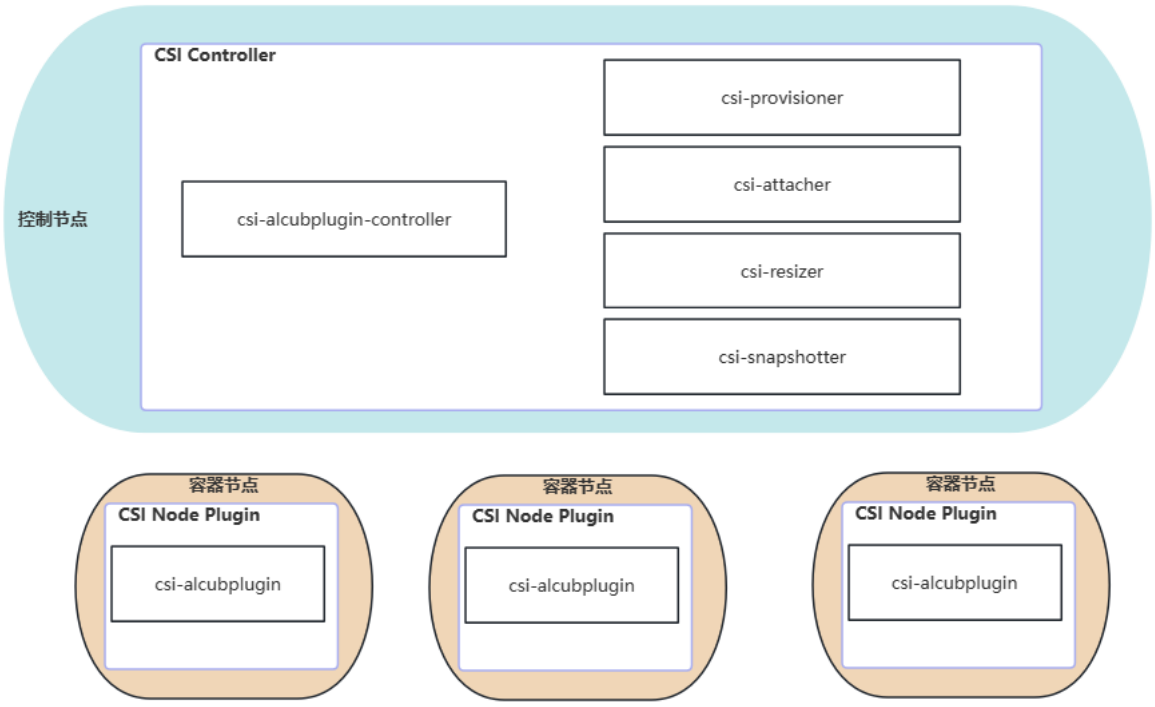
容器 CSI

容器 CSI

（一）什么是容器 CSI：

CSI 定义了一组标准的接口（CSI 提供的标准化接口主要包括创建卷、删除卷、挂载卷、卸载卷、扩展卷），这些接口允许第三方存储系统与容器编排系统（如 Kubernetes）进行集成，这种集成方式通过插件（**CSI Driver**）形式实现，插件充当了容器和存储之间的“桥梁”。

CSI Driver 主要包含两个组件：



1. CSI Controller（控制器负责处理容器编排系统的请求，如创建和删除存储卷。）：

- **定义：** CSI Controller 负责处理来自容器编排系统（如 Kubernetes）的请求，执行相应的存储操作。每个存储系统可能有一个或多个 CSI Controller。
  - 处理卷的生命周期，包括创建、删除、调整大小等。
  - 响应容器编排系统的调度请求，将卷分配给节点。
  - 确保存储系统的状态与容器编排系统的期望状态一致。
  - 组成：csi-provisioner、csi-attacher、csi-resizer、csi-snapshotter、csi-plugin-controller
  - csi-alcubplugin-controller：运行在 Kubernetes 的控制平面上，负责处理集群级别的存储操作。这包括创建存储卷、删除存储卷、扩展存储卷、创建快照等操作。
  - csi-provisioner：为 PVC（Persistent Volume Claim）动态提供 PV（Persistent Volume）。
  - csi-attacher：管理存储卷的挂载和卸载。
  - csi-resizer：用于调整 PV 的大小。
  - csi-snapshotter：用于创建和管理存储快照。

