OpenShift Dedicated 4

存储

为 OpenShift Dedicated 集群配置存储

Last Updated: 2024-06-29

OpenShift Dedicated 4 存储

为 OpenShift Dedicated 集群配置存储

法律通告

摘要

本文档提供有关为 OpenShift Dedicated 集群设置存储的信息。

目录

第 1 章 OPENSHIFT DEDICATED 存储概述

OpenShift Dedicated 支持多种存储类型，包括内部存储和云供应商。您可以在 OpenShift Dedicated 集群中管理持久性和非持久性数据的容器存储。

1.1. OPENSHIFT DEDICATED 存储的常见术语表

该术语表定义了存储内容中使用的常用术语。

访问模式

卷访问模式描述了卷功能。您可以使用访问模式匹配持久性卷声明 (PVC) 和持久性卷 (PV)。以下是访问模式的示例：

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

配置映射

配置映射提供将配置数据注入 pod 的方法。您可以在类型为 **ConfigMap** 的卷中引用存储在配置映射中的数据。在 pod 中运行的应用程序可以使用这个数据。

Container Storage Interface (CSI)

在不同容器编配 (CO) 系统之间管理容器存储的 API 规格。

动态置备

该框架允许您按需创建存储卷，使集群管理员无需预置备持久性存储。

临时存储

Pod 和容器可能需要临时或过渡的本地存储才能进行操作。此临时存储的生命周期不会超过每个 pod 的生命周期，且此临时存储无法在 pod 间共享。

fsGroup

fsGroup 定义 pod 的文件系统组 ID。

hostPath

OpenShift Container Platform 集群中的 hostPath 卷将主机节点的文件系统中的文件或目录挂载到 pod 中。

KMS 密钥

Key Management Service (KMS) 可帮助您在不同服务间实现所需的数据加密级别。您可以使用 KMS 密钥加密、解密和重新加密数据。

本地卷

本地卷代表挂载的本地存储设备，如磁盘、分区或目录。

OpenShift Data Foundation

OpenShift Container Platform 支持的文件、块存储和对象存储的、内部或混合云的持久性存储供应商

持久性存储

Pod 和容器可能需要永久存储才能正常工作。OpenShift Dedicated 使用 Kubernetes 持久性卷 (PV) 框架来允许集群管理员为集群提供持久性存储。开发人员可以在不了解底层存储基础架构的情况下使用 PVC 来请求 PV 资源。

持久性卷 (PV)

OpenShift Dedicated 使用 Kubernetes 持久性卷 (PV) 框架来允许集群管理员为集群提供持久性存储。开发人员可以在不了解底层存储基础架构的情况下使用 PVC 来请求 PV 资源。

持久性卷声明 (PVC)

您可以使用 PVC 将 PersistentVolume 挂载到 Pod 中。您可以在不了解云环境的详情的情况下访问存储。

Pod

一个或多个带有共享资源（如卷和 IP 地址）的容器，在 OpenShift Dedicated 集群中运行。pod 是定义、部署和管理的最小计算单元。

重新声明策略

告知集群在卷被释放后使用什么操作。卷重新声明政策包括 **Retain**、**Recycle** 或 **Delete**。

基于角色的访问控制 (RBAC)

基于角色的访问控制 (RBAC) 是一种根据您机构中个别用户的角色监管计算机或网络资源的访问方法。

无状态应用程序

无状态应用是一种应用程序，它不会为与客户端进行下一个会话而保持在当前会话中生成的客户端数据。

有状态应用程序

有状态应用是一种应用程序，它会将数据保存到持久磁盘存储中。服务器、客户端和应用程序可以使用持久磁盘存储。您可以使用 OpenShift Dedicated 中的 **Statefulset** 对象来管理一组 Pod 的部署和扩展，并保证这些 Pod 的排序和唯一性。

静态置备

集群管理员创建多个 PV。PV 包含存储详情。PV 存在于 Kubernetes API 中，可供使用。

存储

OpenShift Dedicated 支持多种存储类型，包括内部存储和云供应商。您可以在 OpenShift Dedicated 集群中管理持久性和非持久性数据的容器存储。

Storage class

存储类为管理员提供了描述它们所提供的存储类的方法。不同的类可能会映射到服务质量级别、备份策略，以及由集群管理员决定的任意策略。

1.2. 存储类型

OpenShift Dedicated 存储广泛分为两类，即临时存储和持久存储。

1.2.1. 临时存储

Pod 和容器具有临时或临时性，面向无状态应用。临时存储可让管理员和开发人员更好地管理其某些操作的本地存储。如需有关临时存储概述、类型和管理的更多信息，请参阅了解临时存储。

1.2.2. 持久性存储

容器中部署的有状态应用需要持久存储。OpenShift Dedicated 使用名为持久性卷(PV)的预置备存储框架来允许集群管理员置备持久性存储。这些卷中的数据可能超过单个 pod 的生命周期。开发人员可以使用持久性卷声明(PVC)来请求存储要求。如需有关持久性存储概述、配置和生命周期的更多信息，请参阅了解持久性存储。

1.3. CONTAINER STORAGE INTERFACE (CSI)

CSI 是在不同容器编配(CO)系统中管理容器存储的 API 规范。您可以在容器原生环境中管理存储卷，而无需具体了解底层存储基础架构。使用 CSI 时，存储可在不同的容器编配系统中有效，无论您使用的存储供应商是什么。如需有关 CSI 的更多信息，请参阅使用容器存储接口(CSI)。

1.4. 动态置备

通过动态置备，您可以按需创建存储卷，使集群管理员无需预置备存储。有关动态置备的更多信息，请参阅动态置备。

第 2 章 了解临时存储

2.1. 概述

除了持久性存储外，Pod 和容器还需要临时或短暂的本地存储才能进行操作。此临时存储的生命周期不会超过每个 pod 的生命周期，且此临时存储无法在 pod 间共享。

Pod 使用临时本地存储进行涂销空间、缓存和日志。与缺少本地存储相关的问题包括：

- Pod 无法检测到有多少可用的本地存储。
- Pod 无法请求保证的本地存储。
- 本地存储是一个最佳资源。
- pod 可能会因为其他 pod 填充本地存储而被驱除。只有在足够的存储被重新声明前，不会接受这些新 pod。

与持久性卷不同，临时存储没有特定结构，它会被节点上运行的所有 pod 共享，并同时会被系统、容器运行时和 OpenShift Dedicated 使用。临时存储框架允许 Pod 指定其临时本地存储需求。它还允许 OpenShift Dedicated 在适当的时候调度 pod，并保护节点不受过度使用本地存储的影响。

虽然临时存储框架允许管理员和开发人员更好地管理本地存储，但 I/O 吞吐量和延迟不会直接生效。

2.2. 临时存储的类型

主分区中始终提供临时本地存储。创建主分区的基本方法有两种： root 和 runtime。

root
默认情况下，该分区包含 kubelet 根目录、**/var/lib/kubelet/** 和 **/var/log/** 目录。此分区可以在用户 Pod、OS 和 Kubernetes 系统守护进程间共享。Pod 可以通过 **EmptyDir** 卷、容器日志、镜像层和容器可写层来消耗这个分区。kubelet 管理这个分区的共享访问和隔离。这个分区是临时的，应用程序无法预期这个分区中的任何性能 SLA（如磁盘 IOPS）。

Runtime
这是一个可选分区，可用于 overlay 文件系统。OpenShift Dedicated 会尝试识别并提供共享访问以及这个分区的隔离。容器镜像层和可写入层存储在此处。如果 runtime 分区存在，则 **root** 分区不包含任何镜像层或者其它可写入的存储。

2.3. 临时存储管理

集群管理员可以通过设置配额在非终端状态的所有 Pod 中定义临时存储的限制范围，及临时存储请求数量，来管理项目中的临时存储。开发人员也可以在 Pod 和容器级别设置这个计算资源的请求和限值。

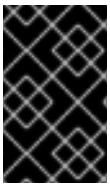
您可以通过指定请求和限值来管理本地临时存储。pod 中的每个容器可以指定以下内容：

- **spec.containers[].resources.limits.ephemeral-storage**
- **spec.containers[].resources.requests.ephemeral-storage**

2.3.1. 临时存储限制和请求单元

临时存储的限制和请求以字节数量来衡量。您可以使用以下后缀之一将存储表示为普通整数或固定点号：E、P、T、G、M、k。您还可以使用两的指数：Ei、Pi、Ti、Gi、Mi Ki。

例如，以下数量全部代表大约相同的值：128974848、129e6、129M 和 123Mi。



重要

每个字节数量的后缀都是区分大小写的。务必使用正确的问题单。使用区分大小写的 "M"，如在 "400M" 中使用的，将请求设置为 400MB。使用区分大小写的 "400Mi" 来请求 400 mebibytes。如果您为临时存储指定 "400m"，则存储请求仅为 0.4 字节。

2.3.2. 临时存储请求和限值示例

以下示例配置文件显示了一个具有两个容器的 pod：

- 每个容器请求 2GiB 本地临时存储。
- 每个容器限制为 4GiB 本地临时存储。
- 在 pod 级别，kubelet 通过添加该 pod 中所有容器的限制来达到总体 pod 存储限制。
  - 在本例中，pod 级别的总存储使用量是所有容器的磁盘用量总和，以及 pod 的 **emptyDir** 卷。
  - 因此，pod 的请求为 4GiB 本地临时存储，限值为 8GiB 本地临时存储。

使用配额和限值的临时存储配置示例

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        ephemeral-storage: "2Gi" (1)
      limits:
        ephemeral-storage: "4Gi" (2)
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  - name: log-aggregator
    image: images.my-company.example/log-aggregator:v6
    resources:
      requests:
        ephemeral-storage: "2Gi"
      limits:
        ephemeral-storage: "4Gi"
    volumeMounts:
    - name: ephemeral
      mountPath: "/tmp"
  volumes:
    - name: ephemeral
      emptyDir: {}
```

1. 容器请求本地临时存储。
2. 本地临时存储的容器限制。

2.3.3. 临时存储配置会影响 pod 调度和驱除

pod 规格中的设置会影响调度程序如何决定调度 pod 以及 kubelet 驱除 pod。

- 首先，调度程序确保调度容器的资源请求总和小于节点的容量。在这种情况下，只有在可用临时存储（可分配资源）超过 4GiB 时，pod 才能分配给节点。
- 其次，在容器级别上，因为第一个容器设置了资源限值，kubelet 驱除管理器会测量此容器的磁盘用量，并在此容器的存储使用量超过其限制时驱除 pod (4GiB)。如果总用量超过总体 pod 存储限制(8GiB)，kubelet 驱除管理器也会标记驱除 pod。

2.4. 监控临时存储

您可以使用 **/bin/df** 作为监控临时容器数据所在卷的临时存储使用情况的工具，即 **/var/lib/kubelet** 和 **/var/lib/containers**。如果集群管理员将 **/var/lib/containers** 放置在单独的磁盘上，则可以使用 **df** 命令来显示 **/var/lib/kubelet** 的可用空间。

要在 **/var/lib** 中显示已用和可用空间的信息，请输入以下命令：

