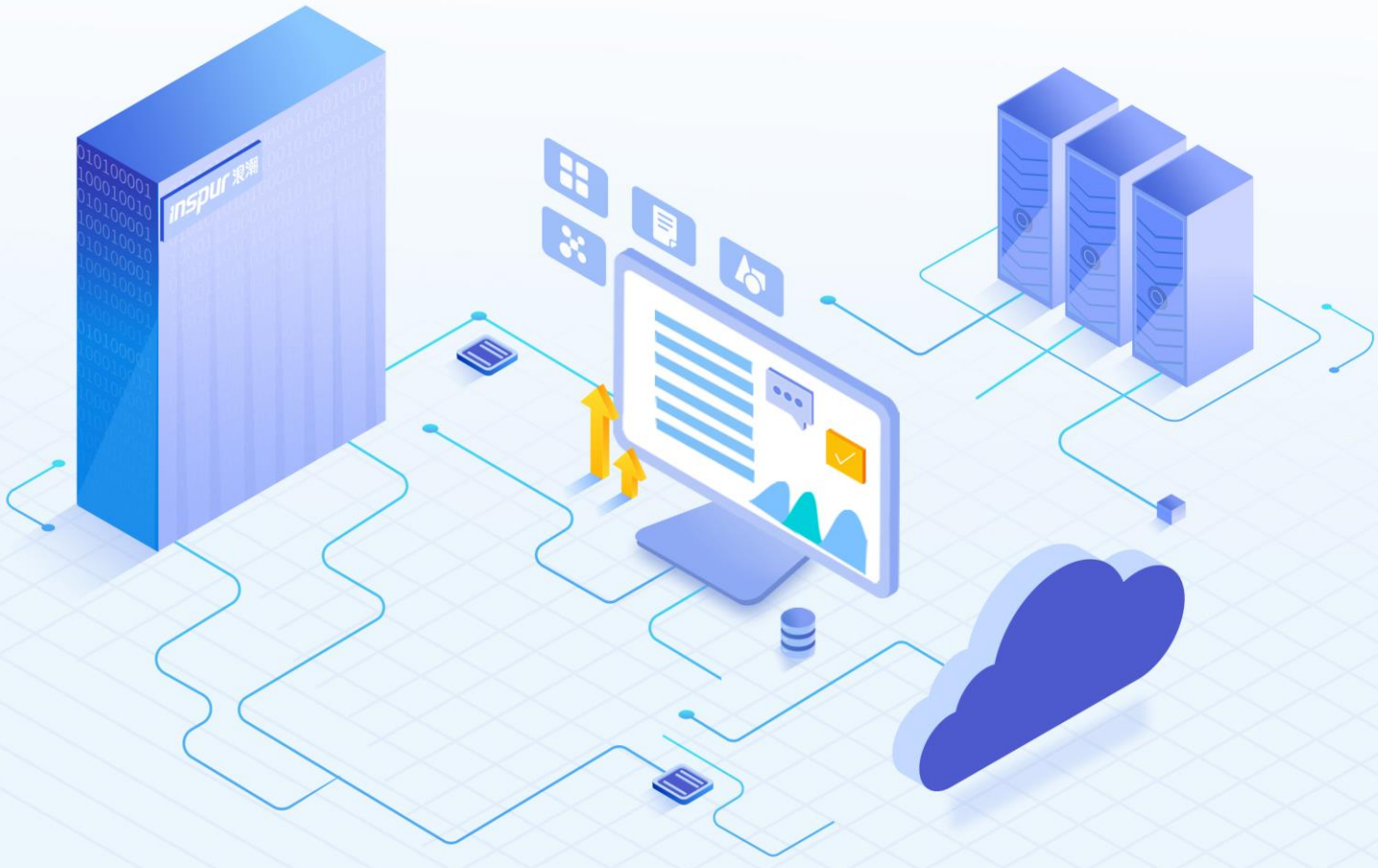
浪潮存储平台

K8sCSIPlugin 主机插件用户手册

文 档 版 本　2.0

发 布 日 期　2022-03-25

适 用 版 本　V2.2.0



尊敬的用户：

衷心感谢您选用浪潮存储！浪潮存储秉承“云存智用 运筹新数据”的新存储之道，致力于为您提供符合新数据时代需求的存储产品和解决方案。

本手册用于帮助您更详细地了解和便捷地使用本存储，涉及的截图仅为示例，最终界面请以实际设备显示的界面为准。

由于产品版本升级或其他原因，本手册内容会不定期进行更新，如有变动恕不另行通知。除非另有约定，本手册仅作为使用指导，本手册中的所有陈述、信息和建议不构成任何明示或暗示的担保。

如果您对本手册有任何疑问或建议，请向浪潮电子信息产业股份有限公司垂询。

技术服务电话：4008600011

地　　址：中国济南市浪潮路 1036 号
浪潮电子信息产业股份有限公司

邮　　编：250101

使用声明

在您正式使用本存储之前，请先阅读以下声明。只有您阅读并且同意以下声明后，方可正式开始使用本存储。如果您对以下声明有任何疑问，请和您的供货商联系或直接与我们联系。如您在开始使用本系统前未就以下声明向我们提出疑问，则默认您已经同意了以下声明。

1. 请不要自行拆卸本存储的机箱及机箱内任何硬件设备。在本存储出现任何硬件故障或您希望对硬件进行任何升级时，请您将机器的详细硬件配置反映给我们的客户服务中心。
2. 请不要将本存储的设备与任何其他型号的相应设备混用。本存储的内存、CPU、CPU 散热片、风扇、硬盘托架、硬盘等都是特殊规格的。
3. 在使用本存储时遇到任何软件问题，请您首先和相应软件的提供商联系。由提供商和我们联系，以方便我们共同沟通和解决您遇到的问题。对于数据库、网络管理软件或其他网络产品的安装、运行问题，我们尤其希望您能够这样处理。
4. 上架安装本存储前，请先仔细阅读相关产品手册中的快速安装指南。我们致力于产品功能和性能的持续提升，部分功能及操作与手册描述可能会有所差异，但不会影响使用。如果您有任何疑难问题，请与我们的客户服务中心联系。
5. **我们特别提醒您：在使用过程中，注意对您的数据进行必要的备份。**
6. 本存储为 A 级产品，在生活环境中可能会造成无线电干扰，需要您对其干扰采取切实可行的措施。
7. 请仔细阅读并遵守本手册的安全声明和安全细则。
8. 本手册中涉及的各软、硬件产品的标识、名称版权归产品的相应公司拥有。

以上声明中，“我们”指代浪潮电子信息产业股份有限公司；浪潮电子信息产业股份有限公司拥有对以上声明的最终解释权。

安全声明

我们非常重视数据安全和隐私，且一如既往地严密关注产品和解决方案的安全性，为您提供更满意的服务。在您正式使用本存储之前，请先阅读以下安全声明。

1. 为了保护您的数据隐私，在调整存储产品用途或淘汰存储设备时，请您将存储软件恢复固件出厂设置、删除信息、清除日志。同时，建议采用第三方安全擦除工具对存储软件所在的系统盘进行全面安全擦除。
2. 您购买的存储产品业务运营或故障定位的过程中可能会获取或使用用户的某些个人数据（如告警邮件接收地址、IP 地址）。因此，您有义务根据所适用国家或地区的法律法规制定必要的用户隐私政策，并采取足够的措施以确保用户的个人数据受到充分的保护。
3. 如需获取本存储开源软件声明，请直接联系我们的客户服务人员。
4. 本存储的某些安全特性需要您自行配置，如认证、传输加密、存储数据加密等，这些配置操作可能会对存储的性能和使用方便性造成一定影响。您可以根据应用环境，权衡是否进行安全特性配置。
5. 本存储自带了部分用于生产、装备、返厂检测维修的接口、命令及定位故障的高级命令，如使用不当，可能会导致设备异常或者业务中断，不建议您自行使用。如需使用，请联系我们的客户服务人员。
6. 我们已全面建立产品安全漏洞应急和处理机制，确保第一时间处理产品安全问题。若您在存储产品使用过程中发现任何安全问题，或者寻求有关产品安全漏洞的必要支持，请直接联系我们的客户服务人员。

