in the **MigPlan** CR. You cannot edit it by using the MTC web console.

**Additional resources**

- For details about the **move** and **copy** actions, see **MTC workflow**.

- For details about the **skip** action, see **Excluding PVCs from migration**.

- For details about the file system and snapshot copy methods, see **About data copy methods**.
- Adding the Kubernetes objects to the `includedResources` section. When the `includedResources` field is specified in the `MigPlan` CR, the plan takes a list of `group-kind` as input. Only resources present in the list are included in the migration.

- Adding the optional `labelSelector` parameter to filter the `includedResources` in the `MigPlan`. When this field is specified, only resources matching the label selector are included in the migration. For example, you can filter a list of `Secret` and `ConfigMap` resources by using the label `app: frontend` as a filter.

**Procedure**

1. Update the `MigPlan` CR to include Kubernetes resources and, optionally, to filter the included resources by adding the `labelSelector` parameter:
   
   a. To update the `MigPlan` CR to include Kubernetes resources:
      
      ```yaml
      apiVersion: migration.openshift.io/v1alpha1
      kind: MigPlan
      metadata:
        name: <migplan>
        namespace: openshift-migration
      spec:
        includedResources:
        - kind: <kind>
          group: ""  
        - kind: <kind>
          group: ""
      ```
      
      ① Specify the Kubernetes object, for example, `Secret` or `ConfigMap`.

   b. Optional: To filter the included resources by adding the `labelSelector` parameter:
      
      ```yaml
      apiVersion: migration.openshift.io/v1alpha1
      kind: MigPlan
      metadata:
        name: <migplan>
        namespace: openshift-migration
      spec:
        includedResources:
        - kind: <kind>
          group: ""  
        - kind: <kind>
          group: ""
        labelSelector:
          matchLabels:
            <label>  
      ```
      
      ① Specify the Kubernetes object, for example, `Secret` or `ConfigMap`.
      
      ② Specify the label of the resources to migrate, for example, `app: frontend`.

2. Create a `MigMigration` CR to migrate the selected Kubernetes resources. Verify that the correct `MigPlan` is referenced in `migPlanRef`:
9.5. MIGRATION CONTROLLER OPTIONS

You can edit migration plan limits, enable persistent volume resizing, or enable cached Kubernetes clients in the **MigrationController** custom resource (CR) for large migrations and improved performance.

9.5.1. Increasing limits for large migrations

You can increase the limits on migration objects and container resources for large migrations with the Migration Toolkit for Containers (MTC).

**IMPORTANT**

You must test these changes before you perform a migration in a production environment.

**Procedure**

1. Edit the **MigrationController** custom resource (CR) manifest:

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

2. Update the following parameters:

   ```yaml
   ... 
   mig_controller_limits_cpu: "1" ¹
   mig_controller_limits_memory: "10Gi" ²
   ...
   mig_controller_requests_cpu: "100m" ³
   mig_controller_requests_memory: "350Mi" ⁴
   ...
   mig_pv_limit: 100 ⁵
   mig_pod_limit: 100 ⁶
   mig_namespace_limit: 10 ⁷
   ...
   ```

   ¹ Specifies the number of CPUs available to the **MigrationController** CR.
   ² Specifies the amount of memory available to the **MigrationController** CR.
   ³
Specifies the number of CPU units available for MigrationController CR requests. 100m represents 0.1 CPU units (100 \* 1e-3).

4. Specifies the amount of memory available for MigrationController CR requests.

5. Specifies the number of persistent volumes that can be migrated.

6. Specifies the number of pods that can be migrated.

7. Specifies the number of namespaces that can be migrated.

3. Create a migration plan that uses the updated parameters to verify the changes. If your migration plan exceeds the MigrationController CR limits, the MTC console displays a warning message when you save the migration plan.

9.5.2. Enabling persistent volume resizing for direct volume migration

You can enable persistent volume (PV) resizing for direct volume migration to avoid running out of disk space on the destination cluster.

When the disk usage of a PV reaches a configured level, the MigrationController custom resource (CR) compares the requested storage capacity of a persistent volume claim (PVC) to its actual provisioned capacity. Then, it calculates the space required on the destination cluster.

A pv_resizing_threshold parameter determines when PV resizing is used. The default threshold is 3%. This means that PV resizing occurs when the disk usage of a PV is more than 97%. You can increase this threshold so that PV resizing occurs at a lower disk usage level.

PVC capacity is calculated according to the following criteria:

- If the requested storage capacity (spec.resources.requests.storage) of the PVC is not equal to its actual provisioned capacity (status.capacity.storage), the greater value is used.

- If a PV is provisioned through a PVC and then subsequently changed so that its PV and PVC capacities no longer match, the greater value is used.

Prerequisites

- The PVCs must be attached to one or more running pods so that the MigrationController CR can execute commands.

