



# OpenShift Container Platform 4.10

## MTC (Migration Toolkit for Containers)

OpenShift Container Platform 4 への移行



# OpenShift Container Platform 4.10 MTC (Migration Toolkit for Containers)

---

OpenShift Container Platform 4 への移行

## 法律上の通知

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

本書では、ステートフルなアプリケーションワークロードを OpenShift Container Platform 4 クラスタ間で移行する方法について説明します。

## 目次

<b>第1章 MTC (MIGRATION TOOLKIT FOR CONTAINERS)</b> .....	<b>4</b>
1.1. 用語	4
1.2. MTC ワークフロー	5
1.3. データのコピー方法	8
1.4. ボリュームの直接移行とイメージの直接移行	9
<b>第2章 MTC (MIGRATION TOOLKIT FOR CONTAINERS) リリースノート</b> .....	<b>11</b>
2.1. MTC (MIGRATION TOOLKIT FOR CONTAINERS 1.7) リリースノート	11
2.2. MTC(MIGRATION TOOLKIT FOR CONTAINERS 1.6) リリースノート	11
2.3. MIGRATION TOOLKIT FOR CONTAINERS 1.5 リリースノート	12
<b>第3章 MTC (MIGRATION TOOLKIT FOR CONTAINERS) のインストール</b> .....	<b>15</b>
3.1. 互換性のガイドライン	15
3.2. OPENSIFT CONTAINER PLATFORM 4.2 での従来の MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の 4.5 へのインストール	16
3.3. MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の OPENSIFT CONTAINER PLATFORM 4.10 へのインストール	17
3.4. プロキシ設定	18
3.5. レプリケーションリポジトリの設定	22
3.6. MTC のアンインストールおよびリソースの削除	29
<b>第4章 ネットワークの制限された環境での MIGRATION TOOLKIT FOR CONTAINERS OPERATOR のインストール</b> .....	<b>31</b>
4.1. 互換性のガイドライン	31
4.2. MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の OPENSIFT CONTAINER PLATFORM 4.10 へのインストール	32
4.3. OPENSIFT CONTAINER PLATFORM 4.2 での従来の MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の 4.5 へのインストール	33
4.4. プロキシ設定	35
4.5. レプリケーションリポジトリの設定	39
4.6. MTC のアンインストールおよびリソースの削除	40
<b>第5章 MIGRATION TOOLKIT FOR CONTAINERS のアップグレード</b> .....	<b>42</b>
5.1. OPENSIFT CONTAINER PLATFORM 4.10 での MTC(MIGRATION TOOLKIT FOR CONTAINERS) のアップグレード	42
5.2. OPENSIFT CONTAINER PLATFORM バージョン 4.2 の MTC (MIGRATION TOOLKIT FOR CONTAINERS) 4.5 へのアップグレード	42
5.3. MTC 1.3 から 1.7 へのアップグレード	44
<b>第6章 移行前のチェックリスト</b> .....	<b>45</b>
6.1. クラスターヘルスチェックリスト	45
6.2. ソースクラスターのチェックリスト	45
6.3. ターゲットクラスターのチェックリスト	46
<b>第7章 ネットワークの考慮事項</b> .....	<b>47</b>
7.1. DNS に関する考慮事項	47
7.2. ネットワークトラフィックリダイレクト戦略	48
<b>第8章 アプリケーションの移行</b> .....	<b>50</b>
8.1. 移行の前提条件	50
8.2. MTC の WEB コンソールを使用したアプリケーションの移行	51
<b>第9章 高度な移行オプション</b> .....	<b>59</b>
9.1. 用語	59

---

9.2. コマンドラインを使用したアプリケーションの移行	60
9.3. 移行フック	72
9.4. 移行計画のオプション	74
9.5. 移行コントローラーオプション	82
<b>第10章 トラブルシューティング</b> .....	<b>86</b>
10.1. MTC ワークフロー	86
10.2. MTC カスタムリソースマニフェスト	90
10.3. ログおよびデバッグツール	97
10.4. 一般的な問題および懸念事項	108
10.5. 移行のロールバック	113



# 第1章 MTC (MIGRATION TOOLKIT FOR CONTAINERS)

MTC (Migration Toolkit for Containers) を使用すると、OpenShift Container Platform 4 クラスター間でステートフルなアプリケーションのワークロードを namespace のレベルで移行できます。



## 注記

OpenShift Container Platform 3 から移行する場合は、[OpenShift Container Platform 3 から 4 への移行](#) および [OpenShift Container Platform 3 へのレガシー Migration Toolkit for Containers Operator のインストール](#) を参照してください。

状態の移行を使用して、同じクラスター内またはクラスター間でアプリケーションを移行できます。

MTC には、Web コンソールおよび API が同梱されており、Kubernetes カスタムリソースに基づいて移行を制御してアプリケーションのダウンタイムを最小限に抑えることができます。

MTC コンソールはデフォルトでターゲットクラスターにインストールされます。Migration Toolkit for Containers Operator を、コンソールを [リモートクラスター](#) にインストールするように設定できます。

以下のトピックの詳細は、[高度な移行オプション](#) を参照してください。

- 移行フックおよび MTC API で移行を自動化する。
- リソースを除外して大規模な移行をサポートし、ボリュームを直接移行する場合の自動 PV サイズ調整を有効化するように移行計画を設定する。

## 1.1. 用語

表1.1 MTC の用語

用語	定義
ソースクラスター	アプリケーションの移行元となるクラスター。
宛先クラスター <sup>[1]</sup>	アプリケーションが移行されるクラスター。
レプリケーションリポジトリ	ボリュームとイメージの直接的な移行時の Kubernetes オブジェクトに使用するオブジェクトストレージ、または間接的な移行時にイメージ、ボリューム、Kubernetes オブジェクトのコピーに使用するオブジェクトストレージ。  レプリケーションリポジトリはすべてのクラスターからアクセスする必要があります。
ホストクラスター	<b>migration-controller</b> Pod および Web コンソールが実行されているクラスター。ホストクラスターは通常宛先クラスターですが、これは必須ではありません。  ホストクラスターには、イメージの直接移行にレジストリルートを公開する必要はありません。



用語	定義
リモートクラスター	<p>通常、リモートクラスターはソースクラスターですが、これは必須ではありません。</p> <p>リモートクラスターには、<b>migration-controller</b> サービスアカウントトークンが含まれる <b>Secret</b> カスタムリソースが必要です。</p> <p>リモートクラスターには、直接のイメージ移行用にセキュアなレジストリルートを開示する必要があります。</p>
間接的な移行	<p>イメージ、ボリューム、および Kubernetes オブジェクトはソースクラスターからレプリケーションリポジトリにコピーされ、その後レプリケーションリポジトリから宛先クラスターにコピーされます。</p>
ボリュームの直接移行	<p>永続ボリュームはソースクラスターから宛先クラスターに直接コピーされます。</p>
イメージの直接移行	<p>イメージはソースクラスターから宛先クラスターに直接コピーされます。</p>
段階移行	<p>データはアプリケーションを停止せずに、宛先クラスターにコピーされます。</p> <p>段階移行を複数回実行すると、カットオーバー移行の時間が短縮されます。</p>
カットオーバー移行	<p>ソースクラスター上のアプリケーションが停止され、アプリケーションリソースが宛先クラスターに移行されます。</p>
状態の移行	<p>アプリケーションの状態は、特定の永続ボリューム要求を宛先クラスターにコピーして移行されます。</p>
ロールバック移行	<p>ロールバック移行は完了した移行をロールバックします。</p>

<sup>1</sup> MTC の Web コンソールの **ターゲットクラスター** を指します。

## 1.2. MTC ワークフロー

MTC (Migration Toolkit for Containers) の Web コンソールまたは Kubernetes API を使用して、Kubernetes リソース、永続ボリュームデータ、および内部コンテナイメージを OpenShift Container Platform 4.10 に移行できます。

MTC は以下のリソースを移行します。

- 移行計画に指定される namespace。
- namespace スコープのリソース: MTC が namespace を移行する場合、サービスや Pod などのその namespace に関連付けられるすべてのオブジェクトおよびリソースを移行します。さらに、namespace に存在するものの、クラスターレベルに存在しないリソースがクラスターレベルに存在するリソースに依存する場合、MTC は両方のリソースを移行します。  
たとえば、SCC (Security Context Constraints) はクラスターレベルに存在するリソースであり、サービスアカウント (SA) は namespace レベルに存在するリソースです。SA が MTC が移行する namespace に存在する場合、MTC は SA にリンクされている SCC を自動的に識別し、

それらの SCC も移行します。同様に、MTC は、namespace の永続ボリューム要求にリンクされている永続ボリュームを移行します。



注記

クラスタースコープのリソースは、リソースによっては手動で移行する必要があります。

- カスタムリソース (CR) およびカスタムリソース定義 (CRD): MTC は、namespace レベルで CR および CRD を自動的に移行します。

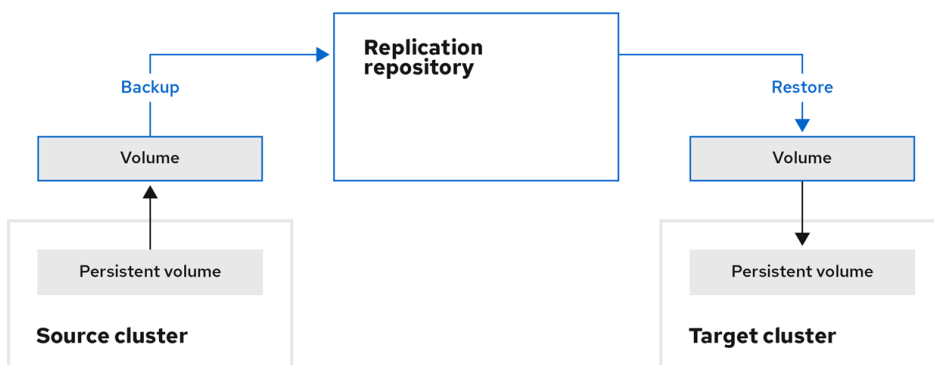
MTC Web コンソールを使用してアプリケーションを移行するには、以下の手順が必要です。

1. すべてのクラスターに MTC (Migration Toolkit for Containers Operator) をインストールします。  
インターネットアクセスが制限されているか、インターネットアクセスのない制限された環境で Migration Toolkit for Containers Operator をインストールできます。ソースおよびターゲットクラスターは、相互に対するネットワークアクセスおよびミラーレジストリーへのネットワークアクセスがなければなりません。
2. MTC がデータ移行に使用する中間オブジェクトストレージであるレプリケーションリポジトリを設定します。  
ソースおよびターゲットクラスターには、移行時にレプリケーションリポジトリへのネットワークアクセスがなければなりません。プロキシサーバーを使用している場合は、レプリケーションリポジトリとクラスター間のネットワークトラフィックを許可するように設定する必要があります。
3. ソースクラスターを MTC の Web コンソールに追加します。
4. レプリケーションリポジトリを MTC の Web コンソールに追加します。
5. 以下のデータ移行オプションのいずれかを使用して、移行計画を作成します。
  - **Copy:** MTC は、データをソースクラスターからレプリケーションリポジトリにコピーし、レプリケーションリポジトリからターゲットクラスターにコピーします。



注記

イメージの直接移行またはボリュームの直接移行を使用している場合、イメージまたはボリュームはソースクラスターからターゲットクラスターに直接コピーされます。

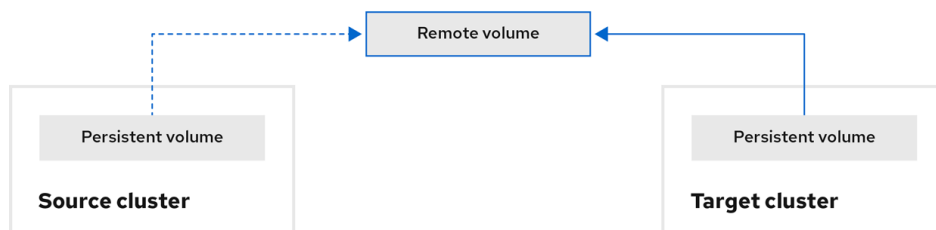


- **Move:** MTC は、ソースクラスターからリモートボリューム (例: NFS) をアンマウントし、リモートボリュームをポイントするターゲットクラスターで PV リソースを作成し、その後にリモートボリュームをターゲットクラスターにマウントします。ターゲットクラスターで実行されているアプリケーションは、ソースクラスターが使用していたものと同じリモートボリュームを使用します。リモートボリュームは、ソースクラスターおよびターゲットクラスターからアクセスできる必要があります。



### 注記

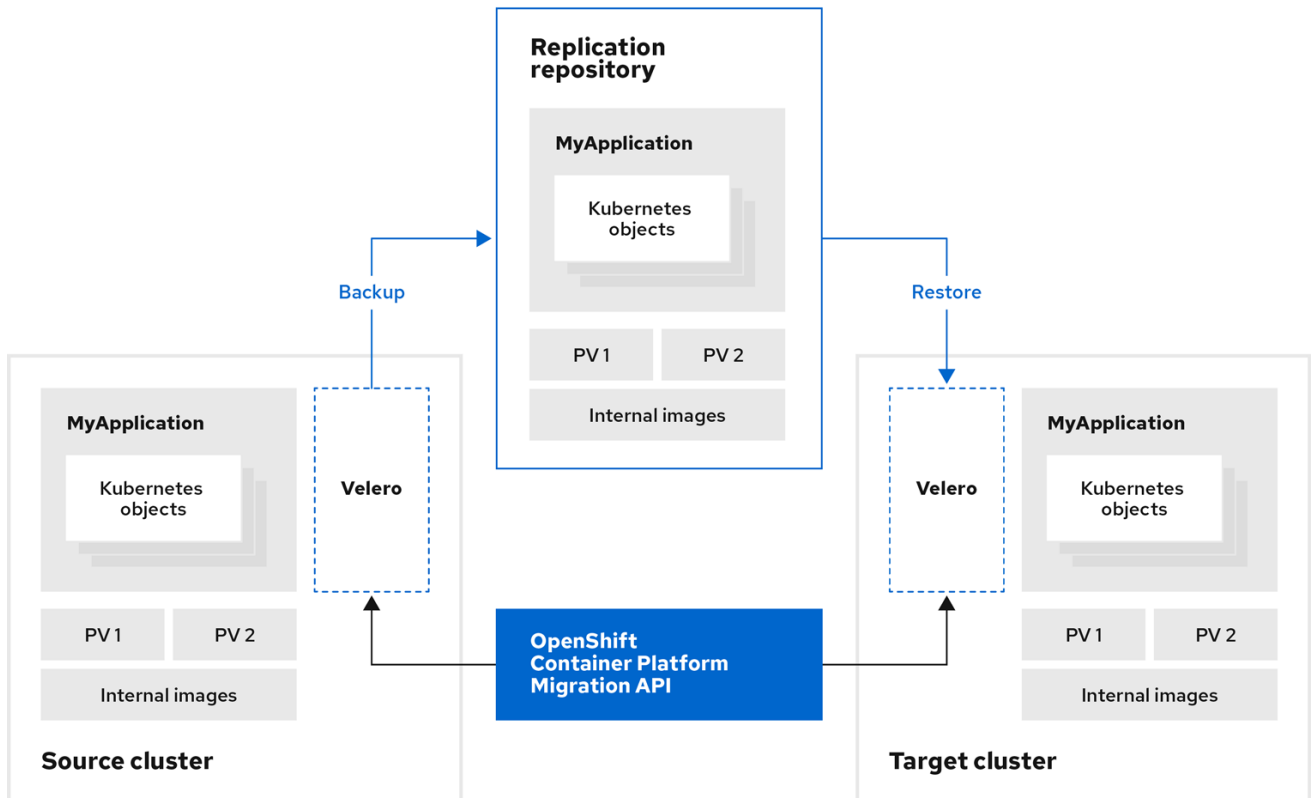
レプリケーションリポジトリはこの図には表示されていませんが、これは移行に必要です。



OpenShift\_45\_1019

6. 以下のオプションのいずれかを使用して、移行計画を実行します。

- **段階** 移行は、アプリケーションを停止せずにデータをターゲットクラスターにコピーします。  
段階移行は複数回実行して、移行前にほとんどのデータがターゲットにコピーされるようにします。1つ以上の段階移行を実行すると、カットオーバー移行の期間が短縮されます。
- **カットオーバー** は、ソースクラスターでアプリケーションを停止し、リソースをターゲットクラスターに移動します。  
オプション: `Halt transactions on the source cluster during migration` のチェックボックスのチェックを解除できます。



OpenShift\_45\_1019

### 1.3. データのコピー方法

MTC (Migration Toolkit for Containers) は、ソースクラスターからターゲットクラスターにデータを移行するために、ファイルシステムおよびスナップショットによるデータのコピー方法をサポートします。ご使用の環境に適した方法で、ストレージプロバイダーでサポートされる方法を選択できます。

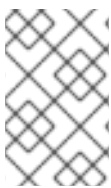
#### 1.3.1. ファイルシステムのコピー方法

MTC は、データファイルをソースクラスターからレプリケーションリポジトリにコピーし、そこからターゲットクラスターにコピーします。

ファイルシステムのコピー方法では、間接移行に Restic を使用し、直接ボリューム移行に Rsync を使用します。

表1.2 ファイルシステムのコピー方法の概要

利点	制限
<ul style="list-style-type: none"> <li>● クラスタで複数の異なるストレージクラスを使用することが可能</li> <li>● すべての S3 ストレージプロバイダーでサポートされている</li> <li>● チェックサムを使用したオプションのデータ検証</li> <li>● ボリュームの直接移行をサポートします。これにより、パフォーマンスを大幅に向上させることができます。</li> </ul>	<ul style="list-style-type: none"> <li>● スナップショットのコピー方法よりも遅い</li> <li>● オプションのデータ検証は、パフォーマンスを大幅に低下させます。</li> </ul>



### 注記

Restic および Rsync PV の移行では、サポートされている PV が **volumeMode=filesystem** のみであることを前提としています。ファイルシステムの移行に **volumeMode=Block** を使用することはサポートされていません。

## 1.3.2. スナップショットのコピー方法

MTC は、ソースクラスタのデータのスナップショットを、クラウドプロバイダーのレプリケーションリポジトリにコピーします。データはターゲットクラスタで復元されます。

スナップショットコピー方式は、Amazon Web Services、Google Cloud Provider、および Microsoft Azure で使用できます。

表1.3 スナップショットのコピー方法の概要

利点	制限
<ul style="list-style-type: none"> <li>● ファイルシステムのコピー方法よりも高速</li> </ul>	<ul style="list-style-type: none"> <li>● クラウドプロバイダーはスナップショットをサポートしている必要があります。</li> <li>● クラスタは同じクラウドプロバイダーになければなりません。</li> <li>● クラスタは、同じ場所またはリージョンにある必要があります。</li> <li>● クラスタには同じストレージクラスがなければなりません。</li> <li>● ストレージクラスにはスナップショットとの互換性がある必要があります。</li> <li>● ボリュームの直接移行はサポートしません。</li> </ul>

## 1.4. ボリュームの直接移行とイメージの直接移行

イメージの直接移行 (DIM) およびボリュームの直接移行 (DVM) を使用して、イメージおよびデータをソースクラスターからターゲットクラスターに直接移行できます。

異なるアベイラビリティゾーンにあるノードで DVM を実行する場合に、移行した Pod は永続ボリューム要求にアクセスできないために移行が失敗する可能性があります。

DIM および DVM では、ソースクラスターからレプリケーションリポジトリにファイルのバックアップを作成し、レプリケーションリポジトリからターゲットクラスターにファイルを復元する中間ステップが省略されるため、大きなパフォーマンス上の利点があります。データは [Rsync](#) で転送されます。

DIM および DVM には追加の前提条件があります。

## 第2章 MTC (MIGRATION TOOLKIT FOR CONTAINERS) リリースノート

MTC (Migration Toolkit for Containers) リリースノートでは、新機能および拡張機能、非推奨となった機能、および既知の問題について説明しています。

MTC を使用すると、namespace の粒度で OpenShift Container Platform クラスター間でアプリケーションワークロードを移行できます。

[OpenShift Container Platform 3 から 4.10](#) および OpenShift Container Platform 4 クラスター間で移行できます。

MTC には、Web コンソールおよび API が同梱されており、Kubernetes カスタムリソースに基づいて移行を制御してアプリケーションのダウンタイムを最小限に抑えることができます。

MTC のサポートポリシーの詳細は、[Red Hat OpenShift Container Platform ライフサイクルポリシー](#)の一部である [OpenShift Application and Cluster Migration Solutions](#) を参照してください。

### 2.1. MTC (MIGRATION TOOLKIT FOR CONTAINERS 1.7) リリースノート

#### 2.1.1. 新機能および機能拡張

本リリースには、以下の新機能および機能拡張が含まれています。

- MTC (Migration Toolkit for Containers) Operator が OpenShift API for Data Protection (OADP) Operator に依存するようになりました。MTC Operator をインストールすると、Operator Lifecycle Manager (OLM) は同じ namespace に OADP Operator を自動的にインストールします。
- **crane tunnel-api** コマンドを使用して 2 つのクラスター間にネットワークトンネルを確立することにより、ファイアウォールの背後にあるソースクラスターからクラウドベースの宛先クラスターに移行できます。
- MTC の Web コンソールでのストレージクラスの変換: 永続ボリューム (PV) のストレージクラスを、同じクラスター内で移行して変換できます。

#### 2.1.2. 既知の問題

本リリースには、以下の既知の問題があります。

- AWS gp2 PVC に利用可能な領域がない場合に、**MigPlan** カスタムリソースには警告は表示されません。(BZ#1963927)
- 宛先ストレージが AWS Elastic File System (EFS) によって動的にプロビジョニングされる PV の場合に、直接および間接データ転送は機能しません。これは、AWS EFS Container Storage Interface (CSI) ドライバーの制限が原因です。(BZ#2085097)
- IBM Cloud のブロックストレージは、同じアベイラビリティゾーンに所属している必要があります。[仮想プライベートクラウドのブロックストレージに関する IBM FAQ](#) を参照してください。

### 2.2. MTC (MIGRATION TOOLKIT FOR CONTAINERS 1.6) リリースノート

### 2.2.1. 新機能および機能拡張

本リリースには、以下の新機能および機能拡張が含まれています。

- 状態の移行: 特定の永続ボリューム要求 (PVC) を選択して、反復可能な状態のみの移行を実行できます。
- "New operator version available" の通知: MTC の Web コンソールのクラスターページには、新しい Migration Toolkit for Containers Operator が利用可能になると通知が表示されます。

### 2.2.2. 非推奨の機能

以下の機能が非推奨になりました。

- MTC バージョン 1.4 はサポートされなくなりました。

### 2.2.3. 既知の問題

本リリースには、以下の既知の問題があります。

- OpenShift Container Platform 3.10 では、**MigrationController** Pod の再起動に過剰に時間を要します。Bugzilla レポートには回避策が含まれています。(BZ#1986796)
- **段階** Pod は、IBM Cloud の従来の OpenShift Container Platform ソースクラスターからボリュームの直接移行時に失敗します。IBM ブロックストレージプラグインでは、同じノードの複数の Pod に、同じボリュームをマウントすることはできません。その結果、PVC は Rsync Pod およびアプリケーション Pod を同時にマウントすることはできません。この問題を解決するには、移行前にアプリケーション Pod を停止します。(BZ#1887526)
- AWS gp2 PVC に利用可能な領域がない場合に、**MigPlan** カスタムリソースには警告は表示されません。(BZ#1963927)
- IBM Cloud のブロックストレージは、同じアベイラビリティゾーンに所属している必要があります。仮想プライベートクラウドのブロックストレージに関する IBM FAQ を参照してください。

## 2.3. MIGRATION TOOLKIT FOR CONTAINERS 1.5 リリースノート

### 2.3.1. 新機能および機能拡張

本リリースには、以下の新機能および機能拡張が含まれています。

- Web コンソールの **Migration 詳細** ページの **Migration** リソースツリーが、追加のリソース、Kubernetes イベント、および移行の監視およびデバッグ用のライブステータス情報と共に強化されました。
- Web コンソールは、数百もの移行プランをサポートできます。
- ソース名前空間は、移行計画で別のターゲット名前空間にマッピングすることができます。以前のバージョンでは、ソース名前空間は同じ名前を持つターゲット名前空間にマップされていました。
- ステータス情報を含むフックフェーズは、移行時に Web コンソールに表示されます。
- Rsync 再試行試行の数は、ボリュームの直接移行時に Web コンソールに表示されます。



- 永続ボリューム (PV) のサイズ変更は、ターゲットクラスターがディスク領域不足しないように、ボリュームの直接移行用に有効にできます。
- PV のサイズ変更をトリガーするしきい値は設定可能です。以前のバージョンでは、ディスクの使用状況が 97% を超えると PV のサイズ変更が発生していました。
- Velero がバージョン 1.6 にアップグレードされ、多くの修正および機能強化が数多く追加されました。
- キャッシュされた Kubernetes クライアントを有効にして、パフォーマンスを向上させることができます。

### 2.3.2. 非推奨の機能

以下の機能が非推奨になりました。

- MTC バージョン 1.2 および 1.3 はサポートされなくなりました。
- **oc convert** コマンドが非推奨になるため、非推奨の API の更手順は、ドキュメントのトラブルシューティングセクションから削除されました。