以上声明中，“我们”指代浪潮电子信息产业股份有限公司；浪潮电子信息产业股份有限公司拥有对以上声明的最终解释权。

安全细则

在使用本存储时，若操作不当，可能会危及您的人身安全。为避免发生意外，在正式使用本存储之前，请务必认真阅读以下安全细则，严格按照要求进行操作。

1. 本存储中的电源设备可能会产生高电压和危险电能，从而导致人身伤害。请勿自行卸下主机盖以拆装、更换系统内部的任何组件。除非另外得到我们的通知，否则只有经过我们培训的维修技术人员才有权拆开主机盖及拆装、更换内部组件。
2. 请将设备连接到适当的电源，仅可使用额定输入标签上指明的外部电源为设备供电。为保护您的设备免受电压瞬间升高或降低所导致的损坏，请使用相关的稳压设备或不间断电源设备。
3. 如果必须使用延长线缆，请使用配有正确接地插头的三芯线缆，并查看延长线缆的额定值，确保插入延长线缆的所有产品的额定电流总和不超过延长线缆额定电流限制的百分之八十。
4. 请务必使用随机配备的供电组件，如电源线、电源插座（如果随机配备）等。为了本存储及使用者的安全，切勿随意更换电源线缆或插头。
5. 为防止因系统漏电而造成电击危险，请务必将本存储和外围设备的电源电缆插入已正确接地的电源插座。在未安装接地导线及不确定是否已有适当接地保护的情况下，请勿操作和使用本存储，并及时与电工联系。
6. 切勿将任何物体塞入本存储的开孔处，否则，可能会导致内部组件短路而引起火灾或电击。
7. 请将本存储置于远离散热片和有热源的地方，切勿堵塞通风孔。
8. 切勿在高潮湿、高灰尘的环境中使用本存储，切勿让食物或液体散落在系统内部或其它组件上。
9. 使用错误型号的电池会有爆炸的危险，需要更换电池时，请先向制造商咨询并使用与制造商推荐型号相同或相近的电池。切勿拆开、挤压、刺戳电池或使其外部接点短路。不要将其丢入火中或水中，也不要暴露在温度超过 60 摄氏度的环境中。请勿尝试打开或维修电池，务必合理处置用完的电池，不要将用完的电池及可能包含电池的电路板及其它组件与其它废品放在一起。有关电池回收政策请与当地废品回收处理机构联系。
10. 由于 3U、4U 和 5U 设备过重，建议使用本公司官方选配导轨上架。如果您使用自备导轨，请联系本公司客服，确保设备可以正常上架。在您使用自备导轨上架本设备前，请确认该导轨承载重量大于 150kg，否则，请务必使用本公司官方选配的导轨。若使用非本公司官方选配的导轨上架本设备，可能导致自备导轨的承载重量小于设备重量，造成设备上架失效等风险，对此，本公司将不承担任何责任或义务。

以上内容中，“我们”指代浪潮电子信息产业股份有限公司；浪潮电子信息产业股份有限公司拥有对以上内容的最终解释权。

目录

1 功能描述

1.1 基本介绍

K8sCSIPlugin 插件使得本存储可以为 Kubernetes 集群中的应用自动地提供持久化存储。通过 K8sCSIPlugin 插件，可以直接使用 Kubernetes 的管理命令自动在本存储上创建、管理卷/共享和快照，并将存储上的卷/共享提供给应用服务使用，用于服务的持久化数据存储。

Kubernetes 集群中的一个服务需要使用本存储提供的卷/共享时，首先在 Kubernetes 上创建一个 PVC（Persistent Volume Claim）资源信息。Kubernetes 根据 PVC 中的信息，通过接口将消息传递给 instorage-csi。instorage-csi 收到消息后会自动在存储上创建满足要求的卷/共享，并在 Kubernetes 上生成一个 PV（Persistent Volume）信息，之后服务就可以使用该卷/共享。Kubernetes 删除 PV 信息时，消息也将传递给 instorage-csi，instorage-csi 根据里面的具体信息，自动将存储上与之对应的卷/共享删除。

Kubernetes 集群中的一个使用持久化存储的服务启动时，需要先将服务所使用的持久化存储映射到某个主机的指定目录上，然后服务才可以使用这些持久化存储。在这个过程中，Kubernetes 会调用与相应 PV 信息对应的卷挂载/卸载插件，K8sCSIPlugin 插件中实现了该功能。当 Kubernetes 需要挂载一个卷时，instorage-csi 会自动在存储上完成卷与主机信息的绑定，并在主机端将对应的卷扫描出来。根据需要，自动对卷进行格式化，然后将卷上的文件系统挂载到 Kubernetes 指定的目录上。然后，Kubernetes 便可以将该目录映射到服务的容器中供服务使用。

K8sCSIPlugin 插件支持 FC 和 iSCSI 存储接口协议，能够实现对卷、快照、共享等的管理，详细说明如表 1-1 所示。

表 1-1 K8sCSIPlugin 插件的功能

存储类型	功能模块	操作
块存储	卷管理	创建普通卷、镜像卷、双活卷
		克隆卷（基于卷克隆、基于快照克隆）
		在线/离线扩容卷
		删除卷
	卷-主机管理	挂载卷到主机（单节点读写，多节点只读）

		挂载卷上的文件系统到主机
		从主机上卸载卷上的文件系统
		从主机上卸载卷
	快照管理	创建快照
		删除快照
	多路径	支持多路径设备
文件存储	共享管理	创建共享（CIFS 和 NFS）
		在线/离线扩容共享
		删除共享
	共享主机管理	挂载共享到主机
		挂载共享上的文件系统到主机
		从主机上卸载共享上的文件系统
		从主机上卸载共享

1.2 约束与限制

版本限制

- Kubernetes 版本：1.15~1.21
- Linux 内核版本：4.4 及以上
- 操作系统版本：CentOS 7.x

共享数量限制

存储系统为 5.1.0.x 及以上版本时，支持创建的最大共享数为 4*N 个（N 为控制器个数）。

适用存储产品

K8sCSIPlugin 插件适用的存储产品如表 1-2 所示。

表 1-2 适用存储产品

产品系列	产品型号

G2 系列	AS2150G2&AS2200G2&AS2600G2&AS5300G2&AS5500G2&AS5600G2&AS5800G2&AS6800G2
G2-F 系列	AS2600G2-F&AS5300G2-F&AS5500G2-F&AS5600G2-F&AS5800G2-F&AS6800G2-F
G5 系列	AS5300G5&AS5500G5&AS5600G5&AS5800G5&AS6800G5&HF5000G5&HF6000G5&HF8000G5
G5-I 系列	AS18000G5-I&HF18000G5-I
其他	HF5500

1.3 应用场景

K8sCSIPlugin 插件主要包括 Identity、Controller、Node 三个服务。

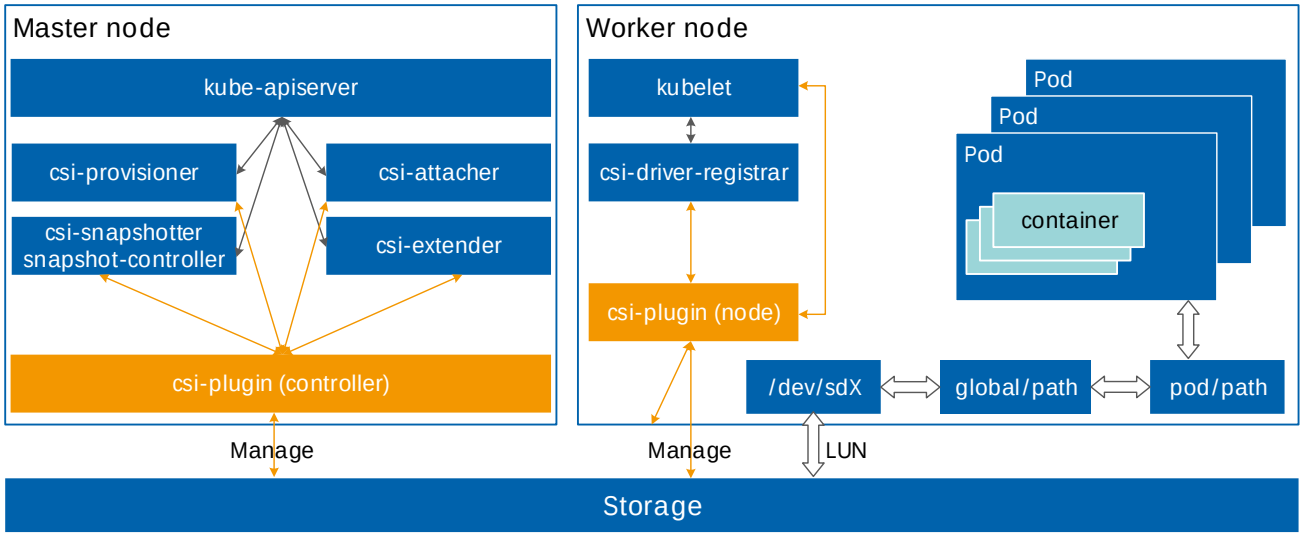
- Identity 服务负责获取插件的信息、能力、状态。
- Controller 服务负责创建、删除、克隆卷/共享和创建、删除快照。
- Node 服务负责挂载、卸载卷/共享。

K8sCSIPlugin 插件支持 all-in-one、controller、nodeworker 三种工作模式。

- all-in-one 模式包含上述三个服务，部署在 Kubernetes 的 Master 节点。
- controller 模式包含 Identity 和 Controller 两个服务，部署在 Kubernetes 的 Master 节点。
- nodeworker 模式包含 Identity 和 Node 两个服务，部署在 Kubernetes 的所有节点。

Kubernetes 集群通过 sidecar 容器与 K8sCSIPlugin 插件进行对接，sidecar 容器服务是为了方便 K8sCSIPlugin 插件与 Kubernetes 集群进行对接开发的桥接服务，本质上属于 Kubernetes CSI 接口支持的一部分。Sidecar 容器按照 Kubernetes 的实现机制与 Kubernetes 进行交互，并按照 CSI 接口定义的规范调用 K8sCSIPlugin 插件。K8sCSIPlugin 插件的应用拓扑图如图 1-1 所示。

