OpenShift Container Platform 4.8

备份和恢复

备份和恢复 OpenShift Container Platform 集群

Last Updated: 2023-04-19

OpenShift Container Platform 4.8 备份和恢复

备份和恢复 OpenShift Container Platform 集群

法律通告

摘要

本文档提供了备份集群数据以及从各种灾难场景中恢复的步骤。

目录

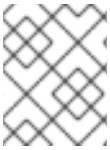
第 1 章 备份和恢复

1.1. OPENSHIFT CONTAINER PLATFORM 中的备份和恢复操作概述

作为集群管理员，您可能需要在一段时间内停止 OpenShift Container Platform 集群，并在以后重启集群。重启集群的一些原因是您需要对集群执行维护或希望降低资源成本。在 OpenShift Container Platform 中，您可以对集群执行安全关闭，以便在以后轻松重启集群。

您必须在关闭集群前 备份 etcd 数据 ；etcd 是 OpenShift Container Platform 的键值存储，它会保留所有资源对象的状态。etcd 备份在灾难恢复中扮演着关键角色。在 OpenShift Container Platform 中，您还可以替换不健康的 etcd 成员。

当您希望集群再次运行时，请安全地重启集群。



注意

集群的证书在安装日期后一年后过期。您可以关闭集群，并在证书仍有效时安全地重启集群。虽然集群自动检索过期的 control plane 证书，但您仍需要批准证书签名请求(CSR)。

您可能会遇到 OpenShift Container Platform 无法按预期工作的一些情况，例如：

- 您有一个在重启后无法正常工作的集群，因为意外状况（如节点故障或网络连接问题）无法正常工作。
- 您已错误地删除了集群中的某些关键内容。
- 您丢失了大多数 control plane 主机，从而导致 etcd 仲裁丢失。

通过使用保存的 etcd 快照，始终可以通过将集群恢复到之前的状态来从灾难中恢复。

1.2. 应用程序备份和恢复操作

作为集群管理员，您可以使用 OpenShift API 进行数据保护(OADP)来备份和恢复在 OpenShift Container Platform 上运行的应用程序。

使用 Velero 1.7，按照命名空间的粒度备份并恢复 Kubernetes 资源和内部镜像。OADP 使用快照或 Restic 来备份和恢复持久性卷(PV)。详情请查看 OADP 功能。

1.2.1. OADP 要求

OADP 有以下要求：

- 您必须以具有 **cluster-admin** 角色的用户身份登录。
- 您必须具有用于存储备份的对象存储，比如以下存储类型之一：
  - OpenShift Data Foundation
  - Amazon Web Services
  - Microsoft Azure
  - Google Cloud Platform

- S3 兼容对象存储

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

- 要使用快照备份 PV，您必须有具有原生快照 API 的云存储，或者支持 Container Storage Interface(CSI)快照，如以下供应商：
  - Amazon Web Services
  - Microsoft Azure
  - Google Cloud Platform
  - 支持 CSI 快照的云存储，如 Ceph RBD 或 Ceph FS

注意

如果您不想使用快照备份 PV，可以使用 Restic，这由 OADP Operator 安装。

1.2.2. 备份和恢复应用程序

您可以通过创建 **Backup** 自定义资源(CR) 来备份应用程序。您可以配置以下备份选项：

- Backup hooks，在备份操作之前或之后运行命令
- Scheduled backups
- Restic backups

您可以通过创建一个 **Restore** CR 来恢复应用程序。您可以配置 restore hook，以便在 init 容器或应用程序容器中运行命令。

第 2 章 安全地关闭集群

本文档描述了安全关闭集群的过程。出于维护或者节约资源成本的原因，您可能需要临时关闭集群。

2.1. 先决条件

- 在关闭集群前进行 etcd 备份。

2.2. 关闭集群

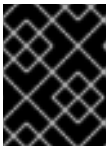
您可以以安全的方式关闭集群，以便稍后重启集群。

注意

您可以在安装日期起的一年内关闭集群，并期望它可以正常重启。安装日期起一年后，集群证书会过期。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。



重要

执行此流程前务必要进行 etcd 备份，以便在重启集群遇到任何问题时可以恢复集群。

流程

1. 如果您要长时间关闭集群，请确定证书过期的日期。

```
$ oc -n openshift-kube-apiserver-operator get secret kube-apiserver-to-kubelet-signer -o jsonpath='{.metadata.annotations.auth\.openshift\.io/certificate-not-after}'
```

输出示例

```
2022-08-05T14:37:50Zuser@user:~ $ 1
```

1 为确保集群可以正常重启，请计划在指定的日期或之前重启集群。当集群重启时，可能需要您手动批准待处理的证书签名请求 (CSR) 来恢复 kubelet 证书。

2. 关闭集群中的所有节点。您可以从云供应商的 Web 控制台完成此操作，或者运行以下循环：

```
$ for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc debug node/${node} -- chroot /host shutdown -h 1; done 1
```

1 **-h 1** 表示此过程在 control-plane 节点关闭前可以持续的时间（以分钟为单位）。对于具有 10 个或更多节点的大型集群，请它设置为 10 分钟或更长时间，以确保所有计算节点都已关闭。

输出示例

```
Starting pod/ip-10-0-130-169us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:17 UTC, use 'shutdown -c' to cancel.

Removing debug pod ...
Starting pod/ip-10-0-150-116us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:29 UTC, use 'shutdown -c' to cancel.
```

使用以下方法关闭节点可让 pod 安全终止，从而减少数据崩溃的可能性。

注意

为大规模集群调整关闭时间：

```
$ for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc debug node/${node} -- chroot /host shutdown -h 10; done
```

注意

在关闭前，不需要排空 OpenShift Container Platform 中附带的标准 pod 的 control plane 节点（也称为 master 节点）。

集群管理员负责确保在集群重启后，彻底重启自己的工作负载。如果因为自定义工作负载的原因已在关闭前排空 control plane 节点，您必须在重启后将 control plane 节点标记为可调度，然后集群才可以重新正常工作。

3. 关闭不再需要的集群依赖项，如外部存储或 LDAP 服务器。在进行操作前请务必查阅您的厂商文档。

其他资源

- 正常重启集群

第 3 章 正常重启集群

本文档论述了在安全关闭后重启集群的过程。

尽管在重启后集群应该可以正常工作，但可能会因为意外状况集群可能无法恢复，例如：

- 关机过程中的 etcd 数据崩溃
- 因硬件原因造成节点故障
- 网络连接问题

如果集群无法恢复，请按照以下步骤 恢复到以前的集群状态。

3.1. 先决条件

- 已安全关闭集群。

3.2. 重启集群

您可以在集群被安全关闭后重启它。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 此流程假设您安全关闭集群。

流程

1. 启动所有依赖设备，如外部存储或 LDAP 服务器。
2. 启动所有集群机器。
   使用适合您的云环境的方法启动机器，例如从云供应商的 Web 控制台启动机器。

   等待大约 10 分钟，然后继续检查 control plane 节点（也称为 master 节点）的状态。
3. 验证所有 control plane 节点都已就绪。

   ```
   $ oc get nodes -l node-role.kubernetes.io/master
   ```

   如果状态为 **Ready**，如以下输出中所示，则代表 control plane 节点已就绪：

   ```
   NAME                           STATUS   ROLES    AGE   VERSION
   ip-10-0-168-251.ec2.internal   Ready    master   75m   v1.21.0
   ip-10-0-170-223.ec2.internal   Ready    master   75m   v1.21.0
   ip-10-0-211-16.ec2.internal    Ready    master   75m   v1.21.0
   ```
4. 如果 control plane 节点*没有*就绪，请检查是否有待批准的证书签名请求 (CSR)。
   a. 获取当前 CSR 列表：

      ```
      $ oc get csr
      ```

b. 查看一个 CSR 的详细信息以验证其是否有效：

```
$ oc describe csr <csr_name> 1
```

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 CSR：

```
$ oc adm certificate approve <csr_name>
```

5. 在 control plane 节点就绪后，验证所有 worker 节点是否已就绪。

```
$ oc get nodes -l node-role.kubernetes.io/worker
```

如果状态为 **Ready**，如下所示，则代表 worker 节点已就绪：

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-179-95.ec2.internal   Ready    worker   64m   v1.21.0
ip-10-0-182-134.ec2.internal  Ready    worker   64m   v1.21.0
ip-10-0-250-100.ec2.internal  Ready    worker   64m   v1.21.0
```

6. 如果 worker 节点 *未* 就绪，请检查是否有待批准的证书签名请求(CSR)。

a. 获取当前 CSR 列表：

```
$ oc get csr
```

b. 查看一个 CSR 的详细信息以验证其是否有效：

```
$ oc describe csr <csr_name> 1
```

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 CSR：

```
$ oc adm certificate approve <csr_name>
```

7. 验证集群是否已正确启动。

a. 检查是否有降级的集群 Operator。

```
$ oc get clusteroperators
```

确定没有 **DEGRADED** 条件为 **True** 的集群 Operator。

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.8.0     True        False         False     59m
cloud-credential                           4.8.0     True        False         False     85m
cluster-autoscaler                         4.8.0     True        False         False     73m
config-operator                            4.8.0     True        False         False     73m
```

```
console                                    4.8.0     True        False         False      62m
csi-snapshot-controller                    4.8.0     True        False         False      66m
dns                                        4.8.0     True        False         False      76m
etcd                                       4.8.0     True        False         False      76m
...
```

b. 检查所有节点是否处于 **Ready** 状态：

```
$ oc get nodes
```

检查所有节点的状态是否为 **Ready**。

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.21.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.21.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.21.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.21.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.21.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.21.0
```

如果集群无法正确启动，您可能需要使用 etcd 备份来恢复集群。

其他资源

- 如果您的集群在重启后无法正常运行，请参阅恢复到以前的集群状态。

第 4 章 应用程序备份和恢复

4.1. OADP 功能和插件

OpenShift API 用于数据保护(OADP)功能，提供用于备份和恢复应用的选项。

默认插件使 Velero 能够与某些云供应商集成，并备份和恢复 OpenShift Container Platform 资源。

4.1.1. OADP 功能

OpenShift API 用于数据保护(OADP)支持以下功能：

Backup

您可以备份集群中的所有资源，或者您可以按类型、命名空间或标签过滤资源。
OADP 通过将 Kubernetes 对象和内部镜像保存为对象存储上的存档文件来备份 Kubernetes 对象和内部镜像。OADP 使用原生云快照 API 或通过容器存储接口(CSI)创建快照来备份持久性卷(PV)。对于不支持快照的云供应商，OADP 使用 Restic 备份资源和 PV 数据。

恢复

您可以从备份中恢复资源和 PV。您可以恢复备份中的所有对象，或者根据命名空间、PV 或标签过滤恢复的对象。

调度

您可以通过指定的间隔调度备份。

Hook

您可以使用 hook 在 pod 上的容器中运行命令，如 **fsfreeze** 以冻结文件系统。您可以将 hook 配置为在备份或恢复之前或之后运行。恢复 hook 可以在 init 容器或应用程序容器中运行。

4.1.2. OADP 插件

用于数据保护(OADP)的 OpenShift API 提供了与存储供应商集成的默认 Velero 插件，以支持备份和恢复操作。您可以根据 Velero 插件创建自定义插件。

OADP 还为 OpenShift Container Platform 资源备份和 Container Storage Interface(CSI)快照提供了插件。

表 4.1. OADP 插件

OADP 插件	功能	存储位置
aws	使用对象存储备份和恢复 Kubernetes 对象。	AWS S3
	使用快照备份和恢复卷。	AWS EBS
azure	使用对象存储备份和恢复 Kubernetes 对象。	Microsoft Azure Blob 存储
	使用快照备份和恢复卷。	Microsoft Azure 管理的磁盘

OADP 插件	功能	存储位置
gcp	使用对象存储备份和恢复 Kubernetes 对象。	Google Cloud Storage
	使用快照备份和恢复卷。	Google Compute Engine 磁盘
openshift	使用对象存储备份和恢复 OpenShift Container Platform 资源。[1]	对象存储
csi	使用 CSI 快照备份和恢复卷。[2]	支持 CSI 快照的云存储

1. 必需。
2. **csi** 插件使用 Velero CSI beta 快照 API。

4.1.3. 关于 OADP Velero 插件

安装 Velero 时，您可以配置两种类型的插件：

- 默认云供应商插件
- 自定义插件

两种类型的插件都是可选的，但大多数用户都会至少配置一个云供应商插件。

4.1.3.1. 默认 Velero 云供应商插件

当您在部署过程中配置 **oadp_v1alpha1_dpa.yaml** 文件时，您可以安装以下默认 Velero 云供应商插件：

- **aws** (Amazon Web Services)
- **gcp** (Google Cloud Platform)
- **azure** (Microsoft Azure)
- **openshift** (OpenShift Velero plugin)
- **csi** (Container Storage Interface)
- **kubevirt** (KubeVirt)

在部署过程中，您可以在 **oadp_v1alpha1_dpa.yaml** 文件中指定所需的默认插件。

示例文件

以下 **.yaml** 文件会安装 **openshift**、**aws**、**azure** 和 **gcp** 插件：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: dpa-sample
```

```
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - aws
      - azure
      - gcp
```

4.1.3.2. 自定义 Velero 插件

您可在部署期间配置 **oadp_v1alpha1_dpa.yaml** 文件时，通过指定插件 **镜像**和**名称**来安装自定义 Velero 插件。

在部署过程中，您可以在 **oadp_v1alpha1_dpa.yaml** 文件中指定所需的自定义插件。

示例文件

以下 **.yaml** 文件会安装默认的 **openshift**、**azure** 和 **gcp** 插件，以及一个自定义插件，其名称为 **custom-plugin-example** 和镜像 **quay.io/example-repo/custom-velero-plugin** ：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
 name: dpa-sample
spec:
 configuration:
   velero:
     defaultPlugins:
     - openshift
     - azure
     - gcp
     customPlugins:
     - name: custom-plugin-example
       image: quay.io/example-repo/custom-velero-plugin
```

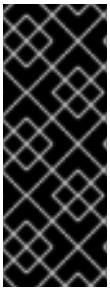
4.2. 安装和配置 OADP

4.2.1. 关于安装 OADP

作为集群管理员，您可以通过安装 OADP Operator 来为数据保护(OADP)安装 OpenShift API。OADP Operator 会安装 Velero 1.7。

要备份 Kubernetes 资源和内部镜像，必须将对象存储用作备份位置，如以下存储类型之一：

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- 多云对象网关
- S3 兼容对象存储，如 Noobaa 或 Minio



重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

您可以使用快照或 Restic 备份持久性卷(PV)。

要使用快照备份 PV，您必须有一个支持原生快照 API 或 Container Storage Interface(CSI)快照的云供应商，如以下云供应商之一：

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform
- 支持 CSI 快照的云供应商，如 OpenShift Container Storage

如果您的云供应商不支持快照，或者您的存储是 NFS，您可以使用 Restic 备份应用程序。

您可以为存储供应商凭证创建一个 **Secret** 对象，然后安装数据保护应用程序。

其他资源

- Velero 文档中的备份位置和快照位置概述。

4.2.2. 为 Amazon Web Services 进行数据保护安装和配置 OpenShift API

您可以通过安装 OADP Operator、为 Velero 配置 AWS，然后安装数据保护应用程序，安装带有 Amazon Web Services(AWS)的数据保护(OADP)的 OpenShift API。

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

要在受限网络环境中安装 OADP Operator，您必须首先禁用默认的 OperatorHub 源并镜像 Operator 目录。详情请参阅在受限网络中使用 Operator Lifecycle Manager。

4.2.2.1. 安装 OADP Operator

您可以使用 Operator Lifecycle Manager(OLM)在 OpenShift Container Platform 4.8 上安装 Data Protection(OADP)Operator 的 OpenShift API。

OADP Operator 会安装 Velero 1.7。

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 Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **OADP Operator**。
3. 选择 **OADP Operator** 并点 **Install**。
4. 点 **Install** 在 **openshift-adp** 项目中安装 Operator。
5. 点 **Operators → Installed Operators** 来验证安装。

4.2.2.2. 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
    }
}
```

8. 创建 **credentials-velero** 文件：

```
$ cat << EOF > ./credentials-velero
[default]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>
EOF
```

在安装数据保护应用程序前，您可以使用 **credentials-velero** 文件为 AWS 创建 **Secret** 对象。

4.2.2.3. 创建用于备份和快照位置的 secret

如果使用相同的凭证，您可以为备份和快照位置创建一个 **Secret** 对象。

Secret 的默认名称为 **cloud-credentials**。

先决条件

- 对象存储和云存储必须使用相同的凭证。
- 您必须为 Velero 配置对象存储。

- 您必须以适当的格式为对象存储创建一个 **credentials-velero** 文件。

注意

DataProtectionApplication 自定义资源(CR)需要一个 **Secret** 进行安装。如果没有指定 **spec.backupLocations.credential.name** 值，则会使用默认名称。

如果您不想指定备份位置或快照位置，则必须使用空 **credentials-velero** 文件创建带有默认名称的 **Secret**。

流程

- 使用默认名称创建 **Secret** ：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

在安装 Data Protection Application 时，**secret** 会在 **DataProtectionApplication** CR 的 **spec.backupLocations.credential** 块中引用。

4.2.2.3.1. 为不同的备份和恢复位置凭证配置 secret

如果您的备份和快照位置使用不同的凭证，您可以在 **credentials-velero** 文件中创建单独的配置集。

然后，您可以创建一个 **Secret** 对象并在 **DataProtectionApplication** 自定义资源(CR)中指定配置集。

流程

1. 使用备份和快照位置的独立配置集创建一个 **credentials-velero** 文件，如下例所示：

```
[backupStorage]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>

[volumeSnapshot]
aws_access_key_id=<AWS_ACCESS_KEY_ID>
aws_secret_access_key=<AWS_SECRET_ACCESS_KEY>
```

2. 使用 **credentials-velero** 文件创建 **Secret** 对象：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero 1
```

3. 在 **DataProtectionApplication** CR 中添加配置集，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - name: default
```

```
    velero:
      provider: aws
      default: true
      objectStorage:
        bucket: <bucket_name>
        prefix: <prefix>
      config:
        region: us-east-1
        profile: "backupStorage"
      credential:
        key: cloud
        name: cloud-credentials
  snapshotLocations:
    - name: default
      velero:
        provider: aws
        config:
          region: us-west-2
          profile: "volumeSnapshot"
```