### 2.3.3. 既知の問題

本リリースには、以下の既知の問題があります。

- 400 を超える移行プランを作成する場合、Microsoft Azure ストレージは利用できません。**MigStorage** カスタムリソースは、以下のメッセージを表示します。**The request is being throttled as the limit has been reached for operation type** (このリクエストは、操作タイプの制限に達したため、スロットルされています)。(BZ#1977226)
- 移行に失敗すると、移行計画は休止状態の Pod のカスタム永続ボリューム (PV) 設定を保持しません。移行を手動でロールバックし、移行計画を削除し、PV 設定で新たな移行計画を作成する必要があります。(BZ#1784899)
- PV のサイズ変更は、**pv\_resizing\_threshold** が 42% 以上でない限り、AWS gp2 ストレージでは期待どおりに動作しません。(BZ#1973148)
- PV のサイズ変更は、以下のシナリオでは OpenShift Container Platform 3.7 および 3.9 ソースクラスターでは機能しません。
  - アプリケーションは、MTC のインストール後にインストールされている。
  - アプリケーション Pod は MTC のインストール後に別のノードで再スケジュールされました。

OpenShift Container Platform 3.7 および 3.9 では、Velero が **Restic** Pod に PV を自動的にマウントできるようにする Mount Propagation 機能をサポートしません。**MigAnalytic** カスタムリソース (CR) は、**Restic** Pod から PV データを収集できず、リソースを **0** として報告します。**MigPlan** CR は以下のようなステータスを表示します。

#### 出力例

```
status:
  conditions:
  - category: Warn
    lastTransitionTime: 2021-07-15T04:11:44Z
    message: Failed gathering extended PV usage information for PVs [nginx-logs nginx-
```

```
html], please see MigAnalytic openshift-migration/ocp-24706-basicvolmig-migplan-1626319591-szwd6 for details
reason: FailedRunningDf
status: "True"
type: ExtendedPVAnalysisFailed
```

PVのサイズ変更を有効にするには、ソースクラスターで Restic daemonset を手動で再起動するか、アプリケーションと同じノードで **Restic** Pod を再起動します。Restic を再起動しない場合、PVのサイズ変更なしにボリュームの直接移行を実行できます。  
([BZ#1982729](#))

### 2.3.4. 技術上の変更点

本リリースには、以下の技術上の変更点があります。

- OpenShift Container Platform バージョン 3.7 から 4.5 では、従来の Migration Toolkit for Containers Operator バージョン 1.5.1 は手動でインストールします。
- OpenShift Container Platform バージョン 4.6 以降では、Migration Toolkit for Containers Operator バージョン 1.5.1 は、Operator Lifecycle Manager を使用してインストールします。

## 第3章 MTC (MIGRATION TOOLKIT FOR CONTAINERS) のインストール

MTC (Migration Toolkit for Containers) を OpenShift Container Platform 4 にインストールできます。



### 注記

OpenShift Container Platform 3 に MTC をインストールするには、[OpenShift Container Platform 3 へのコンテナ Operator 用のレガシー Migration Toolkit のインストール](#) を参照してください。

デフォルトで、MTC Web コンソールおよび **Migration Controller** Pod はターゲットクラスターで実行されます。**Migration Controller** カスタムリソースマニフェストを、MTC の Web コンソールおよび **Migration Controller** Pod を [リモートクラスター](#) で実行するように設定できます。

MTC をインストールした後、オブジェクトストレージをレプリケーションリポジトリとして使用するよう設定する必要があります。

MTC をアンインストールするには、[MTC のアンインストールとリソースの削除](#) を参照してください。

### 3.1. 互換性のガイドライン

OpenShift Container Platform バージョンと互換性がある MTC (Migration Toolkit for Containers) Operator をインストールする必要があります。

#### 定義

##### レガシープラットフォーム

OpenShift Container Platform 4.5 以前。

##### 最新プラットフォーム

OpenShift Container Platform 4.6 以降。

##### レガシー operator

レガシープラットフォーム用に設計された MTC Operator。

##### 最新 operator

最新のプラットフォーム用に設計された MTC Operator。

##### コントロールクラスター

MTC コントローラーと GUI を実行するクラスター。


##### リモートクラスター

Velero を実行する移行のソースクラスターまたは宛先クラスター。コントロールクラスターは、Velero API を介してリモートクラスターと通信し、移行を促進します。

表3.1 MTC 互換性: レガシープラットフォームからの移行

OpenShift Container Platform 4.5 以前

OpenShift Container Platform 4.6 以降

	OpenShift Container Platform 4.5 以前	OpenShift Container Platform 4.6 以降
安定した MTC バージョン	<p>MTC 1.7.z</p> <p>レガシー 1.7 operator: <b>operator.yml</b> ファイルを使用して手動でインストール。</p>  <p><b>重要</b></p> <p>このクラスターをコントロールクラスターにすることはできません。</p>	<p>MTC 1.7.z</p> <p>OLM を使用してインストール、リリースチャンネル <b>release-v1.7</b></p>

### 注記

ネットワークの制限により、最新のクラスターが移行に関与する他のクラスターに接続できないというエッジケースが存在します。たとえば、オンプレミスの OpenShift Container Platform 3.11 クラスターからクラウド内の最新の OpenShift Container Platform クラスターに移行する場合、最新のクラスターは OpenShift Container Platform 3.11 クラスターに接続できません。

MTC 1.7 では、ネットワーク制限のためにいずれかのリモートクラスターがコントロールクラスターと通信できない場合は、**crane tunnel-api** コマンドを使用します。

安定した MTC リリースでは、常に最新のクラスターを制御クラスターとして指定する必要がありますが、この特定のケースでは、レガシークラスターを制御クラスターとして指定し、ワークロードをリモートクラスターにプッシュすることができます。

## 3.2. OPENSIFT CONTAINER PLATFORM 4.2 での従来の MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の 4.5 へのインストール

レガシー Migration Toolkit for Containers Operator を手動で OpenShift Container Platform バージョン 4.2 から 4.5 にインストールできます。

### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインする必要があります。
- **registry.redhat.io** にアクセスできる必要があります。
- **podman** がインストールされている必要があります。

### 手順

1. Red Hat カスタマーポータル認証情報を使用して **registry.redhat.io** にログインします。

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

2. 次のコマンドを実行して、**operator.yml** ファイルをダウンロードします。

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

3. 次のコマンドを実行して、**controller.yml** ファイルをダウンロードします。

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

4. ソースクラスターにログインします。
5. クラスターが **registry.redhat.io** で認証できることを確認します。

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

6. Migration Toolkit for Containers Operator オブジェクトを作成します。

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

### 出力例

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

- 1** **Error from server (AlreadyExists)** メッセージは無視できます。これらは、以降のリソースで提供される OpenShift Container Platform 4 以前のバージョン用にリソースを作成する Migration Toolkit for Containers Operator が原因です。

7. **MigrationController** オブジェクトを作成します。

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

8. MTC Pod が実行されていることを確認します。

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

## 3.3. MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の OPENSIFT CONTAINER PLATFORM 4.10 へのインストール

Operator Lifecycle Manager を使用して OpenShift Container Platform 4.10 に Migration Toolkit for Containers Operator をインストールします。

## 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。

## 手順

1. OpenShift Container Platform Web コンソールで、**Operators** → **OperatorHub** をクリックします。
2. **Filter by keyword** フィールドを使用して、**Migration Toolkit for Containers Operator** を見つけます。
3. **Migration Toolkit for Containers Operator** を選択し、**Install** をクリックします。
4. **Install** をクリックします。  
**Installed Operators** ページで、**Migration Toolkit for Containers Operator** は、**Succeeded** のステータスで **openshift-migration** プロジェクトに表示されます。
5. **Migration Toolkit for Containers Operator** をクリックします。
6. **Provided APIs** の下で **Migration Controller** タイルを見つけ、**Create Instance** をクリックします。
7. **Create** をクリックします。
8. **Workloads** → **Pods** をクリックし、MTC Pod が実行されていることを確認します。

## 3.4. プロキシ設定

OpenShift Container Platform 4.1 以前のバージョンでは、これらのバージョンはクラスター全体の **proxy** オブジェクトをサポートしないため、Migration Toolkit for Containers Operator のインストール後に、**MigrationController** カスタムリソース (CR) マニフェストでプロキシを設定する必要があります。

OpenShift Container Platform 4.2 から 4.10 の場合、MTC (Migration Toolkit for Containers) はクラスター全体のプロキシ設定を継承します。クラスター全体のプロキシ設定を上書きする場合は、プロキシパラメーターを変更できます。

### 3.4.1. ボリュームの直接移行

MTC 1.4.2 で、ボリュームの直接移行 (DVM) が導入されました。DVM は1つのプロキシのみをサポートします。ターゲットクラスターもプロキシの背後にある場合、ソースクラスターはターゲットクラスターのルートにアクセスできません。

プロキシの背後にあるソースクラスターから DVM を実行する場合には、トランスポート層で機能する TCP プロキシを設定して、SSL 接続を独自の SSL 証明書で復号化および再暗号化せずに透過的に転送する必要があります。Stunnel プロキシは、このようなプロキシの例です。

#### 3.4.1.1. DVM の TCP プロキシ設定

TCP プロキシ経由でソースとターゲットクラスターの間直接接続を設定し、プロキシを使用できるように **MigrationController** CR の **stunnel\_tcp\_proxy** 変数を設定できます。

apiVersion: migration.openshift.io/v1alpha1

```

kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  [...]
  stunnel_tcp_proxy: http://username:password@ip:port

```

ボリュームの直接移行 (DVM) は、プロキシの Basic 認証のみをサポートします。さらに、DVM は、TCP 接続を透過的にトンネルできるプロキシの背後でのみ機能します。中間者モードの HTTP/HTTPS プロキシは機能しません。既存のクラスター全体にわたるプロキシはこの動作をサポートしない可能性があります。その結果、DVM のプロキシ設定は、MTC の通常のプロキシ設定とは異なる状態に保たれます。

### 3.4.1.2. HTTP/HTTPS プロキシの代わりに TCP プロキシを使用する理由

DVM を有効にするには、OpenShift ルートを介してソースおよびターゲットクラスター間で Rsync を実行します。トラフィックは、TCP プロキシである Stunnel を使用して暗号化されます。ソースクラスターで実行している Stunnel は、ターゲット Stunnel との TLS 接続を開始し、暗号化されたチャンネルでデータを転送します。

OpenShift のクラスター全体の HTTP/HTTPS プロキシは通常、外部サーバーで独自の TLS セッションをネゴシエートする中間者モードで設定されます。ただし、これは Stunnel では機能しません。Stunnel では、プロキシによって TLS セッションが変更されないようにする必要があります。基本的には、プロキシを透過的なトンネルにし、単純に TCP 接続をそのまま転送する必要があります。したがって、TCP プロキシを使用する必要があります。

### 3.4.1.3. 既知の問題

#### 移行が **Upgrade request required** エラーで失敗する

移行コントローラーは SPDY プロトコルを使用してリモート Pod 内でコマンドを実行します。リモートクラスターがプロキシまたは、SPDY プロトコルをサポートしないファイアウォールの背後にある場合には、移行コントローラーはリモートコマンドの実行に失敗します。移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: SPDY プロトコルをサポートするプロキシを使用します。

SPDY プロトコルのサポートに加えて、このプロキシまたはファイアウォールでは、**Upgrade** HTTP ヘッダーを API サーバーに渡す必要もあります。クライアントはこのヘッダーを使用して API サーバーと Websocket 接続を開きます。**Upgrade** ヘッダーがプロキシまたはファイアウォールでブロックされると、移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: プロキシで **Upgrade** ヘッダーが転送されるようにしてください。

## 3.4.2. 移行用のネットワークポリシーのチューニング

OpenShift は、クラスターで使用されるネットワークプラグインに基づいて **NetworkPolicy** または **EgressFirewalls** を使用した Pod との間のトラフィックの制限をサポートします。移行に関連するソース namespace のいずれかがこのようなメカニズムを使用して Pod へのネットワークトラフィックを制限する場合には、この制限により移行時に Rsync Pod へのトラフィックが誤って停止される可能性があります。

ソースおよびターゲットクラスターの両方で実行される Rsync Pod は OpenShift Route 経由で相互に接続する必要があります。既存の **NetworkPolicy** または **EgressNetworkPolicy** オブジェクトは、これらのトラフィックの制限が課されないように Rsync Pod を自動的に取り除くように設定できます。

### 3.4.2.1. NetworkPolicy の設定

#### 3.4.2.1.1. Rsync Pod からの egress トラフィック

Rsync Pod の一意のラベルを使用し、同期元または同期先 namespace の **NetworkPolicy** 設定がこのタイプのトラフィックをブロックする場合に egress トラフィックがそれらを通過することを許可できます。以下のポリシーは、namespace の Rsync Pod からの全 egress トラフィックを許可します。

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

#### 3.4.2.1.2. Rsync Pod への ingress トラフィック

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

### 3.4.2.2. EgressNetworkPolicy 設定

**EgressNetworkPolicy** オブジェクトまたは **Egress ファイアウォール** は、egress トラフィックをクラスターからブロックするために設計された OpenShift コンストラクトです。

**NetworkPolicy** オブジェクトとは異なり、Egress ファイアウォールは namespace のすべての Pod に適用されるためにプロジェクトレベルで機能します。そのため、Rsync Pod の一意のラベルを使用すると、この制限から除外するのは Rsync Pod だけではありません。ただし、ソースおよびターゲットクラスターの CIDR 範囲をポリシーの **Allow** ルールに追加して、2つのクラスター間で直接接続を設定できます。

Egress ファイアウォールが存在するクラスターに基づいて、他のクラスターの CIDR 範囲を追加して、2つの間の egress トラフィックを許可できます。

```
apiVersion: network.openshift.io/v1
kind: EgressNetworkPolicy
```



```

metadata:
  name: test-egress-policy
  namespace: <namespace>
spec:
  egress:
  - to:
    cidrSelector: <cidr_of_source_or_target_cluster>
    type: Deny

```

### 3.4.2.3. データ転送用の代替エンドポイントの選択

デフォルトでは、DVM は OpenShift Container Platform ルートをエンドポイントとして使用して、PV データを宛先クラスターに転送します。クラスターポロジで許可されている場合は、サポートされている別の種類のエンドポイントを選択できます。

クラスターごとに、**MigrationController** CR で適切な 宛先 クラスターに **rsync\_endpoint\_type** 変数を設定することで、エンドポイントを設定できます。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  [...]
  rsync_endpoint_type: [NodePort|ClusterIP|Route]

```

### 3.4.2.4. RsyncPod の補足グループの設定

PVC が共有ストレージを使用する場合、Pod がアクセスを許可するように RsyncPod 定義に補足グループを追加して、そのストレージへのアクセスを設定できます。

表3.2 RsyncPod の補足グループ

変数	タイプ	デフォルト	説明
<b>src_supplemental_groups</b>	string	設定されていません	ソース Rsync Pod の補足グループのコンマ区切りリスト
<b>target_supplemental_groups</b>	string	設定されていません	ターゲット Rsync Pod の補足グループのコンマ区切りリスト

#### 使用例

**MigrationController** CR を更新して、これらの補足グループの値を設定できます。

```

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

```

### 3.4.3. プロキシの設定

#### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。

#### 手順

1. **MigrationController** CR マニフェストを取得します。

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

2. プロキシパラメーターを更新します。

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

- ❶ ボリュームの直接移行のための Stunnel プロキシ URL。
- ❷ プロキシを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。

サブドメインのみと一致するように、ドメインの前に `.` を付けます。たとえば、`.y.com` は `x.y.com` に一致しますが、`y.com` には一致しません。\* を使用し、すべての宛先のプロキシをバイパスします。インストール設定で `networking.machineNetwork[].cidr` フィールドで定義されるネットワークに含まれていないワーカーをスケールアップする場合、それらをこの一覧に追加し、接続の問題を防ぐ必要があります。

`httpProxy` または `httpsProxy` フィールドのいずれも設定されていない場合、このフィールドは無視されます。

3. マニフェストを **migration-controller.yaml** として保存します。
4. 更新したマニフェストを適用します。

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

詳細は、[Configuring the cluster-wide proxy](#) を参照してください。

## 3.5. レプリケーションリポジトリの設定

オブジェクトストレージをレプリケーションリポジトリとして使用するよう設定する必要があります。MTC (Migration Toolkit for Containers) は、データをソースクラスターからレプリケーションリポジトリにコピーしてから、レプリケーションリポジトリからターゲットクラスターにコピーします。

MTC は、ソースクラスターからターゲットクラスターにデータを移行するために、[ファイルシステムおよびスナップショットによるデータのコピー方法](#) をサポートします。ご使用の環境に適しており、ストレージプロバイダーでサポートされる方法を選択します。

MTC は、以下のストレージプロバイダーをサポートしています。

- [Multicloud Object Gateway](#)
- [Amazon Web Services S3](#)
- [Google Cloud Platform](#)
- [Microsoft Azure Blob](#)
- 汎用 S3 オブジェクトストレージ (例: Minio または Ceph S3)

### 3.5.1. 前提条件

- すべてのクラスターには、レプリケーションリポジトリへの中断されないネットワークアクセスが必要です。
- 内部でホストされるレプリケーションリポジトリでプロキシサーバーを使用する場合は、プロキシがレプリケーションリポジトリへのアクセスを許可することを確認する必要があります。

### 3.5.2. Multicloud Object Gateway の認証情報の取得

MCG を Migration Toolkit for Containers (MTC) のレプリケーションリポジトリとして設定するには、Multicloud Object Gateway (MCG) 認証情報と S3 エンドポイントを取得する必要があります。OpenShift API for Data Protection (OADP) の **Secret** カスタムリソース (CR) を作成するには、Multicloud Object Gateway (MCG) 認証情報を取得する必要があります。

MCG は、OpenShift Data Foundation のコンポーネントです。

#### 前提条件

- 適切な [OpenShift Data Foundation deployment guide](#) を使用して、OpenShift Data Foundation をデプロイする必要があります。

#### 手順

1. **NooBaa** カスタムリソースで **describe** コマンドを実行して、S3 エンドポイントである **AWS\_ACCESS\_KEY\_ID** および **AWS\_SECRET\_ACCESS\_KEY** を取得します。これらの認証情報を使用して、MCG をレプリケーションリポジトリとして追加します。

### 3.5.3. Amazon Web Services の設定

Amazon Web Services (AWS) S3 オブジェクトストレージを、Migration Toolkit for Containers (MTC) のレプリケーションリポジトリとして設定します。

#### 前提条件

- **AWS CLI** がインストールされていること。

- AWS S3 ストレージバケットは、ソースクラスターおよびターゲットクラスターからアクセスできる必要があります。
- スナップショットのコピー方法を使用する場合は、以下の条件を満たす必要があります。
  - EC2 Elastic Block Storage (EBS) にアクセスできる必要があります。
  - ソースおよびターゲットクラスターが同じリージョンにある必要があります。
  - ソースおよびターゲットクラスターには、同じストレージクラスがある必要があります。
  - ストレージクラスはスナップショットと互換性がある必要があります。

## 手順

1. **BUCKET** 変数を設定します。

```
$ BUCKET=<your_bucket>
```

2. **REGION** 変数を設定します。

```
$ REGION=<your_region>
```

3. AWS S3 バケットを作成します。

```
$ aws s3api create-bucket \
  --bucket $BUCKET \
  --region $REGION \
  --create-bucket-configuration LocationConstraint=$REGION ❶
```

- ❶ **us-east-1** は **LocationConstraint** をサポートしていません。お住まいの地域が **us-east-1** の場合は、**--create-bucket-configuration LocationConstraint=\$REGION** を省略してください。

4. IAM ユーザーを作成します。

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

- ❶ Velero を使用して複数の S3 バケットを持つ複数のクラスターをバックアップする場合は、クラスターごとに一意のユーザー名を作成します。

5. **velero-policy.json** ファイルを作成します。

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

```

        "ec2:CreateVolume",
        "ec2:CreateSnapshot",
        "ec2>DeleteSnapshot"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3>DeleteObject",
      "s3:PutObject",
      "s3:AbortMultipartUpload",
      "s3:ListMultipartUploadParts"
    ],
    "Resource": [
      "arn:aws:s3:::${BUCKET}/*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:ListBucket",
      "s3:GetBucketLocation",
      "s3:ListBucketMultipartUploads"
    ],
    "Resource": [
      "arn:aws:s3:::${BUCKET}"
    ]
  }
]
}
EOF

```

6. ポリシーを添付して、**velero** ユーザーに必要な最小限の権限を付与します。

```

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

```

7. **velero** ユーザーのアクセスキーを作成します。

```

$ aws iam create-access-key --user-name velero

```

### 出力例

```

{
  "AccessKey": {
    "UserName": "velero",
    "Status": "Active",
    "CreateDate": "2017-07-31T22:24:41.576Z",
    "SecretAccessKey": <AWS_SECRET_ACCESS_KEY>,

```

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

**AWS\_SECRET\_ACCESS\_KEY** と **AWS\_ACCESS\_KEY\_ID** を記録します。認証情報を使用して、AWS をレプリケーションリポジトリとして追加します。

### 3.5.4. Google Cloud Provider の設定

Google Cloud Provider (GCP) ストレージバケットを MTC (Migration Toolkit for Containers) のレプリケーションリポジトリとして設定します。

#### 前提条件

- **gcloud** および **gsutil** CLI ツールがインストールされている必要があります。詳細は、[Google Cloud のドキュメント](#) をご覧ください。
- GCP ストレージバケットは、ソースクラスターおよびターゲットクラスターからアクセスできる必要があります。
- スナップショットのコピー方法を使用する場合は、以下の条件を満たす必要があります。
  - ソースおよびターゲットクラスターが同じリージョンにある必要があります。
  - ソースおよびターゲットクラスターには、同じストレージクラスがある必要があります。
  - ストレージクラスはスナップショットと互換性がある必要があります。

#### 手順

1. GCP にログインします。

```
$ gcloud auth login
```

2. **BUCKET** 変数を設定します。

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

- 1 バケット名を指定します。

3. ストレージバケットを作成します。

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

4. **PROJECT\_ID** 変数をアクティブなプロジェクトに設定します。

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

5. サービスアカウントを作成します。

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

6. サービスアカウントを一覧表示します。

```
$ gcloud iam service-accounts list
```

7. **email** の値と一致するように **SERVICE\_ACCOUNT\_EMAIL** 変数を設定します。

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

8. ポリシーを添付して、**velero** ユーザーに必要最小限の権限を付与します。

```
$ 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
  storage.objects.create
  storage.objects.delete
  storage.objects.get
  storage.objects.list
  iam.serviceAccounts.signBlob
)
```

9. **velero.server** カスタムロールを作成します。

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

10. IAM ポリシーバインディングをプロジェクトに追加します。

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

11. IAM サービスアカウントを更新します。

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

12. IAM サービスアカウントのキーを現在のディレクトリーにある **credentials-velero** ファイルに保存します。

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

**credentials-velero** ファイルを使用して、GCP をレプリケーションリポジトリとして追加します。

### 3.5.5. Microsoft Azure の設定

Microsoft Azure Blob ストレージコンテナを MTC (Migration Toolkit for Containers) のレプリケーションリポジトリとして設定します。

#### 前提条件

- [Azure CLI](#) がインストールされていること。
- Azure Blob ストレージコンテナがソースクラスターおよびターゲットクラスターからアクセスできること。
- スナップショットのコピー方法を使用する場合は、以下の条件を満たす必要があります。
  - ソースおよびターゲットクラスターが同じリージョンにある必要があります。
  - ソースおよびターゲットクラスターには、同じストレージクラスがある必要があります。
  - ストレージクラスはスナップショットと互換性がある必要があります。

#### 手順

1. Azure にログインします。

```
$ az login
```

2. **AZURE\_RESOURCE\_GROUP** 変数を設定します。

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

3. Azure リソースグループを作成します。

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

- 1** 場所を指定します。

4. **AZURE\_STORAGE\_ACCOUNT\_ID** 変数を設定します。

```
$ AZURE_STORAGE_ACCOUNT_ID="velero$(uuidgen | cut -d '-' -f5 | tr '[A-Z]' '[a-z]')"
```

5. Azure ストレージアカウントを作成します。

```
$ 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
```

6. **BLOB\_CONTAINER** 変数を設定します。



```
$ BLOB_CONTAINER=velero
```

- Azure Blob ストレージコンテナを作成します。

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

- velero** のサービスプリンシパルおよび認証情報を作成します。

```
$ 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`
```

- サービスプリンシパルの認証情報を **credentials-velero** ファイルに保存します。

```
$ 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
```

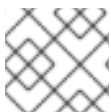
**credentials-velero** ファイルを使用して、Azure をレプリケーションリポジトリとして追加します。

### 3.5.6. 関連情報

- [MTC ワークフロー](#)
- [データのコピー方法](#)
- [MTC の Web コンソールへのレプリケーションリポジトリの追加](#)

## 3.6. MTC のアンインストールおよびリソースの削除

MTC (Migration Toolkit for Containers) をアンインストールし、そのリソースを削除してクラスターをクリーンアップできます。



### 注記

**velero** CRD を削除すると、Velero がクラスターから削除されます。

### 前提条件

- cluster-admin** 権限を持つユーザーとしてログインしている。

## 手順

1. すべてのクラスターで **MigrationController** カスタムリソース (CR) を削除します。

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

2. Operator Lifecycle Manager を使用して OpenShift Container Platform 4 の Migration Toolkit for Containers Operator をアンインストールします。
3. 以下のコマンドを実行して、すべてのクラスターでクラスタースコープのリソースを削除します。

- **migration** カスタムリソース定義 (CRDs):

```
$ oc delete $(oc get crds -o name | grep 'migration.openshift.io')
```

- **Velero** CRD:

```
$ oc delete $(oc get crds -o name | grep 'velero')
```

- **migration** クラスターロール:

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

- **migration-operator** クラスターロール:

