OpenShift Container Platform 4.3

迁移

从 OpenShift Container Platform 3 迁移到 4

Last Updated: 2020-11-30

OpenShift Container Platform 4.3 迁移

从 OpenShift Container Platform 3 迁移到 4

法律通告

摘要

本文档提供了将 OpenShift Container Platform 集群从版本 3 迁移到版本 4 的信息。

目录

第 1 章 从 OPENSHIFT CONTAINER PLATFORM 3 进行迁移

1.1. 关于将 OPENSHIFT CONTAINER PLATFORM 3 迁移到 4

OpenShift Container Platform 4 包含新的技术和功能，这些技术和功能可使集群具有自我管理、灵活性和自动化的特性。OpenShift Container Platform 4 集群部署和管理的方式与 OpenShift Container Platform 3 有很大不同。

要成功从 OpenShift Container Platform 3 转换到 OpenShift Container Platform 4，您必须检查以下信息：

规划系统迁移

了解 OpenShift Container Platform 版本 3 和 4 之间的不同。在迁移前，请确定您已对存储、网络、日志记录、安全及监控等方面进行了规划。

执行迁移

了解并使用以下工具执行迁移操作：

- 用于迁移应用程序负载的集群应用程序迁移 (CAM) 工具
- 用于迁移 control plane 的 Control Plane Migration Assistant (CPMA)

1.2. 规划迁移

在把系统迁移到 OpenShift Container Platform 4.3 前，需要花时间来正确地规划整个转换过程。OpenShift Container Platform 4 引入了与架构相关的更改和增强，因此您用来管理 OpenShift Container Platform 3 集群的操作可能不适用于 OpenShift Container Platform 4。

注意

本计划文档假定您要从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3。

本文档提供了有关 OpenShift Container Platform 3 和 OpenShift Container Platform 4 之间重要的区别，以及在迁移时需要注意的重要迁移注意事项。有关配置 OpenShift Container Platform 4 集群的详细信息，请查阅 OpenShift Container Platform 文档的适当章节。如需了解有关新功能和其他显著技术更改的详细信息，请参阅 OpenShift Container Platform 4.3 发行注记 。

无法将现有 OpenShift Container Platform 3 集群升级到 OpenShift Container Platform 4。您必须开始新的 OpenShift Container Platform 4 安装。然后使用提供的工具来帮助迁移 control plane 设置和应用程序工作负载。

1.2.1. OpenShift Container Platform 3 和 OpenShift Container Platform 4 的比较

在 OpenShift Container Platform 3 中，管理员单独部署 Red Hat Enterprise Linux (RHEL) 主机，然后在这些主机之上安装 OpenShift Container Platform 来组成集群。管理员负责正确配置这些主机并执行更新。

在 OpenShift Container Platform 4 中，OpenShift Container Platform 集群的部署和管理方式有了显著变化。OpenShift Container Platform 4 包括新的技术和功能，如 Operators、MachineSets 和 Red Hat Enterprise Linux CoreOS (RHCOS)，它们是集群操作的核心。这个技术转换使集群能够自我管理以前需要由管理员执行的一些功能。这也确保平台的稳定性和一致性，并简化了安装和扩展。

如需更多信息，请参阅 OpenShift Container Platform 架构。

1.2.1.1. 架构的不同

不可变基础架构
OpenShift Container Platform 4 使用 Red Hat Enterprise Linux CoreOS (RHCOS)，它旨在运行容器化应用程序，并提供有效的安装、基于 Operator 的管理以及简化的升级。RHCOS 是不可变容器主机，而不是类似 RHEL 的可定制操作系统。RHCOS 使 OpenShift Container Platform 4 能够管理和自动化底层容器主机的部署。RHCOS 是 OpenShift Container Platform 的一部分，这意味着所有版本都在容器中运行，并且都使用 OpenShift Container Platform 部署。

在 OpenShift Container Platform 4 中，control plane 节点必须运行 RHCOS，以确保为 control plane 维护全堆栈的自动化。这使得更新和升级的过程比在 OpenShift Container Platform 3 中要容易。

如需更多信息，请参阅 Red Hat Enterprise Linux CoreOS 。

Operator
Operator 是一种打包、部署和管理 Kubernetes 应用程序的方法。Operator 可简化运行另一部分软件的操作复杂性。它们会关检测您的环境，并使用当前状态实时做出决定。高级 Operator 旨在自动升级并对失败做出适当的响应。

如需更多信息，请参阅了解 Operators。

1.2.1.2. 安装和更新的不同

安装过程
对于安装 OpenShift Container Platform 3.11，需要准备 Red Hat Enterprise Linux (RHEL) 主机，设置集群所需的所有配置值，然后运行 Ansible playbook 来安装和设置集群。

对于 OpenShift Container Platform 4.3，需要使用 OpenShift 安装程序创建集群所需的最小资源集合。集群运行后，您可以使用 Operator 来进一步配置集群并安装新服务。首次启动后，RHCOS 系统由 OpenShift Container Platform 集群中运行的 Machine Config Operator (MCO) 进行管理。

如需更多信息，请参阅安装过程。

如果要将 RHEL worker 机器添加到 OpenShift Container Platform 4.3 集群中，您可以使用 Ansible playbook 在集群运行后加入 RHEL worker 机器。如需更多信息，请参阅在 OpenShift Container Platform 集群中添加 RHEL 计算机器。

基础架构选项
对于 OpenShift Container Platform 3.11，需要在自己准备好的且被自己维护的基础架构上安装集群。对于 OpenShift Container Platform 4，除了可以在您自己提供的基础架构上安装集群外，还提供了一个在 OpenShift Container Platform 安装程序置备和集群维护的基础架构上部署集群的选项。

如需更多信息，请参阅 OpenShift Container Platform 安装概述。

升级集群
在 OpenShift Container Platform 3.11 中，您可以运行 Ansible playbook 来升级集群。在 OpenShift Container Platform 4.3 中，集群管理自己的更新，包括集群节点上的 Red Hat Enterprise Linux CoreOS (RHCOS) 的更新。您可以使用 Web 控制台或使用 OpenShift CLI 的 **oc adm upgrade** 命令轻松升级集群，Operator 会自动升级其自身。如果您的 OpenShift Container Platform 4.3 集群有 Red Hat Enterprise Linux worker 机器，那么您仍需要运行 Ansible playbook 来升级这些 worker 机器。

如需更多信息，请参阅更新集群。

1.2.2. 迁移考虑

查看可能会影响 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4 的更改和其他注意事项。

1.2.2.1. 存储注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3 时，请考虑以下与存储相关的变化。

本地卷持久性存储
只有在 OpenShift Container Platform 4.3 中使用 Local Storage Operator 才支持本地存储。不支持使用 OpenShift Container Platform 3.11 的 local provisioner 方法。

如需更多信息，请参阅使用本地卷的持久性存储。

FlexVolume 持久性存储
FlexVolume 插件位置已与 OpenShift Container Platform 3.11 中的不同。它在 OpenShift Container Platform 4.3 中的新位置为 **/etc/kubernetes/kubelet-plugins/volume/exec**。不再支持可附加的 FlexVolume 插件。

如需更多信息，请参阅使用 FlexVolume 的持久性存储。

使用容器存储接口 (CSI) 的持久性存储
使用 Container Storage Interface (CSI) 的持久性存储在 OpenShift Container Platform 3.11 中是一个技术预览功能 。OpenShift Container Platform 4.3 完全支持 CSI 版本 1.1.0，但不附带任何 CSI 驱动程序。您必须安装自己的驱动程序。

如需更多信息，请参阅使用容器存储接口 (CSI) 的持久性存储。

OpenShift Container Storage
Red Hat OpenShift Container Storage 3 可以与 OpenShift Container Platform 3.11 一起使用，使用 Red Hat Gluster Storage 作为后端存储。

Red Hat OpenShift Container Storage 4 可以与 OpenShift Container Platform 4 一起使用，使用 Red Hat Ceph Storage 作为后备存储。

如需更多信息，请参阅使用 Red Hat OpenShift Container Storage 做为持久性存储 和系统互操作性文档 。

可用的持久性存储选项
OpenShift Container Platform 4.3 中更改了对 OpenShift Container Platform 3.11 的以下持久性存储选项的支持：

- GlusterFS 不再被支持。
- CephFS 作为独立产品不再被支持。
- Ceph RBD 作为独立产品不再被支持。

如果您在 OpenShift Container Platform 3.11 中使用了其中之一，则需要选择不同的持久性存储选项以便在 OpenShift Container Platform 4.3 中获得全面支持。

如需更多信息，请参阅了解持久性存储。

1.2.2.2. 网络注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3 时，请考虑以下与网络相关的变化。

网络隔离模式
虽然用户经常切换为使用 **ovn-multitenant**，但是 OpenShift Container Platform 3.11 的默认网络隔离模式是 **ovs-subnet**。OpenShift Container Platform 4.3 的默认网络隔离模式是 NetworkPolicy。

如果您的 OpenShift Container Platform 3.11 集群使用了 **ovs-subnet** 或 **ovs-multitenant** 模式，则建议在 OpenShift Container Platform 4.3 集群中将模式切换为 NetworkPolicy。NetworkPolicy 支持上游，更灵活，同时还提供 **ovs-multitenant** 的功能。如果您要在 OpenShift Container Platform 4.3 中使用 NetworkPolicy 时仍然希望维持 **ovs-multitenant** 的行为，请按照以下步骤使用 NetworkPolicy 配置多租户隔离。

如需更多信息，请参阅关于网络策略。

加密主机间的网络数据
在 OpenShift Container Platform 3.11 中，您可以使用 IPsec 来加密主机间的网络流量。OpenShift Container Platform 4.3 不支持 IPsec。建议使用 Red Hat OpenShift Service Mesh 在服务间启用 mutual TLS。

如需更多信息，请参阅 了解 Red Hat OpenShift Service Mesh 。

1.2.2.3. 日志记录注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3 时，请考虑以下与日志相关的变化。

部署集群日志记录
OpenShift Container Platform 4 通过使用集群日志记录自定义资源，为集群日志记录提供了一个简单的部署机制。部署后，集群日志记录体验与 OpenShift Container Platform 3.11 相同。

如需更多信息，请参阅关于部署和配置集群日志记录。

聚合日志数据
您无法将 OpenShift Container Platform 3.11 的聚合日志记录数据转换到新的 OpenShift Container Platform 4 集群中。

如需更多信息，请参阅关于集群日志记录。

1.2.2.4. 安全考虑

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3 时，请考虑以下与安全相关的变化。

对发现端点的未验证访问
在 OpenShift Container Platform 3.11 中，未经身份验证的用户可以访问发现端点（例如: **/api/*** 和 **/apis/***）。为了安全起见，OpenShift Container Platform 4.3 不再允许对发现端点进行未经身份验证的访问。如果确实需要允许未经身份验证的访问，可根据需要配置 RBAC 设置，但请务必考虑安全性影响，因为这可能会使内部集群组件暴露给外部网络。

用户身份供应商
为 OpenShift Container Platform 4 配置身份供应商包括以下显著的更改：

- OpenShift Container Platform 4.3 中的请求标头身份供应商需要 mutual TLS，而这在 OpenShift Container Platform 3.11 中不需要。
- OpenShift Container Platform 4.3 中简化了 OpenID Connect 身份供应商的配置。现在，它从供应商的 **/.well-known/OpenID-configuration** 端点获取数据。而之前需要在 OpenShift Container Platform 3.11 中指定。

如需更多信息，请参阅了解身份供应商配置。

1.2.2.5. 监控注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.3 时，请考虑以下与监控相关的变化。

监控基础架构可用性的警报
OpenShift Container Platform 3.11 中，触发的确保监控结构可用的默认警报称为 **DeadMansSwitch** 。在 OpenShift Container Platform 4 中，它被重新命名为 **Watchdog** 。如果您在 OpenShift Container Platform 3.11 中使用带有此警报设置的 PagerDuty 集成，则需要在 OpenShift Container Platform 4 中使用带有 **Watchdog** 警报设置的 PagerDuty 集成。

如需更多信息，请参阅应用自定义 Alertmanager 配置。

1.3. 迁移工具和先决条件

您可以使用 Cluster Application Migration（CAM）工具将应用程序工作负载从 OpenShift Container Platform 3.7、3.9、3.10 和 3.11 迁移到 OpenShift Container Platform 4.3。使用 CAM 工具，您可以控制迁移并最小化应用程序的停机时间。

CAM 工具的 Web 控制台和 API，基于 Kubernetes 自定义资源，您可以按照命名空间迁移有状态应用程序工作负载。

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。

注意

OpenShift Container Platform 4 中已弃用服务目录。您可以将服务目录置备的工作负载资源从 OpenShift Container Platform 3 迁移到 4，但无法在迁移后在这些工作负载上执行服务目录操作，如 **provision**、**deprovision** 或 **update**。

CAM 工具会显示有关服务目录资源的消息（如 **ClusterServiceClass**、**ServiceInstance** 或 **ServiceBinding**）无法迁移。

Control Plane Migration Assistant（CPMA）是一个基于 CLI 的工具，可帮助您迁移 control plane。CPMA 处理 OpenShift Container Platform 3 配置文件并生成自定义资源 (CR) 清单文件，这些文件由 OpenShift Container Platform 4.3 Operator 使用。

重要

在开始迁移前，请查看规划迁移中的信息。

1.3.1. 迁移先决条件

- 必须安装 **podman** 。
- 源集群必须是 OpenShift Container Platform 3.7 、3.9 、3.10 或 3.11。
- 必须将源集群升级到最新的 z-stream 版本。

- 需要在所有集群中都有 **cluster-admin** 权限。
- 源和目标集群必须有对复制存储库的不受限制的网络访问权限。
- 安装 Migration controller 的集群必须具有对其他集群的不受限制的访问权限。
- 如果应用程序使用 **openshift** 命名空间中的镜像，则目标集群中必须有所需的镜像版本。如果没有所需的镜像，您必须更新 **imagestreamtags** 的引用以使用与应用程序兼容的可用版本。如果无法更新 **imagestreamtags**，您可以手动将相关的镜像上传到应用程序命名空间中，并更新应用程序以引用它们。

以下 **imagestreamtags** 已从 OpenShift Container Platform 4.2 中 *删除*：

- **dotnet:1.0**, **dotnet:1.1**, **dotnet:2.0**
- **dotnet-runtime:2.0**
- **mariadb:10.1**
- **mongodb:2.4**、**mongodb:2.6**
- **mysql:5.5**、**mysql:5.6**
- **nginx:1.8**
- **nodejs:0.10**、**nodejs:4**、**nodejs:6**
- **perl:5.16**、**perl:5.20**
- **php:5.5**、**php:5.6**
- **postgresql:9.2**, **postgresql:9.4**, **postgresql:9.5**
- **python:3.3**、**python:3.4**
- **ruby:2.0**、**ruby:2.2**

1.3.2. 关于集群应用程序迁移（CAM）工具

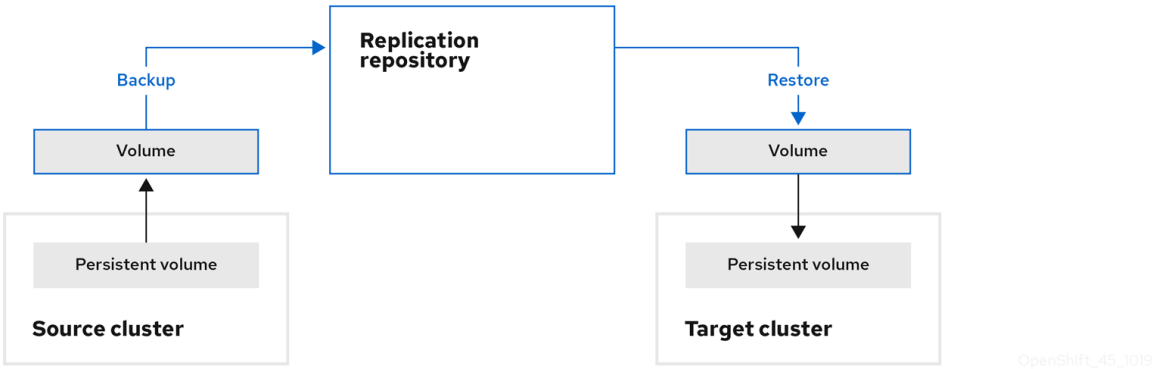
集群应用程序迁移 (CAM) 工具可让您使用 CAM web 控制台或 Kubernetes API 将 OpenShift Container Platform 源集群中的 Kubernetes 资源、持久性卷数据和内部容器镜像迁移到 OpenShift Container Platform 4.3 目标集群。

使用 CAM web 控制台迁移应用程序涉及以下步骤：

1. 在所有集群中安装 Cluster Application Migration Operator
   您可以在有限的或没有互联网访问的受限环境中安装 Cluster Application Migration Operator。源和目标集群必须可以在相互间进行访问，而需要可以访问 registry 的镜像（mirror）.
2. 配置复制存储库，这是 CAM 工具用来迁移数据的中间对象存储
   源和目标集群必须有对复制存储库的不受限制的网络访问权限。在受限环境中，您可以使用内部托管的 S3 存储存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。
3. 在 CAM web 控制台中添加源集群
4. 在 CAM web 控制台中添加复制存储库

5. 创建迁移计划，包含以下数据迁移选项之一：

- **Copy**：CAM 工具将数据从源集群复制到复制存储库，再从复制存储库把数据复制到目标集群。



- **Move**：CAM 工具从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。

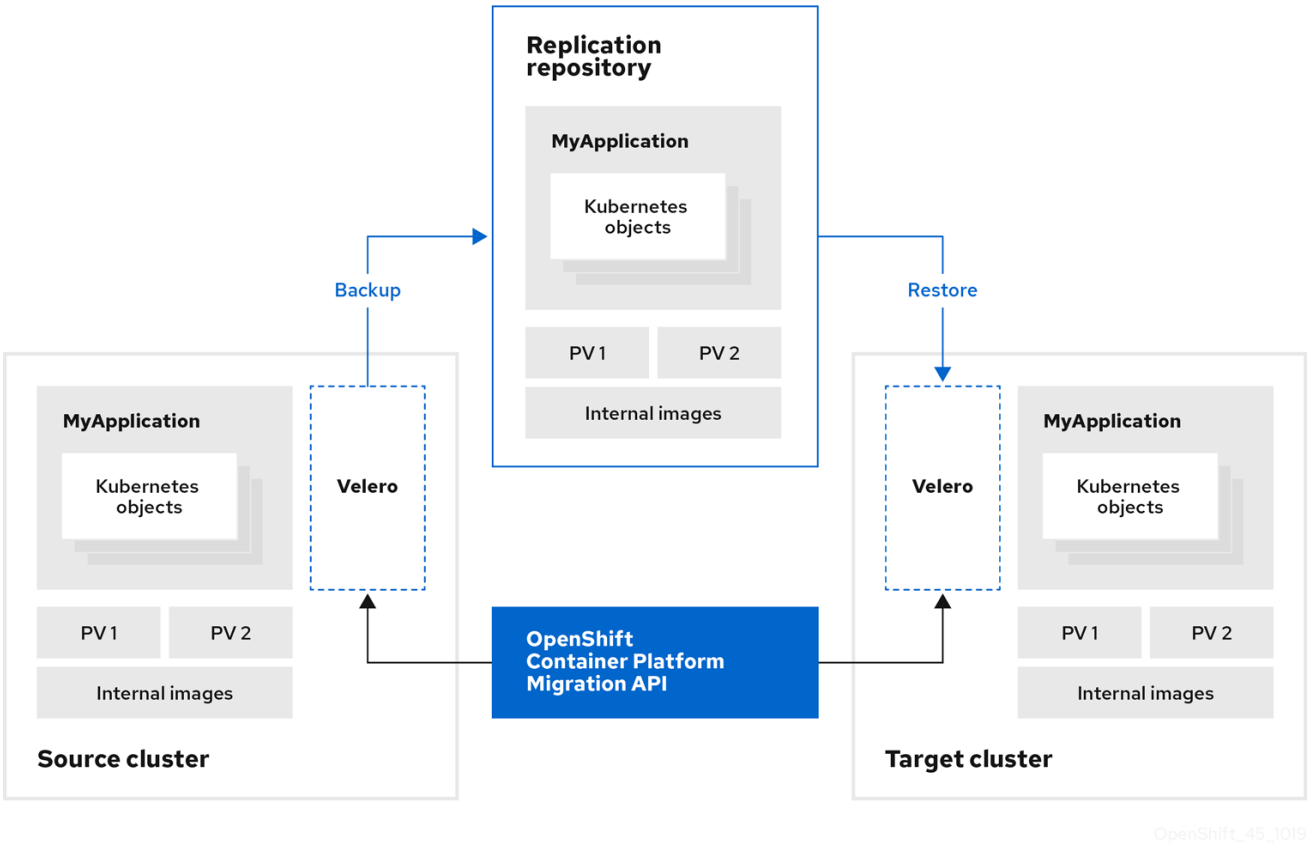
注意

虽然复制存储库没有出现在此图表中，但实际迁移过程需要它。

6. 运行迁移计划，使用以下选项之一：

- **Stage**（可选）在不停止应用程序的情况下将数据复制到目标集群。
  Stage 可以多次运行，以便在迁移前将大多数数据复制到目标。这样可最小化实际迁移时间和应用程序停机时间。

- **Migrate** 在源集群中停止应用程序，并在目标集群中重新创建其资源。您可以选择在不停止应用程序的情况下迁移工作负载。



1.3.3. 关于数据复制方法

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

1.3.3.1. 文件系统复制方法

CAM 工具将数据文件从源集群复制到复制存储库，并从那里复制到目标集群。

表 1.1. 文件系统复制方法概述

优点	限制：
• 集群可以有不同的存储类 • 所有 S3 存储供应商均支持 • 使用 checksum 验证数据（可选）	• 比快照复制方法慢 • 可选的数据校验可能会显著降低性能

1.3.3.2. 快照复制方法

CAM 工具将源集群的数据快照复制到云供应商的对象存储，后者配置为复制存储库。数据在目标集群上恢复。

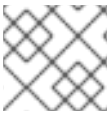
AWS、Google Cloud Provider 和 Microsoft Azure 支持快照复制方法。

表 1.2. 快照复制方法概述

优点	限制：
• 比文件系统复制方法快	• 云供应商必须支持快照。 • 集群必须位于相同的云供应商。 • 集群必须位于同一位置或区域。 • 集群必须具有相同的存储类。 • 存储类必须与快照兼容。

1.3.4. 关于迁移 hook

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。



注意

如果您不想使用 Ansible playbook，您可以创建自定义容器镜像并将其添加到迁移计划中。

迁移 hook 执行的任务包括自定义应用程序默认、手动迁移不受支持的数据类型以及在迁移后更新应用程序。

在源集群或目标集群中以以下迁移步骤之一运行单个迁移 hook:

- PreBackup: 在源集群上启动备份前的任务
- PostBackup: 在源集群中完成备份任务后
- PreRestore: 在目标集群上启动恢复前的任务
- PostRestore: 在目标集群中完成恢复后的任务
  您可以为每个迁移步骤分配一个 hook，单个迁移计划最多可分配四个 hook。

默认 **hook-runner** 镜像为 **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**。该镜像基于 Ansible Runner，并包括用于 Ansible Kubernetes 资源的 **python-openshift** 以及更新的 **oc** 二进制文件。您还可以使用其他 Ansible 模块或工具创建自己的 hook 镜像。

Ansible playbook 作为一个 ConfigMap 挂载到 hook 容器上。hook 容器在具有指定服务帐户和命名空间的集群中作为一个作业（job）运行。作业运行（即使初始 Pod 被驱除或终止），直到达到默认的 **backoffLimit** (**6**)或成功完成为止。