Procedure

1. Log in to the host cluster.

2. Enable PV resizing by patching the MigrationController CR:

   ```bash
   $ oc patch migrationcontroller migration-controller -p '{"spec":
   {
   "enable_dvm_pv_resizing":true}
   }' --type='merge' -n openshift-migration
   ``

   1. Set the value to false to disable PV resizing.

3. Optional: Update the pv_resizing_threshold parameter to increase the threshold:
The default value is 3.

When the threshold is exceeded, the following status message is displayed in the MigPlan CR status:

status:
  conditions:
  ...
  - category: Warn
    durable: true
    lastTransitionTime: "2021-06-17T08:57:01Z"
    message: 'Capacity of the following volumes will be automatically adjusted to avoid disk capacity issues in the target cluster: [pvc-b800eb7b-cf3b-11eb-a3f7-0eae3e0555f3]'
    reason: Done
    status: "False"
    type: PvcCapacityAdjustmentRequired

NOTE

For AWS gp2 storage, this message does not appear unless the `pv_resizing_threshold` is 42% or greater because of the way gp2 calculates volume usage and size. (BZ#1973148)

9.5.3. Enabling cached Kubernetes clients

You can enable cached Kubernetes clients in the MigrationController custom resource (CR) for improved performance during migration. The greatest performance benefit is displayed when migrating between clusters in different regions or with significant network latency.

NOTE

Delegated tasks, for example, Rsync backup for direct volume migration or Velero backup and restore, however, do not show improved performance with cached clients.

Cached clients require extra memory because the MigrationController CR caches all API resources that are required for interacting with MigCluster CRs. Requests that are normally sent to the API server are directed to the cache instead. The cache watches the API server for updates.

You can increase the memory limits and requests of the MigrationController CR if OOMKilled errors occur after you enable cached clients.

Procedure

1. Enable cached clients by running the following command:

```bash
$ oc -n openshift-migration patch migrationcontroller migration-controller --type=json --patch \\  
  '[[ "op": "replace", "path": "/spec/mig_controller_enable_cache", "value": true]]'
```
2. Optional: Increase the MigrationController CR memory limits by running the following command:

```
$ oc -n openshift-migration patch migrationcontroller migration-controller --type=json --patch \\
  '{ "op": "replace", "path": "/spec/mig_controller_limits_memory", "value": <10Gi>}'
```

3. Optional: Increase the MigrationController CR memory requests by running the following command:

```
$ oc -n openshift-migration patch migrationcontroller migration-controller --type=json --patch \\
  '{ "op": "replace", "path": "/spec/mig_controller_requests_memory", "value": <350Mi>}'
```
CHAPTER 10. TROUBLESHOOTING

This section describes resources for troubleshooting the Migration Toolkit for Containers (MTC).

For known issues, see the MTC release notes.

10.1. MTC WORKFLOW

You can migrate Kubernetes resources, persistent volume data, and internal container images to OpenShift Container Platform 4.8 by using the Migration Toolkit for Containers (MTC) web console or the Kubernetes API.

MTC migrates the following resources:

- A namespace specified in a migration plan.
- Namespace-scoped resources: When the MTC migrates a namespace, it migrates all the objects and resources associated with that namespace, such as services or pods. Additionally, if a resource that exists in the namespace but not at the cluster level depends on a resource that exists at the cluster level, the MTC migrates both resources.

For example, a security context constraint (SCC) is a resource that exists at the cluster level and a service account (SA) is a resource that exists at the namespace level. If an SA exists in a namespace that the MTC migrates, the MTC automatically locates any SCCs that are linked to the SA and also migrates those SCCs. Similarly, the MTC migrates persistent volume claims that are linked to the persistent volumes of the namespace.

**NOTE**

Cluster-scoped resources might have to be migrated manually, depending on the resource.

- Custom resources (CRs) and custom resource definitions (CRDs): MTC automatically migrates CRs and CRDs at the namespace level.

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

1. Install the Migration Toolkit for Containers Operator on all clusters.
   You can install the Migration Toolkit for Containers 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 MTC 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 Multi-Cloud Object Gateway (MCG). If you are using a proxy server, you must configure it to allow network traffic between the replication repository and the clusters.

3. Add the source cluster to the MTC web console.

4. Add the replication repository to the MTC web console.

5. Create a migration plan, with one of the following data migration options:
- **Copy**: MTC copies the data from the source cluster to the replication repository, and from the replication repository to the target cluster.

  **NOTE**
  If you are using direct image migration or direct volume migration, the images or volumes are copied directly from the source cluster to the target cluster.

- **Move**: MTC 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 migration.

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

- **Stage** copies data to the target cluster without stopping the application.
  A stage migration can be run multiple times so that most of the data is copied to the target before migration. Running one or more stage migrations reduces the duration of the cutover migration.

- **Cutover** stops the application on the source cluster and moves the resources to the target cluster.
  Optional: You can clear the **Halt transactions on the source cluster during migration** checkbox.
About MTC custom resources
The Migration Toolkit for Containers (MTC) creates the following custom resources (CRs):

1. **MigPlan**
2. **MigCluster**
3. **MigStorage**
4. **BackupStorageLocation**
5. **VolumeSnapshotLocation**
6. **MigMigration**
7. **Backup**
8. **Restore**

User access
1. **MigCluster** (configuration, MTC cluster): Cluster definition

2. **MigStorage** (configuration, MTC cluster): Storage definition

3. **MigPlan** (configuration, MTC cluster): Migration plan

The **MigPlan** CR describes the source and target clusters, replication repository, and namespaces 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, MTC cluster): Location of **Velero** backup objects

5. **VolumeSnapshotLocation** (configuration, MTC cluster): Location of **Velero** volume snapshots

6. **MigMigration** (action, MTC cluster): Migration, 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

**10.2. MTC CUSTOM RESOURCE MANIFESTS**

Migration Toolkit for Containers (MTC) uses the following custom resource (CR) manifests for migrating applications.

**10.2.1. DirectImageMigration**

The **DirectImageMigration** CR copies images directly from the source cluster to the destination cluster.

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: DirectImageMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <direct_image_migration>
spec:
```

92
One or more namespaces containing images to be migrated. By default, the destination namespace has the same name as the source namespace.