2. CSI Node Plugin（运行在每个容器节点上，负责挂载和卸载卷。）：

- **定义：** CSI Node Plugin 运行在容器部署的节点上，负责将卷（Volumes）挂载到容器运行时环境中，以及在容器结束后卸载卷。
    - 处理与本地节点相关的存储任务，如挂载卷、卸载卷、格式化卷等。
    - 与容器运行时（如 Docker、containerd）交互，确保存储资源与容器正确集成。
    - csi-driver-registrar：该容器负责注册 CSI 驱动到 Kubernetes 集群中。它是 CSI 驱动与 Kubernetes 之间的桥梁，确保驱动被正确加载并可在集群中使用。
    - csi-alcubplugin：这是主要的 CSI 插件容器。它实现了 CSI 接口，用于与存储系统进行交互。该容器负责处理存储相关的操作，如创建、删除、挂载、卸载存储卷等。
    - liveness-probe：这个容器用于检查主要插件容器的存活状态。它通过定期执行检查来确保主要插件容器正常运行，并在检测到问题时进行报警或重启。
    - manul-brick：这个容器的具体作用不太明确，可能是特定于你的设置或环境的自定义容器。它可能包含一些特定的逻辑或工具，用于处理存储或与其他容器进行交互。

（二）以安全容器使用高性能盘为例，学习容器 CSI：

1.安装 CSI 插件： 在环境上以对接包形式配置容器 CSI，加载后 csi-alcubplugin pod 运行在安全容器 worker 节点，csi-alcubplugin-provisioner pod 运行在高性能节点

```Python
[root@node-5 ~]# kubectl get po -owide -A |grep csi
```

```
csi-alcub            csi-alcubplugin-2fwmw
4/4     Running   0          6d19h   10.39.1.16      node-13 (安全容器
节点)    <none>           <none>

csi-alcub            csi-alcubplugin-cwwdc
4/4     Running   0          6d19h   10.39.1.17      node-14 (安全容器
节点)    <none>           <none>

csi-alcub            csi-alcubplugin-provisioner-65db66d7f8-gq2hf
5/5     Running   6          6d19h   10.232.4.164    node-5  (高性能存
储节点)    <none>           <none>
```