1.3.5. 关于 Control Plane Migration Assistant

Control Plane Migration Assistant (CPMA) 是一个基于 CLI 的工具，它可帮助您将 control plane 从 OpenShift Container Platform 3.7（或更新版本）迁移到 OpenShift Container Platform 4.3。CPMA 处理 OpenShift Container Platform 3 配置文件并生成自定义资源 (CR) 清单文件，这些文件由 OpenShift Container Platform 4.3 Operator 使用。

因为 OpenShift Container Platform 3 和 4 的配置差别很大，所以不是所有的参数都会被处理。CPMA 可生成报告，描述功能是否被全面支持、部分支持或根本不支持。

配置字段

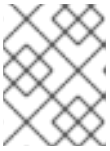
CPMA 使用 Kubernetes 和 OpenShift Container Platform API 来访问 OpenShift Container Platform 3 集群中的以下配置文件：

- master 配置文件（默认为 **/etc/origin/master/master-config.yaml**）
- CRI-O 配置文件（默认为 **/etc/crio/crio.conf**）
- etcd 配置文件（默认为 **/etc/etcd/etcd.conf**）
- 镜像 registry 文件（默认为 **/etc/containers/registries.conf**）
- 依赖性配置文件：
  - 密码文件（例如: HTPasswd）
  - ConfigMaps
  - secret

CR 清单

CPMA 为以下配置生成 CR 清单：

- API 服务器 CA 证书: **100_CPMA-cluster-config-APISecret.yaml**



注意

如果您使用的是未签名的 API 服务器 CA 证书，则必须手动将证书添加到目标集群中。

- CRI-O: **100_CPMA-crio-config.yaml**
- 集群资源配额: **100_CPMA-cluster-quota-resource-x.yaml**
- 项目资源配额: **100_CPMA-resource-quota-x.yaml**
- 可移植镜像 registry（**/etc/registries/registries.conf**）和可移植镜像策略（**etc/origin/master/master-config.yam**）: **100_CPMA-cluster-config-image.yaml**
- OAuth 提供程序：**100_CPMA-cluster-config-oauth.yaml**
- 项目配置: **100_CPMA-cluster-config-project.yaml**
- 调度程序：**100_CPMA-cluster-config-scheduler.yaml**
- SDN: **100_CPMA-cluster-config-sdn.yaml**

1.4. 部署集群应用程序迁移（CAM）工具

您可以在一个 OpenShift Container Platform 4.3 目标集群和一个 OpenShift Container Platform 3 源集群中安装 Cluster Application Migration Operator。默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

注意

可选： 您可以配置 Cluster Application Migration Operator，以便在 OpenShift Container Platform 3 集群或远程集群中安装 CAM 工具。

在受限环境中，您可以从本地镜像 registry 安装 Cluster Application Migration Operator。

在集群中安装 Cluster Application Migration Operator 后,您可以启动 CAM 工具。

1.4.1. 安装 Cluster Application Migration Operator

您可以使用操作生命周期管理器（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator，也可以手动在 OpenShift Container Platform 3 源集群中安装。

1.4.1.1. 在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 点击 **Create**。
9. 点 **Workloads → Pod** 来验证 Controller Manager 、Migration UI 、Restic 和 Velero Pod 是否正在运行。

1.4.1.2. 在 OpenShift Container Platform 3 源集群上安装 Cluster Application Migration Operator

您可以在 OpenShift Container Platform 3 源集群上手工安装 Cluster Application Migration Operator。

先决条件

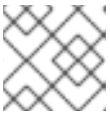
- 访问 **registry.redhat.io**

- OpenShift Container Platform 3 集群被配置为从 **registry.redhat.io** 拉取（pull）镜像。为了拉取镜像，您需要创建一个 **imagestreamsecret**，并把它复制到集群中的每个节点。

流程

1. 使用您的红帽客户门户网站账户登陆到 **registry.redhat.io**：

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



注意

如果系统是为无根 Podman 容器配置的，则此过程不需要 **sudo**。

2. 下载 **operator.yml** 文件：

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

3. 下载 **controller-3.yml** 文件：

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

4. 登录您的 OpenShift Container Platform 3 集群。

5. 验证集群可以在 **registry.redhat.io** 中进行身份验证：

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

6. 创建 Cluster Application Migration Operator CR 对象：

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

输出类似于以下：

```
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) 您可以忽略 **Error from server (AlreadyExists)** 信息。它们是由 Cluster Application Migration Operator 为早期版本的 OpenShift Container Platform 3 创建资源造成的，这些资源在以后的版本中已提供。

7. 创建 Migration controller CR 对象：

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

8. 确认 Velero 和 Restic Pod 正在运行：

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

1.4.2. 在受限环境中安装 Cluster Application Migration Operator

您可以使用操作生命周期管理器（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator，也可以手动在 OpenShift Container Platform 3 源集群中安装。

对于 OpenShift Container Platform 4.3，您可以构建自定义 Operator 目录镜像，将其推送到本地镜像 registry 的镜像系统（mirror），并将 OLM 配置为从本地 registry 安装 Cluster Application Migration Operator。运行 **oc adm catalog mirror** 命令时会创建一个 **mapping.txt** 文件。

在 OpenShift Container Platform 3 集群中，您可以基于 Operator 镜像创建清单文件，并编辑该文件以指向本地镜像 registry。清单文件中的 **image** 值使用来自于 **mapping.txt** 文件的 **sha256** 值。然后，您可以使用本地镜像来创建 Cluster Application Migration Operator。

其他资源

- 在受限网络中使用 Operator Lifecycle Manager

1.4.2.1. 构建 Operator 目录镜像

集群管理员可以构建自定义 Operator 目录镜像，供 Operator Lifecycle Manager (OLM) 使用，并将镜像推送到支持 Docker v2-2 的容器镜像 registry。对于受限网络中的集群，此 registry 可以是集群有网络访问权限的 registry，如在受限网络安装过程中创建的镜像 registry。

重要

OpenShift Container Platform 集群的内部 registry 不能用作目标 registry，因为它不支持没有标签的推送，而在镜像（mirrir）过程中需要这个功能。

在这一示例中，流程假定在使用镜像 registry 时可访问您的网络以及互联网。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- **oc** 版本 4.3.5+
- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

- 如果您正在使用 quay.io 帐户可访问的私有命名空间，您必须设置 Quay 身份验证令牌。使用您的 quay.io 凭证对登录 API 发出请求，从而设置用于 **--auth-token** 标志的 **AUTH_TOKEN** 环境变量：

```
$ AUTH_TOKEN=$(curl -sH "Content-Type: application/json" \
    -XPOST https://quay.io/cnr/api/v1/users/login -d '
    {
        "user": {
            "username": """<quay_username>""",
            "password": """<quay_password>"""
        }
    }' | jq -r '.token')
```

流程

1. 在没有网络访问限制的工作站中，与目标镜像（mirror） registry 进行身份验证：

```
$ podman login <registry_host_name>
```

还可使用 **registry.redhat.io** 验证，以便在构建期间拉取基础镜像：

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

2. 根据 quay.io 中的 **redhat-operators** 目录构建目录镜像，进行标记并将其推送到您的镜像 registry：

```
$ oc adm catalog build \
    --appregistry-org redhat-operators \ 1
    --from=registry.redhat.io/openshift4/ose-operator-registry:v4.3 \ 2
    --filter-by-os="linux/amd64" \ 3
    --to=<registry_host_name>:<port>/olm/redhat-operators:v1 \ 4
    [-a ${REG_CREDS}] \ 5
    [--insecure] \ 6
    [--auth-token "${AUTH_TOKEN}"] 7

INFO[0013] loading Bundles
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
...
Pushed sha256:f73d42950021f9240389f99ddc5b0c7f1b533c054ba344654ff1edaf6bf827e3
to example_registry:5000/olm/redhat-operators:v1
```

1. 从 App Registry 实例中拉取的机构（命名空间）。
2. 使用与目标 OpenShift Container Platform 集群主版本和次版本匹配的标签，将 **--from** 设置为 **ose-operator-registry** 基础镜像。
3. 将 **--filter-by-os** 设置为用于基本镜像的操作系统和架构，该镜像必须与目标 OpenShift Container Platform 集群匹配。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
4. 为您的目录镜像命名并包含标签，例如: **v1**。
5. 可选：如果需要，指定 registry 凭证文件的位置。

6 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。

7 可选：如果使用其他不公开的应用程序 registry 目录，则需要指定 Quay 身份验证令牌。

有时红帽目录中会意外引入无效的清单；当发生这种情况时，您可能会看到一些错误：

```
...
INFO[0014] directory
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
file=4.2 load=package
W1114 19:42:37.876180   34665 builder.go:141] error building database: error loading
package into db: fuse-camel-k-operator.v7.5.0 specifies replacement that couldn't be found
Uploading ... 244.9kB/s
```

这些错误通常不是致命的，如果所提及 Operator 软件包不包含您计划安装的 Operator 或其依赖项，则可以忽略它们。

1.4.2.2. 针对受限网络配置 OperatorHub

集群管理员可以使用自定义 Operator 目录镜像将 OLM 和 OperatorHub 配置为在受限网络环境中使用本地内容。本例中的流程使用之前构建并推送到受支持的 registry 的自定义 **redhat-operators** 目录镜像。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- 推送到受支持的 registry 的自定义 Operator 目录镜像
- **oc** 版本 4.3.5+
- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

流程

1. 通过在 spec 中添加 **disableAllDefaultSources: true** 来禁用默认 OperatorSource：

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

该操作将禁用在 OpenShift Container Platform 安装期间默认配置的默认 OperatorSource。

2. **oc adm catalog mirror** 命令提取自定义 Operator catalog 镜像的内容，以生成镜像（mirror）所需的清单：您可以选择：
   - 允许该命令的默认行为在生成清单后自动将所有镜像内容镜像到您的镜像 registry 中，或者
   - 添加 **--manifests-only** 标志来只生成镜像所需的清单，但并不实际将镜像内容镜像到

registry。这对检查哪些要镜像（mirror）非常有用。如果您只需要部分内容的话，可以对映射列表做出任何修改。然后，您可以使用该文件与 **oc image mirror** 命令一起，在以后的步骤中镜像修改的镜像列表。

在没有网络访问限制的工作站中，运行以下命令：

```
$ oc adm catalog mirror \
    <registry_host_name>:<port>/olm/redhat-operators:v1 \ 1
    <registry_host_name>:<port> \
    [-a ${REG_CREDS}] \ 2
    [--insecure] \ 3
    [--filter-by-os="<os>/<arch>"] \ 4
    [--manifests-only] 5
```

1. 指定 Operator 目录镜像。
2. 可选：如果需要，指定 registry 凭证文件的位置。
3. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。
4. 可选： 因为目录可能会引用支持多个架构和操作系统的镜像，您可以根据架构和操作系统进行过滤来只对匹配的镜像进行镜像（mirror）。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
5. 可选：只生成镜像所需的清单，但并不实际将镜像内容镜像到 registry。

输**出示例**

```
using database path mapping: /:/tmp/190214037
wrote database to /tmp/190214037
using database at: /tmp/190214037/bundles.db 1
...
```

1. 命令生成的临时数据库。

在运行命令后，会在当前目录中生成 **<image_name>-manifests/** 目录以及以下文件：

- 用来定义 ImageContentSourcePolicy 对象的 **imageContentSourcePolicy.yaml**，它可以将节点配置为在 Operator 清单中存储的镜像（image）引用和镜像 (mirror) 的 registry 间进行转换。
- **mapping.txt** 文件，在其中包含所有源镜像，并将它们映射到目标 registry。此文件与 **oc image mirror** 命令兼容，可用于进一步自定义镜像（mirror）配置。

3. 如果您在上一步中使用 **--manifests-only** 标志，并只想镜像部分内容：
   a. 将 **mapping.txt** 文件中的镜像列表改为您的规格。如果您不确定要镜像的镜像子集的名称和版本，请使用以下步骤查找：
      i. 对 **oc adm catalog mirror** 命令生成的临时数据库运行 **sqlite3** 工具，以检索与一般搜索查询匹配的镜像列表。输出以后如何编辑 **mapping.txt** 文件的帮助信息。
      例如，要检索与字符串 **clusterlogging.4.3** 类似的镜像列表：

```
$ echo "select * from related_image \
    where operatorbundle_name like 'clusterlogging.4.3%';" \
    | sqlite3 -line /tmp/190214037/bundles.db 1
```

1 请参阅 **oc adm catalog mirror** 命令的输出结果来查找数据库文件的路径。

输**出示例**

```
image = registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0

image = registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf75
9c506
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0
...
```

ii. 使用上一步的结果来编辑 **mapping.txt** 文件，使其只包含您要镜像的镜像子集。例如，您可以使用前面示例输出中的 **image** 值来找出您的 **mapping.txt** 文件中存在以下匹配行：

mapping.txt 中的匹配镜像映射

```
registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61=<registry_host_name>:<port>/openshift4-ose-logging-kibana5:a767c8f0
registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf75
9c506=<registry_host_name>:<port>/openshift4-ose-oauth-proxy:3754ea2b
```

在这个示例中，如果您只想镜像这些 image，可以在 **mapping.txt** 文件中删除所有其他条目，仅保留上述两行。

b. 在您的没有网络访问限制的工作站中，使用您修改的 **mapping.txt** 文件，使用 **oc image mirror** 命令将镜像镜像到 registry:

```
$ oc image mirror \
    [-a ${REG_CREDS}] \
    -f ./redhat-operators-manifests/mapping.txt
```

4. 应用 ImageContentSourcePolicy:

```
$ oc apply -f ./redhat-operators-manifests/imageContentSourcePolicy.yaml
```

5. 创建一个 CatalogSource 对象来引用您的目录镜像。

a. 按照您的规格修改以下内容，并将它保存为 **catalogsource.yaml** 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
```

```
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: <registry_host_name>:<port>/olm/redhat-operators:v1 1
  displayName: My Operator Catalog
  publisher: grpc
```

1 指定您的自定义 Operator 目录镜像。

b. 使用该文件创建 CatalogSource 对象：

```
$ oc create -f catalogsource.yaml
```

6. 确定成功创建以下资源。

a. 检查 Pod:

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

输出示例

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

b. 检查 CatalogSource:

```
$ oc get catalogsource -n openshift-marketplace
```

输出示例

```
NAME                  DISPLAY               TYPE PUBLISHER  AGE
my-operator-catalog   My Operator Catalog   grpc           5s
```

c. 检查 PackageManifest:

```
$ oc get packagemanifest -n openshift-marketplace
```

输出示例

```
NAME    CATALOG             AGE
etcd    My Operator Catalog  34s
```

现在，您可在受限网络 OpenShift Container Platform 集群的 web 控制台中，通过 **OperatorHub** 安装 Operator。

1.4.2.3. 在一个受限的环境中的 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

先决条件

- 已创建自定义 Operator 目录并将其推送到 registry 的镜像。
- 已将 OLM 配置为从 registry 镜像来安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 点击 **Create**。
9. 点 **Workloads → Pod** 来验证 Controller Manager、Migration UI、Restic 和 Velero Pod 是否正在运行。

1.4.2.4. 在一个受限的环境中的 OpenShift Container Platform 3 源集群上安装 Cluster Application Migration Operator

您可以基于 Cluster Application Migration Operator 镜像创建清单文件，并编辑清单以指向本地的 registry 镜像。您可以使用 OLM 在 OpenShift Container Platform 3 源集群上安装 Cluster Application Migration Operator。

先决条件

- 访问 **registry.redhat.io**
- 没有网络访问限制的 Linux 工作站
- 支持 Docker v2-2 的镜像（mirror）registry
- 自定义 Operator 目录推送到一个 registry 镜像

流程

1. 通过一个没有网络访问限制的工作站，使用您的红帽客户门户网站凭证登录 **registry.redhat.io**:

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

注意

如果系统是为无根 Podman 容器配置的，则此过程不需要 **sudo**。

2. 下载 **operator.yml** 文件：

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

3. 下载 **controller-3.yml** 文件：

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

4. 从在 OpenShift Container Platform 4 集群中运行 **oc adm catalog mirror** 时创建的 **mapping.txt** 文件获取 Operator 镜像值：

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

输出显示了 **registry.redhat.io** 镜像和您的镜像 registry 镜像之间的映射：

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

5. 更新 **operator.yml** 文件中的 **image** 和 **REGISTRY** 的值：

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

1 2 在 **mapping.txt** 文件中指定您的镜像 registry 和 Operator 镜像的 **sha256** 值。

3 指定您的镜像 registry。

6. 登录您的 OpenShift Container Platform 3 集群。

7. 创建 Cluster Application Migration Operator CR 对象：

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

输出类似于以下：

```
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 您可以忽略 **Error from server (AlreadyExists)** 信息。它们是由 Cluster Application Migration Operator 为早期版本的 OpenShift Container Platform 3 创建资源造成的，这些资源在以后的版本中已提供。

8. 创建 Migration controller CR 对象：

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

9. 确认 Velero 和 Restic Pod 正在运行：

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

1.4.3. 启动 CAM web 控制台

您可以在浏览器中启动 CAM web 控制台。

流程

1. 登录到已安装 CAM 工具的 OpenShift Container Platform 集群。
2. 运行以下命令来获取 CAM web 控制台 URL:

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

输出类似于以下： **https://migration-openshift-migration.apps.cluster.openshift.com**。

3. 启动浏览器并进入 CAM web 控制台。

注意

如果在安装 Cluster Application Migration Operator 后尝试立即访问 CAM web 控制台，则该控制台可能无法加载，因为 Operator 仍然在配置集群。等待几分钟后重试。

4. 如果您使用自签名的 CA 证书，则会提示您接受源集群 API 服务器的 CA 证书。网页会引导您接受剩余证书的过程。

5. 使用 OpenShift Container Platform 的**用户名**和**密码**进行登陆。

1.5. 配置复制存储库

您必须将对象存储配置为用作复制存储库。集群应用程序迁移（CAM）工具将数据从源集群复制到复制存储库，然后从复制存储库复制到目标集群。

CAM 工具支持使用文件系统和快照的数据复制方法来把数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

支持以下存储供应商：

- 多云对象网关 (MCG)
- Amazon Web Services (AWS) S3
- Google Cloud Provider (GCP)
- Microsoft Azure
- 通用 S3 对象存储，例如 Minio 或 Ceph S3

源和目标集群必须有对复制存储库的不受限制的网络访问权限。

在受限环境中，您可以创建一个内部托管的复制存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。

1.5.1. 配置 MCG 存储桶做为复制存储库

您可以安装 OpenShift Container Storage Operator，并将一个 Multi-Cloud Object Gateway (MCG) 存储桶配置为复制存储库。

1.5.1.1. 安装 OpenShift Container Storage Operator

您可以从 OperatorHub 安装 OpenShift Container Storage Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** （本例中为 **OCS**）来查找 **OpenShift Container Storage Operator**。
3. 选择 **OpenShift Container Storage Operator** 并点 **Install**。
4. 选择一个 **Update Channel**、**Installation Mode** 和 **Approval Strategy**。
5. 点 **Subscribe**.
   在 **Installed Operators** 页面中，**OpenShift Container Storage Operator** 会出现在 **openshift-storage** 项目中，状态为 **Succeeded**。

1.5.1.2. 创建 Multi-Cloud Object Gateway 存储桶

您可以创建 Multi-Cloud Object Gateway (MCG) 存储桶的自定义资源 (CR) 。

流程

1. 登录到 OpenShift Container Platform 集群：

```
$ oc login
```

2. 使用以下内容创建 **NooBaa** CR 配置文件， **noobaa.yml**：

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

1 2 对于非常小的集群，您可以将 **cpu** 的值改为 **0.1**。

3. 创建 **NooBaa** 对象：

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

4. 使用以下内容创建 **BackingStore** CR 配置文件， **bs.yml**：

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

1 指定 PV 池中的卷数量。

2 指定卷的大小。

3 指定存储类。

5. 创建 **BackingStore** 对象：

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

6. 使用以下内容创建 **BucketClass** CR 配置文件，**bc.yml**：

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

7. 创建 **BucketClass** 对象：

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

8. 使用以下内容创建 **ObjectBucketClaim** CR 配置文件，**obc.yml**：

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

1 记录下在 CAM web 控制台中添加为复制存储库的存储桶的名称。

9. 创建 **ObjectBucketClaim** 对象：

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

10. 监控资源创建过程以验证 **ObjectBucketClaim** 的状态变为 **Bound**：

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

这个过程可能需要五到十分钟。

11. 获取并记录以下值，当您将复制存储库添加到 CAM web 控制台时需要这些值：

    - S3 端点：

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

    - S3 provider access key:

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

    - S3 provider secret access key:

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

1.5.2. 将 AWS S3 存储桶配置为复制存储库

您可以将 AWS S3 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 AWS CLI。
- 如果您使用快照复制方法：
  - 您必须有权访问 EC2 Elastic Block Storage (EBS)。
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 创建 AWS S3 存储桶：

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

   1 指定 S3 存储桶名称。

   2 指定 S3 存储桶区域，例如 **us-east-1**。

2. 创建 IAM 用户 **velero**：

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

3. 创建 EC2 EBS 快照策略：

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

4. 为一个或所有 S3 存储桶创建 AWS S3 访问策略：

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

1 2 要授予对单一 S3 存储桶的访问权限，请指定存储桶名称。要授予对所有 AWS S3 存储桶的访问权限，请指定 * 而不是存储桶名称：

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

5. 将 EC2 EBS 策略附加到 **velero**：

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

6. 将 AWS S3 策略附加到 **velero**：

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

7. 为 **velero** 创建访问密钥：

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

1 2 记录 **AWS_SECRET_ACCESS_KEY** 和 **AWS_ACCESS_KEY_ID** 以将 AWS 存储库添加到 CAM Web 控制台。

1.5.3. 将 Google Cloud Provider 存储桶配置为复制存储库

您可以将 Google Cloud Provider (GCP) 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 **gsutil**。
- 如果您使用快照复制方法：
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。

- 存储类必须与快照兼容。

流程

1. 运行 **gsutil init** 以登录：

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

Your current configuration has been set to: [default]

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

2. 设置 **BUCKET** 变量：

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

1 指定存储桶名称。

3. 创建存储桶：

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

4. 将 **PROJECT_ID** 变量设置为您的活跃项目：

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

5. 创建 **velero** 服务帐户：

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

6. 将 **SERVICE_ACCOUNT_EMAIL** 变量设置为服务帐户的电子邮件地址：

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

7. 向服务帐户授予权限：

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

gcloud iam roles create velero.server \
```

```
    --project $PROJECT_ID \
    --title "Velero Server" \
    --permissions "$(IFS=","; echo "${ROLE_PERMISSIONS[*]}")"

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

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

8. 将服务帐户的密钥保存到当前目录中的 **credentials-velero** 文件中：

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

1.5.4. 将 Microsoft Azure Blob 存储容器配置为复制存储库

您可以将 Microsoft Azure Blob 存储容器配置为复制存储库。

先决条件

- 您必须具有 Azure 存储帐户。
- 您必须安装了 Azure CLI。
- Azure Blob 存储容器必须可以被源和目标集群访问。
- 如果您使用快照复制方法：
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 设置 **AZURE_RESOURCE_GROUP** 变量：

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. 创建 Azure 资源组：

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

1 指定位置。

3. 设置 **AZURE_STORAGE_ACCOUNT_ID** 变量：

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. 创建 Azure 存储帐户：

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

5. 设置 **BLOB_CONTAINER** 变量：

