阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云 原 生 技 术 公 开 课

第 09 讲

应用存储和持久化数据卷：核心知识

至天 阿里巴巴高级开发工程师

关注
阿里巴巴云原生微信公众号
获取更多资料

阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云 原 生 技 术 公 开 课

- 20 位阿里巴巴顶级技术专家权威授课
- 视频教学，全程免费，无需注册
- 公开课官网：https://edu.aliyun.com/roadmap/cloudnative

课程通过后，即颁发 CNCF 官方授权“云原生认证”证书

微信扫描二维码
直接听课

1 Volumes介绍

2 用例解读

3 操作演示

4 架构设计

5 下节预告

Pod Volumes

1. 如果一个Pod中某一个容器异常退出，被kubelet拉起如何保证之前产生的重要数据不丢？
2. 同一个Pod的多个容器如何共享数据？

Kubernetes Volume 类型：

1. **本地存储**： emptydir/hostpath ...
2. **网络存储**：

   in-tree: awsElasticBlockStore/gcePersistentDisk/nfs ...

   out-of-tree: flexvolume/csi等网络存储volume plugins
3. **Projected Volume**： secret/configmap/downwardAPI/serviceAccountToken
4. **PVC与PV体系**

Persistent Volumes

Pod中声明的volume的生命周期与Pod相同，以下常见场景：

1. Pod销毁重建（如Deployment管理的Pod镜像升级）
2. 宿主机故障迁移（如StatefulSet管理的Pod带远程volume迁移）
3. 多Pod共享同一个数据volume
4. 数据volume snapshot, resize 等功能的扩展实现

不足之处：

使用Pod Volumes无法准确表达数据volume复用/共享语义，新功能扩展很难实现。

优化：

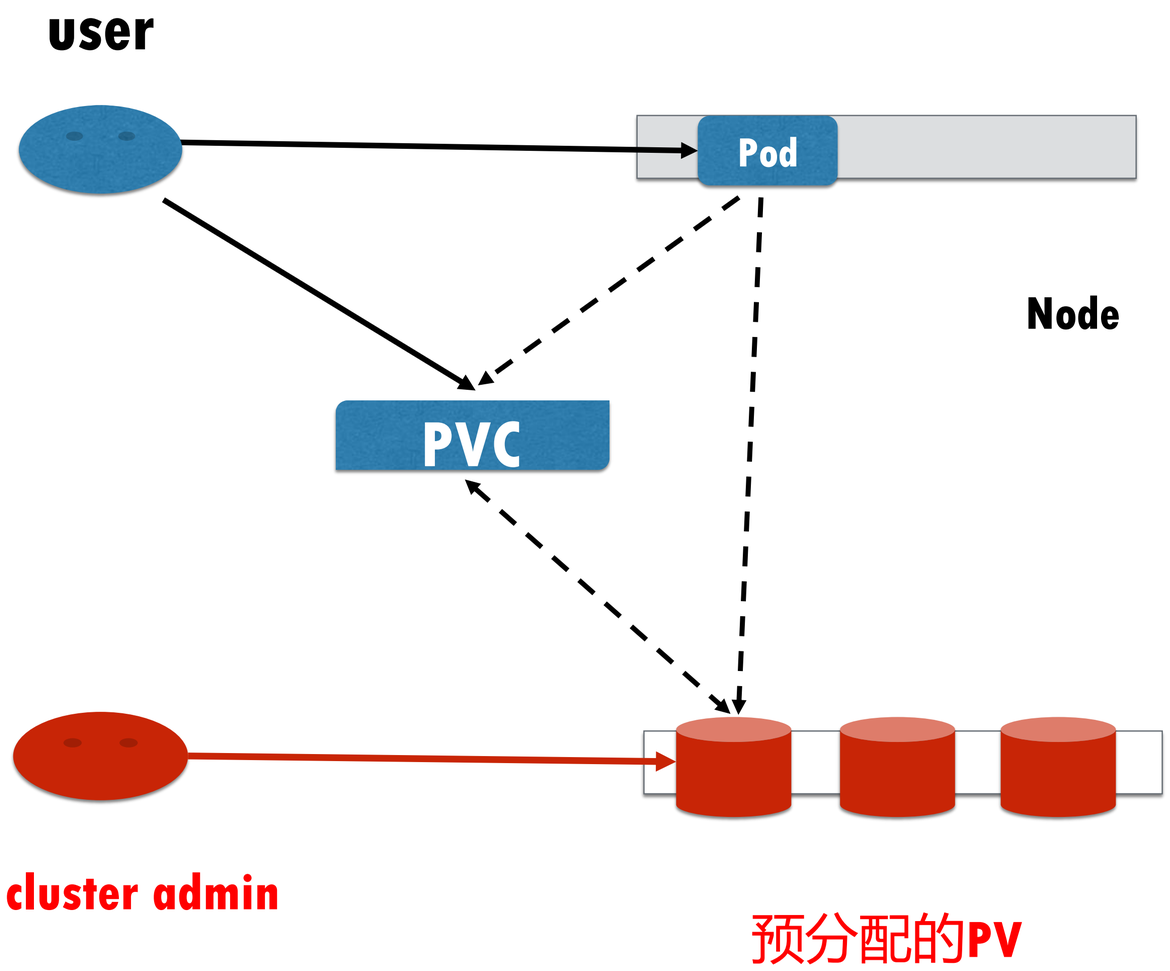
如果能将存储与计算分离，使用不同的组件(Controllers)管理存储与计算资源，解耦 Pod 与 Volume 的生命周期关联，可以很好的解决这些场景下的问题。

