Rook: Storage Orchestrator for Kubernetes

Kiefer Chang
SUSE

Agenda

Kubernetes Recap

Ceph Introduction

Rook Introduction

Rook Architecture

Demo: Deloy a Ceph cluster with Rook

How to Contribute

About Me

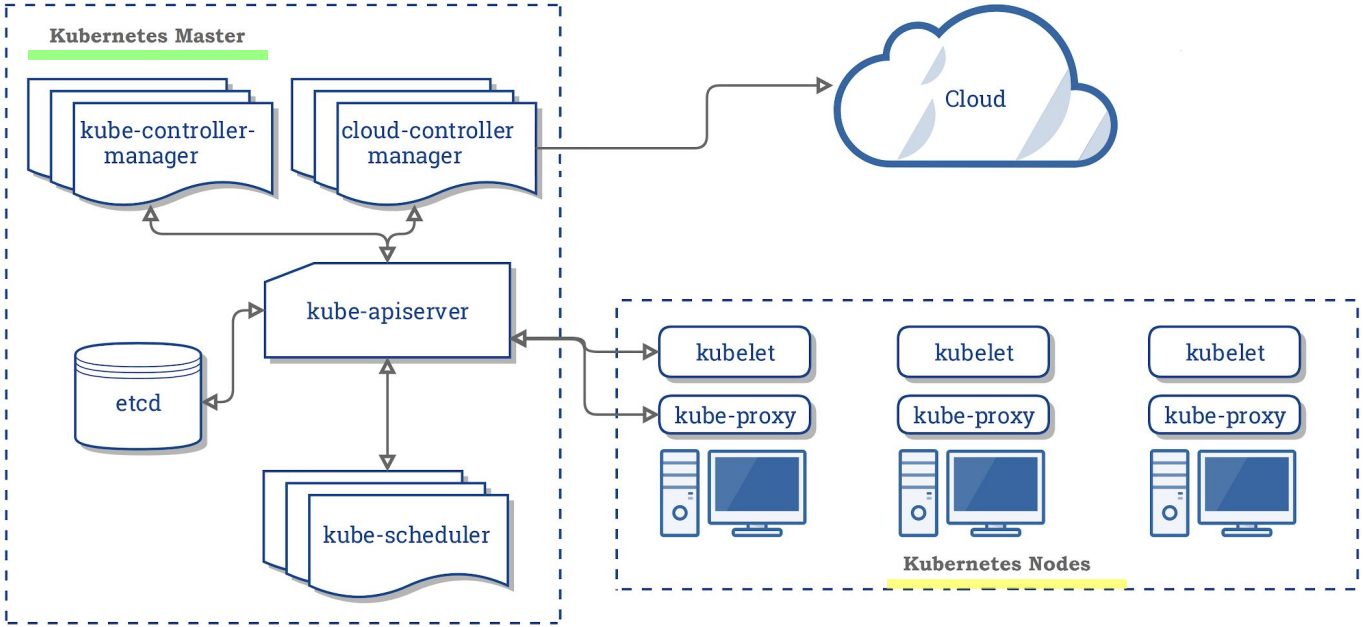
- Kiefer Chang
- Github: https://github.com/bk201
- IRC: kiefer_chang ({#ceph-dashboard,#ceph-devel}@OFTC)
- Experience
  - Storage development
  - OpenStack development (Nova, Cinder, Ironic,..etc.)
  - Ceph development (mainly Dashboard and Orchestrator)
- Work for SUSE (Enterprise Storage Team)

Kubernetes Recap

Session Overview

- K8S cluster and components
- Pods and controllers
- Storage
- ConfigMaps
- Secrets
- CRD

A Kubernetes Cluster

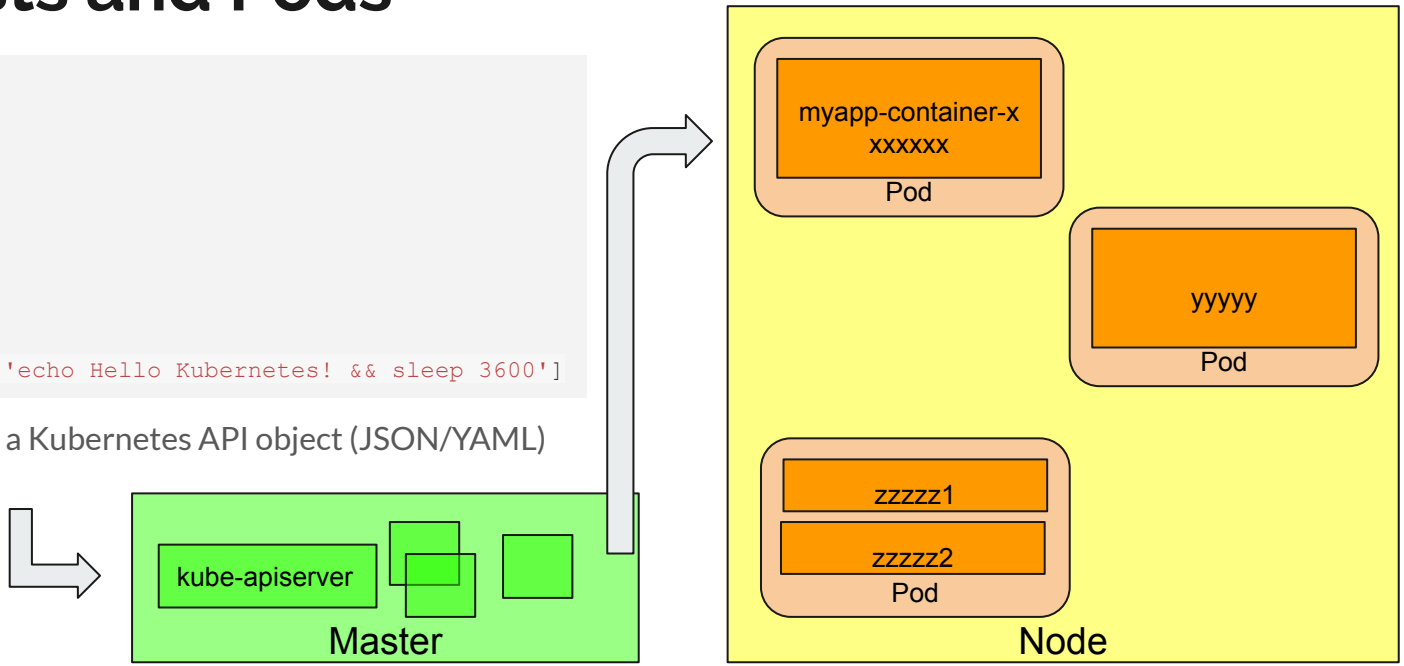


Manifests and Pods

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

Manifests: Specification of a Kubernetes API object (JSON/YAML)