```
$ oc delete clusterrole migration-operator
```

- **Velero** クラスターロール:

```
$ oc delete $(oc get clusterroles -o name | grep 'velero')
```

- **migration** クラスターのロールバインディング:

```
$ oc delete $(oc get clusterrolebindings -o name | grep 'migration.openshift.io')
```

- **migration-operator** クラスターロールバインディング:

```
$ oc delete clusterrolebindings migration-operator
```

- **Velero** クラスターのロールバインディング:

```
$ oc delete $(oc get clusterrolebindings -o name | grep 'velero')
```

## 第4章 ネットワークの制限された環境での MIGRATION TOOLKIT FOR CONTAINERS OPERATOR のインストール

以下の手順を実行して、ネットワークが制限された環境で OpenShift Container Platform 4 に MTC (Migration Toolkit for Containers) をインストールすることができます。

1. [ミラーリングされた Operator カタログ](#) を作成します。  
このプロセスは、[registry.redhat.io](#) イメージとミラーレジストリーイメージ間のマッピングを含む **mapping.txt** ファイルを作成します。**mapping.txt** ファイルは、**レガシー**の Migration Toolkit for Containers Operator を OpenShift Container Platform 4.2 から 4.5 ソースクラスターにインストールするために必要です。
2. Operator Lifecycle Manager を使用して OpenShift Container Platform 4.10 ターゲットクラスターに Migration Toolkit for Containers Operator をインストールします。  
デフォルトで、MTC Web コンソールおよび **Migration Controller** Pod はターゲットクラスターで実行されます。**Migration Controller** カスタムリソースマニフェストを、MTC の Web コンソールおよび **Migration Controller** Pod を [リモートクラスター](#) で実行するように設定できます。
3. Migration Toolkit for Containers Operator をソースクラスターにインストールします。
  - OpenShift Container Platform 4.6 以降: Operator Lifecycle Manager を使用して Migration Toolkit for Containers Operator をインストールします。
  - OpenShift Container Platform 4.2 から 4.5: コマンドラインインターフェイスからレガシー Migration Toolkit for Containers Operator をインストールします。
4. オブジェクトストレージをレプリケーションリポジトリとして使用するよう設定します。



### 注記

OpenShift Container Platform 3 に MTC をインストールするには、[OpenShift Container Platform 3 へのコンテナ Operator 用のレガシー Migration Toolkit のインストール](#) を参照してください。

MTC をアンインストールするには、[MTC のアンインストールとリソースの削除](#) を参照してください。

### 4.1. 互換性のガイドライン

OpenShift Container Platform バージョンと互換性がある MTC (Migration Toolkit for Containers) Operator をインストールする必要があります。

#### 定義

##### レガシープラットフォーム

OpenShift Container Platform 4.5 以前。

##### 最新プラットフォーム

OpenShift Container Platform 4.6 以降。

##### レガシー operator

レガシープラットフォーム用に設計された MTC Operator。

##### 最新 operator

最新のプラットフォーム用に設計された MTC Operator。

## コントロールクラスター

MTC コントローラーと GUI を実行するクラスター。

## リモートクラスター

Velero を実行する移行のソースクラスターまたは宛先クラスター。コントロールクラスターは、Velero API を介してリモートクラスターと通信し、移行を促進します。

表4.1 MTC 互換性: レガシープラットフォームからの移行

	OpenShift Container Platform 4.5 以前	OpenShift Container Platform 4.6 以降
安定した MTC バージョン	<p>MTC 1.7.z</p> <p>レガシー 1.7 operator: <b>operator.yml</b> ファイルを使用して手動でインストール。</p>  <p><b>重要</b></p> <p>このクラスターをコントロールクラスターにすることはできません。</p>	<p>MTC 1.7.z</p> <p>OLM を使用してインストール、リリースチャンネル <b>release-v1.7</b></p>

## 注記

ネットワークの制限により、最新のクラスターが移行に関与する他のクラスターに接続できないというエッジケースが存在します。たとえば、オンプレミスの OpenShift Container Platform 3.11 クラスターからクラウド内の最新の OpenShift Container Platform クラスターに移行する場合、最新のクラスターは OpenShift Container Platform 3.11 クラスターに接続できません。

MTC 1.7 では、ネットワーク制限のためにいずれかのリモートクラスターがコントロールクラスターと通信できない場合は、**crane tunnel-api** コマンドを使用します。

安定した MTC リリースでは、常に最新のクラスターを制御クラスターとして指定する必要がありますが、この特定のケースでは、レガシークラスターを制御クラスターとして指定し、ワークロードをリモートクラスターにプッシュすることができます。

## 4.2. MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の OPENSIFT CONTAINER PLATFORM 4.10 へのインストール

Operator Lifecycle Manager を使用して OpenShift Container Platform 4.10 に Migration Toolkit for Containers Operator をインストールします。

### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。
- ローカルレジストリーのミラーイメージで Operator カタログを作成する必要があります。

### 手順

1. OpenShift Container Platform Web コンソールで、**Operators** → **OperatorHub** をクリックします。
2. **Filter by keyword** フィールドを使用して、**Migration Toolkit for Containers Operator** を見つけます。
3. **Migration Toolkit for Containers Operator** を選択し、**Install** をクリックします。
4. **Install** をクリックします。  
**Installed Operators** ページで、**Migration Toolkit for Containers Operator** は、**Succeeded** のステータスで **openshift-migration** プロジェクトに表示されます。
5. **Migration Toolkit for Containers Operator** をクリックします。
6. **Provided APIs** の下で **Migration Controller** タイルを見つけ、**Create Instance** をクリックします。
7. **Create** をクリックします。
8. **Workloads** → **Pods** をクリックし、MTC Pod が実行されていることを確認します。

### 4.3. OPENSIFT CONTAINER PLATFORM 4.2 での従来の MIGRATION TOOLKIT FOR CONTAINERS OPERATOR の 4.5 へのインストール

レガシー Migration Toolkit for Containers Operator を手動で OpenShift Container Platform バージョン 4.2 から 4.5 にインストールできます。

#### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。
- **registry.redhat.io** にアクセスできる必要があります。
- **podman** がインストールされている必要があります。
- **registry.redhat.io** からファイルをダウンロードするには、ネットワークアクセスのある Linux ワークステーションが必要です。
- Operator カタログのミラーイメージを作成する必要があります。
- ミラーリングされた Operator カタログから Migration Toolkit for Containers Operator を OpenShift Container Platform 4.10 にインストールする必要があります。

#### 手順

1. Red Hat カスタマーポータル認証情報を使用して **registry.redhat.io** にログインします。

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

2. 次のコマンドを実行して、**operator.yml** ファイルをダウンロードします。

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

3. 次のコマンドを実行して、**controller.yml** ファイルをダウンロードします。

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

4. 以下のコマンドを実行して、Operator イメージマッピングを取得します。

```
$ grep openshift-migration-legacy-rhel8-operator ./mapping.txt | grep rhmtc
```

**mapping.txt** ファイルは Operator カタログのミラーリング時に作成されました。出力には、**registry.redhat.io** イメージとミラーレジストリーイメージ間のマッピングが表示されます。

### 出力例

```
registry.redhat.io/rhmtc/openshift-migration-legacy-rhel8-
operator@sha256:468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8
a=<registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-operator
```

5. **ansible** および **operator** コンテナの **image** 値、および **operator.yml** ファイルの **REGISTRY** 値を更新します。

```
containers:
- name: ansible
  image: <registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-
operator@sha256:
<468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a> ①
...
- name: operator
  image: <registry.apps.example.com>/rhmtc/openshift-migration-legacy-rhel8-
operator@sha256:
<468a6126f73b1ee12085ca53a312d1f96ef5a2ca03442bcb63724af5e2614e8a> ②
...
env:
- name: REGISTRY
  value: <registry.apps.example.com> ③
```

① ② ミラーレジストリーおよび Operator イメージの **sha256** 値を指定します。

③ ミラーレジストリーを指定します。

6. ソースクラスターにログインします。

7. Migration Toolkit for Containers Operator オブジェクトを作成します。

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

### 出力例

```
namespace/openshift-migration created
rolebinding.rbac.authorization.k8s.io/system:deployers created
serviceaccount/migration-operator created
customresourcedefinition.apiextensions.k8s.io/migrationcontrollers.migration.openshift.io
```

```

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

```

- 1** **Error from server (AlreadyExists)** メッセージは無視できます。これらは、以降のリソースで提供される OpenShift Container Platform 4 以前のバージョン用にリソースを作成する Migration Toolkit for Containers Operator が原因です。

8. **MigrationController** オブジェクトを作成します。

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

9. MTC Pod が実行されていることを確認します。

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

## 4.4. プロキシ設定

OpenShift Container Platform 4.1 以前のバージョンでは、これらのバージョンはクラスター全体の **proxy** オブジェクトをサポートしないため、Migration Toolkit for Containers Operator のインストール後に、**MigrationController** カスタムリソース (CR) マニフェストでプロキシを設定する必要があります。

OpenShift Container Platform 4.2 から 4.10 の場合、MTC (Migration Toolkit for Containers) はクラスター全体のプロキシ設定を継承します。クラスター全体のプロキシ設定を上書きする場合は、プロキシパラメーターを変更できます。

### 4.4.1. ボリュームの直接移行

MTC 1.4.2 で、ボリュームの直接移行 (DVM) が導入されました。DVM は1つのプロキシのみをサポートします。ターゲットクラスターもプロキシの背後にある場合、ソースクラスターはターゲットクラスターのルートにアクセスできません。

プロキシの背後にあるソースクラスターから DVM を実行する場合には、トランスポート層で機能する TCP プロキシを設定して、SSL 接続を独自の SSL 証明書で復号化および再暗号化せずに透過的に転送する必要があります。Stunnel プロキシは、このようなプロキシの例です。

#### 4.4.1.1. DVM の TCP プロキシ設定

TCP プロキシ経由でソースとターゲットクラスターの間には直接接続を設定し、プロキシを使用できるように **MigrationController** CR の **stunnel\_tcp\_proxy** 変数を設定できます。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration

```

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

ボリュームの直接移行 (DVM) は、プロキシの Basic 認証のみをサポートします。さらに、DVM は、TCP 接続を透過的にトンネルできるプロキシの背後でのみ機能します。中間者モードの HTTP/HTTPS プロキシは機能しません。既存のクラスター全体にわたるプロキシはこの動作をサポートしない可能性があります。その結果、DVM のプロキシ設定は、MTC の通常のプロキシ設定とは異なる状態に保たれます。

#### 4.4.1.2. HTTP/HTTPS プロキシの代わりに TCP プロキシを使用する理由

DVM を有効にするには、OpenShift ルートを介してソースおよびターゲットクラスター間で Rsync を実行します。トラフィックは、TCP プロキシである Stunnel を使用して暗号化されます。ソースクラスターで実行している Stunnel は、ターゲット Stunnel との TLS 接続を開始し、暗号化されたチャンネルでデータを転送します。

OpenShift のクラスター全体の HTTP/HTTPS プロキシは通常、外部サーバーで独自の TLS セッションをネゴシエートする中間者モードで設定されます。ただし、これは Stunnel では機能しません。Stunnel では、プロキシによって TLS セッションが変更されないようにする必要があります。基本的には、プロキシを透過的なトンネルにし、単純に TCP 接続をそのまま転送する必要があります。したがって、TCP プロキシを使用する必要があります。

#### 4.4.1.3. 既知の問題

##### 移行が **Upgrade request required** エラーで失敗する

移行コントローラーは SPDY プロトコルを使用してリモート Pod 内でコマンドを実行します。リモートクラスターがプロキシまたは、SPDY プロトコルをサポートしないファイアウォールの背後にある場合には、移行コントローラーはリモートコマンドの実行に失敗します。移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: SPDY プロトコルをサポートするプロキシを使用します。

SPDY プロトコルのサポートに加えて、このプロキシまたはファイアウォールでは、**Upgrade** HTTP ヘッダーを API サーバーに渡す必要もあります。クライアントはこのヘッダーを使用して API サーバーと Websocket 接続を開きます。**Upgrade** ヘッダーがプロキシまたはファイアウォールでブロックされると、移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: プロキシで **Upgrade** ヘッダーが転送されるようにしてください。

#### 4.4.2. 移行用のネットワークポリシーのチューニング

OpenShift は、クラスターで使用されるネットワークプラグインに基づいて **NetworkPolicy** または **EgressFirewalls** を使用した Pod との間のトラフィックの制限をサポートします。移行に関連するソース namespace のいずれかがこのようなメカニズムを使用して Pod へのネットワークトラフィックを制限する場合には、この制限により移行時に Rsync Pod へのトラフィックが誤って停止される可能性があります。

ソースおよびターゲットクラスターの両方で実行される Rsync Pod は OpenShift Route 経由で相互に接続する必要があります。既存の **NetworkPolicy** または **EgressNetworkPolicy** オブジェクトは、これらのトラフィックの制限が課されないように Rsync Pod を自動的に取り除くように設定できます。

#### 4.4.2.1. NetworkPolicy の設定

##### 4.4.2.1.1. Rsync Pod からの egress トラフィック



Rsync Pod の一意のラベルを使用し、同期元または同期先 namespace の **NetworkPolicy** 設定がこのタイプのトラフィックをブロックする場合に egress トラフィックがそれらを通過することを許可できます。以下のポリシーは、namespace の Rsync Pod からの全 egress トラフィックを許可します。

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

#### 4.4.2.1.2. Rsync Pod への ingress トラフィック

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

#### 4.4.2.2. EgressNetworkPolicy 設定

**EgressNetworkPolicy** オブジェクトまたは **Egress ファイアウォール** は、egress トラフィックをクラスターからブロックするために設計された OpenShift コンストラクトです。

**NetworkPolicy** オブジェクトとは異なり、Egress ファイアウォールは namespace のすべての Pod に適用されるためにプロジェクトレベルで機能します。そのため、Rsync Pod の一意のラベルを使用すると、この制限から除外するのは Rsync Pod だけではありません。ただし、ソースおよびターゲットクラスターの CIDR 範囲をポリシーの **Allow** ルールに追加して、2つのクラスター間で直接接続を設定できます。

Egress ファイアウォールが存在するクラスターに基づいて、他のクラスターの CIDR 範囲を追加して、2つの間の egress トラフィックを許可できます。

```
apiVersion: network.openshift.io/v1
kind: EgressNetworkPolicy
metadata:
  name: test-egress-policy
  namespace: <namespace>
spec:
```

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

#### 4.4.2.3. データ転送用の代替エンドポイントの選択

デフォルトでは、DVM は OpenShift Container Platform ルートをエンドポイントとして使用して、PV データを宛先クラスターに転送します。クラスターポロジで許可されている場合は、サポートされている別の種類のエンドポイントを選択できます。

クラスターごとに、**MigrationController** CR で適切な **宛先** クラスターに **rsync\_endpoint\_type** 変数を設定することで、エンドポイントを設定できます。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  [...]
  rsync_endpoint_type: [NodePort|ClusterIP|Route]
```

#### 4.4.2.4. RsyncPod の補足グループの設定

PVC が共有ストレージを使用する場合、Pod がアクセスを許可するように RsyncPod 定義に補足グループを追加して、そのストレージへのアクセスを設定できます。

表4.2 RsyncPod の補足グループ

変数	タイプ	デフォルト	説明
<b>src_supplemental_groups</b>	string	設定されていません	ソース Rsync Pod の補足グループのコンマ区切りリスト
<b>target_supplemental_groups</b>	string	設定されていません	ターゲット Rsync Pod の補足グループのコンマ区切りリスト

#### 使用例

**MigrationController** CR を更新して、これらの補足グループの値を設定できます。

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

#### 4.4.3. プロキシの設定

##### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。

## 手順

1. **MigrationController** CR マニフェストを取得します。

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

2. プロキシパラメーターを更新します。

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

- ❶ ボリュームの直接移行のための Stunnel プロキシ URL。
- ❷ プロキシを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。

サブドメインのみと一致するように、ドメインの前に `.` を付けます。たとえば、`.y.com` は `x.y.com` に一致しますが、`y.com` には一致しません。\* を使用し、すべての宛先のプロキシをバイパスします。インストール設定で `networking.machineNetwork[].cidr` フィールドで定義されるネットワークに含まれていないワーカーをスケールアップする場合、それらをこの一覧に追加し、接続の問題を防ぐ必要があります。

`httpProxy` または `httpsProxy` フィールドのいずれも設定されていない場合、このフィールドは無視されます。

3. マニフェストを **migration-controller.yaml** として保存します。
4. 更新したマニフェストを適用します。

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

詳細は、[Configuring the cluster-wide proxy](#) を参照してください。

## 4.5. レプリケーションリポジトリの設定

Multicloud Object Gateway は、ネットワークが制限された環境で唯一サポートされるオプションです。

MTC は、ソースクラスターからターゲットクラスターにデータを移行するために、[ファイルシステムおよびスナップショットによるデータのコピー方法](#) をサポートします。ご使用の環境に適した方法で、ストレージプロバイダーでサポートされる方法を選択できます。

### 4.5.1. 前提条件

- すべてのクラスターには、レプリケーションリポジトリへの中断されないネットワークアクセスが必要です。
- 内部でホストされるレプリケーションリポジトリでプロキシサーバーを使用する場合は、プロキシがレプリケーションリポジトリへのアクセスを許可することを確認する必要があります。

#### 4.5.2. Multicloud Object Gateway の認証情報の取得

OpenShift API for Data Protection (OADP) の **Secret** カスタムリソース (CR) を作成するには、Multicloud Object Gateway (MCG) 認証情報を取得する必要があります。

MCG は、OpenShift Data Foundation のコンポーネントです。

##### 前提条件

- 適切な [OpenShift Data Foundation deployment guide](#) を使用して、OpenShift Data Foundation をデプロイする必要があります。

##### 手順

1. **NooBaa** カスタムリソースで **describe** コマンドを実行して、S3 エンドポイントである **AWS\_ACCESS\_KEY\_ID** および **AWS\_SECRET\_ACCESS\_KEY** を取得します。

#### 4.5.3. 関連情報

- Red Hat OpenShift Data Foundation ドキュメントの [Disconnected environment](#)。
- [MTC ワークフロー](#)
- [データのコピー方法](#)
- [MTC の Web コンソールへのレプリケーションリポジトリの追加](#)

## 4.6. MTC のアンインストールおよびリソースの削除

MTC (Migration Toolkit for Containers) をアンインストールし、そのリソースを削除してクラスターをクリーンアップできます。



##### 注記

**velero** CRD を削除すると、Velero がクラスターから削除されます。

##### 前提条件

- **cluster-admin** 権限を持つユーザーとしてログインしている。

##### 手順

1. すべてのクラスターで **MigrationController** カスタムリソース (CR) を削除します。

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

- Operator Lifecycle Manager を使用して OpenShift Container Platform 4 の Migration Toolkit for Containers Operator をアンインストールします。
- 以下のコマンドを実行して、すべてのクラスターでクラスタースコープのリソースを削除します。

- **migration** カスタムリソース定義 (CRDs):

```
$ oc delete $(oc get crds -o name | grep 'migration.openshift.io')
```

- **Velero** CRD:

```
$ oc delete $(oc get crds -o name | grep 'velero')
```

- **migration** クラスターロール:

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

- **migration-operator** クラスターロール:

```
$ oc delete clusterrole migration-operator
```

- **Velero** クラスターロール:

```
$ oc delete $(oc get clusterroles -o name | grep 'velero')
```

- **migration** クラスターのロールバインディング:

```
$ oc delete $(oc get clusterrolebindings -o name | grep 'migration.openshift.io')
```

- **migration-operator** クラスターロールバインディング:

```
$ oc delete clusterrolebindings migration-operator
```

- **Velero** クラスターのロールバインディング:

```
$ oc delete $(oc get clusterrolebindings -o name | grep 'velero')
```

## 第5章 MIGRATION TOOLKIT FOR CONTAINERS のアップグレード

Migration Toolkit for Containers (MTC) は Operator Lifecycle Manager を使用して OpenShift Container Platform 4.10 でアップグレードできます。

従来の Migration Toolkit for Containers Operator を再インストールすることにより、OpenShift Container Platform 4.5 以前のバージョンで MTC をアップグレードできます。



### 重要

MTC バージョン 1.3 からアップグレードする場合、**MigPlan** カスタムリソース (CR) を更新する追加の手順を実行する必要があります。

### 5.1. OPENSIFT CONTAINER PLATFORM 4.10 での MTC(MIGRATION TOOLKIT FOR CONTAINERS) のアップグレード

Migration Toolkit for Containers (MTC) は Operator Lifecycle Manager を使用して OpenShift Container Platform 4.10 でアップグレードできます。

#### 前提条件

- **cluster-admin** 権限を持つユーザーとしてログインしている。

#### 手順

1. OpenShift Container Platform コンソールで、**Operators** → **Installed Operators** に移動します。  
更新が保留中の Operator は **Upgrade available** のステータスを表示します。
2. **Migration Toolkit for Containers Operator** をクリックします。
3. **Subscription** タブをクリックします。アップグレードの承認を必要とするアップグレードは、**Upgrade Status** の横に表示されます。たとえば、**1 requires approval** が表示される可能性があります。
4. **1 requires approval** をクリックしてから、**Preview Install Plan** をクリックします。
5. アップグレードに利用可能なリソースとして一覧表示されているリソースを確認し、**Approve** をクリックします。
6. **Operators** → **Installed Operators** ページに戻り、アップグレードの進捗をモニターします。完了時に、ステータスは **Succeeded** および **Up to date** に変更されます。
7. **Workloads** → **Pods** をクリックし、MTC Pod が実行されていることを確認します。

### 5.2. OPENSIFT CONTAINER PLATFORM バージョン 4.2 の MTC (MIGRATION TOOLKIT FOR CONTAINERS) 4.5 へのアップグレード

MTC (Migration Toolkit for Containers) は、レガシーの Migration Toolkit for Containers Operator を手動でインストールすることで、OpenShift Container Platform バージョン 4.2 で 4.5 にアップグレードできます。

## 前提条件

- **cluster-admin** 権限を持つユーザーとしてログインしている。
- **registry.redhat.io** にアクセスできる必要があります。
- **podman** がインストールされている必要があります。

## 手順

1. 以下のコマンドを実行して、Red Hat Customer Portal の認証情報で **registry.redhat.io** にログインします。

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

2. 次のコマンドを実行して、**operator.yml** ファイルをダウンロードします。

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

3. 次のコマンドを実行して、Migration Toolkit for Containers Operator を置き換えます。

```
$ oc replace --force -f operator.yml
```

4. 次のコマンドを実行して、**migration-operator** デプロイメントを **0** にスケールし、デプロイメントを停止します。

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

5. **migration-operator** デプロイメントを **1** にスケールしてデプロイメントを開始し、次のコマンドを実行して変更を適用します。

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

6. 次のコマンドを実行して、**migration-operator** がアップグレードされたことを確認します。

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

7. 次のコマンドを実行して、**controller.yml** ファイルをダウンロードします。

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

8. 次のコマンドを実行して、**migration-controller** オブジェクトを作成します。

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

9. 次のコマンドを実行して、MTC Pod が実行されていることを確認します。

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

## 5.3. MTC 1.3 から 1.7 へのアップグレード

MTC (Migration Toolkit for Containers) バージョン 1.3.x を 1.7 にアップグレードする場  
合、**MigrationController** Pod が実行されているクラスターで **MigPlan** カスタムリソース (CR) マニ  
フェストを更新する必要があります。

**indirectImageMigration** および **indirectVolumeMigration** パラメーターは MTC 1.3 に存在しないた  
め、バージョン 1.4 のそれらのデフォルト値は **false** になります。つまり、イメージの直接移行および  
ボリュームの直接移行が有効にされます。直接移行の要件が満たされないため、これらのパラメーター  
の値が **true** に変更されない限り、移行計画は **Ready** 状態になりません。

### 前提条件

- **cluster-admin** 権限を持つユーザーとしてログインしている。

### 手順

1. **MigrationController** Pod が実行されるクラスターにログインします。
2. **MigPlan** CR マニフェストを取得します。

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

3. 以下のパラメーター値を更新し、ファイルを **migplan.yaml** として保存します。

```
...
spec:
  indirectImageMigration: true
  indirectVolumeMigration: true
```

4. **MigPlan** CR マニフェストを置き換えて変更を適用します。

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

5. 更新された **MigPlan** CR マニフェストを取得して変更を確認します。

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



## 第6章 移行前のチェックリスト

MTC (Migration Toolkit for Containers) を使用してアプリケーションのワークロードを移行する前に、以下のチェックリストを確認してください。

### 6.1. クラスターヘルスチェックリスト

- クラスターが特定のプラットフォームおよびインストール方法についての最小ハードウェア要件を満たしている (ベアメタル など)。
- すべての MTC の前提条件を満たしている。
- すべてのノードに有効な OpenShift Container Platform サブスクリプションがある。
- ノードの正常性を確認している。
- アイデンティティプロバイダーが機能している。
- 移行ネットワークの最小スループットは 10 Gbps である。
- クラスターには移行用に十分なリソースがある。



#### 注記

クラスターには、通常のワークロードにほかに移行を実行するために、追加のメモリー、CPU、およびストレージが必要です。実際のリソース要件は、単一の移行計画で移行される Kubernetes リソースの数によって異なります。リソース要件を見積もるため、非実稼働環境で移行をテストする必要があります。

- クラスターの etcd ディスクのパフォーマンスが fio で確認されている。

### 6.2. ソースクラスターのチェックリスト

- 以下のコマンドを実行して、異常な設定のある永続ボリューム (PV) が Terminating 状態のままであることを確認している。

