阿里云 × CLOUD NATIVE COMPUTING FOUNDATION

云原生技术公开课

关注“阿里巴巴云原生”公众号
获取第一手技术资料

第 10 讲

应用存储和持久化数据卷：存储快照与拓扑调度

至天 阿里巴巴高级开发工程师

1 基本知识

2 用例解读

3 操作演示

4 处理流程

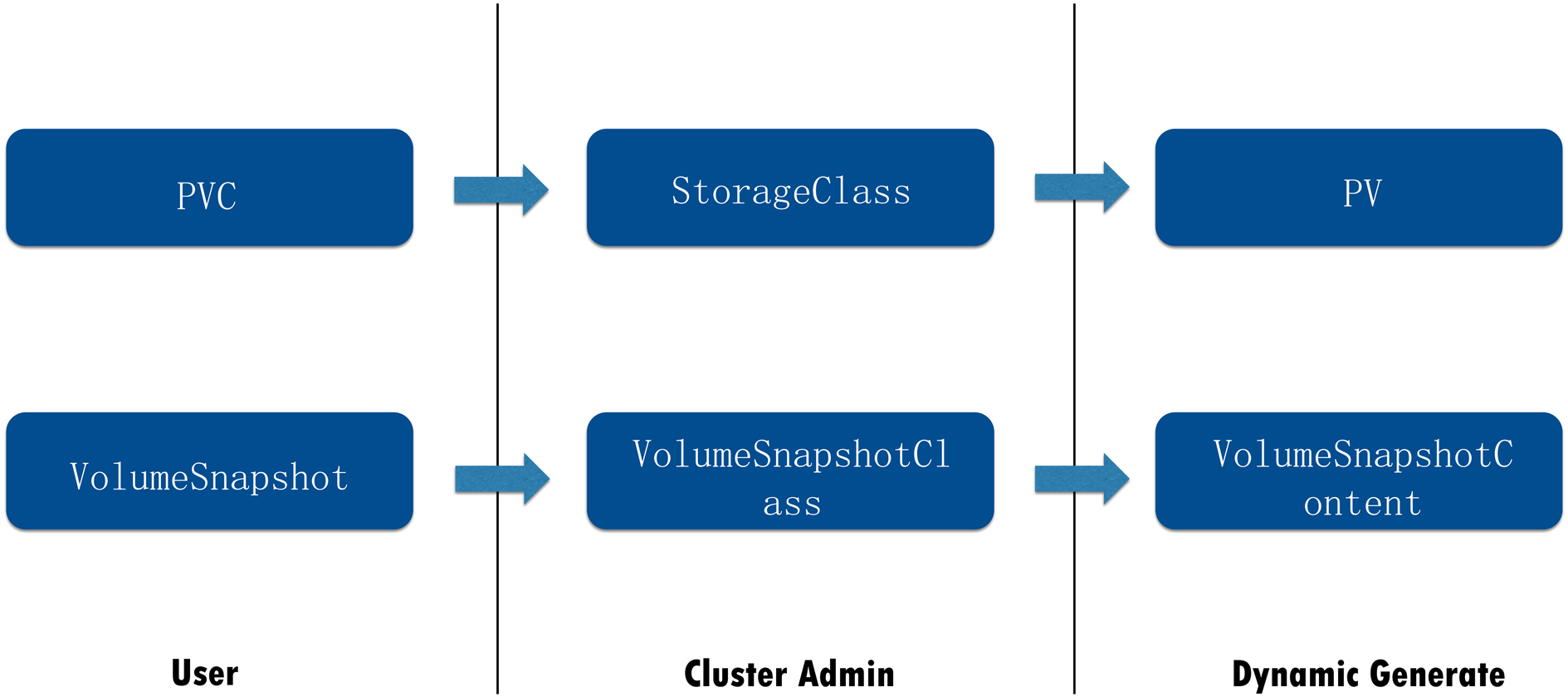
5 课后习题

存储快照产生背景

1. 如何保证重要数据在误操作之后可以快速恢复，以提高操作容错率？
2. 如何能够快速进行复制，迁移重要数据的动作？如进行环境复制与数据开发

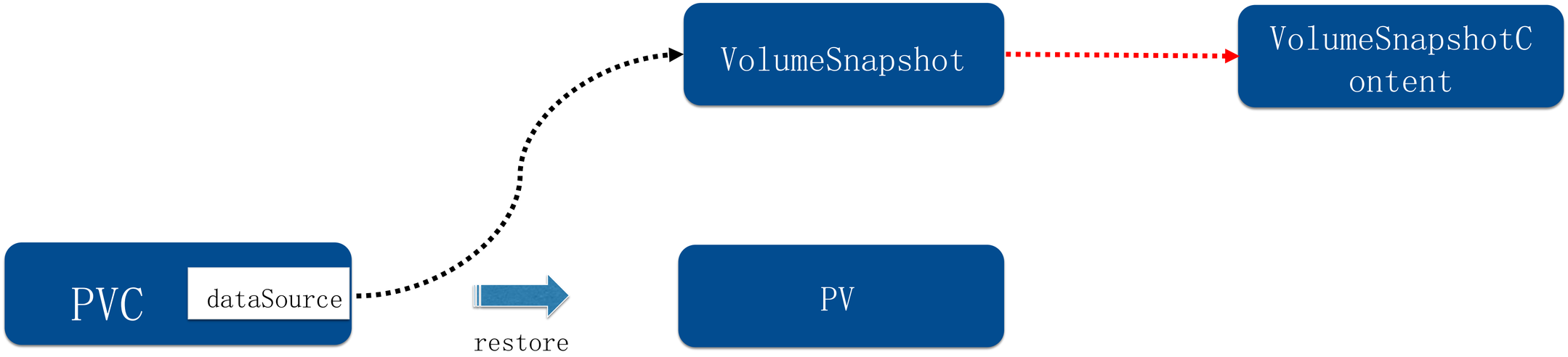
Kubernetes CSI Snapshotter controller正是为了解决这些问题而设计的

存储快照用户接口 - snapshot



存储快照用户接口 - restore

PersistentVolumeClaim 扩展字段 **.spec.dataSource** 可以指定为 **VolumeSnapshot** 对象，从而根据 **PVC** 对象生成的新 **PV** 对象，对应的存储数据是从 **VolumeSnapshot** 关联的 **VolumeSnapshotContent** **restore** 的



本讲讨论 Topology（拓扑）的含义

这里讨论的拓扑是指对 **Kubernetes** 集群中 **nodes** 的“位置”关系一种人为划分规则，通过在 **node** 的 **labels**中设置以标识自己属于具体的拓扑域，如 **Node labels** 包括：

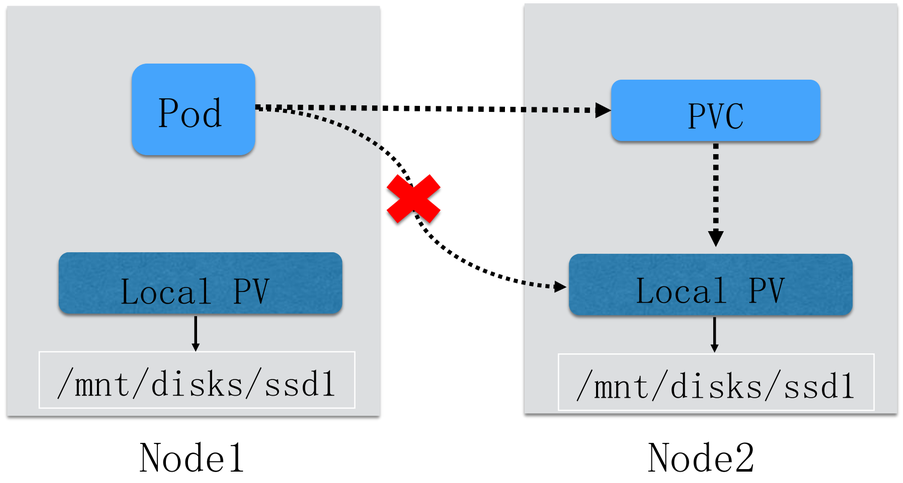
- **kubernetes.io/hostname => nodename** 拓扑域为 **node** 范围
- **failure-domain.beta.kubernetes.io/region => us-central1** 拓扑域为 **Region** 范围
- **failure-domain.beta.kubernetes.io/zone => us-central1-a** 拓扑域为 **Zone** 范围

