OpenShift Container Platform 4.8

从版本 3 迁移到 4

迁移到 OpenShift Container Platform 4

Last Updated: 2023-04-19

OpenShift Container Platform 4.8 从版本 3 迁移到 4

迁移到 OpenShift Container Platform 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 迁移到 4 概述

OpenShift Container Platform 4 集群与 OpenShift Container Platform 3 集群不同。OpenShift Container Platform 4 集群包含新的技术和功能，可使集群具有自我管理、灵活和自动化的。如需了解更多有关从 OpenShift Container Platform 3 迁移到 4 的信息，请参阅关于从 OpenShift Container Platform 3 迁移到 4。

1.1. OPENSHIFT CONTAINER PLATFORM 3 和 4 之间的区别

在从 OpenShift Container Platform 3 迁移到 4 之前，您可以检查 OpenShift Container Platform 3 和 4 之间的区别。请查看以下信息：

- 构架
- 安装和更新
- 存储、网络、日志记录、安全性和监控注意事项

1.2. 规划网络注意事项

在从 OpenShift Container Platform 3 迁移到 4 之前，请查看 OpenShift Container Platform 3 和 4 之间的不同：

- DNS 注意事项
  - 将目标集群的 DNS 域与客户端隔离。
  - 将目标集群设置为接受源 DNS 域。

您可以通过命名空间将有状态应用程序工作负载从 OpenShift Container Platform 3 迁移到 4。如需了解更多有关 MTC 的信息，请参阅了解 MTC。

注意

如果要从 OpenShift Container Platform 3 迁移，请参阅关于从 OpenShift Container Platform 3 迁移到 4以及在 OpenShift Container Platform 3 上安装旧的 MTC Operator。

1.3. 安装 MTC

查看以下任务来安装 MTC：

1. 使用 Operator Lifecycle Manager(OLM)在目标集群中安装 MTC Operator。
2. 在源集群中手动安装旧的 MTC Operator。
3. 配置对象存储，以用作复制存储库。

1.4. 升级 MTC

您可以使用 OLM 在 OpenShift Container Platform 4.8 上升级 Migration Toolkit for Containers (MTC)。您可以通过重新安装旧的 Migration Toolkit for Containers Operator 在 OpenShift Container Platform 3 上升级 MTC。

1.5. 查看预迁移清单

在使用 Migration Toolkit for Containers(MTC)迁移应用程序工作负载前，请查看迁移前检查。

1.6. 迁移应用程序

您可以使用 MTC web 控制台或命令行迁移应用程序。

1.7. 高级迁移选项

您可以使用以下选项自动化迁移和修改 MTC 自定义资源，以提高大型迁移的性能：

- 运行状态迁移
- 创建迁移 hook
- 编辑、排除和映射迁移的资源
- 为大型迁移配置迁移控制器

1.8. 迁移故障排除

您可以执行以下故障排除任务：

- 使用 MTC web 控制台查看迁移计划资源
- 查看迁移计划聚合日志文件
- 使用迁移日志读取器
- 访问性能指标
- 使用 **must-gather** 工具
- 使用 Velero CLI 调试 **Backup** 和 **Restore** CR
- 使用 MTC 自定义资源进行故障排除
- 检查常见问题

1.9. 回滚一个迁移

您可以使用 MTC web 控制台、CLI 或手动回滚迁移。

1.10. 卸载 MTC 并删除资源

您可以卸载 MTC 并删除其资源来清理集群。

第 2 章 关于从 OPENSHIFT CONTAINER PLATFORM 3 迁移到 4

OpenShift Container Platform 4 包含新的技术和功能，可使集群具有自我管理、灵活性和自动化的特点。OpenShift Container Platform 4 集群的部署和管理与 OpenShift Container Platform 3 不同。

从 OpenShift Container Platform 3 迁移到 4 的最有效方法是使用 CI/CD 管道在应用程序生命周期管理框架中自动执行部署。

如果您没有 CI/CD 管道，或者正在迁移有状态应用程序，您可以使用 Migration Toolkit for Containers (MTC) 迁移应用程序工作负载。

要成功过渡到 OpenShift Container Platform 4，请查看以下信息：

OpenShift Container Platform 3 和 4 之间的区别

- 构架
- 安装和升级
- 存储、网络、日志记录、安全性和监控注意事项

关于 Migration Toolkit for Containers（MTC）

- 工作流
- 持久性卷 (PV) 的文件系统和快照复制方法
- 直接卷迁移
- 直接镜像迁移

高级迁移选项

- 使用迁移 hook 自动迁移
- 使用 MTC API
- 从迁移计划中排除资源
- 为大规模迁移配置 **MigrationController** 自定义资源
- 为直接卷迁移启用自动 PV 大小调整
- 启用缓存的 Kubernetes 客户端以提高性能

有关新功能、增强功能、技术更改以及已知的问题，请参阅 MTC 发行注记。

第 3 章 OPENSHIFT CONTAINER PLATFORM 3 和 4 之间的区别

OpenShift Container Platform 4.8 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。

有关配置 OpenShift Container Platform 4 集群的详情，请查看 OpenShift Container Platform 文档的相关章节。如需有关新功能和其他显著技术更改的信息，请参阅 OpenShift Container Platform 4.8 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。

3.1. 构架

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、机器集和 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。

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（RHCOS）。

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 Operator。

3.2. 安装和升级

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。

在 OpenShift Container Platform 4.8 中，您可以使用 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 Red Hat Enterprise Linux（RHEL）worker 机器添加到 OpenShift Container Platform 4.8 集群，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 4.8 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 4.8 集群有 RHEL 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。

3.3. 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。

3.3.1. 存储注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.8 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 4.8 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。OpenShift Container Platform 4.8 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 4.8 提供了 几个 CSI 驱动程序。您还可以安装自己的驱动程序。

如需更多信息，请参阅使用 Container Storage Interface(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 的持久性存储 和互操作性文档。

可用的持久性存储选项
OpenShift Container Platform 4.8 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 4.8 中获得全面支持。

如需更多信息，请参阅了解持久性存储。

3.3.2. 网络注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.8 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 4.8 的默认网络隔离模式由网络策略控制。

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 4.8 集群中使用网络策略。网络策略由上游社区支持，它更灵活并提供 **ovs-multitenant** 的功能。如果您要在 OpenShift Container Platform 4.8 中使用网络策略时仍然希望维持 **ovs-multitenant** 的行为，请按照以下步骤使用网络策略配置多租户隔离。

如需更多信息，请参阅关于网络策略。

3.3.3. 日志记录注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.8 时，请考虑以下与日志相关的变化。

部署 OpenShift Logging
OpenShift Container Platform 4 通过使用集群日志记录自定义资源为 OpenShift Logging 提供了一个简单的部署机制。

如需更多信息，请参阅安装 OpenShift Logging。

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 OpenShift Logging。

不支持的日志配置
OpenShift Container Platform 4.8 不再支持 OpenShift Container Platform 3.11 中的一些日志记录配置。

有关明确不支持的日志问题单的更多信息，请参阅维护和支持。

3.3.4. 安全考虑

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.8 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 4.8 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 4.8 中的请求标头身份提供程序需要 mutual TLS，而这在 OpenShift Container Platform 3.11 中不需要。
- OpenShift Container Platform 4.8 中简化了 OpenID Connect 身份提供程序的配置。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，请参阅了解身份提供程序配置。

OAuth 令牌存储格式
新创建的 OAuth HTTP 持有者令牌不再匹配其 OAuth 访问令牌对象的名称。对象名称现在是一个 bearer 令牌哈希，它不再敏感。这可减少泄漏敏感信息的风险。

3.3.5. 监控注意事项

在从 OpenShift Container Platform 3.11 转换到 OpenShift Container Platform 4.8 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。

第 4 章 网络注意事项

检查迁移后用于重定向应用程序网络流量的策略。

4.1. DNS 注意事项

目标集群的 DNS 域与源集群的域不同。默认情况下，应用程序在迁移后获取目标集群的 FQDN。

要保留迁移的应用程序的源 DNS 域，请选择下面描述的两个选项之一。

4.1.1. 将目标集群的 DNS 域与客户端隔离

您可以允许发送到源集群的 DNS 域的客户端请求访问目标集群的 DNS 域，而无需将目标集群公开给客户端。

流程

1. 将外部网络组件（如应用程序负载均衡器或反向代理）放在客户端和目标集群之间。
2. 更新 DNS 服务器上的源集群中的应用程序 FQDN，以返回 exterior 网络组件的 IP 地址。
3. 配置网络组件，将源域中为应用接收的请求发送到目标集群域中的负载均衡器。
4. 为 ***.apps.source.example.com** 域创建一个通配符 DNS 记录，指向源集群的负载均衡器的 IP 地址。
5. 为每个应用程序创建一个 DNS 记录，指向目标集群前面的 exterior 网络组件的 IP 地址。特定的 DNS 记录的优先级高于通配符记录，因此在解决应用 FQDN 时不会发生冲突。

注意

- 外部网络组件必须终止所有安全的 TLS 连接。如果连接传递给目标集群负载均衡器，目标应用程序的 FQDN 会公开给客户端，证书发生错误。
- 应用程序不得将引用目标集群域的链接返回给客户端。否则，应用的某些部分可能无法加载或正常工作。

4.1.2. 设置目标集群以接受源 DNS 域

您可以设置目标集群，以接受源集群的 DNS 域中迁移的应用程序的请求。

流程

对于非安全 HTTP 访问和安全 HTTPS 访问，请执行以下步骤：

1. 在目标集群的项目中创建一个路由，该路由配置为接受源集群中处理的应用程序 FQDN 的请求：

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

新路由就位后，服务器接受对该 FQDN 的任何请求，并将它发送到对应的应用容器集。另外，当迁移应用程序时，会在目标集群域中创建另一个路由。请求会使用这些主机名之一到达迁移的应用。

2. 使用您的 DNS 供应商创建 DNS 记录，将源集群中的应用的 FQDN 指向目标集群的默认负载均衡器的 IP 地址。这会将来自源集群的流量重定向到目标集群。
   应用程序的 FQDN 解析到目标集群的负载均衡器。默认入口控制器路由器接受对该 FQDN 的请求，因为公开了该主机名的路由。

对于安全 HTTPS 访问，请执行以下步骤：

1. 将在安装过程中创建的默认入口控制器的 x509 证书替换为自定义证书。
2. 将这个证书配置为在 **subjectAltName** 字段中为源和目标集群包含通配符 DNS 域。
   新证书对于保护使用 DNS 域进行的连接有效。

其他资源

- 如需更多信息，请参阅替换默认入口证书。

4.2. 网络流量重定向策略

迁移成功后，您必须将无状态应用的网络流量从源集群重定向到目标集群。

重定向网络流量的策略基于以下假设：

- 应用程序 pod 在源集群和目标集群上运行。
- 每个应用都有一个包含源集群主机名的路由。
- 源集群主机名的路由包含 CA 证书。
- 对于 HTTPS，目标路由器 CA 证书包含用于源集群的通配符 DNS 记录的 Subject 备用名称。

考虑以下策略并选择符合您目标的策略。

- 同时重定向所有应用的所有网络流量
  更改源集群的通配符 DNS 记录，使其指向目标集群路由器的虚拟 IP 地址 (VIP)。

  此策略适用于简单应用程序或小型环境的迁移。

- 为单个应用重定向网络流量
  使用指向目标集群路由器 VIP 的源集群主机名为每个应用程序创建一个 DNS 记录。这个 DNS 记录优先于源集群通配符 DNS 记录。

- 为单个应用逐步重定向网络流量

  1. 创建一个代理，用于将流量定向到每个应用程序的源集群路由器的 VIP 和目标集群路由器的 VIP。
  2. 使用指向代理的源集群主机名为每个应用程序创建一个 DNS 记录。
  3. 配置应用的代理条目，将流量百分比路由到目标集群路由器的 VIP，并将其余流量路由到源集群路由器的 VIP。
  4. 逐渐增加您路由到目标集群路由器 VIP 的流量百分比，直到所有网络流量被重定向为止。

- 基于用户的单个应用程序流量重定向
  使用此策略，您可以过滤用户请求的 TCP/IP 标头，以便为预定义的用户组重定向网络流量。这允许您在重定向整个网络流量之前测试用户特定版本的重定向过程。

1. 创建一个代理，用于将流量定向到每个应用程序的源集群路由器的 VIP 和目标集群路由器的 VIP。

2. 使用指向代理的源集群主机名为每个应用程序创建一个 DNS 记录。

3. 配置应用的代理条目，将匹配给定标头模式的流量（如 **测试客户** ）路由到目标集群路由器的 VIP，并将其余流量路由到源集群路由器的 VIP。

4. 将流量分阶段重定向到目标集群路由器的 VIP，直到所有流量都位于目标集群路由器的 VIP 上。

第 5 章 关于 MIGRATION TOOLKIT FOR CONTAINERS（MTC）

MTC (Migration Toolkit for Containers) 可让您以命名空间的粒度将有状态应用程序工作负载从 OpenShift Container Platform 3 迁移到 4.8。

重要

在开始迁移前，请查看 OpenShift Container Platform 3 和 4 之间的区别 。

MTC 提供了一个基于 Kubernetes 自定义资源的 web 控制台和 API，可帮助您控制迁移并最小化应用程序停机时间。

MTC 控制台默认安装在目标集群中。您可以配置 MTC Operator 在 OpenShift Container Platform 3 源集群或远程集群中安装控制台。

MTC 支持将数据从源集群迁移到目标集群的文件系统和快照数据复制方法。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。

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**。如果无法迁移服务目录资源，MTC 控制台会显示一条消息。

5.1. 术语

表 5.1. MTC 术语

术语	定义
源集群	从中迁移应用程序的集群。
目标集群[1]	将应用程序迁移到的集群。
复制软件仓库	用于在间接迁移过程中复制镜像、卷和 Kubernetes 对象的对象存储，或者用于直接卷迁移或直接镜像迁移期间 Kubernetes 对象的对象存储。 复制存储库必须可以被所有集群访问。
主机集群	运行 **migration-controller** pod 和 Web 控制台的集群。主机集群通常是目标集群，但这不是必需的。 主机集群不需要公开的 registry 路由来直接迁移镜像。
远程集群	远程集群通常是源集群，但这不是必需的。 远程集群需要一个包含 **migration-controller** 服务帐户令牌的 **Secret** 自定义资源。 远程集群需要一个公开的安全 registry 路由来直接迁移镜像。
间接迁移	镜像、卷和 Kubernetes 对象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。

术语	定义
直接卷迁移	持久性卷直接从源集群复制到目标集群。
直接镜像迁移	镜像直接从源集群复制到目标集群。
阶段迁移	在不停止应用程序的情况下，数据将复制到目标集群。 多次运行阶段迁移会缩短迁移的持续时间。
剪切迁移	应用在源集群中停止，其资源迁移到目标集群。
状态迁移	通过将特定的持久性卷声明复制到目标集群来迁移应用程序状态。
回滚迁移	回滚迁移会回滚一个已完成的迁移。

[1] 在 MTC web 控制台中称为 *目标集群*。

5.2. MTC 工作流

您可以使用 MTC web 控制台或 Kubernetes API 将 Kubernetes 资源、持久性卷数据和内部容器镜像迁移到 OpenShift Container Platform 4.8。

MTC 迁移以下资源：

- 在迁移计划中指定的命名空间。
- 命名空间范围的资源：当 MTC 迁移命名空间时，它会迁移与该命名空间关联的所有对象和资源，如服务或 Pod。另外，如果一个资源在命名空间中存在但不在集群级别，这个资源依赖于集群级别存在的另外一个资源，MTC 会迁移这两个资源。
  例如，安全性上下文约束（SCC）是一个存在于集群级别的资源，服务帐户（SA）是存在于命名空间级别的资源。如果 MTC 迁移的命名空间中存在 SA，MTC 会自动找到链接到 SA 的所有 SCC，并迁移这些 SCC。同样，MTC 会迁移链接到命名空间持久性卷的持久性卷声明。

  注意

  根据资源，可能需要手动迁移集群范围的资源。
- 自定义资源 (CR) 和自定义资源定义 (CRD)：MTC 在命名空间级别自动迁移 CR 和 CRD。

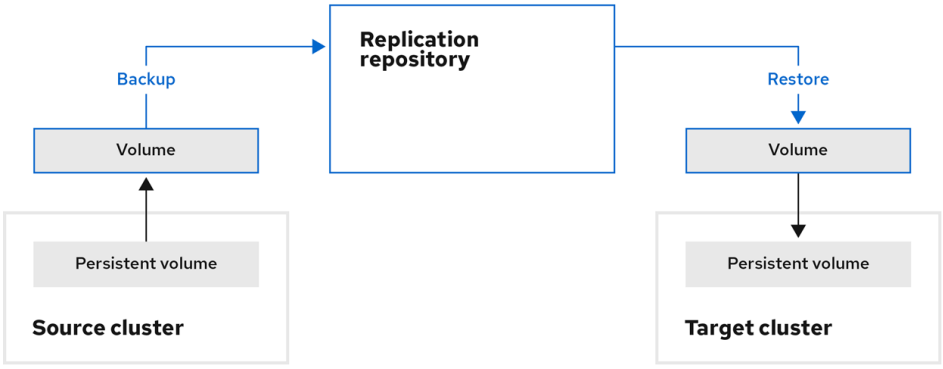
使用 MTC Web 控制台迁移应用程序涉及以下步骤：

1. 在所有集群中安装 MTC Operator。
   您可以在有限的或没有互联网访问的受限环境中为 Containers Operator 安装 Migration Toolkit。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 MTC 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仓库的不受限制的网络访问权限。在受限环境中，您可以使用 Multi-Cloud Object Gateway (MCG)。如果使用代理服务器，您必须将其配置为允许复制仓库和集群间的网络流量。

3. 在 MTC web 控制台中添加源集群。
4. 在 MTC 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**：MTC 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。

     注意

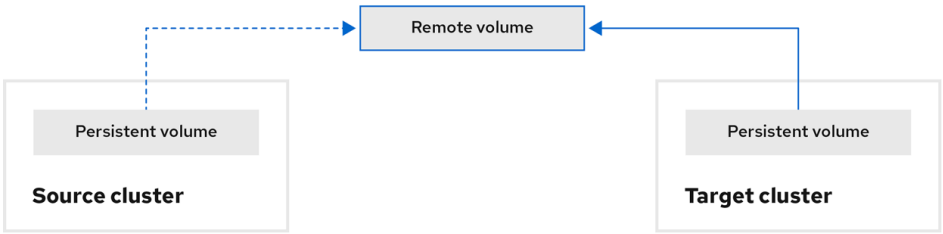
     如果您使用直接镜像迁移或直接卷迁移，则镜像或卷会直接从源集群复制到目标集群。