```
$ oc get pv
```

- 以下のコマンドを実行して、ステータスが Running または Completed 以外の Pod の有無について確認している。

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

- 以下のコマンドを実行して、再起動数の高い Pod の有無について確認している。

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

Pod が Running 状態であっても、再起動数が多くなると、根本的な問題を示唆している可能性があります。

- クラスター証明書が移行プロセスの期間 有効である。

以下のコマンドを実行して、保留中の証明書署名要求の有無を確認している。

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

レジストリーは [推奨ストレージタイプ](#) を使用している。

イメージをレジストリーを読み取り、これに書き込むことができる。

[etcd クラスタ](#) が正常である。

ソースクラスタの [平均の API サーバーの応答時間](#) は 50 ミリ秒未満である。

### 6.3. ターゲットクラスタのチェックリスト

クラスタに、データベース、ソースコードリポジトリ、コンテナイメージレジストリー、CI/CD ツールなどの外部サービスにアクセスするための正しいネットワーク設定およびパーミッションがある。

クラスタによって提供されるサービスを使用する外部アプリケーションおよびサービスに、クラスタにアクセスするための正しいネットワーク設定およびパーミッションがある。

内部コンテナイメージの依存関係の要件を満たしている。

ターゲットクラスタおよびレプリケーションリポジトリに十分なストレージ容量がある。

## 第7章 ネットワークの考慮事項

移行後にアプリケーションネットワークトラフィックをリダイレクトするための戦略を確認します。

### 7.1. DNS に関する考慮事項

ターゲットクラスターの DNS ドメインは、ソースクラスターのドメインとは異なります。デフォルトでは、アプリケーションは移行後にターゲットクラスターの FQDN を取得します。

移行したアプリケーションのソース DNS ドメインを保持するには、以下で説明する 2 つのオプションのいずれかを選択します。

#### 7.1.1. クライアントからのターゲットクラスターの DNS ドメイン分離

ターゲットクラスターをクライアントに公開せずに、ソースクラスターの DNS ドメインに送信されるクライアントの要求がターゲットクラスターの DNS ドメインに到達するように許可できます。

##### 手順

1. クライアントとターゲットクラスターの間、アプリケーションロードバランサーやリバースプロキシなどの外部ネットワークコンポーネントを配置します。
2. DNS サーバーのソースクラスターのアプリケーション FQDN を更新して、外部ネットワークコンポーネントの IP アドレスを返します。
3. ソースドメインのアプリケーション用に受信された要求をターゲットクラスタードメインのロードバランサーに送信するようにネットワークコンポーネントを設定します。
4. ソースクラスターのロードバランサーの IP アドレスを参照する `*.apps.source.example.com` ドメインのワイルドカード DNS レコードを作成します。
5. ターゲットクラスターの前に、外部ネットワークコンポーネントの IP アドレスを参照する各アプリケーションに DNS レコードを作成します。特定の DNS レコードの優先順位はワイルドカードレコードよりも高くなるため、アプリケーションの FQDN の解決時に競合は発生しません。

##### 注記

- 外部ネットワークコンポーネントは、すべてのセキュアな TLS 接続を終了する必要があります。接続がターゲットクラスターロードバランサーに渡されると、ターゲットアプリケーションの FQDN がクライアントに公開され、証明書エラーが発生します。
- アプリケーションは、ターゲットクラスタードメインを参照するリンクをクライアントに返すことはできません。そうしないと、アプリケーションの一部が正しくロードまたは機能しない可能性があります。

#### 7.1.2. ソース DNS ドメインを受け入れるためのターゲットクラスターの設定

ソースクラスターの DNS ドメインで移行したアプリケーションの要求を受け入れるようにターゲットクラスターを設定できます。

##### 手順

セキュアではない HTTP アクセスとセキュアな HTTPS アクセスの両方で、以下の手順を実行します。

1. ソースクラスター内のアプリケーションの FQDN にアドレス指定された要求を受け入れるように設定されたターゲットクラスターのプロジェクトにルートを作成します。

```
$ oc expose svc <app1-svc> --hostname <app1.apps.source.example.com> \
-n <app1-namespace>
```

この新規ルートが有効な場合には、サーバーはその FQDN の要求を受け入れて、対応するアプリケーション Pod に送信します。さらに、アプリケーションを移行すると、ターゲットクラスタードメインに別のルートが作成されます。要求は、これらのホスト名のいずれかを使用して、移行されたアプリケーションに到達します。

2. ソースクラスター内のアプリケーションの FQDN がターゲットクラスターのデフォルトのロードバランサーの IP アドレスを指す DNS プロバイダーを使用して DNS レコードを作成します。これにより、トラフィックがソースクラスターからターゲットクラスターにリダイレクトされます。

アプリケーションの FQDN は、ターゲットクラスターのロードバランサーに対して解決します。デフォルトの Ingress コントローラーは、そのホスト名のルートが公開されるため、対象となる FQDN の要求を受け入れます。

セキュアな HTTPS アクセスには、以下の追加手順を実行します。

1. インストールプロセス中に作成されたデフォルトの Ingress コントローラーの x509 証明書をカスタム証明書に置き換えます。
2. この証明書に、**subjectAltName** フィールドにソースおよびターゲットクラスター両方のワイルドカード DNS ドメインが含まれるように設定します。  
新しい証明書は、いずれかの DNS ドメインを使用して確立された接続を保護するために有効です。

## 関連情報

- 詳細は、[デフォルト ingress 証明書の置き換え](#) を参照してください。

## 7.2. ネットワークトラフィックリダイレクト戦略

移行が成功したら、ステートレスアプリケーションのネットワークトラフィックをソースクラスターからターゲットクラスターにリダイレクトする必要があります。

ネットワークトラフィックをリダイレクトするための戦略は、次の前提に基づいています。

- アプリケーション Pod は、ソースクラスターとターゲットクラスターの両方で実行しています。
- 各アプリケーションには、ソースクラスターのホスト名を含むルートがあります。
- ソースクラスターのホスト名を持つルートには、CA 証明書が含まれています。
- HTTPS の場合、ターゲットルーターの CA 証明書には、ソースクラスターのワイルドカード DNS レコードのサブジェクト代替名が含まれています。

次の戦略を検討し、目的に合った戦略を選択してください。

- すべてのアプリケーションのすべてのネットワークトラフィックを同時にリダイレクトします。  
ソースクラスターのワイルドカード DNS レコードを変更して、ターゲットクラスタールーターの仮想 IP アドレス (VIP) を指すようにします。

この戦略は、単純なアプリケーションまたは小規模な移行に適しています。

- 個々のアプリケーションのネットワークトラフィックをリダイレクトする  
ソースクラスターのホスト名がターゲットクラスタールーターのVIPを指すように、アプリケーションごとにDNSレコードを作成します。このDNSレコードは、ソースクラスターのワイルドカードDNSレコードよりも優先されます。
- 個々のアプリケーションのネットワークトラフィックを徐々にリダイレクトする
  1. アプリケーションごとに、ソースクラスタールーターのVIPとターゲットクラスタールーターのVIPの両方にトラフィックを転送できるプロキシを作成します。
  2. ソースクラスターのホスト名がプロキシを指すように、アプリケーションごとにDNSレコードを作成します。
  3. トラフィックの一部をターゲットクラスタールーターのVIPにルーティングし、残りのトラフィックをソースクラスタールーターのVIPにルーティングするように、アプリケーションのプロキシエントリを設定します。
  4. すべてのネットワークトラフィックがリダイレクトされるまで、ターゲットクラスタールーターのVIPにルーティングするトラフィックの割合を徐々に増やします。
- 個々のアプリケーションのトラフィックのユーザーベースのリダイレクト  
この戦略を使用すると、ユーザー要求のTCP/IPヘッダーをフィルタリングして、事前定義されたユーザーグループのネットワークトラフィックをリダイレクトできます。これにより、ネットワークトラフィック全体をリダイレクトする前に、特定のユーザー集団でリダイレクトプロセスをテストできます。
  1. アプリケーションごとに、ソースクラスタールーターのVIPとターゲットクラスタールーターのVIPの両方にトラフィックを転送できるプロキシを作成します。
  2. ソースクラスターのホスト名がプロキシを指すように、アプリケーションごとにDNSレコードを作成します。
  3. **test customers** など、特定のヘッダーパターンに一致するトラフィックをターゲットクラスタールーターのVIPにルーティングし、残りのトラフィックをソースクラスタールーターのVIPにルーティングするように、アプリケーションのプロキシエントリを設定します。
  4. すべてのトラフィックがターゲットクラスタールーターのVIPに到達するまで、トラフィックをターゲットクラスタールーターのVIPに段階的にリダイレクトします。

## 第8章 アプリケーションの移行

MTC (Migration Toolkit for Containers) の Web コンソールまたは [コマンドライン](#) でアプリケーションを移行できます。

ほとんどのクラスタースコープのリソースは MTC で処理されません。アプリケーションがクラスタースコープのリソースを必要とする場合、ターゲットクラスターでそれらを手動で作成する必要がある場合があります。

段階移行とカットオーバー移行を使用して、クラスター間でアプリケーションを移行することができます。

- Stage 移行は、アプリケーションを停止せずにデータをソースクラスターからターゲットクラスターにコピーします。段階移行は複数回実行して、カットオーバー移行の期間を短縮できます。
- Cutover 移行はソースクラスターのトランザクションを停止し、リソースをターゲットクラスターに移動します。

状態の移行を使用して、アプリケーションの状態を移行できます。

- 状態の移行は、選択した永続ボリューム要求 (PVC) をコピーします。
- 状態の移行を使用して、同じクラスター内の namespace を移行できます。

移行時に、MTC (Migration Toolkit for Containers) は以下の namespace アノテーションを保持します。

- **openshift.io/sa.scc.mcs**
- **openshift.io/sa.scc.supplemental-groups**
- **openshift.io/sa.scc.uid-range**  
これらのアノテーションは UID 範囲を保持し、コンテナがターゲットクラスターのファイルシステムのパーミッションを保持できるようにします。移行された UID が、ターゲットクラスターの既存の namespace または今後の namespace 内の UID を重複させるリスクがあります。

### 8.1. 移行の前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。

#### イメージの直接移行

- ソースクラスターのセキュアな OpenShift イメージレジストリーが公開されていることを確認する必要があります。
- 公開されるレジストリーへのルートを作成しておく必要があります。

#### ボリュームの直接移行

- クラスターがプロキシを使用する場合に Stunnel TCP プロキシを設定している必要があります。

#### クラスター

- ソースクラスターは、最新の z-stream リリースにアップグレードされる必要があります。
- MTC のバージョンは、すべてのクラスターで同一である必要があります。

## ネットワーク

- クラスターには、レプリケーションリポジトリに対して、また各クラスター間で無制限のネットワークアクセスが必要です。
- **move** を使用して永続ボリュームをコピーする場合、クラスターにはリモートボリュームへの無制限のネットワークアクセスが必要です。
- OpenShift Container Platform 4 クラスターで以下のポートを有効にする必要があります。
  - **6443** (API サーバー)
  - **443** (ルート)
  - **53** (DNS)
- TLS を使用している場合は、レプリケーションリポジトリでポート **443** を有効にする必要があります。

## 永続ボリューム (PV)

- PV は有効である必要があります。
- PV は永続ボリューム要求にバインドされる必要があります。
- スナップショットを使用して PV をコピーする場合には、以下の前提条件が追加されます。
  - クラウドプロバイダーはスナップショットをサポートしている必要があります。
  - PV に同じクラウドプロバイダーがなければなりません。
  - PV は同じ地理的リージョンにある必要があります。
  - PV には同じストレージクラスがなければなりません。

## 8.2. MTC の WEB コンソールを使用したアプリケーションの移行

クラスターおよびレプリケーションリポジトリを MTC の Web コンソールを使用して設定する必要があります。次に、移行計画を作成し、これを実行できます。

### 8.2.1. MTC の Web コンソールの起動

ブラウザで MTC (Migration Toolkit for Containers) Web コンソールを起動できます。

#### 前提条件

- MTC の Web コンソールには、OpenShift Container Platform Web コンソールにアクセスする必要があります。
- MTC の Web コンソールには、OAuth 認証サーバーへのネットワークアクセスが必要です。





```
a3ViZXJuZXRlcy5pby9zZXJ2aWNlYWVudC9uYW1lc3BhY2UiOiJtaWciLCJrdWJlcm5ldGV:
LmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoibWlnLXRva2VuLWVs4dDjyIiwia3ViZXJuZ
XRlcy5pby9zZXJ2aWNlYWVudC9zZXJ2aWNlWFJY291bnQubmFtZSI6Im1pZylsImt1YmV
ybmv0ZXMuaW8vc2VydmljZWJY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImE1YjFiYWw
wLWmxYmYtMTFOS05Y2NiLTAYOWRmODYwYjMwOCIsInN1Yil6InN5c3RlbnR1bTpxZXJ2aWNlY
WNjb3VudDptaWc6bWlnIn0.xqeeAINK7UXpdRqAtOj70qhBJPeMwmgLomV9iFxr5RoqUgKchZ
RG2J2rkqmPm6vr7K-
cm7ibD1lBpdQJCcVDuoHYsFgV4mp9vgOfn9osSDp2TGikwNz4Az95e81xnjVUmzh-
NjDsEpw71DH92iHV_xt2sTwtzftS49LpPW2LjrV0evtNBP_t_RfskdArt5VSv25eORI7zScqfe1CiM
kcVbf2UqACQjo3LbkpfN26HAioO2oH0ECPiRzT0Xyh-KwFutJLS9Xgghyw-
LD9kPKcE_xbbJ9Y4Rqajh7WdPYuB0Jd9DPVrslmzK-F6cgHHYoZEv0SvLQi-
PO0rpDrcjOEEQQ
```

3. MTC の Web コンソールで、**Clusters** をクリックします。
4. **Add cluster** をクリックします。
5. 以下のフィールドに値を入力します。
  - **Cluster name:** クラスタ名には、小文字 (**a-z**) および数字 (**0-9**) を含めることができます。スペースや国際的な文字を含めることはできません。
  - **URL:** API サーバー URL を指定します (例: **https://<www.example.com>:8443**)。
  - **Service account token migration-controller** サービスアカウントトークンを貼り付けます。
  - **Exposed route host to image registry** イメージの直接移行を使用している場合、ソースクラスタのイメージレジストリーへの公開されたルート指定します。以下のコマンドを実行してルートを作成します。
    - OpenShift Container Platform 3 の場合:

```
$ oc create route passthrough --service=docker-registry --port=5000 -n default
```
    - OpenShift Container Platform 4 の場合:

```
$ oc create route passthrough --service=image-registry --port=5000 -n openshift-image-registry
```
  - **Azure クラスタ:** Azure スナップショットを使用してデータをコピーする場合は、このオプションを選択する必要があります。
  - **Azure resource group:** このフィールドは、**Azure cluster** が選択されている場合に表示されます。Azure リソースグループを指定します。
  - **Require SSL verification:** オプション: このオプションを選択してクラスタへの SSL 接続を検証します。
  - **CA bundle file:** このフィールドは、**Require SSL verification** が選択されている場合に表示されます。自己署名証明書用にカスタム CA 証明書バンドルファイルを作成している場合は、**Browse** をクリックして CA バンドルファイルを選択し、これをアップロードします。
6. **Add cluster** をクリックします。  
クラスタが **Clusters** 一覧に表示されます。

### 8.2.3. MTC の Web コンソールへのレプリケーションリポジトリの追加

MTC (Migration Toolkit for Containers) の Web コンソールに、オブジェクトストレージをレプリケーションリポジトリとして追加できます。

MTC は、以下のストレージプロバイダーをサポートしています。

- Amazon Web Services (AWS) S3
- Multi-Cloud Object Gateway (MCG)
- 汎用 S3 オブジェクトストレージ (例: Minio または Ceph S3)
- Google Cloud Provider (GCP)
- Microsoft Azure Blob

#### 前提条件

- オブジェクトストレージをレプリケーションリポジトリとして設定する必要があります。

#### 手順

1. MTC の Web コンソールで、**Replication repositories** をクリックします。
2. **Add repository** をクリックします。
3. **Storage provider type** を選択し、以下のフィールドに入力します。
  - **AWS および MCG を含む S3 プロバイダー向けの AWS**
    - **Replication repository name** MTC の Web コンソールでレプリケーションリポジトリ名を指定します。
    - **S3 bucket name**: S3 バケットの名前を指定します。
    - **S3 bucket region**: S3 バケットリージョンを指定します。AWS S3 の場合に**必須**です。一部の S3 プロバイダーの場合は、**オプション**になります。予測値については、S3 プロバイダーの製品ドキュメントを確認してください。
    - **S3 endpoint**: バケットではなく S3 サービスの URL を指定します (例: **https://<s3-storage.apps.cluster.com>**)。汎用 S3 プロバイダーの場合は**必須**です。**https://** 接頭辞を使用する必要があります。
    - **S3 provider access key**: AWS の場合は **<AWS\_SECRET\_ACCESS\_KEY>** を指定し、MCG および他の S3 プロバイダーの場合は S3 プロバイダーアクセスキーを指定します。
    - **S3 provider secret access key**: AWS の場合は **<AWS\_ACCESS\_KEY\_ID>** を指定し、MCG および他の S3 プロバイダーの場合は S3 プロバイダーシークレットアクセスキーを指定します。
    - **Require SSL verification**: 汎用 S3 プロバイダーを使用している場合は、このチェックボックスをクリアします。
    - 自己署名証明書用にカスタム CA 証明書バンドルを作成している場合は、**Browse** をクリックして Base64 でエンコードされたファイルを参照します。

- **GCP:**
    - **Replication repository name** MTC の Web コンソールでレプリケーションリポジトリ名を指定します。
    - **GCP bucket name**: GCP バケットの名前を指定します。
    - **GCP credential JSON blob: credentials-velero** ファイルに文字列を指定します。
  - **Azure:**
    - **Replication repository name** MTC の Web コンソールでレプリケーションリポジトリ名を指定します。
    - **Azure resource group**: Azure Blob ストレージのリソースグループを指定します。
    - **Azure storage account name**: Azure Blob ストレージアカウント名を指定します。
    - **Azure credentials - INI file contents credentials-velero** ファイルに文字列を指定します。
4. **Add repository** をクリックし、接続の検証を待機します。
  5. **Close** をクリックします。  
新規リポジトリが **Replication repositories** 一覧に表示されます。

#### 8.2.4. MTC の Web コンソールでの移行計画の作成

MTC (Migration Toolkit for Containers) Web コンソールで移行計画を作成できます。

##### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインしている必要があります。
- 同じ MTC バージョンがすべてのクラスターにインストールされていることを確認する必要があります。
- クラスターおよびレプリケーションリポジトリを MTC の Web コンソールに追加する必要があります。
- **move** データコピー方法を使用して永続ボリューム (PV) を移行する場合、ソースクラスターおよびターゲットクラスターには、リモートボリュームへの中断されないネットワークアクセスが必要です。
- イメージの直接移行を使用する必要がある場合は、ソースクラスターのイメージレジストリーに公開されたルートを指定する必要があります。これは、MTC の Web コンソールまたは **MigCluster** カスタムリソースマニフェストを更新して実行できます。

##### 手順

1. MTC Web コンソールで、**Migration plans** をクリックします。
2. **Add migration plan** をクリックします。
3. **Plan name** を入力します。

移行計画名には、253 以上の小文字の英数字 (**a-z, 0-9**) を使用できず、スペースやアンダースコア (**\_**) を含めることはできません。

4. **Source cluster**、**Target cluster**、および **Repository** を選択します。
5. **Next** をクリックします。
6. 移行用のプロジェクトを選択します。
7. オプション: プロジェクトの横にある編集アイコンをクリックして、ターゲットの namespace を変更します。
8. **Next** をクリックします。
9. 各 PV の **移行タイプ** を選択します。
  - **Copy** オプションは、ソースクラスターの PV のデータをレプリケーションリポジトリにコピーしてから、データを同様の特徴のある新規に作成された PV でターゲットクラスターで復元します。
  - **Move** オプションは、ソースクラスターからリモートボリューム (例: NFS) をアンマウントし、リモートボリュームをポイントするターゲットクラスターで PV リソースを作成し、その後リモートボリュームをターゲットクラスターにマウントします。ターゲットクラスターで実行されているアプリケーションは、ソースクラスターが使用していたものと同じリモートボリュームを使用します。
10. **Next** をクリックします。
11. 各 PV の **Copy method** を選択します。
  - **スナップショットのコピー** は、クラウドプロバイダーのスナップショット機能を使用してデータのバックアップおよび復元を行います。この場合、**ファイルシステムのコピー** を使用する場合よりもはるかに高速になります。
  - **ファイルシステムのコピー** は、ソースクラスターのファイルをバックアップし、それらをターゲットクラスターで復元します。  
ファイルシステムのコピー方法は、ボリュームの直接移行に必要です。
12. **Verify copy** を選択して、**ファイルシステムのコピー** で移行されたデータを確認します。データは、各ソースファイルのチェックサムを生成し、復元後のチェックサムを確認して検証されます。データ検証は、パフォーマンスを大幅に低下させます。
13. **Target storage class** を選択します。  
**Filesystem copy** を選択している場合、ターゲットストレージクラスを変更できます。
14. **Next** をクリックします。
15. **Migration options** ページで、ソースクラスターに公開されたイメージレジストリルートを設定した場合に **Direct image migration** オプションが選択されます。**Filesystem copy** でデータを移行する場合、**Direct PV migration** オプションが選択されます。  
直接の移行オプションは、イメージおよびファイルをソースクラスターからターゲットクラスターに直接コピーします。このオプションは、イメージおよびファイルをソースクラスターからレプリケーションリポジトリにコピーしてから、レプリケーションリポジトリからターゲットクラスターにコピーする場合よりもはるかに高速になります。
16. **Next** をクリックします。
17. オプション: **Add Hook** をクリックし、移行計画にフックを追加します。

フックはカスタムコードを実行します。単一の移行計画計画に最大 4 つのフックを追加できません。各フックは異なる移行ステップで実行されます。

- a. Web コンソールに表示するフックの名前を入力します。
- b. フックが Ansible Playbook の場合は **Ansible playbook** を選択し、**Browse** をクリックして Playbook をアップロードするか、フィールドに Playbook の内容を貼り付けます。
- c. オプション: デフォルトのフックイメージを使用していない場合は、Ansible ランタイムイメージを指定します。
- d. フックが Ansible Playbook ではない場合には、**Custom container image** をクリックし、イメージ名とパスを指定します。  
カスタムコンテナイメージには、Ansible Playbook を含めることができます。
- e. **Source cluster** または **Target cluster** を選択します。
- f. **Service account name** および **Service account namespace** を入力します。
- g. フックの移行手順を選択します。
  - **preBackup**: アプリケーションのワークロードがソースクラスターでバックアップされる前
  - **postBackup**: アプリケーションのワークロードがソースクラスターでバックアップされた後
  - **preRestore**: アプリケーションのワークロードがターゲットクラスターで復元される前
  - **postRestore**: アプリケーションのワークロードがターゲットクラスターで復元された後
- h. **Add** をクリックします。

18. **Finish** をクリックします。  
移行計画は、**Migration plans** 一覧に表示されます。

## 永続ボリュームのコピー方法に関する他のリソース

- [MTC ファイルシステムのコピー方法](#)
- [MTC スナップのコピー方法](#)

### 8.2.5. MTC の Web コンソールでの移行計画の実行

MTC (Migration Toolkit for Containers) の Web コンソールで作成した移行計画を使用してアプリケーションとデータを移行できます。



#### 注記

移行時に、MTC は移行された永続ボリューム (PV) の回収ポリシーをターゲットクラスターで **Retain** に設定します。

**Backup** カスタムリソースには、元の回収ポリシーを示す **PVOriginalReclaimPolicy** アノテーションが含まれます。移行した PV の回収ポリシーを手動で復元できます。


## 前提条件

MTC の Web コンソールには以下が含まれている必要があります。

- **Ready** 状態のソースクラスター
- **Ready** 状態のターゲットクラスター
- レプリケーションリポジトリ
- 有効な移行計画

## 手順

1. MTC の Web コンソールにログインし、**Migration plans** をクリックします。

2. 移行計画の横にある Options メニュー  をクリックし、**Migration** で以下のいずれかのオプションを選択します。
  - **ステージ** は、アプリケーションを停止せずにデータをソースクラスターからターゲットクラスターにコピーします。
  - **Cutover** はソースクラスターのトランザクションを停止し、リソースをターゲットクラスターに移動します。  
オプション: **Cutover migration** ダイアログで、**移行時にソースクラスターで Halt トランザクションを消去**できます。
  - **State** は、選択した永続ボリューム要求 (PVC) をコピーします。



### 重要

状態の移行を使用して、クラスター間で namespace を移行しないでください。代わりにステージまたはカットオーバー移行を使用してください。

- **State migration** ダイアログで1つ以上の PVC を選択し、**Migrate** をクリックします。
3. 移行が完了したら、アプリケーションが OpenShift Container Platform Web コンソールで正常に移行されていることを確認します。
    - a. **Home** → **Projects** をクリックします。
    - b. 移行されたプロジェクトをクリックしてそのステータスを表示します。
    - c. **Routes** セクションで **Location** をクリックし、アプリケーションが機能していることを確認します (該当する場合)。
    - d. **Workloads** → **Pods** をクリックし、Pod が移行した namespace で実行されていることを確認します。
    - e. **Storage** → **Persistent volumes** をクリックして、移行した永続ボリュームが正常にプロビジョニングされていることを確認します。

## 第9章 高度な移行オプション

移行を自動化し、**MigPlan** および **MigrationController** カスタムリソースを変更して大規模な移行を実行し、パフォーマンスを向上させることができます。

### 9.1. 用語

表9.1 MTC の用語

用語	定義
ソースクラスター	アプリケーションの移行元となるクラスター。
宛先クラスター <sup>[1]</sup>	アプリケーションが移行されるクラスター。
レプリケーションリポジトリ	ボリュームとイメージの直接的な移行時の Kubernetes オブジェクトに使用するオブジェクトストレージ、または間接的な移行時にイメージ、ボリューム、Kubernetes オブジェクトのコピーに使用するオブジェクトストレージ。  レプリケーションリポジトリはすべてのクラスターからアクセスできる必要があります。
ホストクラスター	<b>migration-controller</b> Pod および Web コンソールが実行されているクラスター。ホストクラスターは通常宛先クラスターですが、これは必須ではありません。  ホストクラスターには、イメージの直接移行にレジストリールートを公開する必要はありません。
リモートクラスター	通常、リモートクラスターはソースクラスターですが、これは必須ではありません。  リモートクラスターには、 <b>migration-controller</b> サービスアカウントトークンが含まれる <b>Secret</b> カスタムリソースが必要です。  リモートクラスターには、直接のイメージ移行用にセキュアなレジストリールートを公開する必要があります。
間接的な移行	イメージ、ボリューム、および Kubernetes オブジェクトはソースクラスターからレプリケーションリポジトリにコピーされ、その後レプリケーションリポジトリから宛先クラスターにコピーされます。
ボリュームの直接移行	永続ボリュームはソースクラスターから宛先クラスターに直接コピーされます。
イメージの直接移行	イメージはソースクラスターから宛先クラスターに直接コピーされます。
段階移行	データはアプリケーションを停止せずに、宛先クラスターにコピーされます。  段階移行を複数回実行すると、カットオーバー移行の時間が短縮されます。

用語	定義
カットオーバー移行	ソースクラスター上のアプリケーションが停止され、アプリケーションリソースが宛先クラスターに移行されます。
状態の移行	アプリケーションの状態は、特定の永続ボリューム要求を宛先クラスターにコピーして移行されます。
ロールバック移行	ロールバック移行は完了した移行をロールバックします。

<sup>1</sup> MTC の Web コンソールの **ターゲットクラスター** を指します。

## 9.2. コマンドラインを使用したアプリケーションの移行

移行を自動化するために、コマンドラインインターフェイス (CLI) を使用して MTC API でアプリケーションを移行できます。

### 9.2.1. 移行の前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインする必要があります。

#### イメージの直接移行

- ソースクラスターのセキュアな OpenShift イメージレジストリーが公開されていることを確認する必要があります。
- 公開されるレジストリーへのルートを作成しておく必要があります。

#### ボリュームの直接移行

- クラスターがプロキシを使用する場合に Stunnel TCP プロキシを設定している必要があります。

#### クラスター

- ソースクラスターは、最新の z-stream リリースにアップグレードされる必要があります。
- MTC のバージョンは、すべてのクラスターで同一である必要があります。

#### ネットワーク

- クラスターには、レプリケーションリポジトリーに対して、また各クラスター間で無制限のネットワークアクセスが必要です。
- **move** を使用して永続ボリュームをコピーする場合、クラスターにはリモートボリュームへの無制限のネットワークアクセスが必要です。
- OpenShift Container Platform 4 クラスターで以下のポートを有効にする必要があります。
  - **6443** (API サーバー)



- **443** (ルート)
- **53** (DNS)
- TLS を使用している場合は、レプリケーションリポジトリでポート **443** を有効にする必要があります。

### 永続ボリューム (PV)

- PV は有効である必要があります。
- PV は永続ボリューム要求にバインドされる必要があります。
- スナップショットを使用して PV をコピーする場合には、以下の前提条件が追加されます。
  - クラウドプロバイダーはスナップショットをサポートしている必要があります。
  - PV に同じクラウドプロバイダーがなければなりません。
  - PV は同じ地理的リージョンにある必要があります。
  - PV には同じストレージクラスがなければなりません。

## 9.2.2. イメージの直接移行用のレジストリルートの作成

イメージを直接移行するには、すべてのリモートクラスターで公開されている OpenShift イメージレジストリーへのルートを作成する必要があります。

### 前提条件

- OpenShift イメージレジストリーは、すべてのリモートクラスター上の外部トラフィックに公開する必要があります。  
デフォルトで OpenShift Container Platform 4 レジストリーを公開しておく。

### 手順

- OpenShift Container Platform 4 レジストリーへのルートを作成するには、以下のコマンドを実行します。

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

## 9.2.3. プロキシ設定

OpenShift Container Platform 4.1 以前のバージョンでは、これらのバージョンはクラスター全体の **proxy** オブジェクトをサポートしないため、Migration Toolkit for Containers Operator のインストール後に、**MigrationController** カスタムリソース (CR) マニフェストでプロキシを設定する必要があります。

OpenShift Container Platform 4.2 から 4.10 の場合、MTC (Migration Toolkit for Containers) はクラスター全体のプロキシ設定を継承します。クラスター全体のプロキシ設定を上書きする場合は、プロキシパラメーターを変更できます。

### 9.2.3.1. ボリュームの直接移行

MTC 1.4.2 で、ボリュームの直接移行 (DVM) が導入されました。DVM は1つのプロキシのみをサポートします。ターゲットクラスターもプロキシの背後にある場合、ソースクラスターはターゲットクラスターのルートにアクセスできません。

プロキシの背後にあるソースクラスターから DVM を実行する場合には、トランスポート層で機能する TCP プロキシを設定して、SSL 接続を独自の SSL 証明書で復号化および再暗号化せずに透過的に転送する必要があります。Stunnel プロキシは、このようなプロキシの例です。

### 9.2.3.1.1. DVM の TCP プロキシ設定

TCP プロキシ経由でソースとターゲットクラスターの間直接接続を設定し、プロキシを使用できるように **MigrationController** CR の `stunnel_tcp_proxy` 変数を設定できます。

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

ボリュームの直接移行 (DVM) は、プロキシの Basic 認証のみをサポートします。さらに、DVM は、TCP 接続を透過的にトンネルできるプロキシの背後でのみ機能します。中間者モードの HTTP/HTTPS プロキシは機能しません。既存のクラスター全体にわたるプロキシはこの動作をサポートしない可能性があります。その結果、DVM のプロキシ設定は、MTC の通常のプロキシ設定とは異なる状態に保たれます。

### 9.2.3.1.2. HTTP/HTTPS プロキシの代わりに TCP プロキシを使用する理由

DVM を有効にするには、OpenShift ルートを介してソースおよびターゲットクラスター間で Rsync を実行します。トラフィックは、TCP プロキシである Stunnel を使用して暗号化されます。ソースクラスターで実行している Stunnel は、ターゲット Stunnel との TLS 接続を開始し、暗号化されたチャンネルでデータを転送します。

OpenShift のクラスター全体の HTTP/HTTPS プロキシは通常、外部サーバーで独自の TLS セッションをネゴシエートする中間者モードで設定されます。ただし、これは Stunnel では機能しません。Stunnel では、プロキシによって TLS セッションが変更されないようにする必要があります。基本的には、プロキシを透過的なトンネルにし、単純に TCP 接続をそのまま転送する必要があります。したがって、TCP プロキシを使用する必要があります。

### 9.2.3.1.3. 既知の問題

#### 移行が **Upgrade request required** エラーで失敗する

移行コントローラーは SPDY プロトコルを使用してリモート Pod 内でコマンドを実行します。リモートクラスターがプロキシまたは、SPDY プロトコルをサポートしないファイアウォールの背後にある場合には、移行コントローラーはリモートコマンドの実行に失敗します。移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: SPDY プロトコルをサポートするプロキシを使用します。

SPDY プロトコルのサポートに加えて、このプロキシまたはファイアウォールでは、**Upgrade** HTTP ヘッダーを API サーバーに渡す必要もあります。クライアントはこのヘッダーを使用して API サーバーと WebSocket 接続を開きます。**Upgrade** ヘッダーがプロキシまたはファイアウォールでブロックされると、移行に失敗し、**Upgrade request required** というエラーメッセージが表示されます。回避策: プロキシで **Upgrade** ヘッダーが転送されるようにしてください。

### 9.2.3.2. 移行用のネットワークポリシーのチューニング

OpenShift は、クラスターで使用されるネットワークプラグインに基づいて **NetworkPolicy** または **EgressFirewalls** を使用した Pod との間のトラフィックの制限をサポートします。移行に関連するソース namespace のいずれかがこのようなメカニズムを使用して Pod へのネットワークトラフィックを制限する場合には、この制限により移行時に Rsync Pod へのトラフィックが誤って停止される可能性があります。

ソースおよびターゲットクラスターの両方で実行される Rsync Pod は OpenShift Route 経由で相互に接続する必要があります。既存の **NetworkPolicy** または **EgressNetworkPolicy** オブジェクトは、これらのトラフィックの制限が課されないように Rsync Pod を自動的に取り除くように設定できます。

#### 9.2.3.2.1. NetworkPolicy の設定

##### 9.2.3.2.1.1. Rsync Pod からの egress トラフィック

Rsync Pod の一意のラベルを使用し、同期元または同期先 namespace の **NetworkPolicy** 設定がこのタイプのトラフィックをブロックする場合に egress トラフィックがそれらを通過することを許可できます。以下のポリシーは、namespace の Rsync Pod からの全 egress トラフィックを許可します。

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

##### 9.2.3.2.1.2. Rsync Pod への ingress トラフィック

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

#### 9.2.3.2.2. EgressNetworkPolicy 設定

**EgressNetworkPolicy** オブジェクトまたは **Egress ファイアウォール** は、egress トラフィックをクラスターからブロックするために設計された OpenShift コンストラクトです。

**NetworkPolicy** オブジェクトとは異なり、Egress ファイアウォールは namespace のすべての Pod に適用されるためにプロジェクトレベルで機能します。そのため、Rsync Pod の一意のラベルを使用すると、この制限から除外するのは Rsync Pod だけではありません。ただし、ソースおよびターゲットクラスターの CIDR 範囲をポリシーの **Allow** ルールに追加して、2つのクラスター間で直接接続を設定できます。

Egress ファイアウォールが存在するクラスターに基づいて、他のクラスターの CIDR 範囲を追加して、2つの間の egress トラフィックを許可できます。

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

#### 9.2.3.2.3. データ転送用の代替エンドポイントの選択

デフォルトでは、DVM は OpenShift Container Platform ルートをエンドポイントとして使用して、PV データを宛先クラスターに転送します。クラスターポロジで許可されている場合は、サポートされている別の種類のエンドポイントを選択できます。

クラスターごとに、**MigrationController** CR で適切な **宛先** クラスターに **rsync\_endpoint\_type** 変数を設定することで、エンドポイントを設定できます。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  [...]
  rsync_endpoint_type: [NodePort|ClusterIP|Route]
```