```
$ df -h /var/lib
```

输出显示 **/var/lib** 中的临时存储使用情况：

输出示例

```
Filesystem  Size  Used Avail Use% Mounted on
/dev/disk/by-partuuid/4cd1448a-01    69G   32G   34G  49% /
```

第 3 章 了解持久性存储

3.1. 持久性存储概述

管理存储与管理计算资源不同。OpenShift Dedicated 使用 Kubernetes 持久性卷 (PV) 框架来允许集群管理员为集群提供持久性存储。开发者可以使用持久性卷声明 (PVC) 来请求 PV 资源而无需具体了解底层存储基础架构。

PVC 是特定于一个项目的，开发人员可创建并使用它作为使用 PV 的方法。PV 资源本身并不特定于某一个项目; 它们可以在整个 OpenShift Dedicated 集群间共享，并可以被任何项目使用。在 PV 绑定到 PVC 后，就不会将 PV 绑定到额外的 PVC。这意味着这个绑定 PV 被限制在一个命名空间（绑定的项目）中。

PV 由 **PersistentVolume** API 对象定义，它代表了集群中现有存储的片段，这些存储可以由集群管理员静态置备，也可以使用 **StorageClass** 对象动态置备。它与一个节点一样，是一个集群资源。

PV 是卷插件，与 **Volumes** 资源类似，但PV 的生命周期独立于任何使用它的 pod。PV 对象获取具体存储（NFS、iSCSI 或者特定 cloud-provider 的存储系统）的实现详情。

重要

存储的高可用性功能由底层的存储架构提供。

PVC 由 **PersistentVolumeClaim** API 项定义，它代表了开发人员对存储的一个请求。它与一个 pod 类似，pod 会消耗节点资源， PVC 消耗 PV 资源。例如：pod 可以请求特定级别的资源，比如 CPU 和内存，而 PVC 可以请求特定的存储容量和访问模式。例如：它们可以被加载为“只允许加载一次，可读写”，或“可以加载多次，只读”。

3.2. 卷和声明的生命周期

PV 是集群中的资源。PVC 是对这些资源的请求，也是对该资源的声明检查。PV 和 PVC 之间的交互有以下生命周期。

3.2.1. 置备存储

根据 PVC 中定义的开发人员的请求，集群管理员配置一个或者多个动态置备程序用来置备存储及一个匹配的 PV。

3.2.2. 绑定声明

当您创建 PVC 时，您会要求特定的存储量，指定所需的访问模式，并创建一个存储类来描述和分类存储。master 中的控制循环会随时检查是否有新的 PVC，并把新的 PVC 与一个适当的 PV 进行绑定。如果没有适当的 PV，则存储类的置备程序会创建一个适当的 PV。

所有 PV 的大小可能会超过 PVC 的大小。这在手动置备 PV 时尤为如此。要最小化超额，OpenShift Dedicated 将会把 PVC 绑定到匹配所有其他标准的最小 PV。

如果匹配的卷不存在，或者相关的置备程序无法创建所需的存储，则请求将会处于未绑定的状态。当出现了匹配的卷时，相应的声明就会与其绑定。例如：在一个集群中有多个手动置备的 50Gi 卷。它们无法和一个请求 100Gi 的 PVC 相匹配。当在这个集群中添加了一个 100Gi PV 时， PVC 就可以和这个 PV 绑定。

3.2.3. 使用 pod 和声明的 PV

pod 使用声明（claim）作为卷。集群通过检查声明来找到绑定的卷，并为 pod 挂载相应的卷。对于那些支持多个访问模式的卷，您必须指定作为 pod 中的卷需要使用哪种模式。

一旦您的声明被绑定后，被绑定的 PV 就会专属于您，直到您不再需要它。您可以通过在 pod 的 volumes 定义中包括 **persistentVolumeClaim** 来调度 pod 并访问声明的 PV。

注意

如果将具有高文件数的持久性卷附加到 pod，则这些 pod 可能会失败，或者可能需要很长时间才能启动。如需更多信息，请参阅在 OpenShift 中使用具有高文件计数的持久性卷时，为什么 pod 无法启动或占用大量时间来实现"Ready"状态？

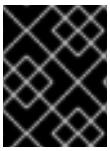
3.2.4. 释放持久性卷

当不再需要使用一个卷时，您可以从 API 中删除 PVC 对象，这样相应的资源就可以被重新声明。当声明被删除后，这个卷就被认为是已被释放，但它还不可以被另一个声明使用。这是因为之前声明者的数据仍然还保留在卷中，这些数据必须根据相关政策进行处理。

3.2.5. 持久性卷的重新声明策略

持久性卷重新声明（reclaim）策略指定了在卷被释放后集群可以如何使用它。卷重新声明政策包括 **Retain**、**Recycle** 或 **Delete**。

- **Retain** 策略可为那些支持它的卷插件手动重新声明资源。
- **Recycle** 策略在从其请求中释放后，将卷重新放入到未绑定的持久性卷池中。



重要

在 OpenShift Dedicated 4 中弃用了 **Recycle** 重新声明策略。我们推荐使用动态置备功能。

- **Delete** 策略删除 OpenShift Dedicated 以及外部基础架构（如 Amazon Elastic Block Store (Amazon EBS) 或 VMware vSphere）中相关的存储资源中的 **PersistentVolume**。

注意

动态置备的卷总是被删除。

3.2.6. 手动重新声明持久性卷

删除持久性卷生命（PVC）后，持久性卷（PV）仍然存在，并被视为"released（释放）"。但是，由于之前声明的数据保留在卷中，所以无法再使用 PV。

流程

要以集群管理员的身份手动重新声明 PV:

1. 删除 PV。

```
$ oc delete pv <pv-name>
```

外部基础架构中关联的存储资产（如 AWS EBS 或 GCE PD 卷）在 PV 被删除后仍然存在。

2. 清理相关存储资产中的数据。

3. 删除关联的存储资产。另外，若要重复使用同一存储资产，请使用存储资产定义创建新 PV。

重新声明的 PV 现在可供另一个 PVC 使用。

3.2.7. 更改持久性卷的重新声明策略

更改持久性卷的重新声明策略：

1. 列出集群中的持久性卷：

```
$ oc get pv
```

输出示例

```
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS
CLAIM            STORAGECLASS     REASON    AGE
 pvc-b6efd8da-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim1     manual                   10s
 pvc-b95650f8-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim2     manual                   6s
 pvc-bb3ca71d-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim3     manual                   3s
```

2. 选择一个持久性卷并更改其重新声明策略：

```
$ oc patch pv <your-pv-name> -p '{"spec":{"persistentVolumeReclaimPolicy":"Retain"}}'
```

3. 验证您选择的持久性卷是否具有正确的策略：

```
$ oc get pv
```

输出示例

```
NAME                                       CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS
CLAIM            STORAGECLASS     REASON    AGE
 pvc-b6efd8da-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim1     manual                   10s
 pvc-b95650f8-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Delete          Bound
default/claim2     manual                   6s
 pvc-bb3ca71d-b7b5-11e6-9d58-0ed433a7dd94   4Gi        RWO           Retain          Bound
default/claim3     manual                   3s
```

在前面的输出中，绑定到声明 **default/claim3** 的卷现在具有 **Retain** 重新声明策略。当用户删除声明 **default/claim3** 时，这个卷不会被自动删除。

3.3. 持久性卷（PV）

每个 PV 都会包括一个 **spec** 和 **status**，它们分别代表卷的规格和状态，例如：

PersistentVolume 对象定义示例

■

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001 (1)
spec:
  capacity:
    storage: 5Gi (2)
  accessModes:
    - ReadWriteOnce (3)
  persistentVolumeReclaimPolicy: Retain (4)
  ...
status:
  ...
```

1. 持久性卷的名称。
2. 卷可以使用的存储容量。
3. 访问模式，用来指定读写权限及挂载权限。
4. 重新声明策略，指定在资源被释放后如何处理它。

3.3.1. PV 类型

OpenShift Dedicated 支持以下持久性卷插件：

- AWS Elastic Block Store (EBS)
- GCP 持久性磁盘
- GCP Filestore

3.3.2. 容量

一般情况下，一个持久性卷（PV）有特定的存储容量。这可以通过使用 PV 的 **capacity** 属性来设置。

目前，存储容量是唯一可以设置或请求的资源。以后可能会包括 IOPS 、 throughput 等属性。

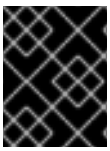
3.3.3. 访问模式

一个持久性卷可以以资源供应商支持的任何方式挂载到一个主机上。不同的供应商具有不同的功能，每个 PV 的访问模式可以被设置为特定卷支持的特定模式。例如：NFS 可以支持多个读写客户端，但一个特定的 NFS PV 可能会以只读方式导出。每个 PV 都有自己一组访问模式来描述指定的 PV 功能。

声明会与有类似访问模式的卷匹配。用来进行匹配的标准只包括访问模式和大小。声明的访问模式代表一个请求。比声明要求的条件更多的资源可能会匹配，而比要求的条件更少的资源则不会被匹配。例如：如果一个声明请求 RWO，但唯一可用卷是一个 NFS PV（RWO+ROX+RWX），则该声明与这个 NFS 相匹配，因为它支持 RWO。

系统会首先尝试直接匹配。卷的模式必须与您的请求匹配，或包含更多模式。大小必须大于或等于预期值。如果两个卷类型（如 NFS 和 iSCSI）有相同的访问模式，则一个要求这个模式的声明可能会与其中任何一个进行匹配。不同的卷类型之间没有匹配顺序，在同时匹配时也无法选择特定的一个卷类型。

所有有相同模式的卷都被分组，然后按大小（由小到大）进行排序。绑定程序会获取具有匹配模式的组群，并按容量顺序进行查找，直到找到一个大小匹配的项。。



重要

卷访问模式描述了卷功能。它们不会被强制限制。存储供应商会最终负责处理由于资源使用无效导致的运行时错误。供应商中的错误会在运行时作为挂载错误显示。

下表列出了访问模式：

表 3.1. 访问模式

访问模式	CLI 缩写	描述
ReadWriteOnce	**RWO**	卷只可以被一个节点以读写模式挂载。
ReadWriteOncePod [1]	**RWOP**	卷可以被一个单独的节点上的单个 pod 以读写模式挂载。

1. RWOP 使用 SELinux 挂载功能。此功能依赖于驱动程序，在 ODF、AWS EBS、Azure Disk、GCP PD、IBM VPC Block、Cinder 和 vSphere 中默认启用。如需第三方驱动程序，请联系您的存储厂商。

表 3.2. 支持的持久性卷访问模式

卷插件	ReadWriteOnce [1]	ReadWriteOncePod	ReadOnlyMany	ReadWriteMany
AWS EBS [2]	▮	▮		
AWS EFS	▮	▮	▮	▮
GCP 持久性磁盘	▮	▮		
GCP Filestore	▮	▮	▮	▮
LVM 存储	▮	▮		

1. ReadWriteOnce（RWO）卷不能挂载到多个节点上。如果节点失败，系统不允许将附加的 RWO 卷挂载到新节点上，因为它已经分配给了故障节点。如果因此遇到多附件错误消息，请强制在关闭或崩溃的节点上删除 pod，以避免关键工作负载中的数据丢失，例如在附加动态持久性卷时。

2. 为依赖 AWS EBS 的 pod 使用重新创建的部署策略。

3. 只有原始块卷支持 ReadWriteMany (RWX) 访问模式用于光纤通道和 iSCSI。如需更多信息，请参阅 "Block volume support"。

3.3.4. 阶段

卷可以处于以下几个阶段：

表 3.3. 卷阶段

阶段	描述
Available	可用资源，还未绑定到任何声明。
Bound	卷已绑定到一个声明。
Released	以前使用这个卷的声明已被删除，但该资源还没有被集群重新声明。
Failed	卷的自动重新声明失败。

您可以运行以下命令来查看绑定到 PV 的 PVC 名称：

```
$ oc get pv <pv-claim>
```

3.3.4.1. 挂载选项

您可以使用属性 **mountOptions** 在挂载 PV 时指定挂载选项。

例如：

挂载选项示例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  mountOptions: 1
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
  persistentVolumeReclaimPolicy: Retain
  claimRef:
    name: claim1
    namespace: default
```

1 在将 PV 挂载到磁盘时使用指定的挂载选项。

以下 PV 类型支持挂载选项：

- AWS Elastic Block Store (EBS)
- GCE Persistent Disk

3.4. 持久性卷声明 (PVC)

每个 **PersistentVolumeClaim** 对象都会包括一个 **spec** 和 **status**，它们分别代表了声明的规格和状态。例如：

PersistentVolumeClaim 对象定义示例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim 1
spec:
  accessModes:
    - ReadWriteOnce 2
  resources:
    requests:
      storage: 8Gi 3
  storageClassName: gold 4
status:
  ...
```