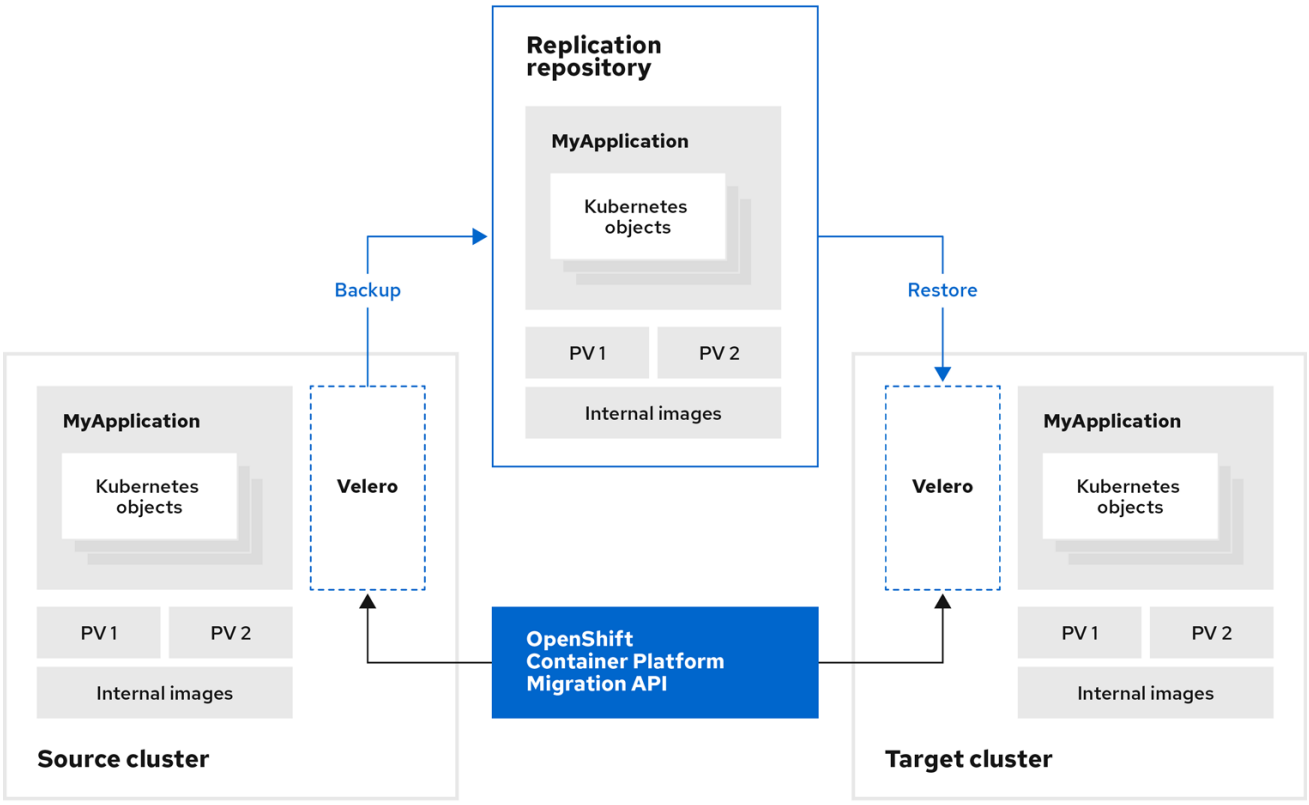
   - **Move**：MTC 从源集群中卸载一个远程卷（例如 NFS），在目标集群上创建一个指向这个远程卷的 PV 资源，然后在目标集群中挂载远程卷。在目标集群中运行的应用程序使用源集群使用的同一远程卷。远程卷必须可以被源集群和目标集群访问。

     注意

     虽然复制仓库没有出现在此图表中，但迁移需要它。



6. 运行迁移计划，使用以下选项之一：
   - **stage** 在不停止应用程序的情况下将数据复制到目标集群。
     阶段迁移可以多次运行，以便在迁移前将大多数数据复制到目标。运行一个或多个阶段迁移可缩短迁移的持续时间。
   - **cutover** 会停止源集群上的应用程序，并将资源移到目标集群。
     可选：您可以清除 **Halt transactions on the source cluster during migration**多选设置。



5.3. 关于数据复制方法

Migration Toolkit for Containers (MTC) 支持将数据从源集群迁移到目标集群的文件系统和快照数据复制方法。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。

5.3.1. 文件系统复制方法

MTC 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。

文件系统复制方法使用 Restic 进行间接迁移，或使用 Rsync 进行直接卷迁移。

表 5.2. 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 存储供应商都支持。 • 使用 checksum 验证数据（可选）。 • 支持直接卷迁移，这会显著提高性能。	• 比快照复制方法慢。 • 可选的数据校验可能会显著降低性能。

注意

Restic 和 Rsync PV 迁移假设支持的 PV 仅是 **volumeMode=filesystem**。*不支持*在文件系统迁移中使用 **volumeMode=Block**。

5.3.2. 快照复制方法

MTC 将源集群数据的快照复制到云供应商的复制仓库。数据在目标集群上恢复。

快照复制方法可用于 Amazon Web Services、Google Cloud Provider 和 Microsoft Azure。

表 5.3. 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。 • 不支持直接卷迁移。

5.4. 直接卷迁移和直接镜像迁移

您可以使用直接镜像迁移（DIM）和直接卷迁移（DVM）将镜像和数据直接从源集群迁移到目标集群。

如果您使用位于不同可用区的节点运行 DVM，迁移可能会失败，因为迁移的 pod 无法访问持久性卷声明。

DIM 和 DVM 具有显著的性能优势，因为跳过将文件从源集群备份到复制存储库以及从复制存储库恢复到目标集群的中间步骤。使用 Rsync 传输数据。

DIM 和 DVM 还有其他先决条件。

第 6 章 安装 MTC

您可以在 OpenShift Container Platform 3 和 4 上安装 MTC。

使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 上安装了 MTC Operator 后，您可以在 OpenShift Container Platform 3 上手动安装旧的 MTC Operator。

默认情况下，MTC web 控制台和 **Migration Controller** pod 在目标集群中运行。您可以配置 **Migration Controller** 自定义资源清单来在源集群或远程集群中运行 MTC web 控制台和 **Migration Controller** pod。

安装 MTC 后，您必须配置对象存储以用作复制存储库。

要卸载 MTC，请参阅卸载 MTC 并删除资源。

6.1. 兼容性指南

您必须安装与 OpenShift Container Platform 版本兼容的 MTC。

定义

旧平台

OpenShift Container Platform 4.5 及更早版本。

现代平台

OpenShift Container Platform 4.6 及更新的版本。

旧 Operator

针对传统平台设计的 MTC Operator。

现代 operator

针对现代平台设计的 MTC Operator。

控制集群

运行 MTC 控制器和 GUI 的集群。

远程集群

运行 Velero 的迁移的源或目标集群。Control Cluster 通过 Velero API 与远程集群通信，以驱动迁移。

表 6.1. MTC 兼容性：从传统平台迁移

	OpenShift Container Platform 4.5 或更早版本	OpenShift Container Platform 4.6 或更高版本
稳定 MTC 版本	MTC 1.7.*z* 旧版 1.7 运算符：使用 **operator.yml** 文件手动安装。 **重要** 此集群不能是控制集群。	MTC 1.7.*z* 使用 OLM 安装，发行频道 **release-v1.7**

注意

在某些情况下，网络的限制可能会阻止现代集群连接到迁移中需要涉及的其他集群。例如，当从内部的 OpenShift Container Platform 3.11 集群迁移到云环境中的现代 OpenShift Container Platform 集群时，，现代集群无法连接到 OpenShift Container Platform 3.11 集群。

对于 MTC 1.7，如果一个远程集群因为网络限制而无法与控制集群进行通信，请使用 **crane tunnel-api** 命令。

对于稳定（stable）的 MTC 发行版本，虽然您应该始终将最现代化的集群指定为控制集群，但是在这种情况下，可能需要将旧的集群指定为控制集群，并将工作负载推送到远程集群。

6.2. 在 OPENSHIFT CONTAINER PLATFORM 3 上安装旧的 MTC OPERATOR

您可以在 OpenShift Container Platform 3 上手动安装旧的 MTC Operator。

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。
- 您必须有权访问 **registry.redhat.io**。
- 必须安装 **podman** 。
- 您必须创建一个镜像流 secret，并将其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
```

2. 输入以下命令下载 **operator.yml** 文件：

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

3. 输入以下命令下载 **controller.yml** 文件：

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

4. 登录到您的源集群。
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 MTC Operator 对象的 Migration Toolkit:

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

输出示例

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

1 您可以忽略 **Error from server (AlreadyExists)** 信息。它们是由 MTC Operator 为早期版本的 OpenShift Container Platform 4 创建资源造成的，这些资源在以后的版本中已提供。

7. 创建 **MigrationController** 对象：

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

8. 验证 MTC Pod 是否正在运行：

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

6.3. 在 OPENSHIFT CONTAINER PLATFORM 4.8 上安装 MTC OPERATOR

您可以使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 上安装 MTC Operator。

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **MTCs Operator**。
3. 选择 **Migration Toolkit for Containers Operator** 并点 **Install**。
4. 点击 **Install**。
   在 **Installed Operators** 页中，**openshift-migration** 项目中会出现状态为 **Succeeded** 的 **Migration Toolkit for Containers Operator**。
5. 点 **Migration Toolkit for Containers Operator**。

6. 在 **Provided APIs** 下，找到 **Migration Controller** 标题，再点 **Create Instance**。

7. 点击 **Create**。

8. 点 **Workloads → Pods** 来验证 MTC pod 正在运行 。

6.4. 代理配置

对于 OpenShift Container Platform 4.1 及更早的版本，您必须在安装 Migration Toolkit for Containers Operator 后，在 **MigrationController** 自定义资源 (CR) 清单中配置代理，因为这些版本不支持集群范围的 **proxy** 对象。

对于 OpenShift Container Platform 4.2 到 4.8，MTC 会继承集群范围的代理设置。如果要覆盖集群范围的代理设置，可以更改代理参数。

6.4.1. 直接卷迁移

MTC 1.4.2 中引入了直接卷迁移(DVM)。DVM 只支持一个代理。如果目标集群也位于代理后面，则源集群无法访问目标集群的路由。

如果要从代理后面的源集群执行 DVM，您必须配置一个 TCP 代理，该代理可在传输层进行透明处理，并在不使用自己的 SSL 证书的情况下转发 SSL 连接。Stunnel 代理是此类代理的示例。

6.4.1.1. DVM 的 TCP 代理设置

您可以通过 TCP 代理在源和目标集群之间设置直接连接，并在 **MigrationController** CR 中配置 **stunnel_tcp_proxy** 变量来使用代理：

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

直接卷迁移(DVM)只支持代理的基本身份验证。此外，DVM 仅适用于可透明地传输 TCP 连接的代理。在 man-in-the-middle 模式中的 HTTP/HTTPS 代理无法正常工作。现有的集群范围的代理可能不支持此行为。因此，DVM 的代理设置意与 MTC 中常见的代理配置不同。

6.4.1.2. 为什么使用 TCP 代理而不是 HTTP/HTTPS 代理？

您可以通过 OpenShift 路由在源和目标集群之间运行 Rsync 来启用 DVM。流量通过 TCP 代理(Stunnel)加密。在源集群上运行的 Stunnel 会启动与目标 Stunnel 的 TLS 连接，并通过加密频道来传输数据。

OpenShift 中的集群范围 HTTP/HTTPS 代理通常在 man-in-the-middle 模式进行配置，其中它们将自己的 TLS 会话与外部服务器协商。但是，这不适用于 Stunnel。Stunnel 要求代理不处理它的 TLS 会话，基本上使代理成为一个透明的隧道，只需按原样转发 TCP 连接。因此，您必须使用 TCP 代理。

6.4.1.3. 已知问题

迁移失败并显示 Upgrade request required 错误

迁移控制器使用 SPDY 协议在远程 pod 中执行命令。如果远程集群位于代理或不支持 SPDY 协议的防火墙后，迁移控制器将无法执行远程命令。迁移失败并显示出错信息 **Upgrade request required**。临时解决方案：使用支持 SPDY 协议的代理。

除了支持 SPDY 协议外，代理或防火墙还必须将 **Upgrade** HTTP 标头传递给 API 服务器。客户端使用此标头打开与 API 服务器的 websocket 连接。如果代理或防火墙阻止 **Upgrade** 标头，则迁移会失败，并显示出错信息 **Upgrade request required**。临时解决方案：确保代理转发 **Upgrade** 标头。

6.4.2. 为迁移调优网络策略

OpenShift 支持根据集群使用的网络插件，限制使用 *NetworkPolicy* 或 *EgressFirewalls* 的流量。如果任何涉及迁移的源命名空间使用此类机制将网络流量限制到 pod，限制可能会在迁移过程中停止到 Rsync pod 的流量。

在源和目标集群上运行的 rsync pod 必须通过 OpenShift Route 相互连接。可将现有的 *NetworkPolicy* 或 *EgressNetworkPolicy* 对象配置为从这些流量限制自动排除 Rsync pod。

6.4.2.1. NetworkPolicy 配置

6.4.2.1.1. 来自 Rsync pod 的出口流量

如果源或目标命名空间中的 **NetworkPolicy** 配置阻止这种类型的流量，您可以使用 Rsync pod 的唯一标签来允许出口流量从它们传递。以下策略允许来自命名空间中 Rsync pod 的**所有**出口流量：

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

6.4.2.1.2. 到 Rsync pod 的入口流量

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

6.4.2.2. EgressNetworkPolicy 配置

EgressNetworkPolicy 对象或 *Egress Firewalls* 是 OpenShift 构造，用于阻止离开集群的出口流量。

与 **NetworkPolicy** 对象不同，egress Firewall 在项目级别工作，因为它适用于命名空间中的所有 pod。因此，Rsync pod 的唯一标签不会使只有 Rsync pod 的 Rsync pod 冲突。但是，您可以将源集群或目标集群的 CIDR 范围添加到策略的 *Allow* 规则中，以便可以在两个集群之间设置直接连接。

根据存在 Egress Firewall 的集群，您可以添加其他集群的 CIDR 范围来允许两者间的出口流量：

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

6.4.2.3. 为 Rsync pod 配置补充组

当 PVC 使用共享存储时，您可以通过将 supplemental 组添加到 Rsync pod 定义来配置对存储的访问，以便 pod 允许访问：

表 6.2. Rsync pod 的附加组群

变量	类型	Default（默认）	描述
src_supplemental_groups	字符串	未设置	用于源 Rsync pod 的以逗号分隔的补充组列表
target_supplemental_groups	字符串	未设置	目标 Rsync pod 的，以逗号分隔的补充组列表

用法示例

可以更新 **MigrationController** CR，以便为这些补充组设置值：

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

6.4.3. 配置代理

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

流程

1. 获取 **MigrationController** CR 清单：

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

2. 更新代理参数：

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

1. 用于直接卷迁移的 stunnel 代理 URL。
2. 要排除代理的目标域名、域、IP 地址或其他网络 CIDR 的逗号分隔列表。

   在域前面加上 **.** 以仅匹配子域。例如： **.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 * 可对所有目的地绕过所有代理。如果您扩展了未包含在安装配置中 **networking.machineNetwork[].cidr** 字段定义的 worker，您必须将它们添加到此列表中，以防止连接问题。

   如果未设置 **httpProxy** 和 **httpsProxy** 字段，则此字段将被忽略。

3. 将清单保存为 **migration-controller.yaml**。

4. 应用更新的清单：

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

如需更多信息，请参阅配置集群范围代理。

6.5. 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。MTC 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。

MTC 支持将数据从源集群迁移到目标集群的文件系统和快照数据复制方法。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 Platform (GCP)
- Microsoft Azure Blob
- 通用 S3 对象存储，例如 Minio 或 Ceph S3

6.5.1. 先决条件

- 所有集群都必须具有对复制存储库的不间断网络访问权限。
- 如果您将代理服务器与内部托管的复制存储库搭配使用，您必须确保代理允许访问复制存储库。

6.5.2. 配置多云对象网关

您可以安装 OpenShift Container Storage Operator，并将一个 Multi-Cloud Object Gateway（MCG）存储桶配置为 Migration Toolkit for Containers（MTC）的复制仓库。

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

6.5.2.2. 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 -u <username>
```

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_backing_store>
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 1
    resources:
      requests:
        storage: <volume_size> 2
    storageClass: <storage_class> 3
  type: pv-pool
```

1 指定持久性卷池中的卷数量。

2 指定卷的大小，例如 **50Gi**。

3 指定存储类，如 **gp2**。

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_bucket_class>
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
```

```
    - backingStores:
      - <mcg_backing_store>
      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: <bucket>
  namespace: openshift-storage
spec:
  bucketName: <bucket> 1
  storageClassName: <storage_class>
  additionalConfig:
    bucketclass: <mcg_bucket_class>
```

1 记录在 MTC web 控制台中添加复制存储库的存储桶名称。

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 MTC 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 --decode
```

- S3 provider secret access key:

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

6.5.3. 配置 Amazon Web Services S3

您可以将 Amazon Web Services（AWS）S3 存储桶配置为 Migration Toolkit for Containers（MTC）的复制仓库。

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> \ 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>/*" (1)
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket>" (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，如下例所示：

输出示例

```
"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)
    }
}
```

记录 **AWS_SECRET_ACCESS_KEY** 和 **AWS_ACCESS_KEY_ID**。您可以使用凭证将 AWS 添加为复制存储库。

6.5.4. 配置 Google Cloud Platform

您可以将 Google Cloud Platform (GCP) 存储桶配置为 Migration Toolkit for Containers (MTC) 的复制仓库。

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. 登录到 **gsutil**:

```
$ gsutil init
```

输出示例

```
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> 1
```

1 2 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** IAM 服务帐户：

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

6. 创建 **SERVICE_ACCOUNT_EMAIL** 变量：

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

7. 创建 **ROLE_PERMISSIONS** 变量：

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

8. 创建 **velero.server** 自定义角色：

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

9. 为项目添加 IAM 策略绑定：

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

10. 更新 IAM 服务帐户：

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

11. 将 IAM 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
```