图 1-1 K8sCSIPlugin 插件应用拓扑图



上图中，controller 模式的 K8sCSIPlugin 插件通过 csi-provisioner、csi-attacher、csi-snapshotter、snapshot-controller、csi-extender 等 sidecar 服务接收 kubernetes 集群的任务，并与存储交互完成管理任务。nodeworker 模式的 K8sCSIPlugin 插件通过 sidecar 服务 csi-driver-registrar 注册到 kubernetes 集群工作节点的 kubelet 服务中，后续 kubelet 服务直接调用 nodeworker 模式的 K8sCSIPlugin 插件来完成工作节点卷挂载/卸载相关任务。挂载设备时，nodeworker 模式的 K8sCSIPlugin 插件会根据主机信息在存储上映射卷，并在主机上扫描挂载设备。卸载设备时，nodeworker 模式的 K8sCSIPlugin 插件卸载和删除设备，并在存储上解除卷映射。

2 安装与部署

为了在 Kubernetes 集群中使用本存储，需要提前完成存储的初始化，并将 K8sCSIPlugin 插件部署在集群中。

- all-in-one 模式和 controller 模式的插件部署在 Kubernetes 的 Master 节点。
- nodeworker 模式的插件部署在 Kubernetes 的所有节点。

说明

- 部署 K8sCSIPlugin 插件前，需要确保 Kubernetes 集群状态正常。
- 部署时，无需重启 Kubernetes 服务。
- 升级时，先删除插件容器对应的 pods 并修改部署文件，然后重新创建 pods。

2.1 搭建 Kubernetes 集群

Kubernetes 是一个开源的容器编排管理平台。Kubernetes 集群的搭建过程、使用方法请参考官方网站 https://kubernetes.io/。

由于 Kubernetes 是一个开源系统，各个厂商可以基于 Kubernetes 发行自己的版本。不同厂商版本的使用，请参考各厂商提供的使用手册及相关文档。

2.2 安装 K8sCSIPlugin 插件

请按如下顺序完成 K8sCSIPlugin 插件的安装部署，每个步骤的详细说明见后续对应内容模块。

1. 解压安装包。
2. 安装软件包。
3. 拉取镜像。
4. 加载镜像。
5. 制作镜像。
6. 增加 kube-apiserver 服务的启动参数。
7. 增加 kube-controller-manager 服务的启动参数。
8. 增加 kubelet 服务的启动参数。
9. 创建 RBAC 权限信息。

10. 创建存储配置信息。
11. **可选**：创建 v1beta1 版本的快照资源。
12. 部署 K8sCSIPlugin 插件。

解压安装包

在官网获取 K8sCSIPlugin 插件安装包，解压并查看安装包中的文件。

- ./csiplugin/csiplugin 文件：K8sCSIPlugin 插件。
- ./csiplugin/Dockerfile 文件：用于构建 K8sCSIPlugin 插件镜像。
- ./csiplugin/deploy 目录：用于存放部署文件。
- ./csiplugin/demo 目录：用于存放示例文件。

```
root@lab:~/workspace$ ls
K8sPlugin_V2.1.0.Build20200114_amd64.tar.gz
root@lab:~/ K8sPlugin_V2.1.0.Build20200114_amd64# tree
├─── csiplugin
│    └── csiplugin
│    └── Dockerfile
│    └── deploy
│        └─── configMap.yaml
│        └─── csi-rbac.yaml
│        └─── gen-deploy-file-for-snapshot-v1alpha1.sh
│        └─── gen-deploy-file-for-snapshot-v1beta1.sh
│        └─── README.md
│        └─── images
│            └─── centos-7.6.1812.for-csiplugin.docker-image
│            └─── csi-attacher-v2.2.0
│            └─── csi-node-driver-registrar-v1.3.0
│            └─── csi-provisioner-v1.4.0
│            └─── csi-provisioner-v1.6.0
│            └─── csi-resizer-v0.5.0
│            └─── csi-resizer-v0.1.0
│            └─── csi-snapshotter-v1.2.2
```

```
|               └──── csi-snapshotter-v2.1.0
|               └──── snapshot-controller-v2.1.0
|    └── demo
|          └──── gen-demo-file-for-snapshot-v1alpha1.sh
|          └──── gen-demo-file-for-snapshot-v1beta1.sh
```

安装软件包

- 使用 iSCSI 方式对接存储时，需要在每个 K8s 节点上执行以下命令安装 iSCSI 客户端（版本号 6.2.0.874-11.el7~6.2.0.874-17.el7）：

  ```
  yum -y install iscsi-initiator-utils
  ```

  执行以下命令启动 iscsid 服务：

  ```
  systemctl enable iscsid
  systemctl start iscsid
  ```

- 使用多路径软件聚合单路径时，需要在每个 K8s 节点上执行以下命令安装多路径：

  ```
  yum -y install device-mapper device-mapper-multipath
  ```

  执行如下命令将配置文件 multipath.conf 拷贝到/etc 目录下，multipath.conf 文件的修改请参考“3.2 启用多路径”。

  ```
  cp /usr/share/doc/device-mapper-multipath-*.*.*/multipath.conf /etc
  ```

  执行如下命令启动多路径：

  ```
  modprobe dm-multipath
  modprobe dm-round-robin
  modprobe dm-service-time
  systemctl enable multipathd
  systemctl start multipathd
  ```

- 对接存储的文件设备（NFS）时，需要在每个 K8s 节点上执行以下命令安装 NFS：

  ```
  yum -y install nfs-utils
  ```

- 对接存储的文件设备（CIFS）时，需要在每个 K8s 节点上执行以下命令安装 CIFS：

  ```
  yum -y install cifs-utils
  ```

拉取镜像

部署过程依赖 sidecar 镜像，不同 kubernetes 版本对应的 sidecar 镜像（个数和版本）不同，二者的对应关系请参考“2.4 sidecar 镜像版本兼容性说明”。

根据 kubernetes 的版本，分别拉取 5 或 6 个 sidecar 镜像，以拉取 csi-snapshotter v1.2.2 为例，命令如下：

```
docker load –i csi-snapshotter-v1.2.2
```

拉取完成后，可执行以下命令查看已拉取的镜像：

```
docker images | grep csi
```

说明：拉取镜像后，查询到的镜像名为空时，可使用 docker tag 命令手动为镜像打上对应的 tag。

加载镜像

执行以下命令加载镜像：

```
docker load -i centos-7.6.1812.for-csiplugin.docker-image
```

加载完成后，可执行以下命令查看已加载的镜像。

```
docker images | grep centos
```

说明：加载后，查询到的镜像名为空时，可使用 docker tag 命令手动为镜像打上对应的 tag。

制作镜像

在第一个 csiplugin 目录下执行以下命令，注意命令中的“.”。

```
chmod -R 777 csiplugin
docker build -t csiplugin:v2.1.0 -f Dockerfile .
```

制作完成后，可执行以下命令查看已制作的镜像：

```
docker images | grep csiplugin
```

增加 kube-apiserver 服务的启动参数

feature-gates 的支持情况请参考“2.5 feature-gates 兼容性说明”。

对于 K8s-v1.15 和 K8s-v1.16，在/etc/kubernetes/manifests/kube-apiserver.yaml 文件中增加以下代码：

```
--feature-gates=VolumeSnapshotDataSource=true
--feature-gates=VolumePVCDataSource=true
--feature-gates=ExpandInUsePersistentVolumes=true
--feature-gates=ExpandCSIVolumes=true
```

增加完成后，kube-apiserver 服务会自动重启。执行以下命令查看是否生效：

```
ps aux | grep kube-apiserver
```

增加 kube-controller-manager 服务的启动参数

feature-gates 的支持情况请参考“2.5 feature-gates 兼容性说明”。

在/etc/kubernetes/manifests/kube-controller-manager.yaml 文件中增加以下代码：

```
--feature-gates=ExpandInUsePersistentVolumes=true
--feature-gates=ExpandCSIVolumes=true
```

增加完成后，kube-controller-manager 服务会自动重启。执行以下命令查看是否生效：

```
ps aux | grep kube-controller-manager
```

增加 kubelet 服务的启动参数

feature-gates 的支持情况请参考“2.5 feature-gates 兼容性说明”。

1. 在/usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf 文件中增加以下代码：

   ```
   --feature-gates=ExpandInUsePersistentVolumes=true
   --feature-gates=ExpandCSIVolumes=true
   ```

2. 增加完成后，手动执行以下命令重启 kubelet 服务：

   ```
   systemctl daemon-reload
   systemctl restart kubelet
   ```

3. 执行以下命令查看是否生效：

   ```
   ps aux | grep kubelet
   ```

创建 RBAC 权限信息

在 csiplugin/deploy 目录下，执行以下命令创建 RBAC 权限信息：

```
kubectl create -f csi-rbac.yaml
```

创建存储配置信息

参考“3.3K8sCSIPlugin 插件存储配置文件说明”修改 configMap.yaml 中的存储信息。在 csiplugin/deploy 目录下，执行以下命令创建存储配置信息：

```
kubectl create –f configMap.yaml
```