```
$ BLOB_CONTAINER=velero
```

6. 创建 Azure Blob 存储容器：

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

7. 为 **velero** 创建服务主体和凭证：

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

8. 在 **credentials-velero** 文件中保存服务主体的凭证：

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

1.6. 使用 CAM WEB 控制台迁移应用程序

您可以通过在 CAM web 控制台中添加集群和复制存储库来迁移应用程序工作负载。然后，您可以创建并运行迁移计划。

如果集群或复制存储库有自签名证书保护，您可以创建 CA 证书捆绑包文件或禁用 SSL 验证。

1.6.1. 创建 CA 证书捆绑包文件

如果您使用自签名证书来保护集群或复制存储库的安全，则证书验证可能会失败，出错信息如下：**Certificate signed by unknown authority**。

您可以创建自定义 CA 证书捆绑包文件，并在添加集群或复制存储库时将其上传到 CAM web 控制台。

流程

从远程端点下载 CA 证书，并将其保存为 CA 捆绑包文件：

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

1. 指定端点的主机 FQDN 和端口，如 **api.my-cluster.example.com:6443**。
2. 指定 CA 捆绑包文件的名称。

1.6.2. 把集群添加到 CAM web 控制台中

您可以将源集群添加到 CAM web 控制台。

先决条件

如果要使用 Azure 快照复制数据：

- 在添加源集群时，您必须提供 Azure 资源组名称。
- 源和目标集群必须位于同一 Azure 资源组且位于同一位置。

流程

1. 登录到集群。
2. 获取服务帐户令牌：

```
$ oc sa get-token mig -n openshift-migration
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwi
a3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJtaWciLCJrdWJlcm5ldGVz
LmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoibWlnLXRva2VuLWs4dDJyIiwia3ViZXJuZ
XRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6Im1pZyIsImt1YmV
ybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImE1YjFiYWM
wLWMxYmYtMTFlOS05Y2NiLTAyOWRmODYwYjMwOCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlY
WNjb3VudDptaWc6bWlnIn0.xqeeAINK7UXpdRqAtOj70qhBJPeMwmgLomV9iFxr5RoqUgKchZ
RG2J2rkqmPm6vr7K-
cm7ibD1IBpdQJCcVDuoHYsFgV4mp9vgOfn9osSDp2TGikwNz4Az95e81xnjVUmzh-
NjDsEpw71DH92iHV_xt2sTwtzftS49LpPW2LjrV0evtNBP_t_RfskdArt5VSv25eORl7zScqfe1CiM
kcVbf2UqACQjo3LbkpfN26HAioO2oH0ECPiRzT0Xyh-KwFutJLS9Xgghyw-
LD9kPKcE_xbbJ9Y4Rqajh7WdPYuB0Jd9DPVrsImzK-F6cgHHYoZEv0SvLQi-
PO0rpDrcjOEQQ
```

3. 登录到 CAM web 控制台。
4. 在 **Clusters** 部分，点 **Add cluster**。
5. 填写以下字段：
   - **Cluster name**：可包括小写字母（**a-z**）和数字（**0-9**）。不能包含空格或国际字符。
   - **URL**：集群 API 服务器的 URL，如 **https://<master1.example.com>:8443**。
   - **Service account token**：从源集群获取的字符串。

- **Azure cluster**：可选。如果要使用 Azure 快照复制数据，请选择此项。
- **Azure resource group**：如果选中了 **Azure cluster**，则会出现此字段。
- 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 CA 捆绑包文件。

6. 点 **Add cluster**。
   集群会出现在 **Clusters** 部分。

1.6.3. 在 CAM web 控制台中添加复制程序库

您可以将对象存储桶作为复制存储库添加到 CAM web 控制台。

先决条件

- 您必须配置用于迁移数据的对象存储桶。

流程

1. 登录到 CAM web 控制台。
2. 在 **Replication repositories** 部分，点 **Add repository**。
3. 选择 **Storage provider type** 并填写以下字段：
   - **AWS** 适用于 S3、MCSG 和通用 S3 供应商：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **S3 bucket name**：指定您创建的 S3 存储桶的名称。
     - **S3 bucket region**：指定 S3 存储桶区域。AWS S3 **必填**。**Optional** 用于其他 S3 供应商。
     - **S3 端点**：指定 S3 服务的 URL，而不是存储桶，例如：**https://<s3-storage.apps.cluster.com>**。通用 S3 供应商**必填**。您必须使用 **https://** 前缀。
     - **S3 provider access key**：为 AWS 指定 **<AWS_SECRET_ACCESS_KEY>**，或者为 MCG 指定 S3 供应商访问密钥。
     - **S3 provider secret access key**：为 AWS 指定 **<AWS_ACCESS_KEY_ID>**，或者为 MCG 指定 S3 供应商 secret 访问密钥。
     - **Require SSL verification**：如果您使用的是通用 S3 供应商，则清除此复选框。
     - 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 Base64 编码的 CA 捆绑包文件。
   - **GCP**：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **GCP bucket name**：指定 GCP 存储桶的名称。
     - **GCP credential JSON blob**：在 **credentials-velero** 文件中指定字符串。
   - **Azure**：

- **Replication repository name**：指定 CAM web 控制台中的复制存储库。
- **Azure resource group**：指定 Azure Blob 存储的资源组。
- **Azure storage account name**：指定 Azure Blob 存储帐户名称
- **Azure credentials - INI file contents**：在 **credentials-velero** 文件中指定字符串。

4. 点 **Add repository** 并等待连接验证。
5. 点 **Close**。
   新存储库会出现在 **Replication repositories** 部分。

1.6.4. 为大型迁移修改迁移计划限制

您可以更改大型迁移的迁移计划限制。

重要

您需要首先在自己的环境对所做的更改进行测试，以避免迁移失败。

单个迁移计划有以下默认限制：

- 10 个命名空间
  如果超过这个限制，CAM web 控制台会显示一个 **Namespace limit exceeded**错误，您将无法创建迁移计划。
- 100 个 Pod
  如果超过 Pod 限制，CAM web 控制台会显示类似以下示例的警告信息: **Plan has been validated with warning condition(s).查看警告信息。pod limit: 100 exceeded, found: 104**。
- 100 个持久性卷（PV）
  如果超过持久性卷限制，则 CAM web 控制台会显示类似的警告信息。

流程

1. 编辑迁移控制器 CR：

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

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

2. 更新以下参数：

```
...
migration_controller: true

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

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

1.6.5. 在 CAM web 控制台中创建迁移计划

您可以在 CAM web 控制台中创建迁移计划。

先决条件

- CAM web 控制台必须包含以下内容：
  - 源集群
  - 目标集群，它会在 CAM 工具安装过程中自动添加
  - 复制软件仓库
- 源和目标集群必须可以通过网络相互访问，并可以访问复制存储库。
- 如果要使用快照复制数据，则源和目标集群必须在同一云供应商（AWS、GCP 或 Azure）以及同一区域中。

流程

1. 登录到 CAM web 控制台。
2. 在 **Plans** 部分，点 **Add Plan**。
3. 输入 **Plan name** 并点 **Next**。
   Plan name 最多可包含 253 个小写字母数字字符（**a-z，0-9**）。它不能包含空格或下划线（_）。
4. 选一个 **Source cluster**。
5. 选一个 **Target cluster**。
6. 选一个 **Replication repository**。
7. 选择要迁移的项目并点 **Next**。
8. 选择 **Copy** 或 **Move** PV：
   - **Copy** 将源集群的 PV 中的数据复制到复制存储库中，然后在目标集群中新创建的具有类似特征的 PV 上恢复它。
     可选： 您可以通过选择 **Verify copy**来使用文件系统的方法来验证复制的数据。这个选项为每个源文件生成 checksum 并在恢复后对其进行检查。这可能会大大降低性能。
   - **Move** 从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。
9. 点 **Next**。
10. 为 PV 选择 **Copy method**：
    - **Snapshot** 使用云供应商的快照功能备份和恢复磁盘。它比 **Filesystem** 快得多。

注意

存储和集群必须位于同一区域，存储类必须兼容。

- **Filesystem** 将源磁盘中的数据文件复制到新创建的目标磁盘。

11. 为 PV 选择一个 **Storage class**。
如果选择了 **Filesystem** 复制方法，您可以在迁移过程中更改存储类，例如：从 Red Hat Gluster Storage 或 NFS 存储改为 Red Hat Ceph Storage。

12. 点 **Next**。

13. 如果您要添加迁移 hook，请点击 **Add Hook** 并执行以下步骤：

    a. 指定存储桶的名称。

    b. 选择 **Ansible playbook** 来使用您自己的 playbook 或 **Custom container image** 来使用以其他语言编写的 hook。

    c. 点击 **Browse** 上传 playbook。

    d. 可选： 如果您未使用默认 Ansible 运行时镜像，请指定自定义 Ansible 镜像。

    e. 指定要运行 hook 的集群。

    f. 获取服务帐户名称。

    g. 指定命名空间。

    h. 选择您希望 hook 运行的迁移步骤：

       - PreBackup: 在源集群上启动备份前的任务
       - PostBackup: 在源集群中完成备份任务后
       - PreRestore: 在目标集群上启动恢复前的任务
       - PostRestore: 在目标集群中完成恢复后的任务

14. 点 **Add**。
您可以在迁移计划中添加最多四个 hook，将每个 hook 分配给不同的迁移步骤。

15. 点 **Finish**。

16. 点 **Close**。
迁移计划会出现在 **Plans** 部分。

1.6.6. 在 CAM web 控制台中运行迁移计划

您可以使用在 CAM web 控制台中创建的迁移计划来 stage 或迁移应用程序和数据。

先决条件

CAM web 控制台必须包含以下内容：

- 源集群

- 目标集群，它会在 CAM 工具安装过程中自动添加
- 复制软件仓库
- 有效的迁移计划

流程

1. 登录目标集群上的 CAM web 控制台。
2. 选择迁移计划。
3. 点 **Stage** 以在不停止应用程序的情况下，将数据从源集群复制到目标集群。
   您可以多次运行 **Stage** 以减少实际迁移时间。
4. 当准备好迁移应用程序工作负载时，点 **Migrate**。
   Migrate 在源集群中停止应用程序工作负载，并在目标集群中重新创建其资源。
5. 另外，还可以在 **Migrate** 窗口中选择 **Do not stop applications on the source cluster during migration**。
6. 点 **Migrate**。
7. 可选： 要停止迁移过，请点击 Options 菜单  并选择 **Cancel**。
8. 迁移完成后，在 OpenShift Container Platform web 控制台中确认已成功迁移了应用程序：
   a. 点 **Home → Projects**。
   b. 点迁移的项目查看其状态。
   c. 在 **Routes** 部分，点击 **Location** 验证应用程序是否正常运行。
   d. 点 **Workloads → Pods** 来验证 Pod 在迁移的命名空间中运行。
   e. 点 **Storage → Persistent volumes** 确认正确置备了被迁移的持久性卷。

1.7. 使用 CONTROL PLANE MIGRATION ASSISTANT (CPMA) 迁移 CONTROL PLANE 设置

Control Plane Migration Assistant (CPMA) 是一个基于 CLI 的工具，它可帮助您将 control plane 从 OpenShift Container Platform 3.7（或更新版本）迁移到 OpenShift Container Platform 4.3。CPMA 处理 OpenShift Container Platform 3 配置文件并生成自定义资源 (CR) 清单文件，这些文件由 OpenShift Container Platform 4.3 Operator 使用。

1.7.1. 安装 Control Plane Migration Assistant

您可以从红帽客户门户网站下载 Control Plane Migration Assistant (CPMA) 二进制文件，并在 Linux、MacOSX 或者 Windows 操作系统中安装它。

流程

1. 在 Red Hat 客户门户网站 中，导航至 **Downloads → Red Hat OpenShift Container Platform**。

2. 在 **Download Red Hat OpenShift Container Platform**页面中，从 **Product Variant** 列表中选择 **Red Hat OpenShift Container Platform**。
3. 从 **Version** 列表中选择 **CPMA 1.0 for RHEL 7**。这个二进制文件适用于 RHEL 7 和 RHEL 8。
4. 点 **Download Now** 为 Linux 或 MacOSX 下载 **cpma**，或为 Windows 下载 **cpma.exe**。
5. 对于 Linux 或 MacOSX，把文件保存在由 **$PATH** 定义的目录中；对于 Windows，把文件保存在由 **%PATH%** 定义的目录中。
6. 对于 Linux，把文件设置为可执行：

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

1.7.2. 使用 Control Plane Migration Assistant

Control Plane Migration Assistant (CPMA) 生成 CR 清单，由 OpenShift Container Platform 4.3 Operator 使用，并产生包括哪些 OpenShift Container Platform 3 的功能被完全支持、部分支持或根本不支持的报告。

CPMA 可在远程模式下运行，使用 SSH 从源集群中检索配置文件，也可以使用本地模式，使用源集群配置文件的本地副本获取配置文件。

先决条件

- 源集群必须是 OpenShift Container Platform 3.7 或更高版本。
- 必须将源集群更新至最新的同步版本。
- 必须在源集群中运行环境健康检查来确定没有诊断错误或警告。
- CPMA 二进制文件必须是可执行文件。
- 必须具有源集群的 **cluster-admin** 权限。

流程

1. 登陆到 OpenShift Container Platform 3 集群：

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

1 OpenShift Container Platform 3 主（master）节点。您必须登录到集群以接收 Kubernetes 和 OpenShift Container Platform API 的令牌。

2. 运行 CPMA。根据每个提示输入，如下例所示：

```
$ cpma --manifests=false 1
? Do you wish to save configuration for future use? true
? What will be the source for OCP3 config files? Remote host 2
? Path to crio config file /etc/crio/crio.conf
? Path to etcd config file /etc/etcd/etcd.conf
? Path to master config file /etc/origin/master/master-config.yaml
? Path to node config file /etc/origin/node/node-config.yaml
? Path to registries config file /etc/containers/registries.conf
```

```
? Do wish to find source cluster using KUBECONFIG or prompt it? KUBECONFIG
? Select cluster obtained from KUBECONFIG contexts master1-example-com:443
? Select master node master1.example.com
? SSH login root (3)
? SSH Port 22
? Path to private SSH key /home/user/.ssh/openshift_key
? Path to application data, skip to use current directory .
INFO[29 Aug 19 00:07 UTC] Starting manifest and report generation
INFO[29 Aug 19 00:07 UTC] Transform:Starting for - API
INFO[29 Aug 19 00:07 UTC] APITransform::Extract
INFO[29 Aug 19 00:07 UTC] APITransform::Transform:Reports
INFO[29 Aug 19 00:07 UTC] Transform:Starting for - Cluster
INFO[29 Aug 19 00:08 UTC] ClusterTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportQuotas
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportPVs
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportNamespaces
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportNodes
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportRBAC
INFO[29 Aug 19 00:08 UTC] ClusterReport::ReportStorageClasses
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Crio
INFO[29 Aug 19 00:08 UTC] CrioTransform::Extract
WARN[29 Aug 19 00:08 UTC] Skipping Crio: No configuration file available
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Docker
INFO[29 Aug 19 00:08 UTC] DockerTransform::Extract
INFO[29 Aug 19 00:08 UTC] DockerTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - ETCD
INFO[29 Aug 19 00:08 UTC] ETCDTransform::Extract
INFO[29 Aug 19 00:08 UTC] ETCDTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - OAuth
INFO[29 Aug 19 00:08 UTC] OAuthTransform::Extract
INFO[29 Aug 19 00:08 UTC] OAuthTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - SDN
INFO[29 Aug 19 00:08 UTC] SDNTransform::Extract
INFO[29 Aug 19 00:08 UTC] SDNTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Image
INFO[29 Aug 19 00:08 UTC] ImageTransform::Extract
INFO[29 Aug 19 00:08 UTC] ImageTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Transform:Starting for - Project
INFO[29 Aug 19 00:08 UTC] ProjectTransform::Extract
INFO[29 Aug 19 00:08 UTC] ProjectTransform::Transform:Reports
INFO[29 Aug 19 00:08 UTC] Flushing reports to disk
INFO[29 Aug 19 00:08 UTC] Report:Added: report.json
INFO[29 Aug 19 00:08 UTC] Report:Added: report.html
INFO[29 Aug 19 00:08 UTC] Successfully finished transformations
```

(1) **--manifests=false**：不生成 CR 清单

(2) **Remote host**：远程模式

(3) **SSH login**：为了访问配置文件，SSH 用户必须在 OpenShift Container Platform 3 集群上具有 **sudo** 权限。

如果您没有指定输出目录，CPMA 会在当前目录中创建以下文件和目录：

- **cpma.yaml** 文件：运行 CPMA 时提供的配置选项

- **master1.example.com/**: master 节点中的配置文件
- **report.json**: JSON 格式的报告
- **report.html**: HTML 格式的报告

3. 在浏览器中打开 **report.html** 文件来查看 CPMA 报告。
4. 如果生成 CR 清单，将 CR 清单应用到 OpenShift Container Platform 4.3 集群，如下例所示：

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

1.8. 故障排除

您可以查看迁移自定义资源 (CR)，并下载日志来排除迁移失败的问题。

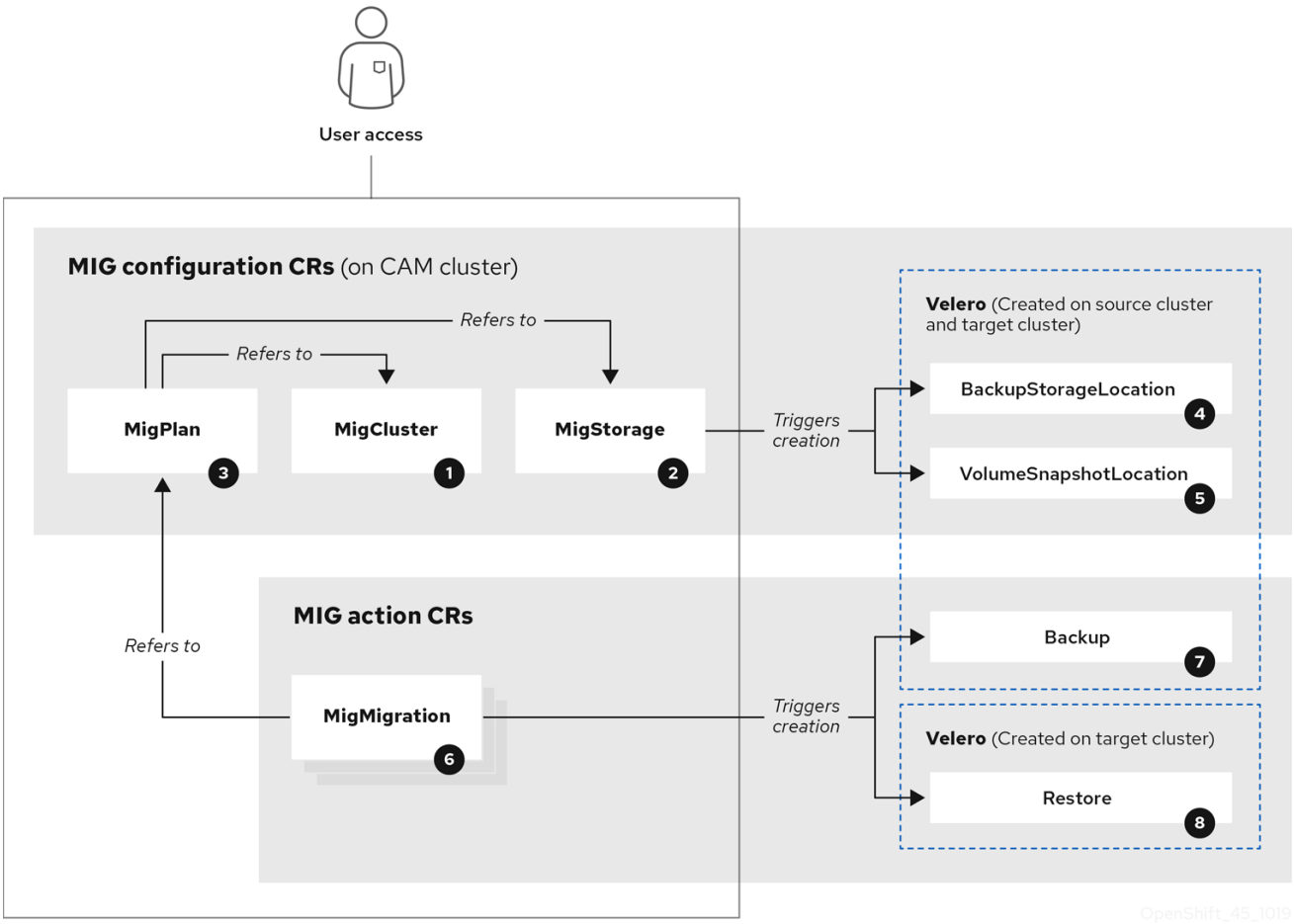
如果应用程序在迁移失败时停止，您必须手动回滚，以防止数据崩溃。

注意

如果应用程序在迁移过程中没有停止，则不需要手动回滚，因为原始应用程序仍然在源集群中运行。

1.8.1. 查看迁移自定义资源

集群应用程序迁移 (CAM) 工具会创建以下 CR：

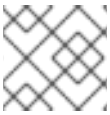


1 MigCluster （配置，CAM 集群）：集群定义

2 MigStorage （配置，CAM 集群）：存储定义

3 MigPlan （配置，CAM 集群）：迁移计划

MigPlan CR 描述了要迁移的源和目标集群、存储库和命名空间。它与 0 个、1 个或多个 MigMigration CR 关联。



注意

删除 MigPlan CR 会删除关联的 MigMigration CR。

4 BackupStorageLocation （配置，CAM 集群）：Velero 备份对象的位置

5 VolumeSnapshotLocation （配置，CAM 集群）：Velero 卷快照的位置

6 MigMigration（操作，CAM 集群）：Migration，在迁移期间创建

在每次进行 stage 或迁移数据时都会创建一个 MigMigration CR。每个 MigMigration CR 都会与一个 MigPlan CR 关联。

7 Backup（操作，源集群）：当运行迁移计划时，MigMigration CR 在每个源集群上创建两个 Velero 备份 CR：

- 备份 CR #1 用于Kubernetes 对象
- 备份 CR #2 用于 PV 数据

8 Restore （操作，目标集群）：在运行迁移计划时，MigMigration CR 在目标集群上创建两个 Velero 恢复 CR：

- 恢复 CR #1（使用备份 CR #2）用于 PV 数据
- 恢复 CR #2（使用备份 CR #1）用于 Kubernetes 对象

流程

1. 获取 CR 名称：

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

1 指定迁移 CR，如 **migmigration**。

输出结果类似以下：

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

2. 查看 CR：

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

输出结果类似以下示例。

MigMigration 示例

```
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>
```

Velero 备份 CR #2 示例（PV 数据）

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.105.179:5000
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-44dd3bd5-c9f8-11e9-95ad-0205fe66cbb6
  creationTimestamp: "2019-08-29T01:03:15Z"
  generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
  generation: 1
```

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

Velero 恢复 CR #2 示例（Kubernetes 资源）

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  annotations:
    openshift.io/migrate-copy-phase: final
    openshift.io/migrate-quiesce-pods: "true"
    openshift.io/migration-registry: 172.30.90.187:5000
```

```
    openshift.io/migration-registry-dir: /socks-shop-mig-plan-registry-36f54ca7-c925-11e9-825a-
06fa9fb68c88
  creationTimestamp: "2019-08-28T00:09:49Z"
  generateName: e13a1b60-c927-11e9-9555-d129df7f3b96-
  generation: 3
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: e18252c9-c927-11e9-825a-06fa9fb68c88
    migration-final-restore: e18252c9-c927-11e9-825a-06fa9fb68c88
  name: e13a1b60-c927-11e9-9555-d129df7f3b96-gb8nx
  namespace: openshift-migration
  resourceVersion: "82329"
  selfLink: /apis/velero.io/v1/namespaces/openshift-migration/restores/e13a1b60-c927-11e9-9555-
d129df7f3b96-gb8nx
  uid: 26983ec0-c928-11e9-825a-06fa9fb68c88
spec:
  backupName: e13a1b60-c927-11e9-9555-d129df7f3b96-sz24f
  excludedNamespaces: null
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  includedNamespaces: null
  includedResources: null
  namespaceMapping: null
  restorePVs: true
status:
  errors: 0
  failureReason: ""
  phase: Completed
  validationErrors: null
  warnings: 15
```

