OpenShift Container Platform 4.7 Migration Toolkit for Containers

Migrating to OpenShift Container Platform 4
Abstract

This document provides instructions for migrating your OpenShift Container Platform cluster from version 3 to version 4, and also for migrating from an earlier OpenShift Container Platform 4 release to the latest version.
## Table of Contents

### CHAPTER 1. ABOUT THE {MTC-FULL}

- Additional resources

- 1.1. MIGRATION TOOLKIT FOR CONTAINERS WORKFLOW
- 1.2. ABOUT DATA COPY METHODS
  - 1.2.1. File system copy method
  - 1.2.2. Snapshot copy method
- 1.3. DIRECT VOLUME MIGRATION AND DIRECT IMAGE MIGRATION

### CHAPTER 2. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS

- 2.1. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENShift CONTAINER PLATFORM 4
- 2.2. CONFIGURING A REPLICATION REPOSITORY
  - Additional resources
  - 2.2.1. Configuring Multi-Cloud Object Gateway
    - 2.2.1.1. Installing the OpenShift Container Storage Operator
    - 2.2.1.2. Creating the Multi-Cloud Object Gateway storage bucket
  - 2.2.2. Configuring Amazon Web Services S3
  - 2.2.3. Configuring Google Cloud Platform
  - 2.2.4. Configuring Microsoft Azure Blob

### CHAPTER 3. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS IN A RESTRICTED NETWORK ENVIRONMENT

- 3.1. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENShift CONTAINER PLATFORM 4
- 3.2. CONFIGURING A REPLICATION REPOSITORY
  - 3.2.1. Configuring Multi-Cloud Object Gateway
    - 3.2.1.1. Installing the OpenShift Container Storage Operator
    - 3.2.1.2. Creating the Multi-Cloud Object Gateway storage bucket

### CHAPTER 4. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS

- 4.1. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENShift CONTAINER PLATFORM 4
- 4.2. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENShift CONTAINER PLATFORM 3
- 4.3. UPGRADING MTC 1.3 TO 1.4

### CHAPTER 5. PREMIGRATION CHECKLISTS

- 5.1. CLUSTER HEALTH CHECKLIST
- 5.2. SOURCE CLUSTER CHECKLIST
- 5.3. TARGET CLUSTER CHECKLIST

### CHAPTER 6. MIGRATING YOUR APPLICATIONS

- 6.1. MIGRATION PREREQUISITES
  - 6.1.1. Configuring an Stunnel proxy for direct volume migration
- 6.2. MIGRATING YOUR APPLICATIONS BY USING THE MTC WEB CONSOLE
  - 6.2.1. Launching the MTC web console
  - 6.2.2. Adding a cluster to the MTC web console
  - 6.2.3. Adding a replication repository to the MTC web console
  - 6.2.4. Creating a migration plan in the MTC web console
    - Additional resources for persistent volume copy methods
  - 6.2.5. Running a migration plan in the MTC web console
CHAPTER 7. ADVANCED MIGRATION OPTIONS ......................................................... 40
Additional resources ................................................. 40
7.1. MTC CUSTOM RESOURCES ......................................................... 40
  7.1.1. About MTC custom resources ......................................................... 40
  7.1.2. MTC custom resource manifests ......................................................... 41
    7.1.2.1. DirectImageMigration ......................................................... 41
    7.1.2.2. DirectImageStreamMigration ......................................................... 42
    7.1.2.3. DirectVolumeMigration ......................................................... 42
    7.1.2.4. DirectVolumeMigrationProgress ......................................................... 43
    7.1.2.5. MigAnalytic ......................................................... 43
    7.1.2.6. MigCluster ......................................................... 44
    7.1.2.7. MigHook ......................................................... 44
    7.1.2.8. MigMigration ......................................................... 45
    7.1.2.9. MigPlan ......................................................... 46
    7.1.2.10. MigStorage ......................................................... 47
  7.2. MIGRATING YOUR APPLICATIONS WITH THE MTC API ......................................................... 48
    7.2.1. About migrating applications ......................................................... 49
      7.2.1.1. Terminology ......................................................... 49
    7.2.2. Migration prerequisites ......................................................... 51
      7.2.2.1. Configuring an Stunnel proxy for direct volume migration ......................................................... 52
      7.2.3. Migrating your applications from the command line ......................................................... 52
  7.3. MIGRATION HOOKS ......................................................... 57
    7.3.1. About migration hooks ......................................................... 58
    7.3.2. Writing an Ansible playbook for a migration hook ......................................................... 58
      7.3.2.1. Ansible modules ......................................................... 58
      7.3.2.2. Environment variables ......................................................... 59
  7.4. INCREASING LIMITS FOR LARGE MIGRATIONS ......................................................... 60
  7.5. EXCLUDING RESOURCES FROM A MIGRATION PLAN ......................................................... 61