#### 9.2.3.2.4. RsyncPod の補足グループの設定

PVC が共有ストレージを使用する場合、Pod がアクセスを許可するように RsyncPod 定義に補足グループを追加して、そのストレージへのアクセスを設定できます。

表9.2 RsyncPod の補足グループ

変数	タイプ	デフォルト	説明
<b>src_supplemental_groups</b>	string	設定されていません	ソース Rsync Pod の補足グループのコンマ区切りリスト

変数	タイプ	デフォルト	説明
<code>target_supplemental_groups</code>	string	設定されていません	ターゲット Rsync Pod の補足グループのコンマ区切りリスト

## 使用例

**MigrationController** CR を更新して、これらの補足グループの値を設定できます。

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

### 9.2.3.3. プロキシの設定

#### 前提条件

- **cluster-admin** 権限を持つユーザーとしてすべてのクラスターにログインする必要があります。

#### 手順

1. **MigrationController** CR マニフェストを取得します。

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

2. プロキシパラメーターを更新します。

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

- ❶ ボリュームの直接移行のための Stunnel プロキシ URL。
- ❷ プロキシを除外するための宛先ドメイン名、ドメイン、IP アドレス、または他のネットワーク CIDR のコンマ区切りの一覧。

サブドメインのみと一致するように、ドメインの前に `.` を付けます。たとえば、`.y.com` は `x.y.com` に一致しますが、`y.com` には一致しません。\* を使用し、すべての宛先のプロキシをバイパスします。インストール設定で `networking.machineNetwork[].cidr` フィールドで定義されるネットワークに含まれていないワーカーをスケールアップする場合、それらをこの一覧に追加し、接続の問題を防ぐ必要があります。

**httpProxy** または **httpsProxy** フィールドのいずれも設定されていない場合、このフィールドは無視されます。

3. マニフェストを **migration-controller.yaml** として保存します。
4. 更新したマニフェストを適用します。

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

### 9.2.4. MTC API を使用したアプリケーションの移行

MTC (Migration Toolkit for Containers) API を使用してコマンドラインからアプリケーションを移行できます。

#### 手順

1. host クラスターの **MigCluster** CR マニフェストを作成します。

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

2. リモートクラスターごとに **Secret** オブジェクトマニフェストを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: <cluster_secret>
  namespace: openshift-config
type: Opaque
data:
  saToken: <sa_token> ❶
EOF
```

- ❶ リモートクラスターの base64 でエンコードされた **migration-controller** サービスアカウント (SA) トークンを指定します。以下のコマンドを実行してトークンを取得できます。

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

3. それぞれのリモートクラスターについて **MigCluster** CR マニフェストを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  name: <remote_cluster> ❶
```

```

namespace: openshift-migration
spec:
  exposedRegistryPath: <exposed_registry_route> ❷
  insecure: false ❸
  isHostCluster: false
  serviceAccountSecretRef:
    name: <remote_cluster_secret> ❹
    namespace: openshift-config
  url: <remote_cluster_url> ❺
EOF

```

- ❶ リモートクラスターの **Cluster** CR を指定します。
- ❷ オプション: イメージの直接移行には、公開されるレジストリールートを指定します。
- ❸ SSL 検証は、**false** の場合に有効になります。CA 証明書は、**true** の場合は必要ではなく、チェックされません。
- ❹ リモートクラスターの **Secret** オブジェクトを指定します。
- ❺ リモートクラスターの URL を指定します。

4. すべてのクラスターが **Ready** 状態にあることを確認します。

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

5. レプリケーションリポジトリの **Secret** オブジェクトマニフェストを作成します。

```

$ 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> ❶
  aws-secret-access-key: <secret_key_base64> ❷
EOF

```

- ❶ キー ID を base64 形式で指定します。
- ❷ シークレットキーを base64 形式で指定します。

AWS 認証情報はデフォルトで base64 でエンコードされます。それぞれのキーを使用して以下のコマンドを実行して、認証情報をエンコードする必要があります。

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

- ❶ キー ID またはシークレットキーを指定します。どちらの値も base64 でエンコードする必要があります。

6. レプリケーションリポジトリの **MigStorage** CR マニフェストを作成します。

```
$ 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 バケット名を指定します。
- ② オブジェクトストレージの **Secrets** CR を指定します。オブジェクトストレージの **Secrets** CR に保存される認証情報が正しいことを確認する必要があります。
- ③ ストレージプロバイダーを指定します。
- ④ オプション: スナップショットを使用してデータをコピーする場合は、オブジェクトストレージの **Secrets** CR を指定します。オブジェクトストレージの **Secrets** CR に保存される認証情報が正しいことを確認する必要があります。
- ⑤ オプション: スナップショットを使用してデータをコピーする場合は、ストレージプロバイダーを指定します。

7. **MigStorage** CR が **Ready** 状態にあることを確認します。

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

8. **MigPlan** CR マニフェストを作成します。

```
$ 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 ①
  indirectVolumeMigration: true ②
```



```

migStorageRef:
  name: <migstorage> ❸
  namespace: openshift-migration
namespaces:
  - <source_namespace_1> ❹
  - <source_namespace_2>
  - <source_namespace_3>:<destination_namespace> ❺
srcMigClusterRef:
  name: <remote_cluster> ❻
  namespace: openshift-migration
EOF

```

- ❶ **false** の場合、直接的なイメージ移行が有効にされます。
- ❷ **false** の場合、直接的なボリューム移行が有効にされます。
- ❸ **MigStorage** CR インスタンスの名前を指定します。
- ❹ namespace を1つ以上指定します。デフォルトで、宛先 namespace の名前は同じです。
- ❺ 宛先 namespace が異なる場合には、宛先 namespace を指定します。
- ❻ ソースクラスター **MigCluster** インスタンスの名前を指定します。

9. **MigPlan** インスタンスが **Ready** 状態にあることを確認します。

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

10. **MigMigration** CR マニフェストを作成し、**MigPlan** インスタンスに定義された移行を開始します。

```

$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  name: <migmigration>
  namespace: openshift-migration
spec:
  migPlanRef:
    name: <migplan> ❶
    namespace: openshift-migration
  quiescePods: true ❷
  stage: false ❸
  rollback: false ❹
EOF

```

- ❶ **MigPlan** CR 名を指定します。
- ❷ **true** の場合、ソースクラスターの Pod は停止します。
- ❸ **true** の場合、アプリケーションを停止せずにほとんどのデータをコピーする段階移行が実行されます。
- ❹ **true** の場合、完了した移行がロールバックされます。

11. **MigMigration** CR の進捗を監視して移行を確認します。

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

出力は以下のようになります。

**出力例**

```
Name:      c8b034c0-6567-11eb-9a4f-0bc004db0fbc
Namespace: openshift-migration
Labels:    migration.openshift.io/migplan-name=django
Annotations: openshift.io/touch: e99f9083-6567-11eb-8420-0a580a81020c
API Version: migration.openshift.io/v1alpha1
Kind:      MigMigration
...
Spec:
  Mig Plan Ref:
    Name:      migplan
    Namespace: openshift-migration
  Stage:      false
Status:
  Conditions:
    Category:      Advisory
    Last Transition Time: 2021-02-02T15:04:09Z
    Message:        Step: 19/47
    Reason:         InitialBackupCreated
    Status:         True
    Type:           Running
    Category:       Required
    Last Transition Time: 2021-02-02T15:03:19Z
    Message:        The migration is ready.
    Status:         True
    Type:           Ready
    Category:       Required
    Durable:        true
    Last Transition Time: 2021-02-02T15:04:05Z
    Message:        The migration registries are healthy.
    Status:         True
    Type:           RegistriesHealthy
  Itinerary:      Final
  Observed Digest:
7fae9d21f15979c71ddc7dd075cb97061895caac5b936d92fae967019ab616d5
  Phase:          InitialBackupCreated
  Pipeline:
    Completed: 2021-02-02T15:04:07Z
    Message:    Completed
    Name:       Prepare
    Started:    2021-02-02T15:03:18Z
    Message:    Waiting for initial Velero backup to complete.
    Name:       Backup
    Phase:      InitialBackupCreated
    Progress:
      Backup openshift-migration/c8b034c0-6567-11eb-9a4f-0bc004db0fbc-wpc44: 0 out of
estimated total of 0 objects backed up (5s)
    Started:    2021-02-02T15:04:07Z
    Message:    Not started
```

```

Name:      StageBackup
Message:   Not started
Name:      StageRestore
Message:   Not started
Name:      DirectImage
Message:   Not started
Name:      DirectVolume
Message:   Not started
Name:      Restore
Message:   Not started
Name:      Cleanup
Start Timestamp: 2021-02-02T15:03:18Z
Events:
Type Reason Age From Message
---- -
Normal Running 57s migmigration_controller Step: 2/47
Normal Running 57s migmigration_controller Step: 3/47
Normal Running 57s (x3 over 57s) migmigration_controller Step: 4/47
Normal Running 54s migmigration_controller Step: 5/47
Normal Running 54s migmigration_controller Step: 6/47
Normal Running 52s (x2 over 53s) migmigration_controller Step: 7/47
Normal Running 51s (x2 over 51s) migmigration_controller Step: 8/47
Normal Ready 50s (x12 over 57s) migmigration_controller The migration is ready.
Normal Running 50s migmigration_controller Step: 9/47
Normal Running 50s migmigration_controller Step: 10/47

```

### 9.2.5. 状態の移行

アプリケーションの状態を設定する Persistent Volume Claim(永続ボリューム要求、PVC) を移行するために MTC (Migration Toolkit for Containers) を使用して反復可能な状態のみの移行を実行できます。移行計画から他の PVC を除外して、指定された PVC を移行します。PVC をマップし、ソースおよびターゲット PVC が同期されるようにできます。永続ボリューム (PV) データがターゲットクラスターにコピーされます。PV 参照は移動されず、アプリケーション Pod はソースクラスターでの実行を継続します。

状態の移行は、OpenShift GitOps などの外部 CD メカニズムと併用されるように特別に設計されています。MTC を使用して状態を移行する間に GitOps を使用してアプリケーションマニフェストを移行できます。

CI/CD パイプラインがある場合には、それらをターゲットクラスターにデプロイすることでステートレスコンポーネントを移行できます。次に、MTC を使用してステートフルコンポーネントを移行できます。

クラスター間または同じクラスター間で状態の移行を実行できます。



#### 重要

状態の移行は、アプリケーションの状態を設定するコンポーネントのみを移行します。namespace 全体を移行する場合は、ステージまたはカットオーバー移行を使用します。

#### 前提条件

- ソースクラスターのアプリケーションの状態は、**PersistentVolumeClaims** でプロビジョニングされた **PersistentVolume** で永続化されます。

- アプリケーションのマニフェストは、ソースクラスターとターゲットクラスターの両方からアクセスできる中央リポジトリで利用できます。

## 手順

1. 永続ボリュームデータをソースからターゲットクラスターに移行します。  
この手順は、必要に応じて何度でも実行することができます。ソースアプリケーションは実行を継続します。
2. ソースアプリケーションを休止します。  
これは、ワークロードリソースのレプリカを、ソースクラスターに直接設定するか、GitHub でマニフェストを更新して Argo CD アプリケーションを再同期することで、0 に設定できます。
3. アプリケーションマニフェストのクローンをターゲットクラスターに作成します。  
Argo CD を使用して、アプリケーションマニフェストのクローンをターゲットクラスターに作成できます。
4. 残りのボリュームデータをソースからターゲットクラスターに移行します。  
最終的なデータ移行を実行して、状態移行プロセス中にアプリケーションによって作成された新しいデータを移行します。
5. クローンを作成したアプリケーションが休止状態の場合は、停止を解除します。
6. DNS レコードをターゲットクラスターに切り替えて、ユーザートラフィックを移行されたアプリケーションにリダイレクトします。



## 注記

MTC 1.6 は、状態移行の実行時にアプリケーションを自動的に停止できません。PV データのみ移行できます。したがって、アプリケーションの停止や停止解除に CD メカニズムを使用する必要があります。

MTC 1.7 では、明示的なステージおよびカットオーバーフローが導入されました。ステージングを使用して、必要なだけデータ転送を行うことができます。その後、カットオーバーを実行すると、ソースアプリケーションが自動的に停止します。

## 関連情報

- 状態移行用の PVC を選択するには、[移行からの PVC の除外](#) を参照してください。
- ソース PV データを宛先クラスターのプロビジョニングされた PVC に移行する [PVC のマッピング](#) を参照してください。
- アプリケーションの状態を設定する Kubernetes オブジェクトを移行するには、[Kubernetes オブジェクトの移行](#) を参照してください。

## 9.3. 移行フック

単一の移行計画に最大 4 つの移行フックを追加し、各フックを移行の異なるフェーズで実行できます。移行フックは、アプリケーションの休止状態のカスタマイズ、サポート外のデータタイプの手動の移行、および移行後のアプリケーションの更新などのタスクを実行します。

移行フックは、以下の移行手順のいずれかでソースまたはターゲットクラスターで実行されます。

- **PreBackup:** リソースがソースクラスターでバックアップされる前

- **PostBackup**: リソースがソースクラスターでバックアップされた後
- **PreRestore**: リソースがターゲットクラスターで復元される前
- **PostRestore**: リソースがターゲットクラスターで復元された後

フックを作成するには、デフォルトの Ansible イメージまたはカスタムフックコンテナで実行される Ansible Playbook を作成します。

## Ansible Playbook

Ansible Playbook はフックコンテナに設定マップとしてマウントされます。フックコンテナは、**MigPlan** カスタムリソースに指定されるクラスター、サービスアカウント、および namespace を使用してジョブとして実行されます。ジョブは、デフォルトの再試行数 6 に達するか、正常に完了するまで実行を継続します。これは、最初の Pod がエビクトされるか、強制終了される場合でも継続されます。

デフォルトの Ansible ランタイムイメージは **registry.redhat.io/rhmtc/openshift-migration-hook-runner-rhel7:1.7** です。このイメージは Ansible Runner イメージをベースとしており、Ansible Kubernetes リソースの **python-openshift** および更新された **oc** バイナリーが含まれます。

## カスタムフックコンテナ

デフォルトの Ansible イメージの代わりにカスタムフックコンテナを使用できます。

### 9.3.1. 移行フックの Ansible Playbook の作成

Ansible Playbook を作成して移行フックとして使用することができます。フックは、MTC Web コンソールを使用するか、**MigPlan** カスタムリソース (CR) マニフェストに **spec.hooks** パラメーターの値を指定して移行計画に追加できます。

Ansible Playbook はフックコンテナに設定マップとしてマウントされます。フックコンテナは、**MigPlan** で指定されるクラスター、サービスアカウントおよび namespace を使用してジョブとして実行されます。フックコンテナは指定されたサービスアカウントトークンを使用して、タスクがクラスターで実行される前に認証を必要としないようにします。

#### 9.3.1.1. Ansible モジュール

Ansible **shell** モジュールを使用して **oc** コマンドを実行できます。

##### shell モジュールの例

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

**k8s\_info** などの **kubernetes.core** モジュールを使用して Kubernetes リソースと対話できます。

##### k8s\_facts モジュールの例