创建 v1beta1 版本的快照资源

K8s-v1.15 和 K8s-v1.16 不支持 v1beta1 版本的快照资源，可直接转至下一步“部署 K8sCSIPlugin 插件”。

对于 K8s-v1.17 及以上版本，执行以下命令创建 v1beta1 版本的快照资源：

```
kubectl apply -f snapshot/external-snapshotter-2.1-crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f snapshot/external-snapshotter-2.1-crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl apply -f snapshot/external-snapshotter-2.1-crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

部署 K8sCSIPlugin 插件

- 对于 K8s-v1.15 和 K8s-v1.16：
  a. 在 csiplugin/deploy 目录下，执行以下命令创建部署文件 csi-instorage-deploy.yaml：
     ```
     ./gen-deploy-file-for-snapshot-v1alpha1.sh
     ```
  b. 修改部署文件 csi-instorage-deploy.yaml 中 instorage-csi 部分的镜像名称，请参考步骤“制作镜像”的镜像命名。
  c. 执行以下命令部署 K8sCSIPlugin 插件：
     ```
     kubectl create -f csi-instorage-deploy.yaml
     ```
  d. 执行以下命令查看插件部署是否成功：
     ```
     kubectl get pods
     ```

图 2-1 查看部署结果

```
[root@node1 csiplugin]# kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
instorage-csi-controller-85b897477c-f9trg   5/5     Running   6          7d23h
instorage-csi-node-kvpgw                2/2     Running   2          7d23h
nginx-01                                1/1     Running   1          7d23h
[root@node1 csiplugin]#
```

- 对于 K8s-v1.17 及以上版本：

a. 在 csiplugin/deploy 目录下，执行以下命令创建部署文件 csi-instorage-deploy.yaml：

```
./gen-deploy-file-for-snapshot-v1beta1.sh
```

b. 修改部署文件 csi-instorage-deploy.yaml 中 instorage-csi 部分的镜像名称，请参考步骤“制作镜像”的镜像命名。

c. 执行以下命令部署 K8sCSIPlugin 插件：

```
kubectl create -f csi-instorage-deploy.yaml
```

d. 执行以下命令查看插件是否部署成功：

```
kubectl get pods
```

图 2-2 查看部署结果

完成部署

至此，K8sCSIPlugin 插件部署完毕，后续可参考“3 功能配置与管理”进行配置。

说明

插件是以容器方式部署的，可执行以下命令查看日志：

```
kubectl logs ${pod_name} -c instorage-csi
```

2.3 升级 K8sCSIPlugin 插件

部署 K8sCSIPlugin 插件后，需要升级时，请参考以下说明完成升级操作。为方便描述，以如下目录为例进行说明。

- 旧版 K8sCSIPlugin 插件解压后的目录为：/home/K8sPlugin_V2.1.0.Build_Old_amd64
- 新版 K8sCSIPlugin 插件解压缩后的目录为：/home/K8sPlugin_V2.1.1.Build_New_amd64

1. 在某个 K8s-master 节点上：将旧版 K8sCSIPlugin 插件的 configmap.yaml 文件拷贝到新版插件的对应目录下，进行覆盖。命令如下：

```
cp /home/K8sPlugin_V2.1.0.Build_Old_amd64/csiplugin/deploy/configmap.xml
/home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin/deploy
```

2. 在某个 K8s-master 节点上：将旧版 K8sCSIPlugin 插件的 csi-instorage-deploy.yaml 文件拷贝到新版插件的对应目录，进行覆盖，命令如下：

```
cp /home/K8sPlugin_V2.1.0.Build_Old_amd64/csiplugin/deploy/csi-instorage-deploy.yaml
/home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin/deploy
```

3. 在某个 K8s-master 节点上：卸载旧版 K8sCSIPlugin 插件，命令如下：

```
kubectl delete -f /home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin/deploy/csi-instorage-deploy.yaml
```

4. 在每个 K8s 节点上：在目录/home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin 下制作新镜像，命令如下：

```
chmod -R 755 csiplugin
docker build -t csiplugin:v2.1.1 -f Dockerfile .
```

5. 完成新镜像的制作后，执行如下命令查看已制作的镜像。该示例中制作的新镜像为 csiplugin:v2.1.1。

```
docker images | grep csiplugin
```

6. 在某个 K8s-master 节点上：修改 csi-instorage-deploy.yaml 文件。

   在 /home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin/deploy 目录下，将 csi-instorage-deploy.yaml 中容器 instorage-csi 的镜像修改为 csiplugin:v2.1.1（共两处）。

   说明：镜像及版本必须与步骤 5 制作的新镜像保持一致。

7. 在某个 K8s-master 节点上：安装新版本的 K8sCSIPlugin。命令如下：

```
kubectl apply -f /home/K8sPlugin_V2.1.1.Build_New_amd64/csiplugin/deploy/csi-instorage-deploy.yaml
```

2.4 sidecar 镜像版本兼容性说明

K8sCSIPlugin 插件部署过程需要使用 sidecar 镜像，sidecar 镜像的版本需与 kubernetes 的版本兼容。

K8s-v1.15 和 K8s-v1.16 需要使用 5 个 sidecar 镜像，推荐的版本配置如表 2-1 所示。

表 2-1 sidecar 镜像版本兼容性列表

kubernetes	csi-provisioner	csi- attacher	csi-snapshotter	csi- resizer	csi-node-driver-registrar
client：v1.15.2 server: v1.15.0	v1.4.0	v2.2.0	v1.2.2	v0.5.0	v1.3.0
client：v1.16.0	v1.4.0	v2.2.0	v1.2.2	v0.5.0	v1.3.0

server：v1.16.0					

K8s-v1.17 及以上版本需要使用 6 个 sidecar 镜像，推荐的版本配置如表 2-2 所示。

表 2-2 sidecar 镜像版本兼容性列表（volumesnapshot 为 v1beta1）

kubernetes	csi-provisioner	csi-attacher	csi-snapshotter	snapshot-controller	csi-resizer	csi-node-driver-registrar
client：v1.17.0 server: v1.17.0	v1.6.0	v2.2.0	v2.1.0	v2.1.0	v0.5.0	v1.3.0
client：v1.18.0 server：v1.18.0	v1.6.0	v2.2.0	v2.1.0	v2.1.0	v0.5.0	v1.3.0
client：v1.19.0 server：v1.19.0	v1.6.0	v2.2.0	v2.1.0	v2.1.0	v0.5.0	v1.3.0
client：v1.20.0 server：v1.20.0	v1.6.0	v2.2.0	v2.1.0	v2.1.0	v0.5.0	v1.3.0
client：v1.21.0 server：v1.21.0	v1.6.0	v2.2.0	v2.1.0	v2.1.0	v0.5.0	v1.3.0

2.5 feature-gates 兼容性说明

K8sCSIPlugin 插件对块设备支持克隆卷、扩容卷、创建快照，这些特性要求 kubernetes 开启相应的 feature-gates，相关 feature-gates 在 kubernetes 不同版本的支持情况如表 2-3 所示。

表 2-3 feature-gates 版本兼容性列表

特性	默认值	阶段	开始版本	结束版本
VolumeSnapshotDataSource	false	Alpha	1.12	1.16
	true	Beta	1.17	-
VolumePVCDataSource	false	Alpha	1.15	1.15
	true	Beta	1.16	1.17
	true	GA	1.18	-
ExpandInUsePersistentVolumes	false	Alpha	1.11	1.14

	true	Beta	1.15	-
ExpandCSIVolumes	false	Alpha	1.14	1.15
	true	Beta	1.16	-

3 功能配置与管理

为了在 Kubernetes 集群中使用本存储，需要提前完成存储的初始化并创建存储池。确保存储可以在集群中的各节点被访问，存储数据层通道（iSCSI、FC）可以正常使用，然后根据集群的具体信息修改配置文件。

3.1 配置网络

配置管理 IP

注意

- 确保插件所在节点具有存储系统的 SSH 访问权限。
- 确保本存储至少已配置 iSCSI 或 FC 中的一种。

插件需要访问本存储的管理接口，使用 SSH 的方式与管理接口通信。需要配置本存储系统的 IP、SSH 端口。

配置 iSCSI 网络

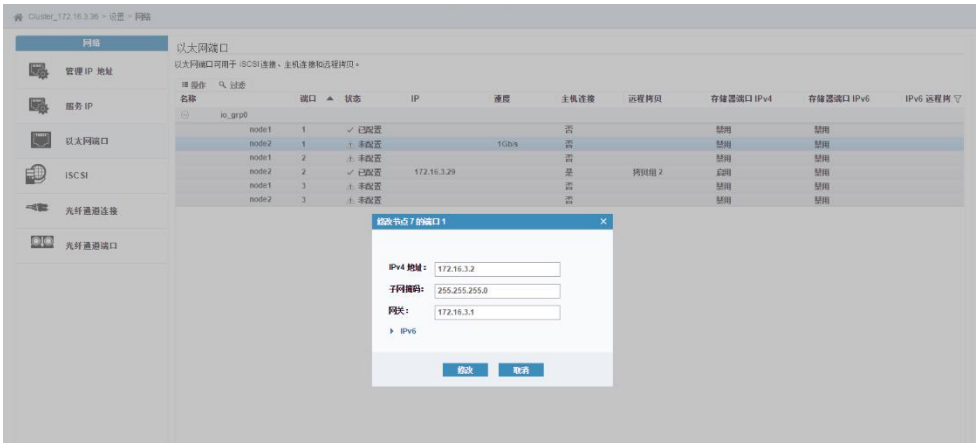
注意：使用 iSCSI 时，确保各工作节点与存储系统有可供访问的 iSCSI 网络。

使用 iSCSI 时，需要每个存储节点至少有一个 iSCSI 的 IP 地址。插件会直接从存储系统中获取 iSCSI IP，不需要单独提供，但是必须在存储管理系统中设置 iSCSI 端口的 IP，设置方法如下：

在“设置 > 网络 > 以太网端口”页面中，选择已经连通的端口，单击鼠标右键或“操作”打开操作菜单。单击“修改 IP 设置”为以太网端口设置 IP 地址，如图 3-1 所示。

说明：iSCSI IP 不能与 NAS IP 配置成相同地址。

图 3-1 添加 iSCSI 端口 IP



配置 FC 网络

注意：使用 FC 时，确保各工作节点与存储系统有可供访问的 FC 网络。

使用 FC 时，需要每个存储节点至少配置有一个 WWPN 端口。插件会使用所有可用的 WWPN 端口将卷挂载至目标主机。插件将直接从存储系统中获取 WWPN，不需要单独提供。

3.2 启用多路径

为了提升 SAN 存储卷的可靠性，在生产环境中通常会启用多路径。在 Kubernetes 环境中，如果多路径配置不当，在实际使用中可能会使用到单路径设备，则该路径出现损坏时可能会产生 I/O 错误。

在 Linux 环境下，本存储利用 Linux 系统自带的 multipathd 服务进行多路径聚合。要启用多路径，首先需要保证 Kubernetes 各节点均按照要求部署安装 multipathd 服务。然后配置 Kubernetes 环境中各工作节点的多路径，以正确使用多路径。

1. 在 multipath.conf 配置文件中配置设备黑名单。

   说明

   - 不能将本存储提供的设备路径加入设备黑名单。
   - multipath.conf 配置文件的具体路径，请咨询 Kubernetes 容器平台提供商。Linux 系统下，该文件默认路径为/etc/multipath.conf。

   在 Kubernetes 环境中，一个工作节点上会同时挂载多个卷，每个卷又有多条路径，从而使得节点上的 sdX 设备非常多。多路径配置文件中的 blacklist 配置组中的参数会过滤符合 blacklist 条件的路径。配置的过滤设备不当可能会导致部分路径无法进行多路径聚合。

   例如，将 devnode 参数配置成“^sda”时，会过滤 sdaa、sdab 等路径，不进行路径聚合，导致路径缺

失。应该配置为“^sda$”，即仅将 sda 设备屏蔽掉。

在 Linux 环境中，执行以下命令获取 multipath.conf 配置文件的具体用法。