- **Pods**: smallest unit to be scheduled, contain one or more containers



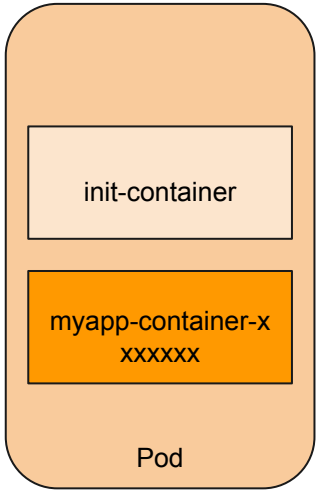
init Containers

Run before the app containers are started in a Pod

Run to completion and must complete successfully

Useful for

- Register services
- Wait for something
- Utilities



Pod Controllers

ReplicaSet

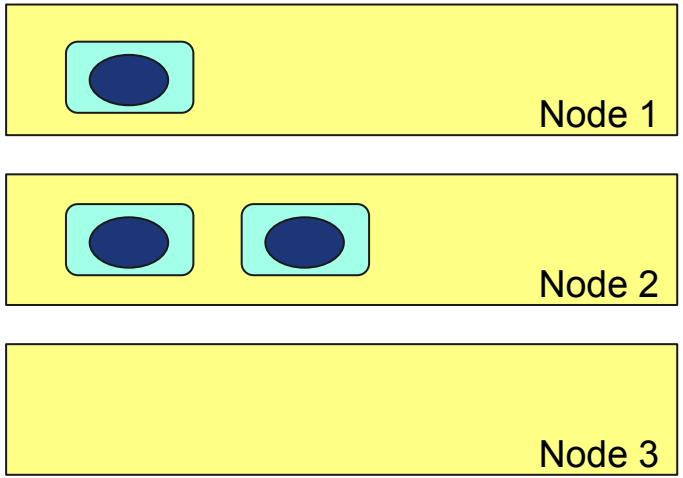
Deployments

DaemonSet

Jobs

ReplicaSet

maintain a stable set of replica Pods




```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

Deployment

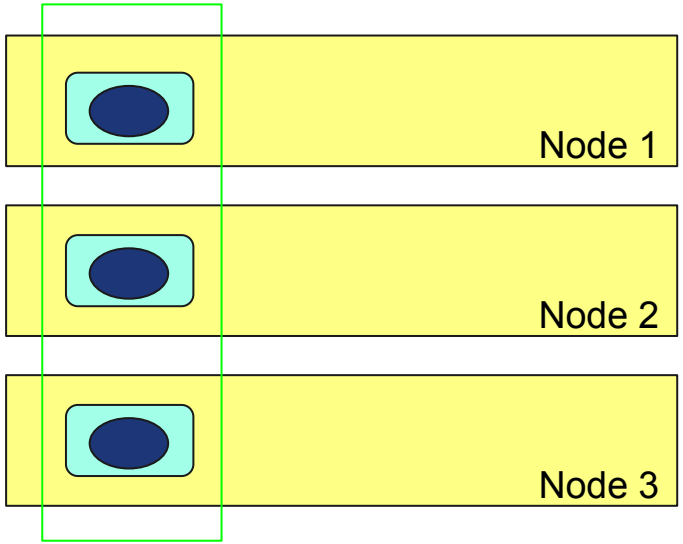
provides declarative updates for Pods and ReplicaSets

- Strategy to replace old Pods
  - Recreate
  - Rolling update*

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
          - containerPort: 80
```

DaemonSet

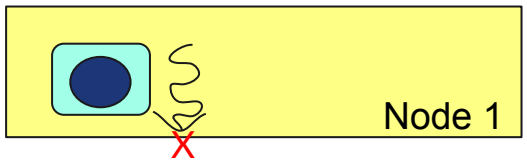
Ensures that all (or some) Nodes run a copy of a Pod



- Storage daemon, such as `glusterd`, `ceph`, on each node.
- Logs collection daemon on every node, such as `fluentd` or `logstash`.
- Monitoring daemon on every node, such as Prometheus Node Exporter..

Jobs

Creates one or more Pods and ensures that a specified number of them successfully terminate.




```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
    backoffLimit: 4
```

Storage

Volumes

Persistent Volumes

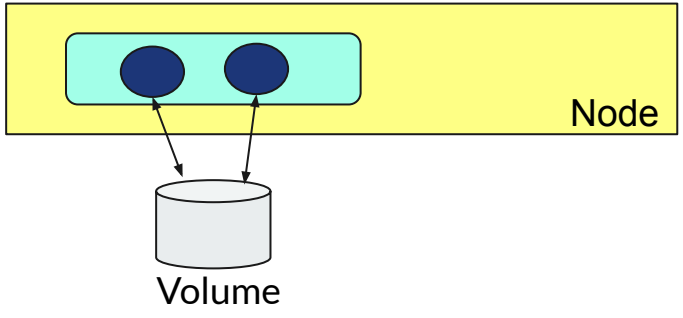
Storage class

Persistent Volume Claim

Volumes

Data are not lost when Pods are crashed and restarted

Share data between containers in a Pod



Types of Volumes

Kubernetes supports several types of Volumes:

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- csi
- downwardAPI
- emptyDir
- fc (fibre channel)
- flexVolume
- flocker
- gcePersistentDisk
- gitRepo (deprecated)
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- projected
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

Persistent Volumes

Volume data are not deleted after Pods termination

Provisioned

- by an administrator
- or dynamically

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Storage Classes

“Classes” of storage

QoS, durability, backup policy…, etc.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```

Persistent Volume Claim

A request for storage

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```

How to Claim a PV

POD

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim
```

PV

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: standard
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator:
In, values: [dev]}
```

Storage Class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```

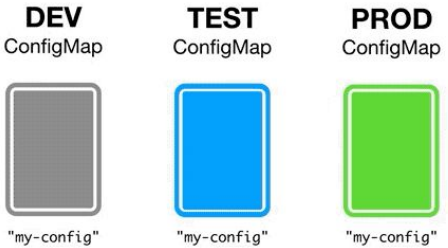
ConfigMaps

A dictionary of configs (key-value pairs)

Decouple configs and applications

Sources:

- files (ini, JSON, YMAL…), directory
- literals



https://matthewpalmer.net/kubernetes-app-developer/articles/ultimate-configmap-guide-kubernetes.html

Secrets

Store and manage sensitive information, such as passwords, OAuth tokens, and ssh keys

- Outside of Pods
- Stored in etcd

How to use secrets in Pods

- Volumes
- Environment variables

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
```

CRD (Custom Resource Definition)

Extends the Kubernetes API or allows you to introduce your own API