```
- 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 }}"

```

**fail** モジュールを使用して、ゼロ以外の終了ステータスが正常に生成されない場合にゼロ以外の終了ステータスを生成し、フックの成功または失敗が検出されるようにします。フックはジョブとして実行され、フックの成功または失敗のステータスはジョブコンテナの終了ステータスに基づいて表示されません。

### fail モジュールの例

```

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

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

```

#### 9.3.1.2. 環境変数

**MigPlan** CR 名および移行 namespace は環境変数としてフックコンテナに渡されます。これらの変数は **lookup** プラグインを使用してアクセスされます。

### 環境変数の例

```

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

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

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

```

## 9.4. 移行計画のオプション

**MigPlan** カスタムリソース (CR) のコンポーネントを除外、編集、およびマップできます。

### 9.4.1. リソースの除外

移行についてのリソース負荷を減らしたり、別のツールでイメージや PV を移行するために、MTC (Migration Toolkit for Containers) からイメージストリーム、永続ボリューム (PV)、またはサブスクリプションなどのリソースを除外することができます。

デフォルトで、MTC は移行からサービスカタログリソースおよび Operator Lifecycle Manager (OLM) リソースを除外します。これらのリソースはサービスカタログ API グループおよび OLM API グループの一部ですが、現時点でこれらは移行についてサポートされません。

#### 手順

1. **MigrationController** カスタムリソースマニフェストを編集します。

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

2. 特定のリソースを除外するパラメーターを追加して、**spec** セクションを更新します。独自の除外パラメーターを持たないリソースの場合は、**additional\_excluded\_resources** パラメーターを追加します。

```
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
  additional_excluded_resources: ③
  - resource1
  - resource2
  ...
```

- ① **disable\_image\_migration: true** を追加して、移行からイメージストリームを除外します。**MigrationController** Pod が再起動すると、**imagestreams** が **main.yml** の **excluded\_resources** リストに追加されます。
- ② **disable\_pv\_migration: true** を追加して、移行計画から PV を除外します。**MigrationController** Pod が再起動すると、**persistentvolumes** と **persistentvolumeclaims** が **main.yml** の **excluded\_resources** リストに追加されます。PV 移行を無効にすると、移行計画の作成時に PV 検出も無効にできます。
- ③ 除外する OpenShift Container Platform リソースを **additional\_excluded\_resources** リストに追加できます。

3. **MigrationController** Pod が再起動し、変更が適用されるまで 2 分待機します。
4. リソースが除外されていることを確認します。

```
$ oc get deployment -n openshift-migration migration-controller -o yaml | grep EXCLUDED_RESOURCES -A1
```

出力には、除外されたリソースが含まれます。

## 出力例

```
name: EXCLUDED_RESOURCES
value:
resource1,resource2,imagetags,templateinstances,clusterserviceversions,packagemanifests,sul
scriptions,servicebrokers,servicebindings,serviceclasses,serviceinstances,serviceplans,imaget
ams,persistentvolumes,persistentvolumeclaims
```

### 9.4.2. 名前空間のマッピング

**MigPlan** カスタムリソース (CR) で namespace をマッピングした場合には、namespace の UID および GID の範囲が移行時にコピーされるため、namespace が移行元または移行先ホストで複製されないようにする必要があります。

#### 同じ宛先 namespace にマッピングされた 2 つのソース namespace

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

ソース namespace を同じ名前の namespace にマップする場合には、マッピングを作成する必要はありません。デフォルトでは、ソースの namespace とターゲット namespace の名前は同じです。

#### 誤った namespace マッピング

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

#### 正しい namespace リファレンス

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

### 9.4.3. Persistent Volume Claim (永続ボリューム要求、PVC) の除外

移行しない PVC を除外して、状態移行用に永続ボリューム要求 (PVC) を選択します。永続ボリューム (PV) の検出後に **MigPlan** カスタムリソース (CR) の **spec.persistentVolumes.pvc.selection.action** パラメーターを設定して PVC を除外します。

#### 前提条件

- **MigPlan** CR が **Ready** 状態にある。

#### 手順

- **spec.persistentVolumes.pvc.selection.action** パラメーターを **MigPlan** CR に追加し、それを **skip** に設定します。

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



```

kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  ...
  persistentVolumes:
  - capacity: 10Gi
    name: <pv_name>
    pvc:
  ...
  selection:
    action: skip

```

#### 9.4.4. 永続ボリューム要求 (PVC) のマッピング

永続ボリューム (PV) データをソースクラスターから、PVC をマッピングすることで、**MigPlan** CR の宛先クラスターですでにプロビジョニングされている永続ボリューム要求 (PVC) に移行できます。このマッピングにより、移行したアプリケーションの宛先 PVC がソース PVC と同期されます。

PV の検出後に **MigPlan** カスタムリソース (CR) の **spec.persistentVolumes.pvc.name** パラメーターを更新して PVC をマップします。

##### 前提条件

- **MigPlan** CR が **Ready** 状態にある。

##### 手順

- **MigPlan** CR の **spec.persistentVolumes.pvc.name** パラメーターを更新します。

```

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

```

- ❶ ソースクラスターに PVC を指定し、宛先クラスターで PVC を指定します。宛先 PVC が存在しない場合、これは作成されます。このマッピングを使用して、移行時に PVC 名を変更できます。

#### 9.4.5. 永続ボリューム属性の編集

**MigPlan** カスタムリソース (CR) を作成した後、**MigrationController** CR は永続ボリューム (PV) を検出します。**spec.persistentVolumes** ブロックと **status.destStorageClasses** ブロックが **MigPlan** CR に追加されます。

**spec.persistentVolumes.selection** ブロックの値を編集できます。 **spec.persistentVolumes.selection** ブロックの外部で値を変更すると、 **MigPlan** CR が **MigrationController** CR によって調整されるときに値が上書きされます。



## 注記

**spec.persistentVolumes.selection.storageClass** パラメーターのデフォルト値は、次のロジックによって決定します。

1. ソースクラスター PV が Gluster または NFS の場合のデフォルトは、 **accessMode: ReadWriteMany** の場合は **cephfs**、 **accessMode: ReadWriteOnce** の場合は **cephrbd** です。
2. PV が Gluster でも NFS でもない場合、もしくは、 **cephfs** または **cephrbd** が使用できない場合、デフォルトは同じプロビジョナーのストレージクラスです。
3. 同じプロビジョナーのストレージクラスが使用できない場合、デフォルトは宛先クラスターのデフォルトのストレージクラスです。

**storageClass** 値を、 **MigPlan** CR の **status.destStorageClasses** ブロック内の任意の **name** パラメーターの値に変更できます。

**storageClass** 値が空の場合、移行後、PV にはストレージクラスがありません。このオプションは、たとえば、PV を宛先クラスターの NFS ボリュームに移動する場合に適しています。

## 前提条件

- **MigPlan** CR が **Ready** 状態にある。

## 手順

- **MigPlan** CR で **spec.persistentVolumes.selection** 値を編集します。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  persistentVolumes:
    - capacity: 10Gi
      name: pvc-095a6559-b27f-11eb-b27f-021bddcaf6e4
      proposedCapacity: 10Gi
      pvc:
        accessModes:
          - ReadWriteMany
        hasReference: true
        name: mysql
        namespace: mysql-persistent
      selection:
        action: <copy> 1
        copyMethod: <filesystem> 2
        verify: true 3
  
```

```
storageClass: <gp2> 4
accessMode: <ReadWriteMany> 5
storageClass: cephfs
```

- 1 許可される値は、**move**、**copy**、および **skip** です。サポートされているアクションが1つだけの場合、デフォルト値はサポートされているアクションです。複数のアクションがサポートされている場合、デフォルト値は **copy** です。
- 2 許可される値は、**snapshot** および **filesystem** です。デフォルト値は **filesystem** です。
- 3 MTC Web コンソールでファイルシステムコピーの検証オプションを選択すると、**verify** パラメーターが表示されます。**false** に設定できます。
- 4 デフォルト値を、**MigPlan** CR の **status.destStorageClasses** ブロック内の任意の **name** パラメーターの値に変更できます。値が指定されていない場合、PV は移行後にストレージクラスを持ちません。
- 5 使用できる値は **ReadWriteOnce** および **ReadWriteMany** です。この値が指定されていない場合、デフォルトはソースクラスター PVC のアクセスモードです。アクセスモードは、**MigPlan** でのみ編集できます。MTC Web コンソールを使用して編集することはできません。

#### 9.4.6. MTC の Web コンソールでのストレージクラスの変換

永続ボリューム (PV) のストレージクラスは、同じクラスター内で移行することで変換できます。これを実行するには、MTC (Migration Toolkit for Containers) Web コンソールで移行計画を作成し、実行する必要があります。

##### 前提条件

- MTC が実行されているクラスターで **cluster-admin** 権限を持つユーザーとしてログインしている必要があります。
- クラスターを MTC Web コンソールに追加する必要があります。

##### 手順

1. OpenShift Container Platform Web コンソールの左側のナビゲーションペインで、**Projects** をクリックします。
2. プロジェクト一覧で、プロジェクトをクリックします。  
**Project details** ページが開きます。
3. **DeploymentConfig** 名をクリックします。実行中の Pod の名前をメモします。
4. プロジェクトの YAML タブを開きます。PV を検索し、対応する 永続ボリュームクレーム (PVC) の名前をメモします。
5. MTC Web コンソールで、**Migration plans** をクリックします。
6. **Add migration plan** をクリックします。
7. **Plan name** を入力します。  
移行計画名には、3 - 63 の小文字の英数字 (**a-z, 0-9**) を使用できず、スペースやアンダースコア (**\_**) を含むことはできません。

8. **Migration type** メニューから **Storage class conversion** を選択します。
9. **Source cluster** リストから、ストレージクラスの変換に必要なクラスターを選択します。
10. **Next** をクリックします。  
**Namespaces** ページが開きます。
11. 必要なプロジェクトを選択します。
12. **Next** をクリックします。  
**Persistent volumes** ページが開きます。このページには、デフォルトで選択されている PV がすべて表示されます。
13. それぞれの PV について、必要なターゲットストレージクラスを選択します。
14. **Next** をクリックします。  
ウィザードは新しい移行プランを検証し、準備状態にあることを示します。
15. **Close** をクリックします。  
新しい計画が **Migration plans** ページに表示されます。
16. 変換を開始するには、新しい計画のオプションメニューをクリックします。  
**Migrations** の下に、**Stage** と **Cutover** の 2 つのオプションが表示されます。



#### 注記

カットオーバー移行は、アプリケーションの PVC 参照を更新します。

ステージ移行は、アプリケーションの PVC 参照を更新しません。

17. オプションを選択します。  
選択したオプションに応じて、**Stage migration** または **Cutover migration** の通知が表示されます。
18. **Migrate** をクリックします。  
選択したオプションに応じて、**Stage started** または **Cutover started** のメッセージが表示されます。
19. 現在の移行ステータスを表示するには、**Migrations** 列の数字をクリックします。  
**Migrations** ページが開きます。
20. 現在の移行の詳細を表示し、その進捗をモニタリングするには、**Type** 列から移行を選択します。  
**Migration details** ページが開きます。移行が **DirectVolume** の手順に進み、その手順のステータスが **Running Rsync Pods to migrate Persistent Volume data** になると、**View details** をクリックしてコピーの詳細なステータスを確認できます。
21. ブレッドクラムバーで **Stage** または **Cutover** をクリックし、すべての手順が完了するまで待ちます。
22. OpenShift Container Platform Web コンソールの **PersistentVolumeClaims** タブを開きます。  
新しい PVC は初期 PVC の名前が表示できますが、**new** で終わります。これは、ターゲットストレージクラスを使用しています。
23. 左側のナビゲーションペインで、**Pods** をクリックします。プロジェクトの Pod が再び実行されていることを確認します。

## 関連情報

- **move** および **copy** アクションの詳細は、[MTC ワークフロー](#) を参照してください。
- **skip** アクションの詳細については、[移行からの PVC の除外](#) を参照してください。
- ファイルシステムとスナップショットのコピー方法の詳細については、[データのコピー方法について](#) を参照してください。

### 9.4.7. MTCAPI を使用した Kubernetes オブジェクトの状態移行の実行

すべての PV データを移行した後に、Migration Toolkit for Containers (MTC) API を使用して、アプリケーションを設定する Kubernetes オブジェクトの状態を移行を1回限りで実行できます。

これを行うには、**MigPlan** カスタムリソース (CR) フィールドを設定して、Kubernetes リソースのリストに追加のラベルセクターを提供し、それらのリソースをさらにフィルタリングしてから、**MigMigrationCR** を作成して移行します。**MigPlan** リソースは、移行後に終了します。



#### 注記

Kubernetes リソースの選択は API 限定の機能です。CLI を使用して、**MigPlan** CR を更新し、その **MigMigrationCR** を作成する必要があります。MTC Web コンソールは、Kubernetes オブジェクトの移行をサポートしていません。



#### 注記

移行後に、**MigPlan** CR の **closed** パラメーターは **true** に設定されます。この **MigPlan** CR の別の **MigMigration** CR を作成することはできません。

以下のいずれかのオプションを使用して、Kubernetes オブジェクトを **MigPlan** CR に追加します。

- Kubernetes オブジェクトを **includedResources** セクションに追加します。**MigPlan** CR で **includedResources** フィールドが指定されている場合に、プランは **グループの種類** のリストを入力として受け取ります。リストに存在するリソースのみが移行に含まれます。
- オプションの **labelSelector** パラメーターを追加して、**MigPlan** の **includedResources** をフィルター処理します。このフィールドを指定すると、ラベルセクターに一致するリソースのみが移行に含まれます。たとえば、ラベル **app:frontend** をフィルターとして使用して、**Secret** リソースと **ConfigMap** リソースのリストをフィルタリングできます。

## 手順

1. **MigPlan** CR を更新して、Kubernetes リソースを含め、オプションで、**labelSelector** パラメーターを追加して含まれているリソースをフィルタリングします。
  - a. **MigPlan** CR を更新して Kubernetes リソースを含めるには以下を実行します。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  includedResources:
    - kind: <kind> ①
```

```
group: ""
- kind: <kind>
group: ""
```

- 1 Kubernetes オブジェクトを指定します (例: **Secret** または **ConfigMap**)。

- b. オプション: **labelSelector** パラメーターを追加して、含まれているリソースをフィルター処理するには、次のようにします。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  includedResources:
    - kind: <kind> 1
      group: ""
    - kind: <kind>
      group: ""
  ...
  labelSelector:
    matchLabels:
      <label> 2
```

- 1 Kubernetes オブジェクトを指定します (例: **Secret** または **ConfigMap**)。

- 2 移行するリソースのラベルを指定します (例: **app: frontend**)。

2. **MigMigration** CR を作成して、選択した Kubernetes リソースを移行します。正しい **MigPlan** が **migPlanRef** で参照されていることを確認します。

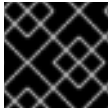
```
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  generateName: <migplan>
  namespace: openshift-migration
spec:
  migPlanRef:
    name: <migplan>
    namespace: openshift-migration
  stage: false
```

## 9.5. 移行コントローラーオプション

移行計画の制限を編集したり、永続ボリュームのサイズ変更を有効にしたり、大規模な移行およびパフォーマンスを向上させる **MigrationController** カスタムリソース (CR) でキャッシュされた Kubernetes クライアントを有効にすることもできます。

### 9.5.1. 大規模な移行についての制限の引き上げ

MTC (Migration Toolkit for Containers) を使用した大規模な移行の場合には、移行オブジェクトおよびコンテナリソースで制限を引き上げることができます。



## 重要

実稼働環境で移行を実行する前に、これらの変更をテストする必要があります。

## 手順

1. **MigrationController** カスタムリソース (CR) マニフェストを編集します。

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

2. 以下のパラメーターを更新します。

```
...
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 **MigrationController** CR で利用可能な CPU の数を指定します。
- ② **MigrationController** CR で利用可能なメモリー量を指定します。
- ③ **MigrationController** CR 要求で利用可能な CPU ユニットの数を指定します。100m は 0.1 CPU ユニット ( $100 * 1e-3$ ) を表します。
- ④ **MigrationController** CR 要求で利用可能なメモリーの量を指定します。
- ⑤ 移行可能な永続ボリュームの数を指定します。
- ⑥ 移行可能な Pod の数を指定します。
- ⑦ 移行可能な namespace の数を指定します。

3. 更新されたパラメーターを使用して変更を検証する移行計画を作成します。  
移行計画が **MigrationController** CR の制限を超える場合、MTC コンソールには移行計画を保存する際に警告メッセージが表示されます。

### 9.5.2. ボリュームの直接移行での永続ボリュームサイズ変更の有効化

ボリュームの直接移行用に永続ボリューム (PV) のサイズ変更を有効にして、宛先クラスターでディスク領域が不足しないようにします。

PV のディスク使用量が設定されたレベルに達すると、**MigrationController** カスタムリソース (CR) は永続ボリューム要求 (PVC) で必要なストレージ容量と実際のプロビジョニング容量とを比較します。次に、この CR は宛先クラスターに必要な領域を計算します。

**pv\_resizing\_threshold** パラメーターは、PV のサイズ変更が使用されるタイミングを決定します。デフォルトのしきい値は **3%** です。つまり、PV のディスク使用量が **97%** を超える場合に PV のサイズ変更が発生します。PV のサイズ変更はディスク使用量が低いレベルで実行されるように、このしきい値を引き上げることができます。

PVC の容量は以下の基準に従って計算されます。

- PVC の要求されるストレージ容量 (**spec.resources.requests.storage**) が実際のプロビジョニングされた容量 (**status.capacity.storage**) と等しくない場合には、より大きい値が使用されます。
- PV が PVC 経由でプロビジョニングされ、その後に変更されて PV および PVC の容量が一致しなくなった場合に、より大きい値が使用されます。

### 前提条件

- PVC は、**MigrationController** CR がコマンドを実行できるように実行中の Pod 1 つ以上に割り当てする必要があります。

### 手順

1. ホストクラスターにログインします。
2. **MigrationController** CR のパッチを適用して PV のサイズ変更を有効にします。

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

- ① PV のサイズ変更を無効にするには、値を **false** に設定します。

3. 必要に応じて、**pv\_resizing\_threshold** パラメーターを更新して、しきい値を増やします。

```
$ oc patch migrationcontroller migration-controller -p '{"spec":{"pv_resizing_threshold":41}}' \
--type='merge' -n openshift-migration
```

- ① デフォルト値は **3** です。

しきい値を超えると、以下のステータスメッセージが **MigPlan** CR ステータスに表示されます。

```
status:
  conditions:
  ...
  - category: Warn
    durable: true
    lastTransitionTime: "2021-06-17T08:57:01Z"
    message: 'Capacity of the following volumes will be automatically adjusted to avoid disk
```



```
capacity issues in the target cluster: [pvc-b800eb7b-cf3b-11eb-a3f7-0eae3e0555f3]
reason: Done
status: "False"
type: PvCapacityAdjustmentRequired
```



### 注記

AWS gp2 ストレージの場合に、gp2 がボリューム使用量とサイズを計算する方法が原因で、**pv\_resizing\_threshold** が 42% 以上でない限り、このメッセージが表示されます。(BZ#1973148)

### 9.5.3. キャッシュされた Kubernetes クライアントの有効化

移行時にパフォーマンスを向上させるために、キャッシュされた Kubernetes クライアントを **MigrationController** カスタムリソース (CR) で有効にできます。パフォーマンスに関する利点は、異なるリージョンのクラスター間で移行する場合や、ネットワークレイテンシーが大きい場合の移行時に発揮されます。



### 注記

委譲されたタスク (例: 直接ボリューム移行または Velero バックアップおよび復元用の Rsync バックアップ) では、キャッシュされたクライアントのパフォーマンスは向上されません。

**MigrationController** CR は **MigCluster** CR との対話に必要なすべての API リソースをキャッシュするため、キャッシュされたクライアントには追加のメモリが必要です。通常 API サーバーに送信される要求は、代わりにキャッシュに転送されます。このキャッシュは API サーバーで更新がないかを監視します。

キャッシュされたクライアントを有効にした後に **OOMKilled** エラーが発生すると、**MigrationController** CR のメモリ制限および要求を増やすことができます。

### 手順

1. 以下のコマンドを実行して、キャッシュされたクライアントを有効化します。