```
man 5 multipath.conf
```

2. 在 multipath.conf 配置文件配置本存储推荐的设备参数。

目前，本存储推荐的多路径配置已经合入到多路径工具的社区版本中。使用旧版本多路径工具时，需要在配置文件中加入推荐的多路径配置，即在 devices 配置组中增加本存储的 device 配置内容，如下：

```
device{
        vendor "INSTORAGE"
        product "MCS"
        path_grouping_policy group_by_prio
        path_selector "round-robin 0"
        prio alua
        path_checker tur
        failback immediate
        no_path_retry “20”
        rr_min_io 1000
        dev_loss_tmo 120
        fast_io_fail_tmo 5
}
```

3.3 K8sCSIPlugin 插件存储配置文件说明

K8sCSIPlugin 插件的存储配置文件为插件目录中的 deploy/configMap.yaml 文件。具体配置和说明（表 3-1）如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: instorage-01
data:
  instorage.yaml: |
    log:
      enabled: false
```

```
    logdir: log
    level: ""
    logrotatemaxsize: 0
  host:
    link: iscsi
    forceUseMultipath: false
    scsiScanRetryTimes: 3
    scsiScanWaitInterval: 3
    iscsiPathCheckRetryTimes: 3
    iscsiPathCheckWaitInterval: 3
    multipathSearchRetryTimes: 3
    multipathSearchWaitInterval: 3
    multipathResizeDelay: 1
    fcScanLevel: lun
    hostnamePrefiex: ""
    hostCommandTimeout: 0
  storage:
  - name: storage-01
    type: AS18000
    host: 10.0.0.1:22
    username: username
    password: password
    shadow: ""
    deviceUsername: superuser
    devicePassword: "00000000"
    deviceShadow: ""
    barrierPath: ""
    portalTable:
      table: {}
      default: []
    task:
      maxCreateVolumeNumber: 4
      maxDeleteVolumeNumber: 4
      maxPublishVolumeNumber: 4
```

```
    maxUnpublishVolumeNumber: 4
    maxStageVolumeNumber: 4
    maxUnstageVolumeNumber: 4
    maxCreateSnapshotNumber: 4
    maxDeleteSnapshotNumber: 4
    maxExpandVolumeNumber: 4
```

表 3-1 K8sCSIPlugin 插件存储配置说明

配置名称	说明
log.enabled	开启插件日志（配置 true 或 flase 都会开启）
log.logdir	日志输出目录
log.level	日志输出级别，包括 debug、info、warning、error 和 fatal
log.logrotatemaxsize	日志文件滚动大小阈值（暂未启用）
host.link	数据通道连接类型 ● iscsi：使用 iSCSI 连接方式 ● fc：使用 FC 连接方式
host. forceUseMultipath	是否强制使用多路径 ● true：强制 ● false：不强制
host. scsiScanRetryTimes	SCSI 设备扫描尝试次数
host. scsiScanWaitInterval	SCSI 设备扫描失败后等待间隔，单位为秒
host. iscsiPathCheckRetryTimes	iSCSI 路径扫描检查尝试次数
host. iscsiPathCheckWaitInterval	iSCSI 路径扫描失败后等待间隔，单位为秒
host. multipathSearchRetryTimes	多路径设备查找重试次数
host. multipathSearchWaitInterval	多路径设备查找失败后等待间隔，单位为秒
host. multipathResizeDelay	在线扩容时，多路径 resize 命令延迟时间，单位为秒
host. attachExtendFileLockPath	挂载、卸载、扩容操作并发控制文件锁路径，默认使用配置文件
host.fcScanLevel	FC 环境下扫描主机的方式，可以设置为：

	● lun：默认值，即根据 lunID 进行扫描 ● ctl ● all
host.hostnamePrefiex	主机名前缀，默认是空
host.hostCommandTimeout	主机端命令的超时时间，默认为 5 分钟
storage[].name	存储名称，配置文件范围内唯一，区分多个存储。当前只支持一个存储
storage[].type	存储类型，必须配置且类型为 AS18000
storage[].host	存储 SSH 访问路径，格式为 IP:Port，如 10.0.0.1:22
storage[].username	存储访问时的用户名
storage[].password	存储访问时的密码明文
storage[].shadow	存储访问时的密码明文加密后的密文 ● 设置了 password 时，优先使用 password 设置的密码明文 ● 未设置 password 时，使用 shadow 设置的密码密文。执行以下命令获取密码密文： ./tools/cipher_helper -e [password]
storage[].deviceUsername	分布式存储第二层登录的用户名和密码，均为固定值，无需设置
storage[].devicePassword	
storage[].deviceShadow	存储访问时的第二层密码明文加密后的密文，无需设置
storage[].barrierPath	双活卷扩容命令出错后生成的屏障文件所在的目录，默认为插件程序所在目录 目前，双活卷扩容由插件组合调用存储端命令完成，多个命令调用中间出错以后，无法回滚，且存储端不能再执行对应主机上的卷挂载/卸载/双活卷扩容操作。因此，命令失败后会在该目录生成屏障文件，阻止后续任务执行。
storage[].portalTable.table	主机端口映射表，key 为 nodeID，value 为 portals
storage[].portalTable.default	默认主机端口的列表
storage[].task.maxCreateVolumeNumber	同一时刻对每个存储创建卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）

storage[].task.maxDeleteVolumeNumber	同一时刻对每个存储删除卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxPublishVolumeNumber	同一时刻对每个存储挂载卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxUnpublishVolumeNumber	同一时刻对每个存储卸载卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxStageVolumeNumber	同一时刻对每个存储准备卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxUnstageVolumeNumber	同一时刻对每个存储清理卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxCreateSnapshotNumber	同一时刻对每个存储创建快照的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxDeleteSnapshotNumber	同一时刻对每个存储删除快照的最大个数，默认为 4（如果设置成 0，自动更改为默认值）
storage[].task.maxExpandVolumeNumber	同一时刻对每个存储扩容卷的最大个数，默认为 4（如果设置成 0，自动更改为默认值）

4 在 Kubernetes 中使用存储

通过 K8sCSIPlugin 插件，能够在 Kubernetes 集群中使用本存储。Kubernetes 中使用持久化存储的方式为通过 PVC（Persistent Volume Claim）使用卷。

卷的整个生命周期包括创建、挂载、卸载、删除，分别对应创建 pvc、创建 pod、删除 pod、删除 pvc。执行 gen-demo-file-for-snapshot-v1beta1.sh 脚本可获取相关配置文件示例，其中${pool-name}为数据池。

```
./gen-demo-file-for-snapshot-v1beta1.sh -p ${pool_name}
```

下面以块存储为例，详细说明在 Kubernetes 中使用存储的方法。

4.1 对接单套存储创建卷及 POD

通过新建卷创建 POD 时，可以利用 K8sCSIPlugin 插件自动在存储上创建满足要求的卷/共享，并在删除对应 PV 时自动删除存储上对应的卷/共享，也可以手动创建、删除卷/共享以及创建、删除集群上对应的 PV。