CHAPTER 8. TROUBLESHOOTING ......................................................... 63
8.1. LOGS AND DEBUGGING TOOLS ......................................................... 63
  8.1.1. Viewing migration plan resources ......................................................... 63
  8.1.2. Viewing a migration plan log ......................................................... 63
  8.1.3. Using the migration log reader ......................................................... 64
  8.1.4. Using the must-gather tool ......................................................... 64
  8.1.5. Using the Velero CLI to debug Backup and Restore CRs ......................................................... 65
    8.1.5.1. Velero command syntax ......................................................... 65
    8.1.5.2. Help command ......................................................... 66
    8.1.5.3. Describe command ......................................................... 66
    8.1.5.4. Logs command ......................................................... 66
  8.1.6. Debugging a partial migration failure ......................................................... 66
  8.1.7. Using MTC custom resources for troubleshooting ......................................................... 68
    Additional resources for debugging tools ......................................................... 71
  8.2. COMMON ISSUES AND CONCERNS ......................................................... 71
    8.2.1. Direct volume migration does not complete ......................................................... 71
    8.2.2. Error messages and resolutions ......................................................... 72
      8.2.2.1. CA certificate error displayed when accessing the MTC console for the first time ......................................................... 72
      8.2.2.2. OAuth timeout error in the MTC console ......................................................... 72
      8.2.2.3. Certificate signed by unknown authority error ......................................................... 73
      8.2.2.4. Backup storage location errors in the Velero pod log ......................................................... 73
8.2.2.5. Pod volume backup timeout error in the Velero pod log 73
8.2.2.6. Restic verification errors in the MigMigration custom resource 74
8.2.2.7. Restic permission error when migrating from NFS storage with root_squash enabled 75
8.2.3. Known issues 76

8.3. ROLLING BACK A MIGRATION 77
8.3.1. Rolling back a migration by using the MTC web console 77
8.3.2. Rolling back a migration from the command line interface 78
CHAPTER 1. ABOUT THE {MTC-FULL}

The Migration Toolkit for Containers (MTC) web console and API, based on Kubernetes custom resources, enable you to migrate stateful application workloads at the granularity of a namespace.

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.

Additional resources

- Advanced migration options
  Automate your migration with migration hooks and the MTC API.

  Configure your migration plan to exclude resources, support large-scale migrations, and enable automatic PV resizing for direct volume migration.

Additional resources

- Advanced migration options for automating migration

1.1. MIGRATION TOOLKIT FOR CONTAINERS WORKFLOW

You can migrate Kubernetes resources, persistent volume data, and internal container images to OpenShift Container Platform 4.7 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.
   
   ![Diagram of replication process](image)

   **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.

   ![Diagram of direct migration](image)

   - **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.

   ![Diagram of move process](image)

6. Run the migration plan, with one of the following options:
**Stage** (optional) copies data to the target cluster without stopping the application. Staging can be run multiple times so that most of the data is copied to the target before migration. This minimizes the duration of the migration and the application downtime.

**Migrate** stops the application on the source cluster and recreates its resources on the target cluster. Optionally, you can migrate the workload without stopping the application.

### 1.2. 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.

#### 1.2.1. File system copy method

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

**Table 1.1. File system copy method summary**
### 1.2.2. 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.

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

**Table 1.2. Snapshot copy method summary**

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>● Faster than the file system copy method.</td>
<td>● Cloud provider must support snapshots.</td>
</tr>
<tr>
<td>● Cloud provider must support snapshots.</td>
<td>● Clusters must be on the same cloud provider.</td>
</tr>
<tr>
<td>● Clusters must be on the same cloud provider.</td>
<td>● Clusters must be in the same location or region.</td>
</tr>
<tr>
<td>● Clusters must be in the same location or region.</td>
<td>● Clusters must have the same storage class.</td>
</tr>
<tr>
<td>● Clusters must have the same storage class.</td>
<td>● Storage class must be compatible with snapshots.</td>
</tr>
<tr>
<td>● Storage class must be compatible with snapshots.</td>
<td>● Does not support direct volume migration.</td>
</tr>
<tr>
<td>● Does not support direct volume migration.</td>
<td></td>
</tr>
</tbody>
</table>

### 1.3. DIRECT VOLUME MIGRATION AND DIRECT IMAGE MIGRATION

You can use *direct image migration* and *direct volume migration* to migrate images and data directly from the source cluster to the target cluster.

Direct migration has significant performance benefits because it skips 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.

Direct migration uses *Rsync* to transfer the files.

**NOTE**

Direct image migration and direct volume migration have additional prerequisites.
CHAPTER 2. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS

You can install the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4.

IMPORTANT

You must install the same MTC version on all clusters.

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.

2.1. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSHIFT CONTAINER PLATFORM 4

You can install the MTC Operator on OpenShift Container Platform 4 by using the OpenShift Container Platform web console.

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.

NOTE

Do not change the subscription approval option to Automatic. The Migration Toolkit for Containers version must be the same on the source and the target clusters.

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. If you do not want to run the MTC web console and the Migration Controller pod on the cluster, update the following parameters in the migration-controller custom resource manifest:

```
spec:
...
```
2.2. 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. You can select a method that is suited for your environment and is supported by your storage provider.

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.

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

Additional resources

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

2.2.1. 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).