1. PVC 的名称。
2. 访问模式，用来指定读写权限及挂载权限。
3. PVC 可用的存储量。
4. 声明所需的 **StorageClass** 的名称。

3.4.1. 存储类

另外，通过在 **storageClassName** 属性中指定存储类的名称，声明可以请求一个特定的存储类。只有具有请求的类的 PV（**storageClassName**的值与 PVC 中的值相同）才会与 PVC 绑定。集群管理员可配置动态置备程序为一个或多个存储类提供服务。集群管理员可根据需要创建与 PVC 的规格匹配的 PV。

重要

根据使用的平台，Cluster Storage Operator 可能会安装一个默认的存储类。此存储类由 Operator 拥有和控制。不能在定义注解和标签之外将其删除或修改。如果需要实现不同的行为，则必须定义自定义存储类。

集群管理员也可以为所有 PVC 设置默认存储类。当配置了默认存储类时， PVC 必须明确要求将存储类 **StorageClass** 或 **storageClassName**设为 **""**， 以便绑定到没有存储类的 PV。

注意

如果一个以上的存储类被标记为默认，则只能在 **storageClassName** 被显式指定时才能创建 PVC。因此，应只有一个存储类被设置为默认值。

3.4.2. 访问模式

声明在请求带有特定访问权限的存储时，使用与卷相同的格式。

3.4.3. Resources

象 pod 一样，声明可以请求具体数量的资源。在这种情况下，请求用于存储。同样的资源模型适用于卷和声明。

3.4.4. 声明作为卷

pod 通过将声明作为卷来访问存储。在使用声明时，声明需要和 pod 位于同一个命名空间。集群在 pod 的命名空间中找到声明，并使用它来使用这个声明后台的**PersistentVolume**。卷被挂载到主机和 pod 中，例如：

挂载卷到主机和 pod 示例

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
      - mountPath: "/var/www/html" 1
        name: mypd 2
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim 3
```

1. 在 pod 中挂载卷的路径。
2. 要挂载的卷的名称。不要挂载到容器 root、/ 或主机和容器中相同的任何路径。如果容器有足够权限，可能会损坏您的主机系统（如主机的 **/dev/pts** 文件）。使用 **/host** 挂载主机是安全的。
3. 要使用的 PVC 名称（存在于同一命名空间中）。

3.5. 块卷支持

OpenShift Dedicated 可以静态置备原始块卷。这些卷没有文件系统。对于可以直接写入磁盘或者实现其自己的存储服务的应用程序来说，使用它可以获得性能优势。

原始块卷可以通过在 PV 和 PVC 规格中指定 **volumeMode: Block**来置备。

重要

使用原始块卷的 pod 需要配置为允许特权容器。

下表显示了哪些卷插件支持块卷。

表 3.4. 块卷支持

卷插件	手动置备	动态置备	完全支持
Amazon Elastic Block Store (Amazon EBS)	▮	▮	▮
Amazon Elastic File Storage (Amazon EFS)			
GCP	▮	▮	▮
LVM 存储	▮	▮	▮

3.5.1. 块卷示例

PV 示例

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: block-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  volumeMode: Block 1
  persistentVolumeReclaimPolicy: Retain
  fc:
    targetWWNs: ["50060e801049cfd1"]
    lun: 0
    readOnly: false
```

1 需要把 **volumeMode** 设置为 **Block** 来代表这个 PV 是一个原始块卷。

PVC 示例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Block 1
  resources:
    requests:
      storage: 10Gi
```

1 需要把 **volumeMode** 设置为 **Block** 来代表请求一个原始块 PVC。

Pod 规格示例

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-block-volume
spec:
  containers:
    - name: fc-container
      image: fedora:26
      command: ["/bin/sh", "-c"]
      args: [ "tail -f /dev/null" ]
      volumeDevices:  1
        - name: data
          devicePath: /dev/xvda  2
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: block-pvc  3
```

1. 对于块设备，使用 **VolumeDevices**而不是 **volumeMounts**。只有 **persistentVolumeClaim** 源可以和原始块卷一起使用。
2. 使用 **devicePath** 而不是 **mountPath** 来代表到原始块映射到系统的物理设备的路径。
3. 卷源必须是 **persistentVolumeClaim** 类型，且必须与期望的 PVC 的名称匹配。

表 3.5. volumeMode 接受的值

值	默认
Filesystem	是
Block	否

表 3.6. 块卷的绑定方案

PV volumeMode	PVC volumeMode	绑定结果
Filesystem	Filesystem	绑定
Unspecified	Unspecified	绑定
Filesystem	Unspecified	绑定
Unspecified	Filesystem	绑定
Block	Block	绑定

PV volumeMode	PVC volumeMode	绑定结果
Unspecified	Block	无绑定
Block	Unspecified	无绑定
Filesystem	Block	无绑定
Block	Filesystem	无绑定

重要

未指定值时将使用默认值 **Filesystem**。

3.6. 使用 FSGROUP 减少 POD 超时

如果存储卷包含很多文件（1,000,000 或更多），您可能会遇到 pod 超时问题。

这是因为，在默认情况下，OpenShift Dedicated 会递归更改每个卷内容的所有权和权限，以便在挂载卷时与 pod 的 **securityContext** 中指定的 **fsGroup** 匹配。对于大型卷，检查和更改所有权和权限可能会非常耗时，从而会减慢 pod 启动的速度。您可以使用 **securityContext** 中的 **fsGroupChangePolicy** 字段来控制 OpenShift Dedicated 检查和管理卷的所有权和权限的方式。

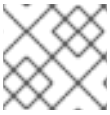
fsGroupChangePolicy 定义在 pod 中公开卷之前更改卷的所有权和权限的行为。此字段仅适用于支持 **fsGroup**- 控制的所有权和权限。此字段有两个可能的值：

- **OnRootMismatch** ：仅当 root 目录的权限和所有权与卷的预期权限不匹配时才会更改权限和所有权。这有助于缩短更改卷的所有权和权限所需的时间，以减少 pod 超时。
- **Always** ：当卷被挂载时，始终更改卷的权限和所有权。

fsGroupChangePolicy 示例

```
securityContext:
  runAsUser: 1000
  runAsGroup: 3000
  fsGroup: 2000
  fsGroupChangePolicy: "OnRootMismatch" 1
  ...
```

1 **OnRootMismatch** 指定跳过递归权限更改，这有助于避免 pod 超时问题。



注意

fsGroupChangePolicyfield 对临时卷类型没有影响，如 secret、configMap 和 emptydir。

第 4 章 配置持久性存储

4.1. 使用 AWS ELASTIC BLOCK STORE 的持久性存储

OpenShift Dedicated 集群预构建了四个存储类，它们使用 Amazon Elastic Block Store (Amazon EBS) 卷。这些存储类可供使用，并假设您对 Kubernetes 和 AWS 有一定的了解。

以下是四个预构建的存储类：

名称	Provisioner
gp2	kubernetes.io/aws-ebs
gp2-csi	ebs.csi.aws.com
gp3 （默认）	kubernetes.io/aws-ebs
gp3-csi	ebs.csi.aws.com

gp3 存储类被设置为默认存储类，但您可以选择任何存储类作为默认存储类。

Kubernetes 持久性卷框架允许管理员提供带有持久性存储的集群，并使用户可以在不了解底层存储架构的情况下请求这些资源。您可以动态置备 Amazon EBS 卷。持久性卷不与某个特定项目或命名空间相关联，它们可以在 OpenShift Dedicated 集群间共享。持久性卷声明是针对某个项目或者命名空间的，相应的用户可请求它。您可以定义 KMS 密钥来加密 AWS 上的 container-persistent 卷。默认情况下，使用 OpenShift Dedicated 版本 4.10 及之后的版本创建的集群使用 gp3 存储和 AWS EBS CSI 驱动程序 。

重要

存储的高可用性功能由底层存储供应商实现。

4.1.1. 创建 EBS 存储类

存储类用于区分和划分存储级别和使用。通过定义存储类，用户可以获得动态置备的持久性卷。

4.1.2. 创建持久性卷声明

先决条件

当存储可以被挂载为 OpenShift Dedicated 中的卷之前，它必须已存在于底层的存储系统中。

流程

1. 在 OpenShift Dedicated 控制台中，点 **Storage → Persistent Volume Claims**。
2. 在持久性卷声明概述页中，点 **Create Persistent Volume Claim**。
3. 在出现的页面中定义所需选项。
    a. 从下拉菜单中选择之前创建的存储类。

b. 输入存储声明的唯一名称。

c. 选择访问模式。此选择决定了存储声明的读写访问权限。

d. 定义存储声明的大小。

4. 点击 **Create** 创建持久性卷声明，并生成一个持久性卷。

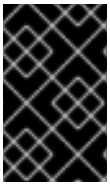
4.1.3. 卷格式

在 OpenShift Dedicated 挂载卷并将其传递给容器之前，它会检查卷是否包含由持久性卷定义中的 **fsType** 参数指定的文件系统。如果没有使用文件系统格式化该设备，该设备中的所有数据都会被删除，并使用指定的文件系统自动格式化该设备。

此验证可让您将未格式化的 AWS 卷用作持久性卷，因为 OpenShift Dedicated 在首次使用前会进行格式化。

4.1.4. 一个节点上的 EBS 卷的最大数目

默认情况下，OpenShift Dedicated 最多支持把 39 个 EBS 卷附加到一个节点。这个限制与 AWS 卷限制一致。卷限制取决于实例类型。



重要

作为集群管理员，您必须使用树内或 Container Storage Interface（CSI）卷及其相应的存储类，但不得同时使用这两个卷类型。对于 in-tree 和 CSI 卷，最大附加的 EBS 卷数量会单独计算，因此每种类型您都最多可以有 39 个 EBS 卷。

有关访问额外存储选项（如卷快照）的详情，请参考 AWS Elastic Block Store CSI Driver Operator 。

4.1.5. 使用 KMS 密钥在 AWS 上加密容器持久性卷

在部署到 AWS 时，定义在 AWS 上加密容器持久性卷的 KMS 密钥很有用。

先决条件

- 底层基础架构必须包含存储。
- 您必须在 AWS 上创建客户 KMS 密钥。

流程

1. 创建存储类：

```
$ cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class-name> 1
parameters:
  fsType: ext4 2
  encrypted: "true"
  kmsKeyId: keyvalue 3
provisioner: ebs.csi.aws.com
```

```
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
EOF
```

1. 指定存储类的名称。
2. 在置备的卷中创建的文件系统。
3. 指定加密 container-persistent 卷时要使用的密钥的完整 Amazon 资源名称 (ARN)。如果没有提供任何密钥，但 **encrypted** 字段被设置为 **true**，则使用默认的 KMS 密钥。请参阅 AWS 文档中的查找 AWS 的密钥 ID 和密钥 ARN。

2. 使用指定 KMS 密钥的存储类创建 PVC：

```
$ cat << EOF | oc create -f -
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  storageClassName: <storage-class-name>
  resources:
    requests:
      storage: 1Gi
EOF
```

3. 创建工作负载容器以使用 PVC：

```
$ cat << EOF | oc create -f -
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: httpd
      image: quay.io/centos7/httpd-24-centos7
      ports:
        - containerPort: 80
      volumeMounts:
        - mountPath: /mnt/storage
          name: data
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: mypvc
EOF
```

4.1.6. 其他资源

- 有关访问额外存储选项的信息，如卷快照，请参阅 AWS Elastic Block Store CSI Driver Operator，这些内容无法在树状卷插件中使用。

4.2. 使用 GCE PERSISTENT DISK 的持久性存储

OpenShift Dedicated 支持 GCE Persistent Disk 卷(gcePD)。您可以使用 GCE 为 OpenShift Dedicated 集群置备持久性存储。我们假设您对 Kubernetes 和 GCE 有一定的了解。

Kubernetes 持久性卷框架允许管理员提供带有持久性存储的集群，并使用户可以在不了解底层存储架构的情况下请求这些资源。

GCE Persistent Disk 卷可以动态部署。

持久性卷不与某个特定项目或命名空间相关联，它们可以在 OpenShift Dedicated 集群间共享。持久性卷声明是针对某个项目或者命名空间的，相应的用户可请求它。