4.2.2.4. 配置数据保护应用程序

您可以配置 Velero 资源分配并启用自签名 CA 证书。

4.2.2.4.1. 设置 Velero CPU 和内存分配

您可以通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为 **Velero** pod 设置 CPU 和内存分配。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.configuration.velero.podConfig.ResourceAllocations** 块中的值，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

1 2 1 1 在 millicpus 或 CPU 单元中指定值。默认值为 **500m** 或 **1** CPU 单元。

2 默认值为 **512Mi**。

3 默认值为 **500m** 或 **1** CPU 单元。

4 默认值为 **256Mi**。

4.2.2.4.2. 启用自签名 CA 证书

您必须通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为对象存储启用自签名 CA 证书，以防止由未知颁发机构签名的证书。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.backupLocations.velero.objectStorage.caCert** 参数和 **spec.backupLocations.velero.config** 参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
        config:
          insecureSkipTLSVerify: "false" 2
...
```

1 指定 Base46 编码的 CA 证书字符串。

2 必须为 **false** 来禁用 SSL/TLS 安全性。

4.2.2.5. 安装数据保护应用程序

您可以通过创建 **DataProtectionApplication** API 的实例来安装数据保护应用程序(DPA)。

先决条件

- 您必须安装 OADP Operator。

- 您必须将对象存储配置为备份位置。
- 如果使用快照来备份 PV，云供应商必须支持原生快照 API 或 Container Storage Interface(CSI) 快照。
- 如果备份和快照位置使用相同的凭证，您必须创建带有默认名称 **cloud-credentials** 的 **Secret**。
- 如果备份和快照位置使用不同的凭证，则必须使用默认名称 **cloud-credentials** 创建一个 **Secret**，其中包含备份和快照位置凭证的独立配置集。

注意

如果您不想在安装过程中指定备份或快照位置，您可以使用空 **credentials-velero** 文件创建默认 **Secret**。如果没有默认 **Secret**，安装将失败。

流程

1. 点 **Operators → Installed Operators** 并选择 OADP Operator。
2. 在 **Provided APIs** 下，点 **DataProtectionApplication** 框中的 **Create 实例**。
3. 点 **YAML View** 并更新 **DataProtectionApplication** 清单的参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - openshift (1)
        - aws
    restic:
      enable: true (2)
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket_name> (3)
          prefix: <prefix> (4)
        config:
          region: <region>
          profile: "default"
        credential:
          key: cloud
          name: cloud-credentials (5)
  snapshotLocations: (6)
    - name: default
      velero:
        provider: aws
```

```
    config:
      region: <region> 7
      profile: "default"
```

1. **openshift** 插件是必须的，才能在 OpenShift Container Platform 集群上备份和恢复命名空间。
2. 如果要禁用 Restic 安装，设置为 **false**。Restic 部署一个守护进程集，这意味着每个 worker 节点都运行有 **Restic** pod。您可以通过在 **Backup** CR 中添加 **spec.defaultVolumesToRestic: true** 来配置 Restic 以进行备份。
3. 指定存储桶作为备份存储位置。如果存储桶不是 Velero 备份的专用存储桶，您必须指定一个前缀。
4. 如果存储桶用于多个目的，请为 Velero 备份指定一个前缀，如 **velero**。
5. 指定您创建的 **Secret** 对象的名称。如果没有指定这个值，则使用默认值 **cloud-credentials**。如果您指定了自定义名称，则使用自定义名称进行备份位置。
6. 如果您使用 CSI 快照或 Restic 备份 PV，则不需要指定快照位置。
7. 快照位置必须与 PV 位于同一区域。

4. 点 **Create**。
5. 通过查看 OADP 资源来验证安装：

```
$ oc get all -n openshift-adp
```

输出示例

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                     1/1     Running   0          94s
pod/restic-m4lts                                     1/1     Running   0          94s
pod/restic-pv4kr                                     1/1     Running   0          95s
pod/velero-588db7f655-n842v                          1/1     Running   0          95s

NAME                                                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager   1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                             1/1     1            1           96s
```

```
NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47   1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd   1         1         1       96s
replicaset.apps/velero-588db7f655                             1         1         1       96s
```

4.2.2.5.1. 在 DataProtectionApplication CR 中启用 CSI

您可以在 **DataProtectionApplication** 自定义资源(CR)中启用 Container Storage Interface(CSI)来备份持久性卷，以使用 CSI 快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。

流程

- 编辑 **DataProtectionApplication** CR，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) 添加 **csi** 默认插件。

(2) 添加 **EnableCSI** 功能标志。

4.2.3. 为 Microsoft Azure 的数据保护安装和配置 OpenShift API

您可以通过安装 OADP Operator、为 Velero 配置 Azure，然后安装数据保护应用程序，使用 Microsoft Azure 安装 OpenShift API for Data ProtectionP。

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

要在受限网络环境中安装 OADP Operator，您必须首先禁用默认的 OperatorHub 源并镜像 Operator 目录。详情请参阅在受限网络中使用 Operator Lifecycle Manager。

4.2.3.1. 安装 OADP Operator

您可以使用 Operator Lifecycle Manager(OLM)在 OpenShift Container Platform 4.8 上安装 Data Protection(OADP)Operator 的 OpenShift API。

OADP Operator 会安装 Velero 1.7。

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 Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **OADP Operator**。
3. 选择 **OADP Operator** 并点 **Install**。
4. 点 **Install** 在 **openshift-adp** 项目中安装 Operator。
5. 点 **Operators → Installed Operators** 来验证安装。

4.2.3.2. 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. 获取存储帐户访问密钥：

```
$ AZURE_STORAGE_ACCOUNT_ACCESS_KEY=`az storage account keys list \
  --account-name $AZURE_STORAGE_ACCOUNT_ID \
  --query "[?keyName == 'key1'].value" -o tsv`
```

8. 创建 **credentials-velero** 文件：

```
$ 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=${AZURE_RESOURCE_GROUP}
AZURE_STORAGE_ACCOUNT_ACCESS_KEY=${AZURE_STORAGE_ACCOUNT_ACCES
S_KEY} 1
AZURE_CLOUD_NAME=AzurePublicCloud
EOF
```

1 必需。如果 **credentials-velero** 文件只包含服务主体凭证，则无法备份内部镜像。

在安装 Data Protection 应用前，您可以使用 **credentials-velero** 文件为 Azure 创建 **Secret** 对象。

4.2.3.3. 创建用于备份和快照位置的 secret

如果使用相同的凭证，您可以为备份和快照位置创建一个 **Secret** 对象。

Secret 的默认名称为 **cloud-credentials-azure**。

先决条件

- 对象存储和云存储必须使用相同的凭证。
- 您必须为 Velero 配置对象存储。
- 您必须以适当的格式为对象存储创建一个 **credentials-velero** 文件。

注意

DataProtectionApplication 自定义资源(CR)需要一个 **Secret** 进行安装。如果没有指定 **spec.backupLocations.credential.name** 值，则会使用默认名称。

如果您不想指定备份位置或快照位置，则必须使用空 **credentials-velero** 文件创建带有默认名称的 **Secret**。

流程

- 使用默认名称创建 **Secret** ：

```
$ oc create secret generic cloud-credentials-azure -n openshift-adp --from-file
cloud=credentials-velero
```

在安装 Data Protection Application 时，**secret** 会在 **DataProtectionApplication** CR 的 **spec.backupLocations.credential** 块中引用。

4.2.3.3.1. 为不同的备份和恢复位置凭证配置 secret

如果您的备份和恢复位置使用不同的凭证，请创建两个 **Secret** 对象：

- 具有自定义名称的备份位置 **Secret**。自定义名称在 **DataProtectionApplication** 自定义资源(CR)的 **spec.backupLocations** 块中指定。
- 带有默认名称 **cloud-credentials-azure** 的快照位置 **Secret**。此 **Secret** 不在 **DataProtectionApplication** CR 中指定。

流程

1. 为您的云供应商为快照位置创建一个 **credentials-velero** 文件。
2. 使用默认名称为快照位置创建 **Secret** ：

```
$ oc create secret generic cloud-credentials-azure -n openshift-adp --from-file
cloud=credentials-velero
```

3. 为您的对象存储创建一个用于备份位置的 **credentials-velero** 文件。
4. 使用自定义名称为备份位置创建 **Secret** ：

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-
velero
```

5. 将带有自定义名称的 **Secret** 添加到 **DataProtectionApplication** CR 中，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        config:
          resourceGroup: <azure_resource_group>
          storageAccount: <azure_storage_account_id>
          subscriptionId: <azure_subscription_id>
          storageAccountKeyEnvVar: AZURE_STORAGE_ACCOUNT_ACCESS_KEY
        credential:
          key: cloud
          name: <custom_secret> (1)
        provider: azure
        default: true
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
  snapshotLocations:
    - velero:
        config:
          resourceGroup: <azure_resource_group>
          subscriptionId: <azure_subscription_id>
          incremental: "true"
        name: default
        provider: azure
```

(1) 具有自定义名称的备份位置 **Secret**。

4.2.3.4. 配置数据保护应用程序

您可以配置 Velero 资源分配并启用自签名 CA 证书。

4.2.3.4.1. 设置 Velero CPU 和内存分配

您可以通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为 **Velero** pod 设置 CPU 和内存分配。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.configuration.velero.podConfig.ResourceAllocations** 块中的值，如下例所示：

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

```
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        resourceAllocations:
          limits:
            cpu: "1" 1
            memory: 512Mi 2
          requests:
            cpu: 500m 3
            memory: 256Mi 4
```

1. 在 millicpus 或 CPU 单元中指定值。默认值为 **500m** 或 **1** CPU 单元。
2. 默认值为 **512Mi**。
3. 默认值为 **500m** 或 **1** CPU 单元。
4. 默认值为 **256Mi**。

4.2.3.4.2. 启用自签名 CA 证书

您必须通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为对象存储启用自签名 CA 证书，以防止**由未知颁发机构签名的**证书。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.backupLocations.velero.objectStorage.caCert** 参数和 **spec.backupLocations.velero.config** 参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
```

```
      config:
        insecureSkipTLSVerify: "false" 2
...
```

1 指定 Base46 编码的 CA 证书字符串。

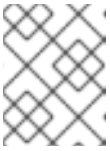
2 必须为 **false** 来禁用 SSL/TLS 安全性。

4.2.3.5. 安装数据保护应用程序

您可以通过创建 **DataProtectionApplication** API 的实例来安装数据保护应用程序(DPA)。

先决条件

- 您必须安装 OADP Operator。
- 您必须将对象存储配置为备份位置。
- 如果使用快照来备份 PV，云供应商必须支持原生快照 API 或 Container Storage Interface(CSI)快照。
- 如果备份和快照位置使用相同的凭证，您必须创建带有默认名称 **cloud-credentials-azure** 的 **Secret**。
- 如果备份和快照位置使用不同的凭证，您必须创建两个 **Secret** ：
  - 带有备份位置的自定义名称的 **secret**。您可以将此 **Secret** 添加到 **DataProtectionApplication** CR 中。
  - 带有默认名称 **cloud-credentials-azure** 的 **secret**，用于快照位置。这个 **Secret** 不在 **DataProtectionApplication** CR 中被引用。



注意

如果您不想在安装过程中指定备份或快照位置，您可以使用空 **credentials-velero** 文件创建默认 **Secret**。如果没有默认 **Secret**，安装将失败。

流程

1. 点 **Operators → Installed Operators** 并选择 OADP Operator。
2. 在 **Provided APIs** 下，点 **DataProtectionApplication** 框中的 **Create 实例**。
3. 点 **YAML View** 并更新 **DataProtectionApplication** 清单的参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - azure
```

```
      - openshift 1
    restic:
      enable: true 2
  backupLocations:
    - velero:
        config:
          resourceGroup: <azure_resource_group> 3
          storageAccount: <azure_storage_account_id> 4
          subscriptionId: <azure_subscription_id> 5
          storageAccountKeyEnvVar: AZURE_STORAGE_ACCOUNT_ACCESS_KEY
        credential:
          key: cloud
          name: cloud-credentials-azure 6
        provider: azure
        default: true
        objectStorage:
          bucket: <bucket_name> 7
          prefix: <prefix> 8
  snapshotLocations: 9
    - velero:
        config:
          resourceGroup: <azure_resource_group>
          subscriptionId: <azure_subscription_id>
          incremental: "true"
        name: default
        provider: azure
```

1. **openshift** 插件是必须的，才能在 OpenShift Container Platform 集群上备份和恢复命名空间。
2. 如果要禁用 Restic 安装，设置为 **false**。Restic 部署一个守护进程集，这意味着每个 worker 节点都运行有 **Restic** pod。您可以通过在 **Backup** CR 中添加 **spec.defaultVolumesToRestic: true** 来配置 Restic 以进行备份。
3. 指定 Azure 资源组。
4. 指定 Azure 存储帐户 ID。
5. 指定 Azure 订阅 ID。
6. 如果没有指定这个值，则使用默认值 **cloud-credentials-azure**。如果您指定了自定义名称，则使用自定义名称进行备份位置。
7. 指定存储桶作为备份存储位置。如果存储桶不是 Velero 备份的专用存储桶，您必须指定一个前缀。
8. 如果存储桶用于多个目的，请为 Velero 备份指定一个前缀，如 **velero**。
9. 如果您使用 CSI 快照或 Restic 备份 PV，则不需要指定快照位置。

4. 点 **Create**。
5. 通过查看 OADP 资源来验证安装：

```
$ oc get all -n openshift-adp
```

输出示例

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                     1/1     Running   0          94s
pod/restic-m4lts                                     1/1     Running   0          94s
pod/restic-pv4kr                                     1/1     Running   0          95s
pod/velero-588db7f655-n842v                          1/1     Running   0          95s

NAME                                                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager   1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry   1/1     1            1           96s
deployment.apps/velero                             1/1     1            1           96s

NAME                                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47   1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd   1         1         1       96s
replicaset.apps/velero-588db7f655                          1         1         1       96s
```

4.2.3.5.1. 在 DataProtectionApplication CR 中启用 CSI

您可以在 **DataProtectionApplication** 自定义资源(CR)中启用 Container Storage Interface(CSI)来备份持久性卷，以使用 CSI 快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。

流程

- 编辑 **DataProtectionApplication** CR，如下例所示：

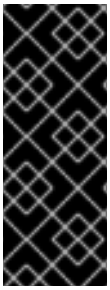
```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
```

```
    - csi (1)
  featureFlags:
  - EnableCSI (2)
```

1. 添加 **csi** 默认插件。
2. 添加 **EnableCSI** 功能标志。

4.2.4. 安装和配置 OpenShift API 以进行 Google Cloud Platform 的数据保护

您可以通过安装 OADP Operator、为 Velero 配置 GCP，然后安装数据保护应用程序，使用 Google Cloud Platform(GCP)安装 OpenShift API for Data Protection(GCP)。



重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

要在受限网络环境中安装 OADP Operator，您必须首先禁用默认的 OperatorHub 源并镜像 Operator 目录。详情请参阅在受限网络中使用 Operator Lifecycle Manager。

4.2.4.1. 安装 OADP Operator

您可以使用 Operator Lifecycle Manager(OLM)在 OpenShift Container Platform 4.8 上安装 Data Protection(OADP)Operator 的 OpenShift API。

OADP Operator 会安装 Velero 1.7。

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 Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **OADP Operator**。
3. 选择 **OADP Operator** 并点 **Install**。
4. 点 **Install** 在 **openshift-adp** 项目中安装 Operator。
5. 点 **Operators → Installed Operators** 来验证安装。

4.2.4.2. 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 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
```

在安装 Data Protection Application 前，您可以使用 **credentials-velero** 文件为 GCP 创建 **Secret** 对象。

4.2.4.3. 创建用于备份和快照位置的 secret

如果使用相同的凭证，您可以为备份和快照位置创建一个 **Secret** 对象。

Secret 的默认名称为 **cloud-credentials-gcp**。

先决条件

- 对象存储和云存储必须使用相同的凭证。
- 您必须为 Velero 配置对象存储。
- 您必须以适当的格式为对象存储创建一个 **credentials-velero** 文件。

流程

- 使用默认名称创建 **Secret** ：

```
$ oc create secret generic cloud-credentials-gcp -n openshift-adp --from-file
cloud=credentials-velero
```

在安装 Data Protection Application 时，**secret** 会在 **DataProtectionApplication** CR 的 **spec.backupLocations.credential** 块中引用。

4.2.4.3.1. 为不同的备份和恢复位置凭证配置 secret

如果您的备份和恢复位置使用不同的凭证，请创建两个 **Secret** 对象：

- 具有自定义名称的备份位置 **Secret**。自定义名称在 **DataProtectionApplication** 自定义资源(CR)的 **spec.backupLocations** 块中指定。
- 带有默认名称 **cloud-credentials-gcp** 的快照位置 **Secret**。此 **Secret** 不在 **DataProtectionApplication** CR 中指定。

流程

1. 为您的云供应商为快照位置创建一个 **credentials-velero** 文件。
2. 使用默认名称为快照位置创建 **Secret** ：

```
$ oc create secret generic cloud-credentials-gcp -n openshift-adp --from-file
cloud=credentials-velero
```

3. 为您的对象存储创建一个用于备份位置的 **credentials-velero** 文件。
4. 使用自定义名称为备份位置创建 **Secret** ：

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-
velero
```

5. 将带有自定义名称的 **Secret** 添加到 **DataProtectionApplication** CR 中，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
...
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
          name: <custom_secret> 1
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
  snapshotLocations:
```

```
  - velero:
      provider: gcp
      default: true
      config:
        project: <project>
        snapshotLocation: us-west1
```

1 具有自定义名称的备份位置 **Secret**。

4.2.4.4. 配置数据保护应用程序

您可以配置 Velero 资源分配并启用自签名 CA 证书。

4.2.4.4.1. 设置 Velero CPU 和内存分配

您可以通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为 **Velero** pod 设置 CPU 和内存分配。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.configuration.velero.podConfig.ResourceAllocations** 块中的值，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        resourceAllocations:
          limits:
            cpu: "1" 1
            memory: 512Mi 2
          requests:
            cpu: 500m 3
            memory: 256Mi 4
```