2.2.1.1. Installing the OpenShift Container Storage Operator

You can install the OpenShift Container Storage Operator from OperatorHub.
### Procedure

1. In the OpenShift Container Platform web console, click **Operators → OperatorHub**.

2. Use **Filter by keyword** (in this case, **OCS**) to find the **OpenShift Container Storage Operator**.

3. Select the **OpenShift Container Storage Operator** and click **Install**.

4. Select an **Update Channel**, **Installation Mode**, and **Approval Strategy**.

5. Click **Install**.

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

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

You can create the Multi-Cloud Object Gateway (MCG) storage bucket’s custom resources (CRs).

### Procedure

1. Log in to the OpenShift Container Platform cluster:

   ```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
   ```

   - 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
   ```
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
class: 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:

   ```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
spec:
  bucketName: <bucket> 1
   ```
Record the bucket name for adding the replication repository to the MTC web console.

9. Create the **ObjectBucketClaim** object:
   
   ```shell
   $ oc create -f obc.yml
   ```

10. Watch the resource creation process to verify that the **ObjectBucketClaim** status is **Bound**:
    
    ```shell
    $ 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:
      
      ```shell
      $ oc get route -n openshift-storage s3
      ```
    
    - S3 provider access key:
      
      ```shell
      $ oc get secret -n openshift-storage migstorage -o go-template='{{ .data.AWS_ACCESS_KEY_ID }}' | base64 --decode
      ```
    
    - S3 provider secret access key:
      
      ```shell
      $ oc get secret -n openshift-storage migstorage -o go-template='{{ .data.AWS_SECRET_ACCESS_KEY }}' | base64 --decode
      ```

2.2.2. Configuring Amazon Web Services S3

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

**Prerequisites**

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

1. Create an AWS S3 bucket:

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

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

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

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

3. Create an EC2 EBS snapshot policy:

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

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

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

**Example output**

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

1. Record the **AWS_SECRET_ACCESS_KEY** and the **AWS_ACCESS_KEY_ID** for adding the AWS repository to the MTC web console.

2.2.3. Configuring Google Cloud Platform

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

**Prerequisites**

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

**Procedure**

1. Log in to `gsutil`:

   ```
   $ gsutil init
   ```

   **Example output**

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

2. Set the **BUCKET** variable:

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

   Specify your bucket name.

3. Create a storage bucket:

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

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

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

5. Create a `velero` IAM service account:
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
    ``

### 2.2.4. 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:

   ```bash
   $ 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`

$ 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
CHAPTER 3. 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.

**IMPORTANT**

You must install the same MTC version on all clusters.

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.

### 3.1. INSTALLING THE MIGRATION TOOLKIT FOR CONTAINERS OPERATOR ON OPENSSHIFT CONTAINER PLATFORM 4

You can install the MTC Operator on OpenShift Container Platform 4 by using the OpenShift Container Platform web console.

**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. If you do not want to run the MTC web console and the Migration Controller pod on the cluster, update the following parameters in the migration-controller custom resource manifest:

   ```yaml
   spec:
   ...
   migration_controller: false
   ```
3.2. 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.

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.

3.2.1. 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).

3.2.1.1. Installing the OpenShift Container Storage Operator

You can install the OpenShift Container Storage Operator from OperatorHub.

Procedure

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

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

3.2.1.2. Creating the Multi-Cloud Object Gateway storage bucket

You can create the Multi-Cloud Object Gateway (MCG) storage bucket’s custom resources (CRs).
Procedure

1. Log in to the OpenShift Container Platform cluster:

   $ oc login -u <username>

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

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

   For a very small cluster, you can change the value to 0.1.

3. Create the NooBaa object:

   $ oc create -f noobaa.yml

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

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

   Specify the number of volumes in the persistent volume pool.

   Specify the size of the volumes, for example, 50Gi.
3 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:
  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
spec:
  bucketName: <bucket>  ①
  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:

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

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

```bash
$ 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
  ```
CHAPTER 4. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS

You can upgrade the Migration Toolkit for Containers (MTC) by using the OpenShift Container Platform web console.

**IMPORTANT**

You must ensure that you upgrade to the same MTC version on all clusters.

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

4.1. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSHIFT CONTAINER PLATFORM 4

You can upgrade the Migration Toolkit for Containers (MTC) on OpenShift Container Platform 4 by using the OpenShift Container Platform web console.

**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.

4.2. UPGRADING THE MIGRATION TOOLKIT FOR CONTAINERS ON OPENSHIFT CONTAINER PLATFORM 3

You can upgrade Migration Toolkit for Containers (MTC) on OpenShift Container Platform 3 with podman.