当然也可以自定义一个 **key:value pair** 来定义一个具体的拓扑域， 如 **rack: rack1** 可以用来将属于同一个机架（**rack**）上的服务器（**nodes**）划分为一组（一个具体的拓扑域 **rack1**），以区别另一个**rack**上的一组机器的“位置”关系。

存储拓扑调度产生背景

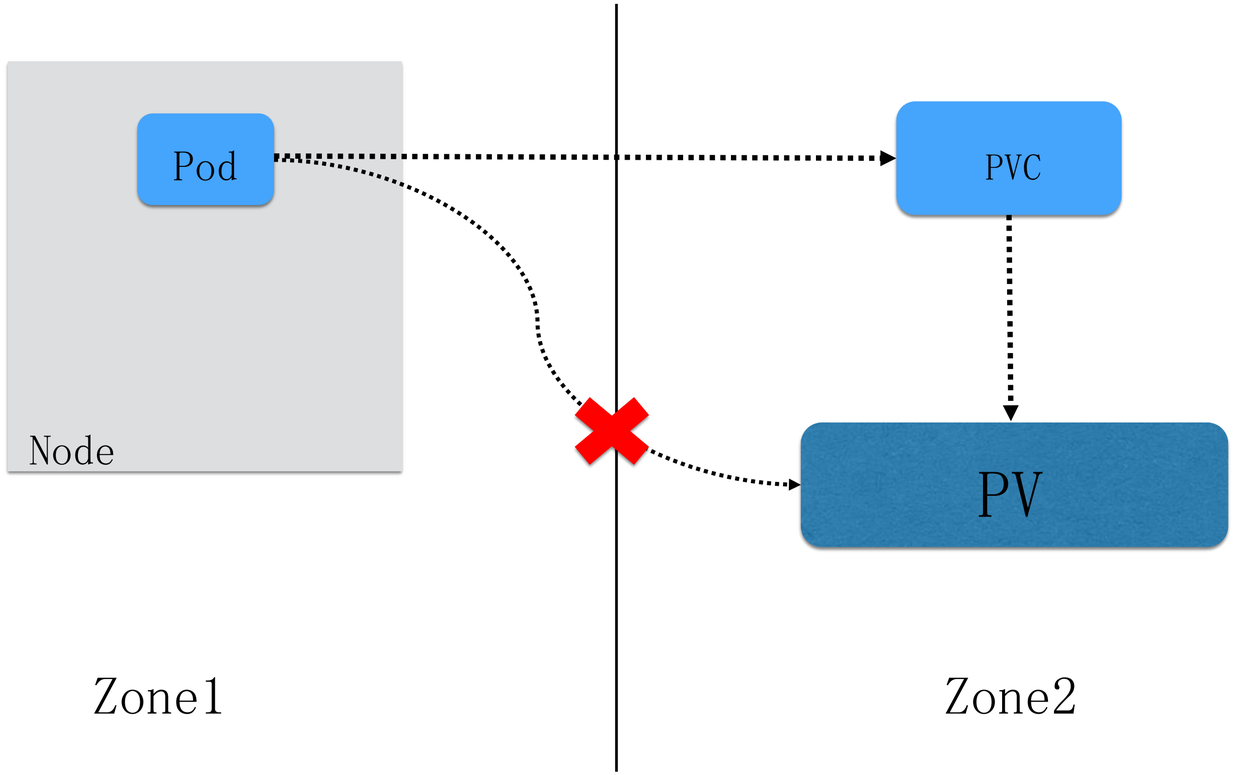
Kubernetes 中通过 PVC&PV 体系将存储与计算分离，即使用不同的 Controllers 管理存储资源和计算资源。但如果创建的 PV 有访问“位置”（.spec.nodeAffinity）的限制，也就是只有在特定的一些 nodes 上才能访问 PV，原有的创建 Pod 的流程与创建 PV 的流程的分离，就无法保证新创建的Pod 被调度到可以访问 PV 的 node 上，最终导致 Pod 无法正常运行起来。

场景1： Local PV 只能在指定的 Node 上被 Pod 使用



存储拓扑调度产生背景

场景2：单 Region 多 Zone K8s 集群，阿里云云盘当前只能被同一个 Zone 的 Node 上的 Pod 访问



存储拓扑调度

1. 本质问题

PV在Binding 或者 Dynamic Provisioning 时，并不知道使用它的Pod被会调度到哪些 Node 上？但 PV 本身的访问对 Node 的“位置”(拓扑)有限制。