通过新建卷创建 POD 的整体流程如下：

1. 创建 StorageClass 资源。
2. 创建一个 PVC 来请求一个具体的卷。
   K8sCSIPlugin 插件完成存储上卷/共享的创建以及 Kubernetes 中 PV 的创建后，PVC 会与对应的 PV 进行关联。
3. 创建 Pod 来使用该 PVC。

下面按顺序对每个过程进行详细介绍。

创建 StorageClass 资源

StorageClass 资源用于表示一种卷的类型，具有以下两个作用：

- 作为 PV 和 PVC 之间互相关联的纽带。
- 标识插件的名称和插件创建卷/共享过程中使用的参数信息。

在 Kubernetes 集群中创建 StorageClass 资源的详细过程如下：

1. 查看并修改配置文件 csi-sc.yaml，配置示例如下，其中：
   - provisioner 属性用于标识插件的名称，默认值为 csi-instorage，即使用 K8sCSIPlugin 插件。

- parameters 属性用于指定创建卷时的参数，参数的详细说明请参考“4.8 storageclass 配置文件及参数说明”。
- reclaimPolicy 属性用于指定 pv 的回收策略。
- allowVolumeExpansion 属性用于指定是否支持在线扩容，默认为 false 即不支持，true 为支持。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sc
provisioner: csi-instorage
parameters:
  devKind: block
  volPoolName: Pool0
  volThin: "true"
  volThinResize: "2"
  volThinGrainSize: "256"
  volThinWarning: "20"
reclaimPolicy: Delete
allowVolumeExpansion: false
```

2. 执行以下命令创建 StorageClass 资源。

```
kubectl apply -f csi-sc.yaml
```

返回信息如下：

```
storageclass.storage.k8s.io "csi-sc" created
```

3. 执行以下命令查看创建的 StorageClass 资源信息。

```
kubectl get sc
```

返回信息如下：

```
NAME     PROVISIONER    AGE
csi-sc   csi-instorage  33s
```

创建 PVC

在 Kubernetes 集群中创建 PVC，声明使用存储资源，详细过程如下：

1. 查看并修改配置文件 csi-pvc-dynamic.yaml，配置示例如下，其中：

- spec.accessModes 属性用于指定 pvc 的访问权限。spec.volumeMode 为 FileSystem 时，支持 ReadWriteOnce 和 ReadOnlyMany。spec.volumeMode 为 Block 时，支持 ReadWriteOnce、ReadOnlyMany 和 ReadWriteMany。
- spec.volumeMode 属性用于指定卷模式，支持 FileSystem 和 Block。
- spec.resources.requests.storage 属性用于指定卷大小。
- spec.storageClassName 属性用于指定所使用的 storageclass。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-01
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-sc
```

2. 执行以下命令创建 PVC。

```
kubectl apply -f csi-pvc-dynamic.yaml
```

返回信息如下：

```
persistentvolumeclaim " csi-pvc-01" created
```

3. 执行以下命令查看创建的 PVC 信息。

```
kubectl get pvc
```

返回信息如下：

```
NAME         STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
csi-pvc-01   Bound    pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            csi-sc         9s
```

该示例中创建了一个名称为 csi-pvc-01 的 PVC，K8sCSIPlugin 插件收到 PVC 创建的消息后，会根据 StorageClass 的参数信息在存储上创建对应的卷/共享，并在 Kubernetes 集群中创建对应的 PV 资源。之后，PVC 会与 PV 互相绑定，具体信息如下表所示。

表 4-1 对应的 PV 资源

PVC	对应 PV 名称	SC 名称
csi-pvc-01	pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd	csi-sc

查看存储上的卷/共享信息，确认 PV 对应的卷已经创建，如下图所示。

图 4-1 查看卷信息

4. 执行以下命令查看创建的 PVC 绑定的 PV 信息。

```
kubectl get pv
```

返回信息如下：

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            Delete           Bound    default/csi-pvc-01    csi-sc                  21h
```

创建 Pod

PVC 与 PV 互相绑定后，就能够利用 PVC 为 Pod 中的服务提供持久化存储。创建 Pod 并在 Pod 中使用 PVC 的详细过程如下：

1. 查看并修改配置文件 pod-use-dynamic-pvc.yaml，配置示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-01
spec:
  containers:
    - name: nginx
      image: nginx:v1
      ports:
        - containerPort: 80
      volumeMounts:
        - name: k8s-test-01
          mountPath: /mnt
  volumes:
```

```
  - name: k8s-test-01
    persistentVolumeClaim:
      claimName: csi-pvc-01
```

2. 执行以下命令创建 Pod。

```
kubectl create -f pod-use-dynamic-pvc.yaml
```

返回信息如下：

```
pod/nginx-01 created
```

该示例中创建了一个名称为 nginx-01 的 Pod，将 PVC（csi-pvc-01）所声明的卷映射到 Pod 中的 nginx 容器的/mnt 目录。

3. 执行以下命令查看创建的 Pod 信息。

```
kubectl get pods
```

返回信息如下：

```
NAME          READY   STATUS    RESTARTS   AGE
nginx-01      1/1     Running   0          46m
```

4. 查看工作节点，可以看到卷被正确地映射和挂载，如下图所示。

图 4-2 查看卷信息

```
[root@node1 dynamic]# ls -l /dev/disk/by-path | grep 241
lrwxrwxrwx 1 root root  9 12月 17 16:33 ip-100.7.46.117:3260-iscsi-iqn.2004-12.com.inspur:mcs.cluster241.node1-lun-0 -> ../../sde
lrwxrwxrwx 1 root root  9 12月 17 16:33 ip-100.7.46.118:3260-iscsi-iqn.2004-12.com.inspur:mcs.cluster241.node2-lun-0 -> ../../sdf
[root@node1 dynamic]# mount |grep pvc
/dev/mapper/36005076000d189c0d0000000000001e7 on /var/lib/kubelet/pods/87fdc933-2780-4b78-b7ee-2a22c95d43f1/volumes/kubernetes.io~csi/pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd/mount type ext4 (rw,relatime,stripe=8,data=ordered)
```

4.2 对接多套存储

K8sCSIPlugin 插件支持对接多套存储，且只需启动一套插件即可对接多套存储。修改存储配置文件 configMap.yaml、StorageClass 配置文件，然后参照“4.1 对接单套存储创建卷及 POD”进行其他配置，即可对接多套存储。修改方法如下：

1. 在存储配置文件 configMap.yaml 的 data.storage 中配置多套存储信息，示例如下：

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: instorage-01
data:
  instorage.yaml: |
    ......
    storage:
    - name: storage-01
      type: AS18000
      host: 10.0.0.1:22
      username: username
      password: password
    - name: storage-02
      type: AS18000
      host: 10.0.0.10:22
      username: username
      password: password
```

2. 在 StorageClass 配置文件中使用 devName 属性指定存储信息，示例如下：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-18000-block-sc
provisioner: csi-instorage
parameters:
  devName: storage-02
  devKind: block
  volPoolName: pool0
  volThin: "true"
reclaimPolicy: Delete
```

4.3 通过 PVC 创建快照

创建快照的过程与创建 PVC 的过程类似，会涉及以下三个扩展资源：

- VolumeSnapshotClass：相当于 SC
- VolumeSnapshot：相当于 PVC

- VolumeSnapshotContent：相当于 PV

首先需要创建 VolumeSnapshotClass 资源，然后再创建快照，详细说明如下。

创建 VolumeSnapshotClass 资源

VolumeSnapshotClass 资源用于表示一种快照类型，具有以下作用：

- 作为 VolumeSnapshot 和 VolumeSnapshotContent 之间互相关联的纽带
- 用于标识插件的名称和插件创建快照过程中使用的参数信息。

在 Kubernetes 集群中创建 VolumeSnapshotClass 资源的详细过程如下：

1. 查看并修改配置文件 vsc.yaml，配置示例如下，其中：
   - snapshotter 属性用于标识插件的名称，默认值为 csi-instorage，即使用 K8sCSIPlugin 插件。
   - deletionPolicy 属性用于指定删除策略。

```
kind: VolumeSnapshotClass
apiVersion: snapshot.storage.k8s.io/v1alpha1
metadata:
  name: csi-vsc
snapshotter: csi-instorage
deletionPolicy: Delete
```

2. 执行以下命令创建 VolumeSnapshotClass 资源。

```
kubectl create -f vsc.yaml
```

   返回信息如下：

```
volumesnapshotclass.snapshot.storage.k8s.io/csi-vsc created
```

3. 执行以下命令查看创建的 VolumeSnapshotClass 资源信息。

```
kubectl get volumesnapshotclass
```

   返回信息如下：

```
NAME       AGE
csi-vsc    12d
```

创建快照

在 Kubernetes 集群中创建 VolumeSnapshot 的详细过程如下：

1. 查看并修改配置文件 vs.yaml，配置示例如下，其中：

- spec.volumeSnapshotClassName 属性用于指定 VolumeSnapshotClass 资源。
- spec.source.persistentVolumeClaimName 属性用于指定源 PVC。

```
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: csi-vs-01
spec:
  volumeSnapshotClassName: csi-vsc
  source:
    persistentVolumeClaimName: csi-pvc-01
kind: PersistentVolumeClaim
```