重要

存储的高可用性功能由底层的存储架构提供。

其他资源

- GCE Persistent Disk

4.2.1. 创建 GCE 存储类

存储类用于区分和划分存储级别和使用。通过定义存储类，用户可以获得动态置备的持久性卷。

4.2.2. 创建持久性卷声明

先决条件

当存储可以被挂载为 OpenShift Dedicated 中的卷之前，它必须已存在于底层的存储系统中。

流程

1. 在 OpenShift Dedicated 控制台中，点 **Storage → Persistent Volume Claims**。
2. 在持久性卷声明概述页中，点 **Create Persistent Volume Claim**。
3. 在出现的页面中定义所需选项。
   a. 从下拉菜单中选择之前创建的存储类。
   b. 输入存储声明的唯一名称。
   c. 选择访问模式。此选择决定了存储声明的读写访问权限。
   d. 定义存储声明的大小。
4. 点击 **Create** 创建持久性卷声明，并生成一个持久性卷。

4.2.3. 卷格式

在 OpenShift Dedicated 挂载卷并将其传递给容器之前，它会检查卷是否包含由持久性卷定义中的 **fsType** 参数指定的文件系统。如果没有使用文件系统格式化该设备，该设备中的所有数据都会被删除，并使用指定的文件系统自动格式化该设备。

此验证可让您将未格式化的 GCE 卷用作持久性卷，因为 OpenShift Dedicated 在首次使用前会对其进行格式化。

第 5 章 使用 CONTAINER STORAGE INTERFACE (CSI)

5.1. 配置 CSI 卷

容器存储接口 (CSI) 允许 OpenShift Dedicated 使用支持 CSI 接口的存储后端提供的持久性存储。

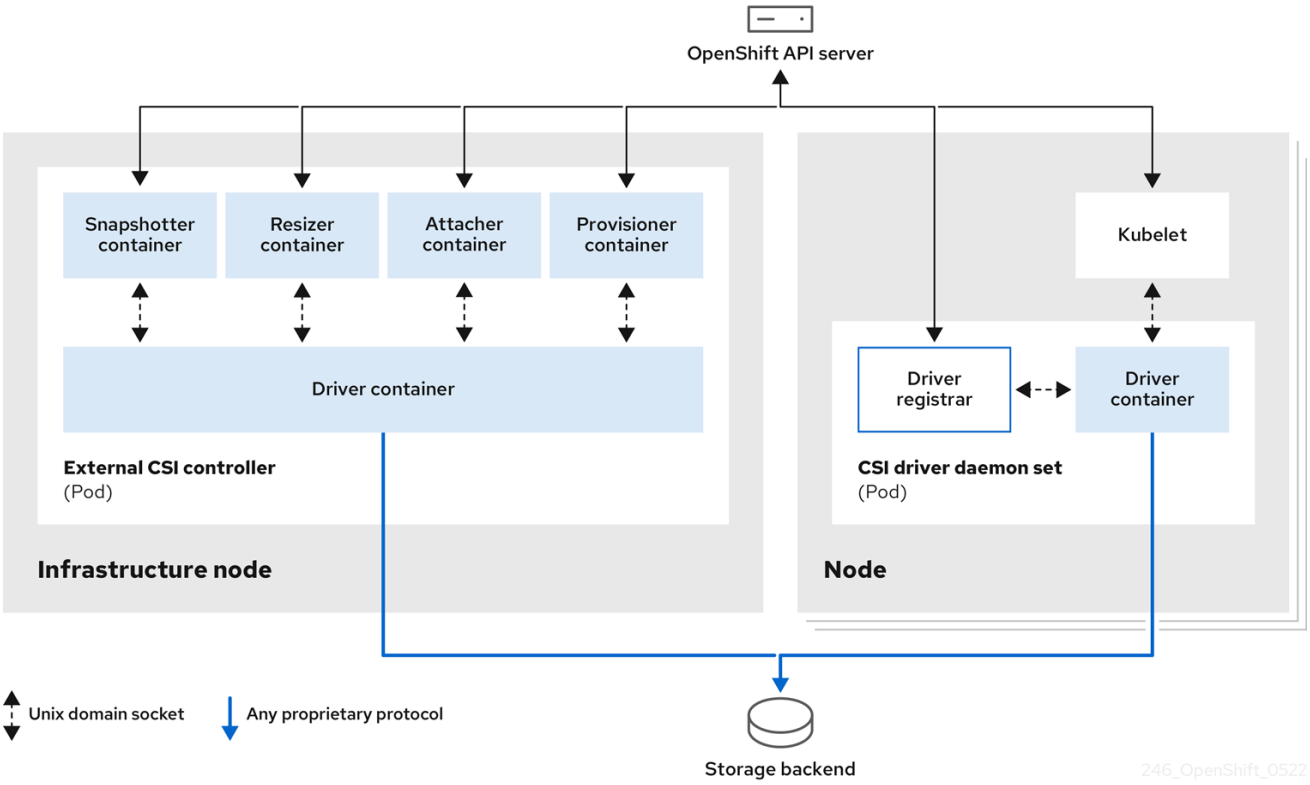
注意

OpenShift Dedicated 4 支持 CSI 规范 版本 1.6.0。

5.1.1. CSI 架构

CSI 驱动程序通常由容器镜像提供。这些容器不知道其运行的 OpenShift Dedicated。要在 OpenShift Dedicated 中使用与 CSI 兼容的存储后端，集群管理员必须部署几个组件，作为 OpenShift Dedicated 和存储驱动程序间的桥接。

下图显示了在 OpenShift Dedicated 集群中运行的组件的高级概述。



对于不同的存储后端，可以运行多个 CSI 驱动程序。每个驱动程序需要其自身的外部控制器部署，以及带驱动程序和 CSI 注册器的守护进程集。

5.1.1.1. 外部 CSI 控制器

外部 CSI 控制器是一个部署，它部署带有以下五个容器的一个或多个 pod：

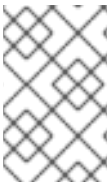
- snapshotter 容器监视 **VolumeSnapshot** 和 **VolumeSnapshotContent** 对象，并负责创建和删除 **VolumeSnapshotContent** 对象。

- resizer 容器是一个 sidecar 容器，它会在 PersistentVolumeClaim 对象中监视 **PersistentVolumeClaim** 更新，并在对 **PersistentVolumeClaim** 对象请求更多存储时针对 CSI 端点触发 **ControllerExpandVolume** 操作。
- 一个外部 CSI attacher 容器，它会将 OpenShift Dedicated 的 **attach** 和 **detach** 调用转换为相关的 CSI 驱动程序的 **ControllerPublish** 和 **ControllerUnpublish** 调用。
- 一个外部 CSI 置备程序容器，它可将 OpenShift Dedicated 的 **provision** 和 **delete** 调用转换为相应的 CSI 驱动程序的 **CreateVolume** 和 **DeleteVolume** 调用。
- 一个 CSI 驱动程序容器。

CSI attacher 和 CSI provisioner 容器使用 UNIX 域套接字与 CSI 驱动程序容器进行交互，确保没有 CSI 通讯会离开 pod。从 pod 以外无法访问 CSI 驱动程序。

注意

attach、 **detach**、 **provision**和 **delete** 操作通常需要 CSI 驱动程序在存储后端使用凭证。在 infrastructure 节点上运行 CSI controller pod，因此即使在一个计算节点上发生严重的安全破坏时，凭据也不会暴露给用户进程。



注意

当不支持第三方的 **attach** 或 **detach** 操作时，还需要为 CSI 驱动程序运行外部的附加器。外部附加器不会向 CSI 驱动程序发出任何 **ControllerPublish** 或 **ControllerUnpublish** 操作。然而，它仍必须运行方可实现所需的 OpenShift Dedicated attachment API。

5.1.1.2. CSI 驱动程序守护进程集

CSI 驱动程序守护进程集在每个节点上运行一个 pod，它允许 OpenShift Dedicated 挂载 CSI 驱动程序提供的存储，并使用它作为持久性卷 (PV) 的用户负载 (pod) 。安装了 CSI 驱动程序的 pod 包含以下容器：

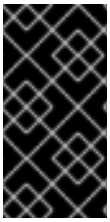
- 一个 CSI 驱动程序注册器，它会在节点上运行的 **openshift-node** 服务中注册 CSI 驱动程序。在节点上运行的 **openshift-node** 进程然后使用节点上可用的 UNIX 域套接字直接连接到 CSI 驱动程序。
- 一个 CSI 驱动程序。

在节点上部署的 CSI 驱动程序应该在存储后端中拥有尽量少的凭证。OpenShift Dedicated 只使用节点插件的 CSI 调用集合，如**NodePublish/NodeUnpublish** 和 **NodeStage/NodeUnstage**（如果这些调用已被实现）。

5.1.2. OpenShift Dedicated 支持的 CSI 驱动程序

OpenShift Dedicated 默认安装某些 CSI 驱动程序，为用户提供树状卷插件无法进行的存储选项。

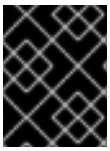
要创建挂载到这些支持的存储资产中的 CSI 置备的持久性卷，OpenShift Dedicated 会默认安装必要的 CSI 驱动程序 Operator、CSI 驱动程序和所需的存储类。如需有关 Operator 和驱动程序的默认命名空间的更多信息，请参阅特定 CSI Driver Operator 的文档。



重要

默认情况下，AWS EFS 和 GCP Filestore CSI 驱动程序不会被安装，必须手动安装。有关安装 AWS EFS CSI 驱动程序的步骤，请参阅设置 AWS Elastic File Service CSI Driver Operator。有关安装 GCP Filestore CSI 驱动程序的步骤，请参阅 Google Compute Platform Filestore CSI Driver Operator。

下表描述了 OpenShift Dedicated 支持的 CSI 驱动程序及其支持的 CSI 功能，如卷快照和调整大小。



重要

如果下表中没有列出您的 CSI 驱动程序，您必须遵循 CSI 存储厂商提供的安装说明来使用其支持的 CSI 功能。

表 5.1. OpenShift Dedicated 中支持的 CSI 驱动程序和功能

CSI 驱动程序	CSI 卷快照	CSI 克隆	CSI 调整大小	内联临时卷
AWS EBS	▮		▮	
AWS EFS				
Google Compute Platform (GCP) 持久磁盘 (PD)	▮	▮	▮	
GCP Filestore	▮		▮	
LVM 存储	▮	▮	▮	

5.1.3. 动态置备

动态置备持久性存储取决于 CSI 驱动程序和底层存储后端的功能。CSI 驱动的供应商应该提供了在 OpenShift Dedicated 中创建存储类及进行配置的参数文档。

创建的存储类可以被配置为启用动态置备。

流程

- 创建一个默认存储类，以保证所有不需要特殊存储类的 PVC 由安装的 CSI 驱动程序来置备。

```
# oc create -f - << EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class> 1
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
```

```
provisioner: <provisioner-name> 2
parameters:
EOF
```

1 要创建的存储类的名称。

2 已安装的 CSI 驱动程序名称。

5.1.4. 使用 CSI 驱动程序示例

以下示例在没有对该模板进行任何修改的情况下安装了一个默认的 MySQL 模板，。

先决条件

- CSI 驱动程序已被部署。
- 为动态置备创建了存储类。

流程

- 创建 MySQL 模板：

```
# oc new-app mysql-persistent
```

输出示例

```
--> Deploying template "openshift/mysql-persistent" to project default
...
```

```
# oc get pvc
```

输出示例

```
NAME             STATUS    VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
mysql            Bound     kubernetes-dynamic-pv-3271ffcb4e1811e8   1Gi
RWO            cinder         3s
```