**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:

   $ sudo podman login registry.redhat.io

2. Download the latest operator.yml file:

   $ sudo podman cp $(sudo podman create \registry.redhat.io/rhmtc/openshift-migration-rhel7-operator:v1.4):/operator.yml ./

   1 You can specify a z-stream release, if necessary.

3. Replace the Migration Toolkit for Containers Operator:

   $ oc replace --force -f operator.yml

4. Apply the changes:

   • For MTC 1.1.2 and earlier versions, delete the Restic pods:

     $ oc delete pod <restic_pod>

   • For MTC 1.2 and later versions:

     a. Scale the migration-operator deployment to 0 to stop the deployment:

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

     b. Scale the migration-operator deployment to 1 to start the deployment and apply the changes:

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

5. Verify that the migration-operator was upgraded:

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

6. Download the latest controller-3.yml file:

   $ sudo podman cp $(sudo podman create \registry.redhat.io/rhmtc/openshift-migration-rhel7-operator:v1.4):/controller-3.yml ./

7. Create the migration-controller object:

   $ oc create -f controller-3.yml
8. If your OpenShift Container Platform version is 3.10 or earlier, set the security context constraint of the `migration-controller` service account to `anyuid` to enable direct image migration and direct volume migration:

```
$ oc adm policy add-scc-to-user anyuid -z migration-controller -n openshift-migration
```

9. Verify that the MTC pods are running:

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

10. If you have previously added the OpenShift Container Platform 3 cluster to the MTC web console, you must update the service account token in the web console because the upgrade process deletes and restores the `openshift-migration` namespace:

   a. Obtain the service account token:

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

   b. In the MTC web console, click **Clusters**.

   c. Click the Options menu next to the cluster and select **Edit**.

   d. Enter the new service account token in the **Service account token** field.

   e. Click **Update cluster** and then click **Close**.

4.3. UPGRAADING MTC 1.3 TO 1.4

If you are upgrading Migration Toolkit for Containers (MTC) version 1.3.x to 1.4, 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 have MTC 1.3 installed.
- 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:

```
$ oc get migplan <migplan> -o yaml -n openshift-migration
```

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

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

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

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

   ```
   $ oc get migplan <migplan> -o yaml -n openshift-migration
   ```
CHAPTER 5. PREMIGRATION CHECKLISTS

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

5.1. CLUSTER HEALTH CHECKLIST

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

☐ The MTC version is the same on all clusters.

☐ 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.

5.2. SOURCE CLUSTER CHECKLIST

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

   ```bash
   $ oc get pv
   ```

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

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

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

   ```bash
   $ oc get pods --all-namespaces --field-selector=status.phase=Running -o 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.

## 5.3. 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 6. MIGRATING YOUR APPLICATIONS

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

6.1. 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.

## 6.1.1. Configuring an Stunnel proxy for direct volume migration

If you are performing direct volume migration from a source cluster behind a proxy, you must configure an Stunnel proxy in the `MigrationController` custom resource (CR). Stunnel creates a transparent tunnel between the source and target clusters for the TCP connection without changing the certificates.

### NOTE

Direct volume migration supports only one proxy. The source cluster cannot access the route of the target cluster if the target cluster is also behind a proxy.

### Prerequisites

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

### Procedure

1. Log in to the cluster on which the `MigrationController` pod runs.

2. Get the `MigrationController` CR manifest:

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

3. Add the `stunnel_tcp_proxy` parameter:

   ```yaml
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: <migration-controller>
     namespace: openshift-migration
   ...
   spec:
     stunnel_tcp_proxy: <stunnel_proxy>  
   ```

   Specify the Stunnel proxy: `http://<user>:<password>@<ip_address>:<port>`.

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

5. Apply the updated manifest:

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

### 6.2. 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.
6.2.1. 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:

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

   The output resembles the following: `https://migration-openshift-migration.apps.cluster.openshift.com`.
3. Launch a browser and navigate to the 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*.

6.2.2. 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.

**Procedure**

1. Log in to the cluster.
2. Obtain the *migration-controller* service account token:
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, for example, `www.example.apps.cluster.com`. You can specify a port. The default port is `5000`.
   - **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.

### 6.2.3. 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.

**Example output**

```
eyJhbGciOiJSUzI1NiIsImtpZCI6Ij9.eyJpc3MiOiJwdWJcIiwiY2xhc3 quYWJcIiwic3BhY2UiOiJSQWJcIiwiY2FtcGxlIjoiZWRtaW4ifSwicGhvbm RciOiIwMzkuMzc2Iiwic3VzcGVzIjoiZWRtaW4ifSwicm9vdFNvbnRlbnquYXV0a mltiZGV2IjoiZWRtaW4iLCJpc3MiOiJwdWJcIiwiY2xhc3QubGlua3ktZGV2Ijoi ZWRtaW4ifSwicGhvbmRciOiIwMzkuMzc2Iiwic3VzcGVzIjoiZWRtaW4ifSwicm9v dFNvbnRlbnquYXV0a3ktZGV2IjoiZWRtaW4iLCJpc3MiOiJwdWJcIiwiY2xhc3Qub Glua3ktZGV2IjoiZWRtaW4ifSwicGhvbmRciOiIwMzkuMzc2Iiwic3VzcGVzIjoiZ WRtaW4ifSwicm9vdFNvbnRlbnquYXV0a3ktZGV2IjoiZWRtaW4iLCJpc3MiOiJwd WJcIiwiY2xhc3QubGlua3ktZGV2IjoiZWRtaW4ifSwicGhvbmRciOiIwMzkuMzcu Iiwic3VzcGVzIjoiZWRtaW4ifSwicm9vdFNvbnRlbnquYXV0a3ktZGV2IjoiZWRt aW4ifSwicGhvbmRciOiIwMzkuMzc2Iiwic3VzcGVzIjoiZWRtaW4ifSwicm9vdFNv bnr parental”}>x.x.x.x.kกระบวนการ</p>
``
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 check box 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.