```yaml
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: cephfilesystems.ceph.rook.io
spec:
  group: ceph.rook.io
  names:
    kind: CephFilesystem
    listKind: CephFilesystemList
    plural: cephfilesystems
    singular: cephfilesystem
  scope: Namespaced
  version: v1
  validation:
    openAPIV3Schema:
      properties:
        spec:
          properties:
            metadataServer:
              properties:
                activeCount:
                  minimum: 1
                  maximum: 10
                  type: integer
                activeStandby:
                  type: boolean
                annotations: {}
                placement: {}
                resources: {}
```

Ceph Introduction

Ceph Introduction & How to Contribute

https://docs.google.com/presentation/d/1-UWfgiXsZjz50j7IYwNwtdAnFzWAbfMb7IAA8JvKwQs/edit#slide=id.g741af3ec10_1_0

What is Ceph

- An distributed scalable storage
- Open sourced (Mostly LGPL 2.1 & LGPL 3)
- Reliable storage on unreliable hardware
  - No SPoF (Single Point of Failure)
  - Erasure coding or replication for data durability
- Self-healing, auto rebalancing and placement optimization
- Run on commodity hardware
- Enterprise ready

ceph foundation

THE LINUX FOUNDATION

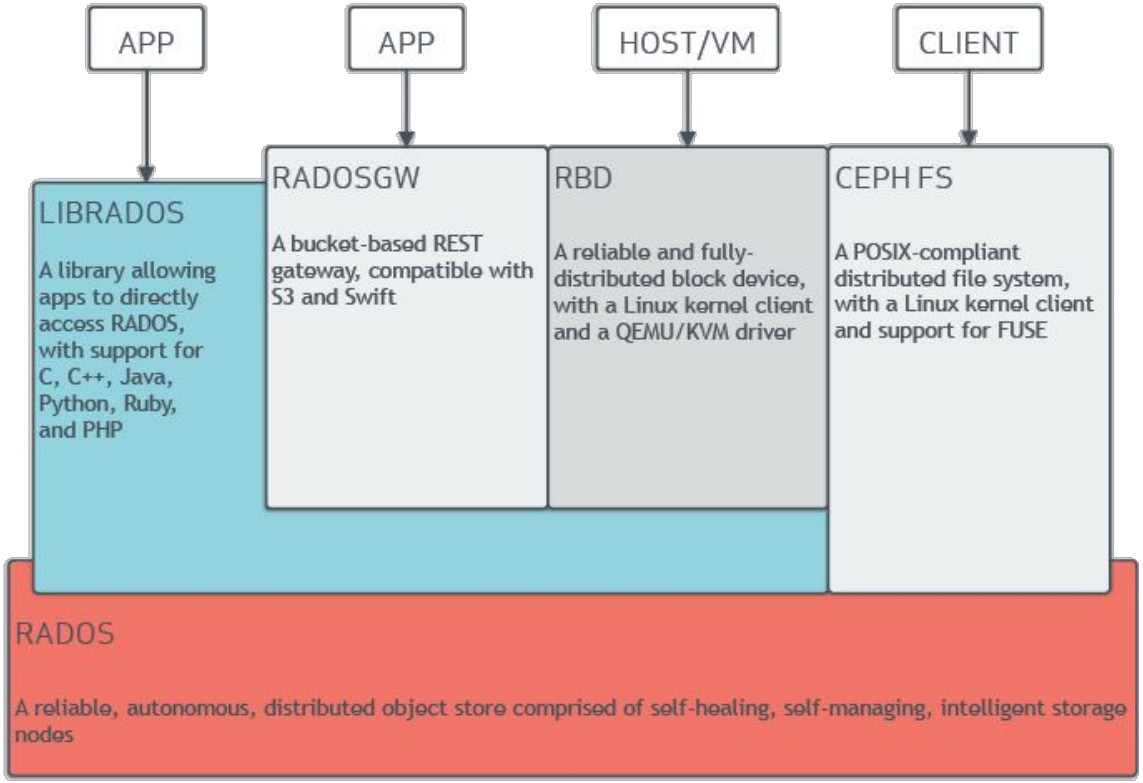
PREMIER MEMBERS

GENERAL MEMBERS

ASSOCIATE MEMBERS

Ceph is a Unified Storage

- Native protocols
  - Object (radosgw)