Source namespace mapped to a destination namespace with a different name.

10.2.2. DirectImageStreamMigration

The DirectImageStreamMigration CR copies image stream references directly from the source cluster to the destination cluster.

```
apiVersion: migration.openshift.io/v1alpha1
kind: DirectImageStreamMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <direct_image_stream_migration>
spec:
  srcMigClusterRef:
   name: <source_cluster>
   namespace: openshift-migration
  destMigClusterRef:
   name: <destination_cluster>
   namespace: openshift-migration
  imageStreamRef:
   name: <image_stream>
   namespace: <source_image_stream_namespace>
  destNamespace: <destination_image_stream_namespace>
```

10.2.3. DirectVolumeMigration

The DirectVolumeMigration CR copies persistent volumes (PVs) directly from the source cluster to the destination cluster.

```
apiVersion: migration.openshift.io/v1alpha1
kind: DirectVolumeMigration
metadata:
  name: <direct_volume_migration>
  namespace: openshift-migration
spec:
  createDestinationNamespaces: false
  deleteProgressReportingCRs: false
  destMigClusterRef:
   name: <host_cluster>
```

1. One or more namespaces containing images to be migrated. By default, the destination namespace has the same name as the source namespace.
2. Source namespace mapped to a destination namespace with a different name.
Set to **true** to create namespaces for the PVs on the destination cluster.

Set to **true** to delete `DirectVolumeMigrationProgress` CRs after migration. The default is **false** so that `DirectVolumeMigrationProgress` CRs are retained for troubleshooting.

Update the cluster name if the destination cluster is not the host cluster.

Specify one or more PVCs to be migrated.

### 10.2.4. DirectVolumeMigrationProgress

The `DirectVolumeMigrationProgress` CR shows the progress of the `DirectVolumeMigration` CR.

```yaml
apiVersion: migration.openshift.io/v1alpha1
class: DirectVolumeMigrationProgress
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <direct_volume_migration_progress>
spec:
  clusterRef:
    name: <source_cluster>
  namespace: openshift-migration
  podRef:
    name: <rsync_pod>
  namespace: openshift-migration
```

### 10.2.5. MigAnalytic

The MigAnalytic CR collects the number of images, Kubernetes resources, and the persistent volume (PV) capacity from an associated MigPlan CR.

You can configure the data that it collects.

```yaml
apiVersion: migration.openshift.io/v1alpha1
class: MigAnalytic
metadata:
  annotations:
    migplan: <migplan>
  name: <miganalytic>
  namespace: openshift-migration
  labels:
    migplan: <migplan>
spec:
  analyzeImageCount: true
```
analyzeK8SResources: true
analyzePVCapacity: true
listImages: false
listImagesLimit: 50
migPlanRef:
  name: <migplan>
  namespace: openshift-migration

Optional: Returns the number of images.
Optional: Returns the number, kind, and API version of the Kubernetes resources.
Optional: Returns the PV capacity.
Returns a list of image names. The default is false so that the output is not excessively long.
Optional: Specify the maximum number of image names to return if listImages is true.

10.2.6. MigCluster

The MigCluster CR defines a host, local, or remote cluster.

apiVersion: migration.openshift.io/v1alpha1
class: MigCluster
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <host_cluster>
  namespace: openshift-migration
spec:
  isHostCluster: true
  # The 'azureResourceGroup' parameter is relevant only for Microsoft Azure.
  azureResourceGroup: <azure_resource_group>
  caBundle: <ca_bundle_base64>
  insecure: false
  refresh: false
  # The 'restartRestic' parameter is relevant for a source cluster.
  restartRestic: true
  # The following parameters are relevant for a remote cluster.
  exposedRegistryPath: <registry_route>
  url: <destination_cluster_url>
  serviceAccountSecretRef:
    name: <source_secret>
    namespace: openshift-config

Update the cluster name if the migration-controller pod is not running on this cluster.
The migration-controller pod runs on this cluster if true.
Microsoft Azure only: Specify the resource group.
Optional: If you created a certificate bundle for self-signed CA certificates and if the insecure parameter value is false, specify the base64-encoded certificate bundle.
5  Set to **true** to disable SSL verification.

6  Set to **true** to validate the cluster.

7  Set to **true** to restart the Restic pods on the source cluster after the Stage pods are created.

8  Remote cluster and direct image migration only: Specify the exposed secure registry path.

9  Remote cluster only: Specify the URL.

10 Remote cluster only: Specify the name of the **Secret** CR.

### 10.2.7. MigHook

The MigHook CR defines a migration hook that runs custom code at a specified stage of the migration. You can create up to four migration hooks. Each hook runs during a different phase of the migration.

You can configure the hook name, runtime duration, a custom image, and the cluster where the hook will run.

The migration phases and namespaces of the hooks are configured in the MigPlan CR.