6.2.4. **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, the **MigCluster** custom resource manifest of the source cluster must specify the exposed route of the internal image registry.

**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**, and click **Next**.

5. On the **Namespaces** page, select the projects to be migrated and click **Next**.

6. On the **Persistent volumes** page, click a **Migration type** for each PV:
   - The **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.
   - The **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.

7. Click Next.

8. On the Copy options page, 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.

9. 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.

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

11. Click Next.

12. 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.

13. Click Next.

14. Optional: On the Hooks page, 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
15. 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

6.2.5. Running a migration plan in the MTC web console

You can stage or 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 Stage to copy data from the source cluster to the target cluster without stopping the application.
   You can run Stage multiple times to reduce the actual migration time.

3. When you are ready to migrate the application workload, the Options menu beside a migration plan and select Migrate.

4. Optional: In the Migrate window, you can select Do not stop applications on the source cluster during migration.
5. Click Migrate.

6. 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 7. ADVANCED MIGRATION OPTIONS

This section describes advanced options for automating your migration and for modifying the migration plan.

Additional resources

- MTC workflow

7.1. MTC CUSTOM RESOURCES

This section describes the custom resources (CRs) that are used by the Migration Toolkit for Containers (MTC).

7.1.1. About MTC custom resources

The Migration Toolkit for Containers (MTC) creates the following custom resources (CRs):

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.

BackupStorageLocation (configuration, MTC cluster): Location of Velero backup objects

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

MigMigration (action, MTC cluster): Migration, created every time you stage or migrate data. Each MigMigration CR is associated with a MigPlan CR.

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

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

7.1.2. MTC custom resource manifests

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

7.1.2.1. 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:
  srcMigClusterRef:
    name: <source_cluster>
    namespace: openshift-migration
destMigClusterRef:
  name: <destination_cluster>
  namespace: openshift-migration
namespaces:  
  - <source_namespace_1>
  - <source_namespace_2>:<destination_namespace_3>
```
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.

7.1.2.2. DirectImageStreamMigration

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

```yaml
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>
```

7.1.2.3. DirectVolumeMigration

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

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: DirectVolumeMigration
metadata:
  name: <direct_volume_migration>
  namespace: openshift-migration
spec:
  createDestinationNamespaces: false 1
  deleteProgressReportingCRs: false 2
  destMigClusterRef:
    name: <host_cluster> 3
    namespace: openshift-migration
  persistentVolumeClaims:
    - name: <pvc> 4
      namespace: <pvc_namespace>
  srcMigClusterRef:
    name: <source_cluster>
    namespace: openshift-migration
```

1. 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.

### 7.1.2.4. 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
```

### 7.1.2.5. 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.

7.1.2.6. MigCluster

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

```yaml
apiVersion: migration.openshift.io/v1alpha1
kind: 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
```

1. Update the cluster name if the migration-controller pod is not running on this cluster.
2. The migration-controller pod runs on this cluster if true.
3. Microsoft Azure only: Specify the resource group.
4. 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.

7.1.2.7. MigHook

The MigHook CR defines a migration hook that runs custom code at a specified stage of the migration.
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`.

### 7.1.2.8. 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. 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.
rollback: false
stage: false
quiescePods: true
keepAnnotations: true
verify: false
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.

7.1.2.9. 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.

apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migplan>
  namespace: openshift-migration
spec:
  closed: false
  srcMigClusterRef:
    name: <source_cluster>
    namespace: openshift-migration
destMigClusterRef:
  name: <destination_cluster>
  namespace: openshift-migration
hooks: 2
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`.

### 7.1.2.10. 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
dependent: 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.
7. AWS only: If you granted public access to the AWS bucket, specify the bucket URL.
8. AWS only: Specify the AWS signature version for authenticating requests to the bucket, for example, **4**.
9. Snapshot copy method only: Specify the geographical region of the clusters.
10. Snapshot copy method only: Specify the name of the **Secret** CR that you created for the storage.
11. Set to **true** to validate the cluster.

### 7.2. MIGRATING YOUR APPLICATIONS WITH THE MTC API
This section describes how to migrate your applications with the MTC API from the command line interface (CLI).

### 7.2.1. About migrating applications

You can migrate applications from a local cluster to a remote cluster, from a remote cluster to a local cluster, and between remote clusters.

#### 7.2.1.1. Terminology

**Source cluster**
- Cluster from which the applications are migrated.

**Destination cluster**
- Cluster to which the applications are migrated.

**Replication repository**
- Object storage.
- Requires network access to all clusters.

**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**
- Volumes are copied directly from the source cluster to the destination cluster.
- Significantly faster than indirect migration.

**Direct image migration**
- Images are copied directly from the source cluster to the destination cluster.
- Significantly faster than indirect migration.

**Host cluster**
- Cluster on which the `migration-controller` pod and the web console run.
- Usually the same as the destination cluster and the local cluster but this is not a requirement.
- Does not require an exposed secure registry route for direct image migration.

**Remote cluster**
- Usually the same as the source cluster but this is not a requirement.
- Requires an exposed secure registry route for direct image migration.
- Requires a **Secret** CR containing the `migration-controller` service account token.
7.2.1.2. Mapping destination namespaces in the MigPlan custom resource (CR)

If you map destination namespaces in the MigPlan 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
```