  - Block device (RBD)
  - File system (Ceph FS)
- Legacy protocols
  - iSCSI
  - NFS (on CephFS/radosgw)
  - Samba



Ceph Architecture - MONs and OSDs

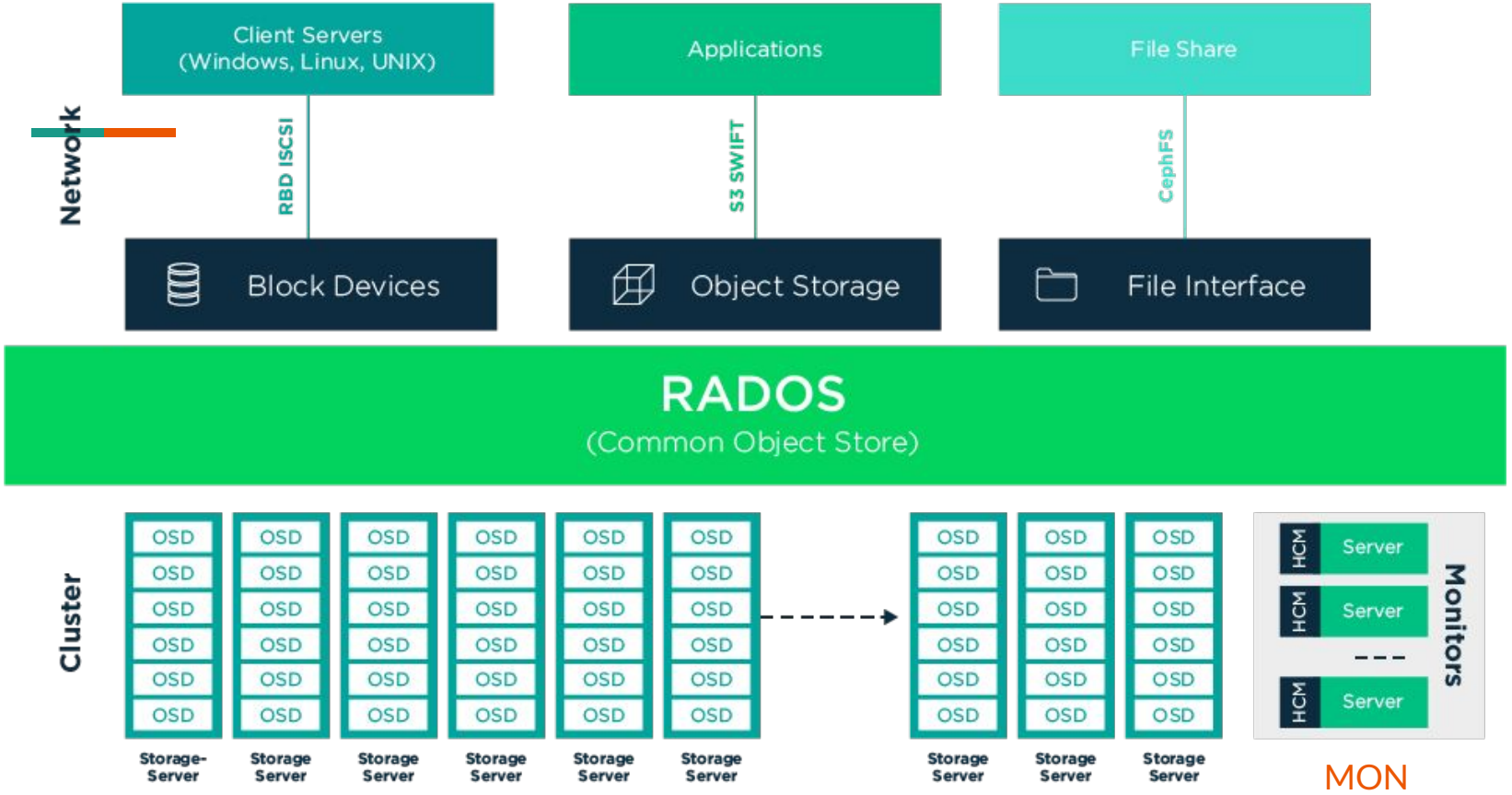
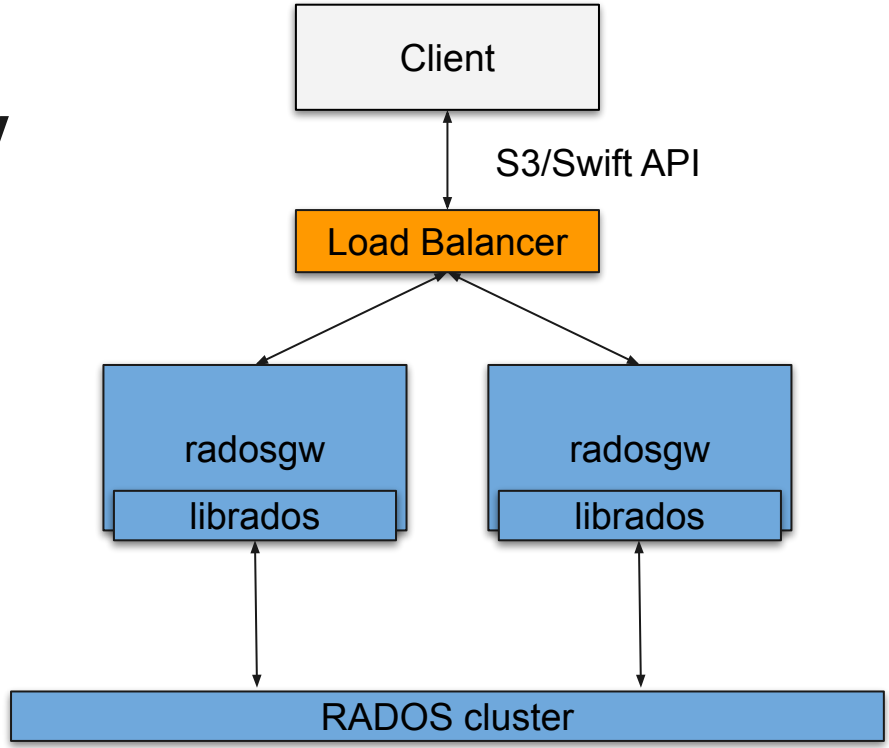


Image from: https://www.suse.com/products/suse-enterprise-storage/

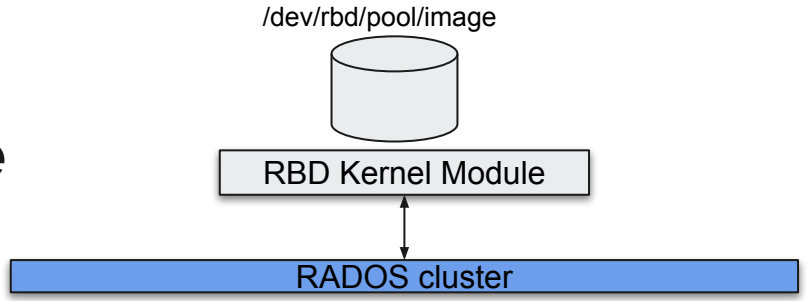
RGW: RADOS Gateway

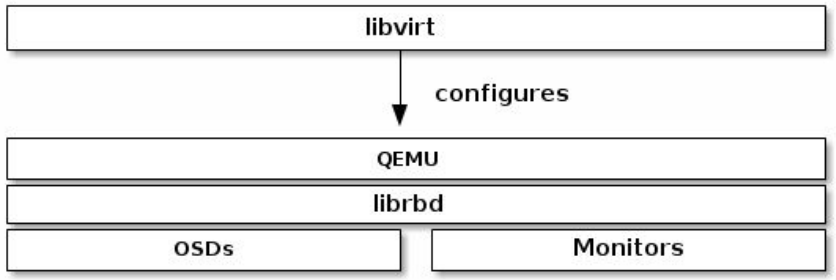
- S3/Swift compatible API
  - Users/buckets/objects
  - Encryption
  - Compression
  - Object expiry
  - CORS
- Use cases
  - Public cloud storage
  - Static files
  - VM images
  - Archiving and backup
  - NFS gateway



RBD: RADOS Block Device

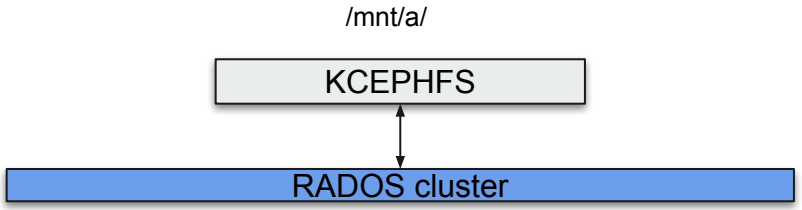
- KVM (qemu-kvm) and Linux clients
- Thin-provisioned
- Snapshots and clones
- Cross-sites mirroring
- Use cases
  - VM image/disks
    - For OpenStack components like Nova, Glance, and Cinder
    - Proxmox, Nebula...
  - Kubernetes persistent volume
  - iSCSI gateway





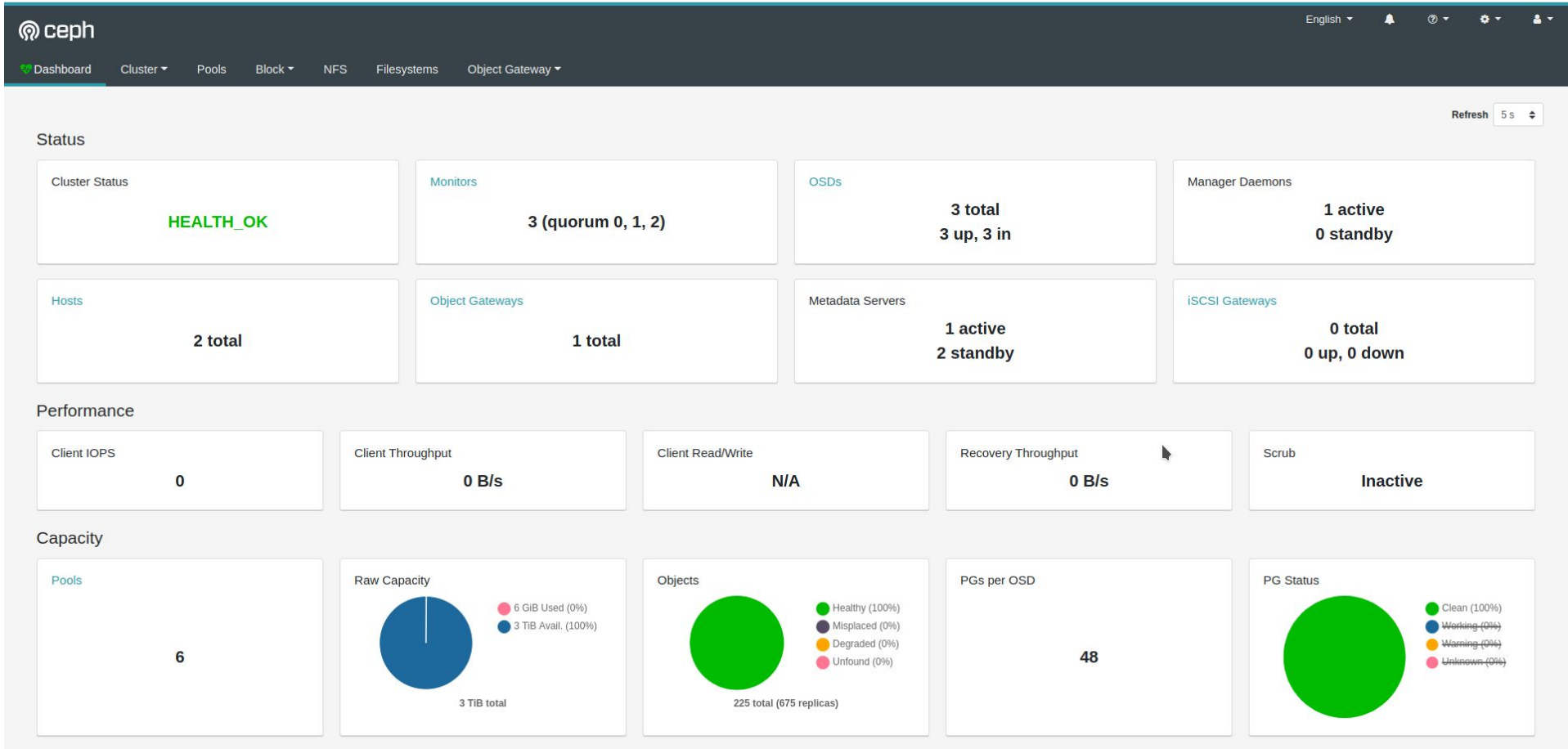
CephFS: File System

- Shared file system
- Strong consistency
- Scale metadata/data independently
- Snapshots and quotas
- xattrs and lockings features
- Use cases
  - Use as traditional FS, it's nearly POSIX-compatible*
  - Shared file system for VMs, like OpenStack Manila
  - NFS gateway for windows/others

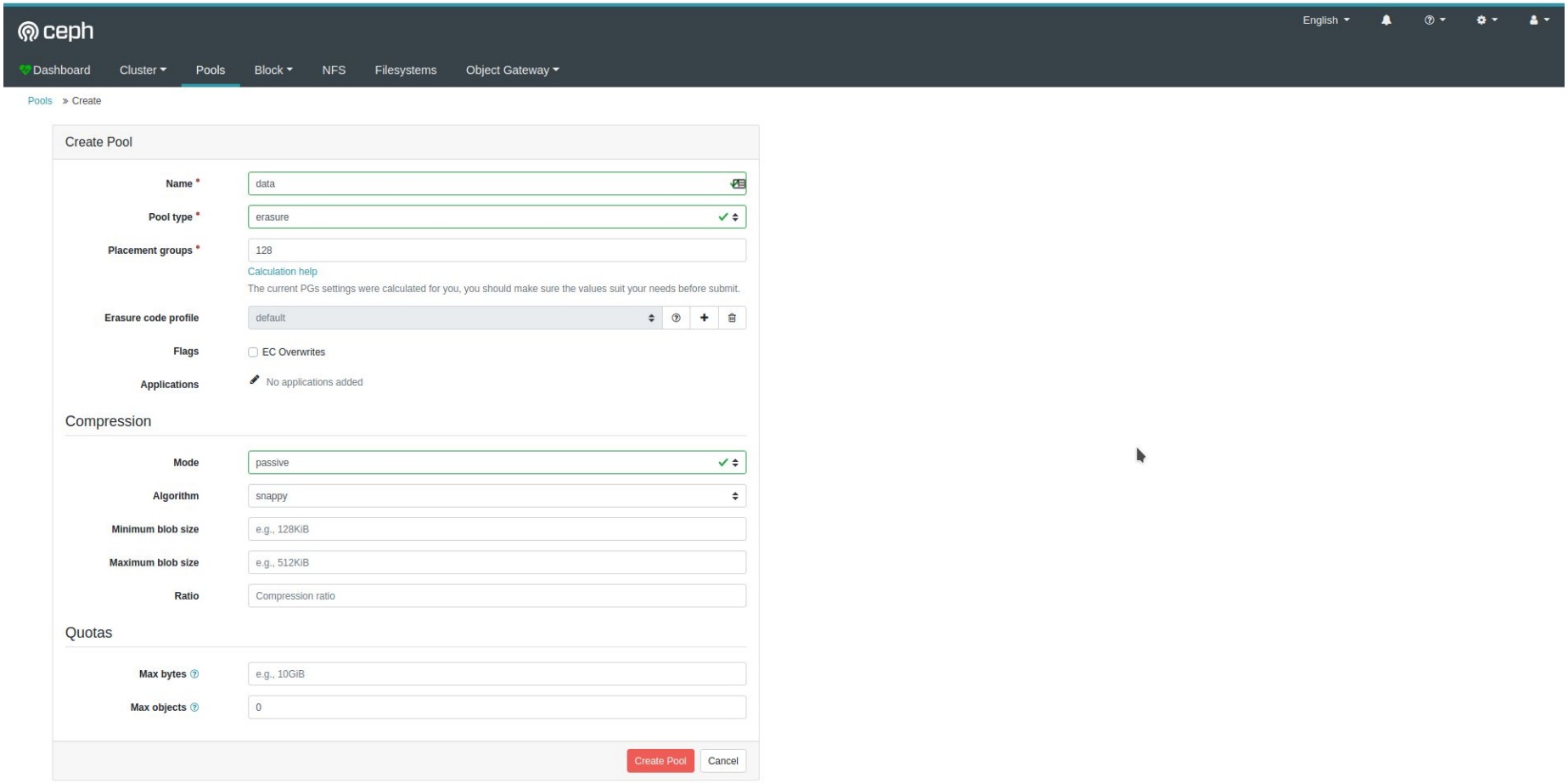


* https://docs.ceph.com/docs/master/cephfs/posix/