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigHook
metadata:
  generateName: <hook_name_prefix>  
  name: <mighook>  
  namespace: openshift-migration
spec:
  activeDeadlineSeconds: 1800  
  custom: false  
  image: <hook_image>  
  playbook: <ansible_playbook_base64>  
  targetCluster: source
```

1  Optional: A unique hash is appended to the value for this parameter so that each migration hook has a unique name. You do not need to specify the value of the **name** parameter.

2  Specify the migration hook name, unless you specify the value of the **generateName** parameter.

3  Optional: Specify the maximum number of seconds that a hook can run. The default is 1800.

4  The hook is a custom image if **true**. The custom image can include Ansible or it can be written in a different programming language.

5  Specify the custom image, for example, **quay.io/konveyor/hook-runner:latest**. Required if **custom** is **true**.

6  Base64-encoded Ansible playbook. Required if **custom** is **false**.

7  Specify the cluster on which the hook will run. Valid values are **source** or **destination**.

### 10.2.8. MigMigration
The **MigMigration** CR runs a **MigPlan** CR.

You can configure a **Migmigration** CR to run a stage or incremental migration, to cancel a migration in progress, or to roll back a completed migration.

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
  - controller-tools.k8s.io: "1.0"
  name: <migmigration>
  namespace: openshift-migration
spec:
canceled: false 1
rollback: false 2
stage: false 3
quiescePods: true 4
keepAnnotations: true 5
verify: false 6
migPlanRef:
  name: <migplan>
  namespace: openshift-migration
```

1. Set to **true** to cancel a migration in progress.
2. Set to **true** to roll back a completed migration.
3. Set to **true** to run a stage migration. Data is copied incrementally and the pods on the source cluster are not stopped.
4. Set to **true** to stop the application during migration. The pods on the source cluster are scaled to 0 after the **Backup** stage.
5. Set to **true** to retain the labels and annotations applied during the migration.
6. Set to **true** to check the status of the migrated pods on the destination cluster are checked and to return the names of pods that are not in a **Running** state.

### 10.2.9. MigPlan

The **MigPlan** CR defines the parameters of a migration plan.

You can configure destination namespaces, hook phases, and direct or indirect migration.

**NOTE**

By default, a destination namespace has the same name as the source namespace. If you configure a different destination namespace, you must ensure that the namespaces are not duplicated on the source or the destination clusters because the UID and GID ranges are copied during migration.

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
```
The migration has completed if `true`. You cannot create another `MigMigration` CR for this `MigPlan` CR.

Optional: You can specify up to four migration hooks. Each hook must run during a different migration phase.

Optional: Specify the namespace in which the hook will run.

Optional: Specify the migration phase during which a hook runs. One hook can be assigned to one phase. Valid values are `PreBackup`, `PostBackup`, `PreRestore`, and `PostRestore`.

Optional: Specify the name of the `MigHook` CR.

Optional: Specify the namespace of `MigHook` CR.

Optional: Specify a service account with `cluster-admin` privileges.

Direct image migration is disabled if `true`. Images are copied from the source cluster to the replication repository and from the replication repository to the destination cluster.

Direct volume migration is disabled if `true`. PVs are copied from the source cluster to the replication repository and from the replication repository to the destination cluster.

Specify one or more source namespaces. If you specify only the source namespace, the destination namespace is the same.
Specify the destination namespace if it is different from the source namespace.

The MigPlan CR is validated if true.

### 10.2.10. MigStorage

The MigStorage CR describes the object storage for the replication repository.

Amazon Web Services (AWS), Microsoft Azure, Google Cloud Storage, Multi-Cloud Object Gateway, and generic S3-compatible cloud storage are supported.

AWS and the snapshot copy method have additional parameters.

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: MigStorage
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migstorage>
  namespace: openshift-migration
spec:
  backupStorageProvider: <backup_storage_provider>  
  volumeSnapshotProvider: <snapshot_storage_provider>  
  backupStorageConfig:
    awsBucketName: <bucket>  
    awsRegion: <region>  
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret>  
    awsKmsKeyId: <key_id>  
    awsPublicUrl: <public_url>  
    awsSignatureVersion: <signature_version>  
  volumeSnapshotConfig:
    awsRegion: <region>  
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret>  
    refresh: false
```

1. Specify the storage provider.
2. Snapshot copy method only: Specify the storage provider.
3. AWS only: Specify the bucket name.
4. AWS only: Specify the bucket region, for example, us-east-1.
5. Specify the name of the Secret CR that you created for the storage.
6. AWS only: If you are using the AWS Key Management Service, specify the unique identifier of the key.
AWS only: If you granted public access to the AWS bucket, specify the bucket URL.

AWS only: Specify the AWS signature version for authenticating requests to the bucket, for example, 4.

Snapshot copy method only: Specify the geographical region of the clusters.

Snapshot copy method only: Specify the name of the Secret CR that you created for the storage.

Set to true to validate the cluster.

10.3. LOGS AND DEBUGGING TOOLS

This section describes logs and debugging tools that you can use for troubleshooting.

10.3.1. Viewing migration plan resources

You can view migration plan resources to monitor a running migration or to troubleshoot a failed migration by using the MTC web console and the command line interface (CLI).

Procedure

1. In the MTC web console, click Migration Plans.
2. Click the Migrations number next to a migration plan to view the Migrations page.
3. Click a migration to view the Migration details.
4. Expand Migration resources to view the migration resources and their status in a tree view.

NOTE
To troubleshoot a failed migration, start with a high-level resource that has failed and then work down the resource tree towards the lower-level resources.