PersistentVolumeClaim（PVC）设计意图

有了PV，为什么又设计了PVC?

1. **职责分离**，PVC中只用声明自己需要的存储size、access mode（单node独占还是多node共享？只读还是读写访问？）等业务真正关心的存储需求（不用关心存储实现细节），PV和其对应的后端存储信息则由交给cluster admin统一运维和管控，安全访问策略更容易控制。
2. **PVC简化了User对存储的需求，PV才是存储的实际信息的承载体**，通过kube-controller-manager中的PersisentVolumeController将PVC与合适的PV bound到一起，从而满足User对存储的实际需求。
3. PVC像是面向对象编程中抽象出来的**接口**，PV是接口对应的**实现**。

Static Volume Provisioning



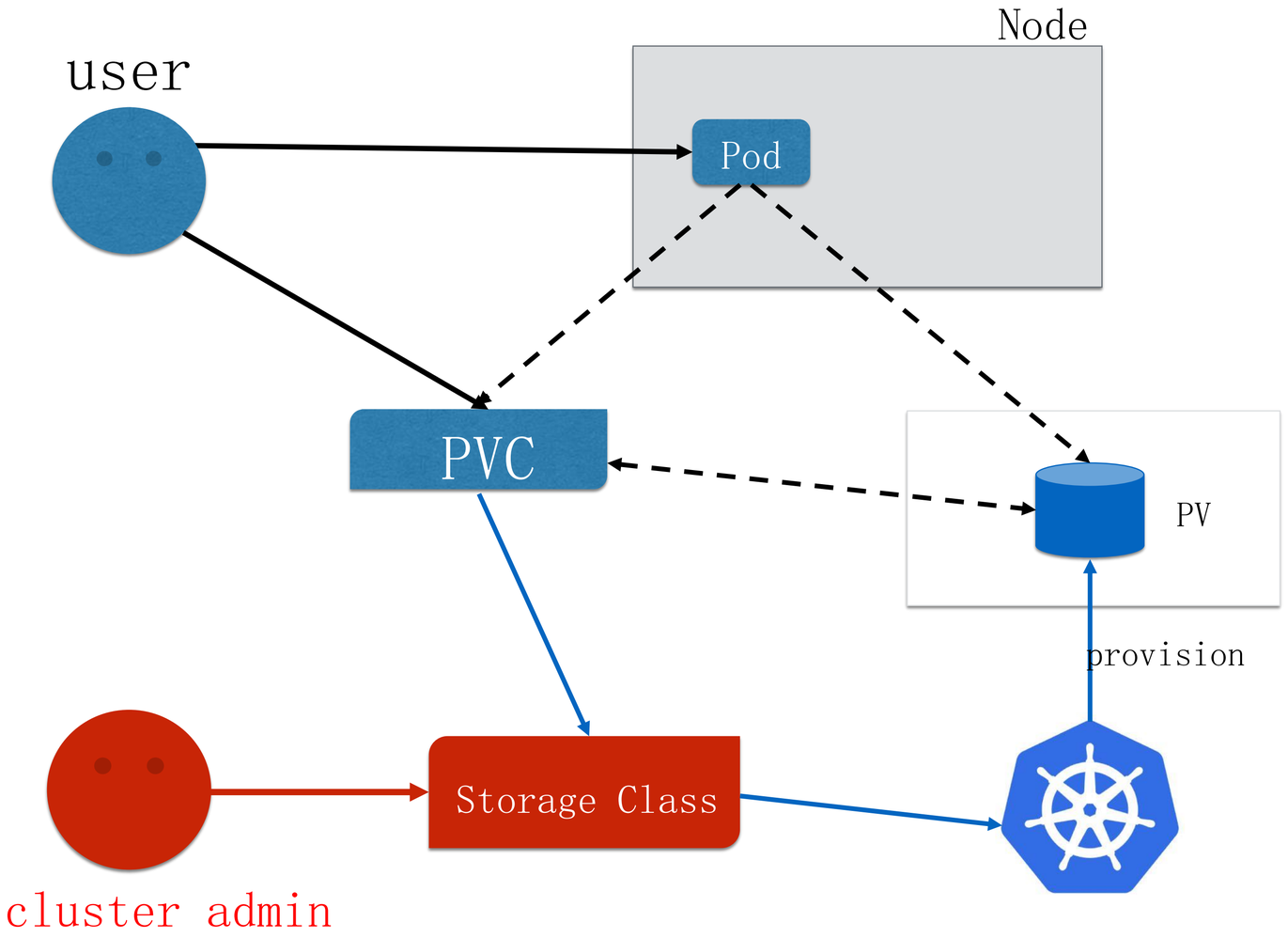
Static Volume Provisioning的不足：

Cluster Admin需要提前规化或预测存储需求，而User的需求是多样化的，很容易导致User提交的PVC找不到合适的PV。

更好的方式：

Cluster Admin只创建不同类型存储的模板，User在PVC中指定使用哪种存储模板以及自己需要的大小、访问方式等参数，**然后 K8s 自动生成相应的PV对象。**

Dynamic Volume Provisioning



这里的StorageClass就是前文所说的创建PV的模板，它包含了创建某种具体类型PV所需的参数信息，user无需关心这些PV的细节。

而K8s则会结合PVC和SC两者的信息动态创建PV对象。

1 Volumes介绍

2 用例解读

3 操作演示

4 架构设计

5 下节预告

Pod Volumes使用

- **.spec.volumes** 声明**pod**的**volumes**信息
- **.spec.containers.volumeMounts** 声明**container**如何使用**pod**的**volumes**
- 多个**container**共享同一个**volume**时，可以通过**.spec.containers.volumeMounts. subPath**隔离不同容器在同个**volume**上数据存储的路径

```
apiVersion: v1
kind: Pod
metadata:
    name: test-pod
spec:
    containers:
    - name: container-1
      Image: ubuntu:18.04
      volumeMounts:
       -  name: cache-volume
          mountPath: /cache          # 容器内挂载路径
          subPath: cache1            # 在cache-volume建立子目录 cache1
       -  name: hostpath-volume
          mountPath: /data
          readOnly: true             # 只读挂载
    - name: container-2
      image: ubuntu:18.04
      volumeMounts:
      -  mountPath: /cache
         name: cache-volume
         subPath: cache2
    volumes:
    - name: cache-volume
      emptyDir: {}
    - name: hostpath-volume
      hostPath:
          path: /tmp/data            # 宿主上路径，pod删除之后该目录仍然存在
```