1 在 millicpus 或 CPU 单元中指定值。默认值为 **500m** 或 **1** CPU 单元。

2 默认值为 **512Mi**。

3 默认值为 **500m** 或 **1** CPU 单元。

4 默认值为 **256Mi**。

4.2.4.4.2. 启用自签名 CA 证书

您必须通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为对象存储启用自签名 CA 证书，以防止由未知颁发机构签名的证书。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.backupLocations.velero.objectStorage.caCert** 参数和 **spec.backupLocations.velero.config** 参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
        config:
          insecureSkipTLSVerify: "false" 2
...
```

1. 指定 Base46 编码的 CA 证书字符串。
2. 必须为 **false** 来禁用 SSL/TLS 安全性。

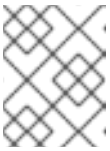
4.2.4.5. 安装数据保护应用程序

您可以通过创建 **DataProtectionApplication** API 的实例来安装数据保护应用程序(DPA)。

先决条件

- 您必须安装 OADP Operator。
- 您必须将对象存储配置为备份位置。
- 如果使用快照来备份 PV，云供应商必须支持原生快照 API 或 Container Storage Interface(CSI)快照。
- 如果备份和快照位置使用相同的凭证，您必须创建带有默认名称 **cloud-credentials-gcp** 的 **Secret**。

- 如果备份和快照位置使用不同的凭证，您必须创建两个 **Secret** ：
  - 带有备份位置的自定义名称的 **secret**。您可以将此 **Secret** 添加到 **DataProtectionApplication** CR 中。
  - 带有默认名称 **cloud-credentials-gcp** 的 **secret**，用于快照位置。这个 **Secret** 不在 **DataProtectionApplication** CR 中被引用。



注意

如果您不想在安装过程中指定备份或快照位置，您可以使用空 **credentials-velero** 文件创建默认 **Secret**。如果没有默认 **Secret**，安装将失败。

流程

1. 点 **Operators → Installed Operators** 并选择 OADP Operator。
2. 在 **Provided APIs** 下，点 **DataProtectionApplication** 框中的 **Create 实例**。
3. 点 **YAML View** 并更新 **DataProtectionApplication** 清单的参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - gcp
        - openshift (1)
    restic:
      enable: true (2)
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
          name: cloud-credentials-gcp (3)
        objectStorage:
          bucket: <bucket_name> (4)
          prefix: <prefix> (5)
  snapshotLocations: (6)
    - velero:
        provider: gcp
        default: true
        config:
          project: <project>
          snapshotLocation: us-west1 (7)
```

(1) **openshift** 插件是必须的，才能在 OpenShift Container Platform 集群上备份和恢复命名空间。

2. 如果要禁用 Restic 安装，设置为 **false**。Restic 部署一个守护进程集，这意味着每个 worker 节点都运行有 **Restic** pod。您可以通过在 **Backup** CR 中添加
3. 如果没有指定这个值，则使用默认值 **cloud-credentials-gcp**。如果您指定了自定义名称，则使用自定义名称进行备份位置。
4. 指定存储桶作为备份存储位置。如果存储桶不是 Velero 备份的专用存储桶，您必须指定一个前缀。
5. 如果存储桶用于多个目的，请为 Velero 备份指定一个前缀，如 **velero**。
6. 如果您使用 CSI 快照或 Restic 备份 PV，则不需要指定快照位置。
7. 快照位置必须与 PV 位于同一区域。

4. 点 **Create**。
5. 通过查看 OADP 资源来验证安装：

```
$ oc get all -n openshift-adp
```

输出示例

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                     1/1     Running   0          94s
pod/restic-m4lts                                     1/1     Running   0          94s
pod/restic-pv4kr                                     1/1     Running   0          95s
pod/velero-588db7f655-n842v                          1/1     Running   0          95s

NAME                                                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager     1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry    1/1     1            1           96s
deployment.apps/velero                               1/1     1            1           96s

NAME                                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47     1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd    1         1         1       96s
replicaset.apps/velero-588db7f655                               1         1         1       96s
```

4.2.4.5.1. 在 DataProtectionApplication CR 中启用 CSI

您可以在 **DataProtectionApplication** 自定义资源(CR)中启用 Container Storage Interface(CSI)来备份持久性卷，以使用 CSI 快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。

流程

- 编辑 **DataProtectionApplication** CR，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) 添加 **csi** 默认插件。

(2) 添加 **EnableCSI** 功能标志。

4.2.5. 安装和配置 OpenShift API 以进行 Google Cloud Platform 的数据保护

您可以通过安装 OADP Operator、创建 **Secret** 对象，然后安装数据保护应用程序，安装带有 Multicloud Object Gateway(MCG)的数据保护(OADP)的 OpenShift API。

MCG 是 OpenShift Container Storage(OCS)的一个组件。您可以将 MCG 配置为 **DataProtectionApplication** 自定义资源(CR)中的备份位置。

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

如果您的云供应商具有原生快照 API，请配置快照位置。如果您的云供应商不支持快照，或者存储是 NFS，您可以使用 Restic 创建备份。

您不需要在 Restic 或 Container Storage Interface(CSI)快照的 **DataProtectionApplication** CR 中指定快照位置。

要在受限网络环境中安装 OADP Operator，您必须首先禁用默认的 OperatorHub 源并镜像 Operator 目录。详情请参阅在受限网络中使用 Operator Lifecycle Manager。

4.2.5.1. 安装 OADP Operator

您可以使用 Operator Lifecycle Manager(OLM)在 OpenShift Container Platform 4.8 上安装 Data Protection(OADP)Operator 的 OpenShift API。

OADP Operator 会安装 Velero 1.7。

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 Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **OADP Operator**。
3. 选择 **OADP Operator** 并点 **Install**。
4. 点 **Install** 在 **openshift-adp** 项目中安装 Operator。
5. 点 **Operators → Installed Operators** 来验证安装。

4.2.5.2. 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。

4.2.5.2.1. 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**。

4.2.5.2.2. 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
```

4.2.5.3. 创建用于备份和快照位置的 secret

如果使用相同的凭证，您可以为备份和快照位置创建一个 **Secret** 对象。

Secret 的默认名称为 **cloud-credentials**。

先决条件

- 对象存储和云存储必须使用相同的凭证。
- 您必须为 Velero 配置对象存储。
- 您必须以适当的格式为对象存储创建一个 **credentials-velero** 文件。

流程

- 使用默认名称创建 **Secret** ：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

在安装 Data Protection Application 时，**secret** 会在 **DataProtectionApplication** CR 的 **spec.backupLocations.credential** 块中引用。

4.2.5.3.1. 为不同的备份和恢复位置凭证配置 secret

如果您的备份和恢复位置使用不同的凭证，请创建两个 **Secret** 对象：

- 具有自定义名称的备份位置 **Secret**。自定义名称在 **DataProtectionApplication** 自定义资源(CR) 的 **spec.backupLocations** 块中指定。
- 带有默认名称 **cloud-credentials** 的快照位置 **Secret**。此 **Secret** 不在 **DataProtectionApplication** CR 中指定。

流程

1. 为您的云供应商为快照位置创建一个 **credentials-velero** 文件。
2. 使用默认名称为快照位置创建 **Secret** ：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

3. 为您的对象存储创建一个用于备份位置的 **credentials-velero** 文件。
4. 使用自定义名称为备份位置创建 **Secret** ：

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-velero
```

5. 将带有自定义名称的 **Secret** 添加到 **DataProtectionApplication** CR 中，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - aws
        - openshift
    restic:
      enable: true
  backupLocations:
    - velero:
        config:
          profile: "default"
          region: minio
          s3Url: <url>
          insecureSkipTLSVerify: "true"
          s3ForcePathStyle: "true"
        provider: aws
        default: true
        credential:
          key: cloud
          name:  <custom_secret> (1)
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
```

(1) 具有自定义名称的备份位置 **Secret**。

4.2.5.4. 配置数据保护应用程序

您可以配置 Velero 资源分配并启用自签名 CA 证书。

4.2.5.4.1. 设置 Velero CPU 和内存分配

您可以通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为 **Velero** pod 设置 CPU 和内存分配。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.configuration.velero.podConfig.ResourceAllocations** 块中的值，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        resourceAllocations:
          limits:
            cpu: "1" (1)
            memory: 512Mi (2)
          requests:
            cpu: 500m (3)
            memory: 256Mi (4)
```

1. 在 millicpus 或 CPU 单元中指定值。默认值为 **500m** 或 **1** CPU 单元。
2. 默认值为 **512Mi**。
3. 默认值为 **500m** 或 **1** CPU 单元。
4. 默认值为 **256Mi**。

4.2.5.4.2. 启用自签名 CA 证书

您必须通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为对象存储启用自签名 CA 证书，以防止**由未知颁发机构签名的**证书。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.backupLocations.velero.objectStorage.caCert** 参数和 **spec.backupLocations.velero.config** 参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
```

```
      objectStorage:
        bucket: <bucket>
        prefix: <prefix>
        caCert: <base64_encoded_cert_string> 1
      config:
        insecureSkipTLSVerify: "false" 2
...
```

1 指定 Base46 编码的 CA 证书字符串。

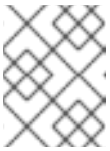
2 必须为 **false** 来禁用 SSL/TLS 安全性。

4.2.5.5. 安装数据保护应用程序

您可以通过创建 **DataProtectionApplication** API 的实例来安装数据保护应用程序(DPA)。

先决条件

- 您必须安装 OADP Operator。
- 您必须将对象存储配置为备份位置。
- 如果使用快照来备份 PV，云供应商必须支持原生快照 API 或 Container Storage Interface(CSI) 快照。
- 如果备份和快照位置使用相同的凭证，您必须创建带有默认名称 **cloud-credentials** 的 **Secret**。
- 如果备份和快照位置使用不同的凭证，您必须创建两个 **Secret** ：
  - 带有备份位置的自定义名称的 **secret**。您可以将此 **Secret** 添加到 **DataProtectionApplication** CR 中。
  - 带有默认名称 **cloud-credentials** 的 **secret**，用于快照位置。这个 **Secret** 没有在 **DataProtectionApplication** CR 中被引用。



注意

如果您不想在安装过程中指定备份或快照位置，您可以使用空 **credentials-velero** 文件创建默认 **Secret**。如果没有默认 **Secret**，安装将失败。

流程

1. 点 **Operators → Installed Operators** 并选择 OADP Operator。
2. 在 **Provided APIs** 下，点 **DataProtectionApplication** 框中的 **Create 实例**。
3. 点 **YAML View** 并更新 **DataProtectionApplication** 清单的参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
```

```
configuration:
  velero:
    defaultPlugins:
      - aws
      - openshift 1
  restic:
    enable: true 2
backupLocations:
  - velero:
      config:
        profile: "default"
        region: minio
        s3Url: <url> 3
        insecureSkipTLSVerify: "true"
        s3ForcePathStyle: "true"
      provider: aws
      default: true
      credential:
        key: cloud
        name: cloud-credentials 4
      objectStorage:
        bucket: <bucket_name> 5
        prefix: <prefix> 6
```

1. **openshift** 插件是必须的，才能在 OpenShift Container Platform 集群上备份和恢复命名空间。
2. 如果要禁用 Restic 安装，设置为 **false**。Restic 部署一个守护进程集，这意味着每个 worker 节点都运行有 **Restic** pod。您可以通过在 **Backup** CR 中添加 **spec.defaultVolumesToRestic: true** 来配置 Restic 以进行备份。
3. 指定 S3 端点的 URL。
4. 如果没有指定这个值，则使用默认值 **cloud-credentials**。如果您指定了自定义名称，则使用自定义名称进行备份位置。
5. 指定存储桶作为备份存储位置。如果存储桶不是 Velero 备份的专用存储桶，您必须指定一个前缀。
6. 如果存储桶用于多个目的，请为 Velero 备份指定一个前缀，如 **velero**。

4. 点 **Create**。
5. 通过查看 OADP 资源来验证安装：

```
$ oc get all -n openshift-adp
```

输出示例

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                     1/1     Running   0          94s
pod/restic-m4lts                                     1/1     Running   0          94s
```

```
pod/restic-pv4kr                                  1/1     Running   0          95s
pod/velero-588db7f655-n842v                       1/1     Running   0          95s

NAME                                                TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager   1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry  1/1     1            1           96s
deployment.apps/velero                             1/1     1            1           96s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47   1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd  1         1         1       96s
replicaset.apps/velero-588db7f655                             1         1         1       96s
```

4.2.5.5.1. 在 DataProtectionApplication CR 中启用 CSI

您可以在 **DataProtectionApplication** 自定义资源(CR)中启用 Container Storage Interface(CSI)来备份持久性卷，以使用 CSI 快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。

流程

- 编辑 **DataProtectionApplication** CR，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) 添加 **csi** 默认插件。

(2) 添加 **EnableCSI** 功能标志。

4.2.6. 为 OpenShift Container Storage 的数据保护安装和配置 OpenShift API

您可以通过安装 OADP Operator 并配置备份位置和快照位置，通过安装 OpenShift Container Storage(OCS)安装 OpenShift API for Data Protection(OADP)。然后，您要安装数据保护应用程序。

您可以在 **DataProtectionApplication** 自定义资源(CR)中将 Multicloud Object Gateway 或任何 S3 兼容对象存储配置为备份位置。

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

如果云供应商具有原生快照 API，您可以在 **DataProtectionApplication** CR 中将云存储配置为快照位置。您不需要为 Restic 或 Container Storage Interface(CSI)快照指定快照。

要在受限网络环境中安装 OADP Operator，您必须首先禁用默认的 OperatorHub 源并镜像 Operator 目录。详情请参阅在受限网络中使用 Operator Lifecycle Manager。

4.2.6.1. 安装 OADP Operator

您可以使用 Operator Lifecycle Manager(OLM)在 OpenShift Container Platform 4.8 上安装 Data Protection(OADP)Operator 的 OpenShift API。

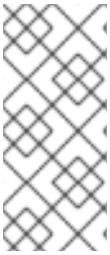
OADP Operator 会安装 Velero 1.7。

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 Web 控制台中，点击 **Operators → OperatorHub**。
2. 使用 **Filter by keyword** 字段查找 **OADP Operator**。
3. 选择 **OADP Operator** 并点 **Install**。
4. 点 **Install** 在 **openshift-adp** 项目中安装 Operator。
5. 点 **Operators → Installed Operators** 来验证安装。



注意

安装 OADP Operator 后，如果云供应商支持原生快照 API，将对象存储配置为备份位置，云存储为快照位置。

如果云供应商不支持快照，或者存储是 NFS，您可以使用 Restic 创建备份。Restic 不需要快照位置。

4.2.6.2. 创建用于备份和快照位置的 secret

如果使用相同的凭证，您可以为备份和快照位置创建一个 **Secret** 对象。

Secret 的默认名称为 **cloud-credentials**，除非您为备份存储供应商指定了默认插件。

先决条件

- 对象存储和云存储必须使用相同的凭证。
- 您必须为 Velero 配置对象存储。
- 您必须以适当的格式为对象存储创建一个 **credentials-velero** 文件。

流程

- 使用默认名称创建 **Secret** ：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

在安装 Data Protection Application 时，**secret** 会在 **DataProtectionApplication** CR 的 **spec.backupLocations.credential** 块中引用。

4.2.6.2.1. 为不同的备份和恢复位置凭证配置 secret

如果您的备份和恢复位置使用不同的凭证，请创建两个 **Secret** 对象：

- 具有自定义名称的备份位置 **Secret**。自定义名称在 **DataProtectionApplication** 自定义资源(CR)的 **spec.backupLocations** 块中指定。
- 带有默认名称 **cloud-credentials** 的快照位置 **Secret**。此 **Secret** 不在 **DataProtectionApplication** CR 中指定。

流程

1. 为您的云供应商为快照位置创建一个 **credentials-velero** 文件。
2. 使用默认名称为快照位置创建 **Secret** ：

```
$ oc create secret generic cloud-credentials -n openshift-adp --from-file cloud=credentials-velero
```

3. 为您的对象存储创建一个用于备份位置的 **credentials-velero** 文件。
4. 使用自定义名称为备份位置创建 **Secret** ：

```
$ oc create secret generic <custom_secret> -n openshift-adp --from-file cloud=credentials-velero
```

5. 将带有自定义名称的 **Secret** 添加到 **DataProtectionApplication** CR 中，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
```

```
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - csi
        - openshift
    featureFlags:
    - EnableCSI
    restic:
      enable: true
  backupLocations:
    - velero:
        provider: gcp
        default: true
        credential:
          key: cloud
          name: <custom_secret> 1
        objectStorage:
          bucket: <bucket_name>
          prefix: <prefix>
```

1 具有自定义名称的备份位置 **Secret**。

4.2.6.3. 配置数据保护应用程序

您可以配置 Velero 资源分配并启用自签名 CA 证书。

4.2.6.3.1. 设置 Velero CPU 和内存分配

您可以通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为 **Velero** pod 设置 CPU 和内存分配。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.configuration.velero.podConfig.ResourceAllocations** 块中的值，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  configuration:
    velero:
      podConfig:
        resourceAllocations:
```

```
      limits:
        cpu: "1" 1
        memory: 512Mi 2
      requests:
        cpu: 500m 3
        memory: 256Mi 4
```

1. 在 millicpus 或 CPU 单元中指定值。默认值为 **500m** 或 **1** CPU 单元。
2. 默认值为 **512Mi**。
3. 默认值为 **500m** 或 **1** CPU 单元。
4. 默认值为 **256Mi**。

4.2.6.3.2. 启用自签名 CA 证书

您必须通过编辑 **DataProtectionApplication** 自定义资源(CR)清单来为对象存储启用自签名 CA 证书，以防止由未知颁发机构签名的证书。

先决条件

- 您必须安装了 OpenShift API for Data Protection(OADP)Operator。

流程

- 编辑 **DataProtectionApplication** CR 清单的 **spec.backupLocations.velero.objectStorage.caCert** 参数和 **spec.backupLocations.velero.config** 参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
spec:
...
  backupLocations:
    - name: default
      velero:
        provider: aws
        default: true
        objectStorage:
          bucket: <bucket>
          prefix: <prefix>
          caCert: <base64_encoded_cert_string> 1
        config:
          insecureSkipTLSVerify: "false" 2
...
```

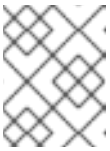
1. 指定 Base46 编码的 CA 证书字符串。
2. 必须为 **false** 来禁用 SSL/TLS 安全性。

4.2.6.4. 安装数据保护应用程序

您可以通过创建 **DataProtectionApplication** API 的实例来安装数据保护应用程序(DPA)。

先决条件

- 您必须安装 OADP Operator。
- 您必须将对象存储配置为备份位置。
- 如果使用快照来备份 PV，云供应商必须支持原生快照 API 或 Container Storage Interface(CSI)快照。
- 如果备份和快照位置使用相同的凭证，您必须创建带有默认名称 **cloud-credentials** 的 **Secret**。
- 如果备份和快照位置使用不同的凭证，您必须创建两个 **Secret** ：
  - 带有备份位置的自定义名称的 **secret**。您可以将此 **Secret** 添加到 **DataProtectionApplication** CR 中。
  - 带有默认名称 **cloud-credentials** 的 **secret**，用于快照位置。这个 **Secret** 没有在 **DataProtectionApplication** CR 中被引用。



注意

如果您不想在安装过程中指定备份或快照位置，您可以使用空 **credentials-velero** 文件创建默认 **Secret**。如果没有默认 **Secret**，安装将失败。

流程

1. 点 **Operators → Installed Operators** 并选择 OADP Operator。
2. 在 **Provided APIs** 下，点 **DataProtectionApplication** 框中的 **Create 实例**。
3. 点 **YAML View** 并更新 **DataProtectionApplication** 清单的参数：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: <dpa_sample>
  namespace: openshift-adp
spec:
  configuration:
    velero:
      defaultPlugins:
        - gcp <.>
        - csi <.>
        - openshift 1
    restic:
      enable: true 2
  backupLocations:
    - velero:
        provider: gcp 3
        default: true
        credential:
          key: cloud
```

```
      name: <default_secret> 4
    objectStorage:
      bucket: <bucket_name> 5
      prefix: <prefix> 6
```