5. Click the Options menu next to a resource and select one of the following options:

   - **Copy oc describe command** copies the command to your clipboard.
     - Log in to the relevant cluster and then run the command. The conditions and events of the resource are displayed in YAML format.

   - **Copy oc logs command** copies the command to your clipboard.
     - Log in to the relevant cluster and then run the command. If the resource supports log filtering, a filtered log is displayed.

   - **View JSON** displays the resource data in JSON format in a web browser. The data is the same as the output for the oc get <resource> command.

10.3.2. Viewing a migration plan log
You can view an aggregated log for a migration plan. You use the MTC web console to copy a command to your clipboard and then run the command from the command line interface (CLI).

The command displays the filtered logs of the following pods:

- Migration Controller
- Velero
- Restic
- Rsync
- Stunnel
- Registry

Procedure

1. In the MTC web console, click Migration Plans.

2. Click the Migrations number next to a migration plan.

3. Click View logs.

4. Click the Copy icon to copy the `oc logs` command to your clipboard.

5. Log in to the relevant cluster and enter the command on the CLI. The aggregated log for the migration plan is displayed.

10.3.3. Using the migration log reader

You can use the migration log reader to display a single filtered view of all the migration logs.

Procedure

1. Get the `mig-log-reader` pod:

   $ oc -n openshift-migration get pods | grep log

2. Enter the following command to display a single migration log:

   $ oc -n openshift-migration logs -f <mig-log-reader-pod> -c color

   The `-c plain` option displays the log without colors.

10.3.4. Accessing performance metrics

The MigrationController custom resource (CR) records metrics and pulls them into on-cluster monitoring storage. You can query the metrics by using Prometheus Query Language (PromQL) to diagnose migration performance issues. All metrics are reset when the Migration Controller pod restarts.

You can access the performance metrics and run queries by using the OpenShift Container Platform web console.
Procedure

1. In the OpenShift Container Platform web console, click Monitoring → Metrics.

2. Enter a PromQL query, select a time window to display, and click Run Queries.
   If your web browser does not display all the results, use the Prometheus console.

10.3.4.1. Provided metrics

The MigrationController custom resource (CR) provides metrics for the MigMigration CR count and for its API requests.

10.3.4.1.1. cam_app_workload_migrations

This metric is a count of MigMigration CRs over time. It is useful for viewing alongside the mtc_client_request_count and mtc_client_request_elapsed metrics to collate API request information with migration status changes. This metric is included in Telemetry.

Table 10.1. cam_app_workload_migrations metric

<table>
<thead>
<tr>
<th>Queryable label name</th>
<th>Sample label values</th>
<th>Label description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>running, idle, failed, completed</td>
<td>Status of the MigMigration CR</td>
</tr>
<tr>
<td>type</td>
<td>stage, final</td>
<td>Type of the MigMigration CR</td>
</tr>
</tbody>
</table>

10.3.4.1.2. mtc_client_request_count

This metric is a cumulative count of Kubernetes API requests that MigrationController issued. It is not included in Telemetry.

Table 10.2. mtc_client_request_count metric

<table>
<thead>
<tr>
<th>Queryable label name</th>
<th>Sample label values</th>
<th>Label description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster</td>
<td><a href="https://migcluster-url:443">https://migcluster-url:443</a></td>
<td>Cluster that the request was issued against</td>
</tr>
<tr>
<td>component</td>
<td>MigPlan, MigCluster</td>
<td>Sub-controller API that issued request</td>
</tr>
<tr>
<td>function</td>
<td>(*ReconcileMigPlan).Reconcile</td>
<td>Function that the request was issued from</td>
</tr>
<tr>
<td>kind</td>
<td>SecretList, Deployment</td>
<td>Kubernetes kind the request was issued for</td>
</tr>
</tbody>
</table>

10.3.4.1.3. mtc_client_request_elapsed
This metric is a cumulative latency, in milliseconds, of Kubernetes API requests that *MigrationController* issued. It is not included in Telemetry.

Table 10.3. mtc_client_request_elapsed metric

<table>
<thead>
<tr>
<th>Queryable label name</th>
<th>Sample label values</th>
<th>Label description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster</td>
<td><a href="https://cluster-url.com:443">https://cluster-url.com:443</a></td>
<td>Cluster that the request was issued against</td>
</tr>
<tr>
<td>component</td>
<td>migplan, migcluster</td>
<td>Sub-controller API that issued request</td>
</tr>
<tr>
<td>function</td>
<td>(*ReconcileMigPlan).Reconcile</td>
<td>Function that the request was issued from</td>
</tr>
<tr>
<td>kind</td>
<td>SecretList, Deployment</td>
<td>Kubernetes resource that the request was issued for</td>
</tr>
</tbody>
</table>

### 10.3.4.1.4. Useful queries

The table lists some helpful queries that can be used for monitoring performance.