您可以使用 **credentials-velero** 文件将 GCP 添加为复制存储库。

6.5.5. 配置 Microsoft Azure Blob

您可以将 Microsoft Azure Blob 存储容器配置为 Migration Toolkit for Containers（MTC）的复制仓库。

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
```

您可以使用 **credentials-velero** 文件将 Azure 添加为复制存储库。

6.5.6. 其他资源

- MTC 工作流
- 关于数据复制方法
- 在 MTC web 控制台中添加复制存储库

6.6. 卸载 MTC 并删除资源

您可以卸载 MTC，并删除其资源来清理集群。

注意

删除 **velero** CRD 会从集群中移除 Velero。

先决条件

- 您必须以具有 **cluster-admin** 权限的用户身份登录。

流程

1. 删除所有集群中的 **MigrationController** 自定义资源 (CR)：

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

2. 使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4 上卸载 MTC Operator。
3. 运行以下命令，删除所有集群中的集群范围资源：
   - **migration** 自定义资源定义 (CRDs)：

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

   - **Velero** CRD：

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

   - **migration** 集群角色：

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

   - **migration-operator** 集群角色：

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

   - **velero** 集群角色：

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

   - **migration** 集群角色绑定：

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

   - **migration-operator** 集群角色绑定：

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

   - **velero** 集群角色绑定：

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

第 7 章 在受限网络环境中安装 MTC

您可以通过执行以下步骤在受限网络环境中的 OpenShift Container Platform 3 和 4 上安装 MTC：

1. 创建已镜像的 Operator 目录。
   此过程会创建一个 **mapping.txt** 文件，其中包含 **registry.redhat.io** 镜像和您的镜像 registry 镜像之间的映射。在源集群中安装 Operator 需要 **mapping.txt** 文件。

2. 使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 目标集群上安装 MTC Operator。
   默认情况下，MTC web 控制台和 **Migration Controller** pod 在目标集群中运行。您可以配置 **Migration Controller** 自定义资源清单来在源集群或远程集群中运行 MTC web 控制台和 **Migration Controller** pod。

3. 使用命令行界面在 OpenShift Container Platform 3 源集群上安装 *旧的* MTC Operator。

4. 配置对象存储，以用作复制存储库。

要卸载 MTC，请参阅卸载 MTC 并删除资源。

7.1. 兼容性指南

您必须安装与 OpenShift Container Platform 版本兼容的 MTC。

定义

旧平台
OpenShift Container Platform 4.5 及更早版本。

现代平台
OpenShift Container Platform 4.6 及更新的版本。

旧 Operator
针对传统平台设计的 MTC Operator。

现代 operator
针对现代平台设计的 MTC Operator。

控制集群
运行 MTC 控制器和 GUI 的集群。

远程集群
运行 Velero 的迁移的源或目标集群。Control Cluster 通过 Velero API 与远程集群通信，以驱动迁移。

表 7.1. MTC 兼容性：从传统平台迁移

	OpenShift Container Platform 4.5 或更早版本	OpenShift Container Platform 4.6 或更高版本

	OpenShift Container Platform 4.5 或更早版本	OpenShift Container Platform 4.6 或更高版本
稳定 MTC 版本	MTC 1.7.*z* 旧版 1.7 运算符：使用 **operator.yml** 文件手动安装。 **重要** 此集群不能是控制集群。	MTC 1.7.*z* 使用 OLM 安装，发行频道 **release-v1.7**

注意

在某些情况下，网络的限制可能会阻止现代集群连接到迁移中需要涉及的其他集群。例如，当从内部的 OpenShift Container Platform 3.11 集群迁移到云环境中的现代 OpenShift Container Platform 集群时，，现代集群无法连接到 OpenShift Container Platform 3.11 集群。

对于 MTC 1.7，如果一个远程集群因为网络限制而无法与控制集群进行通信，请使用 **crane tunnel-api** 命令。

对于稳定（stable）的 MTC 发行版本，虽然您应该始终将最现代化的集群指定为控制集群，但是在这种情况下，可能需要将旧的集群指定为控制集群，并将工作负载推送到远程集群。

7.2. 在 OPENSHIFT CONTAINER PLATFORM 4.8 上安装 MTC OPERATOR

您可以使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 上安装 MTC Operator。

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。
- 您必须从本地 registry 中的镜像创建 Operator 目录。

流程

1. 在 OpenShift Container Platform Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **MTCs Operator**。
3. 选择 **Migration Toolkit for Containers Operator** 并点 **Install**。
4. 点击 **Install**。
   在 **Installed Operators** 页中，**openshift-migration** 项目中会出现状态为 **Succeeded** 的 **Migration Toolkit for Containers Operator**。
5. 点 **Migration Toolkit for Containers Operator**。

6. 在 **Provided APIs** 下，找到 **Migration Controller** 标题，再点 **Create Instance**。
7. 点击 **Create**。
8. 点 **Workloads → Pods** 来验证 MTC pod 正在运行 。

7.3. 在 OPENSHIFT CONTAINER PLATFORM 3 上安装旧的 MTC OPERATOR

您可以在 OpenShift Container Platform 3 上手动安装旧的 MTC Operator。

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。
- 您必须有权访问 **registry.redhat.io**。
- 必须安装 **podman** 。
- 您必须创建一个镜像流 secret，并将其复制到集群中的每个节点。
- 您必须有一个有网络访问权限的 Linux 工作站才能从 **registry.redhat.io** 下载文件。
- 您必须创建 Operator 目录的镜像镜像。
- 您必须从 OpenShift Container Platform 4.8 镜像的 Operator 目录安装 MTC Operator。

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
```

2. 输入以下命令下载 **operator.yml** 文件：

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

3. 输入以下命令下载 **controller.yml** 文件：

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

4. 运行以下命令来获取 Operator 镜像映射：

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

mapping.txt 文件是在对 Operator 目录进行镜像时创建的。输出显示了 **registry.redhat.io** 镜像和您的镜像 registry 镜像之间的映射。

输出示例

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

5. 在 **operator.yml** 文件中，为 **ansible** 和 **operator** 容器更新 **image** 值，并更新 **REGISTRY** 值：

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

1 2 指定您的镜像 registry 和 Operator 镜像的 **sha256** 值。

3 指定您的镜像 registry。

6. 登录到您的源集群。

7. 创建 MTC Operator 对象的 Migration Toolkit:

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

输出示例

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

1 您可以忽略 **Error from server (AlreadyExists)** 信息。它们是由 MTC Operator 为早期版本的 OpenShift Container Platform 4 创建资源造成的，这些资源在以后的版本中已提供。

8. 创建 **MigrationController** 对象：

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

9. 验证 MTC Pod 是否正在运行：

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

7.4. 代理配置

对于 OpenShift Container Platform 4.1 及更早的版本，您必须在安装 Migration Toolkit for Containers Operator 后，在 **MigrationController** 自定义资源 (CR) 清单中配置代理，因为这些版本不支持集群范围的 **proxy** 对象。

对于 OpenShift Container Platform 4.2 到 4.8，MTC 会继承集群范围的代理设置。如果要覆盖集群范围的代理设置，可以更改代理参数。

7.4.1. 直接卷迁移

MTC 1.4.2 中引入了直接卷迁移(DVM)。DVM 只支持一个代理。如果目标集群也位于代理后面，则源集群无法访问目标集群的路由。

如果要从代理后面的源集群执行 DVM，您必须配置一个 TCP 代理，该代理可在传输层进行透明处理，并在不使用自己的 SSL 证书的情况下转发 SSL 连接。Stunnel 代理是此类代理的示例。

7.4.1.1. DVM 的 TCP 代理设置

您可以通过 TCP 代理在源和目标集群之间设置直接连接，并在 **MigrationController** CR 中配置 **stunnel_tcp_proxy** 变量来使用代理：

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

直接卷迁移(DVM)只支持代理的基本身份验证。此外，DVM 仅适用于可透明地传输 TCP 连接的代理。在 man-in-the-middle 模式中的 HTTP/HTTPS 代理无法正常工作。现有的集群范围的代理可能不支持此行为。因此，DVM 的代理设置意与 MTC 中常见的代理配置不同。

7.4.1.2. 为什么使用 TCP 代理而不是 HTTP/HTTPS 代理？

您可以通过 OpenShift 路由在源和目标集群之间运行 Rsync 来启用 DVM。流量通过 TCP 代理(Stunnel)加密。在源集群上运行的 Stunnel 会启动与目标 Stunnel 的 TLS 连接，并通过加密频道来传输数据。

OpenShift 中的集群范围 HTTP/HTTPS 代理通常在 man-in-the-middle 模式进行配置，其中它们将自己的 TLS 会话与外部服务器协商。但是，这不适用于 Stunnel。Stunnel 要求代理不处理它的 TLS 会话，基本上使代理成为一个透明的隧道，只需按原样转发 TCP 连接。因此，您必须使用 TCP 代理。

7.4.1.3. 已知问题

迁移失败并显示 Upgrade request required 错误

迁移控制器使用 SPDY 协议在远程 pod 中执行命令。如果远程集群位于代理或不支持 SPDY 协议的防火墙后，迁移控制器将无法执行远程命令。迁移失败并显示出错信息 **Upgrade request required**。临时解决方案： 使用支持 SPDY 协议的代理。

除了支持 SPDY 协议外，代理或防火墙还必须将 **Upgrade** HTTP 标头传递给 API 服务器。客户端使用此标头打开与 API 服务器的 websocket 连接。如果代理或防火墙阻止 **Upgrade** 标头，则迁移会失败，并显示出错信息 **Upgrade request required**。临时解决方案：确保代理转发 **Upgrade** 标头。

7.4.2. 为迁移调优网络策略

OpenShift 支持根据集群使用的网络插件，限制使用 *NetworkPolicy* 或 *EgressFirewalls* 的流量。如果任何涉及迁移的源命名空间使用此类机制将网络流量限制到 pod，限制可能会在迁移过程中停止到 Rsync pod 的流量。

在源和目标集群上运行的 rsync pod 必须通过 OpenShift Route 相互连接。可将现有的 *NetworkPolicy* 或 *EgressNetworkPolicy* 对象配置为从这些流量限制自动排除 Rsync pod。

7.4.2.1. NetworkPolicy 配置

7.4.2.1.1. 来自 Rsync pod 的出口流量

如果源或目标命名空间中的 **NetworkPolicy** 配置阻止这种类型的流量，您可以使用 Rsync pod 的唯一标签来允许出口流量从它们传递。以下策略允许来自命名空间中 Rsync pod 的**所有**出口流量：

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

7.4.2.1.2. 到 Rsync pod 的入口流量

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

7.4.2.2. EgressNetworkPolicy 配置

EgressNetworkPolicy 对象或 *Egress Firewalls* 是 OpenShift 构造，用于阻止离开集群的出口流量。

与 **NetworkPolicy** 对象不同，egress Firewall 在项目级别工作，因为它适用于命名空间中的所有 pod。因此，Rsync pod 的唯一标签不会使只有 Rsync pod 的 Rsync pod 冲突。但是，您可以将源集群或目标集群的 CIDR 范围添加到策略的 *Allow* 规则中，以便可以在两个集群之间设置直接连接。

根据存在 Egress Firewall 的集群，您可以添加其他集群的 CIDR 范围来允许两者间的出口流量：

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

7.4.2.3. 为 Rsync pod 配置补充组

当 PVC 使用共享存储时，您可以通过将 supplemental 组添加到 Rsync pod 定义来配置对存储的访问，以便 pod 允许访问：

表 7.2. Rsync pod 的附加组群

变量	类型	Default（默认）	描述
src_supplemental_groups	字符串	未设置	用于源 Rsync pod 的以逗号分隔的补充组列表
target_supplemental_groups	字符串	未设置	目标 Rsync pod 的，以逗号分隔的补充组列表

用法示例

可以更新 **MigrationController** CR，以便为这些补充组设置值：

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

7.4.3. 配置代理

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

流程

1. 获取 **MigrationController** CR 清单：

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

2. 更新代理参数：

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

1. 用于直接卷迁移的 stunnel 代理 URL。
2. 要排除代理的目标域名、域、IP 地址或其他网络 CIDR 的逗号分隔列表。

在域前面加上 **.** 以仅匹配子域。例如： **.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 * 可对所有目的地绕过所有代理。如果您扩展了未包含在安装配置中 **networking.machineNetwork[].cidr** 字段定义的 worker，您必须将它们添加到此列表中，以防止连接问题。

如果未设置 **httpProxy** 和 **httpsProxy** 字段，则此字段将被忽略。

3. 将清单保存为 **migration-controller.yaml**。

4. 应用更新的清单：

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

如需更多信息，请参阅配置集群范围代理。

7.5. 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。MTC 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。对于一个受限网络环境，多云对象网关（MCG）是唯一支持的选项。

MTC 支持将数据从源集群迁移到目标集群的文件系统和快照数据复制方法。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。

7.5.1. 先决条件

- 所有集群都必须具有对复制存储库的不间断网络访问权限。
- 如果您将代理服务器与内部托管的复制存储库搭配使用，您必须确保代理允许访问复制存储库。

7.5.2. 配置多云对象网关

您可以安装 OpenShift Container Storage Operator，并将一个 Multi-Cloud Object Gateway（MCG）存储桶配置为 Migration Toolkit for Containers（MTC）的复制仓库。

7.5.2.1. 安装 OpenShift Container Storage Operator

您可以从 OperatorHub 安装 OpenShift Container Storage Operator。

如需更多信息，请参阅 *Red Hat OpenShift Container 存储：规划您的部署中*的断开连接的环境部分。

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

7.5.2.2. 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 -u <username>
```

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_backing_store>
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 3 (1)
    resources:
      requests:
        storage: <volume_size> (2)
    storageClass: <storage_class> (3)
  type: pv-pool
```

(1) 指定持久性卷池中的卷数量。

(2) 指定卷的大小，例如 **50Gi**。

(3) 指定存储类，如 **gp2**。

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_bucket_class>
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
    - backingStores:
      - <mcg_backing_store>
      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: <bucket>
  namespace: openshift-storage
spec:
  bucketName: <bucket> 1
  storageClassName: <storage_class>
  additionalConfig:
    bucketclass: <mcg_bucket_class>
```

1 记录在 MTC web 控制台中添加复制存储库的存储桶名称。

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 MTC 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 --decode
```

- S3 provider secret access key:

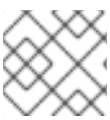
```
$ oc get secret -n openshift-storage migstorage \
  -o go-template='{{ .data.AWS_SECRET_ACCESS_KEY }}' | base64 --decode
```

7.5.3. 其他资源

- MTC 工作流
- 关于数据复制方法
- 在 MTC web 控制台中添加复制存储库

7.6. 卸载 MTC 并删除资源

您可以卸载 MTC，并删除其资源来清理集群。



注意

删除 **velero** CRD 会从集群中移除 Velero。

先决条件

- 您必须以具有 **cluster-admin** 权限的用户身份登录。

流程

1. 删除所有集群中的 **MigrationController** 自定义资源 (CR)：

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

2. 使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4 上卸载 MTC Operator。
3. 运行以下命令，删除所有集群中的集群范围资源：
   - **migration** 自定义资源定义 (CRDs)：

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

   - **Velero** CRD：

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

   - **migration** 集群角色：

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

   - **migration-operator** 集群角色：

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

   - **velero** 集群角色：

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

   - **migration** 集群角色绑定：

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

   - **migration-operator** 集群角色绑定：

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

   - **velero** 集群角色绑定：

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

第 8 章 升级 MTC

您可以使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 上升级 MTC。

您可以通过重新安装旧的 Migration Toolkit for Containers Operator 在 OpenShift Container Platform 3 上升级 MTC。

重要

如果要升级到 MTC 1.3，您必须执行额外步骤来更新 **MigPlan** 自定义资源（CR）。

8.1. 在 OPENSHIFT CONTAINER PLATFORM 4.8 中升级 MTC

您可以使用 Operator Lifecycle Manager 在 OpenShift Container Platform 4.8 上升级 MTC。

先决条件

- 您必须以具有 **cluster-admin** 权限的用户身份登录。

流程

1. 在 OpenShift Container Platform 控制台中导航至 **Operators → Installed Operators**。处于待定升级的 operator 会显示 **Upgrade available** 状态。
2. 点 **Migration Toolkit for Containers Operator**。
3. 点 **Subscription** 标签页。任何需要批准的升级都会在 **Upgrade Status** 旁边显示。例如：它可能会显示 **1 requires approval**。
4. 点 **1 requires approval**，然后点 **Preview Install Plan**。
5. 查看列出可用于升级的资源，并点 **Approve**。
6. 返回 **Operators → Installed Operators** 页面来监控升级的过程。完成后，状态会变为 **Succeeded** 和 **Up to date**。
7. 点 **Workloads → Pods** 来验证 MTC pod 正在运行 。

8.2. 在 OPENSHIFT CONTAINER PLATFORM 3 上升级 MTC

您可以通过手动安装旧的 MTC Operator，在 OpenShift Container Platform 3 上升级 MTC。

先决条件

- 您必须以具有 **cluster-admin** 权限的用户身份登录。
- 您必须有权访问 **registry.redhat.io**。
- 必须安装 **podman** 。

流程

1. 输入以下命令，使用您的红帽客户门户网站凭证登录到 **registry.redhat.io** ：

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

输入以下命令下载 operator.yml 文件：

+

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

1. 输入以下命令替换 Containers Operator 的 Migration Toolkit：

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

2. 输入以下命令将 **migration-operator** 部署扩展到 **0** 以停止部署：

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

3. 输入以下命令将 **migration-operator** 部署扩展到 **1** 以启动部署并应用更改：

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

4. 输入以下命令验证 **migration-operator** 是否已升级：

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

5. 输入以下命令下载 **controller.yml** 文件：

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

6. 运行以下命令来创建 **migration-controller** 对象：

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

7. 如果您之前已将 OpenShift Container Platform 3 集群添加到 MTC web 控制台，必须在 web 控制台中更新服务帐户令牌，因为升级过程会删除并恢复 **openshift-migration** 命名空间：

   a. 输入以下命令来获取服务帐户令牌：

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

   b. 在 MTC web 控制台中点 **Clusters**。

   c. 点集群  旁边的 Options 菜单并选择 **Edit**。

   d. 在 **Service account token 字段中输入新服务帐户令牌**。

   e. 点击 **Update cluster**，然后点击 **Close**。

8. 输入以下命令验证 MTC pod 是否正在运行：

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

8.3. 将 MTC 1.3 升级到 1.7

如果要将 MTC 版本 1.3.x 升级到 1.7，您必须更新运行 **MigrationController** Pod 的集群中的 **MigPlan** 自定义资源(CR)清单。

由于 MTC 1.3 中不存在 **indirectImageMigration** 和 **indirectVolumeMigration** 参数，它们在 1.4 版中的默认值会为 **false**，这意味着启用了直接镜像迁移和直接卷迁移。由于没有满足直接迁移要求，迁移计划无法变为 **Ready** 状态，除非将这些参数值改为 **true**。

先决条件

- 您必须以具有 **cluster-admin** 权限的用户身份登录。

流程

1. 登录到运行 **MigrationController** Pod 的集群。
2. 获取 **MigPlan** CR 清单：

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

3. 更新以下参数值，并将文件保存为 **migplan.yaml**：

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

4. 替换 **MigPlan** CR 清单以应用更改：

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

5. 获取更新的 **MigPlan** CR 清单以验证更改：

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

第 9 章 预迁移检查列表

在使用 Migration Toolkit for Containers（MTC）迁移应用程序工作负载前，请查看以下检查列表。

9.1. 资源

❑ 如果您的应用程序使用内部服务网络或与服务通信的外部路由，则会存在相关路由。

❑ 如果您的应用程序使用集群级别资源，则需要在目标集群中重新创建它们。

❑ 您已 排除了 持久性卷(PV)、镜像流以及您不想迁移的其他资源。

❑ PV 数据已被备份，以应对在应用程序迁移后出现意外行为以及数据被破坏的情况。

9.2. 源集群

❑ 集群满足最低硬件要求。

❑ 已安装了正确的旧 MTC Operator 版本：

- OpenShift Container Platform 版本 3.7 上的 **operator-3.7.yml**。
- OpenShift 容器平台版本 3.9 到 4.5 的 **operator.yml**。

❑ 所有节点都有一个有效的 OpenShift Container Platform 订阅。

❑ 您已执行了所有运行一次的任务。

❑ 已执行所有环境健康检查。

❑ 您已通过运行以下命令检查是否有异常配置的处于 **Terminating** 状态的 PV：

```
$ oc get pv
```

❑ 您已通过运行以下命令检查了状态不是 **Running** 或 **Completed** 的 pod：

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

❑ 您已通过运行以下命令来检查有高重启次数的 pod：

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

即使 pod 处于 **Running** 状态，具有高的重启次数可能表示底层有问题。

❑ 您已使用 修剪 功能从每个命名空间中移除了旧的构建、部署和镜像。

❑ 内部 registry 使用支持的存储类型。

❑ 仅直接镜像迁移：内部 registry 公开给外部流量。

❑ 您可以将镜像读取和写入到 registry。

❑ etcd 集群是健康的。

❑ 源集群中的平均 API 服务器响应时间小于 50 ms。

❑ 集群证书在迁移过程中是有效的。

❑ 您已通过运行以下命令检查是否有待处理的证书签名请求：

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

❑ 身份提供程序正在正常工作。

9.3. 目标集群

❑ 已安装了 MTC Operator 版本 1.5.1。

❑ 满足所有 MTC 的先决条件。

❑ 集群满足特定平台和安装方法（例如在裸机上）的最低硬件要求。

❑ 集群为源集群使用的存储类型定义了存储类，如块卷、文件系统或对象存储。

注意

NFS 不需要定义的存储类。

❑ 集群具有访问外部服务（如数据库、源代码存储库、容器镜像 registry 和 CI/CD 工具）的正确网络配置和权限。

❑ 使用集群提供的服务的外部应用程序和服务具有访问集群的正确网络配置和权限。

❑ 满足内部容器镜像所需的依赖项要求。
如果应用程序使用 OpenShift Container Platform 4.8 不支持的 **openshift** 命名空间中的内部镜像，您可以使用 **podman** 手动更新 OpenShift Container Platform 3 镜像流标签。