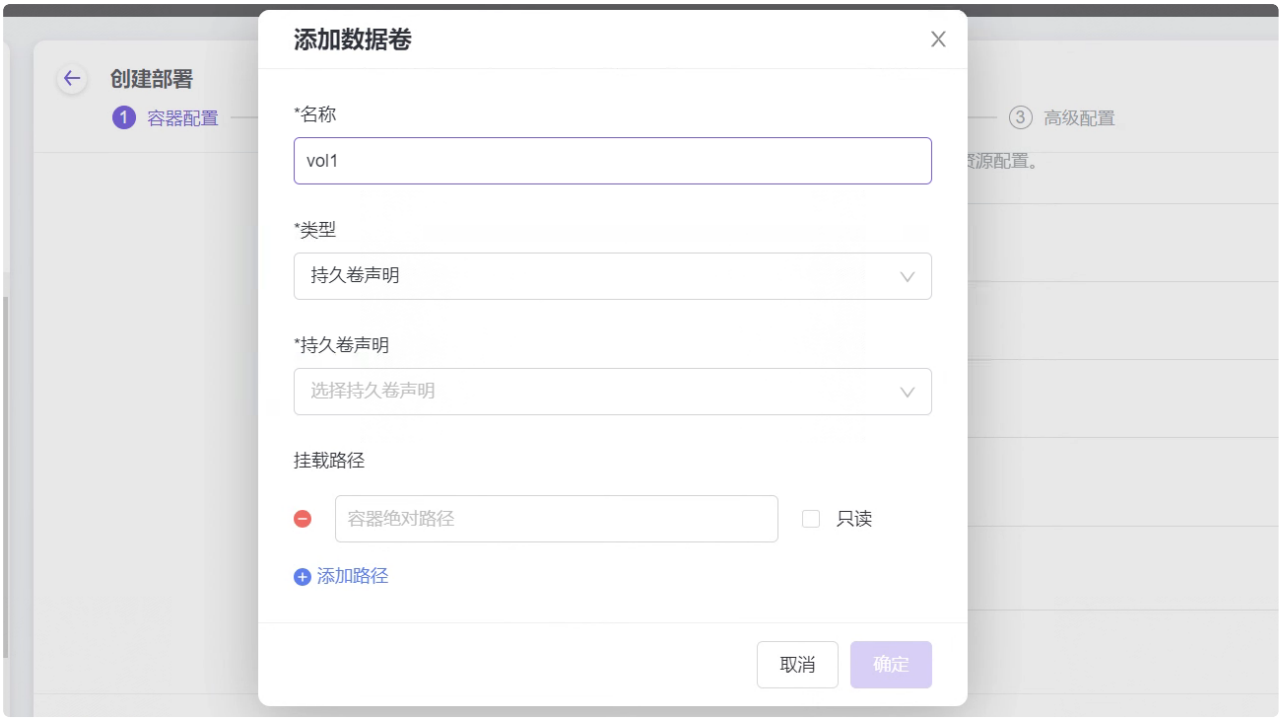
2.创建高性能的 StorageClass（创建时关联 CSI 插件，StorageClass 会通过其关联的 CSI 驱动程序来动态创建 PV）

Python

```
[root@node-5 ~]# kubectl get storageclass

NAME           PROVISIONER                    RECLAIMPOLICY   VOLUME
BINDINGMODE      ALLOWVOLUMEEXPANSION   AGE

capacity       [kubernetes.io/rbd](http://kubernetes.io/rbd)
Delete          Immediate              false                  7d1h

ceph-ssd       [kubernetes.io/rbd](http://kubernetes.io/rbd)
Delete          Immediate              false                  14d

csi-alcub-sc   alcubierre.csi.driver          Delete          Immedi
ate              true                   5d20h

general        [kubernetes.io/rbd](http://kubernetes.io/rbd)
Delete          Immediate              false                  14d

local-disk     [kubernetes.io/no-provisioner](http://kubernetes.io/n
o-provisioner)   Delete          WaitForFirstConsumer   false
14d

[root@node-5 ~]# kubectl describe storageclass csi-alcub-sc

Name:            csi-alcub-sc
```

```
IsDefaultClass:  No

Annotations:     [kubectl.kubernetes.io/last-applied-configuration=
{"allowVolumeExpansion":true,"apiVersion":"storage.k8s.io/v1","kin
d":"StorageClass","metadata":{"annotations":{},"name":"csi-alcub-s
c"},"mountOptions":["discard"],"parameters":{"api_url":"http://alcub
ierre-manul.alcubierre.svc.cluster.local:8192","directVolume":"tru
e","storage_pool":"alcubierre_pool"},"provisioner":"alcubierre.csi.d
river","reclaimPolicy":"Delete](http://kubectl.kubernetes.io/last-ap
plied-configuration={"allowVolumeExpansion":true,"apiVersion":"stora
ge.k8s.io/v1","kind":"StorageClass","metadata":{"annotations":{},"na
me":"csi-alcub-sc"},"mountOptions":["discard"],"parameters":{"api_ur
l":"http://alcubierre-manul.alcubierre.svc.cluster.local:8192","dire
ctVolume":"true","storage_pool":"alcubierre_pool"},"provisioner":"al
cubierre.csi.driver","reclaimPolicy":"Delete)"}

Provisioner:           alcubierre.csi.driver
# Provisioner字段定义存储供应商信息为alcubierre.csi.driver

Parameters:            api_url=http://alcubierre-manul.alcubierre.sv
c.cluster.local:8192,directVolume=true,storage_pool=alcubierre_pool
# Parameters字段定义相关参数，api_url,storage_pool等

AllowVolumeExpansion:  True

MountOptions:

  discard

ReclaimPolicy:      Delete

VolumeBindingMode:  Immediate

Events:             <none>
```