1. 为备份供应商指定默认插件，如 **gcp** （如果适用）。
2. 如果使用 CSI 快照备份 PV，请指定 **csi** 默认插件。**csi** 插件使用 Velero CSI beta 快照 API。您不需要配置快照位置。
3. **openshift** 插件是必须的，才能在 OpenShift Container Platform 集群上备份和恢复命名空间。
4. 如果要禁用 Restic 安装，设置为 **false**。Restic 部署一个守护进程集，这意味着每个 worker 节点都运行有 **Restic** pod。您可以通过在 **Backup** CR 中添加 **spec.defaultVolumesToRestic: true** 来配置 Restic 以进行备份。
5. 指定备份供应商。
6. 如果将默认插件用于备份提供程序，您必须为 **Secret** 指定正确的默认名称，例如 **cloud-credentials-gcp**。如果您指定了自定义名称，则使用自定义名称进行备份位置。如果没有指定 **Secret** 名称，则使用默认名称。

   指定存储桶作为备份存储位置。如果存储桶不是 Velero 备份的专用存储桶，您必须指定一个前缀。

   如果存储桶用于多个目的，请为 Velero 备份指定一个前缀，如 **velero**。

4. 点 **Create**。
5. 通过查看 OADP 资源来验证安装：

```
$ oc get all -n openshift-adp
```

输出示例

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/oadp-operator-controller-manager-67d9494d47-6l8z8   2/2     Running   0          2m8s
pod/oadp-velero-sample-1-aws-registry-5d6968cbdd-d5w9k   1/1     Running   0          95s
pod/restic-9cq4q                                     1/1     Running   0          94s
pod/restic-m4lts                                     1/1     Running   0          94s
pod/restic-pv4kr                                     1/1     Running   0          95s
pod/velero-588db7f655-n842v                          1/1     Running   0          95s

NAME                                                   TYPE        CLUSTER-IP       EXTERNAL-IP
PORT(S)    AGE
service/oadp-operator-controller-manager-metrics-service   ClusterIP   172.30.70.140
<none>        8443/TCP   2m8s
service/oadp-velero-sample-1-aws-registry-svc              ClusterIP   172.30.130.230   <none>
5000/TCP   95s

NAME                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
daemonset.apps/restic   3         3         3       3            3           <none>          96s
```

```
NAME                                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/oadp-operator-controller-manager     1/1     1            1           2m9s
deployment.apps/oadp-velero-sample-1-aws-registry    1/1     1            1           96s
deployment.apps/velero                               1/1     1            1           96s

NAME                                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/oadp-operator-controller-manager-67d9494d47     1         1         1       2m9s
replicaset.apps/oadp-velero-sample-1-aws-registry-5d6968cbdd    1         1         1       96s
replicaset.apps/velero-588db7f655                               1         1         1       96s
```

4.2.6.4.1. 在 DataProtectionApplication CR 中启用 CSI

您可以在 **DataProtectionApplication** 自定义资源(CR)中启用 Container Storage Interface(CSI)来备份持久性卷，以使用 CSI 快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。

流程

- 编辑 **DataProtectionApplication** CR，如下例所示：

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
...
spec:
  configuration:
    velero:
      defaultPlugins:
      - openshift
      - csi (1)
    featureFlags:
    - EnableCSI (2)
```

(1) 添加 **csi** 默认插件。

(2) 添加 **EnableCSI** 功能标志。

4.2.7. 为数据保护卸载 OpenShift API

您可以通过删除 OADP Operator 来卸载 OpenShift API for Data Protection(OADP)。详情请参阅从集群中删除 Operator。

4.3. 备份和恢复

4.3.1. 备份应用程序

您可以通过创建 **Backup** 自定义资源(CR) 来备份应用程序。

Backup CR 为 Kubernetes 资源和内部镜像（S3 对象存储）和持久性卷(PV)创建备份文件，如果云供应商使用原生快照 API 或 Container Storage Interface(CSI) 来创建快照，如 OpenShift Container Storage 4。如需更多信息，请参阅 CSI 卷快照。

重要

S3 存储的 **CloudStorage** API 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能支持范围的详情，请参考 https://access.redhat.com/support/offerings/techpreview/。

如果您的云供应商具有原生快照 API 或支持 Container Storage Interface(CSI)快照，则 **Backup** CR 通过创建快照来备份持久性卷。如需更多信息，请参阅 OpenShift Container Platform 文档中的 CSI 卷快照概述。

如果您的云供应商不支持快照，或者应用程序位于 NFS 数据卷中，您可以使用 Restic 创建备份。

您可以创建 backup hook，以便在备份操作之前或之后运行命令。

您可以通过创建一个 **Schedule** CR 而不是 **Backup** CR 来调度备份。

4.3.1.1. 创建备份 CR

您可以通过创建 **Backup** 备份自定义资源(CR)来备份 Kubernetes 镜像、内部镜像和持久性卷(PV)。

先决条件

- 您必须安装用于数据保护(OADP)Operator 的 OpenShift API。
- **DataProtectionApplication** CR 必须处于 **Ready** 状态。
- 备份位置先决条件：
  - 您必须为 Velero 配置 S3 对象存储。
  - 您必须在 **DataProtectionApplication** CR 中配置了一个备份位置。
- 快照位置先决条件：
  - 您的云供应商必须具有原生快照 API 或支持 Container Storage Interface(CSI)快照。
  - 对于 CSI 快照，您必须创建一个 **VolumeSnapshotClass** CR 来注册 CSI 驱动程序。
  - 您必须在 **DataProtectionApplication** CR 中配置了一个卷位置。

流程

1. 检索 **backupStorageLocations** CR：

```
$ oc get backupStorageLocations
```

输出示例

```
NAME            PHASE      LAST VALIDATED   AGE   DEFAULT
velero-sample-1   Available   11s            31m
```

2. 创建一个 **Backup** CR，如下例所示：

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  labels:
    velero.io/storage-location: default
  namespace: openshift-adp
spec:
  hooks: {}
  includedNamespaces:
  - <namespace> 1
  storageLocation: <velero-sample-1> 2
  ttl: 720h0m0s
```

1 指定要备份的命名空间数组。

2 指定 **backupStorageLocations** CR 的名称。

3. 验证 **Backup** CR 的状态是否为 **Completed** ：

```
$ oc get backup -n openshift-adp <backup> -o jsonpath='{.status.phase}'
```

4.3.1.2. 使用 CSI 快照备份持久性卷

在创建 **Backup** CR 前，您可以通过创建 **VolumeSnapshotClass** 自定义资源(CR)来注册 CSI 驱动程序，使用 Container Storage Interface(CSI)快照备份持久性卷。

先决条件

- 云供应商必须支持 CSI 快照。
- 您必须在 **DataProtectionApplication** CR 中启用 CSI。

流程

- 创建 **VolumeSnapshotClass** CR，如下例所示：

  Ceph RBD

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
deletionPolicy: Retain
metadata:
  name: <volume_snapshot_class_name>
  labels:
    velero.io/csi-volumesnapshot-class: "true"
    snapshotter: openshift-storage.rbd.csi.ceph.com
driver: openshift-storage.rbd.csi.ceph.com
```

```
  parameters:
    clusterID: openshift-storage
    csi.storage.k8s.io/snapshotter-secret-name: rook-csi-rbd-provisioner
    csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
```

Ceph FS

```
  apiVersion: snapshot.storage.k8s.io/v1
  kind: VolumeSnapshotClass
  metadata:
    name: <volume_snapshot_class_name>
    labels:
      velero.io/csi-volumesnapshot-class: "true"
  driver: openshift-storage.cephfs.csi.ceph.com
  deletionPolicy: Retain
  parameters:
    clusterID: openshift-storage
    csi.storage.k8s.io/snapshotter-secret-name: rook-csi-cephfs-provisioner
    csi.storage.k8s.io/snapshotter-secret-namespace: openshift-storage
```

其他云供应商

```
  apiVersion: snapshot.storage.k8s.io/v1
  kind: VolumeSnapshotClass
  metadata:
    name: <volume_snapshot_class_name>
    labels:
      velero.io/csi-volumesnapshot-class: "true"
  driver: <csi_driver>
  deletionPolicy: Retain
```

现在，您可以创建一个 **Backup** CR。

4.3.1.3. 使用 Restic 备份应用程序

您可以通过编辑备份自定义资源(CR)来使用 Restic **Backup** 资源、内部镜像和持久性卷备份 Kubernetes 资源。

您不需要在 **DataProtectionApplication** CR 中指定快照位置。

先决条件

- 您必须安装用于数据保护(OADP)Operator 的 OpenShift API。
- 您不能将 **DataProtectionApplication** CR 中的 **spec.configuration.restic.enable** 设置为 **false** 来禁用默认的 Restic 安装。
- **DataProtectionApplication** CR 必须处于 **Ready** 状态。

流程

- 编辑 **Backup** CR，如下例所示：

```
  apiVersion: velero.io/v1
```

```
kind: Backup
metadata:
  name: <backup>
  labels:
    velero.io/storage-location: default
  namespace: openshift-adp
spec:
  defaultVolumesToRestic: true (1)
...
```

(1) 将 **defaultVolumesToRestic: true** 添加到 **spec** 块中。

4.3.1.4. 创建备份 hook

您可以通过编辑备份自定义资源(CR)来创建 **Backup** hook 以在 pod 中运行的容器中运行命令。

在 pod 备份前运行 *Pre* hook。在备份后运行 *Post* hook。

流程

- 在 **Backup** CR 的 **spec.hooks** 块中添加 hook，如下例所示：

```
apiVersion: velero.io/v1
kind: Backup
metadata:
  name: <backup>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
        - <namespace> (1)
        excludedNamespaces:
        - <namespace>
        includedResources:
        - pods (2)
        excludedResources: []
        labelSelector: (3)
          matchLabels:
            app: velero
            component: server
        pre: (4)
          - exec:
              container: <container> (5)
              command:
              - /bin/uname (6)
              - -a
              onError: Fail (7)
              timeout: 30s (8)
        post: (9)
...
```

1. hook 应用的命名空间数组。如果没有指定这个值，则 hook 适用于所有命名空间。
2. 目前，pod 是唯一支持的资源。
3. 可选：此 hook 仅适用于与标签选择器匹配的对象。
4. 备份前要运行的 hook 数组。
5. 可选：如果没有指定容器，该命令将在 pod 的第一个容器中运行。
6. hook 运行的命令数组。
7. 错误处理允许的值是 **Fail** 和 **Continue**。默认值为 **Fail**。
8. 可选：等待命令运行的时间。默认值为 **30s**。
9. 此块定义了在备份后运行的一组 hook，其参数与 pre-backup hook 相同。

4.3.1.5. 调度备份

您可以通过创建 **Schedule** 自定义资源(CR)而不是 **Backup** CR 来调度备份。

警告

在您的备份调度中留有足够的时间，以便在创建另一个备份前完成了当前的备份。

例如，如果对一个命名空间进行备份通常需要 10 分钟才能完成，则调度的备份频率不应该超过每 15 分钟一次。

先决条件

- 您必须安装用于数据保护(OADP)Operator 的 OpenShift API。
- **DataProtectionApplication** CR 必须处于 **Ready** 状态。

流程

1. 检索 **backupStorageLocations** CR：

```
$ oc get backupStorageLocations
```

输出示例

```
NAME              PHASE       LAST VALIDATED   AGE   DEFAULT
velero-sample-1   Available   11s              31m
```

2. 创建一个 **Schedule** CR，如下例所示：

```
$ cat << EOF | oc apply -f -
```

```
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: <schedule>
  namespace: openshift-adp
spec:
  schedule: 0 7 * * * 1
  template:
    hooks: {}
    includedNamespaces:
    - <namespace> 2
    storageLocation: <velero-sample-1> 3
    defaultVolumesToRestic: true 4
    ttl: 720h0m0s
EOF
```

1. 调度备份的 **cron** 表达式，例如 **0 7 * * *** 代表在每天 7:00 执行备份。
2. 要备份的命名空间数组。
3. **backupStorageLocations** CR 的名称。
4. 可选：如果使用 Restic 备份卷，请添加 **defaultVolumesToRestic: true** 键-值对。

3. 在调度的备份运行后验证 **Schedule** CR 的状态是否为 **Completed** ：

```
$ oc get schedule -n openshift-adp <schedule> -o jsonpath='{.status.phase}'
```

4.3.2. 恢复应用程序

您可以通过创建 **Restore** 自定义资源(CR) 来恢复应用程序备份。

您可以创建 restore hooks，以在应用程序容器启动前或在应用程序容器本身中运行命令。

4.3.2.1. 创建恢复 CR

您可以通过创建一个 **Restore** CR 来恢复 **Backup** 自定义资源(CR)。

先决条件

- 您必须安装用于数据保护(OADP)Operator 的 OpenShift API。
- **DataProtectionApplication** CR 必须处于 **Ready** 状态。
- 您必须具有 Velero **Backup** CR。
- 调整请求的大小，以便持久性卷 (PV) 容量与备份时请求的大小匹配。

流程

1. 创建一个 **Restore** CR，如下例所示：

```
apiVersion: velero.io/v1
```

```
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  backupName: <backup> 1
  excludedResources:
  - nodes
  - events
  - events.events.k8s.io
  - backups.velero.io
  - restores.velero.io
  - resticrepositories.velero.io
  restorePVs: true
```

1 **备份 CR 的名称。**

2. 验证 **Restore** CR 的状态是否为 **Completed** ：

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

3. 验证备份资源是否已恢复：

```
$ oc get all -n <namespace> 1
```

1 备份的命名空间。

4.3.2.2. 创建恢复 hook

您可以创建恢复 hook，以便在 pod 中运行的容器运行命令，同时通过编辑 **Restore** 自定义资源(CR)恢复应用程序。

您可以创建两种类型的恢复 hook：

- **init** hook 将 init 容器添加到 pod，以便在应用程序容器启动前执行设置任务。
  如果您恢复 Restic 备份，则会在恢复 hook init 容器前添加 **restic-wait** init 容器。
- **exec** hook 在恢复的 pod 的容器中运行命令或脚本。

流程

- 在 **Restore** CR 的 **spec.hooks** 块中添加 hook，如下例所示：

```
apiVersion: velero.io/v1
kind: Restore
metadata:
  name: <restore>
  namespace: openshift-adp
spec:
  hooks:
    resources:
      - name: <hook_name>
        includedNamespaces:
```

```
    - <namespace> 1
    excludedNamespaces:
    - <namespace>
    includedResources:
    - pods 2
    excludedResources: []
    labelSelector: 3
      matchLabels:
        app: velero
        component: server
    postHooks:
    - init:
        initContainers:
        - name: restore-hook-init
          image: alpine:latest
          volumeMounts:
          - mountPath: /restores/pvc1-vm
            name: pvc1-vm
          command:
          - /bin/ash
          - -c
    - exec:
        container: <container> 4
        command:
        - /bin/bash 5
        - -c
        - "psql < /backup/backup.sql"
        waitTimeout: 5m 6
        execTimeout: 1m 7
        onError: Continue 8
```

1. 可选： hook 应用的命名空间数组。如果没有指定这个值，则 hook 适用于所有命名空间。
2. 目前，pod 是唯一支持的资源。
3. 可选：此 hook 仅适用于与标签选择器匹配的对象。
4. 可选：如果没有指定容器，该命令将在 pod 的第一个容器中运行。
5. hook 运行的命令数组。
6. 可选：如果没有指定 **waitTimeout**，则恢复会无限期等待。您可以指定等待容器启动以及容器中前面的 hook 完成的时间。当容器恢复时，等待超时会启动，可能需要时间让容器拉取并挂载卷。
7. 可选：等待命令运行的时间。默认值为 **30s**。
8. 错误处理的允许值为 **Fail** 和 **Continue**：
   - **Continue**: 只记录命令失败。
   - **Fail**: 任何 pod 中的任何容器中没有更多恢复 hook 运行。 **Restore** CR 的状态将是 **PartiallyFailed**。

4.4. 故障排除

您可以使用 OpenShift CLI 工具或 Velero CLI 工具调试 Velero 自定义资源(CR)。Velero CLI 工具提供更详细的日志和信息。

您可以检查 安装问题、备份和恢复 CR 问题，以及 Restic 问题。

您可以使用 **must-gather** 工具来收集日志、CR 信息和 Prometheus 指标数据。

您可以通过以下方法获取 Velero CLI 工具：

- 下载 Velero CLI 工具
- 访问集群中的 Velero 部署中的 Velero 二进制文件

4.4.1. 下载 Velero CLI 工具

您可以按照 Velero 文档页面中的说明下载并安装 Velero CLI 工具。

该页面包括：

- 使用 Homebrew 的 macOS
- GitHub
- 使用 Chocolatey 的 Windows

先决条件

- 您可以访问启用了 DNS 和容器网络的 Kubernetes 集群 v1.16 或更高版本。
- 您已在本地安装了 **kubectl**。

流程

1. 打开浏览器，进入到 Verleo 网站上的"安装 CLI"。
2. 按照 macOS、GitHub 或 Windows 的适当流程。
3. 根据以下表，下载适用于 OADP 版本的 Velero 版本：