7.2.1.3. Stage migration, migration cancellation, and migration rollback

You can create and associate multiple MigMigration custom resources (CRs) with the same MigPlan CR for the following use cases:

- To perform a stage migration, which copies all available data without stopping the application pods. Running a stage migration reduces the cutover time.
- To cancel a migration in progress.
- To roll back a completed migration.

7.2.1.4. 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:
7.2.2. 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.
7.2.2.1. Configuring an Stunnel proxy for direct volume migration

If you are performing direct volume migration from a source cluster behind a proxy, you must configure an Stunnel proxy in the MigrationController custom resource (CR). Stunnel creates a transparent tunnel between the source and target clusters for the TCP connection without changing the certificates.

**NOTE**

Direct volume migration supports only one proxy. The source cluster cannot access the route of the target cluster if the target cluster is also behind a proxy.

**Prerequisites**

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

**Procedure**

1. Log in to the cluster on which the MigrationController pod runs.
2. Get the MigrationController CR manifest:
   ```
   $ oc get migrationcontroller <migration_controller> -n openshift-migration
   ```
3. Add the `stunnel_tcp_proxy` parameter:
   ```
   apiVersion: migration.openshift.io/v1alpha1
   kind: MigrationController
   metadata:
     name: <migration-controller>
     namespace: openshift-migration
...
   spec:
     stunnel_tcp_proxy: <stunnel_proxy>  
   ```
   Specify the Stunnel proxy: `http://<user>:<password>@<ip_address>:<port>`.
4. Save the manifest as `migration-controller.yaml`.
5. Apply the updated manifest:
   ```
   $ oc replace -f migration-controller.yaml -n openshift-migration
   ```

7.2.3. Migrating your applications from the command line

You can migrate your applications from the command line with the Migration Toolkit for Containers (MTC) API.

**Procedure**

1. Create a MigCluster CR manifest for the host cluster:
   ```
   $ cat << EOF | oc apply -f -
   ```
2. Create a **Secret** CR manifest for each remote cluster:

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

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

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

```yaml
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  name: <remote_cluster>  # Specify the Cluster CR of the remote cluster.  # Optional: For direct image migration, specify the exposed registry route.  # SSL verification is enabled if false. CA certificates are not required or checked if true.  # Specify the Secret CR of the remote cluster.  # Specify the URL of the remote cluster.
namespace: openshift-migration
spec:
  exposedRegistryPath: <exposed_registry_route>  # Specify the exposed registry route.  # SSL verification is enabled if false. CA certificates are not required or checked if true.  # Specify the Secret CR of the remote cluster.  # Specify the URL of the remote cluster.
insecure: false  # SSL verification is enabled if false. CA certificates are not required or checked if true.
isHostCluster: false
serviceAccountSecretRef:
  name: <remote_cluster_secret>  # Specify the Secret CR of the remote cluster.
  namespace: openshift-config
url: <remote_cluster_url>  # Specify the URL of the remote cluster.
EOF
```

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

```bash
$ oc describe cluster <cluster>
```
5. Create a **Secret** CR manifest for the replication repository:

```yaml
$ 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> 1
  aws-secret-access-key: <secret_key_base64> 2
EOF
```

1. Specify the key ID in base64 format.
2. 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:

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

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

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

```yaml
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigStorage
metadata:
  name: <migstorage>
  namespace: openshift-migration
spec:
  backupStorageConfig:
    awsBucketName: <bucket> 1
    credsSecretRef:
      name: <storage_secret> 2
      namespace: openshift-config
  backupStorageProvider: <storage_provider> 3
  volumeSnapshotConfig:
    credsSecretRef:
      name: <storage_secret> 4
      namespace: openshift-config
  volumeSnapshotProvider: <storage_provider> 5
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.
Specify the storage provider.

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.

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:
     destMigClusterRef:
       name: <host_cluster>
       namespace: openshift-migration
     indirectImageMigration: true
     indirectVolumeMigration: true
   migStorageRef:
     name: <migstorage>
     namespace: openshift-migration
   namespaces:
     - <application_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 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:
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=djongo
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: Advisory
  Last Transition Time: 2021-02-01T15:04:09Z
Message: Step: 19/47
Reason: InitialBackupCreated
Status: True
Type: Running
Category: Required
Last Transition Time: 2021-02-01T15:03:19Z
```
CHAPTER 7. ADVANCED MIGRATION OPTIONS

7.3. MIGRATION HOOKS

You can use migration hooks to run custom code at certain points during a migration.
7.3.1. About 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-rhel7:1.4`. 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.