Dashboard - Summary



Dashboard - Day 2 Operations



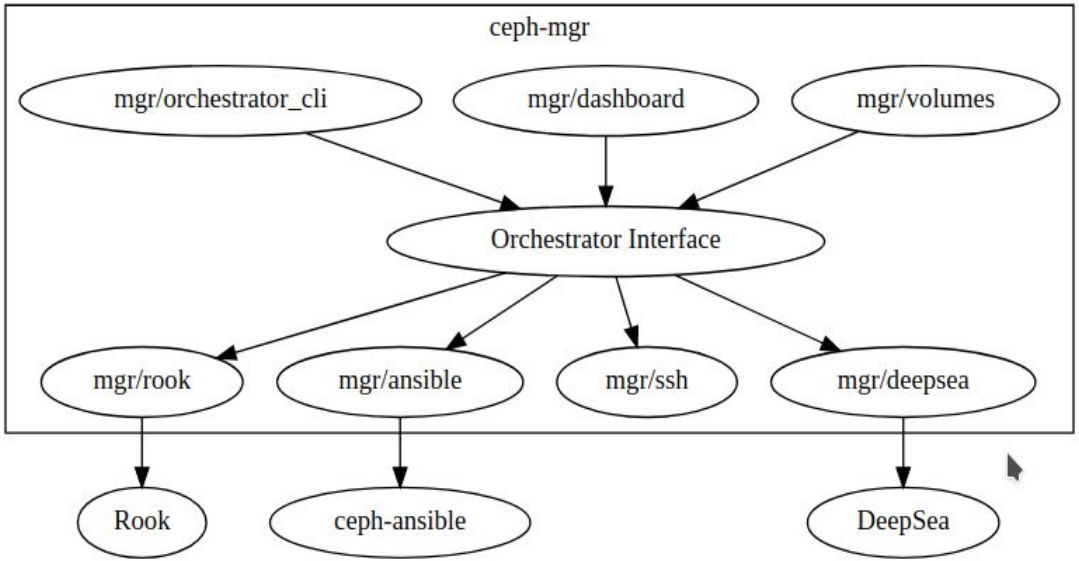
Dashboard - Monitoring Metrics (Grafana Panels)

Deployment Tools

- ceph-deploy
  - https://docs.ceph.com/docs/master/mgr/ssh/
  - ssh + commands
  - Likely to be deprecated, moving to ssh orchestrator
- ssh-orchestrator (under active develop, new in Octopus)
  - https://docs.ceph.com/docs/master/mgr/ssh/
  - ssh + ceph-daemon (https://docs.ceph.com/docs/master/bootstrap/#)
- DeepSea
  - Salt-based
  - https://github.com/SUSE/DeepSea
- Ceph-ansible
  - https://github.com/ceph/ceph-ansible
  - ansible playbooks
- Rook
  - CRD and operator images to deploy Ceph in Kubernetes
  - https://github.com/rook/rook/

Ceph Orchestrator Module

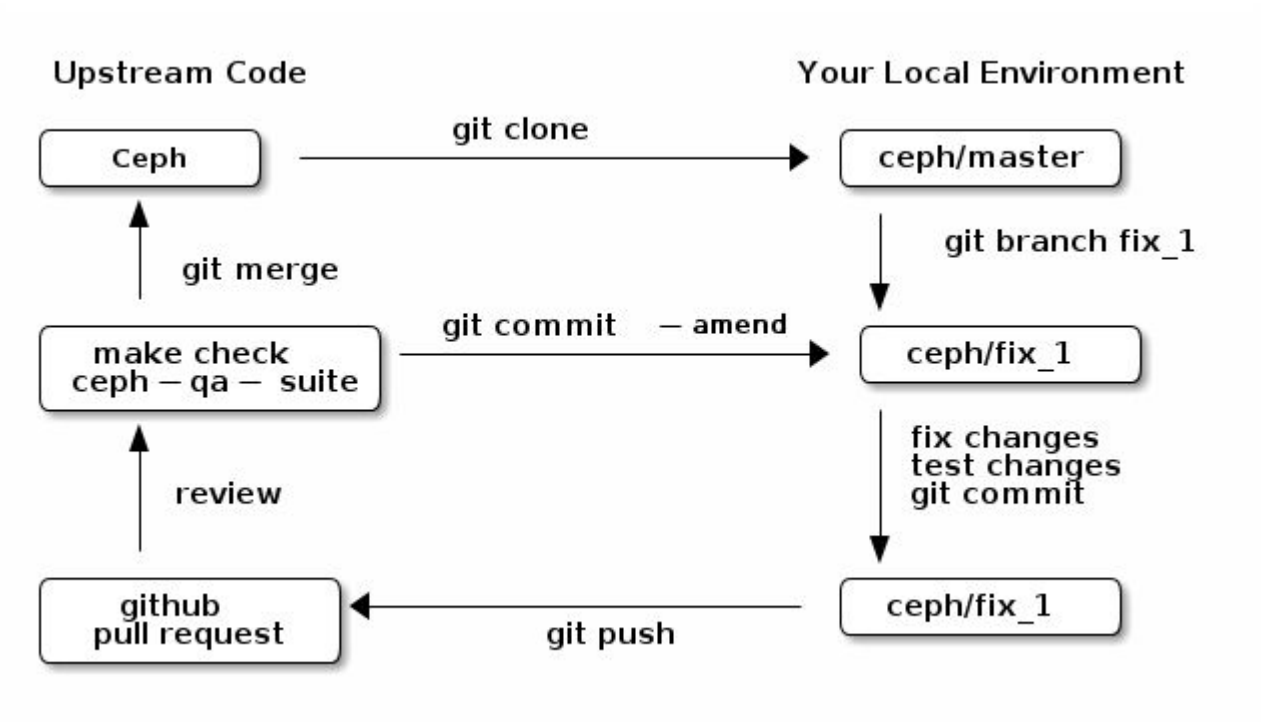
- Common interface for different orchestrators
  - Let Dashboard/CLI/Utilities work with various different backends



Contribution Working Flow

NOTE

Always create an issue on tracker first if possible



https://docs.ceph.com/docs/master/dev/developer_guide

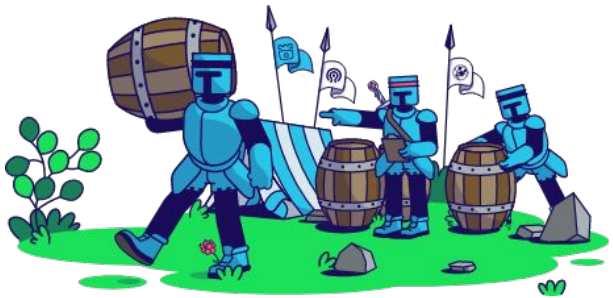
Rook Introduction (Ceph-Rook)

What is Rook

A storage orchestrator for Kubernetes

Leverage Kubernetes' platform power to *bootstrap and manage storage* in Kubernetes

- Ceph
- EdgeFS
- Minio
- CockroachDB
- Cassandra
- NFS
- Yugabyte DB



QuickStart: Ceph

TL;DR (If you have a running Kubernetes cluster)