```
$ oc -n openshift-migration patch migrationcontroller migration-controller --type=json --patch \
'[{ "op": "replace", "path": "/spec/mig_controller_enable_cache", "value": true}]'
```

2. オプション: 以下のコマンドを実行して **MigrationController** CR メモリーの制限を増やします。

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

3. オプション: 以下のコマンドを実行して **MigrationController** CR メモリー要求を増やします。

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

## 第10章 トラブルシューティング

このセクションでは、MTC (Migration Toolkit for Containers) のトラブルシューティングに使用するリソースについて説明します。

既知の問題については、[MTC リリースノート](#) を参照してください。

### 10.1. MTC ワークフロー

MTC (Migration Toolkit for Containers) の Web コンソールまたは Kubernetes API を使用して、Kubernetes リソース、永続ボリュームデータ、および内部コンテナイメージを OpenShift Container Platform 4.10 に移行できます。

MTC は以下のリソースを移行します。

- 移行計画に指定される namespace。
- namespace スコープのリソース: MTC が namespace を移行する場合、サービスや Pod などのその namespace に関連付けられるすべてのオブジェクトおよびリソースを移行します。さらに、namespace に存在するものの、クラスターレベルに存在しないリソースがクラスターレベルに存在するリソースに依存する場合、MTC は両方のリソースを移行します。たとえば、SCC (Security Context Constraints) はクラスターレベルに存在するリソースであり、サービスアカウント (SA) は namespace レベルに存在するリソースです。SA が MTC が移行する namespace に存在する場合、MTC は SA にリンクされている SCC を自動的に識別し、それらの SCC も移行します。同様に、MTC は、namespace の永続ボリューム要求にリンクされている永続ボリュームを移行します。



#### 注記

クラスタースコープのリソースは、リソースによっては手動で移行する必要がある場合があります。

- カスタムリソース (CR) およびカスタムリソース定義 (CRD): MTC は、namespace レベルで CR および CRD を自動的に移行します。

MTC Web コンソールを使用してアプリケーションを移行するには、以下の手順が必要です。

1. すべてのクラスターに MTC (Migration Toolkit for Containers Operator) をインストールします。  
インターネットアクセスが制限されているか、インターネットアクセスのない制限された環境で Migration Toolkit for Containers Operator をインストールできます。ソースおよびターゲットクラスターは、相互に対するネットワークアクセスおよびミラーレジストリーへのネットワークアクセスがなければなりません。
2. MTC がデータ移行に使用する中間オブジェクトストレージであるレプリケーションリポジトリを設定します。  
ソースおよびターゲットクラスターには、移行時にレプリケーションリポジトリへのネットワークアクセスがなければなりません。プロキシサーバーを使用している場合は、レプリケーションリポジトリとクラスター間のネットワークトラフィックを許可するように設定する必要があります。
3. ソースクラスターを MTC の Web コンソールに追加します。
4. レプリケーションリポジトリを MTC の Web コンソールに追加します。

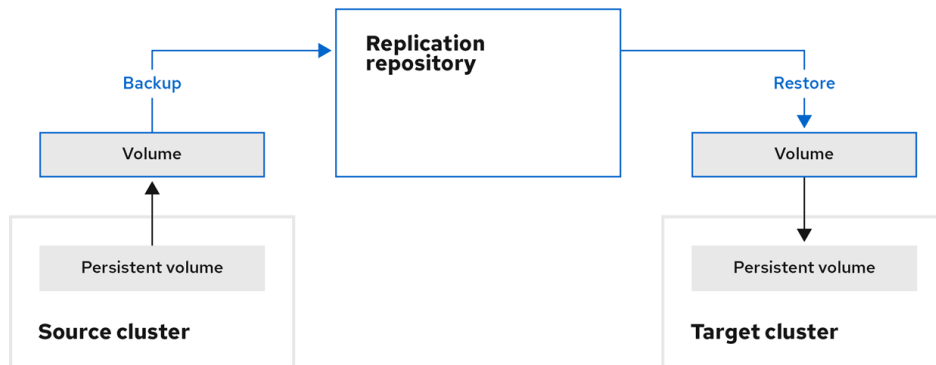
5. 以下のデータ移行オプションのいずれかを使用して、移行計画を作成します。

- **Copy:** MTC は、データをソースクラスターからレプリケーションリポジトリにコピーし、レプリケーションリポジトリからターゲットクラスターにコピーします。



### 注記

イメージの直接移行またはボリュームの直接移行を使用している場合、イメージまたはボリュームはソースクラスターからターゲットクラスターに直接コピーされます。



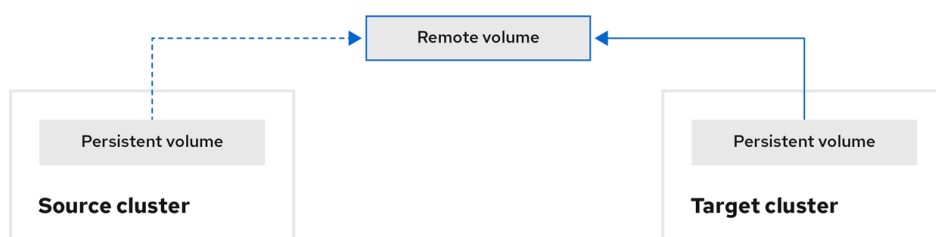
OpenShift\_45\_1019

- **Move:** MTC は、ソースクラスターからリモートボリューム (例: NFS) をアンマウントし、リモートボリュームをポイントするターゲットクラスターで PV リソースを作成し、その後にリモートボリュームをターゲットクラスターにマウントします。ターゲットクラスターで実行されているアプリケーションは、ソースクラスターが使用していたものと同じリモートボリュームを使用します。リモートボリュームは、ソースクラスターおよびターゲットクラスターからアクセスできる必要があります。



### 注記

レプリケーションリポジトリはこの図には表示されていませんが、これは移行に必要です。

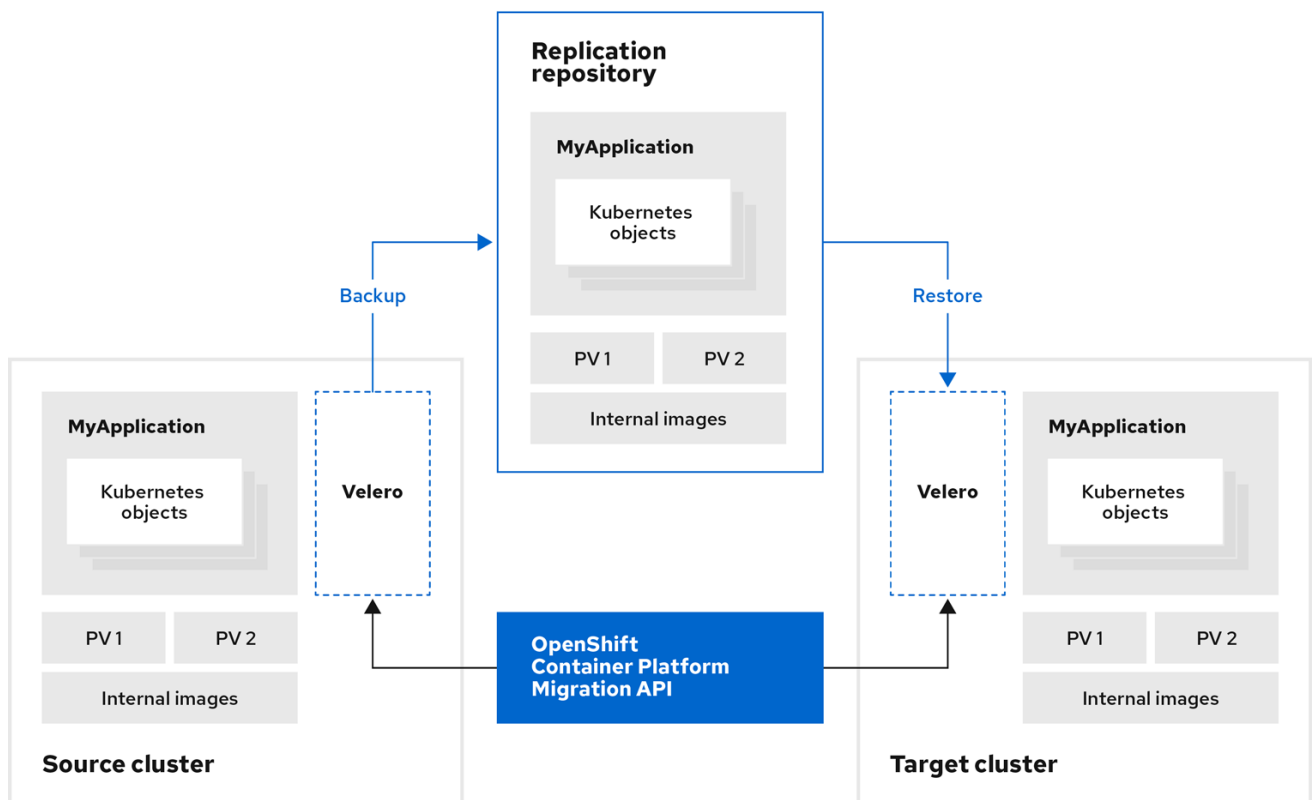


OpenShift\_45\_1019

6. 以下のオプションのいずれかを使用して、移行計画を実行します。

- **段階** 移行は、アプリケーションを停止せずにデータをターゲットクラスターにコピーします。段階移行は複数回実行して、移行前にほとんどのデータがターゲットにコピーされるようにします。1つ以上の段階移行を実行すると、カットオーバー移行の期間が短縮されます。
- **カットオーバー** は、ソースクラスターでアプリケーションを停止し、リソースをターゲットクラスターに移動します。

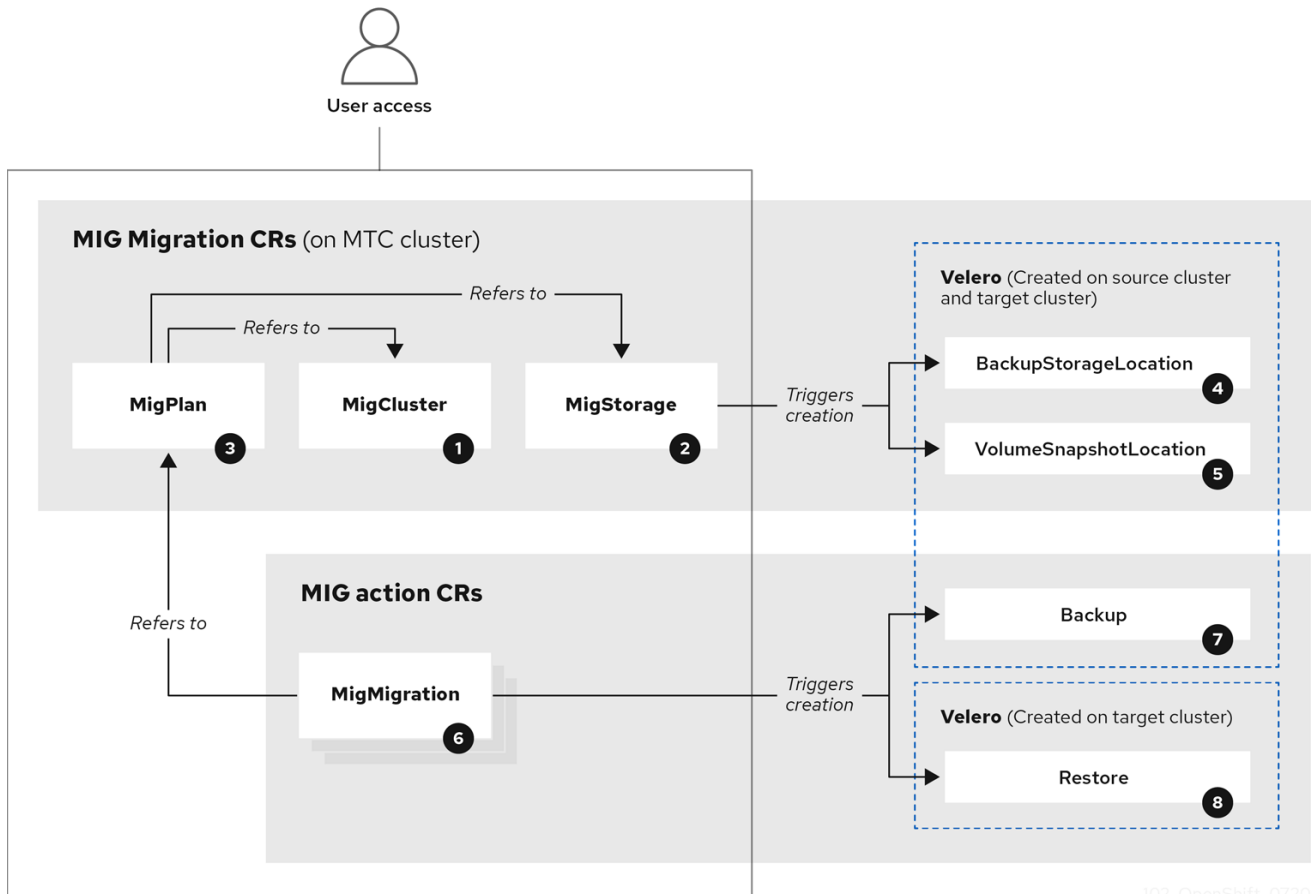
オプション: Halt transactions on the source cluster during migrationのチェックボックスのチェックを解除できます。



OpenShift\_45\_1019

### MTC カスタムリソースについて

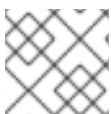
MTC (Migration Toolkit for Containers) は以下のカスタムリソース (CR) を作成します。



102\_OpenShift\_0720

- 1 MigCluster (設定、MTC クラスタ): クラスタ定義
- 2 MigStorage (設定、MTC クラスタ): ストレージ定義
- 3 MigPlan (設定、MTC クラスタ): 移行計画

**MigPlan** CR は、移行されるソースおよびターゲットクラスター、レプリケーションリポジトリ、および namespace を記述します。これは 0、1 または多数の **MigMigration** CR に関連付けられます。



#### 注記

**MigPlan** CR を削除すると、関連付けられた **MigMigration** CR が削除されます。

- 4 BackupStorageLocation (設定、MTC クラスタ): **Velero** バックアップオブジェクトの場所
- 5 VolumeSnapshotLocation (設定、MTC クラスタ): **Velero** ボリュームスナップショットの場所
- 6 MigMigration (アクション、MTC クラスタ): データのステージングまたは移行時に毎回作成される移行。各 **MigMigration** CR は **MigPlan** CR に関連付けられます。
- 7 Backup (アクション、ソースクラスター): 移行計画の実行時に、**MigMigration** CR は各ソースクラスターに 2 つの **Velero** バックアップ CR を作成します。

- Kubernetes オブジェクトのバックアップ CR #1

- PV データのバックアップ CR #2

**8** **Restore** (アクション、ターゲットクラスター): 移行計画の実行時に、**MigMigration** CR はターゲットクラスターに 2 つの **Velero** 復元 CR を作成します。

- PV データの復元 CR #1 (バックアップ CR #2 の使用)
- Kubernetes オブジェクトの復元 CR #2 (バックアップ CR #1 の使用)

## 10.2. MTC カスタムリソースマニフェスト

MTC (Migration Toolkit for Containers) は以下のカスタムリソース (CR) マニフェストを使用して、アプリケーションを移行します。

### 10.2.1. DirectImageMigration

**DirectImageMigration** CR はイメージをソースクラスターから宛先クラスターに直接コピーします。

```
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: 1
    - <source_namespace_1>
    - <source_namespace_2>:<destination_namespace_3> 2
```

**1** 移行するイメージが含まれる namespace を 1 つ以上指定します。デフォルトでは、宛先の namespace の名前はソース namespace と同じになります。

**2** 別の名前で宛先 namespace にマップされるソース namespace。

### 10.2.2. DirectImageStreamMigration

**DirectImageStreamMigration** CR はイメージストリーム参照をソースクラスターから宛先クラスターに直接コピーします。

```
apiVersion: migration.openshift.io/v1alpha1
kind: DirectImageStreamMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <direct_image_stream_migration>
spec:
  srcMigClusterRef:
```

```

name: <source_cluster>
namespace: openshift-migration
destMigClusterRef:
  name: <destination_cluster>
  namespace: openshift-migration
imageStreamRef:
  name: <image_stream>
  namespace: <source_image_stream_namespace>
destNamespace: <destination_image_stream_namespace>

```

### 10.2.3. DirectVolumeMigration

**DirectVolumeMigration** CR は永続ボリューム (PV) をソースクラスターから宛先クラスターに直接コピーします。

```

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** **true** に設定して、宛先クラスターの PV の namespace を作成します。
- 2** **true** に設定して移行後に **DirectVolumeMigrationProgress** CR を削除します。デフォルト値は **false** です。これにより、**DirectVolumeMigrationProgress** CR はトラブルシューティング用に保持されます。
- 3** 宛先クラスターがホストクラスターではない場合は、クラスター名を更新します。
- 4** 移行する PVC を1つ以上指定します。

### 10.2.4. DirectVolumeMigrationProgress

**DirectVolumeMigrationProgress** CR は、**DirectVolumeMigration** CR の進捗を表示します。

```

apiVersion: migration.openshift.io/v1alpha1
kind: DirectVolumeMigrationProgress
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <direct_volume_migration_progress>

```

```
spec:
  clusterRef:
    name: <source_cluster>
    namespace: openshift-migration
  podRef:
    name: <rsync_pod>
    namespace: openshift-migration
```

### 10.2.5. MigAnalytic

**MigAnalytic** CR は、関連付けられた **MigPlan** CR から、イメージの数、Kubernetes リソースおよび 永続ボリューム (PV) 容量を収集します。

収集するデータを設定できます。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigAnalytic
metadata:
  annotations:
    migplan: <migplan>
  name: <miganalytic>
  namespace: openshift-migration
  labels:
    migplan: <migplan>
spec:
  analyzeImageCount: true ①
  analyzeK8SResources: true ②
  analyzePVCapacity: true ③
  listImages: false ④
  listImagesLimit: 50 ⑤
  migPlanRef:
    name: <migplan>
    namespace: openshift-migration
```

- ① オプション: イメージの数を返します。
- ② オプション: Kubernetes リソースの番号、種類、および API バージョンを返します。
- ③ オプション: PV 容量を返します。
- ④ イメージ名の一覧を返します。デフォルトは **false** で、出力が過剰に長くなることはありません。
- ⑤ オプション: **listImages** が **true** の場合、返されるイメージ名の最大数を指定します。

### 10.2.6. MigCluster

**MigCluster** CR は、ホスト、ローカル、またはリモートクラスターを定義します。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
```



```

name: <host_cluster> ❶
namespace: openshift-migration
spec:
  isHostCluster: true ❷
  # The 'azureResourceGroup' parameter is relevant only for Microsoft Azure.
  azureResourceGroup: <azure_resource_group> ❸
  caBundle: <ca_bundle_base64> ❹
  insecure: false ❺
  refresh: false ❻
  # The 'restartRestic' parameter is relevant for a source cluster.
  restartRestic: true ❼
  # The following parameters are relevant for a remote cluster.
  exposedRegistryPath: <registry_route> ❽
  url: <destination_cluster_url> ❾
  serviceAccountSecretRef:
    name: <source_secret> ❿
    namespace: openshift-config

```

- ❶ **migration-controller** Pod がこのクラスターで実行されていない場合には、クラスター名を更新します。
- ❷ **true** の場合、**migration-controller** Pod がこのクラスターで実行されます。
- ❸ Microsoft Azure のみ: リソースグループを指定します。
- ❹ オプション: 自己署名 CA 証明書の証明書バンドルを作成しており、**insecure** なパラメーターの値が **false** の場合、base64 でエンコードされた証明書バンドルを指定します。
- ❺ SSL 検証を無効にするには **true** に設定します。
- ❻ クラスターを検証するには、**true** に設定します。
- ❼ ステージ Pod の作成後に **Restic** Pod をソースクラスターで再起動するには、**true** に設定します。
- ❽ リモートクラスターおよび直接のイメージ移行のみ: 公開されるセキュアなレジストリルートを指定します。
- ❾ リモートクラスターのみ: URL を指定します。
- ❿ リモートクラスターのみ: **Secret** オブジェクトの名前を指定します。

### 10.2.7. MigHook

**MigHook** CR は、指定の移行段階でカスタムコードを実行する移行フックを定義します。最大 4 つの移行フックを作成できます。各フックは異なる移行フェーズで実行されます。

フック名、ランタイム期間、カスタムイメージ、およびフックが実行されるクラスターを設定できます。

フックの移行フェーズおよび namespace は **MigPlan** CR で設定されます。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigHook

```

```

metadata:
  generateName: <hook_name_prefix> ❶
  name: <mighook> ❷
  namespace: openshift-migration
spec:
  activeDeadlineSeconds: 1800 ❸
  custom: false ❹
  image: <hook_image> ❺
  playbook: <ansible_playbook_base64> ❻
  targetCluster: source ❼

```

- ❶ オプション: このパラメーターの値に一意のハッシュが追加され、それぞれの移行フックに一意の名前が追加されます。 **name** パラメーターの値を指定する必要はありません。
- ❷ **generateName** パラメーターを指定しない場合は、移行フック名を指定します。
- ❸ オプション: フックを実行できる最大秒数を指定します。デフォルトは **1800** です。
- ❹ **true** の場合、フックはカスタムイメージです。カスタムイメージには Ansible を含めることも、これを別のプログラミング言語で記述することもできます。
- ❺ カスタムイメージ (例: **quay.io/konveyor/hook-runner:latest**) を指定します。 **custom** が **true** の場合に必要です。
- ❻ base64 でエンコードされた Ansible Playbook。 **custom** が **false** の場合に必要です。
- ❼ フックの実行先のクラスターを指定します。有効な値は **ソース** または **宛先** です。

### 10.2.8. MigMigration

**MigMigration** CR は **MigPlan** CR を実行します。

**Migmigration** CR はステージまたは増分移行を実行し、進行中の移行をキャンセルしたり、完了した移行をロールバックしたりするように設定できます。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migmigration>
  namespace: openshift-migration
spec:
  canceled: false ❶
  rollback: false ❷
  stage: false ❸
  quiescePods: true ❹
  keepAnnotations: true ❺
  verify: false ❻
  migPlanRef:
    name: <migplan>
    namespace: openshift-migration

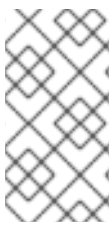
```

- 1 実行中の移行を取り消すには、**true** に設定します。
- 2 完了した移行をロールバックするには、**true** に設定します。
- 3 段階移行を実行するには、**true** に設定します。データが増分的にコピーされ、ソースクラスター上の Pod は停止しません。
- 4 移行時にアプリケーションを停止するには、**true** に設定します。ソースクラスターの Pod は、**Backup** ステージの後に **0** にスケールリングされます。
- 5 移行中に適用されるラベルとアノテーションは保持するには、**true** を設定します。
- 6 宛先クラスターで移行される Pod のステータスをチェックして、**Running** 状態にない Pod の名前を返すには、**true** に設定します。

### 10.2.9. MigPlan

**MigPlan** CR は移行計画のパラメーターを定義します。

宛先 namespace、フックフェーズ、および直接または間接的な移行を設定できます。



#### 注記

デフォルトで、宛先 namespace の名前はソース namespace と同じになります。別の宛先の namespace を設定した場合には、UID および GID の範囲が移行時にコピーされるため、namespace が移行元または移行先ホストで複製されないようにする必要があります。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migplan>
  namespace: openshift-migration
spec:
  closed: false 1
  srcMigClusterRef:
    name: <source_cluster>
    namespace: openshift-migration
  destMigClusterRef:
    name: <destination_cluster>
    namespace: openshift-migration
  hooks: 2
  - executionNamespace: <namespace> 3
    phase: <migration_phase> 4
    reference:
      name: <hook> 5
      namespace: <hook_namespace> 6
      serviceAccount: <service_account> 7
  indirectImageMigration: true 8
  indirectVolumeMigration: false 9
  migStorageRef:

```

```

name: <migstorage>
namespace: openshift-migration
namespaces:
- <source_namespace_1> 10
- <source_namespace_2>
- <source_namespace_3>:<destination_namespace_4> 11
refresh: false 12

```

- 1 true の場合、移行が完了します。この **MigPlan** CR の別の **MigMigration** CR を作成することはできません。
- 2 オプション: 最大 4 つの移行フックを指定できます。各フックは異なる移行フェーズで実行される必要があります。
- 3 オプション: フックが実行される namespace を指定します。
- 4 オプション: フックが実行される移行フェーズを指定します。1 つのフックを 1 つのフェーズに割り当てることができます。有効な値は、**PreBackup**、**PostBackup**、**PreRestore**、および **PostRestore** です。
- 5 オプション: **MigHook** CR の名前を指定します。
- 6 オプション: **MigHook** CR の namespace を指定します。
- 7 オプション: **cluster-admin** 権限でサービスアカウントを指定します。
- 8 false の場合、直接的なイメージ移行が無効にされます。イメージはソースクラスターからレプリケーションリポジトリに、レプリケーションリポジトリから宛先クラスターにコピーされます。
- 9 false の場合、直接的なボリューム移行が無効にされます。PV はソースクラスターからレプリケーションリポジトリに、レプリケーションリポジトリから宛先クラスターにコピーされます。
- 10 namespace を 1 つ以上指定します。ソース namespace のみを指定する場合には、宛先 namespace は同じになります。
- 11 宛先 namespace が異なる場合には、宛先 namespace を指定します。
- 12 true の場合、**MigPlan** CR が検証されます。

## 10.2.10. MigStorage

**MigStorage** CR はレプリケーションリポジトリのオブジェクトストレージを記述します。

Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Storage、Multi-Cloud Object Gateway、および汎用 S3 互換クラウドストレージがサポート対象です。

AWS およびスナップショットのコピー方法には追加のパラメーターがあります。

```

apiVersion: migration.openshift.io/v1alpha1
kind: MigStorage
metadata:
  labels:
    controller-tools.k8s.io: "1.0"

```

```

name: <migstorage>
namespace: openshift-migration
spec:
  backupStorageProvider: <backup_storage_provider> ❶
  volumeSnapshotProvider: <snapshot_storage_provider> ❷
  backupStorageConfig:
    awsBucketName: <bucket> ❸
    awsRegion: <region> ❹
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret> ❺
    awsKmsKeyId: <key_id> ❻
    awsPublicUrl: <public_url> ❼
    awsSignatureVersion: <signature_version> ❽
  volumeSnapshotConfig:
    awsRegion: <region> ❾
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret> ❿
  refresh: false ⓫

```

- ❶ ストレージプロバイダーを指定します。
- ❷ スナップショットのコピー方法のみ: ストレージプロバイダーを指定します。
- ❸ AWS のみ: バケット名を指定します。
- ❹ AWS のみ: バケットリージョン (例: **us-east-1**) を指定します。
- ❺ ストレージ用に作成した **Secret** オブジェクトの名前を指定します。
- ❻ AWS のみ: AWS Key Management Service を使用している場合は、キーの一意の識別子を指定します。
- ❼ AWS のみ: AWS バケットへのパブリックアクセスを付与する場合は、バケット URL を指定します。
- ❽ AWS のみ: バケットに対する要求の認証に使用する AWS 署名バージョン (例: **4**) を指定します。
- ❾ スナップショットを使用したコピー方法のみ: クラスターの地理的なリージョンを指定します。
- ❿ スナップショットを使用したコピー方法のみ: ストレージ用に作成した **Secret** オブジェクトの名前を指定します。
- ⓫ クラスターを検証するには、**true** に設定します。

## 10.3. ログおよびデバッグツール

本セクションでは、トラブルシューティングに使用できるログおよびデバッグツールについて説明します。

### 10.3.1. 移行計画リソースの表示

移行計画リソースを表示して、実行中の移行を監視するか、MTC の Web コンソールおよびコマンドラインインターフェイス (CLI) を使用して失敗した移行のトラブルシューティングを行うことができます。


## 手順

1. MTC Web コンソールで、**Migration plans** をクリックします。
2. 移行計画の横にある **Migrations** 番号をクリックし、**Migrations** ページを表示します。
3. 移行をクリックして、**移行の詳細** を表示します。
4. **Migration resources** を展開して、移行リソースおよびそれらのステータスをツリービューで表示します。



### 注記

移行の失敗をトラブルシューティングするには、失敗した高レベルのリソースで開始し、リソースツリーでより低い位置にあるリソースまで掘り下げます。

5. リソースの横にある Options メニュー  をクリックし、以下のオプションのいずれかを選択します。

- **copy oc describe** コマンドは、コマンドをクリップボードにコピーします。
  - 関連するクラスターにログインしてから、コマンドを実行します。  
リソースの条件およびイベントは YAML 形式で表示されます。
- **Copy oc logs** コマンドは、コマンドをクリップボードにコピーします。
  - 関連するクラスターにログインしてから、コマンドを実行します。  
リソースがログフィルターに対応していると、フィルターされたログが表示されます。
- **JSON ビュー** は、Web ブラウザーで JSON 形式でリソースデータを表示します。  
データは **oc get <resource>** コマンドの出力と同じです。

## 10.3.2. 移行計画ログの表示

移行計画の集計ログを表示できます。MTC の Web コンソールを使用して、コマンドをクリップボードにコピーしてから、コマンドラインインターフェイス (CLI) からコマンドを実行します。

このコマンドは、以下の Pod に関するフィルターされたログを表示します。

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

## 手順

1. MTC Web コンソールで、**Migration plans** をクリックします。
2. 移行計画の横にある **Migrations** 番号をクリックします。
3. **View logs** をクリックします。
4. コピーアイコンをクリックして、**oc logs** コマンドをクリップボードにコピーします。
5. 関連するクラスターにログインし、CLI でコマンドを実行します。  
移行契約の集約ログが表示されます。

### 10.3.3. 移行ログリーダーの使用

移行ログリーダーを使用して、すべての移行ログの単一のフィルタービューを表示できます。

## 手順

1. **mig-log-reader** Pod を取得します。

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

2. 以下のコマンドを実行して、単一の移行ログを表示します。