5.2. 管理默认存储类

5.2.1. 概述

管理默认存储类可让您完成几个不同的目标：

- 禁用动态置备来强制静态置备。
- 当您有其他首选存储类时，防止存储操作器重新创建初始默认存储类。
- 重命名或更改默认存储类

要实现这些目标，您可以更改 **ClusterCSIDriver** 对象中的 **spec.storageClassState** 字段的设置。此字段可能的设置有：

- **Managed**: （默认）Container Storage Interface (CSI) Operator 会主动管理其默认存储类，集群管理员对默认存储类进行的大多数手动更改会被删除，如果您试图手动默认存储类，则它们会被持续重新创建。
- **Unmanaged** ：您可以修改默认存储类。CSI Operator 不会主动管理存储类，因此它不会协调它自动创建的默认存储类。
- **Removed**: CSI operator 删除默认存储类。

5.2.2. 使用 Web 控制台管理默认存储类

先决条件

- 访问 OpenShift Dedicated Web 控制台。
- 使用 cluster-admin 权限访问集群。

流程

使用 Web 控制台管理默认存储类：

1. 登录到 web 控制台。
2. 点 **Administration** > **CustomResourceDefinitions**。
3. 在 **CustomResourceDefinitions** 页面中，键入 **clustercsidriver** 来查找 **ClusterCSIDriver** 对象。
4. 点 **ClusterCSIDriver**，然后点 **Instances** 选项卡。
5. 点所需实例的名称，然后点 **YAML** 选项卡。
6. 添加 **spec.storageClassState** 字段，值为 **Managed**、**Unmanaged** 或 **Removed**。

   示例

```
...
spec:
  driverConfig:
    driverType: ''
  logLevel: Normal
  managementState: Managed
  observedConfig: null
  operatorLogLevel: Normal
  storageClassState: Unmanaged (1)
...
```

   (1) **spec.storageClassState** 字段设置为 "Unmanaged"

7. 点击 **Save**。

5.2.3. 使用 CLI 管理默认存储类

先决条件

- 使用 cluster-admin 权限访问集群。

流程

要使用 CLI 管理存储类，请运行以下命令：

```
oc patch clustercsidriver $DRIVERNAME --type=merge -p "{\"spec\":
{\"storageClassState\":\"${STATE}\"}}" 1
```

1 其中 **${STATE}** 为 "Removed" 或 "Managed" 或 "Unmanaged"。

其中 **$DRIVERNAME** 是置备程序名称。您可以通过运行命令 **oc get sc** 来查找置备程序名称。

5.2.4. 缺少或多个默认存储类

5.2.4.1. 多个默认存储类

如果您将非默认存储类标记为默认存储类，且没有取消设置现有的默认存储类，或者在默认存储类已存在时创建默认存储类，则可能会出现多个默认存储类。当存在多个默认存储类时，任何请求默认存储类 (**pvc.spec.storageClassName**=nil) 的持久性卷声明 (PVC) 都会获得最近创建的默认存储类，无论该存储类的默认存储类是什么，管理员都会在警报仪表板中收到警报，该类有多个默认存储类，**MultipleDefaultStorageClasses**。

5.2.4.2. 缺少默认存储类

PVC 可能会尝试使用不存在的默认存储类：

- 管理员删除默认存储类或将其标记为非默认，然后用户会创建一个请求默认存储类的 PVC。
- 在安装过程中，安装程序会创建一个请求默认存储类的 PVC，该类尚未创建。

在前面的场景中，PVC 会无限期处于 pending 状态。要解决这种情况，请创建一个默认存储类，或声明其中一个现有存储类作为默认值。创建或声明默认存储类后，PVC 就会获取新的默认存储类。如果可能，PVC 最终会绑定到静态或动态置备的 PV，并移出待处理状态。

5.2.5. 更改默认存储类

使用以下步骤更改默认存储类。

例如，您有两个定义的存储类 **gp3** 和 **standard**，您想要将默认存储类从 **gp3** 改为 **standard**。

先决条件

- 使用 cluster-admin 权限访问集群。

流程

更改默认存储类：

1. 列出存储类：

```
$ oc get storageclass
```

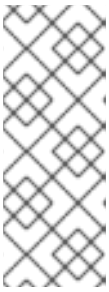
输出示例

```
NAME            TYPE
gp3 (default)    kubernetes.io/aws-ebs 1
standard         kubernetes.io/aws-ebs
```

1 **(default)** 表示默认存储类。

2. 将所需的存储类设为默认存储类。
对于所需的存储类，运行以下命令将 **storageclass.kubernetes.io/is-default-class** 注解设置为 **true** ：

```
$ oc patch storageclass standard -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```



注意

您可以短时间内有多个默认存储类。但是，您应该确保最终只有一个默认存储类。

当存在多个默认存储类时，任何请求默认存储类 (**pvc.spec.storageClassName**=nil) 的持久性卷声明 (PVC) 都会获得最近创建的默认存储类，无论该存储类的默认存储类是什么，管理员都会在警报仪表板中收到警报，该类有多个默认存储类，**MultipleDefaultStorageClasses**。

3. 从旧的默认存储类中删除默认存储类设置。
对于旧的默认存储类，运行以下命令将 **storageclass.kubernetes.io/is-default-class** 注解的值改为 **false** ：

```
$ oc patch storageclass gp3 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-
default-class": "false"}}}'
```

4. 确认更改：

```
$ oc get storageclass
```

输出示例

```
NAME              TYPE
gp3               kubernetes.io/aws-ebs
standard (default)   kubernetes.io/aws-ebs
```

5.3. AWS ELASTIC BLOCK STORE CSI DRIVER OPERATOR

5.3.1. 概述

OpenShift Dedicated 可以使用 AWS EBS CSI 驱动程序置备持久性卷 (PV)。

在使用 Container Storage Interface (CSI) Operator 和驱动时，我们建议用户需要熟悉持久性存储和配置 CSI 卷。

要创建挂载到 AWS EBS 存储资产中的 CSI 置备 PV，OpenShift Dedicated 在 **openshift-cluster-csi-drivers** 命名空间中默认安装 AWS EBS CSI Driver Operator（Red Hat operator)和 AWS EBS CSI 驱动程序。

- *AWS EBS CSI Driver Operator* 默认提供了一个 StorageClass，您可使用它来创建 PVC。如果需要，您可以禁用此默认存储类 (请参阅管理默认存储类)。您还可以选择创建 AWS EBS StorageClass，如使用 Amazon Elastic Block Store 的持久性存储 所述。
- *AWS EBS CSI 驱动程序*允许您创建并挂载 AWS EBS PV。

5.3.2. 关于 CSI

在过去，存储厂商一般会把存储驱动作为 Kubernetes 的一个部分提供。随着容器存储接口 (CSI) 的实现，第三方供应商可以使用标准接口来提供存储插件，而无需更改核心 Kubernetes 代码。

CSI Operators 为 OpenShift Dedicated 用户提供了存储选项，如卷快照，它无法通过 in-tree 卷插件实现。

重要

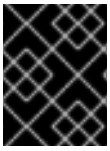
OpenShift Dedicated 默认使用 CSI 插件置备 Amazon Elastic Block Store (Amazon EBS) 存储。

有关在 OpenShift Dedicated 中动态置备 AWS EBS 持久性卷的详情，请参考使用 Amazon Elastic Block Store 的持久性存储。

其他资源

- 使用 Amazon Elastic Block Store 的持久性存储
- 配置 CSI 卷

5.4. 设置 AWS ELASTIC FILE SERVICE CSI DRIVER OPERATOR



重要

此流程特定于 AWS EFS CSI Driver Operator （一个 Red Hat operator），它仅适用于 OpenShift Dedicated 4.10 及更新的版本。

5.4.1. 概述

OpenShift Dedicated 可以使用 AWS EFS CSI 驱动程序 置备持久性卷 (PV)。

在使用 CSI Operator 和驱动程序时，建议使用 持久性存储 和配置 CSI 卷 。

安装 AWS EFS CSI Driver Operator 后，OpenShift Dedicated 在 **openshift-cluster-csi-drivers** 命名空间中默认安装 AWS EFS CSI Operator 和 AWS EFS CSI 驱动程序。这可让 AWS EFS CSI Driver Operator 创建挂载到 AWS EFS 资产中的 CSI 置备 PV。

- 安装之后，*AWS EFS CSI Driver Operator* 不会默认创建存储类来创建持久性卷声明 (PVC)。但是，您可以手动创建 AWS EFS **StorageClass**。AWS EFS CSI Driver Operator 支持动态卷置备，方法是允许按需创建存储卷。这消除了集群管理员预置备存储的需求。
- *AWS EFS CSI 驱动程序* 允许您创建并挂载 AWS EFS PV。

注意

Amazon Elastic File Storage (Amazon EFS) 只支持 egional 卷，不支持 zonal 卷。

5.4.2. 关于 CSI

在过去，存储厂商一般会把存储驱动作为 Kubernetes 的一个部分提供。随着容器存储接口 (CSI) 的实现，第三方供应商可以使用标准接口来提供存储插件，而无需更改核心 Kubernetes 代码。

CSI Operators 为 OpenShift Dedicated 用户提供了存储选项，如卷快照，它无法通过 in-tree 卷插件实现。

5.4.3. 设置 AWS EFS CSI Driver Operator

1. 安装 AWS EFS CSI Driver Operator （一个红帽 operator）。
2. 安装 AWS EFS CSI Driver Operator。
3. 安装 AWS EFS CSI 驱动程序。

5.4.3.1. 安装 AWS EFS CSI Driver Operator

默认情况下，AWS EFS CSI Driver Operator (Red Hat Operator)不会在 OpenShift Dedicated 中安装。使用以下步骤在集群中安装和配置 AWS EFS CSI Driver Operator。

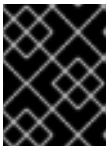
先决条件

- 访问 OpenShift Dedicated Web 控制台。

流程

从 web 控制台安装 AWS EFS CSI Driver Operator：

1. 登录到 web 控制台。
2. 安装 AWS EFS CSI Operator：
   a. 点 **Operators → OperatorHub**。
   b. 通过在过滤框中键入 AWS EFS CSI 来找到 **AWS EFS CSI** Operator。
   c. 点 **AWS EFS CSI Driver Operator** 按钮。



重要

确保选择 **AWS EFS CSI Driver Operator**，而不是 **AWS EFS Operator**。**AWS EFS Operator** 是一个社区 Operator，不受红帽支持。

   a. 在 **AWS EFS CSI Driver Operator** 页面中，点 **Install**。

b. 在 **Install Operator** 页面中，确保：

- 选择 **All namespaces on the cluster (default)**。
- **安装的命名空间** 被设置为 **openshift-cluster-csi-drivers**。

c. 点 **Install**。
安装完成后，AWS EFS CSI Operator 会在 web 控制台的 **Installed Operators** 部分列出。

后续步骤

- 安装 AWS EFS CSI 驱动程序

5.4.3.2. 安装 AWS EFS CSI 驱动程序

安装 AWS EFS CSI Driver Operator 后，您要安装 AWS EFS CSI Driver。

先决条件

- 访问 OpenShift Dedicated Web 控制台。

流程

1. 点 **Administration → CustomResourceDefinitions → ClusterCSIDriver**。
2. 在 **Instances** 选项卡上，单击 **Create ClusterCSIDriver**。
3. 使用以下 YAML 文件：