宿主机上路径：**/var/lib/kubelet/pods/<PodUID>/volumes/kubernetes.io~empty-dir/cache-volume**，由于上面两个容器都通过**subPath**使用该**volume**，所以在该路径下还有两个子目录 **cache1** 和 **cache2**，**pod**删除之后该目录也会被清除

Static Volume Provisioning

以使用阿里云文件存储（NAS）为例：

Cluster Admin:

1. 通过阿里云文件存储控制台，创建NAS文件系统和添加挂载点。
2. 创建PV对象，将NAS文件系统大小，挂载点，以及PV的access mode， reclaim policy等信息添加到PV对象中。

User：

1. 创建PVC对象，声明存储需求。
2. 创建应用pod并通过在.spec.volumes 中通过PVC声明volume, 通过.spec.containers.volumeMounts声明container挂载使用该volume。

系统管理员预先创建 PV

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nas-csi-pv
spec:
  capacity:
    storage: 5Gi                                    # 该volume的总容量大小
  accessModes:
  - ReadWriteMany                                   # 该volume可以被多个node上的pod挂载使用且都具有读写权限
  persistentVolumeReclaimPolicy: Retain     # 该volume使用后被release之后的回收策略
  csi:
    driver: nasplugin.csi.alibabacloud.com    # 指定由什么volume plugin来挂载该volume（需要提前在node上部署）
    volumeHandle: data-id
    volumeAttributes:
      host: "***.cn-beijing.nas.aliyuncs.com"
      path: "/k8s"
      vers: "4.0"
```

```
### 用户创建 PVC ###
# nas-pvc与nas-csi-pv匹配，将由
# PersistentVolumeController将两者
# bound到一起
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: nas-pvc
spec:
    accessModes:
    - ReadWriteMany
    resources:
      requests:
        storage: 5Gi
```

```
### 用户创建 Pod ###
…
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
    - name: nas-pvc
      mountPath: /data
  volumes:
  - name: nas-pvc
    persistentVolumeClaim:
      claimName: nas-pvc
```

Dynamic Volume Provisioning

系统管理员创建

 # StorageClass就像动态创建PV的模板，为创建PV对象提供必要的参数

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
    name: csi-disk
# 指定使用什么volume plugin来create/delete/attach/detach/mount/unmount 新PV
# 该volume plugin需要部署到k8s cluster中
provisioner: diskplugin.csi.alibabacloud.com
parameters:
   regionId: cn-Beijing
   zoneId: cn-beijing-b
   fsType: ext4
   type: cloud_ssd
reclaimPolicy: Delete
```

用户创建 PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
    name: disk-pvc
spec:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 25Gi
    storageClassName: csi-disk
```

用户创建 Pod