1.8.2. 下载迁移日志

您可以在 CAM web 控制台中下载 Velero 、 Restic 和 Migration controller 日志，以排除出现故障的迁移问题。

流程

1. 登录到 CAM 控制台。
2. 点击 **Plans** 查看迁移计划列表。
3. 点击特定迁移计划  的 **Options** 菜单并选择 **Logs**。
4. 点 **Download Logs** 为所有集群下载迁移控制器、Velero 和 Restic 的日志。
5. 要下载特定的日志：
    a. 指定日志选项：

- **Cluster**：选择源、目标或 CAM 主机集群。
- **Log source**：选择 **Velero**、**Restic** 或 **Controller**。
- **Pod source**：选择 Pod 名称，例如：**controller-manager-78c469849c-v6wcf** 此时会显示所选日志。

  您可以通过更改您的选择来清除日志选择设置。

b. 点 **Download Selected** 下载所选日志。

另外，您可以使用 CLI 访问日志，如下例所示：

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

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

1.8.3. 更新已弃用的 API GroupVersionKinds

在 OpenShift Container Platform 4.3 中，一些在 OpenShift Container Platform 3.x 中使用的 API **GroupVersionKinds** (GVKs) 已被弃用。

如果您的源集群使用已弃用的 GVK，在创建迁移计划时会显示以下警告信息：**Some namespaces contain GVKs incompatible with destination cluster**。您可以点击 **See details** 来查看命名空间和不兼容的 GVK。

注意

这个警告并不会阻止迁移继续进行。

在迁移过程中，已弃用的 GVK 保存在用于 Kubernetes 对象的 Velero 备份自定义资源 (CR) #1 中。您可以下载备份 CR，提取已弃用的 GVK **yaml** 文件，并使用 **oc convert** 命令更新它们。然后，您在目标集群中创建更新的 GVK。

流程

1. 运行迁移计划
2. 查看 MigPlan CR:

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

(1) 指定迁移计划的名称。

输出结果类似以下：

```
metadata:
  ...
  uid: 79509e05-61d6-11e9-bc55-02ce4781844a (1)
status:
  ...
  conditions:
```

```
  - category: Warn
    lastTransitionTime: 2020-04-30T17:16:23Z
    message: 'Some namespaces contain GVKs incompatible with destination cluster.
      See: `incompatibleNamespaces` for details'
    status: "True"
    type: GVKsIncompatible
  incompatibleNamespaces:
  - gvks:
    - group: batch
      kind: cronjobs 2
      version: v2alpha1
    - group: batch
      kind: scheduledjobs 3
      version: v2alpha1
```

1 记录 MigPlan UID。

2 3 记录已弃用的 GVK。

3. 获取与 MigPlan UID 关联的 MigMigration 名称：

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

1 指定 MigPlan UID。

4. 获取与 MigMigration 名称关联的 MigMigration UID:

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

1 指定 MigMigration 名称。

5. 获取与 MigMigration UID 关联的 Velero 备份名称：

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

1 指定 MigMigration UID。

6. 在您的本地机器中下载 Velero 备份内容：

- 对于 AWS S3:

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

1 指定存储桶、备份名称和您的本地备份目录名称。

- 对于GCP：

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

1 指定存储桶、备份名称和您的本地备份目录名称。

- 对于 Azure:

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

1 指定备份名称和您的本地备份目录名称。

7. 解压 Velero 备份归档文件：

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

8. 在每个已弃用的 GVK 中以离线模式运行 **oc convert**:

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

1 指定已弃用的 GVK。

9. 在目标集群中创建转换的 GVK:

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

1 指定转换的 GVK。

1.8.4. 错误信息

1.8.4.1. Velero Pod 日志中的 Restic 超时错误消息

如果因为 Restic 超时造成迁移失败，以下出错信息会出现在 Velero Pod 日志中：

```
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 的默认值为一小时。您可以为大型迁移增加这个参数值，请注意，高的值可能会延迟返回出错信息。

流程

1. 在 OpenShift Container Platform web 控制台中导航至 **Operators → Installed Operators**。
2. 点 **Cluster Application Migration Operator**。

3. 在 **MigrationController** 标签页中点 **migration-controller**。
4. 在 **YAML** 标签页中，更新以下参数值：

```
spec:
  restic_timeout: 1h (1)
```

(1) 有效单元是 **h** （小时）、**m** （分钟）和 **s** （秒），例如 **3h30m15s**。

5. 点 **Save**。

1.8.4.2. MigMigration Custom Resource（CR）中的 ResticVerifyErrors

如果迁移使用文件系统数据复制方法的 PV 时数据验证失败，在 MigMigration 自定义资源 (CR) 中会出现以下错误：

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

(1) 错误消息标识了 Restore CR 名称。

(2) **ResticErrors** 也会出现。**ResticErrors** 是一个包括验证错误的一般错误警告。

注意

数据验证错误不会导致迁移过程失败。

您可以检查目标集群的 Restore CR，以识别数据验证错误的来源。

流程

1. 登录到目标集群。
2. 查看 Restore CR:

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

输出通过 **PodVolumeRestore** 错误来指定 PV：

```
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. 查看 **PodVolumeRestore** CR:

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

输出中标识了记录错误的 Restic Pod:

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

4. 查看 Restic Pod 日志：

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

1.8.5. 手动回滚迁移

如果您的应用程序在迁移失败时停止，您必须手动回滚，以防止 PV 中的数据被破坏。

如果应用程序在迁移过程中没有停止，则不需要进行手动回滚，因为原始应用程序仍然在源集群中运行。

流程

1. 在目标集群中，切换到迁移的项目：

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

2. 获取部署的资源：

```
$ oc get all
```

3. 删除部署的资源以确保应用程序没有在目标集群中运行，并访问 PVC 上的数据：

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

4. 要停止 DaemonSet 而不删除它，在 YAML 文件中更新 **nodeSelector** ：

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: hello-daemonset
spec:
  selector:
     matchLabels:
       name: hello-daemonset
```

```
  template:
    metadata:
      labels:
        name: hello-daemonset
    spec:
      nodeSelector:
        role: worker 1
```

1 指定一个没有存在于任何节点上的**nodeSelector** 值。

5. 更新每个 PV 的重新声明策略，以便删除不必要的数据。在迁移过程中，绑定 PV 的重新声明策略是**reclaim**，以确保应用程序从源集群中被删除时不会丢失数据。您可以在回滚过程中删除这些 PV。

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

1 指定 **Recycle** 或 **Delete**。

6. 在源集群中，切换到迁移的项目：

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

7. 获取项目部署的资源：

```
$ oc get all
```

8. 启动每个部署资源的一个或多个副本：

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

9. 如果在操作中被更改了，把 DaemonSet 的 **nodeSelector** 改回其原始值。

1.8.6. 为客户支持问题单收集数据

如果创建一个客户支持问题单，您可以使用 **openshift-migration-must-gather-rhel8** 镜像的 **must-gather** 工具来收集与您的集群相关的信息，并把这些信息上传到红帽客户门户网站。

openshift-migration-must-gather-rhel8 镜像会收集默认的 **must-gather** 镜像不收集的日志和 CR 数据。

流程

1. 进入要存储 **must-gather** 数据的目录。
2. 运行 **oc adm must-gather** 命令：

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

must-gather 工具程序收集集群数据，并把它保存在 **must-gather.local.<uid>** 目录中。

3. 从 **must-gather** 数据中删除验证密钥和其他敏感信息。
4. 创建一个包含 **must-gather.local.<uid>** 目录内容的归档文件：

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

在红帽客户门户中，为您的问题单附上这个压缩文件。

1.8.7. 已知问题

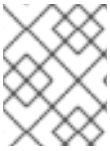
这个版本有以下已知问题：

- 在迁移过程中，集群应用程序迁移 (CAM) 工具会保留以下命名空间注解：
  - **openshift.io/sa.scc.mcs**
  - **openshift.io/sa.scc.supplemental-groups**
  - **openshift.io/sa.scc.uid-range**
    这些注解会保留 UID 范围，确保容器在目标集群中保留其文件系统权限。这可能会存在一定的风险。因为迁移的 UID 可能已存在于目标集群的现有或将来的命名空间中。
    (BZ#1748440)
- 如果一个 AWS 存储桶被添加到 CAM web 控制台，然后将其删除，则其状态会保持为 **True**，这是因为 MigStorage CR 没有被更新。(BZ#1738564)
- 大多数集群范围的资源尚未由 CAM 工具处理。如果应用程序需要集群范围的资源，则可能需要在目标集群上手动创建。
- 如果迁移失败，则迁移计划不会为静默的 pod 保留自定义 PV 设置。您必须手动回滚，删除迁移计划，并使用 PV 设置创建新的迁移计划。(BZ#1784899)
- 如果因为 Restic 超时造成大型迁移失败，您可以提高 Migration controller CR 中的 **restic_timeout** 参数值。
- 如果您选择了为使用文件系统复制方法迁移的 PV 数据进行验证的选项，则性能会非常慢。Velero 为每个文件生成一个 checksum，并在恢复该文件时对其进行检查。
- 在当前发行版本 (CAM 1.2) 中，您无法从 OpenShift Container Platform 3.7 迁移到 4.4，因为某些被源集群使用的 API **GroupVersionKinds** (GVKs) 已被弃用。您可以在迁移后手动更新 GVK。(BZ#1817251)
- 如果您无法在 OpenShift Container Platform 3 集群上安装 CAM 1.2，请下载当前的 **operator.yml** 文件，这可以解决这个问题。(BZ#1843059)

第 2 章 从 OPENSHIFT CONTAINER PLATFORM 4.1 进行迁移

2.1. 迁移工具和先决条件

您可以使用集群应用程序迁移 (CAM) 工具将应用程序工作负载从 OpenShift Container Platform 4.1 迁移到 4.3。使用 CAM 工具，您可以控制迁移并最小化应用程序的停机时间。



注意

只要正确配置了源和目标集群，就可以在相同版本的 OpenShift Container Platform 集群间进行迁移（如从 4.1 迁移到 4.1）。

CAM 工具的 web 控制台和 API，基于 Kubernetes 自定义资源，您可以按照命名空间迁移有状态及无状态的应用程序工作负载。

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。

2.1.1. 迁移先决条件

- 必须将源集群升级到最新的 z-stream 版本。
- 需要在所有集群中都有 **cluster-admin** 权限。
- 源和目标集群必须有对复制存储库的不受限制的网络访问权限。
- 安装 Migration controller 的集群必须具有对其他集群的不受限制的访问权限。
- 如果应用程序使用 **openshift** 命名空间中的镜像，则目标集群中必须有所需的镜像版本。如果没有所需的镜像，您必须更新 **imagestreamtags** 的引用以使用与应用程序兼容的可用版本。如果无法更新 **imagestreamtags**，您可以手动将相关的镜像上传到应用程序命名空间中，并更新应用程序以引用它们。

以下 **imagestreamtags** 已从 OpenShift Container Platform 4.2 中 *删除*：

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

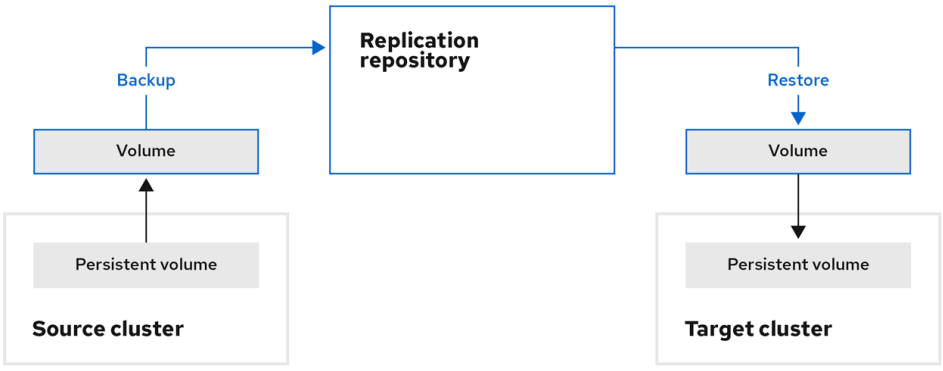
- **postgresql:9.2, postgresql:9.4, postgresql:9.5**
- **python:3.3、python:3.4**
- **ruby:2.0、ruby:2.2**

2.1.2. 关于集群应用程序迁移（CAM）工具

集群应用程序迁移 (CAM) 工具可让您使用 CAM web 控制台或 Kubernetes API 将 OpenShift Container Platform 源集群中的 Kubernetes 资源、持久性卷数据和内部容器镜像迁移到 OpenShift Container Platform 4.3 目标集群。

使用 CAM web 控制台迁移应用程序涉及以下步骤：

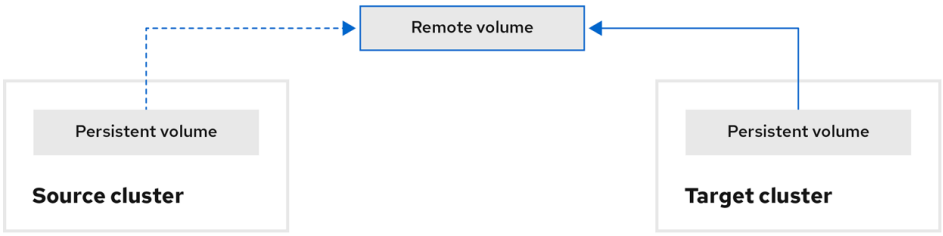
1. 在所有集群中安装 Cluster Application Migration Operator
   您可以在有限的或没有互联网访问的受限环境中安装 Cluster Application Migration Operator。源和目标集群必须可以在相互间进行访问，而需要可以访问 registry 的镜像（mirror）.
2. 配置复制存储库，这是 CAM 工具用来迁移数据的中间对象存储
   源和目标集群必须有对复制存储库的不受限制的网络访问权限。在受限环境中，您可以使用内部托管的 S3 存储存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。
3. 在 CAM web 控制台中添加源集群
4. 在 CAM web 控制台中添加复制存储库
5. 创建迁移计划，包含以下数据迁移选项之一：
   - **Copy**：CAM 工具将数据从源集群复制到复制存储库，再从复制存储库把数据复制到目标集群。



   - **Move**：CAM 工具从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。

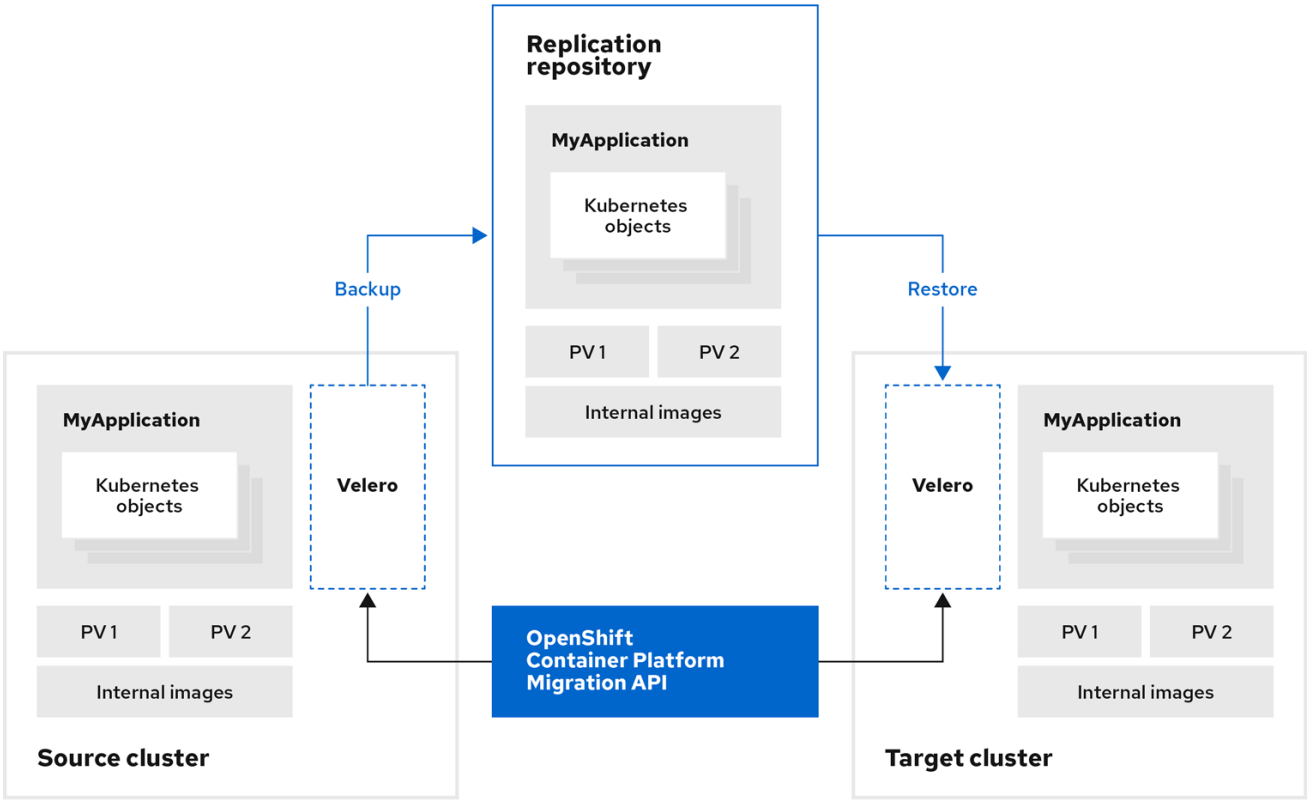
注意

虽然复制存储库没有出现在此图表中，但实际迁移过程需要它。



6. 运行迁移计划，使用以下选项之一：

- **Stage** （可选）在不停止应用程序的情况下将数据复制到目标集群。
  Stage 可以多次运行，以便在迁移前将大多数数据复制到目标。这样可最小化实际迁移时间和应用程序停机时间。
- **Migrate** 在源集群中停止应用程序，并在目标集群中重新创建其资源。您可以选择在不停止应用程序的情况下迁移工作负载。



2.1.3. 关于数据复制方法

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

2.1.3.1. 文件系统复制方法

CAM 工具将数据文件从源集群复制到复制存储库，并从那里复制到目标集群。

表 2.1. 文件系统复制方法概述

优点	限制：
• 集群可以有不同的存储类 • 所有 S3 存储供应商均支持 • 使用 checksum 验证数据（可选）	• 比快照复制方法慢 • 可选的数据校验可能会显著降低性能

2.1.3.2. 快照复制方法

CAM 工具将源集群的数据快照复制到云供应商的对象存储，后者配置为复制存储库。数据在目标集群上恢复。

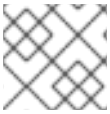
AWS、Google Cloud Provider 和 Microsoft Azure 支持快照复制方法。

表 2.2. 快照复制方法概述

优点	限制：
• 比文件系统复制方法快	• 云供应商必须支持快照。 • 集群必须位于相同的云供应商。 • 集群必须位于同一位置或区域。 • 集群必须具有相同的存储类。 • 存储类必须与快照兼容。

2.1.4. 关于迁移 hook

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。



注意

如果您不想使用 Ansible playbook，您可以创建自定义容器镜像并将其添加到迁移计划中。

迁移 hook 执行的任务包括自定义应用程序默认、手动迁移不受支持的数据类型以及在迁移后更新应用程序。

在源集群或目标集群中以以下迁移步骤之一运行单个迁移 hook:

- PreBackup: 在源集群上启动备份前的任务
- PostBackup: 在源集群中完成备份任务后
- PreRestore: 在目标集群上启动恢复前的任务
- PostRestore: 在目标集群中完成恢复后的任务
  您可以为每个迁移步骤分配一个 hook，单个迁移计划最多可分配四个 hook。

默认 **hook-runner** 镜像为 **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**。该镜像基于 Ansible Runner，并包括用于 Ansible Kubernetes 资源的 **python-openshift** 以及更新的 **oc** 二进制文件。您还可以使用其他 Ansible 模块或工具创建自己的 hook 镜像。

Ansible playbook 作为一个 ConfigMap 挂载到 hook 容器上。hook 容器在具有指定服务帐户和命名空间的集群中作为一个作业（job）运行。作业运行（即使初始 Pod 被驱除或终止），直到达到默认的 **backoffLimit** (**6**)或成功完成为止。

2.2. 部署集群应用程序迁移（CAM）工具

您可以在 OpenShift Container Platform 4.3 目标集群和 4.1 源集群中安装 Cluster Application Migration Operator。默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

注意

可选：您可以配置 Cluster Application Migration Operator，以便在 OpenShift Container Platform 3 集群或远程集群中安装 CAM 工具。

在受限环境中，您可以从本地镜像 registry 安装 Cluster Application Migration Operator。

在集群中安装 Cluster Application Migration Operator 后,您可以启动 CAM 工具。

2.2.1. 安装 Cluster Application Migration Operator

您可以使用操作生命周期管理器（OLM）在 OpenShift Container Platform 4.3 目标集群和 OpenShift Container Platform 4.1 源集群上安装 Cluster Application Migration Operator。

2.2.1.1. 在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。

8. 点击 **Create**。
9. 点 **Workloads → Pod** 来验证 Controller Manager 、 Migration UI 、 Restic 和 Velero Pod 是否正在运行。

2.2.1.2. 在 OpenShift Container Platform 4.1 源集群上安装 Cluster Application Migration Operator

您可以使用 OLM 在 OpenShift Container Platform 4 源集群上安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Catalog → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 更新 **migration_controller** 和 **migration_ui** 参数，并在 **spec** 中添加 **deprecated_cors_configuration** 参数：

```
spec:
  ...
  migration_controller: false
  migration_ui: false
  ...
  deprecated_cors_configuration: true
```

9. 点击 **Create**。
10. 点 **Workloads → Pod** 来验证 Restic 和 Velero Pod 是否正在运行。

2.2.2. 在受限环境中安装 Cluster Application Migration Operator

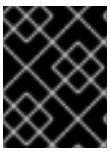
您可以为 OpenShift Container Platform 4 构建自定义 Operator 目录镜像，将其推送到本地镜像 registry，并配置 Operator Lifecycle Manager 从本地 registry 安装 Cluster Application Migration Operator。

其他资源

- 在受限网络中使用 Operator Lifecycle Manager

2.2.2.1. 构建 Operator 目录镜像

集群管理员可以构建自定义 Operator 目录镜像，供 Operator Lifecycle Manager (OLM) 使用，并将镜像推送到支持 Docker v2-2 的容器镜像 registry。对于受限网络中的集群，此 registry 可以是集群有网络访问权限的 registry，如在受限网络安装过程中创建的镜像 registry。



重要

OpenShift Container Platform 集群的内部 registry 不能用作目标 registry，因为它不支持没有标签的推送，而在镜像（mirrir）过程中需要这个功能。

在这一示例中，流程假定在使用镜像 registry 时可访问您的网络以及互联网。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- **oc** 版本 4.3.5+
- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

- 如果您正在使用 quay.io 帐户可访问的私有命名空间，您必须设置 Quay 身份验证令牌。使用您的 quay.io 凭证对登录 API 发出请求，从而设置用于 **--auth-token** 标志的 **AUTH_TOKEN** 环境变量：