❑ 目标集群和复制存储库有足够的存储空间。

❑ 身份提供程序正在工作。

❑ 在目标集群中存在您的应用程序的 DNS 记录。

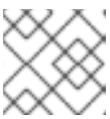
❑ 为每个 OpenShift Container Platform 路由将 **annotations.openshift.io/host.generated** 参数的值设置为 **true**，以更新目标集群的主机名。否则，迁移的路由会保留源集群主机名。

❑ 应用程序使用的证书存在于目标集群中。

❑ 已在目标集群上安装适当的防火墙规则。

❑ 已在目标集群中正确配置负载均衡。

❑ 如果您将对象迁移到与从源迁移的命名空间相同的现有命名空间中，则目标命名空间不包含与正在迁移的对象相同的名称和类型的对象。



注意

迁移前，不要为目标集群上的应用程序创建命名空间，因为这可能导致配额改变。

9.4. 性能

❑ 迁移网络的最小吞吐量为 10 Gbps。

❑ 集群有足够的资源进行迁移。

注意

集群需要额外的内存、CPU 和存储，以便在正常工作负载之上运行迁移。实际的资源要求取决于单个迁移计划中迁移的 Kubernetes 资源数量。您必须在非生产环境中测试迁移，以便估计资源要求。

❑ 节点的内存和 CPU 使用情况是健康的。

❑ 已使用 **fio** 检查了集群的 etcd 磁盘性能。

第 10 章 迁移应用程序

您可以使用 Migration Toolkit for Containers（MTC）web 控制台或命令行来迁移应用程序。

您可以使用阶段迁移和剪切迁移在集群间迁移应用程序：

- 阶段迁移 (Stage migration) 会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。您可以多次运行一个阶段迁移来缩短迁移的持续时间。
- 剪切迁移 (Cutover migration) 将停止源集群中的事务，并将资源移到目标集群。

您可以使用状态迁移来迁移应用程序的状态：

- 状态迁移复制所选持久性卷声明 (PVC)。
- 您可以使用状态迁移来迁移同一集群中的命名空间。

大多数集群范围的资源还没有由 MTC 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。

在迁移过程中，MTC 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。

10.1. 迁移先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

直接镜像迁移

- 您必须确保源集群的安全内部 registry 被公开。
- 您必须创建指向公开 registry 的路由。

直接卷迁移

- 如果您的集群使用代理，您必须配置 Stunnel TCP 代理。

内部镜像

- 如果应用程序使用 **openshift** 命名空间中的内部镜像，您必须确保目标集群中存在所需的镜像版本。
  您可以手动更新镜像流标签，以便在 OpenShift Container Platform 4.8 集群中使用已弃用的 OpenShift Container Platform 3 镜像。

集群

- 源集群必须升级到最新的 MTC z-stream 版本。

- 在所有集群中，MTC 版本必须相同。

Network

- 集群在相互间有无限制的网络访问，并可以访问复制存储库。
- 如果您复制有 **移动** 的持久性卷，集群必须具有对远程卷的不受限制的网络访问权限。
- 您必须在 OpenShift Container Platform 3 集群中启用以下端口：
  - **8443**（API 服务器）
  - **443**（路由）
  - **53** (DNS)
- 您必须在 OpenShift Container Platform 4 集群中启用以下端口：
  - **6443**（API 服务器）
  - **443**（路由）
  - **53** (DNS)
- 如果使用 TLS，则必须在复制存储库中启用端口 **443**。

持久性卷（PV）

- PV 必须有效。
- PV 必须绑定到持久性卷声明。
- 如果使用快照复制 PV，则需要满足以下额外先决条件：
  - 云供应商必须支持快照。
  - PV 必须具有相同的云供应商。
  - PV 必须位于同一区域。
  - PV 必须具有相同的存储类。

迁移先决条件的其他资源

- 为 OpenShift Container Platform 3 手动公开安全 registry
- 更新已弃用的内部镜像

10.2. 使用 MTC WEB 控制台迁移应用程序

您可以使用 MTC web 控制台配置集群和复制存储库。然后，您可以创建并运行迁移计划。

10.2.1. 启动 MTC web 控制台

您可以在浏览器中启动 MTC web 控制台。

先决条件

- MTC web 控制台必须具有到 OpenShift Container Platform Web 控制台的网络访问权限。
- MTC web 控制台必须具有到 OAuth 授权服务器的网络访问权限。

流程

1. 登录到已安装 MTC 的 OpenShift Container Platform 集群。
2. 输入以下命令来获取 MTC 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 MTC web 控制台。

注意

如果在安装 MTC 工具套件 Operator 后尝试立即访问 MTC 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。

10.2.2. 在 MTC web 控制台中添加集群

您可以在 MTC web 控制台中添加一个集群到 Migration Toolkit for Containers（MTC）web 控制台。

先决条件

- 如果要使用 Azure 快照复制数据：
  - 您必须为集群指定 Azure 资源组名称。
  - 集群必须位于同一 Azure 资源组中。
  - 集群必须位于同一地理位置。
- 如果使用直接镜像迁移，则必须公开路由到

流程

1. 登录到集群。
2. 获取 **migration-controller** 服务帐户令牌：

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

输出示例

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwi
a3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJtaWciLCJrdWJlcm5ldGVz
LmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoibWlnLXRva2VuLWs4dDJyIiwia3ViZXJuZ
XRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6Im1pZyIsImt1YmV
ybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImE1YjFiYWM
wLWMxYmYtMTFlOS05Y2NiLTAyOWRmODYwYjMwOCIsInN1YiI6InN5c3RlbTpzZXJ2aWNlY
WNjb3VudDptaWc6bWlnIn0.xqeeAINK7UXpdRqAtOj70qhBJPeMwmgLomV9iFxr5RoqUgKchZ
RG2J2rkqmPm6vr7K-
cm7ibD1IBpdQJCcVDuoHYsFgV4mp9vgOfn9osSDp2TGikwNz4Az95e81xnjVUmzh-
NjDsEpw71DH92iHV_xt2sTwtzftS49LpPW2LjrV0evtNBP_t_RfskdArt5VSv25eORI7zScqfe1CiM
kcVbf2UqACQjo3LbkpfN26HAioO2oH0ECPiRzT0Xyh-KwFutJLS9Xgghyw-
LD9kPKcE_xbbJ9Y4Rqajh7WdPYuB0Jd9DPVrslmzK-F6cgHHYoZEv0SvLQi-
PO0rpDrcjOEQQ
```


3. 在 MTC web 控制台中点 **Clusters**。
4. 点 **Add cluster**。
5. 填写以下字段：
   - **Cluster name**：集群名称可包含小写字母（**a-z**）和数字（**0-9**）。它不能包含空格或国际字符。
   - **URL**：指定 API 服务器 URL，例如 **https://<www.example.com>:8443**。
   - **Service account token**：粘贴 **migration-controller** 服务帐户令牌。
   - **公开的路由主机到镜像 registry**：如果您使用直接镜像迁移，请指定源集群镜像 registry 公开的路由。
     要创建路由，请运行以下命令：
     - 对于 OpenShift Container Platform 3：

       ```
       $ oc create route passthrough --service=docker-registry --port=5000 -n default
       ```

     - 对于 OpenShift Container Platform 4：

       ```
       $ oc create route passthrough --service=image-registry --port=5000 -n openshift-image-registry
       ```

   - **Azure cluster**：如果使用 Azure 快照复制数据，您必须选择此选项。
   - **Azure resource group**：如果选择了 **Azure cluster**，则会显示此字段。指定 Azure 资源组。
   - **需要 SSL 验证**：可选：选择这个选项验证到集群的 SSL 连接。
   - **CA bundle file**：如果选择了 **Require SSL 验证**，则会显示此字段。如果您为自签名证书创建了自定义 CA 证书捆绑包文件，请点 **Browse**，选择 CA 捆绑包文件并上传它。
6. 点 **Add cluster**。
   集群会出现在 **Clusters** 列表中。

10.2.3. 在 MTC web 控制台中添加复制存储库

您可以将对象存储作为复制存储库添加到 MTC web 控制台的 Migration Toolkit for Containers (MTC) web 控制台中。

MTC 支持以下存储供应商：

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

先决条件

- 您必须将对象存储配置为复制存储库。

流程

1. 在 MTC web 控制台中点 **Replication repositories**。
2. 点 **Add repository**。
3. 选择 **Storage provider type** 并填写以下字段：
   - 用于 S3 供应商的 **AWS**，包括 AWS 和 MCG:
     - **Replication repository name**：指定 MTC web 控制台中的复制存储库。
     - **S3 bucket name**：指定 S3 存储桶的名称。
     - **S3 bucket region**：指定 S3 存储桶区域。AWS S3 **必填**。对于某些 S3 供应商是**可选的**。检查 S3 供应商的产品文档，以获取预期值。
     - **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 和其他 S3 供应商指定 S3 供应商访问密钥。
     - **S3 provider secret access key**：为 AWS 指定 **<AWS_ACCESS_KEY_ID>**，或为 MCG 和其他 S3 供应商指定 S3 provider 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。
     - 如果您为自签名证书创建了自定义 CA 证书捆绑包，点 **Browse** 并浏览到 Base64 编码的文件。
   - GCP：
     - **Replication repository name**：指定 MTC 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 MTC 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**。
   新仓库会出现在 **Replication repositories** 列表中。

10.2.4. 在 MTC web 控制台中创建迁移计划

您可以在 Migration Toolkit for Containers（MTC）web 控制台中创建一个迁移计划。

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。
- 您必须确保在所有集群中安装相同的 MTC 版本。
- 您必须在 MTC web 控制台中添加集群和复制存储库。
- 如果要使用 *move* 数据复制方法迁移持久性卷（PV），则源和目标集群必须有对远程卷的不间断网络访问权限。
- 如果要使用直接镜像迁移，您必须指定源集群的镜像 registry 公开的路由。这可以通过使用 MTC web 控制台或更新 **MigCluster** 自定义资源清单来实现。

流程

1. 在 MTC web 控制台中点 **Migration Plan**。
2. 点 **Add migration plan**。
3. 输入 **Plan 名称**。
   迁移计划名称不能超过 253 个小写字母数字字符（**a-z, 0-9**），且不能包含空格或下划线（_）。
4. 选择 **Source cluster**、**Target cluster** 和 **Repository**。
5. 点 **Next**。
6. 选择要迁移的项目。
7. 可选：点击项目旁边的编辑图标来更改目标命名空间。
8. 点 **Next**。
9. 为每个 PV 选择一个 **迁移类型**：
   - **Copy** 选项将源集群的 PV 中的数据复制到复制存储库中，然后在目标集群中恢复新创建的具有类似特征的 PV 上的数据。

- **Move** 选项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。

10. 点 **Next**。
11. 为每个 PV 选择 **Copy method**：
    - **快照复制**使用云供应商的快照功能备份和恢复数据。它比 **Filesystem copy** 要快得多。
    - **Filesystem copy** 备份源集群中的文件，并在目标集群中恢复它们。
      直接卷迁移需要使用文件系统复制方法。
12. 您可以选择 **Verify copy** 来验证使用 **Filesystem copy** 迁移的数据。数据是通过为每个源文件生成 checksum 并在恢复后检查 checksum 来验证。数据校验可能会显著降低性能。
13. 选择 **目标存储类**。
    如果选择了 **Filesystem copy**，您可以更改目标存储类。
14. 点 **Next**。
15. 在 **Migration options** 页面上，如果您为源集群指定了公开的镜像 registry 路由，则会选择 **Direct 镜像迁移** 选项。如果使用 **Filesystem copy** 迁移数据，**Direct PV migration** 选项会被选择。
    直接迁移选项将镜像和文件直接从源集群复制到目标集群。这个选项比将源集群的镜像和文件复制到复制存储库，然后再从复制存储库复制到目标集群要快。
16. 点 **Next**。
17. 可选：点 **Add Hook** 在迁移计划中添加 hook。
    hook 运行自定义代码。您可以在单个迁移计划中最多添加四个 hook。每个 hook 在不同的迁移步骤中运行。
    a. 在 web 控制台中输入要显示的 hook 名称。
    b. 如果 hook 是一个 Ansible playbook，请选择 **Ansible playbook**，然后点 **Browse** 上传 playbook，或在字段中粘贴 playbook 的内容。
    c. 可选： 如果不使用默认 hook 镜像，请指定 Ansible 运行时镜像。
    d. 如果 hook 不是 Ansible playbook，选择 **Custom container image** 并指定镜像名称和路径。
       自定义容器镜像可以包含 Ansible playbook。
    e. 选择 **Source cluster** 或 **Target cluster**。
    f. 输入 **Service account name** 和 **Service account namespace**。
    g. 为 hook 选择迁移步骤：
       - **preBackup**：在应用程序工作负载在源集群中备份前
       - **PostBackup**：在应用程序工作负载在源集群中备份后
       - **preRestore**：在目标集群中恢复应用程序工作负载前
       - **postRestore**：在目标集群中恢复应用程序工作负载后
    h. 点 **Add**。

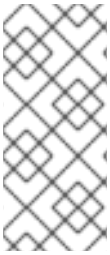
18. 点 **Finish**。
迁移计划显示在 **Migration Plan** 列表中。

其他资源

- MTC 文件系统复制方法
- MTC 快照复制方法

10.2.5. 在 MTC web 控制台中运行迁移计划

您可以使用在 Migration Toolkit for Containers（MTC）web 控制台中创建的迁移计划来迁移应用程序和数据。



注意

迁移过程中，在目标集群中，MTC 将迁移的持久性卷（PV）的重新声明策略设置为 **Retain**。

Backup 自定义资源包含一个 **PVOriginalReclaimPolicy** 注解，用于指示原始重新声明策略。您可以手动恢复迁移 PV 的重新声明策略。

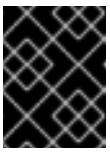
先决条件

MTC web 控制台必须包含以下内容：

- 处于 **Ready** 状态的源集群
- 处于 **Ready** 状态的目标集群
- 复制软件仓库
- 有效的迁移计划

流程

1. 登录到 MTC web 控制台并点 **迁移计划**。
2. 点击迁移计划  旁边的 Options 菜单，并在 **Migration** 中选择以下选项之一：
   - **stage** 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。
   - **cutover** 会停止源集群上的事务，并将资源移到目标集群。
     可选：在 **Cutover 迁移** 对话框中，您可以清除 **Halt transactions on the source cluster during migration** 多选设置。
   - **State** 会复制所选持久性卷声明(PVC)。

     重要

     不要使用状态迁移来在集群之间迁移命名空间。使用 stage 或 cutover migration。

     - 在**状态迁移**对话框中选择一个或多个 PVC 并点 **Migrate**。

3. 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 是否在迁移的命名空间中运行。

   e. 点 **Storage → Persistent volumes** 来验证是否正确置备了已迁移的持久性卷。

第 11 章 高级迁移选项

您可以自动化迁移并修改 **MigPlan** 和 **MigrationController** 自定义资源，以执行大规模迁移并提高性能。

11.1. 术语

表 11.1. MTC 术语

术语	定义
源集群	从中迁移应用程序的集群。
目标集群[1]	将应用程序迁移到的集群。
复制软件仓库	用于在间接迁移过程中复制镜像、卷和 Kubernetes 对象的对象存储，或者用于直接卷迁移或直接镜像迁移期间 Kubernetes 对象的对象存储。 复制存储库必须可以被所有集群访问。
主机集群	运行 **migration-controller** pod 和 Web 控制台的集群。主机集群通常是目标集群，但这不是必需的。 主机集群不需要公开的 registry 路由来直接迁移镜像。
远程集群	远程集群通常是源集群，但这不是必需的。 远程集群需要一个包含 **migration-controller** 服务帐户令牌的 **Secret** 自定义资源。 远程集群需要一个公开的安全 registry 路由来直接迁移镜像。
间接迁移	镜像、卷和 Kubernetes 对象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。
直接卷迁移	持久性卷直接从源集群复制到目标集群。
直接镜像迁移	镜像直接从源集群复制到目标集群。
阶段迁移	在不停止应用程序的情况下，数据将复制到目标集群。 多次运行阶段迁移会缩短迁移的持续时间。
剪切迁移	应用在源集群中停止，其资源迁移到目标集群。
状态迁移	通过将特定的持久性卷声明复制到目标集群来迁移应用程序状态。
回滚迁移	回滚迁移会回滚一个已完成的迁移。

[1] 在 MTC web 控制台中称为 *目标集群*。

11.2. 使用命令行迁移应用程序

您可以使用命令行界面 (CLI) 使用 MTC API 迁移应用程序，以便自动执行迁移。

11.2.1. 迁移先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

直接镜像迁移

- 您必须确保源集群的安全内部 registry 被公开。
- 您必须创建指向公开 registry 的路由。

直接卷迁移

- 如果您的集群使用代理，您必须配置 Stunnel TCP 代理。

内部镜像

- 如果应用程序使用 **openshift** 命名空间中的内部镜像，您必须确保目标集群中存在所需的镜像版本。
  您可以手动更新镜像流标签，以便在 OpenShift Container Platform 4.8 集群中使用已弃用的 OpenShift Container Platform 3 镜像。

集群

- 源集群必须升级到最新的 MTC z-stream 版本。
- 在所有集群中，MTC 版本必须相同。

Network

- 集群在相互间有无限制的网络访问，并可以访问复制存储库。
- 如果您复制有 **移动** 的持久性卷，集群必须具有对远程卷的不受限制的网络访问权限。
- 您必须在 OpenShift Container Platform 3 集群中启用以下端口：
  - **8443**（API 服务器）
  - **443**（路由）
  - **53** (DNS)
- 您必须在 OpenShift Container Platform 4 集群中启用以下端口：
  - **6443**（API 服务器）
  - **443**（路由）
  - **53** (DNS)
- 如果使用 TLS，则必须在复制存储库中启用端口 **443**。

持久性卷（PV）

- PV 必须有效。
- PV 必须绑定到持久性卷声明。
- 如果使用快照复制 PV，则需要满足以下额外先决条件：
  - 云供应商必须支持快照。
  - PV 必须具有相同的云供应商。
  - PV 必须位于同一区域。
  - PV 必须具有相同的存储类。

11.2.2. 创建用于直接镜像迁移的 registry 路由

要直接镜像迁移，您必须在所有远程集群中创建到公开的内部 registry 的路由。

先决条件

- 内部 registry 必须公开给所有远程集群上的外部流量。
  OpenShift Container Platform 4 registry 默认公开。

  OpenShift Container Platform 3 registry 必须手动公开。

流程

- 要创建到 OpenShift Container Platform 3 registry 的路由，请运行以下命令：

