# StorageClass

持久化基础设施的资源映射

每个 `StorageClass` 都包含 `provisioner`、`parameters` 和 `reclaimPolicy` 字段,该资源定义不支持更新。

## 内置Provisioner

| 卷插件 | 内置 |
| --- | --- |
| AWSElasticBlockStore | ✓ |
| AzureFile | ✓ |
| AzureDisk | ✓ |
| CephFS | - |
| Cinder | ✓ |
| FC | - |
| FlexVolume | - |
| Flocker | ✓ |
| GCEPersistentDisk | ✓ |
| Glusterfs | ✓ |
| iSCSI | - |
| Quobyte | ✓ |
| NFS | - |
| RBD | ✓ |
| VsphereVolume | ✓ |
| PortworxVolume | ✓ |
| ScaleIO | ✓ |
| StorageOS | ✓ |
| Local | - |

## Reclaim Policy

- Delete
- Retain

## 可扩展性

PersistentVolume 可以配置为可扩展。将此功能设置为 true 时，允许用户通过编辑相应的 PVC 对象来调整卷大小。

| 卷类型 | Kubernetes 版本要求 |
| --- | --- |
| gcePersistentDisk | 1.11 |
| awsElasticBlockStore | 1.11 |
| Cinder | 1.11 |
| glusterfs | 1.11 |
| rbd | 1.11 |
| Azure File | 1.11 |
| Azure Disk | 1.11 |
| Portworx | 1.11 |
| FlexVolume | 1.13 |
| CSI | 1.14 (alpha), 1.16 (beta) |

# PV & PVC

pvc to pv 是一个对应关系，PVC管理PV的生命周期

回收策略：

- Retained
- Delete

PVC动态扩容：

```
allowVolumeExpansion: true
```

访问模式：

- ReadWriteOnce -- 卷可以被一个节点以读写方式挂载
- ReadOnlyMany -- 卷可以被多个节点以只读方式挂载
- ReadWriteMany -- 卷可以被多个节点以读写方式挂载

# kubernetes中的卷

kubernetes中的卷支持很多类型：

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- downwardAPI

```
volumes:
  - name: podinfo
    downwardAPI:
      items:
        - path: "labels"
          fieldRef:
            fieldPath: metadata.labels
        - path: "annotations"
          fieldRef:
            fieldPath: metadata.annotations
```

```
volumes:
  - name: podinfo
    downwardAPI:
      items:
        - path: "cpu_limit"
          resourceFieldRef:
            containerName: client-container
            resource: limits.cpu
            divisor: 1m
        - path: "cpu_request"
          resourceFieldRef:
            containerName: client-container
            resource: requests.cpu
            divisor: 1m
        - path: "mem_limit"
          resourceFieldRef:
            containerName: client-container
            resource: limits.memory
            divisor: 1Mi
        - path: "mem_request"
          resourceFieldRef:
```

```
            containerName: client-container
            resource: requests.memory
            divisor: 1Mi
```

- emptyDir
- gcePersistentDisk
- fc
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- rbd
- secret
- vsphereVolume