Table 10.4. Useful queries

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtc_client_request_count</td>
<td>Number of API requests issued, sorted by request type</td>
</tr>
<tr>
<td>sum(mtc_client_request_count)</td>
<td>Total number of API requests issued</td>
</tr>
<tr>
<td>mtc_client_request_elapsed</td>
<td>API request latency, sorted by request type</td>
</tr>
<tr>
<td>sum(mtc_client_request_elapsed)</td>
<td>Total latency of API requests</td>
</tr>
<tr>
<td>sum(mtc_client_request_elapsed) / sum(mtc_client_request_count)</td>
<td>Average latency of API requests</td>
</tr>
<tr>
<td>mtc_client_request_elapsed / mtc_client_request_count</td>
<td>Average latency of API requests, sorted by request type</td>
</tr>
<tr>
<td>cam_app_workload_migrations{status=&quot;running&quot;} * 100</td>
<td>Count of running migrations, multiplied by 100 for easier viewing alongside request counts</td>
</tr>
</tbody>
</table>

### 10.3.5. Using the must-gather tool

You can collect logs, metrics, and information about MTC custom resources by using the *must-gather* tool.
The **must-gather** data must be attached to all customer cases.

You can collect data for a one-hour or a 24-hour period and view the data with the Prometheus console.

**Prerequisites**

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

**Procedure**

1. Navigate to the directory where you want to store the **must-gather** data.

2. Run the `oc adm must-gather` command for one of the following data collection options:
   - To collect data for the past hour:
     ```
     $ oc adm must-gather --image=registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7
     ```
     The data is saved as **must-gather/must-gather.tar.gz**. You can upload this file to a support case on the Red Hat Customer Portal.
   - To collect data for the past 24 hours:
     ```
     $ oc adm must-gather --image=registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7 \
     -- /usr/bin/gather_metrics_dump
     ```
     This operation can take a long time. The data is saved as **must-gather/metrics/prom_data.tar.gz**.

**Viewing metrics data with the Prometheus console**

You can view the metrics data with the Prometheus console.

**Procedure**

1. Decompress the **prom_data.tar.gz** file:
   ```
   $ tar -xvzf must-gather/metrics/prom_data.tar.gz
   ```

2. Create a local Prometheus instance:
   ```
   $ make prometheus-run
   ```
   The command outputs the Prometheus URL.

**Output**

- Started Prometheus on http://localhost:9090
3. Launch a web browser and navigate to the URL to view the data by using the Prometheus web console.

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

   $ make prometheus-cleanup

10.3.6. Debugging Velero resources with the Velero CLI tool

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

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

**Syntax**

Use the `oc exec` command to run a Velero CLI command:

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

**Example**

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

**Help option**

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

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

**Describe command**

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

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

**Example**

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

**Logs command**

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

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

**Example**

```bash
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \
  restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```
10.3.7. Debugging a partial migration failure

You can debug a partial migration failure warning message by using the Velero CLI to examine the `Restore` custom resource (CR) logs.

A partial failure occurs when Velero encounters an issue that does not cause a migration to fail. For example, if a custom resource definition (CRD) is missing or if there is a discrepancy between CRD versions on the source and target clusters, the migration completes but the CR is not created on the target cluster.

Velero logs the issue as a partial failure and then processes the rest of the objects in the `Backup` CR.

**Procedure**

1. Check the status of a `MigMigration` CR:

   
   ```
   $ oc get migmigration <migmigration> -o yaml
   ```

   **Example output**

   ```
   status:
   conditions:
   - category: Warn
durable: true
   lastTransitionTime: "2021-01-26T20:48:40Z"
   message: 'Final Restore openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf: partially failed on destination cluster'
status: "True"
type: VeleroFinalRestorePartiallyFailed
- category: Advisory
durable: true
lastTransitionTime: "2021-01-26T20:48:42Z"
message: The migration has completed with warnings, please look at `Warn` conditions.
reason: Completed
status: "True"
type: SucceededWithWarnings
   ```

2. Check the status of the `Restore` CR by using the Velero `describe` command:

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

   **Example output**

   ```
   Phase: PartiallyFailed (run `velero restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf` for more information)
   ```

   **Errors:**

   ```
   Velero: <none>
Cluster: <none>
Namespaces:
   migration-example: error restoring example.com/migration-example/migration-example: the server could not find the requested resource
   ```
3. Check the **Restore** CR logs by using the Velero **logs** command:

```
$ oc -n {namespace} exec deployment/velero -c velero -- ./velero 
  restore logs <restore>
```

**Example output**

```
time="2021-01-26T20:48:37Z" level=info msg="Attempting to restore migration-example: migration-example" logSource="pkg/restore/restore.go:1107" restore=openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf

time="2021-01-26T20:48:37Z" level=info msg="error restoring migration-example: the server could not find the requested resource" logSource="pkg/restore/restore.go:1170" restore=openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

The **Restore** CR log error message, **the server could not find the requested resource**, indicates the cause of the partially failed migration.

### 10.3.8. Using MTC custom resources for troubleshooting

You can check the following Migration Toolkit for Containers (MTC) custom resources (CRs) to troubleshoot a failed migration:

- **MigCluster**
- **MigStorage**
- **MigPlan**
- **BackupStorageLocation**
  
  The **BackupStorageLocation** CR contains a **migrationcontroller** label to identify the MTC instance that created the CR:

  ```
  labels:
  migrationcontroller: ebe13bee-c803-47d0-a9e9-83f380328b93
  ```

- **VolumeSnapshotLocation**
  
  The **VolumeSnapshotLocation** CR contains a **migrationcontroller** label to identify the MTC instance that created the CR:

  ```
  labels:
  migrationcontroller: ebe13bee-c803-47d0-a9e9-83f380328b93
  ```

- **MigMigration**
- **Backup**
  
  MTC changes the reclaim policy of migrated persistent volumes (PVs) to **Retain** on the target cluster. The **Backup** CR contains an **openshift.io/orig-reclaim-policy** annotation that indicates the original reclaim policy. You can manually restore the reclaim policy of the migrated PVs.