2. 流程改进

Binding/Dynamic Provisioning PV 的操作 Delay 到 Pod 调度结果确定之后做，好处：

- 对于 pre-provisioned 的含有 Node Affinity 的 PV 对象，可以在 Pod 运行的 Node 确认之后，根据 Node 找到合适的 PV 对象，然后与 Pod 中使用的 PVC Binding，保证 Pod 运行的 Node 满足 PV 对访问”位置”的要求 。
- 对于dynamic provisioning PV 场景，在 Pod 运行的 Node 确认之后，可以结合 Node 的”位置”信息创建可被该 Node 访问的 PV 对象

3. Kubernetes 相关组件改进

- PV Controller：支持延迟Binding操作
- Dynamic PV Provisioner：动态创建PV时要结合Pod待运行的 Node 的”位置”信息
- Scheduler：选择Node时要考虑 Pod 的 PVC Binding 需求，也就是要结合 pre-provisioned 的 PV 的 Node Affinity以及 dynamic provisioning 时 PVC 指定 StorageClass.AllowedTopologies 的限制

1 基本知识

2 用例解读

3 操作演示

4 处理流程

5 课后习题

Volume Snapshot/Restore示例

```
# 创建VolumeSnapshotClass对象
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshotClass
metadata:
  name: disk-snapshotclass
snapshotter: diskplugin.csi.alibabacloud.com  # 指定Volume Snapshot时使用的Volume Plugin

# 创建VolumeSnapshot对象
apiVersion: snapshot.storage.k8s.io/v1alpha1
kind: VolumeSnapshot
metadata:
  name: disk-snapshot
spec:
  snapshotClassName: disk-snapshotclass
  source:
    name: disk-pvc       # Snapshot的数据源
    kind: PersistentVolumeClaim
```

```
# 从snapshot中恢复数据到新生成的PV对象中
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: restore-pvc
  namespace: simple
spec:
  dataSource:
    name: disk-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: csi-disk
```

Local PV示例

```
# 创建一个no-provisioner StorageClass对象，目的是告诉PV controller遇到 .spec.storageClassName 为 local-storage的PVC暂不做binding操作

kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: local-storage
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer   # 延时binding
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: local-pvc
spec:
  storageClassName: local-storage
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

```
# 创建Local PV对象
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv
spec:
  capacity:
    storage: 60Gi
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /share
  nodeAffinity:  # 限制该PV只能在node1上被使用
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname   # 拓扑域限制：单node可访问
          operator: In
          values:
          - node1
```

限制Dynamic Provision PV拓扑示例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-disk
provisioner: diskplugin.csi.alibabacloud.com
parameters:
   regionId: cn-hangzhou
   fsType: ext4
   type: cloud_ssd
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
-    matchLabelExpressions:
 # 拓扑域限制：动态创建的PV只能在可用区 cn-hangzhou-d被使用
 - key: topology.diskplugin.csi.alibabacloud.com/zone
   values:
   - cn-hangzhou-d
reclaimPolicy: Delete
```

```
# 当该PVC对象被创建之后由于对应StorageClass的 BindingMode为WaitForFirstConsumer并不会马上动态生成PV对象，而是要等到使用该PVC对象的第一个Pod调度结果出来之后，而且kube-scheduler在调度Pod的时候会去选择满足StorageClass.allowedTopologies中指定的拓扑限制的Nodes

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: disk-pvc
spec:
 accessModes:
 - ReadWriteOnce
 resources:
  requests:
    storage: 30Gi
 storageClassName: csi-disk
```

1 基本知识

2 用例解读

3 操作演示

4 处理流程

5 课后习题

云端环境实例操作示例

1. **Volume Snapshot**
2. **Local PV with Topology**
3. **Dynamic Provisioning with Topology**