```
$ AUTH_TOKEN=$(curl -sH "Content-Type: application/json" \
    -XPOST https://quay.io/cnr/api/v1/users/login -d '
    {
        "user": {
            "username": """<quay_username>""",
            "password": """<quay_password>"""
        }
    }' | jq -r '.token')
```

流程

1. 在没有网络访问限制的工作站中，与目标镜像（mirror） registry 进行身份验证：

```
$ podman login <registry_host_name>
```

还可使用 **registry.redhat.io** 验证，以便在构建期间拉取基础镜像：

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

2. 根据 quay.io 中的 **redhat-operators** 目录构建目录镜像，进行标记并将其推送到您的镜像 registry：

```
$ oc adm catalog build \
    --appregistry-org redhat-operators \ 1
    --from=registry.redhat.io/openshift4/ose-operator-registry:v4.3 \ 2
    --filter-by-os="linux/amd64" \ 3
    --to=<registry_host_name>:<port>/olm/redhat-operators:v1 \ 4
    [-a ${REG_CREDS}] \ 5
    [--insecure] \ 6
    [--auth-token "${AUTH_TOKEN}"] 7

INFO[0013] loading Bundles
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
...
Pushed sha256:f73d42950021f9240389f99ddc5b0c7f1b533c054ba344654ff1edaf6bf827e3
to example_registry:5000/olm/redhat-operators:v1
```

1. 从 App Registry 实例中拉取的机构（命名空间）。
2. 使用与目标 OpenShift Container Platform 集群主版本和次版本匹配的标签，将 **--from** 设置为 **ose-operator-registry** 基础镜像。
3. 将 **--filter-by-os** 设置为用于基本镜像的操作系统和架构，该镜像必须与目标 OpenShift Container Platform 集群匹配。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
4. 为您的目录镜像命名并包含标签，例如: **v1**。
5. 可选：如果需要，指定 registry 凭证文件的位置。
6. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。
7. 可选：如果使用其他不公开的应用程序 registry 目录，则需要指定 Quay 身份验证令牌。

有时红帽目录中会意外引入无效的清单；当发生这种情况时，您可能会看到一些错误：

```
...
INFO[0014] directory
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
file=4.2 load=package
W1114 19:42:37.876180   34665 builder.go:141] error building database: error loading
package into db: fuse-camel-k-operator.v7.5.0 specifies replacement that couldn't be found
Uploading ... 244.9kB/s
```

这些错误通常不是致命的，如果所提及 Operator 软件包不包含您计划安装的 Operator 或其依赖项，则可以忽略它们。

2.2.2.2. 针对受限网络配置 OperatorHub

集群管理员可以使用自定义 Operator 目录镜像将 OLM 和 OperatorHub 配置为在受限网络环境中使用本地内容。本例中的流程使用之前构建并推送到受支持的 registry 的自定义 **redhat-operators** 目录镜像。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- 推送到受支持的 registry 的自定义 Operator 目录镜像

- **oc** 版本 4.3.5+
- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

流程

1. 通过在 spec 中添加 **disableAllDefaultSources: true** 来禁用默认 OperatorSource：

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

该操作将禁用在 OpenShift Container Platform 安装期间默认配置的默认 OperatorSource。

2. **oc adm catalog mirror** 命令提取自定义 Operator catalog 镜像的内容，以生成镜像（mirror）所需的清单：您可以选择：

   - 允许该命令的默认行为在生成清单后自动将所有镜像内容镜像到您的镜像 registry 中，或者
   - 添加 **--manifests-only** 标志来只生成镜像所需的清单，但并不实际将镜像内容镜像到 registry。这对检查哪些要镜像（mirror）非常有用。如果您只需要部分内容的话，可以对映射列表做出任何修改。然后，您可以使用该文件与 **oc image mirror** 命令一起，在以后的步骤中镜像修改的镜像列表。

在没有网络访问限制的工作站中，运行以下命令：

```
$ oc adm catalog mirror \
    <registry_host_name>:<port>/olm/redhat-operators:v1 \ 1
    <registry_host_name>:<port> \
    [-a ${REG_CREDS}] \ 2
    [--insecure] \ 3
    [--filter-by-os="<os>/<arch>"] \ 4
    [--manifests-only] 5
```

1. 指定 Operator 目录镜像。
2. 可选：如果需要，指定 registry 凭证文件的位置。
3. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。
4. 可选： 因为目录可能会引用支持多个架构和操作系统的镜像，您可以根据架构和操作系统进行过滤来只对匹配的镜像进行镜像（mirror）。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
5. 可选：只生成镜像所需的清单，但并不实际将镜像内容镜像到 registry。

输出示例

```
using database path mapping: /:/tmp/190214037
wrote database to /tmp/190214037
using database at: /tmp/190214037/bundles.db 1
...
```

1 命令生成的临时数据库。

在运行命令后，会在当前目录中生成 **<image_name>-manifests/** 目录以及以下文件：

- 用来定义 ImageContentSourcePolicy 对象的 **imageContentSourcePolicy.yaml**，它可以将节点配置为在 Operator 清单中存储的镜像（image）引用和镜像 (mirror) 的 registry 间进行转换。
- **mapping.txt** 文件，在其中包含所有源镜像，并将它们映射到目标 registry。此文件与 **oc image mirror** 命令兼容，可用于进一步自定义镜像（mirror）配置。

3. 如果您在上一步中使用 **--manifests-only** 标志，并只想镜像部分内容：

a. 将 **mapping.txt** 文件中的镜像列表改为您的规格。如果您不确定要镜像的镜像子集的名称和版本，请使用以下步骤查找：

i. 对 **oc adm catalog mirror** 命令生成的临时数据库运行 **sqlite3** 工具，以检索与一般搜索查询匹配的镜像列表。输出以后如何编辑 **mapping.txt** 文件的帮助信息。
例如，要检索与字符串 **clusterlogging.4.3** 类似的镜像列表：

```
$ echo "select * from related_image \
    where operatorbundle_name like 'clusterlogging.4.3%';" \
    | sqlite3 -line /tmp/190214037/bundles.db 1
```

1 请参阅 **oc adm catalog mirror** 命令的输出结果来查找数据库文件的路径。

输出示例

```
image = registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0

image = registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf75
9c506
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0
...
```

ii. 使用上一步的结果来编辑 **mapping.txt** 文件，使其只包含您要镜像的镜像子集。
例如，您可以使用前面示例输出中的 **image** 值来找出您的 **mapping.txt** 文件中存在以下匹配行：

mapping.txt 中的匹配镜像映射

```
registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61=<registry_host_name>:<port>/openshift4-ose-logging-kibana5:a767c8f0
```

```
registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf759c506=<registry_host_name>:<port>/openshift4-ose-oauth-proxy:3754ea2b
```

在这个示例中，如果您只想镜像这些 image，可以在 **mapping.txt** 文件中删除所有其他条目，仅保留上述两行。

b. 在您的没有网络访问限制的工作站中，使用您修改的 **mapping.txt** 文件，使用 **oc image mirror** 命令将镜像镜像到 registry:

```
$ oc image mirror \
    [-a ${REG_CREDS}] \
    -f ./redhat-operators-manifests/mapping.txt
```

4. 应用 ImageContentSourcePolicy:

```
$ oc apply -f ./redhat-operators-manifests/imageContentSourcePolicy.yaml
```

5. 创建一个 CatalogSource 对象来引用您的目录镜像。

a. 按照您的规格修改以下内容，并将它保存为 **catalogsource.yaml** 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: <registry_host_name>:<port>/olm/redhat-operators:v1 (1)
  displayName: My Operator Catalog
  publisher: grpc
```

(1) 指定您的自定义 Operator 目录镜像。

b. 使用该文件创建 CatalogSource 对象：

```
$ oc create -f catalogsource.yaml
```

6. 确定成功创建以下资源。

a. 检查 Pod:

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

输出示例

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

b. 检查 CatalogSource:

```
$ oc get catalogsource -n openshift-marketplace
```

输出示例

```
NAME                  DISPLAY               TYPE PUBLISHER  AGE
my-operator-catalog   My Operator Catalog   grpc           5s
```

c. 检查 PackageManifest:

```
$ oc get packagemanifest -n openshift-marketplace
```

输出示例

```
NAME    CATALOG             AGE
etcd    My Operator Catalog  34s
```

现在，您可在受限网络 OpenShift Container Platform 集群的 web 控制台中，通过 **OperatorHub** 安装 Operator。

2.2.2.3. 在一个受限的环境中的 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

先决条件

- 已创建自定义 Operator 目录并将其推送到 registry 的镜像。
- 已将 OLM 配置为从 registry 镜像来安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 点击 **Create**。

9. 点 **Workloads → Pod** 来验证 Controller Manager 、 Migration UI 、 Restic 和 Velero Pod 是否正在运行。

2.2.2.4. 在一个受限的环境中的 OpenShift Container Platform 4.1 源集群上安装 Cluster Application Migration Operator

您可以使用 OLM 在 OpenShift Container Platform 4 源集群上安装 Cluster Application Migration Operator。

先决条件

- 已创建自定义 Operator 目录并将其推送到 registry 的镜像。
- 已将 OLM 配置为从 registry 镜像来安装 Cluster Application Migration Operator。

流程

1. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
2. 选择 **Cluster Application Migration Operator** 并点 **Install**。
3. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
4. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
5. 在 **Provided APIs** 下，点 **View 12 more...**。
6. 点 **Create New → MigrationController**。
7. 点击 **Create**。

2.2.3. 启动 CAM web 控制台

您可以在浏览器中启动 CAM web 控制台。

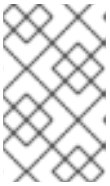
流程

1. 登录到已安装 CAM 工具的 OpenShift Container Platform 集群。
2. 运行以下命令来获取 CAM web 控制台 URL:

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

   输出类似于以下： **https://migration-openshift-migration.apps.cluster.openshift.com**。

3. 启动浏览器并进入 CAM web 控制台。



注意

如果在安装 Cluster Application Migration Operator 后尝试立即访问 CAM web 控制台，则该控制台可能无法加载，因为 Operator 仍然在配置集群。等待几分钟后重试。

4. 如果您使用自签名的 CA 证书，则会提示您接受源集群 API 服务器的 CA 证书。网页会引导您接受剩余证书的过程。

5. 使用 OpenShift Container Platform 的**用户名**和**密码**进行登陆。

2.3. 配置复制存储库

您必须将对象存储配置为用作复制存储库。集群应用程序迁移（CAM）工具将数据从源集群复制到复制存储库，然后从复制存储库复制到目标集群。

CAM 工具支持使用文件系统和快照的数据复制方法来把数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

支持以下存储供应商：

- 多云对象网关 (MCG)
- Amazon Web Services (AWS) S3
- Google Cloud Provider (GCP)
- Microsoft Azure
- 通用 S3 对象存储，例如 Minio 或 Ceph S3

源和目标集群必须有对复制存储库的不受限制的网络访问权限。

在受限环境中，您可以创建一个内部托管的复制存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。

2.3.1. 配置 MCG 存储桶做为复制存储库

您可以安装 OpenShift Container Storage Operator，并将一个 Multi-Cloud Object Gateway (MCG) 存储桶配置为复制存储库。

2.3.1.1. 安装 OpenShift Container Storage Operator

您可以从 OperatorHub 安装 OpenShift Container Storage Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** （本例中为 **OCS**）来查找 **OpenShift Container Storage Operator**。
3. 选择 **OpenShift Container Storage Operator** 并点 **Install**。
4. 选择一个 **Update Channel**、**Installation Mode** 和 **Approval Strategy**。

5. 点 **Subscribe**.
   在 **Installed Operators** 页面中，**OpenShift Container Storage Operator** 会出现在 **openshift-storage** 项目中，状态为 **Succeeded**。

2.3.1.2. 创建 Multi-Cloud Object Gateway 存储桶

您可以创建 Multi-Cloud Object Gateway (MCG) 存储桶的自定义资源 (CR)。

流程

1. 登录到 OpenShift Container Platform 集群：

```
$ oc login
```

2. 使用以下内容创建 **NooBaa** CR 配置文件，**noobaa.yml**：

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

(1) (2) 对于非常小的集群，您可以将 **cpu** 的值改为 **0.1**。

3. 创建 **NooBaa** 对象：

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

4. 使用以下内容创建 **BackingStore** CR 配置文件，**bs.yml**：

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: mcg-pv-pool-bs
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 (1)
    resources:
      requests:
```

```
      storage: 50Gi (2)
    storageClass: gp2 (3)
  type: pv-pool
```

(1) 指定 PV 池中的卷数量。

(2) 指定卷的大小。

(3) 指定存储类。

5. 创建 **BackingStore** 对象：

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

6. 使用以下内容创建 **BucketClass** CR 配置文件，**bc.yml**：

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

7. 创建 **BucketClass** 对象：

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

8. 使用以下内容创建 **ObjectBucketClaim** CR 配置文件，**obc.yml**：

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

(1) 记录下在 CAM web 控制台中添加为复制存储库的存储桶的名称。

9. 创建 **ObjectBucketClaim** 对象：

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

10. 监控资源创建过程以验证 **ObjectBucketClaim** 的状态变为 **Bound**：

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

这个过程可能需要五到十分钟。

11. 获取并记录以下值，当您将复制存储库添加到 CAM web 控制台时需要这些值：

- S3 端点：

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

- S3 provider access key:

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

- S3 provider secret access key:

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

2.3.2. 将 AWS S3 存储桶配置为复制存储库

您可以将 AWS S3 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 AWS CLI。
- 如果您使用快照复制方法：
  - 您必须有权访问 EC2 Elastic Block Storage (EBS)。
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 创建 AWS S3 存储桶：

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

1 指定 S3 存储桶名称。

2 指定 S3 存储桶区域，例如 **us-east-1**。

2. 创建 IAM 用户 **velero**：

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

3. 创建 EC2 EBS 快照策略：

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

4. 为一个或所有 S3 存储桶创建 AWS S3 访问策略：

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

```
        }
    ]
}
EOF
```

1 2 要授予对单一 S3 存储桶的访问权限，请指定存储桶名称。要授予对所有 AWS S3 存储桶的访问权限，请指定 * 而不是存储桶名称：

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

5. 将 EC2 EBS 策略附加到 **velero**：

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

6. 将 AWS S3 策略附加到 **velero**：

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

7. 为 **velero** 创建访问密钥：

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

1 2 记录 **AWS_SECRET_ACCESS_KEY** 和 **AWS_ACCESS_KEY_ID** 以将 AWS 存储库添加到 CAM Web 控制台。

2.3.3. 将 Google Cloud Provider 存储桶配置为复制存储库

您可以将 Google Cloud Provider (GCP) 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 **gsutil**。
- 如果您使用快照复制方法：

- 源和目标集群必须位于同一区域。
- 源和目标集群必须具有相同的存储类。
- 存储类必须与快照兼容。

流程

1. 运行 **gsutil init** 以登录：

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

Your current configuration has been set to: [default]

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

2. 设置 **BUCKET** 变量：

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

1 指定存储桶名称。

3. 创建存储桶：

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

4. 将 **PROJECT_ID** 变量设置为您的活跃项目：

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

5. 创建 **velero** 服务帐户：

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

6. 将 **SERVICE_ACCOUNT_EMAIL** 变量设置为服务帐户的电子邮件地址：

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

7. 向服务帐户授予权限：

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

```
)

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

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

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

8. 将服务帐户的密钥保存到当前目录中的 **credentials-velero** 文件中：

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

2.3.4. 将 Microsoft Azure Blob 存储容器配置为复制存储库

您可以将 Microsoft Azure Blob 存储容器配置为复制存储库。

先决条件

- 您必须具有 Azure 存储帐户。
- 您必须安装了 Azure CLI。
- Azure Blob 存储容器必须可以被源和目标集群访问。
- 如果您使用快照复制方法：
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 设置 **AZURE_RESOURCE_GROUP** 变量：

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. 创建 Azure 资源组：

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

1 指定位置。

3. 设置 **AZURE_STORAGE_ACCOUNT_ID** 变量：

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. 创建 Azure 存储帐户：

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

5. 设置 **BLOB_CONTAINER** 变量：

```
$ BLOB_CONTAINER=velero
```

6. 创建 Azure Blob 存储容器：

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

7. 为 **velero** 创建服务主体和凭证：

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

8. 在 **credentials-velero** 文件中保存服务主体的凭证：

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

2.4. 使用 CAM WEB 控制台迁移应用程序

您可以通过在 CAM web 控制台中添加集群和复制存储库来迁移应用程序工作负载。然后，您可以创建并运行迁移计划。

如果集群或复制存储库有自签名证书保护，您可以创建 CA 证书捆绑包文件或禁用 SSL 验证。

2.4.1. 创建 CA 证书捆绑包文件

如果您使用自签名证书来保护集群或复制存储库的安全，则证书验证可能会失败，出错信息如下：**Certificate signed by unknown authority**。

您可以创建自定义 CA 证书捆绑包文件，并在添加集群或复制存储库时将其上传到 CAM web 控制台。

流程

从远程端点下载 CA 证书，并将其保存为 CA 捆绑包文件：

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

1. 指定端点的主机 FQDN 和端口，如 **api.my-cluster.example.com:6443**。
2. 指定 CA 捆绑包文件的名称。

2.4.2. 把集群添加到 CAM web 控制台中

您可以将源集群添加到 CAM web 控制台。

先决条件

如果要使用 Azure 快照复制数据：

- 在添加源集群时，您必须提供 Azure 资源组名称。
- 源和目标集群必须位于同一 Azure 资源组且位于同一位置。

流程

1. 登录到集群。
2. 获取服务帐户令牌：

```
$ oc sa get-token mig -n openshift-migration
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwi
a3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJtaWciLCJrdWJlcm5ldGVz
LmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoibWlnLXRva2VuLWs4dDJyIiwia3ViZXJuZ
XRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6Im1pZyIsImt1YmV
ybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImE1YjFiYWM
wLWMxYmYtMTFlOS05Y2NiLTAyOWRmODYwYjMwOCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlY
WNjb3VudDptaWc6bWlnIn0.xqeeAINK7UXpdRqAtOj70qhBJPeMwmgLomV9iFxr5RoqUgKchZ
RG2J2rkqmPm6vr7K-
cm7ibD1IBpdQJCcVDuoHYsFgV4mp9vgOfn9osSDp2TGikwNz4Az95e81xnjVUmzh-
NjDsEpw71DH92iHV_xt2sTwtzftS49LpPW2LjrV0evtNBP_t_RfskdArt5VSv25eORl7zScqfe1CiM
kcVbf2UqACQjo3LbkpfN26HAioO2oH0ECPiRzT0Xyh-KwFutJLS9Xgghyw-
LD9kPKcE_xbbJ9Y4Rqajh7WdPYuB0Jd9DPVrsImzK-F6cgHHYoZEv0SvLQi-
PO0rpDrcjOEQQ
```

3. 登录到 CAM web 控制台。
4. 在 **Clusters** 部分，点 **Add cluster**。
5. 填写以下字段：
   - **Cluster name**：可包括小写字母（**a-z**）和数字（**0-9**）。不能包含空格或国际字符。
   - **URL**：集群 API 服务器的 URL，如 **https://<master1.example.com>:8443**。

- **Service account token**：从源集群获取的字符串。
- **Azure cluster**：可选。如果要使用 Azure 快照复制数据，请选择此项。
- **Azure resource group**：如果选中了 **Azure cluster**，则会出现此字段。
- 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 CA 捆绑包文件。

6. 点 **Add cluster**。
   集群会出现在 **Clusters** 部分。

2.4.3. 在 CAM web 控制台中添加复制程序库

您可以将对象存储桶作为复制存储库添加到 CAM web 控制台。

先决条件

- 您必须配置用于迁移数据的对象存储桶。

流程

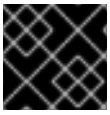
1. 登录到 CAM web 控制台。
2. 在 **Replication repositories** 部分，点 **Add repository**。
3. 选择 **Storage provider type** 并填写以下字段：
   - **AWS** 适用于 S3、MCSG 和通用 S3 供应商：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **S3 bucket name**：指定您创建的 S3 存储桶的名称。
     - **S3 bucket region**：指定 S3 存储桶区域。AWS S3 **必填**。**Optional** 用于其他 S3 供应商。
     - **S3 端点**：指定 S3 服务的 URL，而不是存储桶，例如：**https://<s3-storage.apps.cluster.com>**。通用 S3 供应商**必填**。您必须使用 **https://** 前缀。
     - **S3 provider access key**：为 AWS 指定 **<AWS_SECRET_ACCESS_KEY>**，或者为 MCG 指定 S3 供应商访问密钥。
     - **S3 provider secret access key**：为 AWS 指定 **<AWS_ACCESS_KEY_ID>**，或者为 MCG 指定 S3 供应商 secret 访问密钥。
     - **Require SSL verification**：如果您使用的是通用 S3 供应商，则清除此复选框。
     - 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 Base64 编码的 CA 捆绑包文件。
   - **GCP**：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **GCP bucket name**：指定 GCP 存储桶的名称。
     - **GCP credential JSON blob**：在 **credentials-velero** 文件中指定字符串。

- Azure：
  - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
  - **Azure resource group**：指定 Azure Blob 存储的资源组。
  - **Azure storage account name**：指定 Azure Blob 存储帐户名称
  - **Azure credentials - INI file contents**：在 **credentials-velero** 文件中指定字符串。

4. 点 **Add repository** 并等待连接验证。
5. 点 **Close**。
   新存储库会出现在 **Replication repositories** 部分。

2.4.4. 为大型迁移修改迁移计划限制

您可以更改大型迁移的迁移计划限制。



重要

您需要首先在自己的环境对所做的更改进行测试，以避免迁移失败。

单个迁移计划有以下默认限制：

- 10 个命名空间
  如果超过这个限制，CAM web 控制台会显示一个 **Namespace limit exceeded**错误，您将无法创建迁移计划。
- 100 个 Pod
  如果超过 Pod 限制，CAM web 控制台会显示类似以下示例的警告信息: **Plan has been validated with warning condition(s).查看警告信息。pod limit: 100 exceeded, found: 104。**
- 100 个持久性卷（PV）
  如果超过持久性卷限制，则 CAM web 控制台会显示类似的警告信息。