```
$ oc create route passthrough --service=docker-registry -n default
```

- 要创建到 OpenShift Container Platform 4 registry 的路由，请运行以下命令：

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

11.2.3. 代理配置

对于 OpenShift Container Platform 4.1 及更早的版本，您必须在安装 Migration Toolkit for Containers Operator 后，在 **MigrationController** 自定义资源 (CR) 清单中配置代理，因为这些版本不支持集群范围的 **proxy** 对象。

对于 OpenShift Container Platform 4.2 到 4.8，MTC 会继承集群范围的代理设置。如果要覆盖集群范围的代理设置，可以更改代理参数。

11.2.3.1. 直接卷迁移

MTC 1.4.2 中引入了直接卷迁移(DVM)。DVM 只支持一个代理。如果目标集群也位于代理后面，则源集群无法访问目标集群的路由。

如果要从代理后面的源集群执行 DVM，您必须配置一个 TCP 代理，该代理可在传输层进行透明处理，并在不使用自己的 SSL 证书的情况下转发 SSL 连接。Stunnel 代理是此类代理的示例。

11.2.3.1.1. DVM 的 TCP 代理设置

您可以通过 TCP 代理在源和目标集群之间设置直接连接，并在 **MigrationController** CR 中配置 **stunnel_tcp_proxy** 变量来使用代理：

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

直接卷迁移(DVM)只支持代理的基本身份验证。此外，DVM 仅适用于可透明地传输 TCP 连接的代理。在 man-in-the-middle 模式中的 HTTP/HTTPS 代理无法正常工作。现有的集群范围的代理可能不支持此行为。因此，DVM 的代理设置意与 MTC 中常见的代理配置不同。

11.2.3.1.2. 为什么使用 TCP 代理而不是 HTTP/HTTPS 代理？

您可以通过 OpenShift 路由在源和目标集群之间运行 Rsync 来启用 DVM。流量通过 TCP 代理(Stunnel)加密。在源集群上运行的 Stunnel 会启动与目标 Stunnel 的 TLS 连接，并通过加密频道来传输数据。

OpenShift 中的集群范围 HTTP/HTTPS 代理通常在 man-in-the-middle 模式进行配置，其中它们将自己的 TLS 会话与外部服务器协商。但是，这不适用于 Stunnel。Stunnel 要求代理不处理它的 TLS 会话，基本上使代理成为一个透明的隧道，只需按原样转发 TCP 连接。因此，您必须使用 TCP 代理。

11.2.3.1.3. 已知问题

迁移失败并显示 Upgrade request required 错误

迁移控制器使用 SPDY 协议在远程 pod 中执行命令。如果远程集群位于代理或不支持 SPDY 协议的防火墙后，迁移控制器将无法执行远程命令。迁移失败并显示出错信息 **Upgrade request required**。临时解决方案： 使用支持 SPDY 协议的代理。

除了支持 SPDY 协议外，代理或防火墙还必须将 **Upgrade** HTTP 标头传递给 API 服务器。客户端使用此标头打开与 API 服务器的 websocket 连接。如果代理或防火墙阻止 **Upgrade** 标头，则迁移会失败，并显示出错信息 **Upgrade request required**。临时解决方案：确保代理转发 **Upgrade** 标头。

11.2.3.2. 为迁移调优网络策略

OpenShift 支持根据集群使用的网络插件，限制使用 *NetworkPolicy* 或 *EgressFirewalls* 的流量。如果任何涉及迁移的源命名空间使用此类机制将网络流量限制到 pod，限制可能会在迁移过程中停止到 Rsync pod 的流量。

在源和目标集群上运行的 rsync pod 必须通过 OpenShift Route 相互连接。可将现有的 *NetworkPolicy* 或 *EgressNetworkPolicy* 对象配置为从这些流量限制自动排除 Rsync pod。

11.2.3.2.1. NetworkPolicy 配置

11.2.3.2.1.1. 来自 Rsync pod 的出口流量

如果源或目标命名空间中的 **NetworkPolicy** 配置阻止这种类型的流量，您可以使用 Rsync pod 的唯一标签来允许出口流量从它们传递。以下策略允许来自命名空间中 Rsync pod 的**所有**出口流量：

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

11.2.3.2.1.2. 到 Rsync pod 的入口流量

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

11.2.3.2.2. EgressNetworkPolicy 配置

EgressNetworkPolicy 对象或 *Egress Firewalls* 是 OpenShift 构造，用于阻止离开集群的出口流量。

与 **NetworkPolicy** 对象不同，egress Firewall 在项目级别工作，因为它适用于命名空间中的所有 pod。因此，Rsync pod 的唯一标签不会使只有 Rsync pod 的 Rsync pod 冲突。但是，您可以将源集群或目标集群的 CIDR 范围添加到策略的 *Allow* 规则中，以便可以在两个集群之间设置直接连接。

根据存在 Egress Firewall 的集群，您可以添加其他集群的 CIDR 范围来允许两者间的出口流量：

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

11.2.3.2.3. 为 Rsync pod 配置补充组

当 PVC 使用共享存储时，您可以通过将 supplemental 组添加到 Rsync pod 定义来配置对存储的访问，以便 pod 允许访问：

表 11.2. Rsync pod 的附加组群

变量	类型	Default（默认）	描述
src_supplemental_groups	字符串	未设置	用于源 Rsync pod 的以逗号分隔的补充组列表
target_supplemental_groups	字符串	未设置	目标 Rsync pod 的，以逗号分隔的补充组列表

用法示例

可以更新 **MigrationController** CR，以便为这些补充组设置值：

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

11.2.3.3. 配置代理

先决条件

- 必须使用在所有集群中具有 **cluster-admin** 权限的用户登录。

流程

1. 获取 **MigrationController** CR 清单：

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

2. 更新代理参数：

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

1. 用于直接卷迁移的 stunnel 代理 URL。
2. 要排除代理的目标域名、域、IP 地址或其他网络 CIDR 的逗号分隔列表。

在域前面加上 **.** 以仅匹配子域。例如：**.y.com** 匹配 **x.y.com**，但不匹配 **y.com**。使用 * 可对所有目的地绕过所有代理。如果您扩展了未包含在安装配置中 **networking.machineNetwork[].cidr** 字段定义的 worker，您必须将它们添加到此列表中，以防止连接问题。

如果未设置 **httpProxy** 和 **httpsProxy** 字段，则此字段将被忽略。

3. 将清单保存为 **migration-controller.yaml**。
4. 应用更新的清单：

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

11.2.4. 使用 MTC API 迁移应用程序

您可以使用 MTC API 从命令行迁移应用程序。

流程

1. 为主机集群创建一个 **MigCluster** CR 清单：

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

2. 为每个远程集群创建一个 **Secret** CR 清单：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: <cluster_secret>
  namespace: openshift-config
type: Opaque
data:
  saToken: <sa_token> 1
EOF
```

1 指定远程集群的 base64 编码的 **migration-controller** 服务帐户（SA）令牌。您可以运行以下命令来获取令牌：

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

3. 为每个远程集群创建一个 **MigCluster** CR 清单：

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  name: <remote_cluster> 1
  namespace: openshift-migration
spec:
```

```
  exposedRegistryPath: <exposed_registry_route> 2
  insecure: false 3
  isHostCluster: false
  serviceAccountSecretRef:
    name: <remote_cluster_secret> 4
    namespace: openshift-config
  url: <remote_cluster_url> 5
EOF
```

1. 指定远程集群的 **Cluster** CR。
2. 可选：要直接镜像迁移，请指定公开的 registry 路由。
3. 如果 **false** 则启用 SSL 验证。如果为 **true**，则不需要 CA 证书或不检查 CA 证书。
4. 指定远程集群的 **Secret** CR 对象。
5. 指定远程集群的 URL。

4. 验证所有集群是否处于 **Ready** 状态：

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

5. 为复制存储库创建 **Secret** CR 清单：

```
$ cat << EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  namespace: openshift-config
  name: <migstorage_creds>
type: Opaque
data:
  aws-access-key-id: <key_id_base64> 1
  aws-secret-access-key: <secret_key_base64> 2
EOF
```

1. 指定 base64 格式的密钥 ID。
2. 指定 base64 格式的 secret 密钥。

AWS 凭证默认为 base64 编码。对于其他存储供应商，您必须使用每个密钥运行以下命令来对凭证进行编码：

```
$ echo -n "<key>" | base64 -w 0 1
```

1. 指定密钥 ID 或 secret 密钥。这两个密钥都必须都是 base64 编码。

6. 为复制存储库创建一个 **MigStorage** CR 清单：

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
```

```
kind: MigStorage
metadata:
  name: <migstorage>
  namespace: openshift-migration
spec:
  backupStorageConfig:
    awsBucketName: <bucket> 1
    credsSecretRef:
      name: <storage_secret> 2
      namespace: openshift-config
  backupStorageProvider: <storage_provider> 3
  volumeSnapshotConfig:
    credsSecretRef:
      name: <storage_secret> 4
      namespace: openshift-config
  volumeSnapshotProvider: <storage_provider> 5
EOF
```

1. 指定存储桶名称。
2. 指定对象存储的 **Secrets** CR。您必须确保存储在对象存储的 **Secrets** CR 中的凭证是正确的。
3. 指定存储供应商。
4. 可选：如果要使用快照复制数据，请指定对象存储的 **Secrets** CR。您必须确保存储在对象存储的 **Secrets** CR 中的凭证是正确的。
5. 可选：如果您使用快照复制数据，请指定存储供应商。

7. 验证 **MigStorage** CR 是否处于 **Ready** 状态：

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

8. 创建一个 **MigPlan** CR 清单：

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  destMigClusterRef:
    name: <host_cluster>
    namespace: openshift-migration
  indirectImageMigration: true 1
  indirectVolumeMigration: true 2
  migStorageRef:
    name: <migstorage> 3
    namespace: openshift-migration
  namespaces:
    - <source_namespace_1> 4
    - <source_namespace_2>
```

```
    - <source_namespace_3>:<destination_namespace> 5
  srcMigClusterRef:
    name: <remote_cluster> 6
    namespace: openshift-migration
EOF
```

1. 如果为 **false**，则启用直接镜像迁移。
2. 如果为 **false**，则启用直接卷迁移。
3. 指定 **MigStorage** CR 实例的名称。
4. 指定一个或多个源命名空间。默认情况下，目标命名空间具有相同的名称。
5. 如果目标命名空间与源命名空间不同，请指定目标命名空间。
6. 指定源集群 **MigCluster** 实例的名称。

9. 验证 **MigPlan** 实例是否处于 **Ready** 状态：

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

10. 创建一个 **MigMigration** CR 清单，以启动 **MigPlan** 实例中定义的迁移：

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  name: <migmigration>
  namespace: openshift-migration
spec:
  migPlanRef:
    name: <migplan> 1
    namespace: openshift-migration
  quiescePods: true 2
  stage: false 3
  rollback: false 4
EOF
```

1. 指定 **MigPlan** CR 名称。
2. 如果为 **true**，则源集群上的 pod 会在迁移前停止。
3. 如果为 **true**，则进行阶段（stage）迁移，即在不停止应用程序的情况下复制大多数数据。
4. 如果为 **true**，则会回滚到一个已完成的迁移。

11. 通过观察 **MigMigration** CR 进度来验证迁移：

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

输出类似于以下：

输出示例

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

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

11.2.5. 状态迁移

您可以使用 Migration Toolkit for Containers(MTC)迁移组成应用程序状态的持久性卷声明(PVC)，执行可重复的、仅状态的迁移。您可以通过从迁移计划中排除其他 PVC 来迁移指定的 PVC。您可以映射 PVC 以确保源和目标 PVC 同步。持久性卷 (PV) 数据复制到目标集群。PV 引用不会被移动，应用程序 pod 将继续在源集群中运行。

State 迁移专门设计用于外部 CD 机制，如 OpenShift Gitops。在使用 MTC 迁移状态时，您可以使用 GitOps 迁移应用程序清单。

如果您有 CI/CD 管道，您可以通过在目标集群中部署无状态组件来迁移它们。然后，您可以使用 MTC 迁移有状态组件。

您可以在集群间或同一集群中执行状态迁移。

重要

状态迁移仅迁移构成应用状态的组件。如果要迁移整个命名空间，请使用 stage 或 cutover migration。

先决条件

- 源集群中的应用程序状态在通过 **PersistentVolumeClaims** 置备的 **PersistentVolume** 中保留。
- 应用程序的清单在中央存储库中可用，它们同时可从源和目标集群访问。

流程

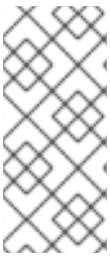
1. 将持久性卷数据从源迁移到目标集群。
   您可以根据需要多次执行此步骤。源应用程序继续运行。
2. 静止源应用程序。
   您可以通过在源集群上直接将工作负载资源副本设置为 **0** 来完成此操作，或者更新 GitHub 中的清单并重新同步 Argo CD 应用程序。

3. 将应用程序清单克隆到目标集群。
   您可以使用 Argo CD 将应用程序清单克隆到目标集群。

4. 将剩余的卷数据从源迁移到目标集群。
   通过执行最终数据迁移，在状态迁移过程中迁移应用程序创建的任何新数据。

5. 如果克隆的应用程序处于静默状态，请取消静止它。

6. 将 DNS 记录切换到目标集群，将用户流量重新定向到已迁移的应用程序。



注意

在执行状态迁移时，MTC 1.6 无法自动静止应用程序。它只能迁移 PV 数据。因此，您必须使用 CD 机制来静止或取消静止应用程序。

MTC 1.7 引入了明确的 Stage 和 Cutover 流。您可以根据需要，使用暂存来执行初始数据传输。然后，您可以执行一个可自动静止源应用程序。

先决条件

- 源集群中的应用程序状态在通过 **PersistentVolumeClaims** 置备的 **PersistentVolume** 中保留。
- 应用程序的清单在中央存储库中可用，它们同时可从源和目标集群访问。

流程

1. 将持久性卷数据从源迁移到目标集群。
   您可以根据需要多次执行此步骤。源应用程序继续运行。

2. 静止源应用程序。
   您可以通过在源集群上直接将工作负载资源副本设置为 **0** 来完成此操作，或者更新 GitHub 中的清单并重新同步 Argo CD 应用程序。

3. 将应用程序清单克隆到目标集群。
   您可以使用 Argo CD 将应用程序清单克隆到目标集群。

4. 将剩余的卷数据从源迁移到目标集群。
   通过执行最终数据迁移，在状态迁移过程中迁移应用程序创建的任何新数据。

5. 如果克隆的应用程序处于静默状态，请取消静止它。

6. 将 DNS 记录切换到目标集群，将用户流量重新定向到已迁移的应用程序。

注意

在执行状态迁移时，MTC 1.6 无法自动静止应用程序。它只能迁移 PV 数据。因此，您必须使用 CD 机制来静止或取消静止应用程序。

MTC 1.7 引入了明确的 Stage 和 Cutover 流。您可以根据需要，使用暂存来执行初始数据传输。然后，您可以执行一个可自动静止源应用程序。

其他资源

- 请参阅从迁移中排除 PVC 以选择状态迁移的 PVC。

- 请参阅映射 PVC，将源 PV 数据迁移到目标集群上置备的 PVC。
- 请参阅迁移 Kubernetes 对象 以迁移组成应用程序状态的 Kubernetes 对象。

11.3. 迁移 HOOK

您可以在单个迁移计划中添加最多四个迁移 hook，每个 hook 在迁移过程的不同阶段运行。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。

迁移 hook 会在以下迁移步骤之一中，在源或目标集群上运行：

- **PreBackup**：在源集群中备份资源前。
- **PostBackup**：在源集群中备份资源后。
- **PreRestore**：在目标集群上恢复资源前。
- **PostRestore**：在目标集群中恢复资源后。

您可以通过创建使用默认 Ansible 镜像运行的 Ansible playbook 或者使用自定义 hook 容器来创建 hook。

Ansible playbook

Ansible playbook 作为一个配置映射挂载到 hook 容器上。hook 容器使用 **MigPlan** 自定义资源中指定的集群、服务帐户和命名空间以作业的形式运行。作业会继续运行，直到达到默认限制的 6 次重试或成功完成为止。即使初始 pod 被驱除或终止，也会继续。

默认 Ansible 运行时镜像为 **registry.redhat.io/rhxetex/openshift-migration-hook-runner-rhel7:1.7**。此镜像基于 Ansible Runner 镜像，并包含 Ansible Kubernetes 资源的 **python-openshift**，以及更新的 **oc** 二进制文件。

自定义 hook 容器

您可以使用自定义 hook 容器而不是默认的 Ansible 镜像。

11.3.1. 为迁移 hook 编写 Ansible playbook

您可以编写 Ansible playbook 以用作迁移 hook。通过使用 MTC web 控制台或在 **MigPlan** 自定义资源（CR）清单中指定 **spec.hooks** 参数的值来在迁移计划中添加 hook。

Ansible playbook 作为一个配置映射挂载到 hook 容器上。hook 容器使用 **MigPlan** CR 中指定的集群、服务帐户和命名空间以作业的形式运行。hook 容器使用指定的服务帐户令牌，以便当任务在集群中运行前无需进行身份验证。

11.3.1.1. Ansible 模块

您可以使用 Ansible **shell** 模块来运行 **oc** 命令。

shell 模块示例

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

您可以使用 **kubernetes.core** 模块（如 **k8s_info** ）与 Kubernetes 资源交互。

k8s_facts 模块示例

```
- hosts: localhost
  gather_facts: false
  tasks:
  - name: Get pod
    k8s_info:
      kind: pods
      api: v1
      namespace: openshift-migration
      name: "{{ lookup( 'env', 'HOSTNAME') }}"
    register: pods

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

在非零退出状态通常不会生成的情况下，可以使用 **fail** 模块生成一个非零退出状态，以确保可以检测到 hook 的成功或失败。hook 以作业形式运行，hook 的成功或失败状态取决于作业容器的退出状态。

fail 模块示例

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

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

11.3.1.2. 环境变量

MigPlan CR 名称和迁移命名空间作为环境变量传递给 hook 容器。这些变量可使用 **lookup** 插件访问。

环境变量示例

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

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

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

11.4. 迁移计划选项

您可以在 **MigPlan** 自定义资源 (CR) 中排除、编辑和映射组件。

11.4.1. 排除资源

您可以从 MTC 迁移计划中排除资源，如镜像流、持久性卷（PV）或订阅，以便减少迁移的资源负载，或使用其他工具迁移镜像或 PV。

默认情况下，MTC 会排除服务目录资源和 Operator Lifecycle Manager（OLM）资源。这些资源是服务目录 API 组和 OLM API 组的一部分，目前还不支持迁移。

流程

1. 编辑 **MigrationController** 自定义资源清单：

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

2. 更新 **spec** 部分，方法是添加参数以排除特定资源，或者在 **exclude_resources** 参数中添加资源（如果它本身没有排除参数）：

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigrationController
metadata:
  name: migration-controller
  namespace: openshift-migration
spec:
  disable_image_migration: true (1)
  disable_pv_migration: true (2)
  ...
  excluded_resources: (3)
  - imagetags
  - templateinstances
  - clusterserviceversions
  - packagemanifests
  - subscriptions
  - servicebrokers
  - servicebindings
  - serviceclasses
  - serviceinstances
  - serviceplans
  - operatorgroups
  - events
  - events.events.k8s.io
```

(1) 添加 **disable_image_migration: true** 以排除迁移中的镜像流。不要编辑 **exclude_resources** 参数。当 **MigrationController** pod 重启时，**镜像流**会添加到 **excluded_resources**。

(2) 添加 **disable_pv_migration: true** 以将 PV 排除在迁移计划之外。不要编辑 **exclude_resources** 参数。当 **MigrationController** Pod 重启时，**persistentvolumes** 和 **persistentvolumeclaims** 会被添加到 **excluded_resources**。禁用 PV 迁移会同时在创建迁移计划时禁用 PV 发现功能。

3 您可以将 OpenShift Container Platform 资源添加到 **exclude_resources** 列表中。不要删除默认排除的资源。对这些进行迁移可能会产生问题，因此必须被排除。