2. 执行以下命令创建 VolumeSnapshot。

```
kubectl create -f vs.yaml
```

返回信息如下：

```
volumesnapshot.snapshot.storage.k8s.io/csi-vs-01 created
```

该示例中创建了一个名称为 csi-vs-01 的 VolumeSnapshot，K8sCSIPlugin 插件收到 VolumeSnapshot 创建的消息后，会根据 VolumeSnapshotClass 的参数信息和 Source 信息，在存储上创建对应的快照，并在 Kubernetes 集群中创建对应的 VolumeSnapshotContent 资源信息。之后，VolumeSnapshot 会与 VolumeSnapshotContent 互相绑定。

3. 执行以下命令查看创建的 VolumeSnapshot。

```
kubectl get volumesnapshotcontent
```

返回信息如下：

```
NAME                                               AGE
snapcontent-002bf754-261a-4ee9-86b8-6ab0796749c2   16s
```

4.4 通过 PVC 克隆 PVC

通过 PVC 方式克隆 PVC 就是通过卷克隆卷。通过 PVC 克隆 PVC 前，需先在 Kubernetes 集群中创建 PVC，详细说明请参考“4.1 对接单套存储创建卷及 POD”。

在 Kubernetes 集群中通过 PVC 方式克隆 PVC，详细过程如下：

1. 查看并修改配置文件 pvc-from-pvc.yaml，配置示例如下，其中：

- spec.dataSource.name 属性用于指定源 PVC，源 PVC 必须已存在。
- spec.dataSource.kind 属性用于指定数据源类型，设置为 PVC。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: csi-sc
  resources:
    requests:
      storage: 1Gi
  volumeMode: Filesystem
  dataSource:
    name: csi-pvc-01
    kind: PersistentVolumeClaim
```

2. 执行以下命令通过 PVC 方式克隆 PVC。

```
kubectl create -f pvc-from-pvc.yaml
```

返回信息如下：

```
persistentvolumeclaim "pvc-from-pvc" created
```

该示例中通过旧的名称为 csi-pvc-01 的 PVC 克隆新的名称为 pvc-from-pvc 的 PVC。

3. 执行以下命令查看 PVC 信息。

```
kubectl get pvc
```

返回信息如下：

```
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
csi-pvc-01    Bound    pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            csi-sc         41m
pvc-from-pvc  Bound    pvc-ad8006a7-8e1c-44cc-81ba-28062b20c0f7   1Gi        RWO            csi-sc         8s
```

4. 执行以下命令查看 PV 信息。

```
kubectl get pv
```

返回信息如下：

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS   REASON   AGE
pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            Delete           Bound    default/csi-pvc-01     csi-sc                  42m
pvc-ad8006a7-8e1c-44cc-81ba-28062b20c0f7   1Gi        RWO            Delete           Bound    default/pvc-from-pvc   csi-sc                  29s
```

4.5 通过快照克隆 PVC

通过 VolumeSnapshot 方式克隆 PVC 就是通过快照克隆卷。通过 VolumeSnapshot 方式克隆 PVC，需先在 Kubernetes 集群中创建 VolumeSnapshot，详细说明请参考“4.3 通过 PVC 创建快照”。

在 Kubernetes 集群中通过快照方式克隆 PVC，详细过程如下：

1. 查看并修改配置文件 pvc-from-snap.yaml，配置示例如下，其中：
   - spec.dataSource.name 属性用于指定源 Snapshot，源 snapshot 必须已存在。
   - spec.dataSource.kind 属性用于指定数据源类型，设置为 VolumeSnapshot。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: csi-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: csi-vs-01
    kind: VolumeSnapshot
apiGroup: snapshot.storage.k8s.io
```

2. 执行以下命令通过快照克隆 PVC。

```
kubectl create -f pvc-from-snap.yaml
```

   返回信息如下：

```
persistentvolumeclaim " pvc-from-snap " created
```

   该示例中通过名称为 csi-vs-01 的快照克隆新的名称为 pvc-from-snap 的 PVC。

3. 执行以下命令查看 PVC 信息。

```
kubectl get pvc
```

返回信息如下：

```
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
csi-pvc-01     Bound    pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            csi-sc         49m
pvc-from-snap  Bound    pvc-89db5dc8-147f-4168-82e7-c7c7c32e6d3d   1Gi        RWO            csi-sc         2m46s
```

4. 执行以下命令查看 PV 信息。

```
kubectl get pv
```

返回信息如下：

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   REASON   AGE
pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            Delete           Bound    default/csi-pvc-01      csi-sc                  49m
pvc-89db5dc8-147f-4168-82e7-c7c7c32e6d3d   1Gi        RWO            Delete           Bound    default/pvc-from-snap   csi-sc                  2m55s
```

4.6 离线扩容

扩容是指对卷进行扩容，在 Kubernetes 集群中体现在 PVC 上。离线扩容是指在卷未挂载使用的情况下，对卷进行扩容操作。离线扩容后，卷容量不会刷新，挂载使用后才会刷新为扩容后的容量。

离线扩容的整体流程如下：

1. 创建 StorageClass 资源。
2. 创建 PVC，声明使用存储资源。
3. 编辑 PVC。
4. 创建 Pod。

下面按顺序对每个过程进行详细介绍。

创建 StorageClass 资源

在 Kubernetes 集群中创建 StorageClass 资源的详细过程如下：

1. 查看并修改配置文件 csi-sc-resizer.yaml，配置示例如下。

   其中，allowVolumeExpansion 属性必须设置 true。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sc-resizer
provisioner: csi-instorage
parameters:
```

```
  volPoolName: Pool0
  volThin: "true"
  volThinResize: "2"
  volThinGrainSize: "256"
  volThinWarning: "20"
reclaimPolicy: Delete
allowVolumeExpansion: true
```

2. 执行以下命令创建支持控制的 StorageClass 资源。

```
kubectl create -f csi-sc-resizer.yaml
```

返回信息如下：

```
storageclass.storage.k8s.io/csi-sc-resizer created
```

3. 执行以下命令查看创建的 StorageClass 资源信息。

```
kubectl get sc
```

返回信息如下：

```
NAME              PROVISIONER        AGE
csi-sc-resizer    csi-instorage      33s
```

创建 PVC

在 Kubernetes 集群中创建 PVC，声明使用存储资源，详细过程如下：

1. 查看并修改配置文件 csi-pvc-resizer.yaml，配置示例如下：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: csi-pvc-resizer-01
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 2Gi
```

```
  storageClassName: csi-sc-resizer
```

2. 执行以下命令创建 PVC。

```
kubectl create -f csi-pvc-resizer.yaml
```

返回信息如下：

```
persistentvolumeclaim/csi-pvc-resizer-01 created
```

3. 执行以下命令查看 PVC 信息。

```
kubectl get pvc
```

返回信息如下：

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
csi-pvc-01          Bound    pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            csi-sc            16h
csi-pvc-resizer-01  Bound    pvc-05491ede-5087-438d-b254-b6f3b89fda02   2Gi        RWO            csi-sc-resizer    4s
```

4. 执行以下命令查看 PV 信息。

```
kubectl get pv
```

返回信息如下：

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                       STORAGECLASS      REASON   AGE
pvc-05491ede-5087-438d-b254-b6f3b89fda02   2Gi        RWO            Delete           Bound    default/csi-pvc-resizer-01  csi-sc-resizer             52s
pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            Delete           Bound    default/csi-pvc-01          csi-sc                     16h
```

编辑 PVC

在 Kubernetes 集群中编辑 PVC，详细过程如下：

1. 执行以下命令编辑 PVC，将 spec.resources.requests.storage 属性的值从 2Gi 改成 3Gi。

```
kubectl edit pvc csi-pvc-resizer-01
```

返回信息如下：

```
persistentvolumeclaim/csi-pvc-resizer-01 edited
```

2. 执行以下命令查看 PVC 信息。

```
kubectl get pvc csi-pvc-resizer-01
```

返回信息如下，此时 csi-pvc-resizer-01 的大小仍然是 2Gi。

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
csi-pvc-resizer-01  Bound    pvc-05491ede-5087-438d-b254-b6f3b89fda02   2Gi        RWO            csi-sc-resizer    5m33s
```

创建 Pod

在 Kubernetes 集群中使用扩容卷创建 POD，详细过程如下：

1. 查看并修改配置文件 pod-use-resizer-pvc.yaml，配置示例如下：

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-03
spec:
  containers:
    - name: nginx
      image: nginx:v1
      ports:
        - containerPort: 80
      volumeMounts:
        - name: k8s-test-03
          mountPath: /mnt
  volumes:
  - name: k8s-test-03
    persistentVolumeClaim:
      claimName: csi-pvc-resizer-01
```

2. 执行以下命令创建 Pod。

```
kubectl create -f pod-use-resizer-pvc.yaml
```

返回如下信息：

```
pod/nginx-03 created
```

3. 执行以下命令查看 Pod 信息。

```
kubectl get pod
```

返回信息如下：

```
NAME                                        READY   STATUS    RESTARTS   AGE
instorage-csi-controller-85b897477c-jgp5n   5/5     Running   0          22h
instorage-csi-node-7dpdd                    2/2     Running   0          22h
nginx-03                                    1/1     Running   0          5m32s
```