```
$ oc -n openshift-migration logs -f <mig-log-reader-pod> -c color 1
```

- 1** **-c plain** オプションは、色なしでログを表示します。

### 10.3.4. パフォーマンスメトリックへのアクセス

**MigrationController** カスタムリソース (CR) はメトリクスを記録し、それらをクラスター上のモニタリングストレージにプルします。Prometheus Query Language (PromQL) を使用してメトリクスをクエリーし、移行のパフォーマンス問題を診断できます。すべてのメトリクスは、Migration コントローラー Pod の再起動時にリセットされます。

パフォーマンスメトリックにアクセスし、OpenShift Container Platform Web コンソールを使用してクエリーを実行できます。

## 手順

1. OpenShift Container Platform Web コンソールで、**Observe** → **Metrics** をクリックします。
2. PromQL クエリーを入力し、表示する期間を選択し、**Run Queries** をクリックします。  
Web ブラウザーにすべての結果が表示されない場合は、Prometheus コンソールを使用します。

#### 10.3.4.1. 提供されるメトリクス

**MigrationController** カスタムリソース (CR) は、**MigMigration** CR 数およびその API 要求のメトリクスを提供します。

##### 10.3.4.1.1. cam\_app\_workload\_migrations

このメトリクスは、一定期間の **MigMigration** CR の数です。 **mtc\_client\_request\_count** および **mtc\_client\_request\_elapsed** メトリックと一緒に表示して、API リクエスト情報を移行ステータスの変更と照合するのに役立ちます。このメトリクスは Telemetry に含まれます。

表10.1 cam\_app\_workload\_migrations メトリクス

クエリー可能なラベル名	サンプルラベル値	ラベルの説明
status	<b>running、idle、failed、completed</b>	<b>MigMigration</b> CR のステータス
type	stage、final	<b>MigMigration</b> CR のタイプ

#### 10.3.4.1.2. mtc\_client\_request\_count

このメトリクスは、**MigrationController** が発行する Kubernetes API 要求の累積数です。これは Telemetry に含まれていません。

表10.2 mtc\_client\_request\_count metric

クエリー可能なラベル名	サンプルラベル値	ラベルの説明
cluster	<b>https://migcluster-url:443</b>	要求が発行されたクラスター
component	<b>MigPlan、 MigCluster</b>	要求を発行したサブコントローラー API
function	<b>(*ReconcileMigPlan).Reconcile</b>	要求が発行された関数
kind	<b>SecretList、 Deployment</b>	要求が発行された Kubernetes の種類

#### 10.3.4.1.3. mtc\_client\_request\_elapsed

このメトリクスは、**MigrationController** が発行する Kubernetes API 要求の累積レイテンシー (ミリ秒単位) です。これは Telemetry に含まれていません。

表10.3 mtc\_client\_request\_elapsed metric

クエリー可能なラベル名	サンプルラベル値	ラベルの説明
cluster	<b>https://cluster-url.com:443</b>	要求が発行されたクラスター
component	<b>migplan、 migcluster</b>	要求を発行したサブコントローラー API
function	<b>(*ReconcileMigPlan).Reconcile</b>	要求が発行された関数



クエリー可能なラベル名	サンプルラベル値	ラベルの説明
kind	<b>SecretList、 Deployment</b>	要求が発行された Kubernetes リソース

#### 10.3.4.1.4. 有用なクエリー

この表には、パフォーマンスの監視に使用できる便利なクエリーが記載されています。

表10.4 有用なクエリー

クエリー	説明
<code>mtc_client_request_count</code>	発行した API 要求の数。要求タイプ別でソート
<code>sum(mtc_client_request_count)</code>	発行された API 要求の合計数
<code>mtc_client_request_elapsed</code>	要求タイプ別にソートされた API 要求のレイテンシー
<code>sum(mtc_client_request_elapsed)</code>	API 要求の合計レイテンシー
<code>sum(mtc_client_request_elapsed) / sum(mtc_client_request_count)</code>	API 要求の平均レイテンシー
<code>mtc_client_request_elapsed / mtc_client_request_count</code>	要求タイプ別にソートされた API 要求の平均レイテンシー
<code>cam_app_workload_migrations{status="running"} * 100</code>	実行中の移行の数。要求数とともに簡単に表示できるように 100 で乗算されます。

#### 10.3.5. must-gather ツールの使用

**must-gather** ツールを使用して、MTC カスタムリソースのログ、メトリクス、および情報を収集できます。

**must-gather** データはすべてのカスタマーケースに割り当てられる必要があります。

1時間または 24 時間のデータを収集し、Prometheus コンソールでデータを表示できます。

##### 前提条件

- **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform クラスターにログインする必要があります。
- OpenShift CLI (**oc**) がインストールされている。

##### 手順

1. **must-gather** データを保存するディレクトリーに移動します。
2. 次のデータ収集オプションのいずれかに対して、**oc adm must-gather** コマンドを実行します。
  - 過去 1 時間のデータを収集するには、以下を実行します。

```
$ oc adm must-gather --image=registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7
```

データは **must-gather/must-gather.tar.gz** として保存されます。このファイルを [Red Hat カスタマーポータル](#) で作成したサポートケースにアップロードすることができます。

- 過去 24 時間のデータを収集するには、以下を実行します。

```
$ oc adm must-gather --image=registry.redhat.io/rhmtc/openshift-migration-must-gather-rhel8:v1.7 \
-- /usr/bin/gather_metrics_dump
```

この操作には長時間かかる場合があります。データは **must-gather/metrics/prom\_data.tar.gz** として保存されます。

### Prometheus コンソールを使用したメトリクスデータの表示

Prometheus コンソールでメトリックデータを表示できます。

#### 手順

1. **prom\_data.tar.gz** ファイルを解凍します。

```
$ tar -xvzf must-gather/metrics/prom_data.tar.gz
```

2. ローカルの Prometheus インスタンスを作成します。

```
$ make prometheus-run
```

このコマンドでは、Prometheus URL が出力されます。

#### 出力

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

3. Web ブラウザーを起動して URL に移動し、Prometheus Web コンソールを使用してデータを表示します。
4. データを確認した後に、Prometheus インスタンスおよびデータを削除します。

```
$ make prometheus-cleanup
```

### 10.3.6. Velero CLI ツールを使用した Velero リソースのデバッグ

Velero CLI ツールを使用して、**Backup** および **Restore** カスタムリソース (CR) をデバッグし、ログを取得できます。

Velero CLI ツールは、OpenShift CLI ツールよりも詳細な情報を提供します。

## 構文

**oc exec** コマンドを使用して、Velero CLI コマンドを実行します。

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
<backup_restore_cr> <command> <cr_name>
```

## 例

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
backup describe 0e44ae00-5dc3-11eb-9ca8-df7e5254778b-2d8ql
```

## ヘルプオプション

**velero --help** オプションを使用して、すべての Velero CLI コマンドを一覧表示します。

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
--help
```

## describe コマンド

**velero describe** コマンドを使用して、**Backup** または **Restore** CR に関連する警告とエラーの要約を取得します。

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
<backup_restore_cr> describe <cr_name>
```

## 例

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
backup describe 0e44ae00-5dc3-11eb-9ca8-df7e5254778b-2d8ql
```

## logs コマンド

**velero logs** コマンドを使用して、**Backup** または **Restore** CR のログを取得します。

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
<backup_restore_cr> logs <cr_name>
```

## 例

```
$ oc -n openshift-migration exec deployment/velero -c velero -- ./velero \  
restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

### 10.3.7. 部分的な移行の失敗のデバッグ

Velero CLI を使用して **Restore** カスタムリソース (CR) ログを確認し、部分的な移行の失敗についての警告メッセージをデバッグできます。

部分的な障害は、Velero で移行の失敗を生じさせない問題が発生する際に見られます。たとえば、カスタムリソース定義 (CRD) がない場合や、ソースクラスターおよびターゲットクラスターの CRD バージョン間で不一致がある場合、移行は完了しますが、CR はターゲットクラスターで作成されません。

Velero は問題を部分的な障害としてログに記録し、**Backup** CR の残りのオブジェクトを処理します。

## 手順

1. **MigMigration** CR のステータスを確認します。

```
$ oc get migmigration <migmigration> -o yaml
```

### 出力例

```
status:
  conditions:
  - category: Warn
    durable: true
    lastTransitionTime: "2021-01-26T20:48:40Z"
    message: 'Final Restore openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-
x4lbf: partially failed on destination cluster'
    status: "True"
    type: VeleroFinalRestorePartiallyFailed
  - category: Advisory
    durable: true
    lastTransitionTime: "2021-01-26T20:48:42Z"
    message: The migration has completed with warnings, please look at `Warn` conditions.
    reason: Completed
    status: "True"
    type: SucceededWithWarnings
```

2. Velero **describe** コマンドを使用して **Restore** CR のステータスを確認します。

```
$ oc -n {namespace} exec deployment/velero -c velero -- ./velero \
restore describe <restore>
```

### 出力例

```
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. Velero **logs** コマンドを使用して **Restore** CR ログを確認します。

```
$ oc -n {namespace} exec deployment/velero -c velero -- ./velero \
restore logs <restore>
```

### 出力例

```
time="2021-01-26T20:48:37Z" level=info msg="Attempting to restore migration-example:
migration-example" logSource="pkg/restore/restore.go:1107" restore=openshift-
migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

```
time="2021-01-26T20:48:37Z" level=info msg="error restoring migration-example: the server
could not find the requested resource" logSource="pkg/restore/restore.go:1170"
restore=openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

**Restore** CR のログエラーメッセージの **the server could not find the requested resource** は、部分的に失敗した移行の原因を示します。

### 10.3.8. トラブルシューティング向けの MTC カスタムリソースの使用

以下の MTC (Migration Toolkit for Containers) カスタムリソース (CR) を確認し、失敗した移行のトラブルシューティングを行うことができます。

- **MigCluster**
- **MigStorage**
- **MigPlan**
- **BackupStorageLocation**  
**BackupStorageLocation** CR には、CR を作成した MTC インスタンスを識別するための **migrationcontroller** ラベルが含まれます。

```
labels:
  migrationcontroller: ebe13bee-c803-47d0-a9e9-83f380328b93
```

- **VolumeSnapshotLocation**  
**VolumeSnapshotLocation** CR には、CR を作成した MTC インスタンスを特定するための **migrationcontroller** ラベルが含まれます。

```
labels:
  migrationcontroller: ebe13bee-c803-47d0-a9e9-83f380328b93
```

- **MigMigration**
- **Backup**  
MTC は、移行された永続ボリューム (PV) の回収ポリシーをターゲットクラスターで **Retain** に変更します。**Backup** CR には、元の回収ポリシーを示す **openshift.io/orig-reclaim-policy** アノテーションが含まれます。移行した PV の回収ポリシーを手動で復元できます。
- **Restore**

#### 手順

1. **openshift-migration** namespace の **MigMigration** CR を一覧表示します。

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

#### 出力例

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

2. **MigMigration** CR を検査します。

-

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

出力は以下の例のようになります。

### MigMigration の出力例

```
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>
```

### PV データを記述する Velero バックアップ CR #2 の出力例

```
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
```

```

migration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
velero.io/storage-location: myrepo-vpzq9
name: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
namespace: openshift-migration
resourceVersion: "87313"
selfLink: /apis/velero.io/v1/namespaces/openshift-migration/backups/88435fe0-c9f8-11e9-85e6-5d593ce65e10-59gb7
uid: c80dbbc0-c9f8-11e9-95ad-0205fe66cbb6
spec:
  excludedNamespaces: []
  excludedResources: []
  hooks:
    resources: []
  includeClusterResources: null
  includedNamespaces:
  - sock-shop
  includedResources:
  - persistentvolumes
  - persistentvolumeclaims
  - namespaces
  - imagestreams
  - imagestreamtags
  - secrets
  - configmaps
  - pods
  labelSelector:
    matchLabels:
      migration-included-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
  storageLocation: myrepo-vpzq9
  ttl: 720h0m0s
  volumeSnapshotLocations:
  - myrepo-wv6fx
status:
  completionTimestamp: "2019-08-29T01:02:36Z"
  errors: 0
  expiration: "2019-09-28T01:02:35Z"
  phase: Completed
  startTimestamp: "2019-08-29T01:02:35Z"
  validationErrors: null
  version: 1
  volumeSnapshotsAttempted: 0
  volumeSnapshotsCompleted: 0
  warnings: 0

```

## Kubernetes リソースを記述する Velero CR #2 の出力例

```

apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.90.187:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-36f54ca7-c925-11e9-825a-

```

```

06fa9fb68c88
creationTimestamp: "2019-08-28T00:09:49Z"
generateName: e13a1b60-c927-11e9-9555-d129df7f3b96-
generation: 3
labels:
  app.kubernetes.io/part-of: migration
  migmigration: e18252c9-c927-11e9-825a-06fa9fb68c88
  migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
namespace: openshift-migration
resourceVersion: "82329"
selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-
d129df7f3b96-gb8nx
uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources:
    - nodes
    - events
    - events.events.k8s.io
    - backups.velero.io
    - restores.velero.io
    - resticrepositories.velero.io
  includedNamespaces: null
  includedResources: null
  namespaceMapping: null
  restorePVs: true
status:
  errors: 0
  failureReason: ""
  phase: Completed
  validationErrors: null
  warnings: 15

```

## 10.4. 一般的な問題および懸念事項

このセクションでは、移行時の問題を引き起こす可能性のある懸念事項および一般的な問題について説明します。

### 10.4.1. ボリュームの直接移行が完了しない

ボリュームの直接移行が完了しない場合、ターゲットクラスターにソースクラスターと同じ **node-selector** アノテーションが含まれていない場合があります。

MTC (Migration Toolkit for Containers) は、SCC (Security Context Constraints) およびスケジューリング要件を保持するためにすべてのアノテーションで namespace を移行します。ボリュームの直接移行の際に、MTC はソースクラスターから移行された namespace のターゲットクラスターで Rsync 転送 Pod を作成します。ターゲットクラスター namespace にソースクラスター namespace と同じアノテーションがない場合、Rsync 転送 Pod はスケジューリングできません。Rsync Pod は **Pending** 状態のままになります。

以下の手順に従って、この問題を特定し、修正できます。

#### 手順



1. **MigMigration** CR のステータスを確認します。

```
$ oc describe migmigration <pod> -n openshift-migration
```

出力には、以下のようなステータス情報が含まれます。

#### 出力例

```
Some or all transfer pods are not running for more than 10 mins on destination cluster
```

2. ソースクラスターで、移行した namespace の詳細を取得します。

```
$ oc get namespace <namespace> -o yaml 1
```

- 1** 移行した namespace を指定します。

3. ターゲットクラスターで、移行した namespace を編集します。

```
$ oc edit namespace <namespace>
```

4. 以下の例のように、欠落している **openshift.io/node-selector** アノテーションを移行した namespace に追加します。

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    openshift.io/node-selector: "region=east"
...
```

5. 移行計画を再度実行します。

## 10.4.2. エラーメッセージおよび解決

このセクションでは、MTC (Migration Toolkit for Containers) で発生する可能性のある一般的なエラーメッセージと、それらの根本的な原因を解決する方法について説明します。

### 10.4.2.1. MTC コンソールへの初回アクセス時に CA 証明書エラーが表示されます。

MTC コンソールへの初回アクセスを試みる際に **CA certificate error** が表示される場合、クラスターのいずれかでの自己署名 CA 証明書の使用が原因である可能性があります。

この問題を解決するには、エラーメッセージに表示される **oauth-authorization-server** URL に移動し、証明書を受け入れます。この問題を永続的に解決するには、Web ブラウザーの信頼ストアに証明書を追加します。

証明書を受け入れた後に **Unauthorized** メッセージが表示される場合は、MTC コンソールに移動し、Web ページを更新します。

### 10.4.2.2. MTC コンソールの OAuth タイムアウトエラー

自己署名証明書を受け入れた後に **connection has timed out** メッセージが MTC コンソールに表示される場合、以下の原因が考えられます。

- OAuth サーバーへのネットワークアクセスが中断された。
- OpenShift Container Platform Web コンソールへのネットワークアクセスが中断された。
- **oauth-authorization-server** URL へのアクセスをブロックするプロキシ設定。詳細は、[MTC console inaccessible because of OAuth timeout error](#) を参照してください。

タイムアウトの原因を特定できます。

- ブラウザーの Web インスペクターで MTC コンソールの Web ページを検査します。
- **Migration UI** Pod ログでエラーの有無を確認します。

#### 10.4.2.3. 不明な認証局エラーで署名された証明書

自己署名証明書を使用して MTC (Migration Toolkit for Containers) のクラスターまたはレプリケーションリポジトリのセキュリティーを保護する場合、証明書の検証は **Certificate signed by unknown authority** というエラーメッセージを出して失敗する可能性があります。

カスタム CA 証明書バンドルファイルを作成し、クラスターまたはレプリケーションリポジトリの追加時に MTC の Web コンソールでこれをアップロードできます。

#### 手順

リモートエンドポイントから CA 証明書をダウンロードし、これを CA バンドルファイルとして保存します。

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

1 エンドポイントのホスト FQDN およびポートを指定します (例: **api.my-cluster.example.com:6443**)。

2 CA バンドルファイルの名前を指定します。

#### 10.4.2.4. Velero Pod ログのバックアップストレージの場所についてのエラー

**Velero Backup** カスタムリソースに、存在しないバックアップストレージロケーション (BSL) が含まれる場合、**Velero** Pod ログには以下のエラーメッセージが表示される可能性があります。

```
$ oc logs <Velero_Pod> -n openshift-migration
```

#### 出力例

```
level=error msg="Error checking repository for stale locks" error="error getting backup storage location: BackupStorageLocation.velero.io \"ts-dpa-1\" not found" error.file="/remote-source/src/github.com/vmware-tanzu/velero/pkg/restic/repository_manager.go:259"
```

これらのエラーメッセージは無視できます。BSL がなくても、移行は失敗しません。

### 10.4.2.5. Velero Pod ログの Pod ボリュームバックアップのタイムアウトエラー

Restic のタイムアウトにより移行が失敗する場合、以下のエラーが **Velero** Pod ログに表示されます。

```
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
```

**restic\_timeout** のデフォルト値は1時間です。大規模な移行では、このパラメーターの値を大きくすることができます。値を高くすると、エラーメッセージが返されるタイミングが遅れる可能性があることに注意してください。

#### 手順

1. OpenShift Container Platform Web コンソールで、**Operators** → **Installed Operators** に移動します。
2. **Migration Toolkit for Containers Operator** をクリックします。
3. **MigrationController** タブで、**migration-controller** をクリックします。
4. **YAML** タブで、以下のパラメーター値を更新します。

```
spec:
  restic_timeout: 1h ❶
```

❶ 有効な単位は **h** (時間)、**m** (分)、および **s** (秒) です (例: **3h30m15s**)。

5. **Save** をクリックします。

### 10.4.2.6. MigMigration カスタムリソースの Restic 検証エラー

ファイルシステムデータのコピー方法を使用して永続ボリュームを移行する際にデータ検証が失敗すると、以下のエラーが **MigMigration** CR に表示されます。

#### 出力例

```
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 ❶
    status: "True"
    type: ResticVerifyErrors ❷
```

❶ エラーメッセージは **Restore** CR 名を識別します。

❷ **ResticVerifyErrors** は、検証エラーが含まれる一般的なエラーの警告です。



## 注記

データ検証エラーによって移行プロセスが失敗することはありません。

**Restore** CRを確認して、データ検証エラーのソースを特定できます。

### 手順

1. ターゲットクラスターにログインします。
2. **Restore** CRを表示します。

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

出力では、**PodVolumeRestore** エラーのある永続ボリュームを特定できます。

### 出力例

```
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. **PodVolumeRestore** CRを表示します。

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

出力では、エラーをログに記録した **Restic** Pod を特定できます。

### 出力例

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

4. **Restic** Pod ログを表示し、エラーを見つけます。

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

#### 10.4.2.7. root\_squash を有効にして NFS ストレージから移行する場合の Restic パーミッションエラー

NFS ストレージからデータを移行していて、**root\_squash** が有効にされている場合、**Restic** は **nfsnobody** にマップされ、これには移行を実行するパーミッションがありません。**Restic** Pod ログには以下のエラーが表示されます。

## 出力例

```
backup=openshift-migration/<backup_id> controller=pod-volume-backup error="fork/exec
/usr/bin/restic: permission denied" error.file="/go/src/github.com/vmware-
tanzu/velero/pkg/controller/pod_volume_backup_controller.go:280"
error.function="github.com/vmware-tanzu/velero/pkg/controller.
(*podVolumeBackupController).processBackup"
logSource="pkg/controller/pod_volume_backup_controller.go:280" name=<backup_id>
namespace=openshift-migration
```

Restic の補助グループを作成し、グループ ID を **MigrationController** CR マニフェストに追加して、この問題を解決することができます。

## 手順

1. NFS ストレージで Restic の補助グループを作成します。
2. NFS ディレクトリーに **setgid** ビットを設定して、グループの所有権が継承されるようにします。
3. **restic\_supplemental\_groups** パラメーターを、ソースおよびターゲットクラスターの **MigrationController** CR マニフェストに追加します。

```
spec:
  restic_supplemental_groups: <group_id> ❶
```

- ❶ 補助グループ ID を指定します。

4. **Restic** Pod が再起動し、変更が適用されるまで待機します。

## 10.5. 移行のロールバック

MTC の Web コンソールまたは CLI を使用して移行をロールバックできます。

[移行を手動でロールバック](#) することもできます。

### 10.5.1. MTC の Web コンソールでの移行のロールバック

MTC (Migration Toolkit for Containers) Web コンソールで移行をロールバックできます。



## 注記

以下のリソースは、直接的なボリューム移行 (DVM) の失敗後もデバッグ用に移行した名前空間に留まります。

- 設定マップ (ソースおよび宛先クラスター)
- **Secret** オブジェクト (ソースクラスターと宛先クラスター)
- **Rsync** CR (ソースクラスター)


これらのリソースはロールバックには影響しません。これらは手動で削除できます。

後で同じ移行プランを正常に実行すると、失敗した移行のリソースが自動的に削除されます。

移行の失敗時にアプリケーションが停止されている場合、永続ボリュームでのデータの破損を防ぐために移行をロールバックする必要があります。

移行時にアプリケーションが停止しなかった場合には、ロールバックは必要ありません。元のアプリケーションがソースクラスター上で依然として実行されているためです。

## 手順

1. MTC Web コンソールで、**Migration plans** をクリックします。
2. 移行計画の横にある Options メニュー  をクリックし、**Migration** の下にある **Rollback** を選択します。
3. **Rollback** をクリックし、ロールバックが完了するまで待機します。  
移行計画の詳細に、**Rollback succeeded** が表示されます。
4. ソースクラスターの OpenShift Container Platform Web コンソールでロールバックが正常に行われたことを確認します。
  - a. **Home** → **Projects** をクリックします。
  - b. 移行されたプロジェクトをクリックしてそのステータスを表示します。
  - c. **Routes** セクションで **Location** をクリックし、アプリケーションが機能していることを確認します (該当する場合)。
  - d. **Workloads** → **Pods** をクリックし、Pod が移行した namespace で実行されていることを確認します。
  - e. **Storage** → **Persistent volumes** をクリックして、移行した永続ボリュームが正常にプロビジョニングされていることを確認します。

### 10.5.2. コマンドラインインターフェイスでの移行のロールバック

コマンドラインインターフェイスで **MigMigration** カスタムリソース (CR) を作成して移行をロールバックできます。



## 注記

以下のリソースは、直接的なボリューム移行 (DVM) の失敗後もデバッグ用に移行した名前空間に留まります。

- 設定マップ (ソースおよび宛先クラスター)
- **Secret** オブジェクト (ソースクラスターと宛先クラスター)
- **Rsync** CR (ソースクラスター)

これらのリソースはロールバックには影響しません。これらは手動で削除できます。

後で同じ移行プランを正常に実行すると、失敗した移行のリソースが自動的に削除されます。

移行の失敗時にアプリケーションが停止されている場合、永続ボリュームでのデータの破損を防ぐために移行をロールバックする必要があります。

移行時にアプリケーションが停止しなかった場合には、ロールバックは必要ありません。元のアプリケーションがソースクラスター上で依然として実行されているためです。

## 手順

1. 以下の例に基づいて **MigMigration** CR オブジェクトを作成します。

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migmigration>
  namespace: openshift-migration
spec:
  ...
  rollback: true
  ...
  migPlanRef:
    name: <migplan> ①
    namespace: openshift-migration
EOF
```

- ① 関連付けられた **MigPlan** CR の名前を指定します。
2. MTC の Web コンソールで、移行したプロジェクトリソースがターゲットクラスターから削除されていることを確認します。
3. 移行したプロジェクトリソースがソースクラスターにあり、アプリケーションが実行中であることを確認します。

### 10.5.3. 移行の手動ロールバック

**stage** Pod を削除して、アプリケーションの停止を解除することで、失敗した移行を手動でロールバックできます。

同じ移行プランを正常に実行すると、失敗した移行のリソースが自動的に削除されます。



### 注記

以下のリソースは、直接的なボリューム移行 (DVM) の失敗後も移行した名前空間に留まります。

- 設定マップ (ソースおよび宛先クラスター)
- **Secret** オブジェクト (ソースクラスターと宛先クラスター)
- **Rsync** CR (ソースクラスター)

これらのリソースはロールバックには影響しません。これらは手動で削除できます。

### 手順

1. すべてのクラスターの **stage** Pod を削除します。

```
$ oc delete $(oc get pods -l migration.openshift.io/is-stage-pod -n <namespace>) 1
```

- 1** **MigPlan** CR に指定される名前空間。

2. レプリカを移行前の数にスケーリングして、ソースクラスターでアプリケーションを減らします。

```
$ oc scale deployment <deployment> --replicas=<premigration_replicas>
```

**Deployment** CR の **migration.openshift.io/preQuiesceReplicas** アノテーションには、レプリカの移行前の数が表示されます。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    migration.openshift.io/preQuiesceReplicas: "1"
```

3. アプリケーション Pod がソースクラスターで実行されていることを確認します。

```
$ oc get pod -n <namespace>
```

### 関連情報

- [Web コンソールの使用によるクラスターからの Operator の削除](#)