1 基本知识

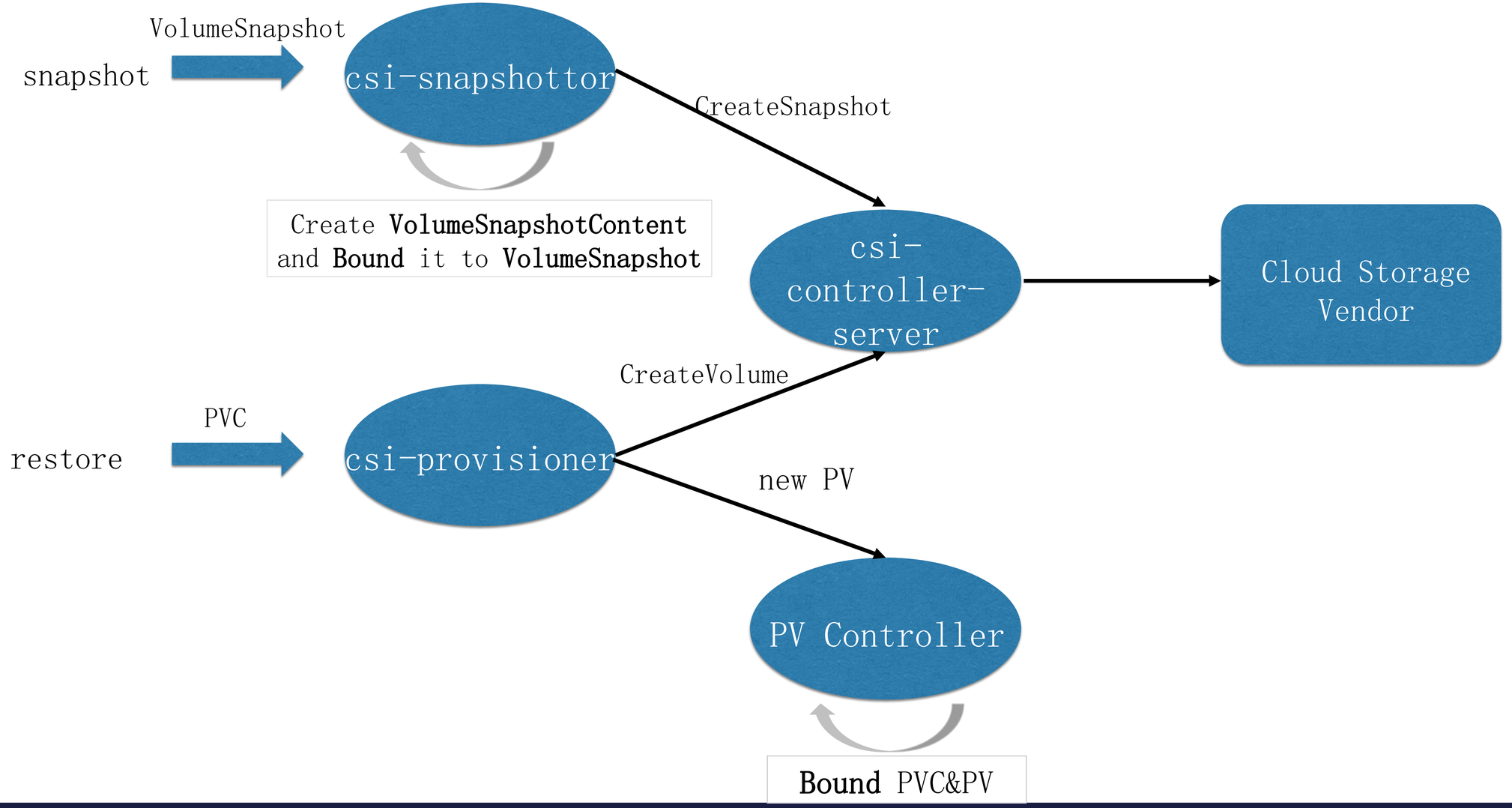
2 用例解读

3 操作演示

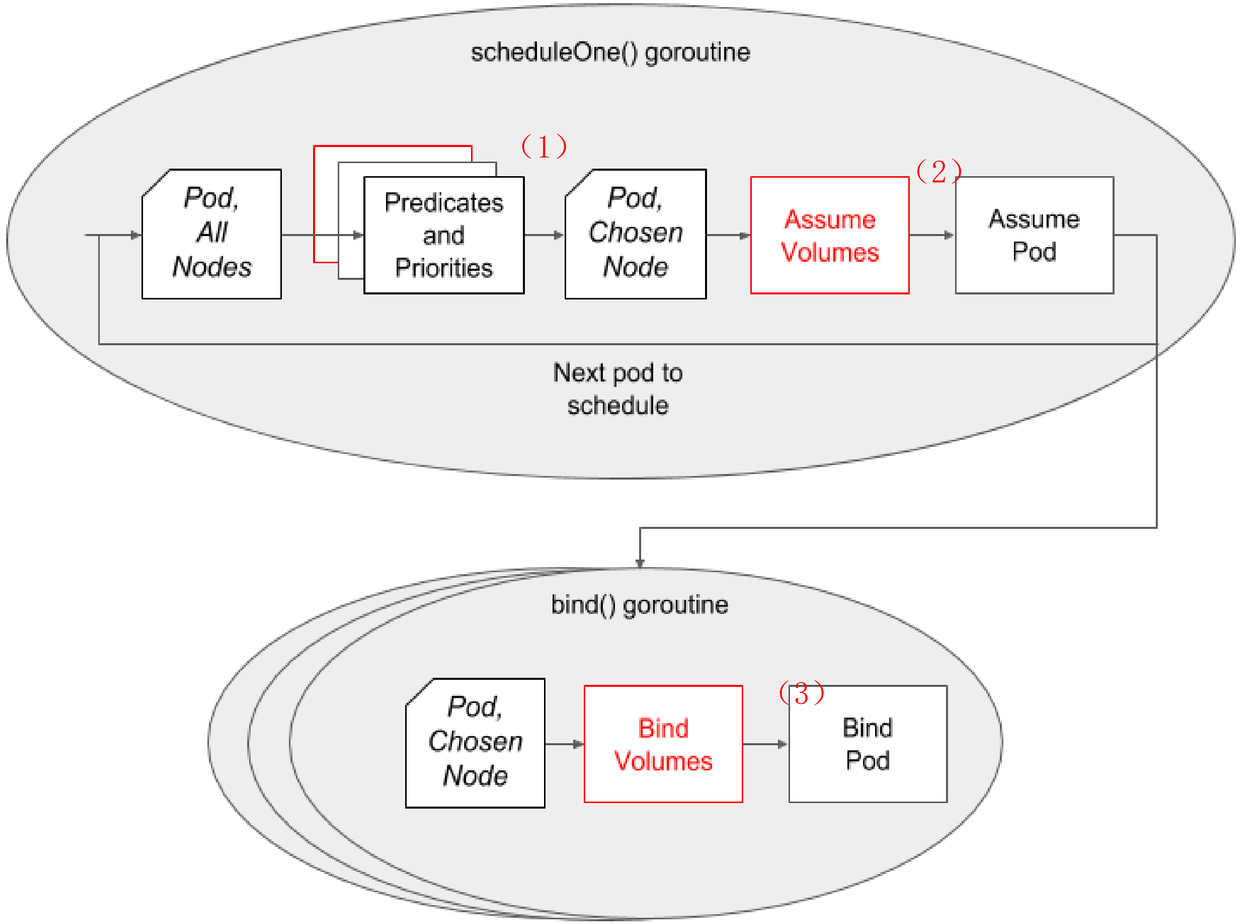
4 处理流程

5 课后习题

Kubernetes对Volume Snapshot/Restore处理流程



Kubernetes对Volume Topology-aware Scheduling处理流程



主流程说明：

1. 首先**PV Controller**对需要**Delay Binding**（通过**StorageClass**设置）的**PVC**暂不做任何处理

2. Scheduler根据**Pod PVCs**过滤**per Node**流程：

- 找到一个**Pod**所有**Bound**的**PVCs**以及需要**Delay Binding**的**PVCs**
- **Bound**的**PVCs**要**check bound**的**PV NodeAffinity**与当前**Node**的拓扑是否匹配，不匹配就**skip this Node**
- **Delay Binding**的**PVCs**，先**check**存量的**PVs**能满足**PVC**的列表，并将它们的**NodeAffinity**与当前**Node**拓扑做匹配，都不匹配进一步**check PVCs**对应的**StorageClass.AllowedTopologies**是否与**Node**的拓扑匹配，不匹配就**skip this Node**

3. 更新经过预选（**Predicates**）和优选（**Priorities**）选中**Node**的**Pod**在**scheduler**中的**PVC&PV cache**, 为**step(3)**做准备

4. 触发相关组件对**Pod**使用的**UnBound PVCs**的**Binding**或**Dynamic Provisioning**流程真正执行

1 基本知识

2 用例解读

3 操作演示

4 架构设计

5 课后习题

关注“阿里巴巴云原生”公众号
获取第一手技术资料