```
apiVersion: operator.openshift.io/v1
kind: ClusterCSIDriver
metadata:
    name: efs.csi.aws.com
spec:
  managementState: Managed
```

4. 点 **Create**。
5. 等待以下 Conditions 更改为 "True" 状态：
   - AWSEFSDriverNodeServiceControllerAvailable
   - AWSEFSDriverControllerServiceControllerAvailable

5.4.4. 创建 AWS EFS 存储类

存储类用于区分和划分存储级别和使用。通过定义存储类，用户可以获得动态置备的持久性卷。

安装后，*AWS EFS CSI Driver Operator* （一个 Red Hat operator） 不会默认创建存储类。但是，您可以手动创建 AWS EFS 存储类。

5.4.4.1. 使用控制台创建 AWS EFS 存储类

流程

1. 在 OpenShift Dedicated 控制台中，点 **Storage → StorageClasses**。
2. 在 **StorageClasses** 页面中，点 **Create StorageClass**。
3. 在 **StorageClass** 页面中，执行以下步骤：
   a. 输入一个名称来指代存储类。
   b. 可选：输入描述。
   c. 选择 reclaim 策略。
   d. 从 **Provisioner** 下拉列表中，选择 **efs.csi.aws.com**。
   e. 可选：为所选置备程序设置配置参数。
4. 点 **Create**。

5.4.4.2. 使用 CLI 创建 AWS EFS 存储类

流程

- 创建 **StorageClass** 对象：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
parameters:
  provisioningMode: efs-ap (1)
  fileSystemId: fs-a5324911 (2)
  directoryPerms: "700" (3)
  gidRangeStart: "1000" (4)
  gidRangeEnd: "2000" (5)
  basePath: "/dynamic_provisioning" (6)
```

(1) **provisioningMode** 必须是 **efs-ap** 才能启用动态置备。

(2) **fileSystemId** 必须是手动创建的 EFS 卷的 ID。

(3) **directoryPerms** 是卷的根目录的默认权限。在本例中，该卷只能被所有者访问。

(4) (5) **gidRangeStart** 和 **gidRangeEnd** 设置用于设置 AWS 访问点 GID 的 POSIX 组 ID(GID)范围。如果未指定，则默认范围为 50000-7000000。每个置备的卷（即 AWS 访问点）都会被分配一个这个范围内的唯一 GID。

(6) **BasePath** 是 EFS 卷上用于创建动态置备卷的目录。在这种情况下，PV 被置备为 EFS 卷上的 "/dynamic_provisioning/<random uuid>"。只有 子目录挂载到使用该 PV 的 pod。

注意

集群管理员可创建几个 **StorageClass** 对象，各自使用不同的 EFS 卷。

5.4.5. 在 AWS 中创建和配置对 EFS 卷的访问

此流程解释了如何在 AWS 中创建和配置 EFS 卷，以便在 OpenShift Dedicated 中使用它们。

先决条件

- AWS 帐户凭证

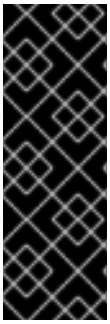
流程

在 AWS 中创建和配置对 EFS 卷的访问：

1. 在 AWS 控制台中，打开 https://console.aws.amazon.com/efs。
2. 点击 **Create 文件系统** ：
   - 输入文件系统的名称。
   - 对于 **Virtual Private Cloud (VPC)** 请选择 OpenShift Dedicated 的虚拟私有云 (VPC)。
   - 接受所有其他选择的默认设置。
3. 等待卷和挂载目标完成完全创建：
   a. 访问 https://console.aws.amazon.com/efs#/file-systems。
   b. 单击您的卷，在 **Network** 选项卡中，等待所有挂载目标变为可用状态（约 1-2 分钟）。
4. 在 **Network** 选项卡上，复制安全组 ID（下一步中您将需要此 ID）。
5. 进入 https://console.aws.amazon.com/ec2/v2/home#SecurityGroups，并查找 EFS 卷使用的安全组。
6. 在 **Inbound rules** 选项卡中，点 **Edit inbound rules**，然后使用以下设置添加新规则，以允许 OpenShift Dedicated 节点访问 EFS 卷：
   - **类型** ：NFS
   - **协议** ：TCP
   - **端口范围** ：2049
   - **源** ：您的节点的自定义/IP 地址范围（例如："10.0.0.0/16"）
     此步骤允许 OpenShift Dedicated 使用集群中的 NFS 端口。
7. 保存规则。

5.4.6. Amazon Elastic File Storage 的动态置备

AWS EFS CSI 驱动程序支持不同于其他 CSI 驱动程序的动态置备形式。它将新 PV 调配为预先存在的 EFS 卷的子目录。PV 相互独立。但是，它们共享相同的 EFS 卷。删除卷时，置备的所有 PV 也会被删除。EFS CSI 驱动程序为每个此类子目录创建一个 AWS Access Point。由于 AWS AccessPoint 限制，您只能从一个 **StorageClass**/EFS 卷动态置备 1000 个 PV。



重要

请注意，EFS 不强制执行 **PVC.spec.resources**。

在以下示例中，您请求 5 GiB 的空间。但是，创建的 PV 是无限的，可以存储任何数量的数据（如 PB）。当在卷中存储太多数据时，一个被破坏的应用甚至恶意应用程序也可能会导致大量开支。

强烈建议在 AWS 中使用 EFS 卷大小监控。

先决条件

- 您已创建了 Amazon Elastic File Storage (Amazon EFS)卷。
- 您已创建了 AWS EFS 存储类。

流程

启用动态置备：

- 照常创建 PVC （或 StatefulSet 或 Template），引用之前创建的 **StorageClass**。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: test
spec:
  storageClassName: efs-sc
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
```

如果您在设置动态置备时遇到问题，请参阅 Amazon Elastic File Storage 故障排除 。

其他资源

- 创建并配置对 Amazon EFS 卷的访问
- 创建 AWS EFS 存储类

5.4.7. 使用 Amazon Elastic File Storage 创建静态 PV

可以使用 Amazon Elastic File Storage (Amazon EFS) 卷作为单个 PV，而无需动态置备。整个卷挂载到 pod。

先决条件

- 您已创建了 Amazon EFS 卷。

流程

- 使用以下 YAML 文件创建 PV：

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
spec:
  capacity: 1
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  csi:
    driver: efs.csi.aws.com
    volumeHandle: fs-ae66151a 2
    volumeAttributes:
      encryptInTransit: "false" 3
```

1 **spec.capacity** 没有任何含义，CSI 驱动程序会忽略它。只有在绑定到 PVC 时才会使用它。应用可以将任意数量的数据存储到卷中。

2 **volumeHandle** 必须与您在 AWS 中创建的 EFS 卷相同。如果您提供自己的访问点，**volumeHandle** 应为 **<EFS volume ID>::<access point ID>**。例如：**fs-6e633ada::fsap-081a1d293f0004630**。

3 如果需要，您可以在传输中禁用加密。加密功能会被默认启用。

如果您在设置静态 PV 时遇到问题，请参阅 Amazon Elastic File Storage 故障排除 。

5.4.8. Amazon Elastic File Storage 安全性

以下信息对于 Amazon Elastic File Storage (Amazon EFS) 安全非常重要。

例如，在使用接入点（例如，使用前面描述的动态置备）时，Amazon 会自动将文件的 GID 替换为接入点的 GID。此外，EFS 在评估文件系统权限时，会考虑访问点的用户 ID、组 ID 和次要组 ID。EFS 忽略 NFS 客户端的 ID。有关接入点的详情请参考 https://docs.aws.amazon.com/efs/latest/ug/efs-access-points.html。

因此，EFS 卷静默忽略 FSGroup；OpenShift Dedicated 无法将卷上的文件 GID 替换为 FSGroup。任何可以访问挂载的 EFS 接入点的 pod 都可以访问其中的任何文件。

与此无关，传输中的加密默认是启用的。如需更多信息，请参阅 https://docs.aws.amazon.com/efs/latest/ug/encryption-in-transit.html。

5.4.9. Amazon Elastic File Storage 故障排除

以下信息提供了有关如何对 Amazon Elastic File Storage (Amazon EFS) 问题进行故障排除的指导：

- AWS EFS Operator 和 CSI 驱动程序在命名空间 **openshift-cluster-csi-drivers** 中运行。
- 要启动收集 AWS EFS Operator 和 CSI 驱动程序的日志，请运行以下命令：

```
$ oc adm must-gather
[must-gather      ] OUT Using must-gather plugin-in image: quay.io/openshift-release-
```

```
dev/ocp-v4.0-art-
dev@sha256:125f183d13601537ff15b3239df95d47f0a604da2847b561151fedd699f5e3a5
[must-gather      ] OUT namespace/openshift-must-gather-xm4wq created
[must-gather      ] OUT clusterrolebinding.rbac.authorization.k8s.io/must-gather-2bd8x
created
[must-gather      ] OUT pod for plug-in image quay.io/openshift-release-dev/ocp-v4.0-art-
dev@sha256:125f183d13601537ff15b3239df95d47f0a604da2847b561151fedd699f5e3a5
created
```

- 要显示 AWS EFS Operator 错误，请查看 **ClusterCSIDriver** 状态：

```
$ oc get clustercsidriver efs.csi.aws.com -o yaml
```

- 如果卷无法挂载到容器集（如下命令的输出中所示）：

```
$ oc describe pod
...
  Type     Reason       Age    From               Message
  ----     ------       ----   ----               -------
  Normal   Scheduled    2m13s  default-scheduler  Successfully assigned default/efs-app to
ip-10-0-135-94.ec2.internal
  Warning  FailedMount  13s    kubelet            MountVolume.SetUp failed for volume "pvc-
d7c097e6-67ec-4fae-b968-7e7056796449" : rpc error: code = DeadlineExceeded desc =
context deadline exceeded (1)
  Warning  FailedMount  10s    kubelet            Unable to attach or mount volumes: unmounted
volumes=[persistent-storage], unattached volumes=[persistent-storage kube-api-access-
9j477]: timed out waiting for the condition
```

(1) 指示卷未挂载的警告消息。

此错误通常是由 AWS 在 OpenShift Dedicated 节点和 Amazon EFS 之间丢弃数据包造成的。

检查以下内容是否正确：

- AWS 防火墙和安全组
- 网络：端口号和 IP 地址

5.4.10. 卸载 AWS EFS CSI Driver Operator

在卸载 AWS EFS CSI Driver Operator （红帽操作器）后，无法访问所有 EFS PV。

先决条件

- 访问 OpenShift Dedicated Web 控制台。

流程

从 web 控制台卸载 AWS EFS CSI Driver Operator：

1. 登录到 web 控制台。
2. 停止所有使用 AWS EFS PV 的应用程序。
3. 删除所有 AWS EFS PV：

a. 点 **Storage → PersistentVolumeClaims**。

b. 选择 AWS EFS CSI Driver Operator 使用的每个 PVC，点击 PVC 最右侧的下拉菜单，然后点 **Delete PersistentVolumeClaims**。

4. 卸载 AWS EFS CSI 驱动程序 ：

注意

在卸载 Operator 前，必须先删除 CSI 驱动程序。

a. 点 **Administration → CustomResourceDefinitions → ClusterCSIDriver**。

b. 在 **Instances** 选项卡上，单击左侧的 **efs.csi.aws.com**，单击下拉菜单，然后单击 **Delete ClusterCSIDriver**。

c. 出现提示时，单击 **Delete**。

5. 卸载 AWS EFS CSI Operator：

a. 点 **Operators → Installed Operators**。

b. 在 **Installed Operators** 页面中，在 **Search by name** 框中输入 AWS EFS CSI 来查找 Operator，然后点击它。

c. 在 **Installed Operators > Operator 详情** 页面的右上角，点 **Actions → Uninstall Operator**。

d. 当在 **Uninstall Operator** 窗口中提示时，点 **Uninstall** 按钮从命名空间中删除 Operator。Operator 在集群中部署的任何应用程序都需要手动清理。
卸载后，AWS EFS CSI Driver Operator 不会在 web 控制台的 **Installed Operators** 部分列出。

注意

在销毁集群 (**openshift-install destroy cluster**) 前，您必须删除 AWS 中的 EFS 卷。如果有使用集群的 VPC 的 EFS 卷，OpenShift Dedicated 集群将无法被销毁。Amazon 不允许删除这样的 VPC。

5.4.11. 其他资源

- 配置 CSI 卷

5.5. GCP PD CSI DRIVER OPERATOR

5.5.1. 概述

OpenShift Dedicated 可以使用 Google Cloud Platform（GCP）持久性存储的 Container Storage Interface（CSI）驱动程序来置备持久性卷（PV）。