3. 等待 2 分钟，使 **MigrationController** Pod 重启，以便应用更改。
4. 验证资源是否排除：

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

输出包含排除的资源：

输出示例

```
    - name: EXCLUDED_RESOURCES
      value:

imagetags,templateinstances,clusterserviceversions,packagemanifests,subscriptions,servicebro
ers,servicebindings,serviceclasses,serviceinstances,serviceplans,imagestreams,persistentvolun
es,persistentvolumeclaims
```

11.4.2. 映射命名空间

如果您在 **MigPlan** 自定义资源 (CR) 中映射命名空间，您必须确保在源或目标集群上不会重复命名空间，因为命名空间的 UID 和 GID 范围在迁移过程中被复制。

两个源命名空间映射到同一目标命名空间

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

如果您希望源命名空间映射到同一名称的命名空间，则不需要创建映射。默认情况下，源命名空间和目标命名空间具有相同的名称。

命名空间映射不正确

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

正确的命名空间引用

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

11.4.3. 持久性卷声明除外

您可以通过排除您不想迁移的 PVC 来为状态迁移选择持久性卷声明 (PVC)。您可以通过在持久性卷(PV)被发现后设置 **MigPlan** 自定义资源(CR)的 **spec.persistentVolumes.pvc.selection.action** 参数来排除 PVC。

先决条件

- **MigPlan** CR 处于 **Ready** 状态。

流程

- 将 **spec.persistentVolumes.pvc.selection.action** 参数添加到 **MigPlan** CR 中，并将其设置为 **skip** ：

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
...
  persistentVolumes:
  - capacity: 10Gi
    name: <pv_name>
    pvc:
...
    selection:
      action: skip
```

11.4.4. 映射持久性卷声明

您可以通过映射 PVC，将持久性卷(PV)数据从源集群迁移到 **MigPlan** CR 中目标集群中已置备的持久性卷声明(PVC)。此映射可确保迁移的应用的目标 PVC 与源 PVC 同步。

您可以在 PV 被发现后，通过更新 **MigPlan** 自定义资源(CR)中的 **spec.persistentVolumes.pvc.name** 参数来映射 PVC。

先决条件

- **MigPlan** CR 处于 **Ready** 状态。

流程

- 更新 **MigPlan** CR 中的 **spec.persistentVolumes.pvc.name** 参数：

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
...
  persistentVolumes:
  - capacity: 10Gi
```

```
  name: <pv_name>
  pvc:
    name: <source_pvc>:<destination_pvc> 1
```

1 指定源集群上的 PVC 和目标集群上的 PVC。如果目标 PVC 不存在，则会创建它。您可以在迁移过程中使用此映射更改 PVC 名称。

11.4.5. 编辑持久性卷属性

创建 **MigPlan** 自定义资源(CR)后，**MigrationController** CR 会发现持久性卷(PV)。**spec.persistentVolumes** 块和 **status.destStorageClasses** 块添加到 **MigPlan** CR 中。

您可以编辑 **spec.persistentVolumes.selection** 块中的值。如果您更改了 **spec.persistentVolumes.selection** 块以外的值，当 **MigrationController** CR 协调 **MigPlan** CR 时这些值会被覆盖。

注意

spec.persistentVolumes.selection.storageClass 参数的默认值由以下逻辑决定：

1. 如果源集群 PV 是 Gluster 或 NFS，则默认为 **cephfs**，用于 **accessMode: ReadWriteMany** 或 **cephrbd**，表示 **accessMode: ReadWriteOnce**。
2. 如果 PV 既不是 Gluster，也不是 NFS，*或* **cephfs** 或 **cephrbd** 不可用，则默认为同一调配器的存储类。
3. 如果没有同一置备程序存储类，则默认是目标集群的默认存储类。

您可以将 **storageClass** 值改为 **MigPlan** CR 的 **status.destStorageClasses** 块中任何 **name** 参数的值。

如果 **storageClass** 值为空，则 PV 在迁移后将没有存储类。例如，当您想要将 PV 移到目标集群上的 NFS 卷时，这个选项是合适的。

先决条件

- **MigPlan** CR 处于 **Ready** 状态。

流程

- 编辑 **MigPlan** CR 中的 **spec.persistentVolumes.selection** 值：

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  name: <migplan>
  namespace: openshift-migration
spec:
  persistentVolumes:
  - capacity: 10Gi
    name: pvc-095a6559-b27f-11eb-b27f-021bddcaf6e4
    proposedCapacity: 10Gi
    pvc:
      accessModes:
```

```
    - ReadWriteMany
    hasReference: true
    name: mysql
    namespace: mysql-persistent
  selection:
    action: <copy> (1)
    copyMethod: <filesystem> (2)
    verify: true (3)
    storageClass: <gp2> (4)
    accessMode: <ReadWriteMany> (5)
  storageClass: cephfs
```

1. 允许的值包括 **move**、**copy** 和 **skip**。如果只支持一个操作，则默认值是支持的动作。如果支持多个操作，则默认值为 **copy**。
2. 允许的值是 **snapshot** 和 **filesystem**。默认值为 **filesystem**。
3. 如果您在 MTC web 控制台中为文件系统复制选择了验证选项，则会显示 **verify** 参数。您可以将其设置为 **false**。
4. 您可以将默认值改为 **MigPlan** CR 的 **status.destStorageClasses** 块中任何 **name** 参数的值。如果没有指定值，则 PV 在迁移后没有存储类。
5. 允许的值有 **ReadWriteOnce** 和 **ReadWriteMany**。如果没有指定这个值，则默认值是源集群 PVC 的访问模式。您只能在 **MigPlan** CR 中编辑访问模式。您不能使用 MTC web 控制台进行编辑。

其他资源

- 有关 **move** 和 **copy** 操作的详情，请参阅 MTC 工作流。
- 有关 **skip** 操作的详情，请参阅从迁移中排除 PVC。
- 有关文件系统和快照复制方法的详情，请参阅关于数据复制方法。

11.4.6. 使用 MTC API 执行 Kubernetes 对象的状态迁移

迁移所有 PV 数据后，您可以使用 Migration Toolkit for Containers (MTC) API 执行组成应用程序的 Kubernetes 对象的一次性状态迁移。

您可以通过配置 **MigPlan** 自定义资源 (CR) 字段来提供一个带有额外标签选择器的 Kubernetes 资源列表来进一步过滤这些资源，然后通过创建 **MigMigration** CR 来执行迁移。**MigPlan** 资源在迁移后关闭。

注意

选择 Kubernetes 资源是一个仅限 API 的功能。您必须更新 **MigPlan** CR，并使用 CLI 为它创建一个 **MigMigration** CR。MTC web 控制台不支持迁移 Kubernetes 对象。

注意

迁移后，**MigPlan** CR 的 **closed** 参数被设置为 **true**。您不能为此 **MigPlan** CR 创建另一个 **MigMigration** CR。

使用以下选项之一将 Kubernetes 对象添加到 **MigPlan** CR 中：

- 将 Kubernetes 对象添加到 **includeResources** 部分。当 **MigPlan** CR 中指定 **includedResources** 字段时，计划会将 **group-kind** 的列表作为输入。只有列表中显示的资源才会包含在迁移中。
- 添加可选的 **labelSelector** 参数，以过滤 **MigPlan** 中的 **includedResources**。当指定此字段时，迁移中仅包含与标签选择器匹配的资源。例如，您可以使用标签 **app: frontend** 作为过滤器来过滤 **Secret** 和 **ConfigMap** 资源列表。

流程

1. 更新 **MigPlan** CR，使其包含 Kubernetes 资源，并可选择性地通过添加 **labelSelector** 参数来过滤包含的资源：
   a. 更新 **MigPlan** CR 使其包含 Kubernetes 资源：

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

1 指定 Kubernetes 对象，如 **Secret** 或 **ConfigMap**。

   b. 可选：要通过添加 **labelSelector** 参数来过滤包含的资源：

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

1 指定 Kubernetes 对象，如 **Secret** 或 **ConfigMap**。

2 指定要迁移的资源标签，如 **app: frontend**。

2. 创建一个 **MigMigration** CR 来迁移所选 Kubernetes 资源。验证 **migPlanRef** 引用了了正确的 **MigPlan** ：

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

11.5. 迁移控制器选项

您可以编辑迁移计划限制，启用持久性卷大小，或者在 **MigrationController** 自定义资源 (CR) 中启用缓存的 Kubernetes 客户端，以用于大型迁移并提高性能。

11.5.1. 为大型迁移增加限制

您可以使用 MTC 为大型迁移增加迁移对象和容器资源的限制。

重要

您必须在生产环境中执行迁移前测试这些更改。

流程

1. 编辑 **MigrationController** 自定义资源（CR）清单：

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

2. 更新以下参数：

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

1. 指定 **MigrationController** CR 可用的 CPU 数量。
2. 指定 **MigrationController** CR 可用的内存量。
3. 指定可用于 **MigrationController** CR 请求的 CPU 单元数。**100m** 代表 0.1 CPU 单元（100 * 1e-3）。
4. 指定可用于 **MigrationController** CR 请求的内存量。

5 指定可迁移的持久性卷数量。

6 指定可迁移的 pod 数量。

7 指定可迁移的命名空间数量。

3. 创建使用更新的参数验证更改的迁移计划。
   如果您的迁移计划超过 **MigrationController** CR 限制，则 MTC 控制台在保存迁移计划时会显示警告信息。

11.5.2. 为直接卷迁移启用持久性卷大小

您可以启用持久性卷（PV）调整直接卷迁移的大小，以避免在目标集群中耗尽磁盘空间。

当 PV 的磁盘用量达到配置级别时，**MigrationController** 自定义资源（CR）会将持久性卷声明（PVC）的请求存储容量与其实际置备的容量进行比较。然后，它会计算目标集群所需的空间。

pv_resizing_threshold 参数决定何时使用 PV 调整大小。默认阈值是 **3%**。这意味着，当 PV 的磁盘用量超过 **97%** 时，PV 会调整大小。您可以提高这个阈值，以便 PV 调整大小在较低的磁盘用量级别上发生。

PVC 容量根据以下标准计算：

- 如果 PVC 请求的存储容量（**spec.resources.requests.storage**）不等于实际置备的容量（**status.capacity.storage**），则会使用较大的值。
- 如果 PV 通过 PVC 置备，然后更改以便其 PV 和 PVC 容量不再匹配，则会使用较大的值。

先决条件

- PVC 必须附加到一个或多个正在运行的 pod，以便 **MigrationController** CR 可以执行命令。

流程

1. 登录主机集群。
2. 通过修补 **MigrationController** CR 来启用 PV 调整大小：

```
$ oc patch migrationcontroller migration-controller -p '{"spec":
{"enable_dvm_pv_resizing":true}}' \ 1
  --type='merge' -n openshift-migration
```

1 将值设为 **false** 可禁用 PV 大小调整。

3. 可选：更新 **pv_resizing_threshold** 参数以增加阈值：

```
$ oc patch migrationcontroller migration-controller -p '{"spec":{"pv_resizing_threshold":41}}' \
1
  --type='merge' -n openshift-migration
```

1 默认值为 **3**。

超过阈值时，**MigPlan** CR 状态中会显示以下状态信息：

-

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

注意

对于 AWS gp2 存储，因为 gp2 计算卷用量和大小的方式，这个信息不会出现，除非 **pv_resizing_threshold** 为 42% 或更高。（BZ#1973148）

11.5.3. 启用缓存的 Kubernetes 客户端

您可以在 **MigrationController** 自定义资源（CR）中启用缓存的 Kubernetes 客户端，以便在迁移过程中提高性能。在位于不同区域的集群之间迁移时，或存在显著的网络延迟时，会显示最大的性能优势。

注意

但是，委派的任务（例如，用于直接卷迁移的 Rsync 备份或 Velero 备份和恢复）并不会显著提高通过缓存的客户端的性能。

缓存的客户端需要额外的内存，因为 **MigrationController** CR 会缓存与 **MigCluster** CR 交互所需的所有 API 资源。通常发送到 API 服务器的请求会被定向到缓存。缓存会监视 API 服务器是否有更新。

如果启用了缓存的客户端后发生 **OOMKilled** 错误，您可以增加 **MigrationController** CR 的内存限值和请求。

流程

1. 运行以下命令启用缓存的客户端：

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

2. 可选：运行以下命令来增加 **MigrationController** CR 内存限值：

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

3. 可选：运行以下命令来增加 **MigrationController** CR 内存请求：

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

第 12 章 故障排除

本节论述了对 Migration Toolkit for Containers（MTC）进行故障排除的资源。

有关已知问题，请参阅 MTC 发行注记。

12.1. MTC 工作流

您可以使用 MTC web 控制台或 Kubernetes API 将 Kubernetes 资源、持久性卷数据和内部容器镜像迁移到 OpenShift Container Platform 4.8。

MTC 迁移以下资源：

- 在迁移计划中指定的命名空间。
- 命名空间范围的资源：当 MTC 迁移命名空间时，它会迁移与该命名空间关联的所有对象和资源，如服务或 Pod。另外，如果一个资源在命名空间中存在但不在集群级别，这个资源依赖于集群级别存在的另外一个资源，MTC 会迁移这两个资源。
  例如，安全性上下文约束（SCC）是一个存在于集群级别的资源，服务帐户（SA）是存在于命名空间级别的资源。如果 MTC 迁移的命名空间中存在 SA，MTC 会自动找到链接到 SA 的所有 SCC，并迁移这些 SCC。同样，MTC 会迁移链接到命名空间持久性卷的持久性卷声明。

  注意

  根据资源，可能需要手动迁移集群范围的资源。

- 自定义资源 (CR) 和自定义资源定义 (CRD)：MTC 在命名空间级别自动迁移 CR 和 CRD。

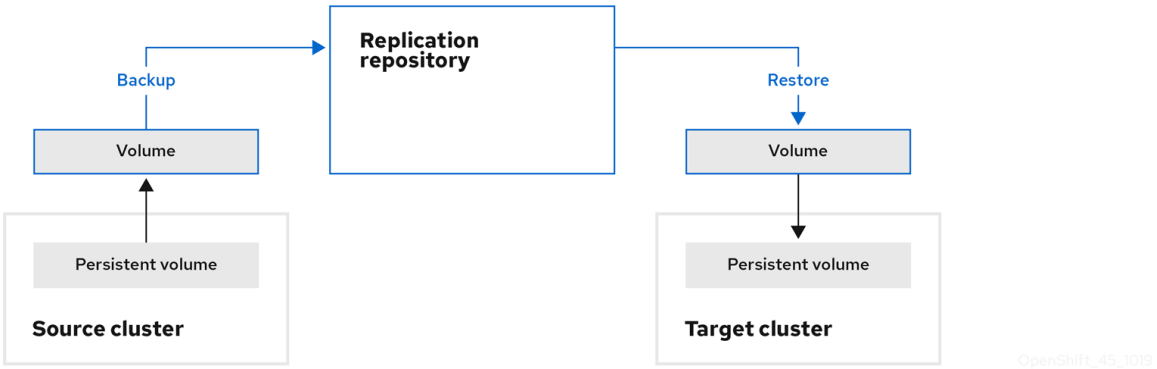
使用 MTC Web 控制台迁移应用程序涉及以下步骤：

1. 在所有集群中安装 MTC Operator。
   您可以在有限的或没有互联网访问的受限环境中为 Containers Operator 安装 Migration Toolkit。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 MTC 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仓库的不受限制的网络访问权限。在受限环境中，您可以使用 Multi-Cloud Object Gateway (MCG)。如果使用代理服务器，您必须将其配置为允许复制仓库和集群间的网络流量。
3. 在 MTC web 控制台中添加源集群。
4. 在 MTC 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**：MTC 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。

     注意

     如果您使用直接镜像迁移或直接卷迁移，则镜像或卷会直接从源集群复制到目标集群。



- **Move**：MTC 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。

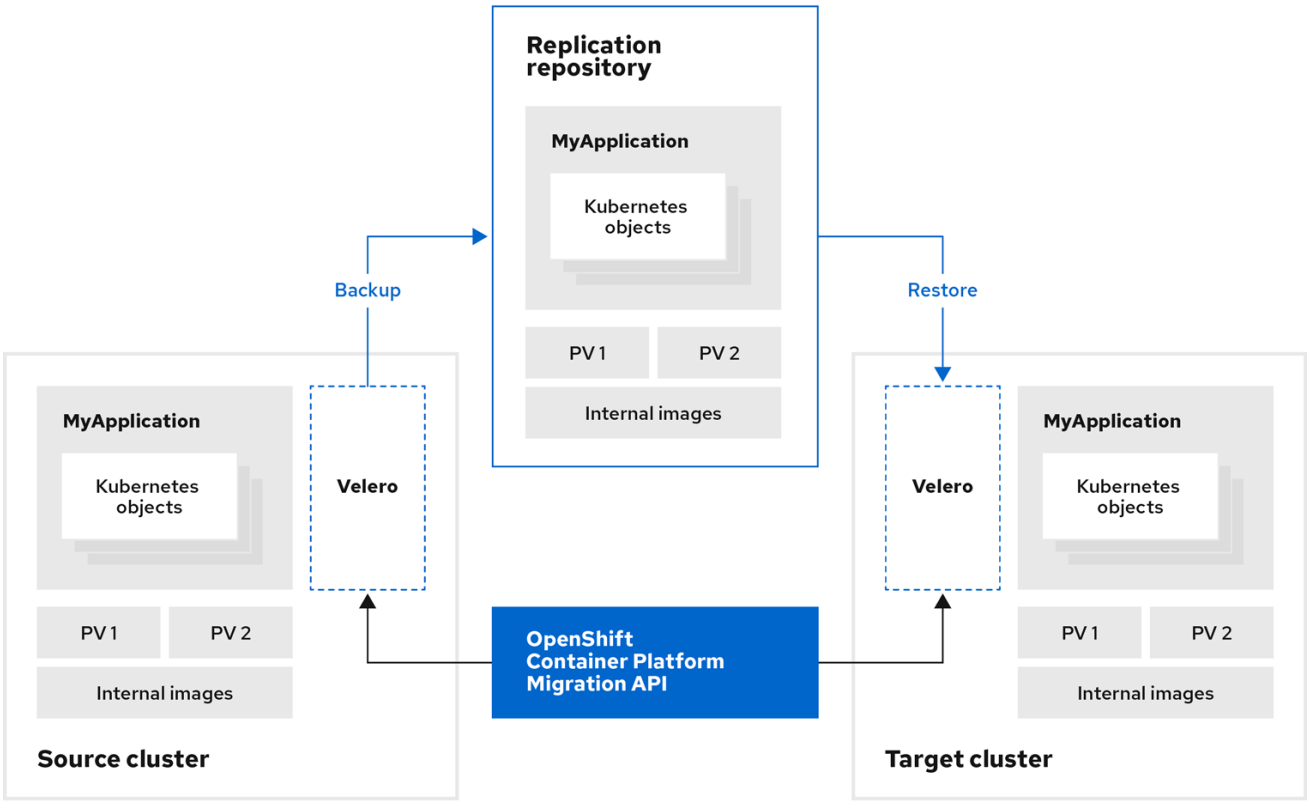
注意

虽然复制仓库没有出现在此图表中，但迁移需要它。

6. 运行迁移计划，使用以下选项之一：

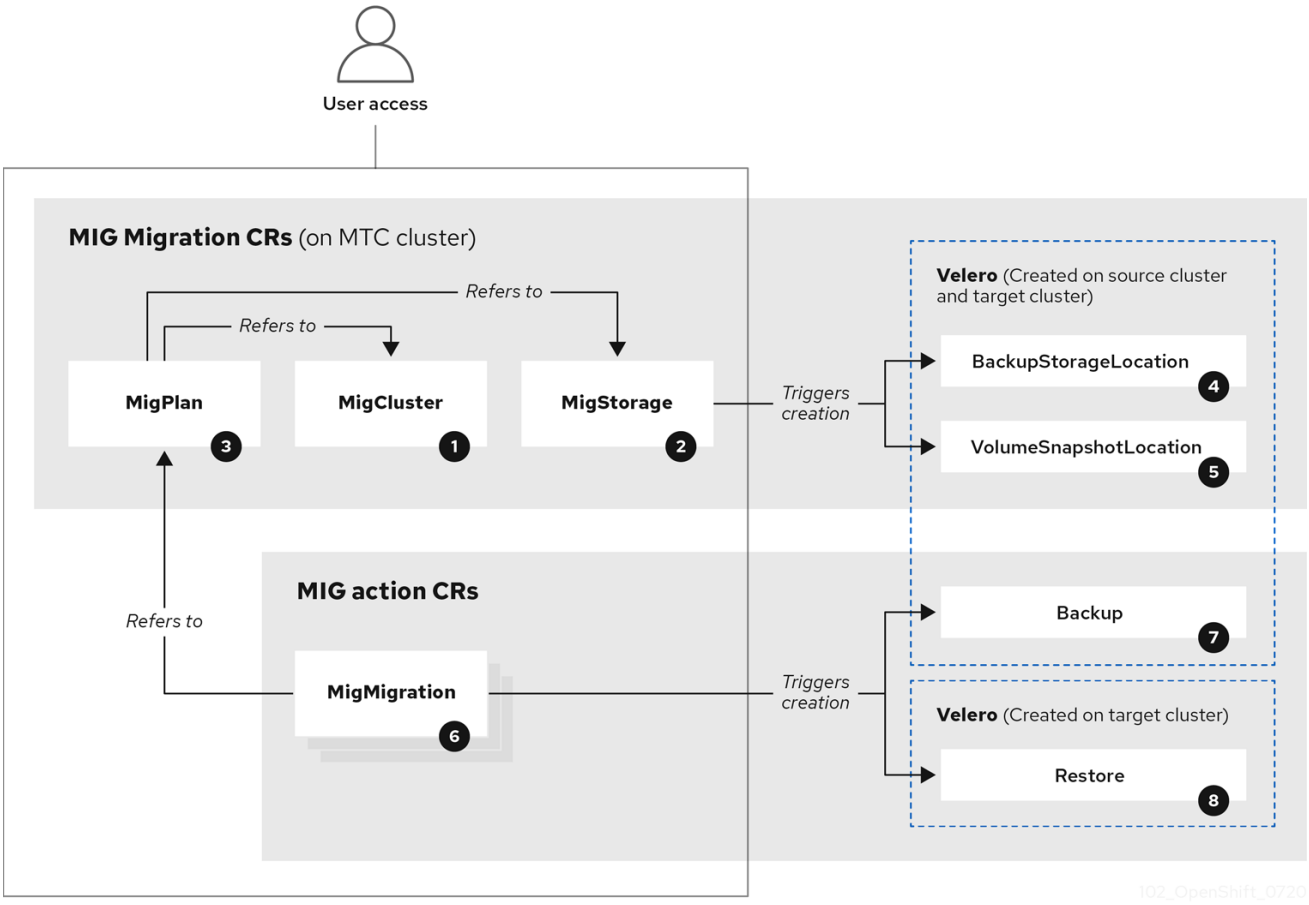
- **stage** 在不停止应用程序的情况下将数据复制到目标集群。
  阶段迁移可以多次运行，以便在迁移前将大多数数据复制到目标。运行一个或多个阶段迁移可缩短迁移的持续时间。

- **cutover** 会停止源集群上的应用程序，并将资源移到目标集群。
  可选：您可以清除 **Halt transactions on the source cluster during migration**多选设置。



关于 MTC 自定义资源

MTC 会创建以下自定义资源（CR）：



1 MigCluster （配置, MTC 集群）: 集群定义

2 MigStorage （配置, MTC 集群）: 存储定义

3 MigPlan （配置, MTC 集群）: 迁移计划

MigPlan CR 描述了要迁移的源和目标集群、复制仓库和命名空间。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, MTC 集群）: **Velero** 备份对象的位置

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

6 MigMigration （操作，MTC 集群）：Migration，在每次进行 stage 或迁移数据时创建。每个 **MigMigration** CR 都与 **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

12.2. MTC 自定义资源清单

MTC 使用以下自定义资源（CR）清单来迁移应用程序。

12.2.1. DirectImageMigration

DirectImageMigration CR 直接将镜像从源集群复制到目标集群。

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