表 4.2. OADP-Velero 版本关系

OADP 版本	Velero 版本
0.2.6	1.6.0
0.5.5	1.7.1
1.0.0	1.7.1
1.0.1	1.7.1
1.0.2	1.7.1

OADP 版本	Velero 版本
1.0.3	1.7.1

4.4.2. 访问集群中的 Velero 部署中的 Velero 二进制文件

您可以使用 shell 命令访问集群中的 Velero 部署中的 Velero 二进制文件。

先决条件

- 您的 **DataProtectionApplication** 自定义资源的状态为 **Reconcile complete**。

流程

- 输入以下命令设定所需的别名：

```
$ alias velero='oc -n openshift-adp exec deployment/velero -c velero -it -- ./velero'
```

4.4.3. 使用 OpenShift CLI 工具调试 Velero 资源

您可以使用 OpenShift CLI 工具检查 Velero 自定义资源(CR)和 **Velero** pod 日志来调试失败的备份或恢复。

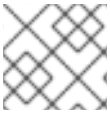
Velero CR
使用 **oc describe** 命令检索与 **Backup** 或 **Restore** CR 关联的警告和错误概述：

```
$ oc describe <velero_cr> <cr_name>
```

Velero pod 日志
使用 **oc logs** 命令检索 **Velero** pod 日志：

```
$ oc logs pod/<velero>
```

Velero pod 调试日志
您可以在 **DataProtectionApplication** 资源中指定 Velero 日志级别，如下例所示。



注意

这个选项可从 OADP 1.0.3 开始。

```
apiVersion: oadp.openshift.io/v1alpha1
kind: DataProtectionApplication
metadata:
  name: velero-sample
spec:
  configuration:
    velero:
      logLevel: warning
```

可用的 **logLevel** 值如下：

- **trace**
- **debug**
- **info**
- **warning**
- 错误
- **fatal**
- **panic**

对于多数日志，建议使用 **debug**。

4.4.4. 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-adp exec deployment/velero -c velero -- ./velero \
  <backup_restore_cr> <command> <cr_name>
```

示例

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

示例

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

示例

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

4.4.5. 安装问题

在安装数据保护应用程序时，您可能会遇到使用无效目录或不正确的凭证导致的问题。

4.4.5.1. 备份存储包含无效目录

Velero pod 日志显示错误消息，**备份存储包含无效的顶级目录**。

原因

对象存储包含不是 Velero 目录的顶级目录。

解决方案

如果对象存储不适用于 Velero，则必须通过设置 **DataProtectionApplication** 清单中的 **spec.backupLocations.velero.objectStorage.prefix** 参数为存储桶指定一个前缀。

4.4.5.2. AWS 凭证不正确

oadp-aws-registry pod 日志会显示错误消息 **InvalidAccessKeyId: The AWS Access Key Id you provided does not exist in our records.**

Velero pod 日志显示错误消息 **NoCredentialProviders: no valid provider in chain**。

原因

用于创建 **Secret** 对象的 **credentials-velero** 文件会错误地格式化。

解决方案

确保 **credentials-velero** 文件已正确格式化，如下例所示：

credentials-velero 文件示例

```
[default] 1
aws_access_key_id=AKIAIOSFODNN7EXAMPLE 2
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

1 AWS 默认配置集。

2 不用使用括号 ('', ") 把值括起来。

4.4.6. 备份和恢复 CR 问题

您可能会遇到 **Backup** 和 **Restore** 自定义资源(CR)的常见问题。

4.4.6.1. 备份 CR 无法检索卷

Backup CR 显示错误消息 **InvalidVolume.NotFound: The volume 'vol-xxxx' does not exist**。

原因

持久性卷(PV)和快照位置位于不同的区域。

解决方案

1. 编辑 **DataProtectionApplication** 清单中的 **spec.snapshotLocations.velero.config.region** 键的值，使快照位置位于与 PV 相同的区域。
2. 创建新的 **Backup** CR。

4.4.6.2. 备份 CR 状态在进行中

Backup CR 的状态保留在 **InProgress** 阶段，且未完成。

原因

如果备份中断，则无法恢复。

解决方案

1. 检索 **Backup** CR 的详细信息：

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

2. 删除 **Backup** CR：

```
$ oc delete backup <backup> -n openshift-adp
```

   您不需要清理备份位置，因为正在进行中的 **Backup** CR 没有上传文件到对象存储。

3. 创建新的 **Backup** CR。

4.4.7. Restic 问题

在使用 Restic 备份应用程序时，您可能会遇到这些问题。

4.4.7.1. 启用了 root_squash 的 NFS 数据卷的 Restic 权限错误

Restic pod 日志显示错误消息, **controller=pod-volume-backup error="fork/exec/usr/bin/restic: permission denied"**。

原因

如果您的 NFS 数据卷启用了 **root_squash**，**Restic** 映射到 **nfsnobody**，且没有创建备份的权限。

解决方案

您可以通过为 **Restic** 创建补充组并将组 ID 添加到 **DataProtectionApplication** 清单中来解决这个问题：

1. 在 NFS 数据卷中为 **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 **spec.configuration.restic.supplementalGroups** 参数和组 ID 添加到 **DataProtectionApplication** 清单中，如下例所示：

```
spec:
  configuration:
    restic:
      enable: true
      supplementalGroups:
      - <group_id> 1
```

1 指定补充组 ID。

4. 等待 **Restic** pod 重启，以便应用更改。

4.4.7.2. 恢复 Restic 备份的 CR 是 "PartiallyFailed", "Failed"，或保留 "InProgress"

Restic 备份的 **Restore** CR 使用 **PartiallyFailed** 或 **Failed** 状态完成，或者保持 **InProgress**，且不完成。

如果状态是 **PartiallyFailed** 或 **Failed**，**Velero** pod 日志会显示错误消息，**level=error msg="unable to successfully restic restore of pod 的 volumes"**。

如果状态是 **InProgress**，**Restore** CR 日志不可用，且 **Restic** pod 日志中不会出现任何错误。

原因

DeploymentConfig 对象重新部署 **Restore** pod，从而导致 **Restore** CR 失败。

解决方案

1. 创建一个 **Restore** CR，它排除 **ReplicationController**、**DeploymentConfig** 和 **TemplateInstances** 资源：

```
$ velero restore create --from-backup=<backup> -n openshift-adp \ 1
  --include-namespaces <namespace> \ 2
  --exclude-resources
replicationcontroller,deploymentconfig,templateinstances.template.openshift.io \
  --restore-volumes=true
```

1 指定 **Backup** CR 的名称。

2 在 **Backup** CR 中指定 **include-namespaces**。

2. 验证 **Restore** CR 的状态是否为 **Completed** ：

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

3. 创建一个 **Restore** CR，以包括 **ReplicationController** 和 **DeploymentConfig** 资源：

```
$ velero restore create --from-backup=<backup> -n openshift-adp \
  --include-namespaces <namespace> \
  --include-resources replicationcontroller,deploymentconfig \
```

```
--restore-volumes=true
```

4. 验证 **Restore** CR 的状态是否为 **Completed** ：

```
$ oc get restore -n openshift-adp <restore> -o jsonpath='{.status.phase}'
```

5. 验证备份资源是否已恢复：

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

4.4.7.3. 在存储桶被强制后重新创建 Restic Backup CR

如果您为命名空间创建 Restic **Backup** CR，请清空 S3 存储桶，然后为同一命名空间重新创建 **Backup** CR，重新创建的 **Backup** CR 会失败。

velero pod 日志显示错误消息 **msg="Error check repository for stale locks"**。

原因

如果在对象存储上删除了 Restic 目录，Velero 不会在 **ResticRepository** 清单中创建 Restic 存储库。详情请参阅(Velero issue 4421)。

4.4.8. 使用 must-gather 工具

您可以使用 **must-gather** 工具收集有关 OADP 自定义资源的日志、指标和信息。

must-gather 数据必须附加到所有客户案例。

您可以使用以下数据收集选项运行 **must-gather** 工具：

- 完全 **must-gather** 数据收集为安装 OADP Operator 的所有命名空间收集 Prometheus metrics、pod 日志和 Velero CR 信息。
- 基本的 **must-gather** 数据收集在特定持续时间内收集 pod 日志和 Velero CR 信息，例如一小时或 24 小时。Prometheus 指标和重复日志不包含在内。
- 使用超时的 **must-gather** 数据收集。如果有许多 **Backup** CR 失败，则数据收集需要很长时间。您可以通过设置超时值来提高性能。
- Prometheus 指标数据转储下载包含 Prometheus 收集的数据的存档文件。

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：
    - 完整的 **must-gather** 数据收集，包括 Prometheus 指标：

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0
```

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。

- 特定持续时间内，基本 **must-gather** 数据收集功能不进行 Prometheus 指标：

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
  -- /usr/bin/gather_<time>_essential 1
```

1 以小时为单位指定时间。允许的值是 **1h**、**6h**、**24h**、**72h** 或 **all**，例如 **gather_1h_essential** 或 **gather_all_essential**。

- 使用超时的 **must-gather** 数据收集：

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
  -- /usr/bin/gather_with_timeout <timeout> 1
```

1 以秒为单位指定超时值。

- Prometheus 指标数据转储：

```
$ oc adm must-gather --image=registry.redhat.io/oadp/oadp-mustgather-rhel8:v1.0 \
  -- /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
```

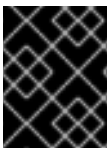
第 5 章 CONTROL PLANE 备份和恢复

5.1. 备份 ETCD

etcd 是 OpenShift Container Platform 的以"键-值"形式进行的存储，它会保留所有资源对象的状态。

定期备份集群的 etcd 数据，并在 OpenShift Container Platform 环境以外的安全位置保存备份数据。不要在第一个证书轮转完成前（安装后的 24 小时内）进行 etcd 备份，否则备份将包含过期的证书。另外，建议您在非高峰期使用 etcd 备份，因为 etcd 快照有较高的 I/O 成本。

确保升级集群后执行 etcd 备份。这很重要，因为当恢复集群时，必须使用从同一 z-stream 发行版本中获取的 etcd 备份。例如，OpenShift Container Platform 4.y.z 集群必须使用从 4.y.z 中获得的 etcd 备份。



重要

通过在 control plane 主机上（也称为 master 主机）执行一次备份脚本来备份集群的 etcd 数据。不要为每个 control plane 主机进行备份。

在进行了 etcd 备份后，就可以恢复到一个以前的集群状态。

5.1.1. 备份 etcd 数据

按照以下步骤，通过创建 etcd 快照并备份静态 pod 的资源来备份 etcd 数据。这个备份可以被保存，并在以后需要时使用它来恢复 etcd 数据。

重要

仅保存单一 control plane 主机（也称为 master 主机）的备份。不要从集群中的每个 control plane 主机进行备份。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已检查是否启用了集群范围代理。

提示

您可以通过查看 **oc get proxy cluster -o yaml** 的输出来检查代理是否已启用。如果 **httpProxy**、**httpsProxy**和 **noProxy** 字段设置了值，则会启用代理。

流程

1. 为 control plane 节点启动一个 debug 会话：

```
$ oc debug node/<node_name>
```

2. 将您的根目录改为 **/host** ：

```
sh-4.2# chroot /host
```

3. 如果启用了集群范围的代理，请确定已导出了 **NO_PROXY**、**HTTP_PROXY**和 **HTTPS_PROXY** 环境变量。

4. 运行 **cluster-backup.sh** 脚本，输入保存备份的位置。

提示

cluster-backup.sh 脚本作为 etcd Cluster Operator 的一个组件被维护，它是 **etcdctl snapshot save** 命令的包装程序（wrapper）。

```
sh-4.4# /usr/local/bin/cluster-backup.sh /home/core/assets/backup
```

脚本输出示例

```
found latest kube-apiserver: /etc/kubernetes/static-pod-resources/kube-apiserver-pod-6
found latest kube-controller-manager: /etc/kubernetes/static-pod-resources/kube-controller-manager-pod-7
found latest kube-scheduler: /etc/kubernetes/static-pod-resources/kube-scheduler-pod-6
found latest etcd: /etc/kubernetes/static-pod-resources/etcd-pod-3
ede95fe6b88b87ba86a03c15e669fb4aa5bf0991c180d3c6895ce72eaade54a1
etcdctl version: 3.4.14
API version: 3.4
{"level":"info","ts":1624647639.0188997,"caller":"snapshot/v3_snapshot.go:119","msg":"created temporary db file","path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db.part"}
{"level":"info","ts":"2021-06-25T19:00:39.030Z","caller":"clientv3/maintenance.go:200","msg":"opened snapshot stream; downloading"}
{"level":"info","ts":1624647639.0301006,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching snapshot","endpoint":"https://10.0.0.5:2379"}
{"level":"info","ts":"2021-06-25T19:00:40.215Z","caller":"clientv3/maintenance.go:208","msg":"completed snapshot read; closing"}
{"level":"info","ts":1624647640.6032252,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched snapshot","endpoint":"https://10.0.0.5:2379","size":"114 MB","took":1.584090459}
{"level":"info","ts":1624647640.6047094,"caller":"snapshot/v3_snapshot.go:152","msg":"saved","path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db"}
Snapshot saved at /home/core/assets/backup/snapshot_2021-06-25_190035.db
{"hash":3866667823,"revision":31407,"totalKey":12828,"totalSize":114446336}
snapshot db and kube resources are successfully saved to /home/core/assets/backup
```

在这个示例中，在 control plane 主机上的 **/home/core/assets/backup/** 目录中创建了两个文件：

- **snapshot_<datetimestamp>.db**：这个文件是 etcd 快照。**cluster-backup.sh** 脚本确认其有效。
- **static_kuberesources_<datetimestamp>.tar.gz**：此文件包含静态 pod 的资源。如果启用了 etcd 加密，它也包含 etcd 快照的加密密钥。

注意

如果启用了 etcd 加密，建议出于安全考虑，将第二个文件与 etcd 快照分开保存。但是，需要这个文件才能从 etcd 快照中进行恢复。

请记住，etcd 仅对值进行加密，而不对键进行加密。这意味着资源类型、命名空间和对象名称是不加密的。

5.2. 替换不健康的 ETCD 成员

本文档描述了替换一个不健康 etcd 成员的过程。

此过程取决于 etcd 成员不健康的原因，如机器没有运行，或节点未就绪，或 etcd pod 处于 crashlooping 状态。

注意

如果您丢失了大多数 control plane 主机（也被称为 master 主机），并导致 etcd 仲裁丢失，那么您必须遵循灾难恢复流程恢复到集群原来的状态，而不是这个过程。

如果 control plane 证书在被替换的成员中无效，则必须遵循从已过期 control plane 证书中恢复的步骤，而不是此过程。

如果 control plane 节点丢失并且创建了一个新节点，etcd 集群 Operator 将处理生成新 TLS 证书并将节点添加为 etcd 成员。

5.2.1. 先决条件

- 在替换不健康的 etcd 成员，需要进行 etcd 备份。

5.2.2. 找出一个不健康的 etcd 成员

您可以识别集群是否有不健康的 etcd 成员。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 使用以下命令检查 **EtcdMembersAvailable** 状态条件的状态：

```
$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="EtcdMembersAvailable")]}{.message}{"\n"}'
```

2. 查看输出：

```
2 of 3 members are available, ip-10-0-131-183.ec2.internal is unhealthy
```

这个示例输出显示 **ip-10-0-131-183.ec2.internal** etcd 成员不健康。

5.2.3. 确定不健康的 etcd 成员的状态

替换不健康 etcd 成员的步骤取决于 etcd 的以下状态：

- 机器没有运行或者该节点未就绪
- etcd pod 处于 crashlooping 状态

此流程决定了 etcd 成员处于哪个状态。这可让您了解替换不健康的 etcd 成员要遵循的步骤。

注意

如果您知道机器没有运行或节点未就绪，但它们应该很快返回健康状态，那么您就不需要执行替换 etcd 成员的流程。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已找到不健康的 etcd 成员。

流程

1. 检查 **机器是否没有运行**:

```
$ oc get machines -A -ojsonpath='{range .items[*]}{@.status.nodeRef.name}{"\t"}
{@.status.providerStatus.instanceState}{"\n"}' | grep -v running
```

输出示例

```
ip-10-0-131-183.ec2.internal  stopped (1)
```

(1) 此输出列出了节点以及节点机器的状态。如果状态不是 **running**，则**代表机器没有运行**。

如果**机器没有运行**，按照 *替换机器没有运行或节点没有就绪的非健康 etcd 成员*过程进行操作。

2. 确定 **节点是否未就绪**。
如果以下任何一种情况是正确的，则代表**节点没有就绪**。

- 如果机器正在运行，检查节点是否不可访问：

```
$ oc get nodes -o jsonpath='{range .items[*]}{"\n"}{.metadata.name}{"\t"}{range
.spec.taints[*]}{.key}{" "}' | grep unreachable
```

输出示例

```
ip-10-0-131-183.ec2.internal node-role.kubernetes.io/master
node.kubernetes.io/unreachable node.kubernetes.io/unreachable (1)
```

(1) 如果节点带有 **unreachable** 污点，则**节点没有就绪**。

- 如果该节点仍然可访问，则检查该节点是否列为 **NotReady**:

```
$ oc get nodes -l node-role.kubernetes.io/master | grep "NotReady"
```

输出示例

```
ip-10-0-131-183.ec2.internal   NotReady   master   122m   v1.21.0 (1)
```

(1) 如果节点列表为 **NotReady**，则 该节点没有就绪。

如果节点没有就绪，按照*替换机器没有运行或节点没有就绪的 etcd 成员*的步骤进行操作。

3. 确定 **etcd Pod 是否处于 crashlooping 状态。**
   如果机器正在运行并且节点已就绪，请检查 etcd pod 是否处于 crashlooping 状态。

   a. 验证所有 control plane 节点（也称为 master 节点）是否都列为 **Ready** ：

   ```
   $ oc get nodes -l node-role.kubernetes.io/master
   ```

   输出示例

   ```
   NAME                           STATUS   ROLES    AGE     VERSION
   ip-10-0-131-183.ec2.internal   Ready    master   6h13m   v1.21.0
   ip-10-0-164-97.ec2.internal    Ready    master   6h13m   v1.21.0
   ip-10-0-154-204.ec2.internal   Ready    master   6h13m   v1.21.0
   ```

   b. 检查 etcd pod 的状态是否为 **Error** 或 **CrashLoopBackOff**:

   ```
   $ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
   ```

   输出示例

   ```
   etcd-ip-10-0-131-183.ec2.internal   2/3   Error     7   6h9m (1)
   etcd-ip-10-0-164-97.ec2.internal    3/3   Running   0   6h6m
   etcd-ip-10-0-154-204.ec2.internal   3/3   Running   0   6h6m
   ```

   (1) 由于此 pod 的状态是 **Error**，因此 **etcd pod 为 crashlooping 状态。**

如果 **etcd pod 为 crashlooping 状态，**请按照*替换 etcd pod 处于 crashlooping 状态的不健康的 etcd 成员*的步骤进行操作。

5.2.4. 替换不健康的 etcd 成员

根据不健康的 etcd 成员的状态，使用以下一个流程：

- 替换机器没有运行或节点未就绪的不健康 etcd 成员
- 替换其 etcd Pod 处于 crashlooping 状态的不健康 etcd 成员
- 替换不健康的裸机 etcd 成员

5.2.4.1. 替换机器没有运行或节点未就绪的不健康 etcd 成员

此流程详细介绍了替换因机器没有运行或节点未就绪造成不健康的 etcd 成员的步骤。

先决条件

- 您已找出不健康的 etcd 成员。
- 您已确认机器没有运行，或者该节点未就绪。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。

重要

执行此流程前务必要进行 etcd 备份，以便在遇到任何问题时可以恢复集群。

流程

1. 删除不健康的成员。
   a. 选择一个*不在*受影响节点上的 pod:
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

   输出示例