3.创建容器并绑定 pvc（通过 pvc 声明存储需求,并指定 StorageClass），定义存储卷的标识符、挂载路径等信息传递给容器 CSI



Python

```
[root@node-5 ~]# kubectl get pvc -n huangtao|grep huangtao

huangtao                    Bound    pvc-8a0df2cc-f29a-4ae8-9727-dbd
0c69ee543   120Gi      RWO            csi-alcub-sc   22m
```

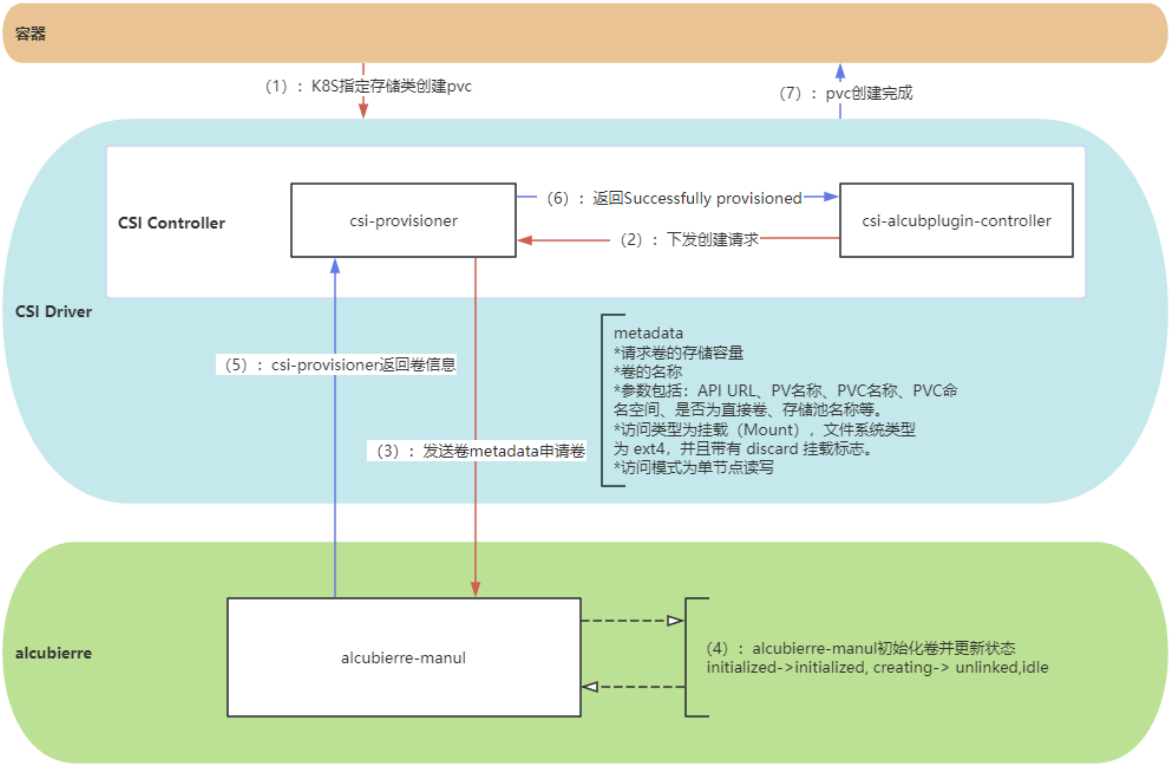
4.容器成功运行，盘挂载到容器指定目录，用户可以对高性能盘进行读写操作。

Python

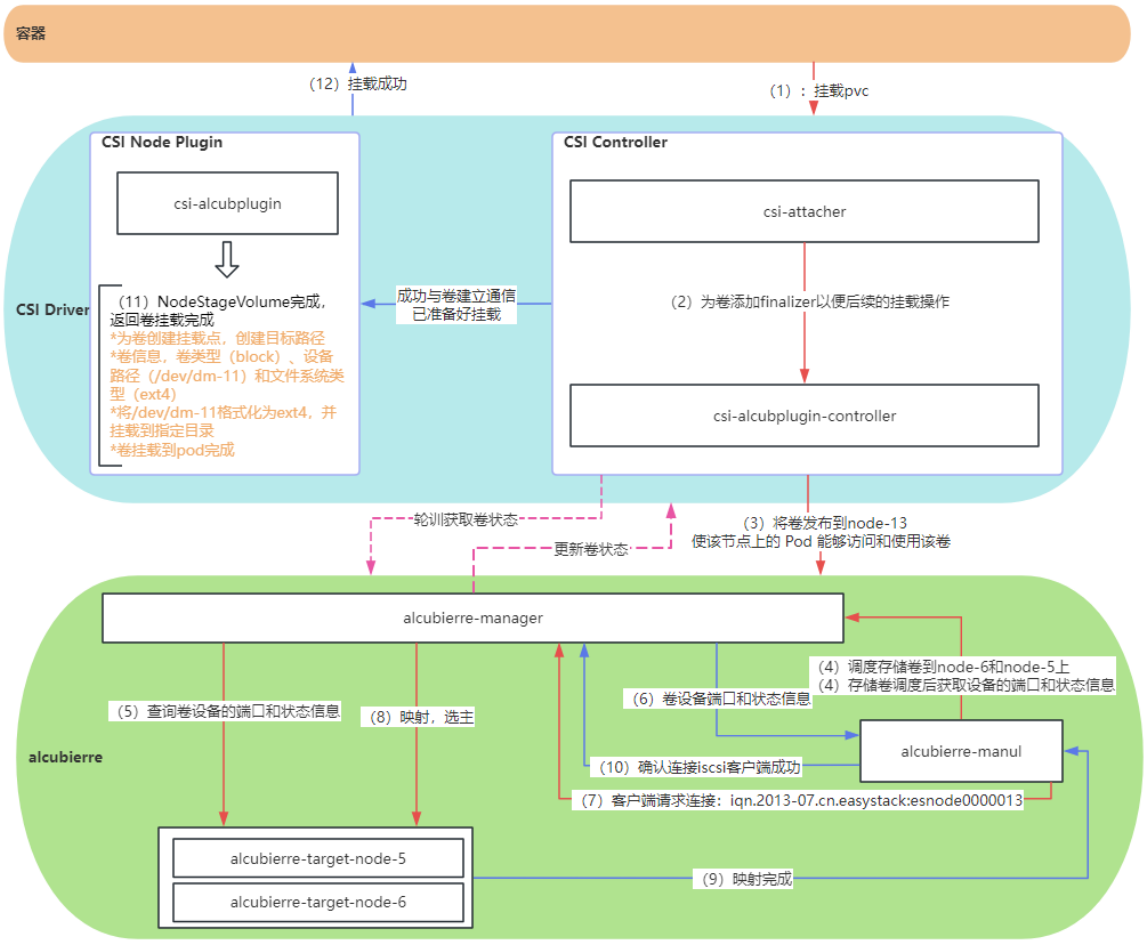
```
[root@node-5 ~]# kubectl exec -it  -n huangtao centos-fio-68549bf7d4
-hpvvl bash

kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a
future version. Use kubectl exec [POD] -- [COMMAND] instead.

[root@centos-fio-68549bf7d4-hpvvl /]# mount |grep /test/tao

/dev/sda on /test/tao type ext4 (rw,relatime,stripe=2048)

[root@centos-fio-68549bf7d4-hpvvl /]#
```