在使用 Container Storage Interface (CSI) Operator 和驱动时，我们建议用户需要熟悉持久性存储和配置 CSI 卷。

要创建挂载到 GCP PD 存储资产中的 CSI 置备持久性卷（PV），OpenShift Dedicated 在 **openshift-cluster-csi-drivers** 命名空间中默认安装 GCP PD CSI Driver Operator 和 GCP PD CSI 驱动程序。

- **GCP PD CSI Driver Operator**：默认情况下，Operator 提供了一个可用来创建 PVC 的存储类。如果需要，您可以禁用此默认存储类 (请参阅管理默认存储类)。您还可以选择创建 GCP PD 存储类，如使用 GCE Persistent Disk 的 Persistent Storage 所述。
- **GCP PD 驱动程序**：该驱动程序可让您创建并挂载 GCP PD PV。

5.5.2. 关于 CSI

在过去，存储厂商一般会把存储驱动作为 Kubernetes 的一个部分提供。随着容器存储接口 (CSI) 的实现，第三方供应商可以使用标准接口来提供存储插件，而无需更改核心 Kubernetes 代码。

CSI Operators 为 OpenShift Dedicated 用户提供了存储选项，如卷快照，它无法通过 in-tree 卷插件实现。

5.5.3. GCP PD CSI 驱动程序存储类参数

Google Cloud Platform（GCP）持久磁盘（PD）Container Storage Interface（CSI）驱动程序使用 CSI **external-provisioner** sidecar 作为控制器。这是和 CSI 驱动程序一起部署的单独的 helper 容器。sidecar 通过触发 **CreateVolume** 操作来管理持久性卷（PV）。

GCP PD CSI 驱动程序使用 **csi.storage.k8s.io/fstype** 参数键来支持动态置备。下表描述了 OpenShift Dedicated 支持的所有 GCP PD CSI 存储类参数。

表 5.2. CreateVolume 参数

参数	值	默认	描述
type	**pd-ssd** 或者 **pd-standard**	**pd-standard**	允许您选择标准的 PV 或使用固态硬盘的 PV。
replication-type	**none** 或 **regional-pd**	**none**	允许您在 zonal 或区域 PV 之间进行选择。
disk-encryption-kms-key	用于加密新磁盘的密钥的完全限定资源标识符。	空字符串	使用客户管理的加密密钥（CMEK）加密新磁盘。

5.5.4. 创建自定义加密的持久性卷

创建 **PersistentVolumeClaim** 对象时，OpenShift Dedicated 会置备一个新的持久性卷（PV）并创建一个 **PersistentVolume** 对象。您可以通过加密新创建的 PV，在 Google Cloud Platform（GCP）中添加自定义加密密钥来保护集群中的 PV。

要加密，您新创建的 PV 使用新的或现有的 Google Cloud Key Management Service（KMS）密钥在集群中使用用户管理的加密密钥（CMEK）。

先决条件

- 已登陆到一个正在运行的 OpenShift Dedicated 集群。
- 您已创建了 Cloud KMS 密钥环以及密钥版本。

有关 CMEK 和 Cloud KMS 资源的更多信息，请参阅使用客户管理的加密密钥（CMEK）。

流程

要创建自定义加密 PV，请完成以下步骤：

1. 使用 Cloud KMS 密钥创建存储类。以下示例启用加密卷的动态置备：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-gce-pd-cmek
provisioner: pd.csi.storage.gke.io
volumeBindingMode: "WaitForFirstConsumer"
allowVolumeExpansion: true
parameters:
  type: pd-standard
  disk-encryption-kms-key: projects/<key-project-id>/locations/<location>/keyRings/<key-ring>/cryptoKeys/<key> (1)
```

(1) 此字段必须是用于加密新磁盘的密钥的资源标识符。值是区分大小写的。有关提供关键 ID 值的更多信息，请参阅检索资源 ID 和 获取 Cloud KMS 资源 ID。

注意

您不能将 **disk-encryption-kms-key** 参数添加到现有的存储类中。但是，您可以删除存储类并使用相同的名称和不同的参数集合重新创建该存储类。如果您这样做，现有类的置备程序必须是 **pd.csi.storage.gke.io**。

2. 使用 **oc** 命令在 OpenShift Dedicated 集群上部署存储类：

```
$ oc describe storageclass csi-gce-pd-cmek
```

输出示例

```
Name:                  csi-gce-pd-cmek
IsDefaultClass:        No
Annotations:           None
Provisioner:           pd.csi.storage.gke.io
Parameters:            disk-encryption-kms-key=projects/key-project-id/locations/location/keyRings/ring-name/cryptoKeys/key-name,type=pd-standard
AllowVolumeExpansion:  true
MountOptions:          none
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
Events:                none
```

3. 创建名为 **pvc.yaml** 的文件，该文件与您在上一步中创建的存储类对象的名称匹配：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: podpvc
spec:
  accessModes:
```

```
  - ReadWriteOnce
storageClassName: csi-gce-pd-cmek
resources:
  requests:
    storage: 6Gi
```

注意

如果将新存储类标记为默认值，可以省略 **storageClassName** 字段。

4. 在集群中应用 PVC:

```
$ oc apply -f pvc.yaml
```

5. 获取 PVC 的状态，并验证它是否已创建并绑定到新置备的 PV:

```
$ oc get pvc
```

输**出示例**

```
NAME     STATUS    VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS     AGE
podpvc   Bound     pvc-e36abf50-84f3-11e8-8538-42010a800002   10Gi       RWO            csi-
gce-pd-cmek  9s
```

注意

如果您的存储类将 **volumeBindingMode** 字段设置为 **WaitForFirstConsumer**，您必须创建一个 pod 来使用 PVC，然后才能验证它。

您的 CMEK 保护 PV 现在可以与 OpenShift Dedicated 集群一起使用。

5.5.5. 其他资源

- 使用 GCE Persistent Disk 的持久性存储
- 配置 CSI 卷

5.6. GOOGLE COMPUTE PLATFORM FILESTORE CSI DRIVER OPERATOR

5.6.1. 概述

OpenShift Dedicated 可以使用 Google Compute Platform (GCP)文件存储存储的 Container Storage Interface (CSI)驱动程序置备持久性卷(PV)。

在使用 CSI Operator 和驱动程序时，建议先熟悉 持久性存储和配置 CSI 卷。

要创建挂载到 GCP Filestore Storage 资产中的 CSI 置备 PV，您可以在 **openshift-cluster-csi-drivers** 命名空间中安装 GCP Filestore CSI Driver Operator 和 GCP Filestore CSI 驱动程序。

- *GCP Filestore CSI Driver Operator* 默认不提供存储类，但如果需要可以创建一个。GCP Filestore CSI Driver Operator 支持动态卷置备，方法是允许按需创建存储卷，使集群管理员无需预置备存储。
- *GCP Filestore CSI 驱动程序*允许您创建并挂载 GCP Filestore PV。

5.6.2. 关于 CSI

在过去，存储厂商一般会把存储驱动作为 Kubernetes 的一个部分提供。随着容器存储接口 (CSI) 的实现，第三方供应商可以使用标准接口来提供存储插件，而无需更改核心 Kubernetes 代码。

CSI Operators 为 OpenShift Dedicated 用户提供了存储选项，如卷快照，它无法通过 in-tree 卷插件实现。

5.6.3. 安装 GCP Filestore CSI Driver Operator

默认情况下，Google Compute Platform (GCP) Filestore Container Storage Interface (CSI) Driver Operator 不会在 OpenShift Dedicated 中安装。使用以下步骤在集群中安装 GCP Filestore CSI Driver Operator。

先决条件

- 访问 OpenShift Dedicated Web 控制台。

流程

从 web 控制台安装 GCP Filestore CSI Driver Operator：

1. 登录 OpenShift Cluster Manager。
2. 选择集群。
3. 点 **Open console**，并使用您的凭证登录。
4. 运行以下命令，在 GCE 项目中启用 Filestore API：

```
$ gcloud services enable file.googleapis.com  --project <my_gce_project> 1
```

1 将 **<my_gce_project>** 替换为您的 Google Cloud 项目。

您还可以使用 Google Cloud web 控制台进行此操作。

5. 安装 GCP Filestore CSI Operator：
   a. 点 **Operators → OperatorHub**。
   b. 通过在过滤器框中输入 GCP Filestore 来查找 **GCP Filestore** CSI Operator。
   c. 点 **GCP Filestore CSI Driver Operator** 按钮。
   d. 在 **GCP Filestore CSI Driver Operator** 页面中，点 **Install**。
   e. 在 **Install Operator** 页面中，确保：
      - 选择 **All namespaces on the cluster (default)**。

- **安装的命名空间** 被设置为 **openshift-cluster-csi-drivers**。

f. 点 **Install**。
安装完成后，GCP Filestore CSI Operator 会在 web 控制台的 **Installed Operators** 部分列出。

6. 安装 GCP Filestore CSI 驱动程序：

a. 点 **Administration** → **CustomResourceDefinitions** → **ClusterCSIDriver**。

b. 在 **Instances** 选项卡上，单击 **Create ClusterCSIDriver**。
使用以下 YAML 文件：

```
apiVersion: operator.openshift.io/v1
kind: ClusterCSIDriver
metadata:
    name: filestore.csi.storage.gke.io
spec:
  managementState: Managed
```

c. 点 **Create**。

d. 等待以下 Conditions 变为 "true" 状态：

- GCPFilestoreDriverCredentialsRequestControllerAvailable
- GCPFilestoreDriverNodeServiceControllerAvailable
- GCPFilestoreDriverControllerServiceControllerAvailable

其他资源

- 在 Google Cloud 中启用 API。
- 使用 Google Cloud Web 控制台启用 API。

5.6.4. 为 GCP Filestore 存储创建存储类

安装 Operator 后，您应该创建一个存储类来动态置备 Google Compute Platform (GCP) 文件存储卷。

先决条件

- 已登陆到正在运行的 OpenShift Dedicated 集群。

流程

创建存储类：

1. 使用以下 YAML 文件示例创建存储类：

YAML 文件示例

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: filestore-csi
```

```
provisioner: filestore.csi.storage.gke.io
parameters:
  connect-mode: DIRECT_PEERING (1)
  network: network-name (2)
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

(1) 对于共享 VPC，使用 **connect-mode** 参数设置为 **PRIVATE_SERVICE_ACCESS**。对于非共享 VPC，值是 **DIRECT_PEERING**，这是默认设置。

(2) 指定应该在其中创建 Filestore 实例的 GCP 虚拟私有云(VPC) 网络的名称。

2. 指定在其中创建 Filestore 实例的 VPC 网络的名称。
建议您指定应在其中创建 Filestore 实例的 VPC 网络。如果没有指定 VPC 网络，Container Storage Interface (CSI) 驱动程序会尝试在项目的默认 VPC 网络中创建实例。

   在 IPI 安装中，VPC 网络名称通常是带有后缀 "-network" 的集群名称。但是，在 UPI 安装中，VPC 网络名称可以是用户选择的任何值。

   对于一个共享的 VPC (**connect-mode** = **PRIVATE_SERVICE_ACCESS**)，网络需要是一个完整的 VPC 名。例如：**projects/shared-vpc-name/global/networks/gcp-filestore-network**。

   您可以使用以下命令检查 **MachineSets** 对象来查找 VPC 网络名称：

```
$ oc -n openshift-machine-api get machinesets -o yaml | grep "network:"
          - network: gcp-filestore-network
(...)
```

   在本例中，这个集群中的 VPC 网络名称为 "gcp-filestore-network"。

5.6.5. 销毁集群和 GCP 文件存储

通常，如果您销毁集群，OpenShift Dedicated 安装程序会删除属于该集群的所有云资源。但是，当销毁集群时，Google Compute Platform (GCP) Filestore 实例不会被自动删除，因此您必须在销毁集群前手动删除使用 Filestore 存储类的所有持久性卷声明 (PVC)。

流程

删除所有 GCP Filestore PVC：

1. 列出使用存储类 **filestore-csi** 创建的所有 PVC：

```
$ oc get pvc -o json -A | jq -r '.items[] | select(.spec.storageClassName == "filestore-csi")
```