- **Restore**
1. List the **MigMigration** CRs in the **openshift-migration** namespace:

   ```bash
   $ oc get migmigration -n openshift-migration
   
   Example output
   
<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>88435fe0-c9f8-11e9-85e6-5d593ce65e10</td>
<td>6m42s</td>
</tr>
</tbody>
</table>
   
2. Inspect the **MigMigration** CR:

   ```bash
   $ oc describe migmigration 88435fe0-c9f8-11e9-85e6-5d593ce65e10 -n openshift-migration
   ```

   The output is similar to the following examples.

### MigMigration example output

```yaml
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 output that describes the PV data

```yaml
apiVersion: velero.io/v1
kind: Backup
metadata:
```
CHAPTER 10. TROUBLESHOOTING

annoucements:
  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
  openshift.io/orig-reclaim-policy: delete
creationTimestamp: "2019-08-29T01:03:15Z"
generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
generation: 1
labels:
  app.kubernetes.io/part-of: migration
  migmigation: 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 output that describes the Kubernetes resources

```yaml
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-06fa9f68c88
  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-06fa9f68c88
    migration-final-restore: e18252c9-c927-11e9-825a-06fa9f68c88
    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-06fa9f68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources: null
  - 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
```

10.4. COMMON ISSUES AND CONCERNS

This section describes common issues and concerns that can cause issues during migration.

10.4.1. Direct volume migration does not complete

If direct volume migration does not complete, the target cluster might not have the same `node-selector` annotations as the source cluster.
Migration Toolkit for Containers (MTC) migrates namespaces with all annotations to preserve security context constraints and scheduling requirements. During direct volume migration, MTC creates Rsync transfer pods on the target cluster in the namespaces that were migrated from the source cluster. If a target cluster namespace does not have the same annotations as the source cluster namespace, the Rsync transfer pods cannot be scheduled. The Rsync pods remain in a **Pending** state.

You can identify and fix this issue by performing the following procedure.

**Procedure**

1. Check the status of the MigMigration CR:

   ```
   $ oc describe migmigration <pod> -n openshift-migration
   ```

   The output includes the following status message:

   **Example output**

   ```
   Some or all transfer pods are not running for more than 10 mins on destination cluster
   ```

2. On the source cluster, obtain the details of a migrated namespace:

   ```
   $ oc get namespace <namespace> -o yaml
   ```

   Specify the migrated namespace.

3. On the target cluster, edit the migrated namespace:

   ```
   $ oc edit namespace <namespace>
   ```

4. Add the missing `openshift.io/node-selector` annotations to the migrated namespace as in the following example:

   ```
   apiVersion: v1
   kind: Namespace
   metadata:
   annotations:
     openshift.io/node-selector: "region=east"
   ...
   ```

5. Run the migration plan again.

**10.4.2. Error messages and resolutions**

This section describes common error messages you might encounter with the Migration Toolkit for Containers (MTC) and how to resolve their underlying causes.

**10.4.2.1. CA certificate error displayed when accessing the MTC console for the first time**

If a **CA certificate error** message is displayed the first time you try to access the MTC console, the likely cause is the use of self-signed CA certificates in one of the clusters.
To resolve this issue, navigate to the `oauth-authorization-server` URL displayed in the error message and accept the certificate. To resolve this issue permanently, add the certificate to the trust store of your web browser.

If an **Unauthorized** message is displayed after you have accepted the certificate, navigate to the MTC console and refresh the web page.

### 10.4.2.2. OAuth timeout error in the MTC console

If a **connection has timed out** message is displayed in the MTC console after you have accepted a self-signed certificate, the causes are likely to be the following:

- Interrupted network access to the OAuth server
- Interrupted network access to the OpenShift Container Platform console
- Proxy configuration that blocks access to the `oauth-authorization-server` URL. See [MTC console inaccessible because of OAuth timeout error](#) for details.

To determine the cause of the timeout:

- Inspect the MTC console web page with a browser web inspector.
- Check the **Migration UI** pod log for errors.

### 10.4.2.3. Certificate signed by unknown authority error

If you use a self-signed certificate to secure a cluster or a replication repository for the Migration Toolkit for Containers (MTC), 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 MTC 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:

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

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.

### 10.4.2.4. Backup storage location errors in the Velero pod log

If a **Velero Backup** custom resource contains a reference to a backup storage location (BSL) that does not exist, the **Velero** pod log might display the following error messages:

```bash
$ oc logs <Velero_Pod> -n openshift-migration
```

**Example output**
You can ignore these error messages. A missing BSL cannot cause a migration to fail.

### 10.4.2.5. Pod volume backup timeout error in the Velero pod log

If a migration fails because Restic times out, the following error is displayed in the **Velero** pod log.

```yaml
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 **Migration Toolkit for Containers Operator**.
3. In the **MigrationController** tab, click **migration-controller**.
4. In the **YAML** tab, update the following parameter value:

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

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

5. Click **Save**.

### 10.4.2.6. Restic verification errors in the MigMigration custom resource

If data verification fails when migrating a persistent volume with the file system data copy method, the following error is displayed in the **MigMigration** CR.

**Example output**

```yaml
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
status: "True"
type: ResticVerifyErrors
```

|   1   |   2   |
1. The error message identifies the **Restore** CR name.

2. **ResticVerifyErrors** is a general error warning type that includes verification errors.

**NOTE**

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

You can check the **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 persistent volume with **PodVolumeRestore** errors.

   **Example output**

   ```
   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.