```
git clone https://github.com/rook/rook.git
cd rook
git checkout v1.1.6

cd cluster/examples/kubernetes/ceph
kubectl create -f common.yaml
kubectl create -f operator.yaml
kubectl create -f cluster-test.yaml
```

** Not for production **

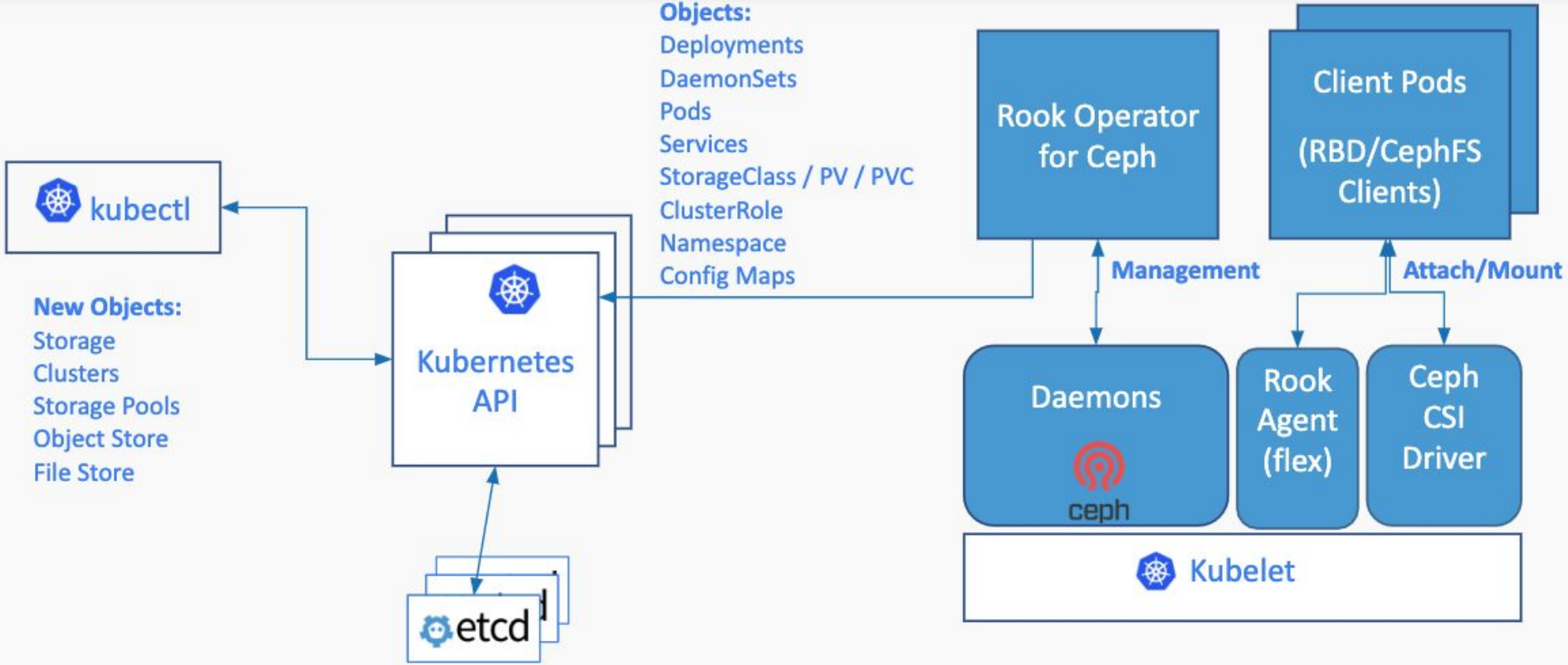
The Operator Pattern

Provision a application in Kubernetes to do the rest of works:

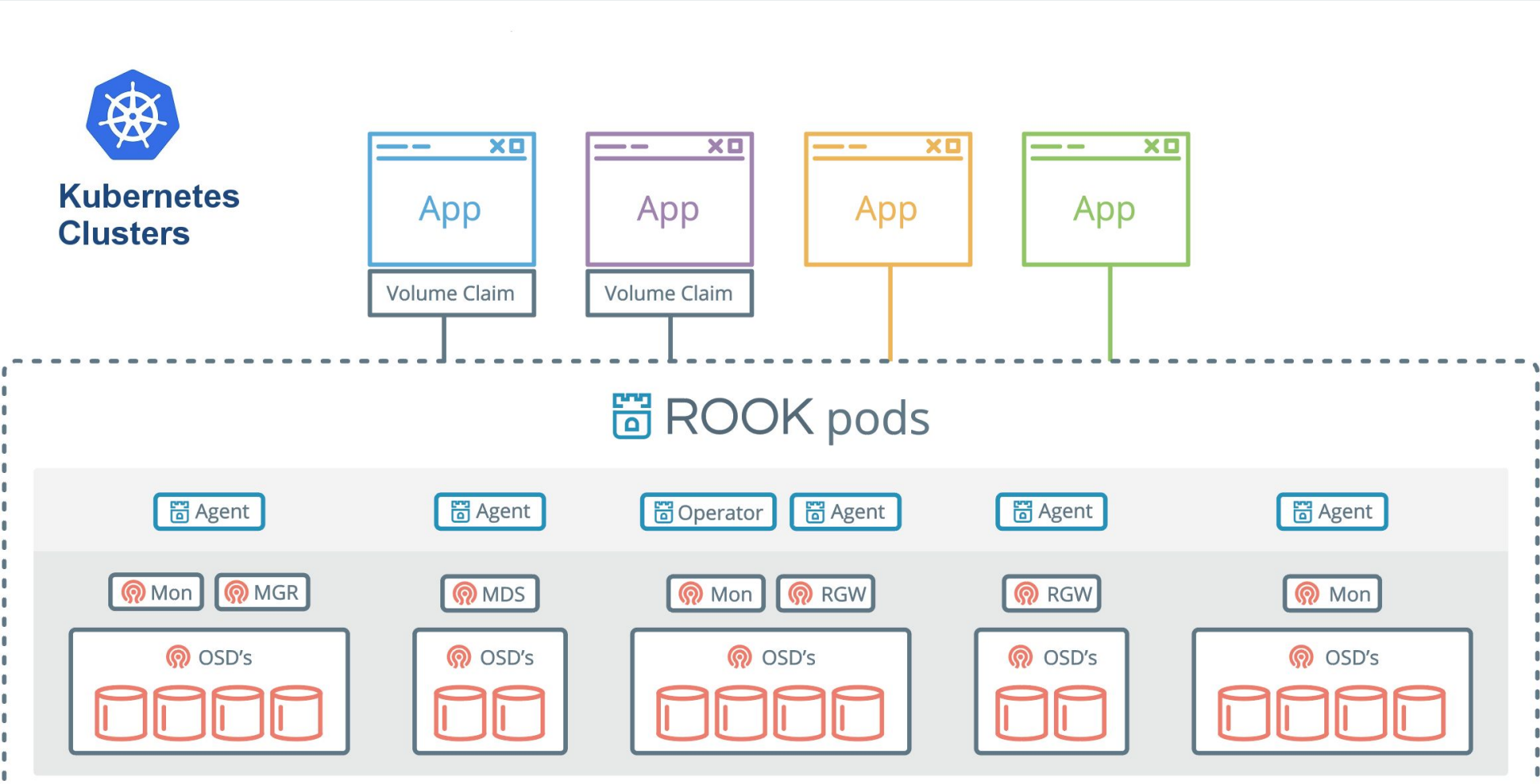
- Bootstrap the Ceph cluster
- Manage Day-2 operations
  - Adding/removing daemons and OSDs
  - Scale out
  - Upgrade

you still need real operators...

Rook Architecture



Rook Architecture



Deploy a Ceph Cluster with Rook

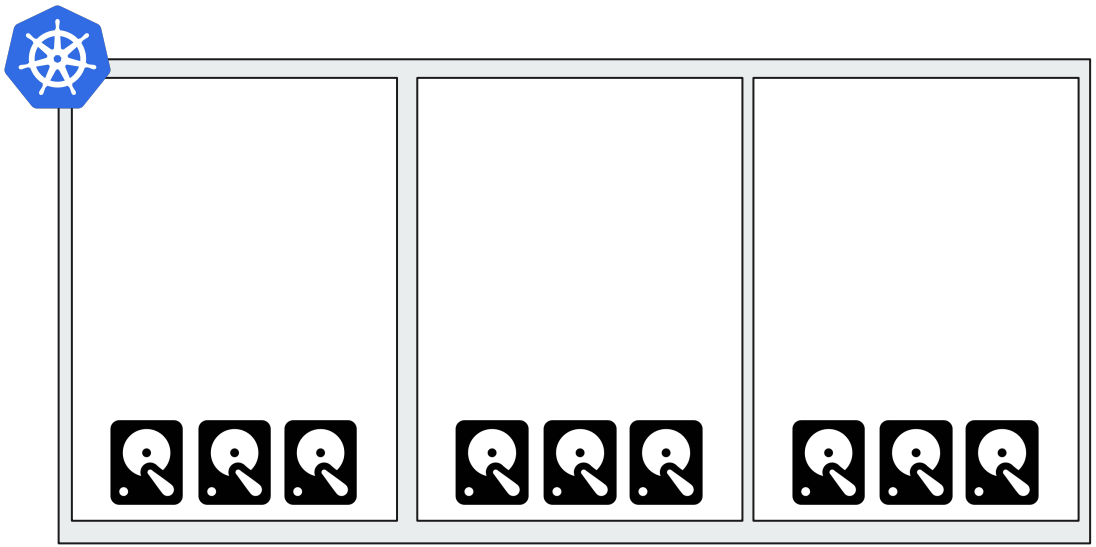
Materials are extracted from Blaine Gardner's presentation: Getting Started as a Rook-Ceph Developer in Cephalocon

Recording: https://youtu.be/P2ZDIdEPyjw

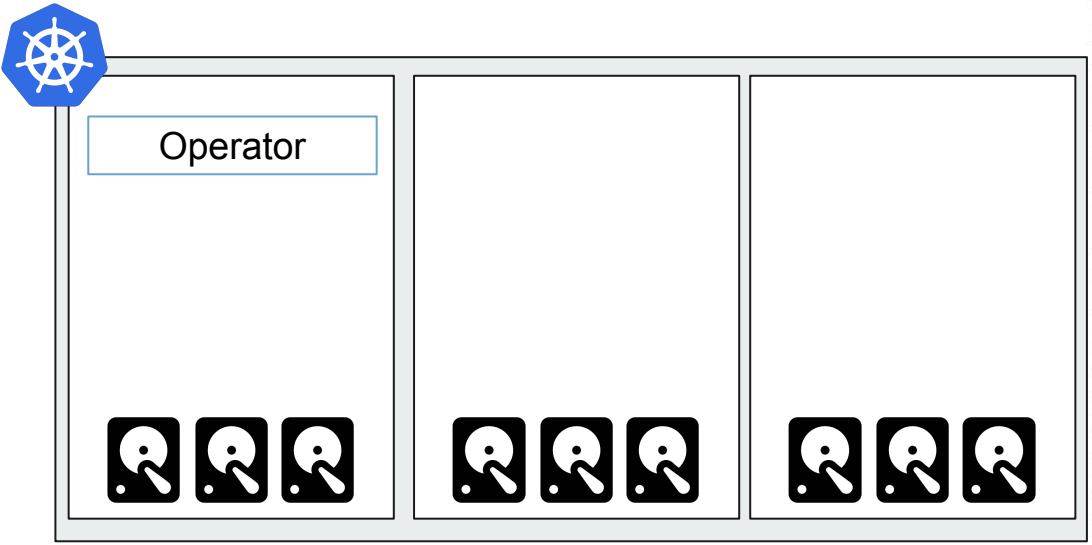
Slides: https://static.sched.com/hosted_files/cephalocon2019/39/Intro%20to%20Rook%20Dev.pdf

Create CRDs (cluster/examples/kubernetes/ceph/common.yaml)