```
    name: <source_cluster>
    namespace: openshift-migration
  destMigClusterRef:
    name: <destination_cluster>
    namespace: openshift-migration
  namespaces: (1)
    - <source_namespace_1>
    - <source_namespace_2>:<destination_namespace_3> (2)
```

1. 包含要迁移的镜像的一个或多个命名空间。默认情况下，目标命名空间的名称与源命名空间相同。
2. 使用不同名称映射到目标命名空间的源命名空间。

12.2.2. DirectImageStreamMigration

DirectImageStreamMigration CR 直接将镜像流引用从源集群复制到目标集群。

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

12.2.3. DirectVolumeMigration

DirectVolumeMigration CR 直接将持久性卷（PV）从源集群复制到目标集群。

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

```
  srcMigClusterRef:
    name: <source_cluster>
    namespace: openshift-migration
```

1. 设置为 **true**，为目标集群上的 PV 创建命名空间。
2. 设置为 **true**，以在迁移后删除 **DirectVolumeMigrationProgress** CR。默认值为 **false**，保留 **DirectVolumeMigrationProgress** CR 以进行故障排除。
3. 如果目标集群不是主机集群，请更新集群名称。
4. 指定要迁移的一个或多个 PVC。

12.2.4. DirectVolumeMigrationProgress

DirectVolumeMigrationProgress CR 显示 **DirectVolumeMigration** CR 的进度。

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

12.2.5. MigAnalytic

MigAnalytic CR 从关联的 **MigPlan** CR 收集镜像、Kubernetes 资源和持久性卷（PV）容量的数量。

您可以配置它收集的数据。

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

```
  migPlanRef:
    name: <migplan>
    namespace: openshift-migration
```

1. 可选：返回镜像数量。
2. 可选：返回 Kubernetes 资源的数量、类型和 API 版本。
3. 可选：返回 PV 容量。
4. 返回镜像名称列表。默认为 **false**，因此输出不会过长。
5. 可选：指定如果 **listImages** 为 **true** 时要返回的最大镜像名称数。

12.2.6. MigCluster

MigCluster CR 定义一个主机、本地或远程集群。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigCluster
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <host_cluster> 1
  namespace: openshift-migration
spec:
  isHostCluster: true 2
# The 'azureResourceGroup' parameter is relevant only for Microsoft Azure.
  azureResourceGroup: <azure_resource_group> 3
  caBundle: <ca_bundle_base64> 4
  insecure: false 5
  refresh: false 6
# The 'restartRestic' parameter is relevant for a source cluster.
  restartRestic: true 7
# The following parameters are relevant for a remote cluster.
  exposedRegistryPath: <registry_route> 8
  url: <destination_cluster_url> 9
  serviceAccountSecretRef:
    name: <source_secret> 10
    namespace: openshift-config
```

1. 如果 **migration-controller** pod 没有在这个集群中运行，请更新集群名称。
2. 如果为 **true**，则 **migration-controller** pod 在此集群中运行。
3. 仅 Microsoft Azure：指定资源组。
4. 可选：如果您为自签名 CA 证书创建了一个证书捆绑包，且 **insecure** 参数值为 **false**，请指定 base64 编码的证书捆绑包。
5. 设置为 **true** 以禁用 SSL 验证。
6. 设置为 **true** 以验证集群。

7 设置为 **true**，以在创建 **Stage** pod 后重启源集群中的 **Restic** pod。

8 远程集群和直接镜像迁移：指定公开的安全 registry 路径。

9 仅远程集群：指定 URL。

10 仅远程集群：指定 **Secret** CR 的名称。

12.2.7. MigHook

MigHook CR 定义一个迁移 hook，它在迁移的指定阶段运行自定义代码。您可以创建最多四个迁移 hook。每个 hook 在迁移的不同阶段运行。

您可以配置 hook 名称、运行时持续时间、自定义镜像，以及 hook 将运行的集群。

hook 的迁移阶段和命名空间在 **MigPlan** CR 中配置。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigHook
metadata:
  generateName: <hook_name_prefix> 1
  name: <mighook> 2
  namespace: openshift-migration
spec:
  activeDeadlineSeconds: 1800 3
  custom: false 4
  image: <hook_image> 5
  playbook: <ansible_playbook_base64> 6
  targetCluster: source 7
```

1 可选：此参数的值后附加一个唯一的哈希值，以便每个迁移 hook 都有一个唯一的名称。您不需要指定 **name** 参数的值。

2 指定迁移 hook 名称，除非指定了 **generateName** 参数的值。

3 可选：指定 hook 可运行的最大秒数。默认值为 **1800**。

4 如果为 **true**，则 hook 是一个自定义镜像。自定义镜像可以包括 Ansible，也可以使用不同的编程语言编写。

5 指定自定义镜像，例如 **quay.io/konveyor/hook-runner:latest**。如果 **custom** 是 **true**，则需要此项。

6 base64 编码的 Ansible playbook.如果 **custom** 是 **false**，则必需。

7 指定要运行 hook 的集群。有效值为 **source** 或 **destination**。

12.2.8. MigMigration

MigMigration CR 运行一个 **MigPlan** CR。

您可以配置 **Migmigration** CR，以运行一个阶段或增量迁移，取消正在进行中的迁移，或回滚已完成的迁移。

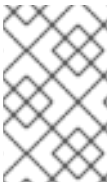
```
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migmigration>
  namespace: openshift-migration
spec:
  canceled: false (1)
  rollback: false (2)
  stage: false (3)
  quiescePods: true (4)
  keepAnnotations: true (5)
 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

1. 设置为 **true** 可取消正在进行中的迁移。
2. 设置为 **true** 以回滚已完成的迁移。
3. 设置为 **true** 以运行暂存迁移。数据会被递增复制，pod 不会在源集群中停止。
4. 设置为 **true** 可在迁移期间停止应用程序。在**备份**阶段后，源集群中的 pod 被缩减为 **0**。
5. 设置为 **true** 以保留迁移过程中应用的标签和注解。
6. 设置为 **true**，以检查目标集群中迁移的 pod 的状态，并返回处于 **Running** 状态的 pod 名称。

12.2.9. MigPlan

MigPlan CR 定义迁移计划的参数。

您可以配置目标命名空间、hook 阶段以及直接或间接迁移。



注意

默认情况下，目标命名空间的名称与源命名空间相同。如果配置了一个不同的目标命名空间，您必须确保不会在源或目标集群上重复命名空间，因为在迁移过程中复制了 UID 和 GID 范围。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigPlan
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migplan>
  namespace: openshift-migration
spec:
  closed: false (1)
  srcMigClusterRef:
    name: <source_cluster>
```

```
  namespace: openshift-migration
destMigClusterRef:
  name: <destination_cluster>
  namespace: openshift-migration
hooks: (2)
  - executionNamespace: <namespace> (3)
    phase: <migration_phase> (4)
    reference:
      name: <hook> (5)
      namespace: <hook_namespace> (6)
    serviceAccount: <service_account> (7)
indirectImageMigration: true (8)
indirectVolumeMigration: false (9)
migStorageRef:
  name: <migstorage>
  namespace: openshift-migration
namespaces:
  - <source_namespace_1> (10)
  - <source_namespace_2>
  - <source_namespace_3>:<destination_namespace_4> (11)
refresh: false (12)
```

1. 如果为 **true**，则迁移已完成。您不能为此 **MigPlan** CR 创建另一个 **MigMigration** CR。
2. 可选：最多可指定四个迁移 hook。每个 hook 必须在不同的迁移阶段运行。
3. 可选：指定运行 hook 的命名空间。
4. 可选：指定 hook 运行期间的迁移阶段。一个 hook 可以分配给一个阶段。有效值为 **PreBackup**、**PostBackup**、**PreRestore** 和 **PostRestore**。
5. 可选：指定 **MigHook** CR 的名称。
6. 可选：指定 **MigHook** CR 的命名空间。
7. 可选：指定一个具有 **cluster-admin** 权限的服务帐户。
8. 如果为 **true**，则禁用直接镜像迁移。镜像从源集群复制到复制存储库，并从复制存储库复制到目标集群。
9. 如果为 **true**，则禁用直接卷迁移。PV 从源集群复制到复制存储库，再从复制存储库复制到目标集群。
10. 指定一个或多个源命名空间。如果只指定源命名空间，则目标命名空间是相同的。
11. 如果目标命名空间与源命名空间不同，请指定它。
12. 如果为 **true** ， **MigPlan** CR 会被验证。

12.2.10. MigStorage

MigStorage CR 描述了复制存储库的对象存储。

支持 Amazon Web Services（AWS）、Microsoft Azure、Google Cloud Storage、Multi-Cloud Object Gateway 和通用 S3 兼容云存储。

AWS 和快照复制方法具有额外的参数。

```
apiVersion: migration.openshift.io/v1alpha1
kind: MigStorage
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migstorage>
  namespace: openshift-migration
spec:
  backupStorageProvider: <backup_storage_provider> 1
  volumeSnapshotProvider: <snapshot_storage_provider> 2
  backupStorageConfig:
    awsBucketName: <bucket> 3
    awsRegion: <region> 4
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret> 5
    awsKmsKeyId: <key_id> 6
    awsPublicUrl: <public_url> 7
    awsSignatureVersion: <signature_version> 8
  volumeSnapshotConfig:
    awsRegion: <region> 9
    credsSecretRef:
      namespace: openshift-config
      name: <storage_secret> 10
  refresh: false 11
```

1. 指定存储供应商。
2. 仅快照复制方法：指定存储供应商。
3. 仅 AWS：指定存储桶名称。
4. 仅 AWS：指定存储桶区域，如 **us-east-1**。
5. 指定您为存储创建的 **Secret** CR 的名称。
6. 仅 AWS：如果您使用 AWS 密钥管理服务，请指定该密钥的唯一标识符。
7. 仅 AWS：如果授予 AWS 存储桶的公共访问权限，请指定存储桶 URL。
8. 仅 AWS：指定向存储桶验证请求的 AWS 签名版本，例如 **4**。
9. 仅快照复制方法：指定集群的地理位置。
10. 仅快照复制方法：指定您为存储创建的 **Secret** CR 的名称。
11. 设置为 **true** 以验证集群。

12.3. 日志和调试工具

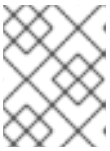
本节论述了可用于故障排除的日志和调试工具。

12.3.1. 查看迁移计划资源

您可以使用 MTC web 控制台和命令行界面（CLI）查看迁移计划资源来监控正在运行的迁移或排除迁移失败的问题。

流程

1. 在 MTC web 控制台中点 **Migration Plans**。
2. 点迁移计划旁边的 **Migrations** 编号来查看 **Migrations** 页面。
3. 点击迁移以查看**迁移详情**。
4. 扩展 **迁移资源**，以在树视图中查看迁移资源及其状态。



注意

要对失败的迁移进行故障排除，请从失败的高级别资源开始，然后向下级资源组成资源树。

5. 点击资源  旁边的 Options 菜单并选择以下选项之一：
   - **复制 oc describe 命令**将命令复制到您的剪贴板。
     - 登录相关集群，然后运行命令。
       资源的条件和事件以 YAML 格式显示。
   - **复制 oc logs 命令**将命令复制到您的剪贴板。
     - 登录相关集群，然后运行命令。
       如果资源支持日志过滤，则会显示过滤的日志。
   - **View JSON** 在 Web 浏览器中以 JSON 格式显示资源数据。
     其数据与 **oc get <resource>** 命令的输出结果相同。

12.3.2. 查看迁移计划日志

您可以查看迁移计划的聚合日志。您可以使用 MTC web 控制台将命令复制到剪贴板中，然后从命令行界面（CLI）运行命令。

该命令显示以下 pod 的过滤日志：

- **Migration Controller**
- **Velero**
- **Restic**
- **Rsync**

- **Stunnel**
- **Registry**

流程

1. 在 MTC web 控制台中点 **Migration Plans**。
2. 点迁移计划旁边的 **Migrations** 号。
3. 单击 **View logs**。
4. 点击 Copy 图标将 **oc logs** 命令复制到您的剪贴板。
5. 登录到相关的集群并在 CLI 中输入命令。
   此时会显示迁移计划的聚合日志。

12.3.3. 使用迁移日志读取器

您可以使用迁移日志读取器显示所有迁移日志的过滤视图。

流程

1. 获取 **mig-log-reader** pod:

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

2. 输入以下命令显示单个迁移日志：

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

(1) **-c plain** 选项显示没有颜色的日志。

12.3.4. 访问性能指标

MigrationController 自定义资源 (CR) 记录指标数据，并将它们拉取到集群监控存储中。您可以使用 Prometheus Query Language (PromQL) 来诊断迁移性能问题，以此查询指标数据。当 Migration Controller pod 重启时，会重置所有指标。

您可以使用 OpenShift Container Platform Web 控制台访问性能指标并运行查询。

流程

1. 在 OpenShift Container Platform web 控制台中点 **Monitoring → Metrics**。
2. 输入 PromQL 查询，选择一个要显示的时间窗口，然后单击 **Run Queries**。
   如果您的 Web 浏览器没有显示所有结果，请使用 Prometheus 控制台。

12.3.4.1. 提供的指标

MigrationController 自定义资源 (CR) 提供了 **MigMigration** CR 计数及其 API 请求的指标。

12.3.4.1.1. cam_app_workload_migrations

此指标是一段时间内的 **MigMigration** CR 计数。它可用于与 **mtc_client_request_count** 和 **mtc_client_request_elapsed** 指标一起查看，以整理迁移状态变化的 API 请求信息。此指标包含在 Telemetry 中。

表 12.1. cam_app_workload_migrations metric

可查询的标签名称	标签值示例	标签描述
status	**running**, **idle**, **failed**, **completed**	**MigMigration** CR 的状态
type	stage, final	**MigMigration** CR 类型

12.3.4.1.2. mtc_client_request_count

此指标是 **MigrationController** 发布的 Kubernetes API 请求的累积计数。它不包含在 Telemetry 中。

表 12.2. mtc_client_request_count metric

可查询的标签名称	标签值示例	标签描述
cluster	**https://migcluster-url:443**	针对发出请求的集群
component	**MigPlan**, **MigCluster**	发出请求的子控制器 API
function	**(*ReconcileMigPlan).Reconcile**	发出请求的功能
kind	**SecretList**、**Deployment**	为 Kubernetes 发出的请求类型

12.3.4.1.3. mtc_client_request_elapsed

这个指标是 **MigrationController** 发布的 Kubernetes API 请求的累积延迟，以毫秒为单位。它不包含在 Telemetry 中。

表 12.3. mtc_client_request_elapsed 指标

可查询的标签名称	标签值示例	标签描述
cluster	**https://cluster-url.com:443**	针对发出请求的集群
component	**migplan**, **migcluster**	发出请求的子控制器 API
function	**(*ReconcileMigPlan).Reconcile**	发出请求的功能
kind	**SecretList**、**Deployment**	为请求发布的 Kubernetes 资源

12.3.4.1.4. 有用的查询

表格中列出了可用于监控性能的一些有用查询。

表 12.4. 有用的查询

查询	描述
mtc_client_request_count	发布的 API 请求数，按请求类型排序
sum(mtc_client_request_count)	发出的 API 请求总数
mtc_client_request_elapsed	API 请求延迟，根据请求类型排序
sum(mtc_client_request_elapsed)	API 请求的总延迟
sum(mtc_client_request_elapsed) / sum(mtc_client_request_count)	API 请求的平均延迟
mtc_client_request_elapsed / mtc_client_request_count	API 请求的平均延迟，按请求类型排序
cam_app_workload_migrations{status="running"} * 100	运行的迁移计数，乘以 100 可更轻松查看请求数

12.3.5. 使用 must-gather 工具

您可以使用 **must-gather** 工具来收集 MTC 自定义资源的日志、指标和相关信息。

must-gather 数据必须附加到所有客户案例。

您可以收集一小时或 24 小时内的数据，并使用 Prometheus 控制台查看数据。

先决条件

- 您必须使用具有 **cluster-admin** 角色的用户登录到 OpenShift Container Platform 集群。
- 已安装 OpenShift CLI (**oc**)。

流程

1. 进入存储 **must-gather** 数据的目录。
2. 为以下数据收集选项之一运行 **oc adm must-gather** 命令：
   - 为过去几小时收集数据：

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

     数据保存为 **must-gather/must-gather.tar.gz**。您可以将此文件上传到红帽客户门户网站中的支持问题单中。

   - 为过去 24 小时收集数据：

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

此操作可能需要很长时间。数据保存为 **must-gather/metrics/prom_data.tar.gz**。

使用 Prometheus 控制台查看指标数据
您可以使用 Prometheus 控制台查看指标数据。

流程

1. 解压缩 **prom_data.tar.gz** 文件：

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

2. 创建本地 Prometheus 实例：