7.3.2. 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.

7.3.2.1. 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
  - name: Print pod name
    debug:
      msg: "{{ pods.resources[0].metadata.name }}"
```

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
```

7.3.2.2. 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` plug-in.

Example environment variables

```yaml
- hosts: localhost
gather_facts: false
tasks:
  - set_fact:
    namespaces: "{{ (lookup('env', 'migration_namespaces')).split(',') }}"
  - debug:
    msg: "{{ item }}"
    with_items: "{{ namespaces }}"
```
7.4. 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:

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

2. Update the following parameters:

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.
7.5. EXCLUDING RESOURCES FROM A MIGRATION PLAN

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 1
     disable_pv_migration: true 2
     ... 
     excluded_resources: 3
     - imagetags
     - templateinstances
     - clusterserviceversions
     - packagemanifests
     - subscriptions
     - servicebrokers
     - servicebindings
     - servicelakes
     - serviceinstances
     - serviceplans
     - operatorgroups
     - events
   ```

   **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 `persistentvolumenonceclaims` 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:

   ```
   $ oc get deployment -n openshift-migration migration-controller -o yaml | grep EXCLUDED_RESOURCES -A1
   
   The output contains the excluded resources:
   
   **Example output**
   
   - name: EXCLUDED_RESOURCES
     value:
     
     imagetags,templateinstances,clusterserviceversions,package manifests,subscriptions,servicebrok ers,servicebindings,serviceclasses,serviceinstances,serviceplans,imagestreams,persistentvolum es,persistentvolumeclaims
   ```
CHAPTER 8. TROUBLESHOOTING

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

8.1. LOGS AND DEBUGGING TOOLS

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

8.1.1. 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. The Migrations page displays the migration types associated with the migration plan, for example, Stage, Migration, or Rollback.
3. Click the Type link to view the Migration details page.
4. Expand Migration resources to view the migration resources and their status.

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.

8.1.2. 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 to view the **Migrations** page.
   The **Migrations** page displays the migration types associated with the migration plan, for example, **Stage** or **Cutover** for warm migration.

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.

8.1.3. 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.

8.1.4. 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 installed.

Procedure

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

2. Run the `oc adm must-gather` command:
   - To gather data for the past hour:
     
     ```
     $ oc adm must-gather --image=registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.4
     ``

     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 gather data for the past 24 hours:
     
     ```
     $ oc adm must-gather --image= \
     registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8: \
     v1.4 -- /usr/bin/gather_metrics_dump
     ``

     This operation can take a long time. The data is saved as `/must-gather/metrics/prom_data.tar.gz`. You can view this file with the Prometheus console.

To view data with the Prometheus console

1. Create a local Prometheus instance:

   ```
   $ make prometheus-run
   ``

   The command outputs the Prometheus URL:

   **Output**

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

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

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

   ```
   $ make prometheus-cleanup
   ```

### 8.1.5. Using the Velero CLI to debug Backup and Restore CRs

You can debug the **Backup** and **Restore** custom resources (CRs) and partial migration failures with the Velero command line interface (CLI). The Velero CLI runs in the **velero** pod.

#### 8.1.5.1. Velero command syntax

Velero CLI commands use the following syntax:
8.1.5.2. Help command

The Velero `help` command lists all the Velero CLI commands:

```bash
$ oc exec $(oc get pods -n openshift-migration -o name | grep velero) -- ./velero --help
```

8.1.5.3. Describe command

The Velero `describe` command provides a summary of warnings and errors associated with a Velero resource:

```bash
$ oc exec $(oc get pods -n openshift-migration -o name | grep velero) -- ./velero describe <resource_id>
```

Example

```bash
$ oc exec $(oc get pods -n openshift-migration -o name | grep velero) -- ./velero backup describe 0e44ae00-5dc3-11eb-9ca8-df7e5254778b-2d8ql
```

8.1.5.4. Logs command

The Velero `logs` command provides the logs associated with a Velero resource:

```bash
velero <resource> logs <resource_id>
```

Example

```bash
$ oc exec $(oc get pods -n openshift-migration -o name | grep velero) -- ./velero restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

8.1.6. 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:
2. Check the status of the **Restore** CR by using the Velero **describe** command:

```
$ oc exec $(oc get pods -n openshift-migration -o name | grep velero) -n openshift-migration -
- ./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 exec $(oc get pods -n openshift-migration -o name | grep velero) -n openshift-migration -
- ./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
```
The `Restore` CR log error message, *the server could not find the requested resource*, indicates the cause of the partially failed migration.

### 8.1.7. 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:
  ```yaml
  labels:
    migrationcontroller: ebe13bee-c803-47d0-a9e9-83f380328b93
  ```
- **VolumeSnapshotLocation**
  The **VolumeSnapshotLocation** CR contains a `migrationcontroller` label to identify the MTC instance that created the CR:
  ```yaml
  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**