```yaml
# Namespace where the operator and other rook resources are created
apiVersion: v1
kind: Namespace
metadata:
  name: rook-ceph
# OLM: BEGIN CEPH CRD
# The CRD declarations
---
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: cephclusters.ceph.rook.io
spec:
  group: ceph.rook.io
  names:
    kind: CephCluster
    listKind: CephClusterList
    plural: cephclusters
    singular: cephcluster
  scope: Namespaced
  version: v1
```



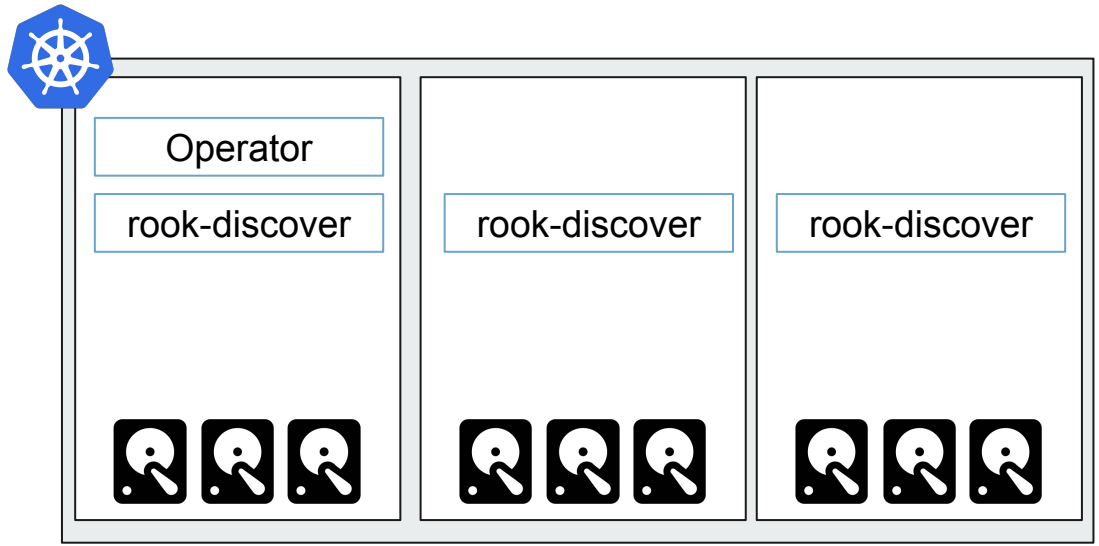
Deploy the Operator Pod



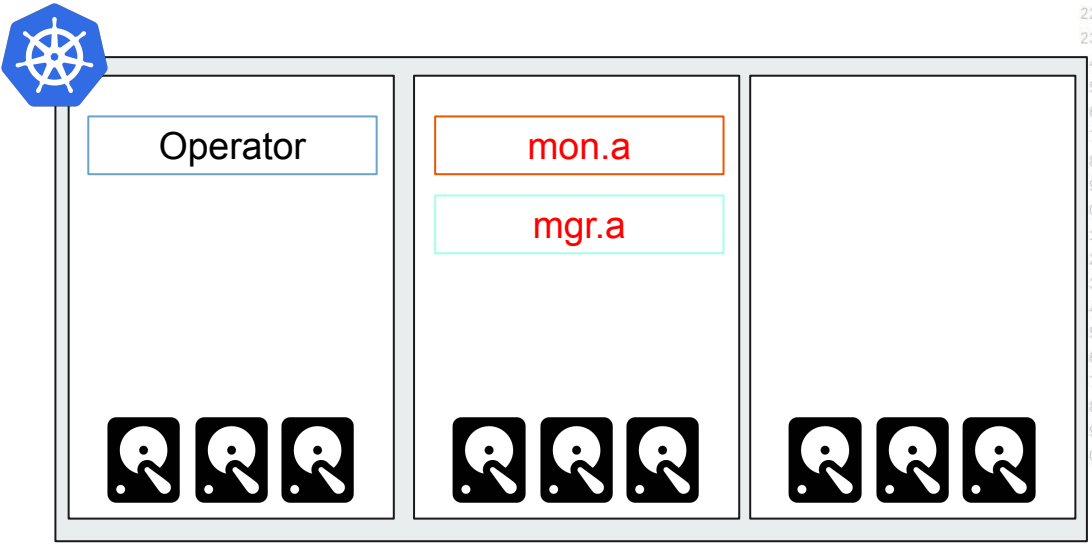