流程

1. 编辑迁移控制器 CR：

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

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

2. 更新以下参数：

```
...
migration_controller: true

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

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

2.4.5. 在 CAM web 控制台中创建迁移计划

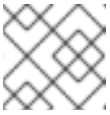
您可以在 CAM web 控制台中创建迁移计划。

先决条件

- CAM web 控制台必须包含以下内容：
  - 源集群
  - 目标集群，它会在 CAM 工具安装过程中自动添加
  - 复制软件仓库
- 源和目标集群必须可以通过网络相互访问，并可以访问复制存储库。
- 如果要使用快照复制数据，则源和目标集群必须在同一云供应商（AWS、GCP 或 Azure）以及同一区域中。

流程

1. 登录到 CAM web 控制台。
2. 在 **Plans** 部分，点 **Add Plan**。
3. 输入 **Plan name** 并点 **Next**。
   Plan name 最多可包含 253 个小写字母数字字符（**a-z，0-9**）。它不能包含空格或下划线（_）。
4. 选一个 **Source cluster**。
5. 选一个 **Target cluster**。
6. 选一个 **Replication repository**。
7. 选择要迁移的项目并点 **Next**。
8. 选择 **Copy** 或 **Move** PV：
   - **Copy** 将源集群的 PV 中的数据复制到复制存储库中，然后在目标集群中新创建的具有类似特征的 PV 上恢复它。
     可选： 您可以通过选择 **Verify copy**来使用文件系统的方法来验证复制的数据。这个选项为每个源文件生成 checksum 并在恢复后对其进行检查。这可能会大大降低性能。
   - **Move** 从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。
9. 点 **Next**。
10. 为 PV 选择 **Copy method**：
    - **Snapshot** 使用云供应商的快照功能备份和恢复磁盘。它比 **Filesystem** 快得多。



注意

存储和集群必须位于同一区域，存储类必须兼容。

- **Filesystem** 将源磁盘中的数据文件复制到新创建的目标磁盘。

11. 为 PV 选择一个 **Storage class**。
如果选择了 **Filesystem** 复制方法，您可以在迁移过程中更改存储类，例如：从 Red Hat Gluster Storage 或 NFS 存储改为 Red Hat Ceph Storage。

12. 点 **Next**。

13. 如果您要添加迁移 hook，请点击 **Add Hook** 并执行以下步骤：

    a. 指定存储桶的名称。

    b. 选择 **Ansible playbook** 来使用您自己的 playbook 或 **Custom container image** 来使用以其他语言编写的 hook。

    c. 点击 **Browse** 上传 playbook。

    d. 可选： 如果您未使用默认 Ansible 运行时镜像，请指定自定义 Ansible 镜像。

    e. 指定要运行 hook 的集群。

    f. 获取服务帐户名称。

    g. 指定命名空间。

    h. 选择您希望 hook 运行的迁移步骤：

       - PreBackup: 在源集群上启动备份前的任务
       - PostBackup: 在源集群中完成备份任务后
       - PreRestore: 在目标集群上启动恢复前的任务
       - PostRestore: 在目标集群中完成恢复后的任务

14. 点 **Add**。
您可以在迁移计划中添加最多四个 hook，将每个 hook 分配给不同的迁移步骤。

15. 点 **Finish**。

16. 点 **Close**。
迁移计划会出现在 **Plans** 部分。

2.4.6. 在 CAM web 控制台中运行迁移计划

您可以使用在 CAM web 控制台中创建的迁移计划来 stage 或迁移应用程序和数据。

先决条件

CAM web 控制台必须包含以下内容：

- 源集群

- 目标集群，它会在 CAM 工具安装过程中自动添加
- 复制软件仓库
- 有效的迁移计划

流程

1. 登录目标集群上的 CAM web 控制台。
2. 选择迁移计划。
3. 点 **Stage** 以在不停止应用程序的情况下，将数据从源集群复制到目标集群。
   您可以多次运行 **Stage** 以减少实际迁移时间。
4. 当准备好迁移应用程序工作负载时，点 **Migrate**。
   Migrate 在源集群中停止应用程序工作负载，并在目标集群中重新创建其资源。
5. 另外，还可以在 **Migrate** 窗口中选择 **Do not stop applications on the source cluster during migration**。
6. 点 **Migrate**。
7. 可选： 要停止迁移过，请点击 Options 菜单  并选择 **Cancel**。
8. 迁移完成后，在 OpenShift Container Platform web 控制台中确认已成功迁移了应用程序：
   a. 点 **Home → Projects**。
   b. 点迁移的项目查看其状态。
   c. 在 **Routes** 部分，点击 **Location** 验证应用程序是否正常运行。
   d. 点 **Workloads → Pods** 来验证 Pod 在迁移的命名空间中运行。
   e. 点 **Storage → Persistent volumes** 确认正确置备了被迁移的持久性卷。

2.5. 故障排除

您可以查看迁移自定义资源 (CR)，并下载日志来排除迁移失败的问题。

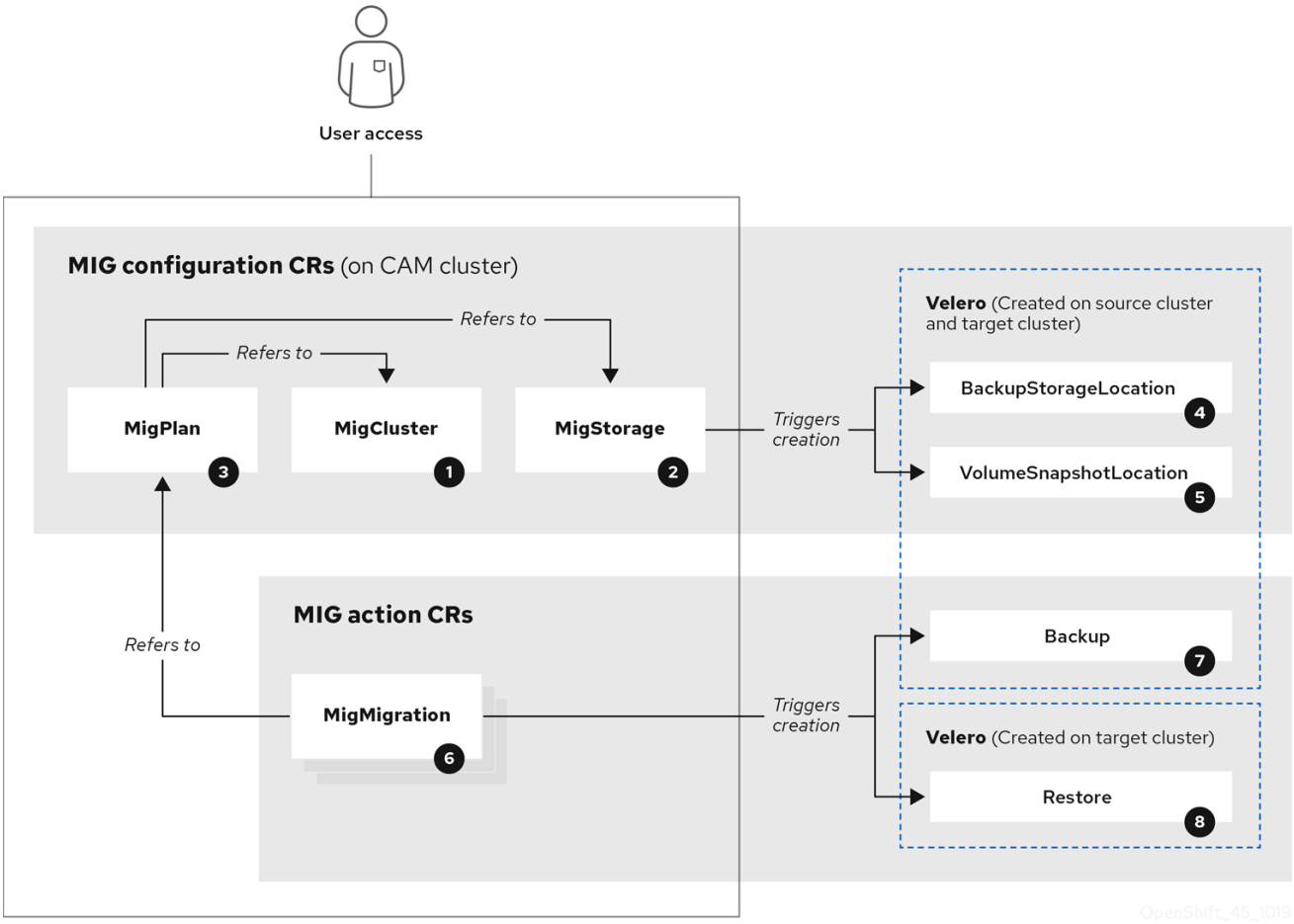
如果应用程序在迁移失败时停止，您必须手动回滚，以防止数据崩溃。

注意

如果应用程序在迁移过程中没有停止，则不需要手动回滚，因为原始应用程序仍然在源集群中运行。

2.5.1. 查看迁移自定义资源

集群应用程序迁移 (CAM) 工具会创建以下 CR：



1 MigCluster （配置，CAM 集群）：集群定义

2 MigStorage （配置，CAM 集群）：存储定义

3 MigPlan （配置，CAM 集群）：迁移计划

MigPlan CR 描述了要迁移的源和目标集群、存储库和命名空间。它与 0 个、1 个或多个 MigMigration CR 关联。

注意

删除 MigPlan CR 会删除关联的 MigMigration CR。

4 BackupStorageLocation （配置，CAM 集群）：Velero 备份对象的位置

5 VolumeSnapshotLocation （配置，CAM 集群）：Velero 卷快照的位置

6 MigMigration（操作，CAM 集群）：Migration，在迁移期间创建

在每次进行 stage 或迁移数据时都会创建一个 MigMigration CR。每个 MigMigration CR 都会与一个 MigPlan CR 关联。

7 Backup（操作，源集群）：当运行迁移计划时，MigMigration CR 在每个源集群上创建两个 Velero 备份 CR：

- 备份 CR #1 用于Kubernetes 对象
- 备份 CR #2 用于 PV 数据

8 Restore （操作，目标集群）：在运行迁移计划时，MigMigration CR 在目标集群上创建两个 Velero 恢复 CR：

- 恢复 CR #1（使用备份 CR #2）用于 PV 数据
- 恢复 CR #2（使用备份 CR #1）用于 Kubernetes 对象

流程

1. 获取 CR 名称：

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

1 指定迁移 CR，如 **migmigration**。

输出结果类似以下：

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

2. 查看 CR：

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

输出结果类似以下示例。

MigMigration 示例

```
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>
```

Velero 备份 CR #2 示例（PV 数据）

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

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

Velero 恢复 CR #2 示例（Kubernetes 资源）

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

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

2.5.2. 下载迁移日志

您可以在 CAM web 控制台中下载 Velero 、 Restic 和 Migration controller 日志，以排除出现故障的迁移问题。

流程

1. 登录到 CAM 控制台。
2. 点击 **Plans** 查看迁移计划列表。
3. 点击特定迁移计划  的 **Options** 菜单并选择 **Logs**。
4. 点 **Download Logs** 为所有集群下载迁移控制器、Velero 和 Restic 的日志。
5. 要下载特定的日志：
   a. 指定日志选项：
      - **Cluster**：选择源、目标或 CAM 主机集群。
      - **Log source**：选择 **Velero**、**Restic** 或 **Controller**。
      - **Pod source**：选择 Pod 名称，例如：**controller-manager-78c469849c-v6wcf** 此时会显示所选日志。

        您可以通过更改您的选择来清除日志选择设置。

   b. 点 **Download Selected** 下载所选日志。

另外，您可以使用 CLI 访问日志，如下例所示：

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

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

2.5.3. 错误信息

2.5.3.1. Velero Pod 日志中的 Restic 超时错误消息

如果因为 Restic 超时造成迁移失败，以下出错信息会出现在 Velero Pod 日志中：

```
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 的默认值为一小时。您可以为大型迁移增加这个参数值，请注意，高的值可能会延迟返回出错信息。

流程

1. 在 OpenShift Container Platform web 控制台中导航至 **Operators → Installed Operators**。
2. 点 **Cluster Application Migration Operator**。
3. 在 **MigrationController** 标签页中点 **migration-controller**。
4. 在 **YAML** 标签页中，更新以下参数值：

```
spec:
  restic_timeout: 1h (1)
```

(1) 有效单元是 **h** （小时）、**m** （分钟）和 **s** （秒），例如 **3h30m15s**。

5. 点 **Save**。

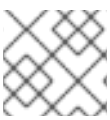
2.5.3.2. MigMigration Custom Resource（CR）中的 ResticVerifyErrors

如果迁移使用文件系统数据复制方法的 PV 时数据验证失败，在 MigMigration 自定义资源 (CR) 中会出现以下错误：

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

(1) 错误消息标识了 Restore CR 名称。

(2) **ResticErrors** 也会出现。**ResticErrors** 是一个包括验证错误的一般错误警告。



注意

数据验证错误不会导致迁移过程失败。

您可以检查目标集群的 Restore CR，以识别数据验证错误的来源。

流程

1. 登录到目标集群。
2. 查看 Restore CR:

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

输出通过 **PodVolumeRestore** 错误来指定 PV：

```
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. 查看 **PodVolumeRestore** CR:

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

输出中标识了记录错误的 Restic Pod:

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

4. 查看 Restic Pod 日志：

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

2.5.4. 手动回滚迁移

如果您的应用程序在迁移失败时停止，您必须手动回滚，以防止 PV 中的数据被破坏。

如果应用程序在迁移过程中没有停止，则不需要进行手动回滚，因为原始应用程序仍然在源集群中运行。

流程

1. 在目标集群中，切换到迁移的项目：

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

2. 获取部署的资源：

```
$ oc get all
```

3. 删除部署的资源以确保应用程序没有在目标集群中运行，并访问 PVC 上的数据：

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

4. 要停止 DaemonSet 而不删除它，在 YAML 文件中更新 **nodeSelector** ：

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

1 指定一个没有存在于任何节点上的**nodeSelector** 值。

5. 更新每个 PV 的重新声明策略，以便删除不必要的数据。在迁移过程中，绑定 PV 的重新声明策略是**reclaim**，以确保应用程序从源集群中被删除时不会丢失数据。您可以在回滚过程中删除这些 PV。

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

1 指定 **Recycle** 或 **Delete**。

6. 在源集群中，切换到迁移的项目：

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

7. 获取项目部署的资源：

```
$ oc get all
```

8. 启动每个部署资源的一个或多个副本：

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

9. 如果在操作中被更改了，把 DaemonSet 的 **nodeSelector** 改回其原始值。

2.5.5. 为客户支持问题单收集数据

如果创建一个客户支持问题单，您可以使用 **openshift-migration-must-gather-rhel8** 镜像的 **must-gather** 工具来收集与您的集群相关的信息，并把这些信息上传到红帽客户门户网站。

openshift-migration-must-gather-rhel8 镜像会收集默认的 **must-gather** 镜像不收集的日志和 CR 数据。

流程

1. 进入要存储 **must-gather** 数据的目录。
2. 运行 **oc adm must-gather** 命令：

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

must-gather 工具程序收集集群数据，并把它保存在 **must-gather.local.<uid>** 目录中。

3. 从 **must-gather** 数据中删除验证密钥和其他敏感信息。
4. 创建一个包含 **must-gather.local.<uid>** 目录内容的归档文件：

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

在红帽客户门户中，为您的问题单附上这个压缩文件。

2.5.6. 已知问题

这个版本有以下已知问题：

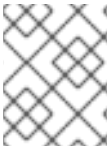
- 在迁移过程中，集群应用程序迁移 (CAM) 工具会保留以下命名空间注解：
  - **openshift.io/sa.scc.mcs**
  - **openshift.io/sa.scc.supplemental-groups**
  - **openshift.io/sa.scc.uid-range**
    这些注解会保留 UID 范围，确保容器在目标集群中保留其文件系统权限。这可能会存在一定的风险。因为迁移的 UID 可能已存在于目标集群的现有或将来的命名空间中。
    (BZ#1748440)
- 如果一个 AWS 存储桶被添加到 CAM web 控制台，然后将其删除，则其状态会保持为 **True**，这是因为 MigStorage CR 没有被更新。(BZ#1738564)
- 大多数集群范围的资源尚未由 CAM 工具处理。如果应用程序需要集群范围的资源，则可能需要在目标集群上手动创建。
- 如果迁移失败，则迁移计划不会为静默的 pod 保留自定义 PV 设置。您必须手动回滚，删除迁移计划，并使用 PV 设置创建新的迁移计划。(BZ#1784899)

- 如果因为 Restic 超时造成大型迁移失败，您可以提高 Migration controller CR 中的 **restic_timeout** 参数值。
- 如果您选择了为使用文件系统复制方法迁移的 PV 数据进行验证的选项，则性能会非常慢。Velero 为每个文件生成一个 checksum，并在恢复该文件时对其进行检查。

第 3 章 从 OPENSHIFT CONTAINER PLATFORM 4.2 和更新版本进行迁移

3.1. 迁移工具和先决条件

您可以使用集群应用程序迁移 (CAM) 工具将应用程序工作负载从 OpenShift Container Platform 4.2 和更高的版本迁移到 4.3。使用 CAM 工具，您可以控制迁移并最小化应用程序的停机时间。



注意

只要正确配置了源和目标集群，您可以在相同版本的 OpenShift Container Platform 集群间进行迁移（如从 4.2 迁移到 4.2，或从 4.3 迁移到 4.3）。

CAM 工具的 web 控制台和 API，基于 Kubernetes 自定义资源，您可以按照命名空间迁移有状态及无状态的应用程序工作负载。

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。

3.1.1. 迁移先决条件

- 必须将源集群升级到最新的 z-stream 版本。
- 需要在所有集群中都有 **cluster-admin** 权限。
- 源和目标集群必须有对复制存储库的不受限制的网络访问权限。
- 安装 Migration controller 的集群必须具有对其他集群的不受限制的访问权限。
- 如果应用程序使用 **openshift** 命名空间中的镜像，则目标集群中必须有所需的镜像版本。如果没有所需的镜像，您必须更新 **imagestreamtags** 的引用以使用与应用程序兼容的可用版本。如果无法更新 **imagestreamtags**，您可以手动将相关的镜像上传到应用程序命名空间中，并更新应用程序以引用它们。

以下 **imagestreamtags** 已从 OpenShift Container Platform 4.2 中 *删除*：

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

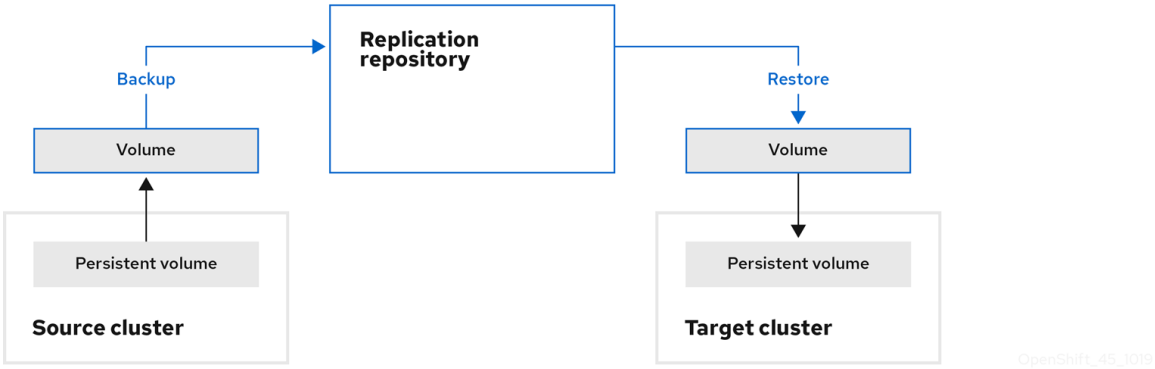
- **php:5.5、php:5.6**
- **postgresql:9.2, postgresql:9.4, postgresql:9.5**
- **python:3.3、python:3.4**
- **ruby:2.0、ruby:2.2**

3.1.2. 关于集群应用程序迁移（CAM）工具

集群应用程序迁移 (CAM) 工具可让您使用 CAM web 控制台或 Kubernetes API 将 OpenShift Container Platform 源集群中的 Kubernetes 资源、持久性卷数据和内部容器镜像迁移到 OpenShift Container Platform 4.3 目标集群。

使用 CAM web 控制台迁移应用程序涉及以下步骤：

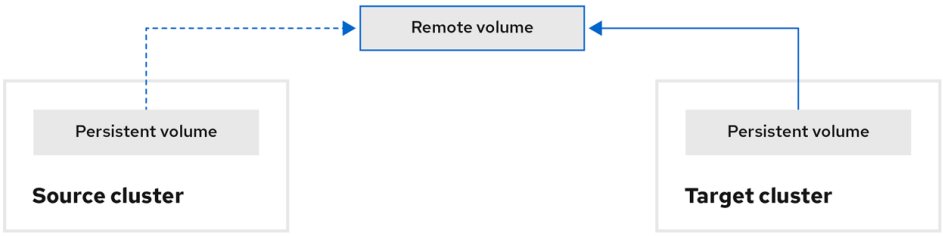
1. 在所有集群中安装 Cluster Application Migration Operator
   您可以在有限的或没有互联网访问的受限环境中安装 Cluster Application Migration Operator。源和目标集群必须可以在相互间进行访问，而需要可以访问 registry 的镜像（mirror）.
2. 配置复制存储库，这是 CAM 工具用来迁移数据的中间对象存储
   源和目标集群必须有对复制存储库的不受限制的网络访问权限。在受限环境中，您可以使用内部托管的 S3 存储存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。
3. 在 CAM web 控制台中添加源集群
4. 在 CAM web 控制台中添加复制存储库
5. 创建迁移计划，包含以下数据迁移选项之一：
   - **Copy**：CAM 工具将数据从源集群复制到复制存储库，再从复制存储库把数据复制到目标集群。



   - **Move**：CAM 工具从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。

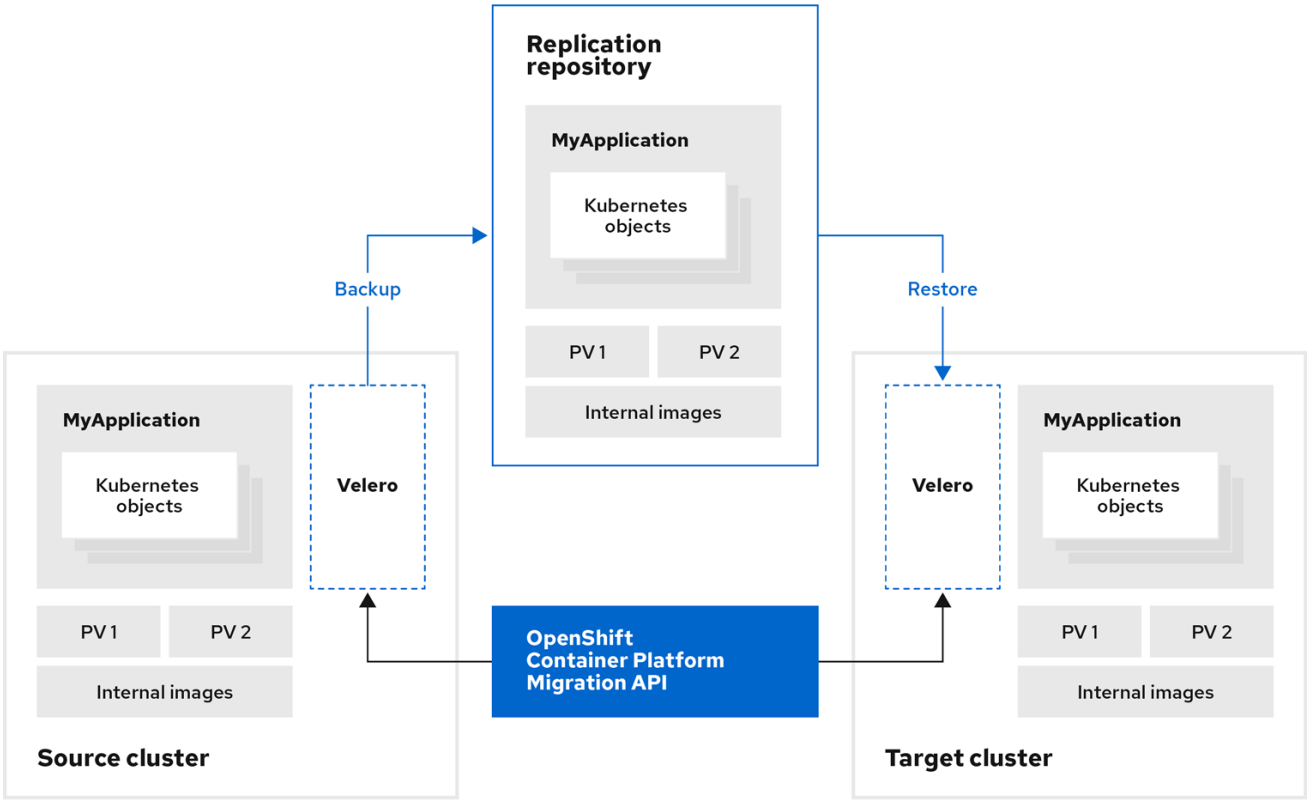
注意

虽然复制存储库没有出现在此图表中，但实际迁移过程需要它。



6. 运行迁移计划，使用以下选项之一：

- **Stage** （可选）在不停止应用程序的情况下将数据复制到目标集群。
  Stage 可以多次运行，以便在迁移前将大多数数据复制到目标。这样可最小化实际迁移时间和应用程序停机时间。
- **Migrate** 在源集群中停止应用程序，并在目标集群中重新创建其资源。您可以选择在不停止应用程序的情况下迁移工作负载。



3.1.3. 关于数据复制方法

CAM 工具支持文件系统和快照数据复制方法，用于将数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

3.1.3.1. 文件系统复制方法

CAM 工具将数据文件从源集群复制到复制存储库，并从那里复制到目标集群。

表 3.1. 文件系统复制方法概述

优点	限制：
• 集群可以有不同的存储类 • 所有 S3 存储供应商均支持 • 使用 checksum 验证数据（可选）	• 比快照复制方法慢 • 可选的数据校验可能会显著降低性能

3.1.3.2. 快照复制方法

CAM 工具将源集群的数据快照复制到云供应商的对象存储，后者配置为复制存储库。数据在目标集群上恢复。

AWS、Google Cloud Provider 和 Microsoft Azure 支持快照复制方法。

表 3.2. 快照复制方法概述

优点	限制：
• 比文件系统复制方法快	• 云供应商必须支持快照。 • 集群必须位于相同的云供应商。 • 集群必须位于同一位置或区域。 • 集群必须具有相同的存储类。 • 存储类必须与快照兼容。

3.1.4. 关于迁移 hook

您可以在迁移期间的特定点使用迁移 hook 运行 Ansible playbook。您在创建迁移计划时可以添加 hook。

注意

如果您不想使用 Ansible playbook，您可以创建自定义容器镜像并将其添加到迁移计划中。

迁移 hook 执行的任务包括自定义应用程序默认、手动迁移不受支持的数据类型以及在迁移后更新应用程序。

在源集群或目标集群中以以下迁移步骤之一运行单个迁移 hook:

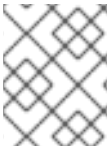
- PreBackup: 在源集群上启动备份前的任务
- PostBackup: 在源集群中完成备份任务后
- PreRestore: 在目标集群上启动恢复前的任务
- PostRestore: 在目标集群中完成恢复后的任务
  您可以为每个迁移步骤分配一个 hook，单个迁移计划最多可分配四个 hook。

默认 **hook-runner** 镜像为 **registry.redhat.io/rhcam-1-2/openshift-migration-hook-runner-rhel7**。该镜像基于 Ansible Runner，并包括用于 Ansible Kubernetes 资源的 **python-openshift** 以及更新的 **oc** 二进制文件。您还可以使用其他 Ansible 模块或工具创建自己的 hook 镜像。

Ansible playbook 作为一个 ConfigMap 挂载到 hook 容器上。hook 容器在具有指定服务帐户和命名空间的集群中作为一个作业（job）运行。作业运行（即使初始 Pod 被驱除或终止），直到达到默认的 **backoffLimit** (**6**)或成功完成为止。

3.2. 部署集群应用程序迁移（CAM）工具

您可以在 OpenShift Container Platform 4.3 目标集群和 4.2 源集群中安装 Cluster Application Migration Operator。默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：



注意

可选：您可以配置 Cluster Application Migration Operator，以便在 OpenShift Container Platform 3 集群或远程集群中安装 CAM 工具。

在受限环境中，您可以从本地镜像 registry 安装 Cluster Application Migration Operator。

在集群中安装 Cluster Application Migration Operator 后,您可以启动 CAM 工具。

3.2.1. 安装 Cluster Application Migration Operator

您可以使用操作生命周期管理器（OLM）在 OpenShift Container Platform 4.3 目标集群和 OpenShift Container Platform 4.1 源集群上安装 Cluster Application Migration Operator。

3.2.1.1. 在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。

8. 点击 **Create**。
9. 点 **Workloads → Pod** 来验证 Controller Manager 、Migration UI 、Restic 和 Velero Pod 是否正在运行。

3.2.1.2. 在 OpenShift Container Platform 4.2 源集群上安装 Cluster Application Migration Operator

您可以使用 OLM 在 OpenShift Container Platform 4 源集群上安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 在 **spec** 小节中更新 **migration_controller 和 migration_ui** 参数：

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

9. 点击 **Create**。
10. 点 **Workloads → Pod** 来验证 Restic 和 Velero Pod 是否正在运行。

3.2.2. 在受限环境中安装 Cluster Application Migration Operator

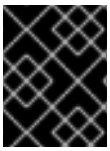
您可以为 OpenShift Container Platform 4 构建自定义 Operator 目录镜像，将其推送到本地镜像 registry，并将 OLM 配置为从本地 registry 安装 Operator。

其他资源

- 在受限网络中使用 Operator Lifecycle Manager

3.2.2.1. 构建 Operator 目录镜像

集群管理员可以构建自定义 Operator 目录镜像，供 Operator Lifecycle Manager (OLM) 使用，并将镜像推送到支持 Docker v2-2 的容器镜像 registry。对于受限网络中的集群，此 registry 可以是集群有网络访问权限的 registry，如在受限网络安装过程中创建的镜像 registry。



重要

OpenShift Container Platform 集群的内部 registry 不能用作目标 registry，因为它不支持没有标签的推送，而在镜像（mirrir）过程中需要这个功能。

在这一示例中，流程假定在使用镜像 registry 时可访问您的网络以及互联网。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- **oc** 版本 4.3.5+
- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

- 如果您正在使用 quay.io 帐户可访问的私有命名空间，您必须设置 Quay 身份验证令牌。使用您的 quay.io 凭证对登录 API 发出请求，从而设置用于 **--auth-token** 标志的 **AUTH_TOKEN** 环境变量：