```
etcd-ip-10-0-131-183.ec2.internal             3/3     Running     0          123m
etcd-ip-10-0-164-97.ec2.internal              3/3     Running     0          123m
etcd-ip-10-0-154-204.ec2.internal             3/3     Running     0          124m
```

   b. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称：
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

   c. 查看成员列表：

```
sh-4.2# etcdctl member list -w table
```

   输出示例

```
+------------------+---------+-----------------------------+---------------------------+---------------
-----------+
|        ID        | STATUS  |             NAME            |        PEER ADDRS         |      CLIENT
ADDRS       |
+------------------+---------+-----------------------------+---------------------------+---------------
-----------+
| 6fc1e7c9db35841d | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 |
https://10.0.131.183:2379 |
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
```

```
https://10.0.154.204:2379 |
+-----------------+---------+----------------------------+-------------------------+---------------
-----------+
```

记录不健康的 etcd 成员的 ID 和名称，因为稍后需要这些值。**$ etcdctl endpoint health** 命令将列出已删除的成员，直到完成替换过程并添加了新成员。

d. 通过向 **etcdctl member remove** 命令提供 ID 来删除不健康的 etcd 成员：

```
sh-4.2# etcdctl member remove 6fc1e7c9db35841d
```

输出示例

```
Member 6fc1e7c9db35841d removed from cluster ead669ce1fbfb346
```

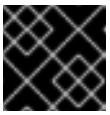
e. 再次查看成员列表，并确认成员已被删除：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+-----------------+---------+----------------------------+-------------------------+---------------
-----------+
|        ID       | STATUS  |            NAME            |       PEER ADDRS        |      CLIENT
ADDRS       |
+-----------------+---------+----------------------------+-------------------------+---------------
-----------+
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+-----------------+---------+----------------------------+-------------------------+---------------
-----------+
```

现在您可以退出节点 shell。



重要

删除成员后，在剩余的 etcd 实例重启时，集群可能无法访问。

2. 输入以下命令关闭仲裁保护：

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides":
{"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
```

此命令可确保您可以成功重新创建机密并推出静态 pod。

3. 删除已删除的不健康 etcd 成员的旧 secret。

a. 列出已删除的不健康 etcd 成员的 secret。

```
$ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal 1
```

1 传递您之前在这个过程中记录的不健康 etcd 成员的名称。

有一个对等的、服务和指标的 secret，如以下输出所示：

输出示例

```
etcd-peer-ip-10-0-131-183.ec2.internal             kubernetes.io/tls                 2      47m
etcd-serving-ip-10-0-131-183.ec2.internal          kubernetes.io/tls                 2      47m
etcd-serving-metrics-ip-10-0-131-183.ec2.internal  kubernetes.io/tls                 2
47m
```

b. 删除已删除的不健康 etcd 成员的 secret。

i. 删除 peer（对等）secret:

```
$ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
```

ii. 删除 serving secret:

```
$ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
```

iii. 删除 metrics secret:

```
$ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-
183.ec2.internal
```

4. 删除并重新创建 control plane 机器（也称为 master 机器）。重新创建此机器后，会强制一个新修订版本并自动扩展 etcd。
如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建 master 时使用的相同方法创建新的 master。

a. 获取不健康成员的机器。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                                 PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
clustername-8qw5l-master-0           Running   m4.xlarge   us-east-1   us-east-1a
3h37m   ip-10-0-131-183.ec2.internal   aws:///us-east-1a/i-0ec2782f8287dfb7e   stopped
1
clustername-8qw5l-master-1           Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2           Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-164-97.ec2.internal    aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
```

```
clustername-8qw5l-worker-us-east-1c-pkg26  Running  m4.large  us-east-1  us-east-1c  3h28m  ip-10-0-170-181.ec2.internal  aws:///us-east-1c/i-06861c00007751b0a  running
```

1 这是不健康节点的 control plane 机器 **ip-10-0-131-183.ec2.internal**。

b. 将机器配置保存到文件系统中的一个文件中：

```
$ oc get machine clustername-8qw5l-master-0 \ 1
    -n openshift-machine-api \
    -o yaml \
    > new-master-machine.yaml
```

1 为不健康的节点指定 control plane 机器的名称。

c. 编辑上一步中创建的 **new-master-machine.yaml** 文件，以分配新名称并删除不必要的字段。

i. 删除整个 **status** 部分：

```
status:
  addresses:
  - address: 10.0.131.183
    type: InternalIP
  - address: ip-10-0-131-183.ec2.internal
    type: InternalDNS
  - address: ip-10-0-131-183.ec2.internal
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Node
    name: ip-10-0-131-183.ec2.internal
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2020-04-20T16:53:50Z"
      lastTransitionTime: "2020-04-20T16:53:50Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: AWSMachineProviderStatus
```

ii. 将 **metadata.name** 字段更改为新名称。
建议您保留与旧机器相同的基础名称，并将结束号码改为下一个可用数字。在本例中，**clustername-8qw5l-master-0** 改为 **clustername-8qw5l-master-3**。

例如：

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

iii. 删除 **spec.providerID** 字段：

```
providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

d. 删除不健康成员的机器：

```
$ oc delete machine -n openshift-machine-api clustername-8qw5l-master-0 1
```

1 为不健康的节点指定 control plane 机器的名称。

e. 验证机器是否已删除：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
clustername-8qw5l-master-1          Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-164-97.ec2.internal    aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

f. 使用 **new-master-machine.yaml** 文件创建新机器：

```
$ oc apply -f new-master-machine.yaml
```

g. 验证新机器是否已创建：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
clustername-8qw5l-master-1          Running     m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-154-204.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
```

```
clustername-8qw5l-master-2                  Running     m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-164-97.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-master-3                  Provisioning  m4.xlarge   us-east-1   us-east-1a
85s     ip-10-0-133-53.ec2.internal   aws:///us-east-1a/i-015b0888fe17bc2c8   running
(1)
clustername-8qw5l-worker-us-east-1a-wbtgd   Running     m4.large    us-east-1   us-
east-1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb    Running     m4.large    us-east-1   us-east-
1b   3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b
running
clustername-8qw5l-worker-us-east-1c-pkg26    Running     m4.large    us-east-1   us-
east-1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

(1) 新机器 **clustername-8qw5l-master-3** 将被创建，并当阶段从 **Provisioning** 变为 **Running** 后就可以使用。

创建新机器可能需要几分钟时间。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

5. 输入以下命令重新打开仲裁保护：

```
$ oc patch etcd/cluster --type=merge -p '\{"spec": {"unsupportedConfigOverrides": null}}
```

6. 您可以输入以下命令验证 **unsupportedConfigOverrides** 部分是否已从对象中删除：

```
$ oc get etcd/cluster -oyaml
```

验证

1. 验证所有 etcd pod 是否都正常运行。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

输出示例

```
etcd-ip-10-0-133-53.ec2.internal             3/3     Running     0          7m49s
etcd-ip-10-0-164-97.ec2.internal             3/3     Running     0          123m
etcd-ip-10-0-154-204.ec2.internal            3/3     Running     0          124m
```

如果上一命令的输出只列出两个 pod，您可以手动强制重新部署 etcd。在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-
3339=ns )"'"}}' --type=merge (1)
```

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

2. 验证只有三个 etcd 成员。

a. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称：
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

b. 查看成员列表：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+--------------------------+--------------------------+----------------
-----------+
|        ID        | STATUS  |          NAME            |        PEER ADDRS        |      CLIENT
ADDRS       |
+------------------+---------+--------------------------+--------------------------+----------------
-----------+
| 5eb0d6b8ca24730c | started |  ip-10-0-133-53.ec2.internal |  https://10.0.133.53:2380 |
https://10.0.133.53:2379 |
| 757b6793e2408b6c | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+------------------+---------+--------------------------+--------------------------+----------------
-----------+
```

如果上一命令的输出列出了超过三个 etcd 成员，您必须删除不需要的成员。

警告

确保删除正确的 etcd 成员；如果删除了正常的 etcd 成员则有可能会导致仲裁丢失。

5.2.4.2. 替换其 etcd Pod 处于 crashlooping 状态的不健康 etcd 成员

此流程详细介绍了替换因 etcd pod 处于 crashlooping 状态造成不健康的 etcd 成员的步骤。

先决条件

- 您已找出不健康的 etcd 成员。
- 已确认 etcd pod 处于 crashlooping 状态。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。

重要

执行此流程前务必要进行 etcd 备份，以便在遇到任何问题时可以恢复集群。

流程

1. 停止处于 crashlooping 状态的 etcd pod。

   a. 对处于 crashlooping 状态的节点进行调试。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

   ```
   $ oc debug node/ip-10-0-131-183.ec2.internal 1
   ```

   1 使用不健康节点的名称来替换它。

   b. 将您的根目录改为 **/host** ：

   ```
   sh-4.2# chroot /host
   ```

   c. 将现有 etcd pod 文件从 Kubelet 清单目录中移出：

   ```
   sh-4.2# mkdir /var/lib/etcd-backup
   ```

   ```
   sh-4.2# mv /etc/kubernetes/manifests/etcd-pod.yaml /var/lib/etcd-backup/
   ```

   d. 将 etcd 数据目录移到不同的位置：

   ```
   sh-4.2# mv /var/lib/etcd/ /tmp
   ```

   现在您可以退出节点 shell。

2. 删除不健康的成员。

   a. 选择一个 *不在* 受影响节点上的 pod。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

   ```
   $ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
   ```

   输出示例

   ```
   etcd-ip-10-0-131-183.ec2.internal           2/3     Error       7          6h9m
   etcd-ip-10-0-164-97.ec2.internal            3/3     Running     0          6h6m
   etcd-ip-10-0-154-204.ec2.internal           3/3     Running     0          6h6m
   ```

   b. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

   ```
   $ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
   ```

   c. 查看成员列表：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
|        ID        | STATUS  |            NAME            |       PEER ADDRS         |      CLIENT
ADDRS       |
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
| 62bcf33650a7170a | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 |
https://10.0.131.183:2379 |
| b78e2856655bc2eb | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380
| https://10.0.154.204:2379 |
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
```

记录不健康的 etcd 成员的 ID 和名称，因为稍后需要这些值。

d. 通过向 **etcdctl member remove** 命令提供 ID 来删除不健康的 etcd 成员 :

```
sh-4.2# etcdctl member remove 62bcf33650a7170a
```

输出示例

```
Member 62bcf33650a7170a removed from cluster ead669ce1fbfb346
```

e. 再次查看成员列表，并确认成员已被删除：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
|        ID        | STATUS  |            NAME            |       PEER ADDRS         |      CLIENT
ADDRS       |
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
| b78e2856655bc2eb | started |  ip-10-0-164-97.ec2.internal |  https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380
| https://10.0.154.204:2379 |
+------------------+---------+----------------------------+--------------------------+----------------
-----------+
```

现在您可以退出节点 shell。

3. 输入以下命令关闭仲裁保护：

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides":
{"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
```

此命令可确保您可以成功重新创建机密并推出静态 pod。

4. 删除已删除的不健康 etcd 成员的旧 secret。

   a. 列出已删除的不健康 etcd 成员的 secret。

   ```
   $ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal 1
   ```

   1 传递您之前在这个过程中记录的不健康 etcd 成员的名称。

   有一个对等的、服务和指标的 secret，如以下输出所示：

   输出示例

   ```
   etcd-peer-ip-10-0-131-183.ec2.internal             kubernetes.io/tls                     2      47m
   etcd-serving-ip-10-0-131-183.ec2.internal          kubernetes.io/tls                     2      47m
   etcd-serving-metrics-ip-10-0-131-183.ec2.internal  kubernetes.io/tls                     2
   47m
   ```

   b. 删除已删除的不健康 etcd 成员的 secret。

      i. 删除 peer（对等）secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
      ```

      ii. 删除 serving secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
      ```

      iii. 删除 metrics secret:

      ```
      $ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-
      183.ec2.internal
      ```

5. 强制 etcd 重新部署。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

   ```
   $ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "single-master-recovery-
   '"$( date --rfc-3339=ns )"'"}}' --type=merge 1
   ```

   1 **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

   当 etcd 集群 Operator 执行重新部署时，它会确保所有 control plane 节点（也称为 master 节点）都有一个可正常工作的 etcd pod。

6. 输入以下命令重新打开仲裁保护：

   ```
   $ oc patch etcd/cluster --type=merge -p '\{"spec": {"unsupportedConfigOverrides": null}}
   ```

7. 您可以输入以下命令验证 **unsupportedConfigOverrides** 部分是否已从对象中删除：

```
$ oc get etcd/cluster -oyaml
```

验证

- 确认新成员可用且健康。

  a. 连接到正在运行的 etcd 容器。
  在一个终端中使用 cluster-admin 用户连接到集群，运行以下命令：

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

  b. 验证所有成员是否健康：

```
sh-4.2# etcdctl endpoint health
```

  输出示例

```
https://10.0.131.183:2379 is healthy: successfully committed proposal: took =
16.671434ms
https://10.0.154.204:2379 is healthy: successfully committed proposal: took =
16.698331ms
https://10.0.164.97:2379 is healthy: successfully committed proposal: took =
16.621645ms
```

5.2.4.3. 替换机器没有运行或节点未就绪的不健康裸机 etcd 成员

此流程详细介绍了替换因机器没有运行或节点未就绪造成不健康的裸机 etcd 成员的步骤。

如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建控制平面节点时使用的相同方法创建新的控制平面。

先决条件

- 您已找出不健康的裸机 etcd 成员。
- 您已确认机器没有运行，或者该节点未就绪。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。

重要

执行此流程前务必要进行 etcd 备份，以便在遇到任何问题时可以恢复集群。

流程

1. 验证并删除不健康的成员。

   a. 选择一个 *不在* 受影响节点上的 pod:
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get pods -n openshift-etcd -o wide | grep etcd | grep -v guard
```

输出示例

```
etcd-openshift-control-plane-0  5/5  Running  11  3h56m  192.168.10.9  openshift-control-plane-0  <none>  <none>
etcd-openshift-control-plane-1  5/5  Running  0   3h54m  192.168.10.10  openshift-control-plane-1  <none>  <none>
etcd-openshift-control-plane-2  5/5  Running  0   3h58m  192.168.10.11  openshift-control-plane-2  <none>  <none>
```

b. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称：
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc rsh -n openshift-etcd etcd-openshift-control-plane-0
```

c. 查看成员列表：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+--------------------+--------------------------+--------------------------+--------------------+
| ID               | STATUS  | NAME               | PEER ADDRS               | CLIENT ADDRS             | IS LEARNER |
+------------------+---------+--------------------+--------------------------+--------------------------+--------------------+
| 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380/ | https://192.168.10.11:2379/ | false |
| 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380/ | https://192.168.10.10:2379/ | false |
| cc3830a72fc357f9 | started | openshift-control-plane-0 | https://192.168.10.9:2380/ | https://192.168.10.9:2379/   | false |
+------------------+---------+--------------------+--------------------------+--------------------------+--------------------+
```

记录不健康的 etcd 成员的 ID 和名称，因为稍后需要这些值。**etcdctl endpoint health** 命令将列出已删除的成员，直到完成替换过程并添加了新成员。

d. 通过向 **etcdctl member remove** 命令提供 ID 来删除不健康的 etcd 成员：

警告

确保删除正确的 etcd 成员；如果删除了正常的 etcd 成员则有可能会导致仲裁丢失。

```
sh-4.2# etcdctl member remove 7a8197040a5126c8
```

输出示例

```
Member 7a8197040a5126c8 removed from cluster b23536c33f2cdd1b
```

e. 再次查看成员列表，并确认成员已被删除：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+--------------------+--------------------------+--------------------------
-+------------------------+
| ID               | STATUS  | NAME                    | PEER ADDRS               | CLIENT
ADDRS                | IS LEARNER |
+------------------+---------+--------------------+--------------------------+--------------------------
-+------------------------+
| 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380/ |
https://192.168.10.11:2379/ | false |
| 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380/ |
https://192.168.10.10:2379/ | false |
+------------------+---------+--------------------+--------------------------+--------------------------
-+------------------------+
```

现在您可以退出节点 shell。

重要

删除成员后，在剩余的 etcd 实例重启时，集群可能无法访问。

2. 输入以下命令关闭仲裁保护：

```
$ oc patch etcd/cluster --type=merge -p '{"spec": {"unsupportedConfigOverrides":
{"useUnsupportedUnsafeNonHANonProductionUnstableEtcd": true}}}'
```

此命令可确保您可以成功重新创建机密并推出静态 pod。

3. 运行以下命令，删除已删除的不健康 etcd 成员的旧 secret。

a. 列出已删除的不健康 etcd 成员的 secret。

```
$ oc get secrets -n openshift-etcd | grep openshift-control-plane-2
```

传递您之前在这个过程中记录的不健康 etcd 成员的名称。

有一个对等的、服务和指标的 secret，如以下输出所示：

```
etcd-peer-openshift-control-plane-2              kubernetes.io/tls   2   134m
etcd-serving-metrics-openshift-control-plane-2  kubernetes.io/tls   2   134m
etcd-serving-openshift-control-plane-2           kubernetes.io/tls   2   134m
```

b. 删除已删除的不健康 etcd 成员的 secret。

i. 删除 peer（对等）secret:

```
$ oc delete secret etcd-peer-openshift-control-plane-2 -n openshift-etcd

secret "etcd-peer-openshift-control-plane-2" deleted
```

ii. 删除 serving secret:

```
$ oc delete secret etcd-serving-metrics-openshift-control-plane-2 -n openshift-etcd

secret "etcd-serving-metrics-openshift-control-plane-2" deleted
```

iii. 删除 metrics secret:

```
$ oc delete secret etcd-serving-openshift-control-plane-2 -n openshift-etcd

secret "etcd-serving-openshift-control-plane-2" deleted
```