# rook-ceph

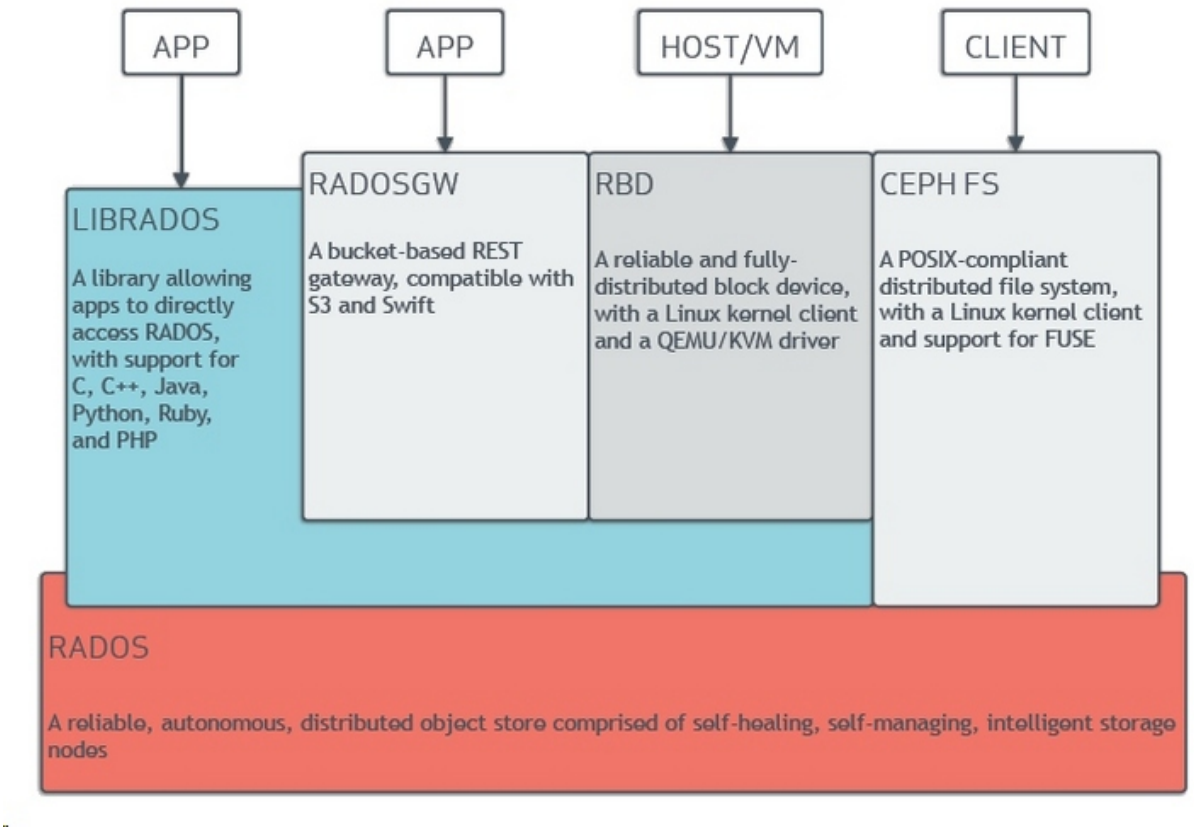
## Ceph是什么

三种存储类型：

- 对象存储
- 块设备
- 共享文件系统

使用rook在Kubernetes集群内部署Ceph集群

### Ceph架构图




## Ceph核心组件说明

- `RADOS`：
  - 可靠、自动、分布式对象存储（Reliable Autonomic Distributed Object Store），它是Ceph存储集群的核心
  - Ceph中的一切都是以对象的形式存储，RADOS管理对象，忽略类型
  - RADOS层确保数据一致性和可靠性，对于数据一致性，它执行数据复制、故障检测和恢复，还包括数据在集群节点间的迁移和再平衡。
- `OSD`：管理存储设备组件
- `MON`：Ceph 监视器（Ceph monitor），MON组件通过一系列的map来跟踪整个集群的健康状态，一个MON为每一个组件维护一个独立的map，如OSD、MON、PG、CRUSH
- `librados`：RADOS库，用以对接高级语言。librados是RBD和RGW的基础，并为CephFS提供POSIX接口。
- `RBD`：Ceph 块设备，它对外提供块存储，可以被映射、格式化进而像其他磁盘一样挂载到服务器。
- `RGW/RADOSGW`：Ceph 对象网关，它提供了一个兼容S3和Swift的restful API接口
- `MDS`：Ceph 元数据服务器，MDS跟踪文件层次结构并存储只供CephFS使用的元数据
- `CephFS`：Ceph 文件系统，CephFS提供了一个任意大小且兼容POSIX的分布式文件系统,CephFS依赖MDS来跟踪文件层次结构，即元数据。
- `MGR`： Ceph Manager，Ceph管理器软件，可以收集整个集群的所有状态。有仪表板插件
- `PG`：placement groups，归置组，一组对象的逻辑集合。
- `PGP`：PGP是为实现定位而设置的PG，它的值应该与PG的总数保持一致。
- `POOL`：Ceph 池，是一个用来存储对象的逻辑分区，方便管理对象。

# rook-ceph实践

### 部署

```
[root@controller01 ceph]# kubectl apply -f crds.yaml
[root@controller01 ceph]# kubectl apply -f common.yaml
[root@controller01 ceph]# kubectl apply -f operator.yaml
[root@controller01 ceph]# kubectl apply -f cluster.yaml
```

集群展示

`https://172.17.78.20:32742/#/dashboard`

获取`admin`用户密码

```
[root@controller01 ceph]# kubectl get secrets -n rook-ceph rook-ceph-dashboard-password -
o jsonpath={.data.password}|base64 -d && echo
```

# 块设备的StorageClass

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: rook-ceph-block
parameters:
  clusterID: rook-ceph
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: rook-ceph
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
  csi.storage.k8s.io/node-stage-secret-namespace: rook-ceph
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: rook-ceph
  imageFeatures: layering
  imageFormat: "2"
  pool: replicapool
provisioner: rook-ceph.rbd.csi.ceph.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

# CephFS的StorageClass

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: rook-cephfs
parameters:
```

```
  clusterID: rook-ceph
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: rook-ceph
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-cephfs-node
  csi.storage.k8s.io/node-stage-secret-namespace: rook-ceph
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-cephfs-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: rook-ceph
  fsName: neueducephfs
  pool: neueducephfs-data0
provisioner: rook-ceph.cephfs.csi.ceph.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

## CephFS PVC

```
[root@controller01 ~]# kubectl get pvc cephfs-pvc -o yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cephfs-pvc
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: rook-cephfs
  volumeMode: Filesystem
```

## CephFS PV

```
[root@controller01 ~]# kubectl get pv pvc-c8f94035-b97f-41d2-8332-002f2d4684a5 -o yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvc-c8f94035-b97f-41d2-8332-002f2d4684a5
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 10Gi
#  claimRef:
#    apiVersion: v1
#    kind: PersistentVolumeClaim
#    name: cephfs-pvc
#    namespace: default
#    resourceVersion: "73951099"
#    uid: c8f94035-b97f-41d2-8332-002f2d4684a5
  persistentVolumeReclaimPolicy: Delete
  storageClassName: rook-cephfs
  volumeMode: Filesystem
```