容器 CSI 参与高性能卷生命周期(创建，挂载，卸载，删除)过程

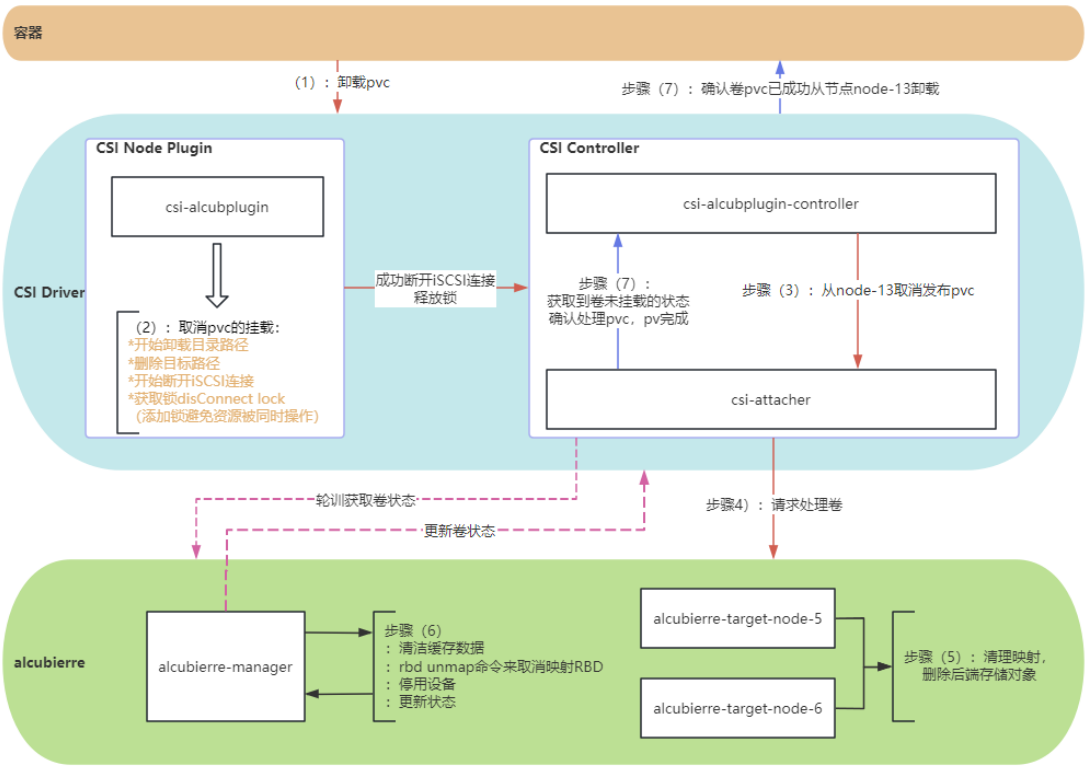
创建流程



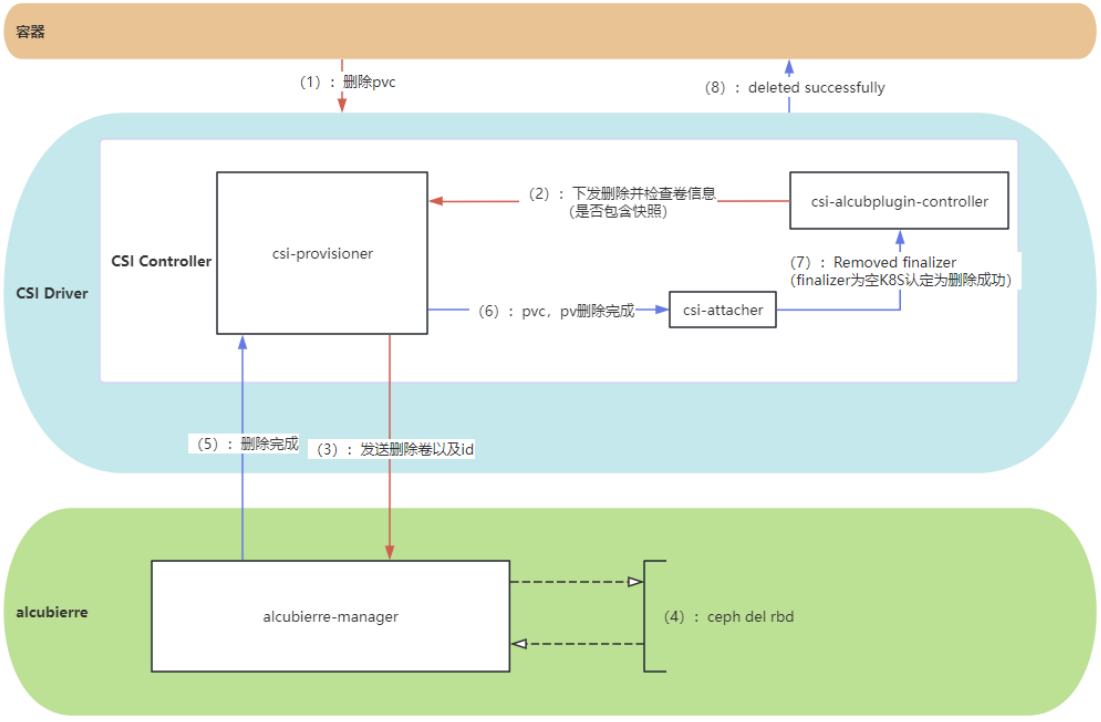
挂载流程


容器
(12) 挂载成功
(1) ：挂载pvc
CSI Node Plugin
csi-alcubplugin
CSI Driver
(11) NodeStageVolume完成，返回卷挂载完成
*为卷创建挂载点，创建目标路径
*卷信息，卷类型（block）、设备路径（/dev/dm-11）和文件系统类型（ext4）
*将/dev/dm-11格式化为ext4，并挂载到指定目录
*卷挂载到pod完成
成功与卷建立通信
已准备好挂载
CSI Controller
csi-attacher
(2) 为卷添加finalizer以便后续的挂载操作
csi-alcubplugin-controller
轮训获取卷状态
更新卷状态
(3) 将卷发布到node-13
使该节点上的 Pod 能够访问和使用该卷
alcubierre-manager
(4) 调度存储卷到node-6和node-5上
(4) 存储卷调度后获取设备的端口和状态信息
(5) 查询卷设备的端口和状态信息
(8) 映射，选主
(6) 卷设备端口和状态信息
alcubierre
alcubierre-manul
(10) 确认连接iscsi客户端成功
(7) 客户端请求连接：iqn.2013-07.cn.easystack:esnode0000013
alcubierre-target-node-5
alcubierre-target-node-6
(9) 映射完成

卸载流程