4. 删除 control plane 机器。
如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建控制平面节点时使用的相同方法创建新的控制平面。

a. 获取不健康成员的机器。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                              PHASE     TYPE   REGION   ZONE   AGE     NODE
PROVIDERID                                                              STATE
examplecluster-control-plane-0    Running                          3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned (1)
examplecluster-control-plane-1    Running                          3h11m   openshift-control-
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-control-plane-2    Running                          3h11m   openshift-control-
plane-2   baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-
61d8-410f-be5b-6a395b056135   externally provisioned
examplecluster-compute-0          Running                          165m    openshift-compute-0
baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-4bb3-80ec-
13a31858241f          provisioned
examplecluster-compute-1          Running                          165m    openshift-compute-1
baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-4241-91dc-
e7ea72ab13b9          provisioned
```

(1) 这是不健康节点的 control plane 机器，**examplecluster-control-plane-2**。

b. 将机器配置保存到文件系统中的一个文件中：

```
$ oc get machine examplecluster-control-plane-2 \ (1)
```

```
-n openshift-machine-api \
-o yaml \
> new-master-machine.yaml
```

1 为不健康的节点指定 control plane 机器的名称。

c. 编辑上一步中创建的 **new-master-machine.yaml** 文件，以分配新名称并删除不必要的字段。

i. 删除整个 **status** 部分：

```
status:
  addresses:
  - address: ""
    type: InternalIP
  - address: fe80::4adf:37ff:feb0:8aa1%ens1f1.373
    type: InternalDNS
  - address: fe80::4adf:37ff:feb0:8aa1%ens1f1.371
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Machine
    name: fe80::4adf:37ff:feb0:8aa1%ens1f1.372
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: machine.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2020-04-20T16:53:50Z"
      lastTransitionTime: "2020-04-20T16:53:50Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: Machine
```

5. 将 **metadata.name** 字段更改为新名称。
建议您保留与旧机器相同的基础名称，并将结束号码改为下一个可用数字。在本例中，**examplecluster-control-plane-2** 改为 **examplecluster-control-plane-3**。

例如：

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: examplecluster-control-plane-3
  ...
```

a. 删除 **spec.providerID** 字段：

```
  providerID: baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-61d8-410f-be5b-6a395b056135
```

b. 删除 **metadata.annotations** 和 **metadata.generation** 字段：

```
  annotations:
    machine.openshift.io/instance-state: externally provisioned
  ...
  generation: 2
```

c. 删除 **spec.conditions**、**spec.lastUpdated**、**spec.nodeRef** 和 **spec.phase** 字段：

```
    lastTransitionTime: "2022-08-03T08:40:36Z"
  message: 'Drain operation currently blocked by: [{Name:EtcdQuorumOperator
  Owner:clusteroperator/etcd}]'
  reason: HookPresent
  severity: Warning
  status: "False"

  type: Drainable
  lastTransitionTime: "2022-08-03T08:39:55Z"
  status: "True"
  type: InstanceExists

  lastTransitionTime: "2022-08-03T08:36:37Z"
  status: "True"
  type: Terminable
  lastUpdated: "2022-08-03T08:40:36Z"
  nodeRef:
  kind: Node
  name: openshift-control-plane-2
  uid: 788df282-6507-4ea2-9a43-24f237ccbc3c
  phase: Running
```

6. 运行以下命令，确保 Bare Metal Operator 可用：

```
$ oc get clusteroperator baremetal
```

输出示例

```
NAME       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
baremetal  4.11.3    True        False         False      3d15h
```

7. 使用以下命令删除不健康成员的机器：

```
$ oc delete machine -n openshift-machine-api examplecluster-control-plane-2
```

如果删除机器因任何原因或者命令被移动而延迟而延迟而延迟，您可以通过删除机器对象终结器字段来强制删除。

重要

不要通过按 **Ctrl+c** 中断机器删除。您必须允许命令继续完成。打开一个新的终端窗口来编辑并删除 finalizer 字段。

```
$ oc edit machine -n openshift-machine-api examplecluster-control-plane-2
```

a. 查找并删除字段：

```
finalizers:
- machine.machine.openshift.io
```

保存您的更改：

```
machine.machine.openshift.io/examplecluster-control-plane-2 edited
```

b. 运行以下命令验证机器是否已删除：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                              PHASE     TYPE   REGION   ZONE   AGE     NODE
PROVIDERID                                                              STATE
examplecluster-control-plane-0    Running                          3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned
examplecluster-control-plane-1    Running                          3h11m   openshift-control-
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-compute-0          Running                          165m    openshift-compute-0
baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-4bb3-80ec-
13a31858241f          provisioned
examplecluster-compute-1          Running                          165m    openshift-compute-1
baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-4241-91dc-
e7ea72ab13b9          provisioned
```

8. 使用以下命令删除旧 **BareMetalHost** 对象：

```
$ oc delete bmh openshift-control-plane-2 -n openshift-machine-api
```

输出示例

```
baremetalhost.metal3.io "openshift-control-plane-2" deleted
```

删除 **BareMetalHost** 和 **Machine** 对象后，**Machine** Controller 会自动删除 **Node** 对象。

如果删除 **BareMetalHost** 对象后，机器节点需要过量删除时间，可以使用以下方法删除机器节点：

```
$ oc delete node openshift-control-plane-2

node "openshift-control-plane-2" deleted
```

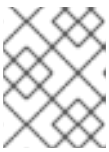
验证节点已被删除：

```
$ oc get nodes

NAME                    STATUS ROLES    AGE    VERSION
openshift-control-plane-0 Ready master 3h24m v1.24.0+9546431
openshift-control-plane-1 Ready master 3h24m v1.24.0+9546431
openshift-compute-0       Ready worker 176m v1.24.0+9546431
openshift-compute-1       Ready worker 176m v1.24.0+9546431
```

9. 创建新的 **BareMetalHost** 对象和 secret，以存储 BMC 凭证：

```
$ cat <<EOF | oc apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: openshift-control-plane-2-bmc-secret
  namespace: openshift-machine-api
data:
  password: <password>
  username: <username>
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: openshift-control-plane-2
  namespace: openshift-machine-api
spec:
  automatedCleaningMode: disabled
  bmc:
    address: redfish://10.46.61.18:443/redfish/v1/Systems/1
    credentialsName: openshift-control-plane-2-bmc-secret
    disableCertificateVerification: true
  bootMACAddress: 48:df:37:b0:8a:a0
  bootMode: UEFI
  externallyProvisioned: false
  online: true
  rootDeviceHints:
    deviceName: /dev/sda
  userData:
    name: master-user-data-managed
    namespace: openshift-machine-api
EOF
```



注意

用户名和密码可从其他裸机主机的 secret 中找到。**bmc:address** 中使用的协议可以从其他 bmh 对象获取。

重要

如果您从现有 control plane 主机重复使用 **BareMetalHost** 对象定义，请不要将 external **Provisioned** 字段保留为 **true**。

如果 OpenShift Container Platform 安装程序置备，现有 control plane **BareMetalHost** 对象可能会将 **externallyProvisioned** 标记设为 **true**。

检查完成后，**BareMetalHost** 对象会被创建并可用置备。

10. 使用可用的 **BareMetalHost** 对象验证创建过程：

```
$ oc get bmh -n openshift-machine-api

NAME                 STATE                  CONSUMER                    ONLINE ERROR   AGE
openshift-control-plane-0 externally provisioned examplecluster-control-plane-0 true
4h48m
openshift-control-plane-1 externally provisioned examplecluster-control-plane-1 true
4h48m
openshift-control-plane-2 available          examplecluster-control-plane-3 true     47m
openshift-compute-0       provisioned        examplecluster-compute-0       true     4h48m
openshift-compute-1       provisioned        examplecluster-compute-1       true     4h48m
```

a. 使用 **new-master-machine.yaml** 文件创建新 control plane 机器：

```
$ oc apply -f new-master-machine.yaml
```

b. 验证新机器是否已创建：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例

```
NAME                                PHASE     TYPE   REGION   ZONE   AGE     NODE
PROVIDERID                                                                  STATE
examplecluster-control-plane-0      Running                          3h11m   openshift-control-
plane-0   baremetalhost:///openshift-machine-api/openshift-control-plane-0/da1ebe11-
3ff2-41c5-b099-0aa41222964e   externally provisioned (1)
examplecluster-control-plane-1      Running                          3h11m   openshift-control-
plane-1   baremetalhost:///openshift-machine-api/openshift-control-plane-1/d9f9acbc-
329c-475e-8d81-03b20280a3e1   externally provisioned
examplecluster-control-plane-2      Running                          3h11m   openshift-control-
plane-2   baremetalhost:///openshift-machine-api/openshift-control-plane-2/3354bdac-
61d8-410f-be5b-6a395b056135   externally provisioned
examplecluster-compute-0            Running                          165m    openshift-compute-
0         baremetalhost:///openshift-machine-api/openshift-compute-0/3d685b81-7410-
4bb3-80ec-13a31858241f        provisioned
examplecluster-compute-1            Running                          165m    openshift-compute-
1         baremetalhost:///openshift-machine-api/openshift-compute-1/0fdae6eb-2066-
4241-91dc-e7ea72ab13b9        provisioned
```

(1) 新机器 **clustername-8qw5l-master-3** 会被创建，并在阶段从 **Provisioning** 变为 **Running** 后就绪。

创建新机器需要几分钟时间。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

c. 运行以下命令验证裸机主机是否被置备，且没有报告的错误：

```
$ oc get bmh -n openshift-machine-api
```

输出示例

```
$ oc get bmh -n openshift-machine-api
NAME                 STATE                  CONSUMER                    ONLINE ERROR AGE
openshift-control-plane-0 externally provisioned examplecluster-control-plane-0 true
4h48m
openshift-control-plane-1 externally provisioned examplecluster-control-plane-1 true
4h48m
openshift-control-plane-2 provisioned          examplecluster-control-plane-3 true
47m
openshift-compute-0       provisioned          examplecluster-compute-0       true
4h48m
openshift-compute-1       provisioned          examplecluster-compute-1       true
4h48m
```

d. 运行以下命令验证新节点是否已添加并处于就绪状态：

```
$ oc get nodes
```

输出示例

```
$ oc get nodes
NAME                 STATUS ROLES    AGE    VERSION
openshift-control-plane-0 Ready master 4h26m v1.24.0+9546431
openshift-control-plane-1 Ready master 4h26m v1.24.0+9546431
openshift-control-plane-2 Ready master 12m   v1.24.0+9546431
openshift-compute-0       Ready worker 3h58m v1.24.0+9546431
openshift-compute-1       Ready worker 3h58m v1.24.0+9546431
```

11. 输入以下命令重新打开仲裁保护：

```
$ oc patch etcd/cluster --type=merge -p '\{"spec": {"unsupportedConfigOverrides": null}}
```

12. 您可以输入以下命令验证 **unsupportedConfigOverrides** 部分是否已从对象中删除：

```
$ oc get etcd/cluster -oyaml
```

验证

1. 验证所有 etcd pod 是否都正常运行。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get pods -n openshift-etcd -o wide | grep etcd | grep -v guard
```

输出示例

```
etcd-openshift-control-plane-0     5/5     Running     0     105m
etcd-openshift-control-plane-1     5/5     Running     0     107m
etcd-openshift-control-plane-2     5/5     Running     0     103m
```

如果上一命令的输出只列出两个 pod，您可以手动强制重新部署 etcd。在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```
(1)

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

要验证是否有完全有三个 etcd 成员，连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称。在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc rsh -n openshift-etcd etcd-openshift-control-plane-0
```

2. 查看成员列表：

```
sh-4.2# etcdctl member list -w table
```

输出示例

```
+------------------+---------+--------------------+--------------------------+--------------------------+---
--------------+
|        ID        | STATUS  |       NAME         |       PEER ADDRS         |       CLIENT ADDRS
|   IS LEARNER     |
+------------------+---------+--------------------+--------------------------+--------------------------+---
--------------+
| 7a8197040a5126c8 | started | openshift-control-plane-2 | https://192.168.10.11:2380 |
https://192.168.10.11:2379 |    false |
| 8d5abe9669a39192 | started | openshift-control-plane-1 | https://192.168.10.10:2380 |
https://192.168.10.10:2379 |    false |
| cc3830a72fc357f9 | started | openshift-control-plane-0 | https://192.168.10.9:2380 |
https://192.168.10.9:2379 |     false |
+------------------+---------+--------------------+--------------------------+--------------------------+---
--------------+
```

注意

如果上一命令的输出列出了超过三个 etcd 成员，您必须删除不需要的成员。

3. 运行以下命令，验证所有 etcd 成员是否健康：

```
# etcdctl endpoint health --cluster
```

输出示例

```
https://192.168.10.10:2379 is healthy: successfully committed proposal: took = 8.973065ms
https://192.168.10.9:2379 is healthy: successfully committed proposal: took = 11.559829ms
https://192.168.10.11:2379 is healthy: successfully committed proposal: took = 11.665203ms
```

4. 运行以下命令，验证所有节点是否处于最新的修订版本：

```
$ oc get etcd -o=jsonpath='{range.items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

```
AllNodesAtLatestRevision
```

5.3. 灾难恢复

5.3.1. 关于灾难恢复

灾难恢复文档为管理员提供了如何从 OpenShift Container Platform 集群可能出现的几个灾难情形中恢复的信息。作为管理员，您可能需要遵循以下一个或多个步骤将集群恢复为工作状态。

重要

灾难恢复要求您至少有一个健康的 control plane 主机（也称为 master 主机）。

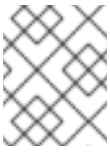
恢复到一个以前的集群状态

如果您希望将集群恢复到一个以前的状态时（例如，管理员错误地删除了一些关键信息），则可以使用这个解决方案。这包括您丢失了大多数 control plane 主机并导致 etcd 仲裁丢失，且集群离线的情况。只要您执行了 etcd 备份，就可以按照这个步骤将集群恢复到之前的状态。
如果适用，可能还需要从过期的 control plane 证书中恢复。

警告

在一个正在运行的集群中恢复到以前的集群状态是破坏性的，而不稳定的操作。这仅应作为最后的手段使用。

在执行恢复前，请参阅关于恢复集群状态以了解有关对集群的影响的更多信息。



注意

如果大多数 master 仍可用，且仍有 etcd 仲裁，请按照以下步骤替换一个不健康的 etcd 成员。

从 control plane 证书已过期的情况下恢复

如果 control plane 证书已经过期，则可以使用这个解决方案。例如：在第一次证书轮转前（在安装后 24 小时内）关闭了集群，您的证书将不会被轮转，且会过期。可以按照以下步骤从已过期的 control plane 证书中恢复。

5.3.2. 恢复到一个以前的集群状态

为了将集群还原到以前的状态，您必须已通过创建快照备份了 etcd 数据 。您将需要使用此快照来还原集群状态。

5.3.2.1. 关于恢复集群状态

您可以使用 etcd 备份将集群恢复到以前的状态。在以下情况中可以使用这个方法进行恢复：

- 集群丢失了大多数 control plane 主机（仲裁丢失）。
- 管理员删除了一些关键内容，必须恢复才能恢复集群。

警告

在一个正在运行的集群中恢复到以前的集群状态是破坏性的，而不稳定的操作。这仅应作为最后的手段使用。

如果您可以使用 Kubernetes API 服务器检索数据，则代表 etcd 可用，且您不应该使用 etcd 备份来恢复。

恢复 etcd 实际相当于把集群返回到以前的一个状态，所有客户端都会遇到一个有冲突的、并行历史记录。这会影响 kubelet、Kubernetes 控制器、SDN 控制器和持久性卷控制器等监视组件的行为。

当 etcd 中的内容与磁盘上的实际内容不匹配时，可能会导致 Operator churn，从而导致 Kubernetes API 服务器、Kubernetes 控制器管理器、Kubernetes 调度程序和 etcd 的 Operator 在磁盘上的文件与 etcd 中的内容冲突时卡住。这可能需要手动操作来解决问题。

在极端情况下，集群可能会丢失持久性卷跟踪，删除已不存在的关键工作负载，重新镜像机器，以及重写带有过期证书的 CA 捆绑包。

5.3.2.2. 恢复到一个以前的集群状态

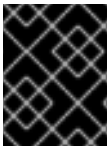
您可以使用已保存的 etcd 备份来恢复以前的集群状态，或者恢复丢失了大多数 control plane 主机（也称为 master 主机）的集群。

重要

恢复集群时，必须使用同一 z-stream 发行版本中获取的 etcd 备份。例如，OpenShift Container Platform 4.7.2 集群必须使用从 4.7.2 开始的 etcd 备份。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 用作恢复主机的健康 control plane 主机。
- SSH 对 control plane 主机的访问。
- 包含从同一备份中获取的 etcd 快照和静态 pod 资源的备份目录。该目录中的文件名必须采用以下格式: **snapshot_<datetimestamp>.db** 和 **static_kuberesources_<datetimestamp>.tar.gz**。



重要

对于非恢复 control plane 节点，不需要建立 SSH 连接或停止静态 pod。您可以逐个删除并重新创建其他非恢复 control plane 机器。

流程

1. 选择一个要用作恢复主机的 control plane 主机。这是您要在其中运行恢复操作的主机。
2. 建立到每个 control plane 节点（包括恢复主机）的 SSH 连接。
   恢复过程启动后，Kubernetes API 服务器将无法访问，因此您无法访问 control plane 节点。因此，建议在一个单独的终端中建立到每个control plane 主机的 SSH 连接。

   重要

   如果没有完成这个步骤，将无法访问 control plane 主机来完成恢复过程，您将无法从这个状态恢复集群。

3. 将 etcd 备份目录复制复制到恢复 control plane 主机上。
   此流程假设您将 **backup** 目录（其中包含 etcd 快照和静态 pod 资源）复制到恢复 control plane 主机的 **/home/core/** 目录中。
4. 在任何其他 control plane 节点上停止静态 pod。