```
…
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
    volumeMounts:
    - name: disk-pvc
      mountPath: /data
  volumes:
  - name: disk-pvc
    persistentVolumeClaim:
    claimName: disk-pvc
```

PV Spec 其他重要字段解析

Capacity：存储总空间

AccessModes： PV访问策略控制列表，必须同 PVC 的访问策略控制列表匹配才能绑定（bound）

- ReadWriteOnce只允许单node访问
- ReadOnlyMany允许多个node只读访问
- ReadWriteMany允许多node读写访问

一个PV可以设置多个访问策略，PVC与PV bound时， PV Controller 会优先找到 AccessModes 列表最短并且匹配 PVC AccessModes 列表的 PV 集合，然后从该集合中找到Capacity最小且符合PVC size 需求的PV对象

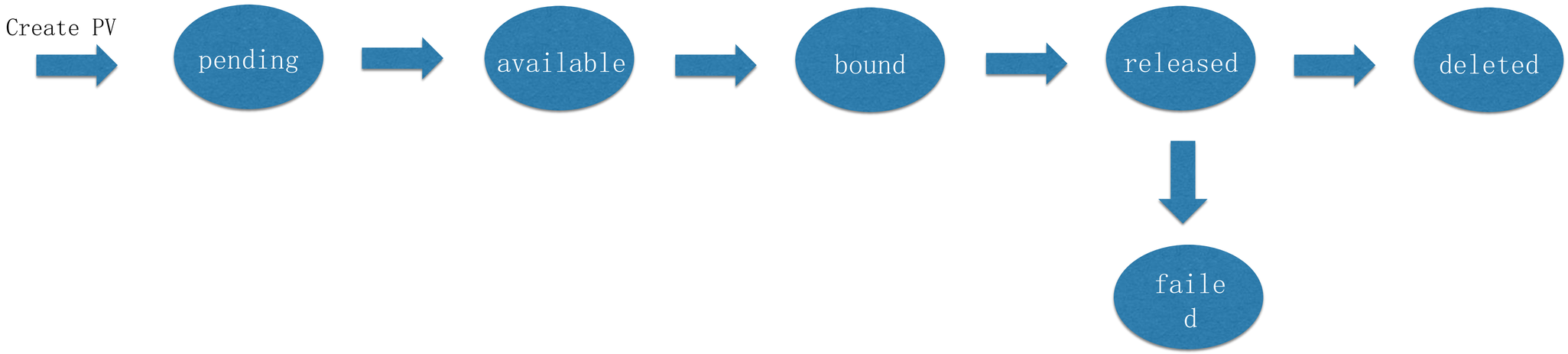
PersistentVolumeReclaimPolicy： PV被release之后(与之bound的PVC被删除)回收再利用策略

- Recycle (已废弃)
- Delete： volume被released之后直接delete， 需要volume plugin支持
- Retain： 默认策略，由系统管理员来手动管理该volume

StorageClassName： PVC可通过该字段找到相同值的PV（静态provisioning），也可通过该字段对应的storageclass从而动态provisioning新PV对象