```
$ make prometheus-run
```

命令输出 Prometheus URL。

输出

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

3. 启动 Web 浏览器，再导航到 URL 以使用 Prometheus Web 控制台查看数据。
4. 查看数据后，删除 Prometheus 实例和数据：

```
$ make prometheus-cleanup
```

12.3.6. 使用 Velero CLI 工具调试 Velero 资源

您可以调试 **Backup** 和 **Restore** 自定义资源(CR)并使用 Velero CLI 工具检索日志。

Velero CLI 工具比 OpenShift CLI 工具提供更详细的信息。

语法
使用 **oc exec** 命令运行 Velero CLI 命令：

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

示例

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

帮助选项
使用 **velero --help** 列出所有 Velero CLI 命令：

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

describe 命令
使用 **velero describe** 命令检索与 **Backup** 或 **Restore** CR 关联的警告和错误概述：

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

示例

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

logs 命令
使用 **velero logs** 命令检索 **Backup** 或 **Restore** CR 的日志：

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

示例

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

12.3.7. 调试部分迁移失败

您可以使用 Velero CLI 检查 **Restore** 自定义资源（CR）日志来调试部分迁移失败警告消息。

当 Velero 遇到没有导致迁移失败的问题时，会导致迁移部分失败。例如，缺少自定义资源定义（CRD），或者源集群和目标集群的 CRD 版本之间存在冲突，则迁移会完成，但不会在目标集群上创建 CR。

Velero 将问题作为部分失败记录，然后处理 **备份** CR 中的其他对象。

流程

1. 检查 **MigMigration** CR 的状态：

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

   输出示例

   ```
   status:
     conditions:
     - category: Warn
       durable: true
       lastTransitionTime: "2021-01-26T20:48:40Z"
       message: 'Final Restore openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-
   x4lbf: partially failed on destination cluster'
       status: "True"
       type: VeleroFinalRestorePartiallyFailed
   ```

```
  - category: Advisory
    durable: true
    lastTransitionTime: "2021-01-26T20:48:42Z"
    message: The migration has completed with warnings, please look at `Warn` conditions.
    reason: Completed
    status: "True"
    type: SucceededWithWarnings
```

2. 使用 Velero **describe** 命令检查 **Restore** CR 的状态：

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

输出示例

```
Phase:  PartiallyFailed (run 'velero restore logs ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf' for more information)

Errors:
  Velero:     <none>
  Cluster:    <none>
  Namespaces:
    migration-example:  error restoring example.com/migration-example/migration-example: the server could not find the requested resource
```

3. 使用 Velero **logs** 命令检查 **Restore** CR 日志：

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

输出示例

```
time="2021-01-26T20:48:37Z" level=info msg="Attempting to restore migration-example: migration-example" logSource="pkg/restore/restore.go:1107" restore=openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
time="2021-01-26T20:48:37Z" level=info msg="error restoring migration-example: the server could not find the requested resource" logSource="pkg/restore/restore.go:1170" restore=openshift-migration/ccc7c2d0-6017-11eb-afab-85d0007f5a19-x4lbf
```

Restore CR 会记录日志错误消息， **the server could not find the requested resource**，代表迁移部分失败的原因。

12.3.8. 使用 MTC 自定义资源进行故障排除

您可以检查以下 MTC 自定义资源（CR）来排除迁移失败的问题：

- **MigCluster**
- **MigStorage**
- **MigPlan**
- **BackupStorageLocation**

BackupStorageLocation CR 包含一个 **migrationcontroller** 标签，用于标识创建 CR 的 MTC 实例：

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

- **VolumeSnapshotLocation**
  VolumeSnapshotLocation CR 包含一个 **migrationcontroller** 标签，用于标识创建 CR 的 MTC 实例：

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

- **MigMigration**
- **Backup**
  在目标集群中，MTC 将迁移的持久性卷（PV）的重新声明策略设置为 **Retain**。**Backup** CR 包含 **openshift.io/orig-reclaim-policy** 注解，用于指示原始重新声明策略。您可以手动恢复迁移 PV 的重新声明策略。
- **恢复**

流程

1. 列出 **openshift-migration** 命名空间中的 **MigMigration** CR:

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

输出示例

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

2. 检查 **MigMigration** CR:

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

输出结果类似以下示例。

MigMigration 示例输出

```
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 示例输出来描述 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
    openshift.io/orig-reclaim-policy: delete
  creationTimestamp: "2019-08-29T01:03:15Z"
  generateName: 88435fe0-c9f8-11e9-85e6-5d593ce65e10-
  generation: 1
  labels:
    app.kubernetes.io/part-of: migration
    migmigration: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    migration-stage-backup: 8886de4c-c9f8-11e9-95ad-0205fe66cbb6
    velero.io/storage-location: myrepo-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 示例输出来描述 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
```

12.4. 常见问题和关注

本节介绍在迁移过程中可能导致问题的常见问题。

12.4.1. 更新已弃用的内部镜像

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 OpenShift Container Platform 4.8 中已弃用 OpenShift Container Platform 3 镜像，您可以使用 **podman** 手动更新镜像流标签。

先决条件

- 必须安装 **podman** 。
- 您必须以具有 **cluster-admin** 权限的用户身份登录。
- 如果您使用不安全的 registry，请将 registry 主机值添加到 **/etc/container/registries.conf** 的 **[registries.insecure]** 部分，以确保 **Podman** 不会遇到 TLS 验证错误。
- 内部 registry 必须在源和目标集群上公开。

流程

1. 确保内部 registry 在 OpenShift Container Platform 3 和 4 集群中公开。
   默认情况下，内部 registry 在 OpenShift Container Platform 4 中公开。
2. 如果您使用不安全的 registry，请将 registry 主机值添加到 **/etc/container/registries.conf** 的 **[registries.insecure]** 部分，以确保 **Podman** 不会遇到 TLS 验证错误。
3. 登录到 OpenShift Container Platform 3 registry：

   ```
   $ podman login -u $(oc whoami) -p $(oc whoami -t) --tls-verify=false <registry_url>:<port>
   ```

4. 登录到 OpenShift Container Platform 4 registry：

   ```
   $ podman login -u $(oc whoami) -p $(oc whoami -t) --tls-verify=false <registry_url>:<port>
   ```

5. 拉取 OpenShift Container Platform 3 镜像：

```
$ podman pull <registry_url>:<port>/openshift/<image>
```

6. 为 OpenShift Container Platform 4 registry 标记 OpenShift Container Platform 3 镜像：

```
$ podman tag <registry_url>:<port>/openshift/<image> \ 1
  <registry_url>:<port>/openshift/<image> 2
```

1. 指定 OpenShift Container Platform 3 集群的 registry URL 和端口。
2. 指定 OpenShift Container Platform 4 集群的 registry URL 和端口。

7. 将镜像推送到 OpenShift Container Platform 4 registry：

```
$ podman push <registry_url>:<port>/openshift/<image> 1
```

1. 指定 OpenShift Container Platform 4 集群。

8. 验证镜像是否具有有效的镜像流：

```
$ oc get imagestream -n openshift | grep <image>
```

输出示例

```
NAME      IMAGE REPOSITORY                                                TAGS    UPDATED
my_image  image-registry.openshift-image-registry.svc:5000/openshift/my_image  latest  32
seconds ago
```

12.4.2. 直接卷迁移未完成

如果直接卷迁移未完成，则目标集群可能没有与源集群相同的 **node-selector** 注解。

MTC 在迁移命名空间时会保留所有注解，以保持安全性上下文约束和调度要求。在直接卷迁移过程中，MTC 在从源集群迁移的命名空间中在目标集群上创建 Rsync 传输 pod。如果目标集群命名空间没有与源集群命名空间相同的注解，则无法调度 Rsync 传输 pod。Rsync pod 处于 **Pending** 状态。

您可以执行以下步骤识别并解决这个问题。

流程

1. 检查 **MigMigration** CR 的状态：

```
$ oc describe migmigration <pod> -n openshift-migration
```

输出包括以下状态消息：

输出示例

```
Some or all transfer pods are not running for more than 10 mins on destination cluster
```

2. 在源集群中，获取迁移的命名空间的详情：

```
$ oc get namespace <namespace> -o yaml 1
```

1 指定迁移的命名空间。

3. 在目标集群中，编辑迁移的命名空间：

```
$ oc edit namespace <namespace>
```

4. 将缺少的 **openshift.io/node-selector** 注解添加到迁移的命名空间中，如下例所示：

```
apiVersion: v1
kind: Namespace
metadata:
  annotations:
    openshift.io/node-selector: "region=east"
...
```

5. 再次运行迁移计划。

12.4.3. 错误信息和解决方案

本节论述了您可能会在 Migration Toolkit for Containers（MTC）中遇到的常见错误消息，以及如何解决其底层原因。

12.4.3.1. 首次访问 MTC 控制台时显示的 CA 证书错误

如果在第一次尝试访问 MTC 控制台时显示 **CA 证书错误**信息，则可能的原因是在一个集群中使用自签名的 CA 证书。

要解决这个问题，进入出错信息中显示的 **oauth-authorization-server** URL 并接受证书。要永久解决这个问题，将证书添加到网页浏览器的信任存储中。

如果您接受证书后显示 **Unauthorized** 信息，进入 MTC 控制台并刷新网页。

12.4.3.2. MTC 控制台中的 OAuth 超时错误

如果在接受自签名证书后，MTC 控制台中显示 **connection has timed out**，其原因可能是：

- 对 OAuth 服务器的网络访问中断
- 对 OpenShift Container Platform 控制台的网络访问中断
- 代理配置中阻断了对 **oauth-authorization-server** URL 的访问。详情请查看 因为 OAuth 超时错误而无法访问 MTC 控制台。

要确定超时的原因：

- 使用浏览器 web 检查器检查 MTC 控制台网页。
- 检查 **Migration UI** pod 日志中的错误。

12.4.3.3. 由未知颁发机构签名的证书错误

如果您使用自签名证书来保护集群或 MTC 的 Migration Toolkit 的复制仓库的安全，则证书验证可能会失败，并显示以下错误消息： **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 MTC 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。

12.4.3.4. 在 Velero pod 日志中有备份存储位置错误

如果 **Velero Backup** 自定义资源包含对不存在的备份存储位置（BSL）的引用， **Velero** pod 日志可能会显示以下错误消息：

```
$ oc logs <Velero_Pod> -n openshift-migration
```

输出示例

```
level=error msg="Error checking repository for stale locks" error="error getting backup storage
location: BackupStorageLocation.velero.io \"ts-dpa-1\" not found" error.file="/remote-
source/src/github.com/vmware-tanzu/velero/pkg/restic/repository_manager.go:259"
```

您可以忽略这些错误消息。缺少 BSL 不会导致迁移失败。

12.4.3.5. Velero pod 日志中的 Pod 卷备份超时错误

如果因为 Restic 超时造成迁移失败，以下错误会在 **Velero** pod 日志中显示。

```
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 **Migration Toolkit for Containers 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. 点击 **Save**。

12.4.3.6. MigMigration 自定义资源中的 Restic 验证错误

如果迁移使用文件系统数据复制方法的持久性卷时数据验证失败，在 **MigMigration** CR 中会显示以下错误。

输出示例

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

(1) 错误消息指定了 **Restore** CR 名称。

(2) **ResticVerifyErrors** 是一个包括验证错误的一般错误警告类型。

注意

数据验证错误不会导致迁移过程失败。

您可以检查 **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
```

输出会标识出带有 **PodVolumeRestore** 错误的持久性卷。

输出示例

```
status:
  phase: Completed
```

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

3. 查看 **PodVolumeRestore** CR:

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

输出中标识了记录错误的 **Restic** pod。

输出示例

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

4. 查看 **Restic** pod 日志以查找错误：

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

12.4.3.7. 从启用了 root_squash 的 NFS 存储中迁移时的 Restic 权限错误

如果您要从 NFS 存储中迁移数据，并且启用了 **root_squash**，**Restic** 会映射到 **nfsnobody**，且没有执行迁移的权限。**Restic** pod 日志中显示以下错误。

输出示例

```
backup=openshift-migration/<backup_id> controller=pod-volume-backup error="fork/exec
/usr/bin/restic: permission denied" error.file="/go/src/github.com/vmware-
tanzu/velero/pkg/controller/pod_volume_backup_controller.go:280"
error.function="github.com/vmware-tanzu/velero/pkg/controller.
(*podVolumeBackupController).processBackup"
logSource="pkg/controller/pod_volume_backup_controller.go:280" name=<backup_id>
namespace=openshift-migration
```

您可以通过为 Restic 创建补充组并将组 ID 添加到 **MigrationController** CR 清单来解决这个问题。

流程

1. 在 NFS 存储上为 Restic 创建补充组。
2. 在 NFS 目录上设置 **setgid** 位，以便继承组所有权。
3. 将 **restic_supplemental_groups** 参数添加到源和目标集群上的 **MigrationController** CR 清单：

```
spec:
  restic_supplemental_groups: <group_id> 1
```

1 指定补充组 ID。

4. 等待 **Restic** pod 重启，以便应用更改。

12.4.4. 已知问题

这个版本有以下已知问题：

- 在迁移过程中，MTC 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)
- 大多数集群范围的资源还没有由 MTC 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**MigrationController** CR 清单中的 **restic_timeout** 参数值（默认为 **1h**）。
- 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。
- 如果您要从 NFS 存储中迁移数据，并且启用了 **root_squash**，将 **Restic** 映射为 **nfsnobody**。迁移失败，**Restic** Pod 日志中显示权限错误。（BZ#1873641）
  您可以通过在 **MigrationController** CR 清单中添加用于 **Restic** 的额外组来解决这个问题：

```
spec:
...
  restic_supplemental_groups:
  - 5555
  - 6666
```

- 如果您使用位于不同可用区的节点执行直接卷迁移，则迁移可能会失败，因为迁移的 pod 无法访问 PVC。（BZ#1947487）

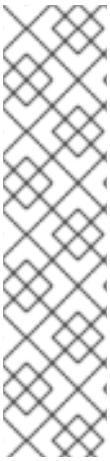
12.5. 回滚一个迁移

您可以使用 MTC web 控制台或 CLI 回滚迁移。

您还可以手动回滚迁移。

12.5.1. 使用 MTC web 控制台回滚迁移

您可以使用 Migration Toolkit for Containers（MTC）web 控制台回滚迁移。



注意

以下资源保留在迁移的命名空间中，以便在直接卷迁移 (DVM) 失败后进行调试：

- 配置映射（源和目标集群）
- **Secret** CR（源和目标集群）
- **Rsync** CR（源集群）

这些资源不会影响回滚。您可以手动删除它们。

如果您稍后成功运行相同的迁移计划，则会自动删除失败迁移中的资源。

如果应用程序在迁移失败时停止，您必须回滚迁移，以防止持久性卷中的数据崩溃。

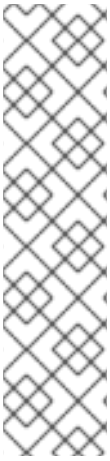
如果应用程序在迁移过程中没有停止，则不需要回滚，因为原始应用程序仍然在源集群中运行。

流程

1. 在 MTC web 控制台中点 **Migration Plan**。
2. 点击迁移计划  旁边的 Options 菜单，并在 **Migration** 下选择 **Rollback**。
3. 点 **Rollback** 并等待回滚完成。
   在迁移计划详情中会显示 **Rollback succeeded**。
4. 验证源集群的 OpenShift Container Platform Web 控制台中是否成功回滚：
   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 是否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。

12.5.2. 使用命令行界面回滚迁移

您可以通过从命令行界面创建 **MigMigration** 自定义资源（CR）来回滚迁移。



注意

以下资源保留在迁移的命名空间中，以便在直接卷迁移 (DVM) 失败后进行调试：

- 配置映射（源和目标集群）
- **Secret** CR（源和目标集群）
- **Rsync** CR（源集群）

这些资源不会影响回滚。您可以手动删除它们。

如果您稍后成功运行相同的迁移计划，则会自动删除失败迁移中的资源。

如果应用程序在迁移失败时停止，您必须回滚迁移，以防止持久性卷中的数据崩溃。

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

流程

1. 根据以下示例创建一个 **MigMigration** CR：

```
$ cat << EOF | oc apply -f -
apiVersion: migration.openshift.io/v1alpha1
kind: MigMigration
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: <migmigration>
  namespace: openshift-migration
spec:
...
  rollback: true
...
  migPlanRef:
    name: <migplan> 1
    namespace: openshift-migration
EOF
```

1 指定关联的 **MigPlan** CR 的名称。

2. 在 MTC web 控制台中，验证迁移的项目资源是否已从目标集群中移除。
3. 验证迁移的项目资源是否存在于源集群中，并且应用程序是否正在运行。

12.5.3. 手动回滚迁移

您可以通过删除 **stage** pod 并取消静止应用程序来手动回滚失败的迁移。

如果您成功运行相同的迁移计划，则会自动删除失败迁移中的资源。

注意

在直接卷迁移失败 (DVM) 后，以下资源会保留在迁移的命名空间中：

- 配置映射（源和目标集群）
- **Secret** CR（源和目标集群）
- **Rsync** CR（源集群）

这些资源不会影响回滚。您可以手动删除它们。

流程

1. 删除所有集群中的 **stage** pod：

```
$ oc delete $(oc get pods -l migration.openshift.io/is-stage-pod -n <namespace>) 1
```

1 **MigPlan** CR 中指定的命名空间。

2. 通过将副本扩展到其预迁移编号，在源集群中取消静默应用程序：

```
$ oc scale deployment <deployment> --replicas=<premigration_replicas>
```

Deployment CR 中的 **migration.openshift.io/preQuiesceReplicas** 注解显示预迁移副本数：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
    migration.openshift.io/preQuiesceReplicas: "1"
```

3. 验证应用程序 pod 是否在源集群中运行：

```
$ oc get pod -n <namespace>
```

其他资源

- 使用 Web 控制台从集群中删除 Operator