```
$ AUTH_TOKEN=$(curl -sH "Content-Type: application/json" \
    -XPOST https://quay.io/cnr/api/v1/users/login -d '
    {
        "user": {
            "username": """<quay_username>""",
            "password": """<quay_password>"""
        }
    }' | jq -r '.token')
```

流程

1. 在没有网络访问限制的工作站中，与目标镜像（mirror） registry 进行身份验证：

```
$ podman login <registry_host_name>
```

还可使用 **registry.redhat.io** 验证，以便在构建期间拉取基础镜像：

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

2. 根据 quay.io 中的 **redhat-operators** 目录构建目录镜像，进行标记并将其推送到您的镜像 registry：

```
$ oc adm catalog build \
    --appregistry-org redhat-operators \ 1
```

```
    --from=registry.redhat.io/openshift4/ose-operator-registry:v4.3 \2
    --filter-by-os="linux/amd64" \3
    --to=<registry_host_name>:<port>/olm/redhat-operators:v1 \4
    [-a ${REG_CREDS}] \5
    [--insecure] \6
    [--auth-token "${AUTH_TOKEN}"] 7

INFO[0013] loading Bundles
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
...
Pushed sha256:f73d42950021f9240389f99ddc5b0c7f1b533c054ba344654ff1edaf6bf827e3
to example_registry:5000/olm/redhat-operators:v1
```

1. 从 App Registry 实例中拉取的机构（命名空间）。
2. 使用与目标 OpenShift Container Platform 集群主版本和次版本匹配的标签，将 **--from** 设置为 **ose-operator-registry** 基础镜像。
3. 将 **--filter-by-os** 设置为用于基本镜像的操作系统和架构，该镜像必须与目标 OpenShift Container Platform 集群匹配。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
4. 为您的目录镜像命名并包含标签，例如: **v1**。
5. 可选：如果需要，指定 registry 凭证文件的位置。
6. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。
7. 可选：如果使用其他不公开的应用程序 registry 目录，则需要指定 Quay 身份验证令牌。

有时红帽目录中会意外引入无效的清单；当发生这种情况时，您可能会看到一些错误：

```
...
INFO[0014] directory
dir=/var/folders/st/9cskxqs53ll3wdn434vw4cd80000gn/T/300666084/manifests-829192605
file=4.2 load=package
W1114 19:42:37.876180   34665 builder.go:141] error building database: error loading
package into db: fuse-camel-k-operator.v7.5.0 specifies replacement that couldn't be found
Uploading ... 244.9kB/s
```

这些错误通常不是致命的，如果所提及 Operator 软件包不包含您计划安装的 Operator 或其依赖项，则可以忽略它们。

3.2.2.2. 针对受限网络配置 OperatorHub

集群管理员可以使用自定义 Operator 目录镜像将 OLM 和 OperatorHub 配置为在受限网络环境中使用本地内容。本例中的流程使用之前构建并推送到受支持的 registry 的自定义 **redhat-operators** 目录镜像。

先决条件

- 具有无限网络访问权限的 Linux 工作站
- 推送到受支持的 registry 的自定义 Operator 目录镜像
- **oc** 版本 4.3.5+

- **podman** 1.4.4+ 版
- 访问支持 Docker v2-2 的镜像（mirror）registry
- 如果您正在使用私有 registry，请将 **REG_CREDS** 环境变量设置为到 registry 凭证的文件路径。例如，对于 **podman** CLI:

```
$ REG_CREDS=${XDG_RUNTIME_DIR}/containers/auth.json
```

流程

1. 通过在 spec 中添加 **disableAllDefaultSources: true** 来禁用默认 OperatorSource：

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

该操作将禁用在 OpenShift Container Platform 安装期间默认配置的默认 OperatorSource。

2. **oc adm catalog mirror** 命令提取自定义 Operator catalog 镜像的内容，以生成镜像（mirror）所需的清单：您可以选择：
   - 允许该命令的默认行为在生成清单后自动将所有镜像内容镜像到您的镜像 registry 中，或者
   - 添加 **--manifests-only** 标志来只生成镜像所需的清单，但并不实际将镜像内容镜像到 registry。这对检查哪些要镜像（mirror）非常有用。如果您只需要部分内容的话，可以对映射列表做出任何修改。然后，您可以使用该文件与 **oc image mirror** 命令一起，在以后的步骤中镜像修改的镜像列表。

   在没有网络访问限制的工作站中，运行以下命令：

```
$ oc adm catalog mirror \
    <registry_host_name>:<port>/olm/redhat-operators:v1 \ 1
    <registry_host_name>:<port> \
    [-a ${REG_CREDS}] \ 2
    [--insecure] \ 3
    [--filter-by-os="<os>/<arch>"] \ 4
    [--manifests-only] 5
```

1. 指定 Operator 目录镜像。
2. 可选：如果需要，指定 registry 凭证文件的位置。
3. 可选：如果您不想为目标 registry 配置信任，请添加 **--insecure** 标志。
4. 可选：因为目录可能会引用支持多个架构和操作系统的镜像，您可以根据架构和操作系统进行过滤来只对匹配的镜像进行镜像（mirror）。有效值是 **linux/amd64**、**linux/ppc64le** 和 **linux/s390x**。
5. 可选：只生成镜像所需的清单，但并不实际将镜像内容镜像到 registry。

输出示例

```
using database path mapping: /:/tmp/190214037
wrote database to /tmp/190214037
```

```
using database at: /tmp/190214037/bundles.db 1
...
```

1 命令生成的临时数据库。

在运行命令后，会在当前目录中生成 **<image_name>-manifests/** 目录以及以下文件：

- 用来定义 ImageContentSourcePolicy 对象的 **imageContentSourcePolicy.yaml**，它可以将节点配置为在 Operator 清单中存储的镜像（image）引用和镜像 (mirror) 的 registry 间进行转换。
- **mapping.txt** 文件，在其中包含所有源镜像，并将它们映射到目标 registry。此文件与 **oc image mirror** 命令兼容，可用于进一步自定义镜像（mirror）配置。

3. 如果您在上一步中使用 **--manifests-only** 标志，并只想镜像部分内容：

a. 将 **mapping.txt** 文件中的镜像列表改为您的规格。如果您不确定要镜像的镜像子集的名称和版本，请使用以下步骤查找：

i. 对 **oc adm catalog mirror** 命令生成的临时数据库运行 **sqlite3** 工具，以检索与一般搜索查询匹配的镜像列表。输出以后如何编辑 **mapping.txt** 文件的帮助信息。
例如，要检索与字符串 **clusterlogging.4.3** 类似的镜像列表：

```
$ echo "select * from related_image \
    where operatorbundle_name like 'clusterlogging.4.3%';" \
    | sqlite3 -line /tmp/190214037/bundles.db 1
```

1 请参阅 **oc adm catalog mirror** 命令的输出结果来查找数据库文件的路径。

输出示例

```
image = registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0

image = registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf75
9c506
operatorbundle_name = clusterlogging.4.3.33-202008111029.p0
...
```

ii. 使用上一步的结果来编辑 **mapping.txt** 文件，使其只包含您要镜像的镜像子集。
例如，您可以使用前面示例输出中的 **image** 值来找出您的 **mapping.txt** 文件中存在以下匹配行：

mapping.txt 中的匹配镜像映射

```
registry.redhat.io/openshift4/ose-logging-
kibana5@sha256:aa4a8b2a00836d0e28aa6497ad90a3c116f135f382d8211e3c55f34f
b36dfe61=<registry_host_name>:<port>/openshift4-ose-logging-kibana5:a767c8f0
```

```
registry.redhat.io/openshift4/ose-oauth-
proxy@sha256:6b4db07f6e6c962fc96473d86c44532c93b146bbefe311d0c348117bf759c506=<registry_host_name>:<port>/openshift4-ose-oauth-proxy:3754ea2b
```

在这个示例中，如果您只想镜像这些 image，可以在 **mapping.txt** 文件中删除所有其他条目，仅保留上述两行。

b. 在您的没有网络访问限制的工作站中，使用您修改的 **mapping.txt** 文件，使用 **oc image mirror** 命令将镜像镜像到 registry:

```
$ oc image mirror \
    [-a ${REG_CREDS}] \
    -f ./redhat-operators-manifests/mapping.txt
```

4. 应用 ImageContentSourcePolicy:

```
$ oc apply -f ./redhat-operators-manifests/imageContentSourcePolicy.yaml
```

5. 创建一个 CatalogSource 对象来引用您的目录镜像。

a. 按照您的规格修改以下内容，并将它保存为 **catalogsource.yaml** 文件：

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: <registry_host_name>:<port>/olm/redhat-operators:v1 (1)
  displayName: My Operator Catalog
  publisher: grpc
```

(1) 指定您的自定义 Operator 目录镜像。

b. 使用该文件创建 CatalogSource 对象：

```
$ oc create -f catalogsource.yaml
```

6. 确定成功创建以下资源。

a. 检查 Pod:

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

输出示例

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

b. 检查 CatalogSource:

```
$ oc get catalogsource -n openshift-marketplace
```

输出示例

```
NAME                  DISPLAY               TYPE PUBLISHER  AGE
my-operator-catalog   My Operator Catalog   grpc           5s
```

c. 检查 PackageManifest:

```
$ oc get packagemanifest -n openshift-marketplace
```

输出示例

```
NAME    CATALOG             AGE
etcd    My Operator Catalog  34s
```

现在，您可在受限网络 OpenShift Container Platform 集群的 web 控制台中，通过 **OperatorHub** 安装 Operator。

3.2.2.3. 在一个受限的环境中的 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator

您可以使用 Operator Lifecycle Manager（OLM）在 OpenShift Container Platform 4.3 目标集群上安装 Cluster Application Migration Operator。

默认情况下，Cluster Application Migration Operator 会在目标集群上安装 CAM 工具：

先决条件

- 已创建自定义 Operator 目录并将其推送到 registry 的镜像。
- 已将 OLM 配置为从 registry 镜像来安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 点击 **Create**。

9. 点 **Workloads → Pod** 来验证 Controller Manager 、 Migration UI 、 Restic 和 Velero Pod 是否正在运行。

3.2.2.4. 在一个受限的环境中的 OpenShift Container Platform 4.2 源集群上安装 Cluster Application Migration Operator

您可以使用 OLM 在 OpenShift Container Platform 4 源集群上安装 Cluster Application Migration Operator。

先决条件

- 已创建自定义 Operator 目录并将其推送到 registry 的镜像。
- 已将 OLM 配置为从 registry 镜像来安装 Cluster Application Migration Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 项（在这里是 **Migration**）找到 **Cluster Application Migration Operator**。
3. 选择 **Cluster Application Migration Operator** 并点 **Install**。
4. 在 **Create Operator Subscription** 页面中，选择 **openshift-migration** 命名空间，并指定批准策略。
5. 点 **Subscribe**.
   在 **Installed Operators** 页中，**Cluster Application Migration Operator** 会出现在 **openshift-migration** 项目中，其状态为 **InstallSucceeded**。
6. 在 **Provided APIs** 下，点 **View 12 more...**。
7. 点 **Create New → MigrationController**。
8. 点击 **Create**。

3.2.3. 启动 CAM web 控制台

您可以在浏览器中启动 CAM web 控制台。

流程

1. 登录到已安装 CAM 工具的 OpenShift Container Platform 集群。
2. 运行以下命令来获取 CAM web 控制台 URL:

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

   输出类似于以下： **https://migration-openshift-migration.apps.cluster.openshift.com**。
3. 启动浏览器并进入 CAM web 控制台。

注意

如果在安装 Cluster Application Migration Operator 后尝试立即访问 CAM web 控制台，则该控制台可能无法加载，因为 Operator 仍然在配置集群。等待几分钟后重试。

4. 如果您使用自签名的 CA 证书，则会提示您接受源集群 API 服务器的 CA 证书。网页会引导您接受剩余证书的过程。
5. 使用 OpenShift Container Platform 的**用户名**和**密码**进行登陆。

3.3. 配置复制存储库

您必须将对象存储配置为用作复制存储库。集群应用程序迁移（CAM）工具将数据从源集群复制到复制存储库，然后从复制存储库复制到目标集群。

CAM 工具支持使用文件系统和快照的数据复制方法来把数据从源集群迁移到目标集群。您可以选择适合于您的环境并受您的存储供应商支持的方法。

支持以下存储供应商：

- 多云对象网关 (MCG)
- Amazon Web Services (AWS) S3
- Google Cloud Provider (GCP)
- Microsoft Azure
- 通用 S3 对象存储，例如 Minio 或 Ceph S3

源和目标集群必须有对复制存储库的不受限制的网络访问权限。

在受限环境中，您可以创建一个内部托管的复制存储库。如果使用代理服务器，您必须确保将复制存储库列入白名单。

3.3.1. 配置 MCG 存储桶做为复制存储库

您可以安装 OpenShift Container Storage Operator，并将一个 Multi-Cloud Object Gateway (MCG) 存储桶配置为复制存储库。

3.3.1.1. 安装 OpenShift Container Storage Operator

您可以从 OperatorHub 安装 OpenShift Container Storage Operator。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** （本例中为 **OCS**）来查找 **OpenShift Container Storage Operator**。
3. 选择 **OpenShift Container Storage Operator** 并点 **Install**。
4. 选择一个 **Update Channel**、**Installation Mode** 和 **Approval Strategy**。

5. 点 **Subscribe**.
   在 **Installed Operators** 页面中，**OpenShift Container Storage Operator** 会出现在 **openshift-storage** 项目中，状态为 **Succeeded**。

3.3.1.2. 创建 Multi-Cloud Object Gateway 存储桶

您可以创建 Multi-Cloud Object Gateway (MCG) 存储桶的自定义资源 (CR) 。

流程

1. 登录到 OpenShift Container Platform 集群：

```
$ oc login
```

2. 使用以下内容创建 **NooBaa** CR 配置文件，**noobaa.yml**：

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

(1) (2) 对于非常小的集群，您可以将 **cpu** 的值改为 **0.1**。

3. 创建 **NooBaa** 对象：

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

4. 使用以下内容创建 **BackingStore** CR 配置文件，**bs.yml**：

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: mcg-pv-pool-bs
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 (1)
    resources:
      requests:
```

```
      storage: 50Gi (2)
    storageClass: gp2 (3)
  type: pv-pool
```

(1) 指定 PV 池中的卷数量。

(2) 指定卷的大小。

(3) 指定存储类。

5. 创建 **BackingStore** 对象：

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

6. 使用以下内容创建 **BucketClass** CR 配置文件，**bc.yml**：

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

7. 创建 **BucketClass** 对象：

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

8. 使用以下内容创建 **ObjectBucketClaim** CR 配置文件，**obc.yml**：

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

(1) 记录下在 CAM web 控制台中添加为复制存储库的存储桶的名称。

9. 创建 **ObjectBucketClaim** 对象：

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

10. 监控资源创建过程以验证 **ObjectBucketClaim** 的状态变为 **Bound**：

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

这个过程可能需要五到十分钟。

11. 获取并记录以下值，当您将复制存储库添加到 CAM web 控制台时需要这些值：

- S3 端点：

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

- S3 provider access key:

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

- S3 provider secret access key:

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

3.3.2. 将 AWS S3 存储桶配置为复制存储库

您可以将 AWS S3 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 AWS CLI。
- 如果您使用快照复制方法：
  - 您必须有权访问 EC2 Elastic Block Storage (EBS)。
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 创建 AWS S3 存储桶：

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

1 指定 S3 存储桶名称。

2 指定 S3 存储桶区域，例如 **us-east-1**。

2. 创建 IAM 用户 **velero**：

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

3. 创建 EC2 EBS 快照策略：

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

4. 为一个或所有 S3 存储桶创建 AWS S3 访问策略：

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

```
      }
    ]
}
EOF
```

1 2 要授予对单一 S3 存储桶的访问权限，请指定存储桶名称。要授予对所有 AWS S3 存储桶的访问权限，请指定 * 而不是存储桶名称：

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

5. 将 EC2 EBS 策略附加到 **velero**：

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

6. 将 AWS S3 策略附加到 **velero**：

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

7. 为 **velero** 创建访问密钥：

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

1 2 记录 **AWS_SECRET_ACCESS_KEY** 和 **AWS_ACCESS_KEY_ID** 以将 AWS 存储库添加到 CAM Web 控制台。

3.3.3. 将 Google Cloud Provider 存储桶配置为复制存储库

您可以将 Google Cloud Provider (GCP) 存储桶配置为复制存储库。

先决条件

- AWS S3 存储桶必须可以被源和目标集群访问。
- 您必须安装了 **gsutil**。
- 如果您使用快照复制方法：

- 源和目标集群必须位于同一区域。
- 源和目标集群必须具有相同的存储类。
- 存储类必须与快照兼容。

流程

1. 运行 **gsutil init** 以登录：

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

Your current configuration has been set to: [default]

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

2. 设置 **BUCKET** 变量：

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

1 指定存储桶名称。

3. 创建存储桶：

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

4. 将 **PROJECT_ID** 变量设置为您的活跃项目：

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

5. 创建 **velero** 服务帐户：

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

6. 将 **SERVICE_ACCOUNT_EMAIL** 变量设置为服务帐户的电子邮件地址：

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

7. 向服务帐户授予权限：

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

```
)

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

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

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

8. 将服务帐户的密钥保存到当前目录中的 **credentials-velero** 文件中：

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

3.3.4. 将 Microsoft Azure Blob 存储容器配置为复制存储库

您可以将 Microsoft Azure Blob 存储容器配置为复制存储库。

先决条件

- 您必须具有 Azure 存储帐户。
- 您必须安装了 Azure CLI。
- Azure Blob 存储容器必须可以被源和目标集群访问。
- 如果您使用快照复制方法：
  - 源和目标集群必须位于同一区域。
  - 源和目标集群必须具有相同的存储类。
  - 存储类必须与快照兼容。