   注意

   不需要手动停止恢复主机上的 pod。恢复脚本将停止恢复主机上的 pod。

   a. 访问不是恢复主机的 control plane 主机。

   b. 将现有 etcd pod 文件从 Kubelet 清单目录中移出：

   ```
   $ sudo mv /etc/kubernetes/manifests/etcd-pod.yaml /tmp
   ```

   c. 验证 etcd pod 是否已停止。

   ```
   $ sudo crictl ps | grep etcd | grep -v operator
   ```

   命令输出应该为空。如果它不是空的，请等待几分钟后再重新检查。

   d. 将现有 Kubernetes API 服务器 pod 文件移出 kubelet 清单目录中：

   ```
   $ sudo mv /etc/kubernetes/manifests/kube-apiserver-pod.yaml /tmp
   ```

   e. 验证 Kubernetes API 服务器 pod 是否已停止。

   ```
   $ sudo crictl ps | grep kube-apiserver | grep -v operator
   ```

   命令输出应该为空。如果它不是空的，请等待几分钟后再重新检查。

   f. 将 etcd 数据目录移到不同的位置：

   ```
   $ sudo mv /var/lib/etcd/ /tmp
   ```

g. 在其他不是恢复主机的 control plane 主机上重复此步骤。

5. 访问恢复 control plane 主机。

6. 如果启用了集群范围的代理，请确定已导出了 **NO_PROXY**、**HTTP_PROXY**和 **HTTPS_PROXY** 环境变量。

提示

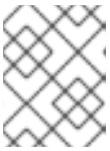
您可以通过查看 **oc get proxy cluster -o yaml** 的输出来检查代理是否已启用。如果 **httpProxy**、**httpsProxy**和 **noProxy** 字段设置了值，则会启用代理。

7. 在恢复 control plane 主机上运行恢复脚本，提供到 etcd 备份目录的路径：

```
$ sudo -E /usr/local/bin/cluster-restore.sh /home/core/backup
```

脚本输出示例

```
...stopping kube-scheduler-pod.yaml
...stopping kube-controller-manager-pod.yaml
...stopping etcd-pod.yaml
...stopping kube-apiserver-pod.yaml
Waiting for container etcd to stop
.complete
Waiting for container etcdctl to stop
.............................complete
Waiting for container etcd-metrics to stop
complete
Waiting for container kube-controller-manager to stop
complete
Waiting for container kube-apiserver to stop
.....................................................................................complete
Waiting for container kube-scheduler to stop
complete
Moving etcd data-dir /var/lib/etcd/member to /var/lib/etcd-backup
starting restore-etcd static pod
starting kube-apiserver-pod.yaml
static-pod-resources/kube-apiserver-pod-7/kube-apiserver-pod.yaml
starting kube-controller-manager-pod.yaml
static-pod-resources/kube-controller-manager-pod-7/kube-controller-manager-pod.yaml
starting kube-scheduler-pod.yaml
static-pod-resources/kube-scheduler-pod-8/kube-scheduler-pod.yaml
```



注意

如果在上次 etcd 备份后更新了节点，则恢复过程可能会导致节点进入 **NotReady** 状态。

8. 检查节点以确保它们处于 **Ready** 状态。

a. 运行以下命令:

```
$ oc get nodes -w
```

输出示例

```
NAME              STATUS  ROLES         AGE     VERSION
host-172-25-75-28   Ready   master        3d20h   v1.23.3+e419edf
host-172-25-75-38   Ready   infra,worker  3d20h   v1.23.3+e419edf
host-172-25-75-40   Ready   master        3d20h   v1.23.3+e419edf
host-172-25-75-65   Ready   master        3d20h   v1.23.3+e419edf
host-172-25-75-74   Ready   infra,worker  3d20h   v1.23.3+e419edf
host-172-25-75-79   Ready   worker        3d20h   v1.23.3+e419edf
host-172-25-75-86   Ready   worker        3d20h   v1.23.3+e419edf
host-172-25-75-98   Ready   infra,worker  3d20h   v1.23.3+e419edf
```

所有节点都可能需要几分钟时间报告其状态。

b. 如果有任何节点处于 **NotReady** 状态，登录到节点，并从每个节点上的 **/var/lib/kubelet/pki** 目录中删除所有 PEM 文件。您可以 SSH 到节点，或使用 web 控制台中的终端窗口。

```
$  ssh -i <ssh-key-path> core@<master-hostname>
```

pki 目录示例

```
sh-4.4# pwd
/var/lib/kubelet/pki
sh-4.4# ls
kubelet-client-2022-04-28-11-24-09.pem  kubelet-server-2022-04-28-11-24-15.pem
kubelet-client-current.pem              kubelet-server-current.pem
```

9. 在所有 control plane 主机上重启 kubelet 服务。

a. 在恢复主机中运行以下命令：

```
$ sudo systemctl restart kubelet.service
```

b. 在所有其他 control plane 主机上重复此步骤。

10. 批准待处理的 CSR：

a. 获取当前 CSR 列表：

```
$ oc get csr
```

输出示例

```
NAME        AGE    SIGNERNAME                            REQUESTOR
CONDITION
csr-2s94x   8m3s   kubernetes.io/kubelet-serving         system:node:<node_name>
Pending 1
csr-4bd6t   8m3s   kubernetes.io/kubelet-serving         system:node:<node_name>
Pending 2
csr-4hl85   13m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
3
csr-zhhhp   3m8s   kubernetes.io/kube-apiserver-client-kubelet
```

```
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
4
...
```

1 2 一个待处理的 kubelet 服务 CSR（用于用户置备的安装）。

3 4 一个待处理的 **node-bootstrapper** CSR。

b. 查看一个 CSR 的详细信息以验证其是否有效：

```
$ oc describe csr <csr_name> 1
```

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 **node-bootstrapper** CSR：

```
$ oc adm certificate approve <csr_name>
```

d. 对于用户置备的安装，请批准每个有效的 kubelet 服务 CSR：

```
$ oc adm certificate approve <csr_name>
```

11. 确认单个成员 control plane 已被成功启动。

a. 从恢复主机上，验证 etcd 容器是否正在运行。

```
$ sudo crictl ps | grep etcd | grep -v operator
```

输出示例

```
3ad41b7908e32
36f86e2eeaaffe662df0d21041eb22b8198e0e58abeeae8c743c3e6e977e8009
About a minute ago   Running             etcd                                0
7c05f8af362f0
```

b. 从恢复主机上，验证 etcd pod 是否正在运行。

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

注意

如果您试图在运行这个命令前运行 **oc login** 并接收以下错误，请等待一些时间以便身份验证控制器启动并再次尝试。

```
Unable to connect to the server: EOF
```

输出示例

```
NAME                                READY   STATUS      RESTARTS   AGE
etcd-ip-10-0-143-125.ec2.internal   1/1     Running     1          2m47s
```

如果状态是 **Pending**，或者输出中列出了多个正在运行的 etcd pod，请等待几分钟，然后再次检查。

c. 对不是恢复主机的每个已丢失的 control plane 主机重复此步骤。

12. 逐个删除并重新创建其他非恢复 control plane 机器。重新创建此机器后，会强制一个新修订版本并自动扩展 etcd。
如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建控制平面节点时使用的相同方法创建新的控制平面。

警告

不要为恢复主机删除和重新创建计算机。

a. 为丢失的 control plane 主机之一获取机器。
在一个终端中使用 cluster-admin 用户连接到集群，运行以下命令：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例：

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
clustername-8qw5l-master-0          Running   m4.xlarge   us-east-1   us-east-1a
3h37m   ip-10-0-131-183.ec2.internal   aws:///us-east-1a/i-0ec2782f8287dfb7e   stopped
1
clustername-8qw5l-master-1          Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

1 这是用于丢失的 control plane 主机 **ip-10-0-131-183.ec2.internal** 的 control plane 机器。

b. 将机器配置保存到文件系统中的一个文件中：

```
$ oc get machine clustername-8qw5l-master-0 \ 1
    -n openshift-machine-api \
    -o yaml \
    > new-master-machine.yaml
```

-

1 为丢失的 control plane 主机指定 control plane 机器的名称。

c. 编辑上一步中创建的 **new-master-machine.yaml** 文件，以分配新名称并删除不必要的字段。

i. 删除整个 **status** 部分：

```
status:
  addresses:
  - address: 10.0.131.183
    type: InternalIP
  - address: ip-10-0-131-183.ec2.internal
    type: InternalDNS
  - address: ip-10-0-131-183.ec2.internal
    type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Node
    name: ip-10-0-131-183.ec2.internal
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
    - lastProbeTime: "2020-04-20T16:53:50Z"
      lastTransitionTime: "2020-04-20T16:53:50Z"
      message: machine successfully created
      reason: MachineCreationSucceeded
      status: "True"
      type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: AWSMachineProviderStatus
```

ii. 将 **metadata.name** 字段更改为新名称。
建议您保留与旧机器相同的基础名称，并将结束号码改为下一个可用数字。在本例中，**clustername-8qw5l-master-0** 被改为 **clustername-8qw5l-master-3** ：

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

iii. 删除 **spec.providerID** 字段：

```
providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

iv. 删除 **metadata.annotations** 和 **metadata.generation** 字段：

```
annotations:
  machine.openshift.io/instance-state: running
```

```
...
generation: 2
```

v. 删除 **metadata.resourceVersion** 和 **metadata.uid** 字段：

```
resourceVersion: "13291"
uid: a282eb70-40a2-4e89-8009-d05dd420d31a
```

d. 删除丢失的 control plane 主机的机器：

```
$ oc delete machine -n openshift-machine-api clustername-8qw5l-master-0 1
```

1 为丢失的 control plane 主机指定 control plane 机器的名称。

e. 验证机器是否已删除：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例：

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                          PROVIDERID                          STATE
clustername-8qw5l-master-1          Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-worker-us-east-1a-wbtgd   Running   m4.large   us-east-1   us-east-
1a   3h28m   ip-10-0-129-226.ec2.internal   aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb   Running   m4.large   us-east-1   us-east-1b
3h28m   ip-10-0-144-248.ec2.internal   aws:///us-east-1b/i-0cb45ac45a166173b   running
clustername-8qw5l-worker-us-east-1c-pkg26   Running   m4.large   us-east-1   us-east-
1c   3h28m   ip-10-0-170-181.ec2.internal   aws:///us-east-1c/i-06861c00007751b0a
running
```

f. 使用 **new-master-machine.yaml** 文件创建新机器：

```
$ oc apply -f new-master-machine.yaml
```

g. 验证新机器是否已创建：

```
$ oc get machines -n openshift-machine-api -o wide
```

输出示例：

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                          PROVIDERID                          STATE
clustername-8qw5l-master-1          Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-master-3          Provisioning   m4.xlarge   us-east-1   us-east-1a
```

```
85s    ip-10-0-173-171.ec2.internal    aws:///us-east-1a/i-015b0888fe17bc2c8  running
(1)
clustername-8qw5l-worker-us-east-1a-wbtgd  Running      m4.large  us-east-1  us-
east-1a  3h28m  ip-10-0-129-226.ec2.internal  aws:///us-east-1a/i-010ef6279b4662ced
running
clustername-8qw5l-worker-us-east-1b-lrdxb  Running      m4.large  us-east-1  us-east-
1b  3h28m  ip-10-0-144-248.ec2.internal  aws:///us-east-1b/i-0cb45ac45a166173b
running
clustername-8qw5l-worker-us-east-1c-pkg26  Running      m4.large  us-east-1  us-
east-1c  3h28m  ip-10-0-170-181.ec2.internal  aws:///us-east-1c/i-06861c00007751b0a
running
```

(1) 新机器 **clustername-8qw5l-master-3** 会被创建，并在阶段从 **Provisioning** 变为 **Running** 后就绪。

创建新机器可能需要几分钟时间。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

h. 对不是恢复主机的每个已丢失的 control plane 主机重复此步骤。

13. 在一个单独的终端窗口中，使用以下命令以具有 **cluster-admin** 角色的用户身份登录到集群：

```
$ oc login -u <cluster_admin> (1)
```

(1) 对于 **<cluster_admin>**，使用 **cluster-admin** 角色指定一个用户名。

14. 强制 etcd 重新部署。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc patch etcd cluster -p='{"spec": {"forceRedeploymentReason": "recovery-"$( date --rfc-
3339=ns )""}}' --type=merge (1)
```

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

当 etcd cluster Operator 执行重新部署时，现有节点开始使用与初始 bootstrap 扩展类似的新 pod。

15. 验证所有节点是否已更新至最新的修订版本。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

查看 etcd 的**NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新的修订。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

```
AllNodesAtLatestRevision
3 nodes are at revision 7 (1)
```

(1) 在本例中，最新的修订版本号是 **7**。

如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

16. 在重新部署 etcd 后，为 control plane 强制进行新的 rollout。由于 kubelet 使用内部负载平衡器连接到 API 服务器，因此 Kubernetes API 将在其他节点上重新安装自己。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令。

    a. 为 Kubernetes API 服务器强制进行新的推出部署：

    ```
    $ oc patch kubeapiserver cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
    ```

    验证所有节点是否已更新至最新的修订版本。

    ```
    $ oc get kubeapiserver -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
    ```

    查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

    ```
    AllNodesAtLatestRevision
    3 nodes are at revision 7 (1)
    ```

    (1) 在本例中，最新的修订版本号是 **7**。

    如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

    b. 为 Kubernetes 控制器管理器强制进行新的推出部署：

    ```
    $ oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
    ```

    验证所有节点是否已更新至最新的修订版本。

    ```
    $ oc get kubecontrollermanager -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
    ```

    查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

    ```
    AllNodesAtLatestRevision
    3 nodes are at revision 7 (1)
    ```

    (1) 在本例中，最新的修订版本号是 **7**。

    如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

    c. 为 Kubernetes 调度程序强制进行新的推出部署：

```
$ oc patch kubescheduler cluster -p='{"spec": {"forceRedeploymentReason": "recovery-
'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

验证所有节点是否已更新至最新的修订版本。

```
$ oc get kubescheduler -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

```
AllNodesAtLatestRevision
3 nodes are at revision 7 (1)
```

(1) 在本例中，最新的修订版本号是 **7**。

如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

17. 验证所有 control plane 主机是否已启动并加入集群。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

输出示例

```
etcd-ip-10-0-143-125.ec2.internal        2/2     Running     0          9h
etcd-ip-10-0-154-194.ec2.internal        2/2     Running     0          9h
etcd-ip-10-0-173-171.ec2.internal        2/2     Running     0          9h
```

为确保所有工作负载在恢复过程后返回到正常操作，请重启存储 Kubernetes API 信息的每个 pod。这包括 OpenShift Container Platform 组件，如路由器、Operator 和第三方组件。

请注意，在完成这个过程后，可能需要几分钟才能恢复所有服务。例如，在重启 OAuth 服务器 pod 前，使用 **oc login** 进行身份验证可能无法立即正常工作。

5.3.2.3. 恢复持久性存储状态的问题和解决方法

如果您的 OpenShift Container Platform 集群使用任何形式的持久性存储，集群的状态通常存储在 etcd 外部。它可能是在 pod 中运行的 Elasticsearch 集群，或者在 **StatefulSet** 对象中运行的数据库。从 etcd 备份中恢复时，还会恢复 OpenShift Container Platform 中工作负载的状态。但是，如果 etcd 快照是旧的，其状态可能无效或过期。

重要

持久性卷（PV）的内容绝不会属于 etcd 快照的一部分。从 etcd 快照恢复 OpenShift Container Platform 集群时，非关键工作负载可能会访问关键数据，反之亦然。

以下是生成过时状态的一些示例情况：

- MySQL 数据库在由 PV 对象支持的 pod 中运行。从 etcd 快照恢复 OpenShift Container Platform 不会使卷恢复到存储供应商上，且不会生成正在运行的 MySQL pod，尽管 pod 会重复尝试启动。您必须通过在存储供应商中恢复卷，然后编辑 PV 以指向新卷来手动恢复这个 pod。

- Pod P1 使用卷 A，它附加到节点 X。如果另一个 pod 在节点 Y 上使用相同的卷，则执行 etcd 恢复时，pod P1 可能无法正确启动，因为卷仍然被附加到节点 Y。OpenShift Container Platform 并不知道附加，且不会自动分离它。发生这种情况时，卷必须从节点 Y 手动分离，以便卷可以在节点 X 上附加，然后 pod P1 才可以启动。

- 在执行 etcd 快照后，云供应商或存储供应商凭证会被更新。这会导致任何依赖于这些凭证的 CSI 驱动程序或 Operator 无法正常工作。您可能需要手动更新这些驱动程序或 Operator 所需的凭证。

- 在生成 etcd 快照后，会从 OpenShift Container Platform 节点中删除或重命名设备。Local Storage Operator 会为从 **/dev/disk/by-id** 或 **/dev** 目录中管理的每个 PV 创建符号链接。这种情况可能会导致本地 PV 引用不再存在的设备。
  要解决这个问题，管理员必须：

  1. 手动删除带有无效设备的 PV。
  2. 从对应节点中删除符号链接。
  3. 删除 **LocalVolume** 或 **LocalVolumeSet** 对象（请参阅 *Storage → Configuring persistent storage → Persistent storage → Persistent storage → Deleting the Local Storage Operator Resources*）。

其他资源

- 如需有关如何创建堡垒主机来访问 OpenShift Container Platform 实例和使用 SSH 的 control plane 节点，请参阅访问主机。

5.3.3. 从 control plane 证书已过期的情况下恢复

5.3.3.1. 从 control plane 证书已过期的情况下恢复

集群可以从过期的 control plane 证书中自动恢复。

但是，您需要手动批准待处理的 **node-bootstrapper** 证书签名请求（CSR）来恢复 kubelet 证书。对于用户置备的安装，您可能需要批准待处理的 kubelet 服务 CSR。

使用以下步骤批准待处理的 CSR：

流程

1. 获取当前 CSR 列表：

```
$ oc get csr
```

输出示例

```
NAME        AGE    SIGNERNAME                            REQUESTOR
CONDITION
csr-2s94x   8m3s   kubernetes.io/kubelet-serving         system:node:<node_name>
Pending 1
csr-4bd6t   8m3s   kubernetes.io/kubelet-serving         system:node:<node_name>
```

```
Pending 2
csr-4hl85  13m  kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper  Pending 3
csr-zhhhp  3m8s  kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper  Pending 4
...
```

1 2 一个待处理的 kubelet 服务 CSR（用于用户置备的安装）。

3 4 一个待处理的 **node-bootstrapper** CSR。

2. 查看一个 CSR 的详细信息以验证其是否有效：

```
$ oc describe csr <csr_name> 1
```

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

3. 批准每个有效的 **node-bootstrapper** CSR：

```
$ oc adm certificate approve <csr_name>
```

4. 对于用户置备的安装，请批准每个有效的 kubelet 服务 CSR：

```
$ oc adm certificate approve <csr_name>
```