   **Example output**

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

4. View the **Restic** pod log to locate the errors:

   $ oc logs -f <restic-nr2v5>
10.4.2.7. Restic permission error when migrating from NFS storage with root_squash enabled

If you are migrating data from NFS storage and root_squash is enabled, Restic maps to nfsnobody and does not have permission to perform the migration. The following error is displayed in the Restic pod log.

Example output

```
backup=openshift-migration/<backup_id> controller=pod-volume-backup error="fork/exec /usr/bin/restic: permission denied" error.file="/go/src/github.com/vmware-tanzu/velero/pkg/controller/pod_volume_backup_controller.go:280"
error.function="github.com/vmware-tanzu/velero/pkg/controller.(*podVolumeBackupController).processBackup"
logSource="pkg/controller/pod_volume_backup_controller.go:280" name=<backup_id>
namespace=openshift-migration
```

You can resolve this issue by creating a supplemental group for Restic and adding the group ID to the MigrationController CR manifest.

Procedure

1. Create a supplemental group for Restic on the NFS storage.

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

3. Add the restic_supplemental_groups parameter to the MigrationController CR manifest on the source and target clusters:

   ```
   spec:
   restic_supplemental_groups: <group_id>
   ```

   Specify the supplemental group ID.

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

10.5. ROLLING BACK A MIGRATION

You can roll back a migration by using the MTC web console or the CLI.

You can also roll back a migration manually.

10.5.1. Rolling back a migration by using the MTC web console

You can roll back a migration by using the Migration Toolkit for Containers (MTC) web console.
NOTE

The following resources remain in the migrated namespaces for debugging after a failed direct volume migration (DVM):

- Config maps (source and destination clusters)
- Secret CRs (source and destination clusters)
- Rsync CRs (source cluster)

These resources do not affect rollback. You can delete them manually.

If you later run the same migration plan successfully, the resources from the failed migration are deleted automatically.

If your application was stopped during a failed migration, you must roll back the migration to prevent data corruption in the persistent volume.

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

Procedure

1. In the MTC web console, click Migration plans.

2. Click the Options menu beside a migration plan and select Rollback under Migration.

3. Click Rollback and wait for rollback to complete.
   In the migration plan details, Rollback succeeded is displayed.

4. Verify that rollback was successful in the OpenShift Container Platform web console of the source cluster:
   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.

10.5.2. Rolling back a migration from the command line interface

You can roll back a migration by creating a MigMigration custom resource (CR) from the command line interface.
NOTE

The following resources remain in the migrated namespaces for debugging after a failed direct volume migration (DVM):

- Config maps (source and destination clusters)
- `Secret` CRs (source and destination clusters)
- `Rsync` CRs (source cluster)

These resources do not affect rollback. You can delete them manually.

If you later run the same migration plan successfully, the resources from the failed migration are deleted automatically.

If your application was stopped during a failed migration, you must roll back the migration to prevent data corruption in the persistent volume.

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

Procedure

1. Create a `MigMigration` CR based on the following example:

   ```
   $ cat << EOF | oc apply -f -
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigMigration
   metadata:
     labels:
       controller-tools.k8s.io: "1.0"
     name: <migmigration>
     namespace: openshift-migration
   spec:
     ...  
     rollback: true  
     ...
   migPlanRef:
     name: <migplan>  
     namespace: openshift-migration
   EOF
   ```

   1. Specify the name of the associated `MigPlan` CR.

2. In the MTC web console, verify that the migrated project resources have been removed from the target cluster.

3. Verify that the migrated project resources are present in the source cluster and that the application is running.

10.5.3. Rolling back a migration manually
You can roll back a failed migration manually by deleting the **stage** pods and unquiescing the application.

If you run the same migration plan successfully, the resources from the failed migration are deleted automatically.

**NOTE**

The following resources remain in the migrated namespaces after a failed direct volume migration (DVM):

- Config maps (source and destination clusters)
- **Secret** CRs (source and destination clusters)
- **Rsync** CRs (source cluster)

These resources do not affect rollback. You can delete them manually.

**Procedure**

1. Delete the **stage** pods on all clusters:

   ```bash
   $ oc delete $(oc get pods -l migration.openshift.io/is-stage-pod -n <namespace>)
   ```

   Namespaces specified in the **MigPlan** CR.

2. Unquiesce the application on the source cluster by scaling the replicas to their pre-migration number:

   ```bash
   $ oc scale deployment <deployment> --replicas=<premigration_replicas>
   ```

   The **migration.openshift.io/preQuiesceReplicas** annotation in the **Deployment** CR displays the pre-migration number of replicas:

   ```
   apiVersion: extensions/v1beta1
   kind: Deployment
   metadata:
     annotations:
       deployment.kubernetes.io/revision: "1"
       migration.openshift.io/preQuiesceReplicas: "1"
   ```

3. Verify that the application pods are running on the source cluster:

   ```bash
   $ oc get pod -n <namespace>
   ```

**Additional resources**

- Deleting Operators from a cluster using the web console