流程

1. 设置 **AZURE_RESOURCE_GROUP** 变量：

```
$ AZURE_RESOURCE_GROUP=Velero_Backups
```

2. 创建 Azure 资源组：

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

1 指定位置。

3. 设置 **AZURE_STORAGE_ACCOUNT_ID** 变量：

```
$ AZURE_STORAGE_ACCOUNT_ID=velerobackups
```

4. 创建 Azure 存储帐户：

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

5. 设置 **BLOB_CONTAINER** 变量：

```
$ BLOB_CONTAINER=velero
```

6. 创建 Azure Blob 存储容器：

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

7. 为 **velero** 创建服务主体和凭证：

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

8. 在 **credentials-velero** 文件中保存服务主体的凭证：

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

3.4. 使用 CAM WEB 控制台迁移应用程序

您可以通过在 CAM web 控制台中添加集群和复制存储库来迁移应用程序工作负载。然后，您可以创建并运行迁移计划。

如果集群或复制存储库有自签名证书保护，您可以创建 CA 证书捆绑包文件或禁用 SSL 验证。

3.4.1. 创建 CA 证书捆绑包文件

如果您使用自签名证书来保护集群或复制存储库的安全，则证书验证可能会失败，出错信息如下：**Certificate signed by unknown authority**。

您可以创建自定义 CA 证书捆绑包文件，并在添加集群或复制存储库时将其上传到 CAM web 控制台。

流程

从远程端点下载 CA 证书，并将其保存为 CA 捆绑包文件：

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

1. 指定端点的主机 FQDN 和端口，如 **api.my-cluster.example.com:6443**。
2. 指定 CA 捆绑包文件的名称。

3.4.2. 把集群添加到 CAM web 控制台中

您可以将源集群添加到 CAM web 控制台。

先决条件

如果要使用 Azure 快照复制数据：

- 在添加源集群时，您必须提供 Azure 资源组名称。
- 源和目标集群必须位于同一 Azure 资源组且位于同一位置。

流程

1. 登录到集群。
2. 获取服务帐户令牌：

```
$ oc sa get-token mig -n openshift-migration
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwi
a3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJtaWciLCJrdWJlcm5ldGVz
LmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoibWlnLXRva2VuLWs4dDJyIiwia3ViZXJuZ
XRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6Im1pZyIsImt1YmV
ybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImE1YjFiYWM
wLWMxYmYtMTFlOS05Y2NiLTAyOWRmODYwYjMwOCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlY
WNjb3VudDptaWc6bWlnIn0.xqeeAINK7UXpdRqAtOj70qhBJPeMwmgLomV9iFxr5RoqUgKchZ
RG2J2rkqmPm6vr7K-
cm7ibD1IBpdQJCcVDuoHYsFgV4mp9vgOfn9osSDp2TGikwNz4Az95e81xnjVUmzh-
NjDsEpw71DH92iHV_xt2sTwtzftS49LpPW2LjrV0evtNBP_t_RfskdArt5VSv25eORl7zScqfe1CiM
kcVbf2UqACQjo3LbkpfN26HAioO2oH0ECPiRzT0Xyh-KwFutJLS9Xgghyw-
LD9kPKcE_xbbJ9Y4Rqajh7WdPYuB0Jd9DPVrsImzK-F6cgHHYoZEv0SvLQi-
PO0rpDrcjOEQQ
```

3. 登录到 CAM web 控制台。
4. 在 **Clusters** 部分，点 **Add cluster**。
5. 填写以下字段：
   - **Cluster name**：可包括小写字母（**a-z**）和数字（**0-9**）。不能包含空格或国际字符。
   - **URL**：集群 API 服务器的 URL，如 **https://<master1.example.com>:8443**。

- **Service account token**：从源集群获取的字符串。
- **Azure cluster**：可选。如果要使用 Azure 快照复制数据，请选择此项。
- **Azure resource group**：如果选中了 **Azure cluster**，则会出现此字段。
- 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 CA 捆绑包文件。

6. 点 **Add cluster**。
   集群会出现在 **Clusters** 部分。

3.4.3. 在 CAM web 控制台中添加复制程序库

您可以将对象存储桶作为复制存储库添加到 CAM web 控制台。

先决条件

- 您必须配置用于迁移数据的对象存储桶。

流程

1. 登录到 CAM web 控制台。
2. 在 **Replication repositories** 部分，点 **Add repository**。
3. 选择 **Storage provider type** 并填写以下字段：
   - **AWS** 适用于 S3、MCSG 和通用 S3 供应商：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **S3 bucket name**：指定您创建的 S3 存储桶的名称。
     - **S3 bucket region**：指定 S3 存储桶区域。AWS S3 **必填**。**Optional** 用于其他 S3 供应商。
     - **S3 端点**：指定 S3 服务的 URL，而不是存储桶，例如：**https://<s3-storage.apps.cluster.com>**。通用 S3 供应商**必填**。您必须使用 **https://** 前缀。
     - **S3 provider access key**：为 AWS 指定 **<AWS_SECRET_ACCESS_KEY>**，或者为 MCG 指定 S3 供应商访问密钥。
     - **S3 provider secret access key**：为 AWS 指定 **<AWS_ACCESS_KEY_ID>**，或者为 MCG 指定 S3 供应商 secret 访问密钥。
     - **Require SSL verification**：如果您使用的是通用 S3 供应商，则清除此复选框。
     - 如果您使用自定义 CA 捆绑包，请点击 **Browse** 并浏览到所需的 Base64 编码的 CA 捆绑包文件。
   - **GCP**：
     - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
     - **GCP bucket name**：指定 GCP 存储桶的名称。
     - **GCP credential JSON blob**：在 **credentials-velero** 文件中指定字符串。

- Azure：
  - **Replication repository name**：指定 CAM web 控制台中的复制存储库。
  - **Azure resource group**：指定 Azure Blob 存储的资源组。
  - **Azure storage account name**：指定 Azure Blob 存储帐户名称
  - **Azure credentials - INI file contents**：在 **credentials-velero** 文件中指定字符串。

4. 点 **Add repository** 并等待连接验证。
5. 点 **Close**。
   新存储库会出现在 **Replication repositories** 部分。

3.4.4. 为大型迁移修改迁移计划限制

您可以更改大型迁移的迁移计划限制。

重要

您需要首先在自己的环境对所做的更改进行测试，以避免迁移失败。

单个迁移计划有以下默认限制：

- 10 个命名空间
  如果超过这个限制，CAM web 控制台会显示一个 **Namespace limit exceeded**错误，您将无法创建迁移计划。
- 100 个 Pod
  如果超过 Pod 限制，CAM web 控制台会显示类似以下示例的警告信息: **Plan has been validated with warning condition(s).查看警告信息。pod limit: 100 exceeded, found: 104**。
- 100 个持久性卷（PV）
  如果超过持久性卷限制，则 CAM web 控制台会显示类似的警告信息。

流程

1. 编辑迁移控制器 CR：

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

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

2. 更新以下参数：

```
...
migration_controller: true

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

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

3.4.5. 在 CAM web 控制台中创建迁移计划

您可以在 CAM web 控制台中创建迁移计划。

先决条件

- CAM web 控制台必须包含以下内容：
  - 源集群
  - 目标集群，它会在 CAM 工具安装过程中自动添加
  - 复制软件仓库
- 源和目标集群必须可以通过网络相互访问，并可以访问复制存储库。
- 如果要使用快照复制数据，则源和目标集群必须在同一云供应商（AWS、GCP 或 Azure）以及同一区域中。

流程

1. 登录到 CAM web 控制台。
2. 在 **Plans** 部分，点 **Add Plan**。
3. 输入 **Plan name** 并点 **Next**。
   Plan name 最多可包含 253 个小写字母数字字符（**a-z，0-9**）。它不能包含空格或下划线（_）。
4. 选一个 **Source cluster**。
5. 选一个 **Target cluster**。
6. 选一个 **Replication repository**。
7. 选择要迁移的项目并点 **Next**。
8. 选择 **Copy** 或 **Move** PV：
   - **Copy** 将源集群的 PV 中的数据复制到复制存储库中，然后在目标集群中新创建的具有类似特征的 PV 上恢复它。
     可选： 您可以通过选择 **Verify copy**来使用文件系统的方法来验证复制的数据。这个选项为每个源文件生成 checksum 并在恢复后对其进行检查。这可能会大大降低性能。
   - **Move** 从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。
9. 点 **Next**。
10. 为 PV 选择 **Copy method**：
    - **Snapshot** 使用云供应商的快照功能备份和恢复磁盘。它比 **Filesystem** 快得多。

注意

存储和集群必须位于同一区域，存储类必须兼容。

- **Filesystem** 将源磁盘中的数据文件复制到新创建的目标磁盘。

11. 为 PV 选择一个 **Storage class**。
如果选择了 **Filesystem** 复制方法，您可以在迁移过程中更改存储类，例如：从 Red Hat Gluster Storage 或 NFS 存储改为 Red Hat Ceph Storage。

12. 点 **Next**。

13. 如果您要添加迁移 hook，请点击 **Add Hook** 并执行以下步骤：

    a. 指定存储桶的名称。

    b. 选择 **Ansible playbook** 来使用您自己的 playbook 或 **Custom container image** 来使用以其他语言编写的 hook。

    c. 点击 **Browse** 上传 playbook。

    d. 可选： 如果您未使用默认 Ansible 运行时镜像，请指定自定义 Ansible 镜像。

    e. 指定要运行 hook 的集群。

    f. 获取服务帐户名称。

    g. 指定命名空间。

    h. 选择您希望 hook 运行的迁移步骤：

       - PreBackup: 在源集群上启动备份前的任务
       - PostBackup: 在源集群中完成备份任务后
       - PreRestore: 在目标集群上启动恢复前的任务
       - PostRestore: 在目标集群中完成恢复后的任务

14. 点 **Add**。
您可以在迁移计划中添加最多四个 hook，将每个 hook 分配给不同的迁移步骤。

15. 点 **Finish**。

16. 点 **Close**。
迁移计划会出现在 **Plans** 部分。

3.4.6. 在 CAM web 控制台中运行迁移计划

您可以使用在 CAM web 控制台中创建的迁移计划来 stage 或迁移应用程序和数据。

先决条件

CAM web 控制台必须包含以下内容：

- 源集群

- 目标集群，它会在 CAM 工具安装过程中自动添加
- 复制软件仓库
- 有效的迁移计划

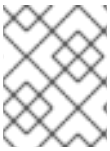
流程

1. 登录目标集群上的 CAM web 控制台。
2. 选择迁移计划。
3. 点 **Stage** 以在不停止应用程序的情况下，将数据从源集群复制到目标集群。
   您可以多次运行 **Stage** 以减少实际迁移时间。
4. 当准备好迁移应用程序工作负载时，点 **Migrate**。
   Migrate 在源集群中停止应用程序工作负载，并在目标集群中重新创建其资源。
5. 另外，还可以在 **Migrate** 窗口中选择 **Do not stop applications on the source cluster during migration**。
6. 点 **Migrate**。
7. 可选： 要停止迁移过，请点击 Options 菜单  并选择 **Cancel**。
8. 迁移完成后，在 OpenShift Container Platform web 控制台中确认已成功迁移了应用程序：
   a. 点 **Home → Projects**。
   b. 点迁移的项目查看其状态。
   c. 在 **Routes** 部分，点击 **Location** 验证应用程序是否正常运行。
   d. 点 **Workloads → Pods** 来验证 Pod 在迁移的命名空间中运行。
   e. 点 **Storage → Persistent volumes** 确认正确置备了被迁移的持久性卷。

3.5. 故障排除

您可以查看迁移自定义资源 (CR)，并下载日志来排除迁移失败的问题。

如果应用程序在迁移失败时停止，您必须手动回滚，以防止数据崩溃。

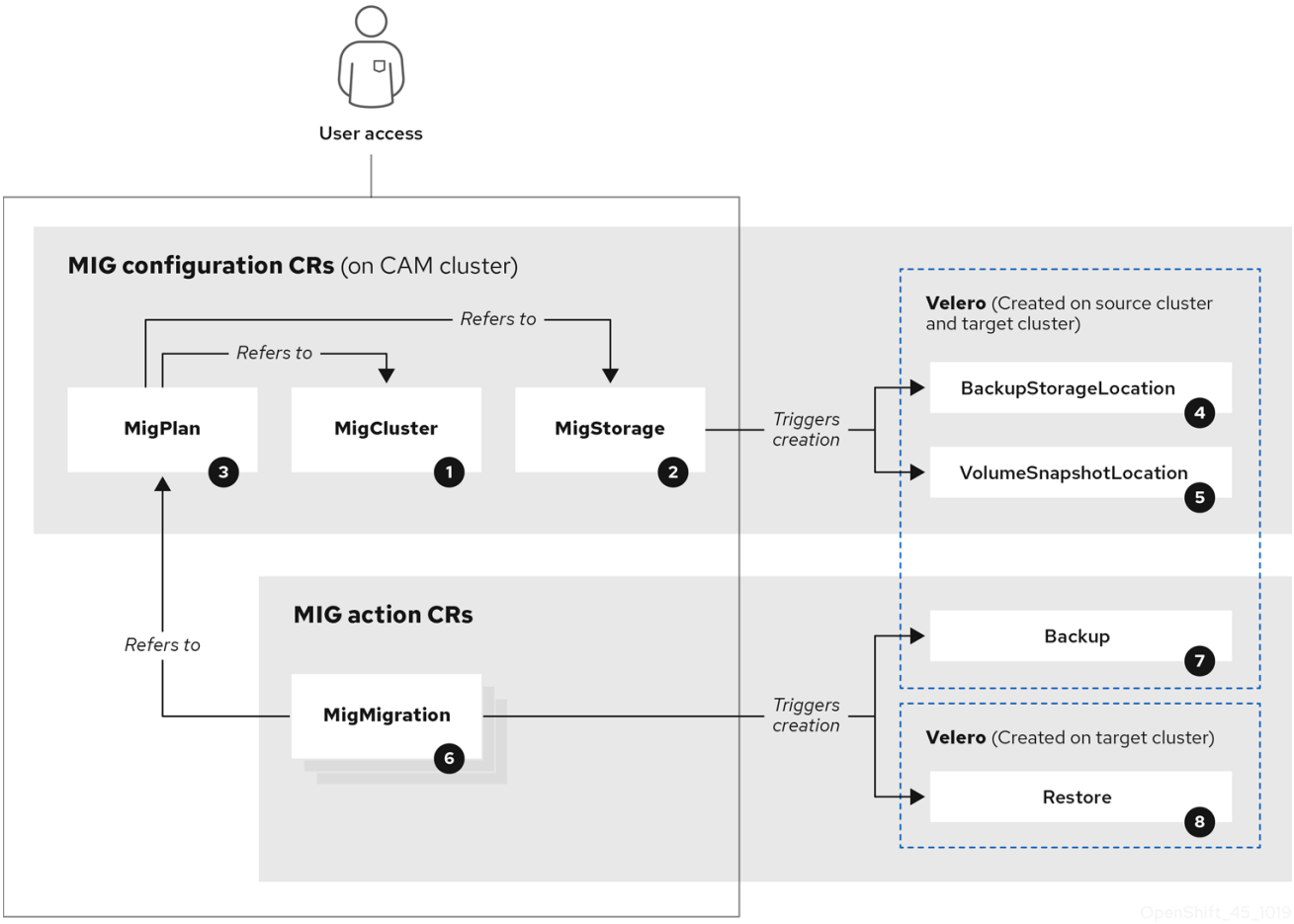


注意

如果应用程序在迁移过程中没有停止，则不需要手动回滚，因为原始应用程序仍然在源集群中运行。

3.5.1. 查看迁移自定义资源

集群应用程序迁移 (CAM) 工具会创建以下 CR：

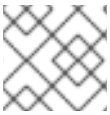


1 MigCluster （配置，CAM 集群）：集群定义

2 MigStorage （配置，CAM 集群）：存储定义

3 MigPlan （配置，CAM 集群）：迁移计划

MigPlan CR 描述了要迁移的源和目标集群、存储库和命名空间。它与 0 个、1 个或多个 MigMigration CR 关联。



注意

删除 MigPlan CR 会删除关联的 MigMigration CR。

4 BackupStorageLocation （配置，CAM 集群）：Velero 备份对象的位置

5 VolumeSnapshotLocation （配置，CAM 集群）：Velero 卷快照的位置

6 MigMigration（操作，CAM 集群）：Migration，在迁移期间创建

在每次进行 stage 或迁移数据时都会创建一个 MigMigration CR。每个 MigMigration CR 都会与一个 MigPlan CR 关联。

7 Backup（操作，源集群）：当运行迁移计划时，MigMigration CR 在每个源集群上创建两个 Velero 备份 CR：

- 备份 CR #1 用于Kubernetes 对象
- 备份 CR #2 用于 PV 数据

8 Restore （操作，目标集群）：在运行迁移计划时，MigMigration CR 在目标集群上创建两个 Velero 恢复 CR：

- 恢复 CR #1（使用备份 CR #2）用于 PV 数据
- 恢复 CR #2（使用备份 CR #1）用于 Kubernetes 对象

流程

1. 获取 CR 名称：

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

1 指定迁移 CR，如 **migmigration**。

输出结果类似以下：

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

2. 查看 CR：

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

输出结果类似以下示例。

MigMigration 示例

```
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>
```

Velero 备份 CR #2 示例（PV 数据）

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

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

Velero 恢复 CR #2 示例（Kubernetes 资源）

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

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

3.5.2. 下载迁移日志

您可以在 CAM web 控制台中下载 Velero 、 Restic 和 Migration controller 日志，以排除出现故障的迁移问题。

流程

1. 登录到 CAM 控制台。
2. 点击 **Plans** 查看迁移计划列表。
3. 点击特定迁移计划  的 **Options** 菜单并选择 **Logs**。
4. 点 **Download Logs** 为所有集群下载迁移控制器、Velero 和 Restic 的日志。
5. 要下载特定的日志：
   a. 指定日志选项：
      - **Cluster**：选择源、目标或 CAM 主机集群。
      - **Log source**：选择 **Velero**、**Restic** 或 **Controller**。
      - **Pod source**：选择 Pod 名称，例如：**controller-manager-78c469849c-v6wcf** 此时会显示所选日志。

        您可以通过更改您的选择来清除日志选择设置。

   b. 点 **Download Selected** 下载所选日志。

另外，您可以使用 CLI 访问日志，如下例所示：

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

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

3.5.3. 错误信息

3.5.3.1. Velero Pod 日志中的 Restic 超时错误消息

如果因为 Restic 超时造成迁移失败，以下出错信息会出现在 Velero Pod 日志中：

```
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 的默认值为一小时。您可以为大型迁移增加这个参数值，请注意，高的值可能会延迟返回出错信息。

流程

1. 在 OpenShift Container Platform web 控制台中导航至 **Operators → Installed Operators**。
2. 点 **Cluster Application Migration Operator**。
3. 在 **MigrationController** 标签页中点 **migration-controller**。
4. 在 **YAML** 标签页中，更新以下参数值：

```
spec:
  restic_timeout: 1h (1)
```

(1) 有效单元是 **h** （小时）、**m** （分钟）和 **s** （秒），例如 **3h30m15s**。

5. 点 **Save**。

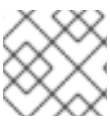
3.5.3.2. MigMigration Custom Resource（CR）中的 ResticVerifyErrors

如果迁移使用文件系统数据复制方法的 PV 时数据验证失败，在 MigMigration 自定义资源 (CR) 中会出现以下错误：

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

(1) 错误消息标识了 Restore CR 名称。

(2) **ResticErrors** 也会出现。**ResticErrors** 是一个包括验证错误的一般错误警告。



注意

数据验证错误不会导致迁移过程失败。

您可以检查目标集群的 Restore CR，以识别数据验证错误的来源。

流程

1. 登录到目标集群。
2. 查看 Restore CR:

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

输出通过 **PodVolumeRestore** 错误来指定 PV：

```
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. 查看 **PodVolumeRestore** CR:

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

输出中标识了记录错误的 Restic Pod:

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

4. 查看 Restic Pod 日志：

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

3.5.4. 手动回滚迁移

如果您的应用程序在迁移失败时停止，您必须手动回滚，以防止 PV 中的数据被破坏。

如果应用程序在迁移过程中没有停止，则不需要进行手动回滚，因为原始应用程序仍然在源集群中运行。

流程

1. 在目标集群中，切换到迁移的项目：

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

2. 获取部署的资源：

```
$ oc get all
```

3. 删除部署的资源以确保应用程序没有在目标集群中运行，并访问 PVC 上的数据：

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

4. 要停止 DaemonSet 而不删除它，在 YAML 文件中更新 **nodeSelector** ：

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

1 指定一个没有存在于任何节点上的**nodeSelector** 值。

5. 更新每个 PV 的重新声明策略，以便删除不必要的数据。在迁移过程中，绑定 PV 的重新声明策略是**reclaim**，以确保应用程序从源集群中被删除时不会丢失数据。您可以在回滚过程中删除这些 PV。

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

1 指定 **Recycle** 或 **Delete**。

6. 在源集群中，切换到迁移的项目：

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

7. 获取项目部署的资源：

```
$ oc get all
```

8. 启动每个部署资源的一个或多个副本：

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

9. 如果在操作中被更改了，把 DaemonSet 的 **nodeSelector** 改回其原始值。

3.5.5. 为客户支持问题单收集数据

如果创建一个客户支持问题单，您可以使用 **openshift-migration-must-gather-rhel8** 镜像的 **must-gather** 工具来收集与您的集群相关的信息，并把这些信息上传到红帽客户门户网站。

openshift-migration-must-gather-rhel8 镜像会收集默认的 **must-gather** 镜像不收集的日志和 CR 数据。

流程

1. 进入要存储 **must-gather** 数据的目录。
2. 运行 **oc adm must-gather** 命令：

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

must-gather 工具程序收集集群数据，并把它保存在 **must-gather.local.<uid>** 目录中。

3. 从 **must-gather** 数据中删除验证密钥和其他敏感信息。
4. 创建一个包含 **must-gather.local.<uid>** 目录内容的归档文件：

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

在红帽客户门户中，为您的问题单附上这个压缩文件。

3.5.6. 已知问题

这个版本有以下已知问题：

- 在迁移过程中，集群应用程序迁移 (CAM) 工具会保留以下命名空间注解：
  - **openshift.io/sa.scc.mcs**
  - **openshift.io/sa.scc.supplemental-groups**
  - **openshift.io/sa.scc.uid-range**
    这些注解会保留 UID 范围，确保容器在目标集群中保留其文件系统权限。这可能会存在一定的风险。因为迁移的 UID 可能已存在于目标集群的现有或将来的命名空间中。
    (BZ#1748440)
- 如果一个 AWS 存储桶被添加到 CAM web 控制台，然后将其删除，则其状态会保持为 **True**，这是因为 MigStorage CR 没有被更新。(BZ#1738564)
- 大多数集群范围的资源尚未由 CAM 工具处理。如果应用程序需要集群范围的资源，则可能需要在目标集群上手动创建。
- 如果迁移失败，则迁移计划不会为静默的 pod 保留自定义 PV 设置。您必须手动回滚，删除迁移计划，并使用 PV 设置创建新的迁移计划。(BZ#1784899)

- 如果因为 Restic 超时造成大型迁移失败，您可以提高 Migration controller CR 中的 **restic_timeout** 参数值。
- 如果您选择了为使用文件系统复制方法迁移的 PV 数据进行验证的选项，则性能会非常慢。Velero 为每个文件生成一个 checksum，并在恢复该文件时对其进行检查。