### Procedure

1. List the **MigMigration** CRs in the **openshift-migration** namespace:
   ```bash
   $ oc get migmigration -n openshift-migration
   ```

   **Example output**

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

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

```
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

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.105.179:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-44dd3bd5-c9f8-11e9-95ad-0205fe66cbb6
    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
    migmigration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    velero.io/storage-location: myrepo-vpqz9
  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: []
includedNamespaces: - sock-shop
includResources: - persistentvolumes
  - namesages
  - imagestreamtags
  - secrets
  - configmaps
  - pods
labelSelector:
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

apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quietse-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-06fa9fb68c88
  migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
  name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  namespace: openshift-migration
  resourceVersion: "82329"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources: 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

Additional resources for debugging tools

- MTC workflow
- MTC custom resources

8.2. COMMON ISSUES AND CONCERNS

This section describes common issues and concerns that can cause issues during migration.

8.2.1. 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
   ```

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.

**8.2.2. 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.

**8.2.2.1. CA certificate error displayed when accessing the MTC console for the first time**

If the MTC console displays a CA certificate error message the first time you try to access it, the likely cause is that a cluster uses self-signed CA certificates.

Navigate to the `oauth-authorization-server` URL in the error message and accept the certificate. To resolve this issue permanently, install the certificate authority so that it is trusted.

If the browser displays an Unauthorized message after you have accepted the CA certificate, navigate to the MTC console and then refresh the web page.

**8.2.2.2. OAuth timeout error in the MTC console**

If the MTC console displays a connection has timed out message after you have accepted a self-signed certificate, the cause is likely to be one of the following:
- Interrupted network access to the OAuth server
- Interrupted network access to the OpenShift Container Platform console
- Proxy configuration blocking access to the OAuth server. See MTC console inaccessible because of OAuth timeout error for details.

To determine the cause:

- Inspect the MTC console web page with a browser web inspector.
- Check the Migration UI pod log for errors.

8.2.2.3. 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:

```
$ 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.

8.2.2.4. 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:

- Error checking repository for stale locks
- Error getting backup storage location: backupstoragelocation.velero.io "my-bsl" not found

You can ignore these error messages. A missing BSL cannot cause a migration to fail.

8.2.2.5. Pod volume backup timeout error in the Velero pod log

If a migration fails because Restic times out, the Velero pod log displays the following error:

```bash
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
```

---

**CHAPTER 8. TROUBLESHOOTING**

---

73
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
   ```

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

5. Click **Save**.

### 8.2.2.6. 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 **MigMigration** CR displays the following error:

**MigMigration CR status**

```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. 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 troubleshoot the data verification error.

**Procedure**

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

```bash
$ 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:

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

The output identifies the **Restic** pod that logged the errors.

**PodVolumeRestore CR with Restic pod error**

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

4. View the **Restic** pod log to locate the errors:

```bash
$ oc logs -f <restic-nr2v5>
```

---

8.2.2.7. **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 **Restic** pod log displays the following error:

**Restic permission error**

```
```

---

CHAPTER 8. TROUBLESHOOTING

75
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:

   ```yaml
   spec:
   restic_supplemental_groups: <group_id>  
   ```

   1. Specify the supplemental group ID.

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

8.2.3. Known issues

This release has the following known issues:

- 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. (BZ#1748440)

- 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.

- If a migration fails, the migration plan does not retain custom PV settings for quiesced pods. You must manually roll back the migration, delete the migration plan, and create a new migration plan with your PV settings. (BZ#1784899)

- If a large migration fails because Restic times out, you can increase the restic_timeout parameter value (default: 1h) in the MigrationController custom resource (CR) manifest.

- If you select the data verification option for PVs that are migrated with the file system copy method, performance is significantly slower.

- If you are migrating data from NFS storage and root_squash is enabled, Restic maps to nfsnobody. The migration fails and a permission error is displayed in the Restic pod log. (BZ#1873641)

  You can resolve this issue by adding supplemental groups for Restic to the MigrationController CR manifest:

  ```yaml
  spec:  
  ```
8.3. ROLLING BACK A MIGRATION

You can roll back a migration by using the MTC web console or the CLI.

8.3.1. 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**

If you roll back a failed direct volume migration, the following resources are preserved in the namespaces specified in the migration plan to help you debug the failed migration:

- Config maps (source and target clusters)
- **Secret** CRs (source and target clusters)
- **Rsync** CRs (source cluster)
- **Service** CRs (target cluster)
- **Route** CRs (target cluster)

These resources must be deleted 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**.

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.

### 8.3.2. 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**

If you roll back a failed direct volume migration, the following resources are preserved in the namespaces specified in the **MigPlan** custom resource (CR) to help you debug the failed migration:

- Config maps (source and destination clusters)
- **Secret** CRs (source and destination clusters)
- **Rsync** CRs (source cluster)
- **Service** CRs (destination cluster)
- **Route** CRs (destination cluster)

These resources must be deleted 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:

```bash
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
name: <migmigration>
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.