2. 删除上一命令列出的所有 PVC：

```
$ oc delete <pvc-name> (1)
```

(1) 将 <pvc-name> 替换为您要删除的任何 PVC 的名称。

5.6.6. 其他资源

- 配置 CSI 卷

第 6 章 通用临时卷

6.1. 概述

通用临时卷是临时卷类型，可以由支持持久性卷和动态置备的所有存储驱动程序提供。通用临时卷与 **emptyDir** 卷类似，它们为从头开始数据提供每个 pod 目录，这通常在置备后为空。

通用临时卷在 pod 规格中内联指定，并遵循 pod 的生命周期。它们与 pod 一起创建和删除。

通用临时卷具有以下功能：

- 存储可以是本地的或者网络附加存储。
- 卷可以有 pod 无法超过的固定大小。
- 根据驱动程序和参数，卷可能有一些初始数据。
- 支持卷上的典型的操作，假设驱动程序支持它们，包括快照、克隆、调整大小和存储容量跟踪。

注意

通用临时卷不支持离线快照和调整大小。

6.2. 生命周期和持久性卷声明

卷声明的参数允许在 pod 的卷源内。支持声明(PVC)的标签、注解和整个字段。当创建这样的 pod 时，临时卷控制器随后会在与 pod 相同的命名空间中创建实际的 PVC 对象（来自 *创建通用临时卷* 流程中显示的模板），并确保在 pod 被删除时删除 PVC。这会以两种方式之一触发卷绑定和置备：

- 另外，如果存储类使用即时卷绑定。
  通过立即绑定，调度程序被强制选择在卷可用后有权访问的节点。
- 当 pod 放入节点时（**WaitForFirstConsumervolume** 绑定模式）。
  建议这个卷绑定选项用于通用临时卷，因为调度程序可以为 pod 选择适当的节点。

就资源限制而言，具有通用临时存储的 pod 是提供该临时存储的 PVC 的所有者。当 pod 被删除时，Kubernetes 垃圾回收器会删除 PVC，然后，后者通常会触发删除卷，因为存储类的默认重新声明策略是删除卷。您可以使用一个带有保留策略的存储类创建 quasi-ephemeral 本地存储：存储会活跃 pod，在这种情况下，您必须确保卷清理单独发生。虽然这些 PVC 存在，它们可以像任何其他 PVC 一样使用。特别是，可以在卷克隆或快照中被引用为数据源。PVC 对象也保存卷的当前状态。

其他资源

- 创建通用临时卷

6.3. 安全性

您可以启用通用临时卷功能，以便可以创建 pod 的用户也可以间接创建持久性卷声明 (PVC)。即使这些用户没有直接创建 PVC 的权限，此功能也可以正常工作。集群管理员必须了解这一点。如果这不适用于您的安全模型，请使用准入 Webhook 来拒绝对象，如具有通用临时卷的 pod。

PVC 的一般命名空间配额仍适用，因此即使允许用户使用此新机制，它们也无法使用它来绕过其他策略。

6.4. 持久性卷声明命名

自动创建的持久性卷声明(PVC)由 pod 名称和卷名称的组合命名，中间带有连字符(-)。这种命名惯例还引入了不同 pod 之间以及 pod 和手动创建 PVC 之间潜在的冲突。

例如，**pod-a** 带有卷 **scratch**，**pod** 带有卷 **a-scratch**，它们都使用相同的 PVC 名称 **pod-a-scratch**。

检测到这样的冲突，如果为 pod 创建，PVC 仅用于临时卷。此检查基于所有权关系。现有 PVC 不会被覆盖或修改，但这不会解决冲突。如果没有正确的 PVC，pod 无法启动。

重要

在在同一命名空间中命名 pod 和卷时要小心，以便不会发生命名冲突。

6.5. 创建通用临时卷

流程

1. 创建 **pod** 对象定义，并将其保存到文件中。
2. 在文件中包括通用临时卷信息。

my-example-pod-with-generic-vols.yaml

```
kind: Pod
apiVersion: v1
metadata:
  name: my-app
spec:
  containers:
    - name: my-frontend
      image: busybox:1.28
      volumeMounts:
      - mountPath: "/mnt/storage"
        name: data
      command: [ "sleep", "1000000" ]
  volumes:
    - name: data 1
      ephemeral:
        volumeClaimTemplate:
          metadata:
            labels:
              type: my-app-ephvol
          spec:
            accessModes: [ "ReadWriteOnce" ]
            storageClassName: "gp2-csi"
            resources:
              requests:
                storage: 1Gi
```

1 通用临时卷声明。

第 7 章 动态置备

7.1. 关于动态置备

StorageClass 资源对象描述并分类了可请求的存储，并提供了根据需要为动态置备存储传递参数的方法。**StorageClass** 也可以作为控制不同级别的存储和访问存储的管理机制。集群管理员(**cluster-admin**)或存储管理员(**storage-admin**)可以在无需了解底层存储卷资源的情况下，定义并创建用户可以请求的 **StorageClass** 对象。

OpenShift Dedicated 持久性卷框架启用了此功能，并允许管理员为集群置备持久性存储。该框架还可让用户在不了解底层存储架构的情况下请求这些资源。

在 OpenShift Dedicated 中，许多存储类型都可用作持久性卷。虽然它们都可以由管理员静态置备，但有些类型的存储是使用内置供应商和插件 API 动态创建的。

7.2. 可用的动态置备插件

OpenShift Dedicated 提供了以下置备程序插件，用于使用集群配置的供应商 API 创建新存储资源的动态部署：

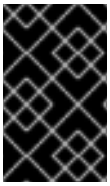
存储类型	provisioner 插件名称	注
Amazon Elastic Block Store (Amazon EBS)	**kubernetes.io/aws-ebs**	当在不同的区中使用多个集群进行动态置备时，使用 **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>**（每个集群的**<cluster_name>** 和 **<cluster_id>** 是唯一的）来标记（tag）每个节点。
GCE 持久性磁盘 (gcePD)	**kubernetes.io/gce-pd**	在多区配置中，建议在每个 GCE 项目中运行一个 OpenShift Dedicated 集群，以避免在当前集群中没有节点的区中创建 PV。
IBM Power® Virtual Server Block	**powervs.csi.ibm.com**	安装后，IBM Power® Virtual Server Block CSI Driver Operator 和 IBM Power® Virtual Server Block CSI Driver 会自动为动态置备创建所需的存储类。

重要

任何选择的置备程序插件还需要根据相关文档为相关的云、主机或者第三方供应商配置。

7.3. 定义存储类

StorageClass 对象目前是一个全局范围的对象，必须由 **cluster-admin** 或 **storage-admin** 用户创建。



重要

根据使用的平台，Cluster Storage Operator 可能会安装一个默认的存储类。此存储类由 Operator 拥有和控制。不能在定义注解和标签之外将其删除或修改。如果需要实现不同的行为，则必须定义自定义存储类。

以下小节描述了 **StorageClass** 对象的基本定义，以及每个支持的插件类型的具体示例。

7.3.1. 基本 StorageClass 对象定义

以下资源显示了用来配置存储类的参数和默认值。这个示例使用 AWS ElasticBlockStore (EBS) 对象定义。

StorageClass 定义示例

```
kind: StorageClass 1
apiVersion: storage.k8s.io/v1 2
metadata:
  name: <storage-class-name> 3
  annotations: 4
    storageclass.kubernetes.io/is-default-class: 'true'
    ...
provisioner: kubernetes.io/aws-ebs 5
parameters: 6
  type: gp3
...
```

1. （必需）API 对象类型。
2. （必需）当前的 apiVersion。
3. （必需）存储类的名称。
4. （可选）存储类的注解。
5. （必需）与这个存储类关联的置备程序类型。
6. （可选）特定置备程序所需的参数，这将根据插件的不同而有所不同。

7.3.2. 存储类注解

要将存储类设置为集群范围的默认值，请在存储类元数据中添加以下注解：

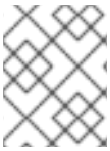
```
storageclass.kubernetes.io/is-default-class: "true"
```

例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
...
```

这允许任何没有指定特定存储类的持久性卷声明（PVC）通过默认存储类自动置备。但是，您的集群可以有多个存储类，但只有其中一个是默认存储类。



注意

beta 注解 **storageclass.beta.kubernetes.io/is-default-class** 当前仍然可用，但将在以后的版本中被删除。

要设置存储类描述，请在存储类元数据中添加以下注解：

```
kubernetes.io/description: My Storage Class Description
```

例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    kubernetes.io/description: My Storage Class Description
...
```

7.3.3. AWS Elastic Block Store (EBS) 对象定义

aws-ebs-storageclass.yaml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: <storage-class-name> 1
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1 2
  iopsPerGB: "10" 3
  encrypted: "true" 4
  kmsKeyId: keyvalue 5
  fsType: ext4 6
```

1. （必需）存储类的名称。持久性卷声明使用此存储类来置备关联的持久性卷。
2. (必需) 从 **io1**, **gp3**, **sc1**, **st1** 选择。默认值为 **gp3**。可用的 Amazon 资源名 (ARN) 值请查看 AWS 文档 。
3. 可选：只适用于 **io1** 卷。每个 GiB 每秒一次 I/O 操作。AWS 卷插件乘以这个值，再乘以请求卷的大小以计算卷的 IOPS。数值上限为 20,000 IOPS，这是 AWS 支持的最大值。详情请查看 AWS 文档 。
4. 可选：是否加密 EBS 卷。有效值为 **true** 或者 **false**。

5 可选：加密卷时使用的密钥的完整 ARN。如果没有提供任何信息，但 **encypted** 被设置为 **true**，则 AWS 会生成一个密钥。有效 ARN 值请查看 AWS 文档 。

6 可选：在动态置备卷中创建的文件系统。这个值被复制到动态配置的持久性卷的 **fstype** 字段中，并在第一个挂载卷时创建文件系统。默认值为 **ext4**。

7.3.4. GCE PersistentDisk (gcePD) 对象定义

gce-pd-storageclass.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <storage-class-name> 1
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard 2
  replication-type: none
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
reclaimPolicy: Delete
```

1 存储类的名称。持久性卷声明使用此存储类来置备关联的持久性卷。

2 选择 **pd-standard** 或 **pd-ssd**。默认为 **pd-standard**。

7.4. 更改默认存储类

使用以下步骤更改默认存储类。

例如，您有两个定义的存储类 **gp3** 和 **standard**，您想要将默认存储类从 **gp3** 改为 **standard**。

先决条件

- 使用 cluster-admin 权限访问集群。

流程

更改默认存储类：

1. 列出存储类：

```
$ oc get storageclass
```

输出示例

```
NAME                TYPE
gp3 (default)       kubernetes.io/aws-ebs 1
standard            kubernetes.io/aws-ebs
```

1 **(default)** 表示默认存储类。

2. 将所需的存储类设为默认存储类。
   对于所需的存储类，运行以下命令将 **storageclass.kubernetes.io/is-default-class** 注解设置为 **true** ：

```
$ oc patch storageclass standard -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class": "true"}}}'
```

注意

您可以短时间内有多个默认存储类。但是，您应该确保最终只有一个默认存储类。

当存在多个默认存储类时，任何请求默认存储类 (**pvc.spec.storageClassName**=nil) 的持久性卷声明 (PVC) 都会获得最近创建的默认存储类，无论该存储类的默认存储类是什么，管理员都会在警报仪表板中收到警报，该类有多个默认存储类，**MultipleDefaultStorageClasses**。

3. 从旧的默认存储类中删除默认存储类设置。
   对于旧的默认存储类，运行以下命令将 **storageclass.kubernetes.io/is-default-class** 注解的值改为 **false** ：

```
$ oc patch storageclass gp3 -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-
default-class": "false"}}}'
```

4. 确认更改：

```
$ oc get storageclass
```

输**出示例**

```
NAME                 TYPE
gp3                  kubernetes.io/aws-ebs
standard (default)   kubernetes.io/aws-ebs
```