4. 执行以下命令查看 PVC 信息。

```
kubectl get pvc
```

返回如下信息，此时 csi-pvc-resizer-01 的大小已经修改为 3Gi。

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
csi-pvc-01          Bound    pvc-6789682a-7d0a-4243-a5c2-329cb28ba8cd   1Gi        RWO            csi-sc            16h
csi-pvc-resizer-01  Bound    pvc-05491ede-5087-438d-b254-b6f3b89fda02   3Gi        RWO            csi-sc-resizer    17m
```

4.7 在线扩容

在线扩容是指在卷已挂载使用的情况下，对卷进行扩容操作。在“4.6 离线扩容”的基础上实现在线扩容。在 Kubernetes 集群中编辑 PVC，详细过程如下：

1. 执行以下命令编辑 PVC，将 spec.resources.requests.storage 属性的值从 3Gi 改成 5Gi。

```
kubectl edit pvc csi-pvc-resizer-01
```

返回信息如下：

```
persistentvolumeclaim/csi-pvc-resizer-01 edited
```

2. 执行以下命令查看 PVC 信息。

```
kubectl get pvc csi-pvc-resizer-01
```

返回信息如下，csi-pvc-resizer-01 的大小已经修改为 5Gi。

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
csi-pvc-resizer-01  Bound    pvc-05491ede-5087-438d-b254-b6f3b89fda02   5Gi        RWO            csi-sc-resizer    22m
```

4.8 storageclass 配置文件及参数说明

storageclass 配置文件的格式为 yaml，StorageClass 资源存储插件的参数说明如表 4-2 所示。

卷类型为块时的具体配置：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-18000-block-sc
provisioner: csi-instorage
parameters:
  devKind: block
```

```
  volPoolName: pool0
  volThin: "true"
reclaimPolicy: Delete
```

卷类型为文件时的具体配置：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-18000-share-sc
provisioner: csi-instorage
parameters:
  devKind: share
  shareProto: nfs
  volPoolName: pool0
  shareAccessClient: 0.0.0.0/24:rw:all_squash:no_root_squash
  volThin: "true"
reclaimPolicy: Delete
```

表 4-2 StorageClass 资源存储插件参数说明

参数名	类型	说明	是否必须
devName	string	存储名称，与存储配置文件的 storage[].name 一致。不设置则默认为第一个存储。	非必须
devKind	string	创建卷的类型（块或文件），取值为 block/share	必须
volPoolName	string	创建卷时所在的存储池的名称	必须
shareProto	string	文件系统共享协议，取值为 nfs/cifs	devKind 为 share 时，该项为必须
volAuxPoolName	string	创建卷时辅助卷所在的存储池的名称	双活/镜像卷时必须
volIOGrp	string	创建卷时所在的 IO 组的 ID 号	双活卷时必须
volAuxIOGrp	string	创建卷时辅助卷所在的 IO 组的 ID 号	双活卷时必须
volThin	string	是否创建精简卷，取值为 true 或 false。开启压缩时，自动开启精简卷	默认 false
volCompress	string	是否创建压缩卷，取值为 true 或 false。开启压	默认 false

		缩时，自动开启精简卷	
volInTier	string	是否开启分层，取值为 true 或 false	默认 false
volLevel	string	卷的等级类型 ● 普通卷为 basic ● 镜像卷为 mirror ● 双活卷为 aa	默认 basic，非必须
volThinResize	string	创建精简卷时，初始的卷大小占实际容量的百分比	精简卷时必须
volThinGrainSize	string	精简卷增长时的块大小，单位为 KiB，可选 32、64、128、256 等值	精简卷时有效，非必须
volThinWarning	string	精简卷告警阈值，告警时容量占实际容量的百分比	精简卷时有效，非必须
volAutoExpand	string	是否开启自动扩张，取值为 true 或 false	默认 false

5 故障分析与解决

- 网络问题

  Kubernetes 集群中所有部署了本存储 K8sCSIPlugin 插件的节点都需要与存储的管理网络互通，插件通过 SSH 连接到存储并执行存储端的相关命令来完成卷管理相关的操作。

- SAN 网络异常

  存储上的卷被容器业务使用时，通过 SAN 网络将卷挂载到主机。SAN 网络异常时，主机端无法正常使用存储上的卷。

- 认证失败

  无法连接到本存储设备，请检查用户名、密码是否配置正确。

- 容器使用的设备是单路经设备，请检查多路径相关的配置是否正确。

- 无法创建/删除卷，执行以下命令查看插件是否被调用：

  ```
  kubectl logs ${controller-pod-name} -c ${container-name}
  ```

- 无法挂载/卸载卷，执行以下命令查看插件是否被调用：

  ```
  kubectl logs ${node-pod-name} -c ${container-name}
  ```

- 无法在工作节点上发现存储上的卷，请检查工作节点与存储之间的数据网络是否互通。

- 无法挂载卷并映射给容器使用。

  K8sCSIPlugin 插件创建的卷默认未进行文件系统格式化，因此第一次被容器使用时，需要使用读写模式挂载，系统才会自动格式化该卷。如果使用只读模式，则系统无法自动格式化该卷，因此无法挂载并映射给容器使用。

6 附录

6.1 参考资料

除本手册外，你还可以通过 Kubernetes 社区或 Kubernetes 提供商获取 Kubernetes 中的相关概念、部署配置方法、使用方法等相关资料。以下提供了部分社区公开资料：

- Kubernetes 社区官方网站：https://kubernetes.io/
- Kubernetes 社区部署安装资料：https://kubernetes.io/docs/setup/
- Kubernetes 社区存储相关资料：https://kubernetes.io/docs/concepts/storage/volumes/
- Kubernetes 开发设计等过程及资料均托管在 GitHub 上：https://github.com/kubernetes/
- Kubernetes 存储相关开发信息：
  https://github.com/kubernetes/community/tree/master/sig-storage
- CSI 接口文档：https://github.com/container-storage-interface/spec

6.2 扩容双活卷

当前存储端本身不支持双活卷的扩容操作，K8sCSIPlugin 插件通过组合存储管理命令来完成双活卷的扩容。

注意事项

通过 K8sCSIPlugin 插件扩容双活卷时，需要注意以下事项：

- 双活卷不能有 I/O 下发，插件通过在主机端冻结双活卷挂载到的文件系统来确保过程中没有 I/O 下发。
- 扩容过程中不能有卷在存储上映射到该主机，或者与该主机解除映射。否则可能造成主机端卷的路径异常，导致系统混乱。插件通过使用文件锁的方式确保插件本身不会同时触发在同一个主机上的双活卷扩容、卷挂载到主机、卷从主机卸载操作。
- 由于存储端双活卷扩容是由多个命令组合完成的，多个命令调用中间出错以后，无法回滚，且存储端不能再执行对应主机上的卷挂载/卸载/双活卷扩容操作。因此，命令失败后会在该主机上指定目录生成屏障文件，避免对处于失败的双活卷产生新的影响。屏障文件名称为“aa-extend-failure.barrier”。双活卷扩容失败后，可以参考日志文件确定扩容过程中已成功的步骤、失败的步骤及未处理的步骤，并在存储端完成修复操作，修复后可删除屏障文件。屏障文件删除后，插件才可以继续执行双活卷扩容/卷挂载/卷卸载命令。
- 执行存储命令组合失败导致双活卷扩容失败时，需要尽快停止使用对应双活卷的业务，避免继续写产生

问题。然后确定具体从哪一步开始失败，可以根据日志里面的信息，尝试访问存储，通过手动执行完成失败命令以及后续命令的执行。

- 双活卷扩容命令组合失败通常不会对主卷造成影响，数据本身不会有异常，但是双活卷中主卷、辅助卷之间的同步关系通常已经不能满足双活卷的要求，需要进行修复。

扩容操作

1. 执行以下命令删除双活卷中主卷和辅助卷之间的远程复制关系。

```
mcsop rmrcrelationship <rc-id>
```

2. 执行以下命令扩容主卷。

```
mcsop expandvdisksize -size <add-size> -unit gb <master-name>
```

3. 执行以下命令扩容辅助卷。

```
mcsop expandvdisksize -size <add-size> -unit gb <aux-name>
```

4. 执行以下命令扩容主卷对应的变更卷。

```
mcsop expandvdisksize -size <add-size> -unit gb <master-change-volume>
```

5. 执行以下命令扩容辅助卷对应的变更卷。

```
mcsop expandvdisksize -size <add-size> -unit gb <aux-change-volume>
```

6. 执行以下命令重建主卷和辅助卷之间的远程复制关系。

```
mcsop mkrcrelationship -master <master-name> -aux <aux-name> -cluster <cluster-name> -sync -activeactive
```

7. 执行以下命令增加主卷在远端站点的访问能力。

```
mcsop addvdiskaccess -iogrp <new-iogrp> <master-name>
```

8. 执行以下命令关联主卷和主卷变更卷。

```
mcsop chrcrelationship -masterchange <master-change-volume> <rc-id>
```

9. 执行以下命令关联辅助卷和辅助卷变更卷。

```
mcsop chrcrelationship -auxchange <aux-change-volume> <rc-id>
```

6.3 缩略语

A		
API	Application Program Interface	应用程序接口
C		
CIFS	Common Internet File Systems	通用互联网文件系统
F		
FC	Fiber Channel	光纤通道
I		
iSCSI	Internet Small Computer System Interface	互联网小型计算机接口
N		
NFS	Network File System	网络文件系统
P		
PV	Persistent Volume	持久卷
PVC	Persistent Volume Claim	持久卷声明
S		
SSH	Secure Shell Protocol	安全外壳协议
W		
WWPN	World Wide Port Name	全球端口名称