NodeAffinity：限制可以访问该volume的nodes，对使用该volume的pod的调度有影响(因为使用该volume的pod只能调度能访问该PV的node上才能正常工作)

PV状态流转



说明：到达released状态的PV无法根据Reclaim Policy回到available状态而再次bound新的PVC。此时，如果想复用原来PV对应的存储中的数据，只有两种方式：

1. 复用old PV中记录的存储信息新建PV对象。
2. 直接从PVC对象复用，即不unbound PVC和PV（即：StatefulSet 处理存储状态的原理）。

1 Volumes介绍

2 用例解读

3 操作演示

4 架构设计

5 下节预告

云端环境实例操作示例

1. **Static Volume Provisioning** 示例

   阿里云 **NAS**

2. **Dynamic Volume Provisioning**示例

   阿里云云盘

1 Volumes介绍

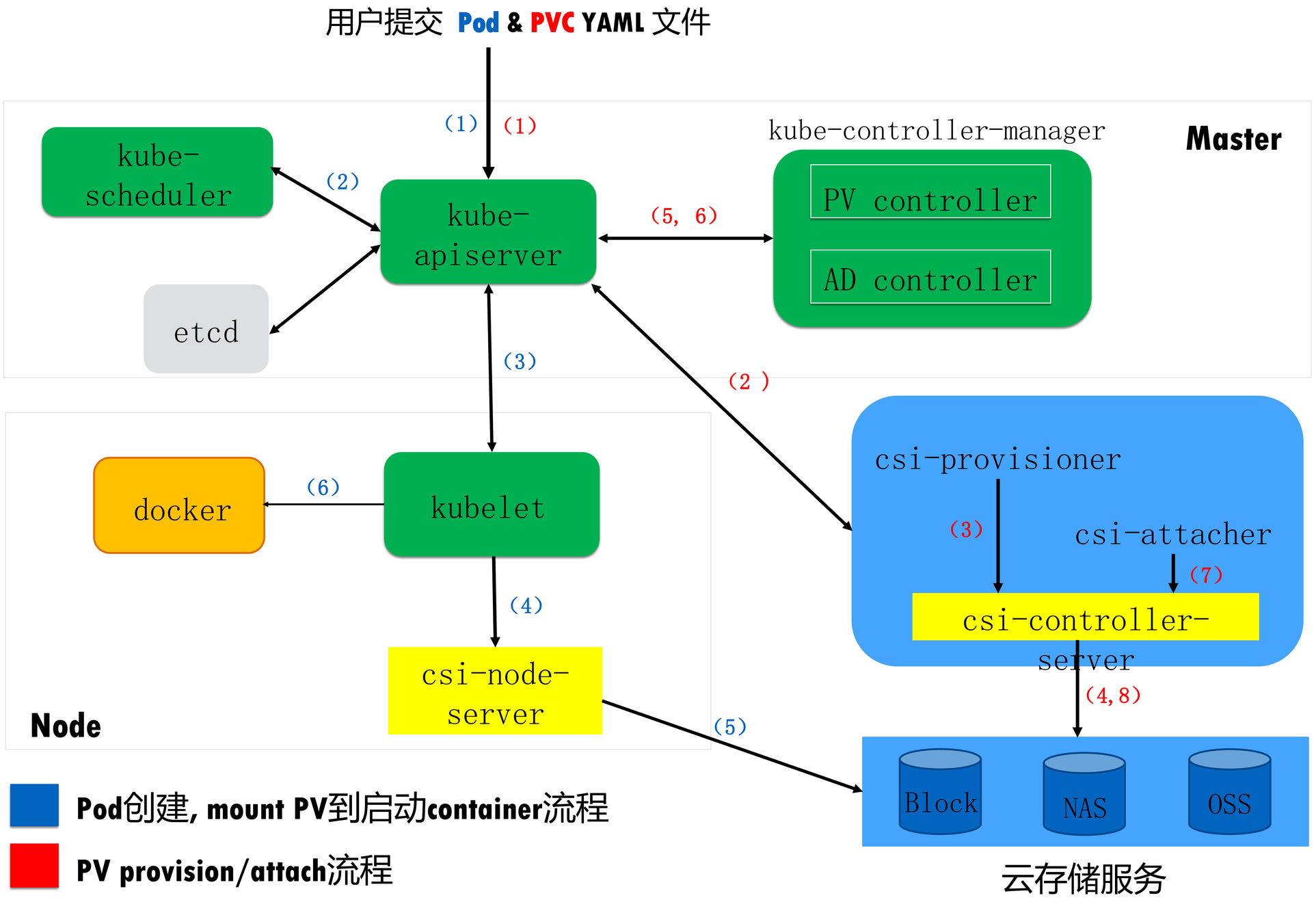
2 用例解读

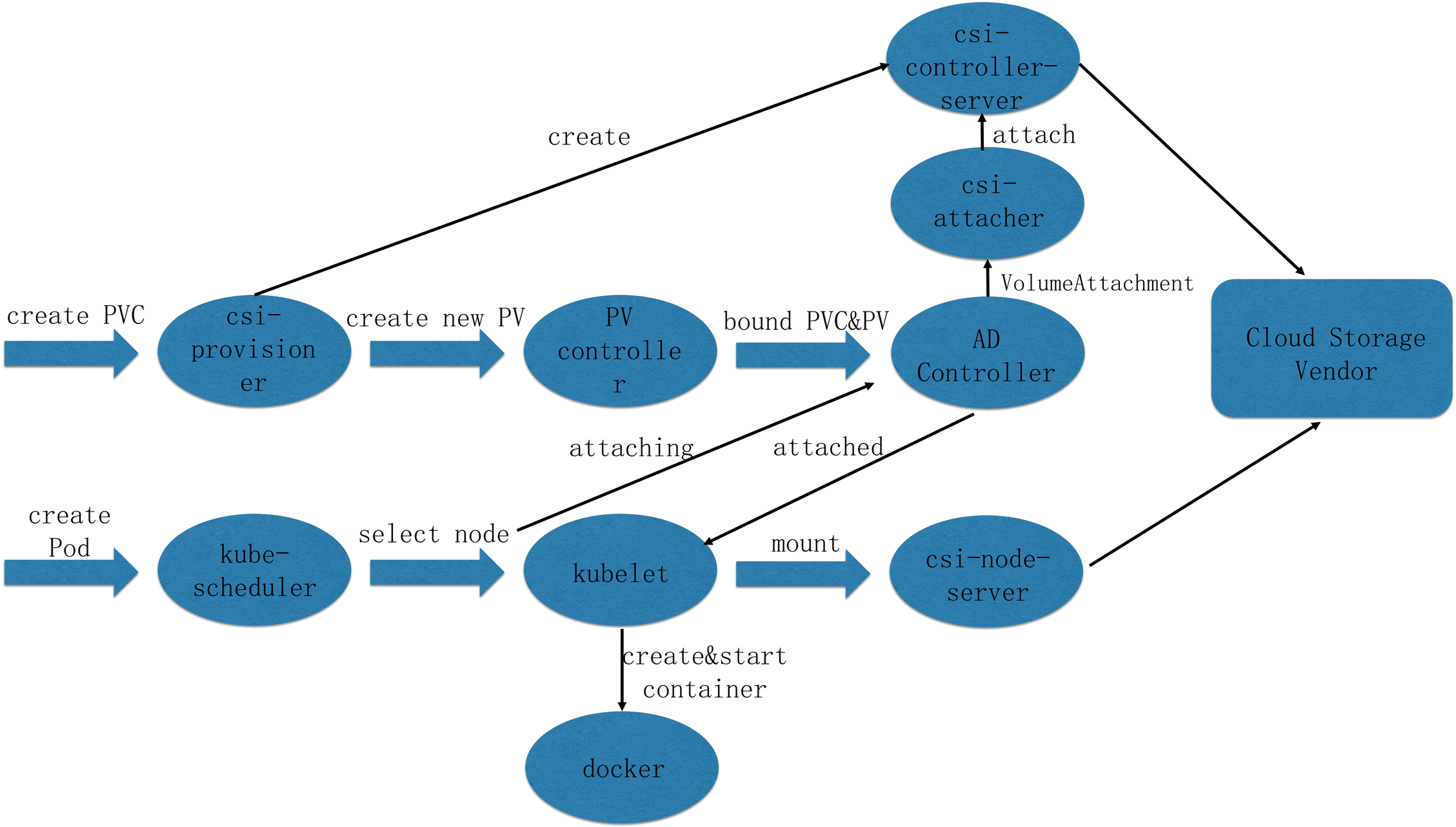
3 操作演示

4 架构设计

5 下节预告

Kubernetes 对 PV + PVC 体系的完整处理流程详解


用户提交 Pod & PVC YAML 文件
(1)
(1)
kube-scheduler
(2)
kube-apiserver
etcd
kube-controller-manager
Master
(5, 6)
PV controller
AD controller
(3)
(2)
csi-provisioner
(3)
csi-attacher
(7)
csi-controller-server
(4,8)
docker
(6)
kubelet
(4)
csi-node-server
Node
(5)
Block
NAS
OSS
Pod创建, mount PV到启动container流程
PV provision/attach流程
云存储服务


csi-controller-server
create
attach
csi-attacher
VolumeAttachment
create PVC
csi-provisioner
create new PV
PV controller
bound PVC&PV
AD Controller
Cloud Storage Vendor
attaching
attached
create Pod
kube-scheduler
select node
kubelet
mount
csi-node-server
create&start container
docker

1 Volumes介绍

2 用例解读

3 操作演示

4 架构设计

5 下节预告

下节预告

应用存储和持久化数据：存储快照与调度

- **Volume Snapshot/Restore**
- **CSI-Snapshotter: Snapshot/Restore** 处理流程
- **Volume Scheduling** 存储调度
- **Volume Scheduling** 处理流程

更多云原生权威资讯、课程答疑、讲师互动与案例分析

关注
「阿里巴巴云原生」**微信公众号**
回复“加群”进入讨论群

阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

后云计算时代，技术人员如何

自我提升？

微信扫描二维码直接听课

谢谢观看

THANK YOU