容器
(1) ：卸载pvc
步骤（7）：确认卷pvc已成功从节点node-13卸载
CSI Driver
CSI Node Plugin
csi-alcubplugin
(2) ：取消pvc的挂载：
*开始卸载目录路径
*删除目标路径
*开始断开iSCSI连接
*获取锁disConnect lock
(添加锁避免资源被同时操作)
成功断开iSCSI连接
释放锁
CSI Controller
csi-alcubplugin-controller
步骤（7）：
获取到卷未挂载的状态
确认处理pvc，pv完成
步骤（3）：从node-13取消发布pvc
csi-attacher
轮训获取卷状态
更新卷状态
步骤4）：请求处理卷
alcubierre
alcubierre-manager
步骤（6）
：清洁缓存数据
：rbd unmap命令来取消映射RBD
：停用设备
：更新状态
alcubierre-target-node-5
alcubierre-target-node-6
步骤（5）：清理映射，
删除后端存储对象

删除流程


容器
(1) ：删除pvc
(8) ：deleted successfully
CSI Driver
CSI Controller
csi-provisioner
(2) ：下发删除并检查卷信息
(是否包含快照)
csi-alcubplugin-controller
(7) ：Removed finalizer
(finalizer为空K8S认定为删除成功)
(6) ：pvc，pv删除完成
csi-attacher
(5) ：删除完成
(3) ：发送删除卷以及id
alcubierre
alcubierre-manager
(4) ：ceph del rbd