```yaml
# OLM: BEGIN OPERATOR DEPLOYMENT
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rook-ceph-operator
  namespace: rook-ceph
  labels:
    operator: rook
    storage-backend: ceph
spec:
  selector:
    matchLabels:
      app: rook-ceph-operator
  replicas: 1
  template:
    metadata:
      labels:
        app: rook-ceph-operator
    spec:
      serviceAccountName: rook-ceph-system
      containers:
      - name: rook-ceph-operator
        image: rook/ceph:master
        args: ["ceph", "operator"]
```

cluster/examples/kubernetes/ceph/operator.yaml

Operator creates Discover Pods



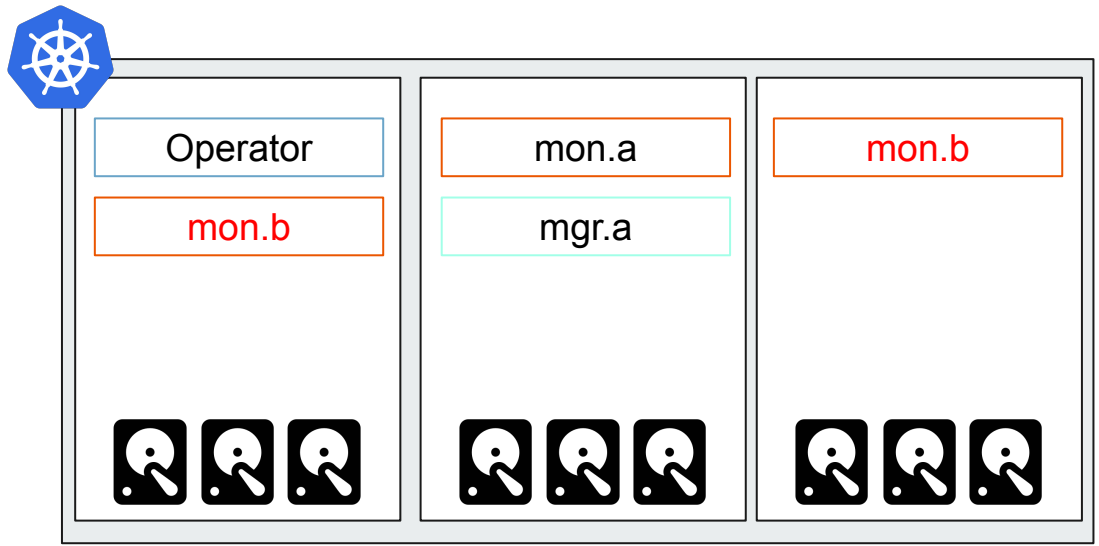
Create Ceph Cluster



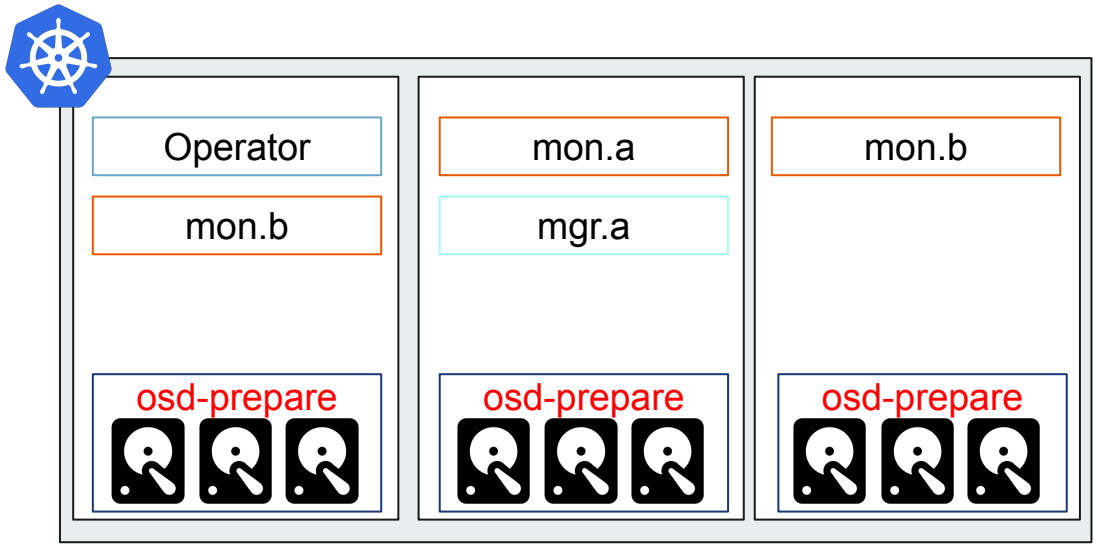

```yaml
apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph
  namespace: rook-ceph
spec:
  cephVersion:
    # The container image used to launch the Ceph daemon pods (mon, mgr, osd, mds, rgw).
    # v13 is mimic, v14 is nautilus, and v15 is octopus.
    # RECOMMENDATION: In production, use a specific version tag instead of the general v14
    # versions running within the cluster. See tags available at https://hub.docker.com/r/
    image: ceph/ceph:v14.2.4-20190917
    # Whether to allow unsupported versions of Ceph. Currently mimic and nautilus are supp
    # Octopus is the version allowed when this is set to true.
    # Do not set to true in production.
    allowUnsupported: false
  # The path on the host where configuration files will be persisted. Must be specified.
  # Important: if you reinstall the cluster, make sure you delete this directory from each
  # In Minikube, the '/data' directory is configured to persist across reboots. Use "/data
  dataDirHostPath: /var/lib/rook
  # Whether or not upgrade should continue even if a check fails
  # This means Ceph's status could be degraded and we don't recommend upgrading but you mi
  # Use at your OWN risk
  # To understand Rook's upgrade process of Ceph, read https://rook.io/docs/rook/master/ce
  skipUpgradeChecks: false
  # set the amount of mons to be started
  mon:
    count: 3
    allowMultiplePerNode: false
```

cluster/examples/kubernetes/ceph/cluster.yaml

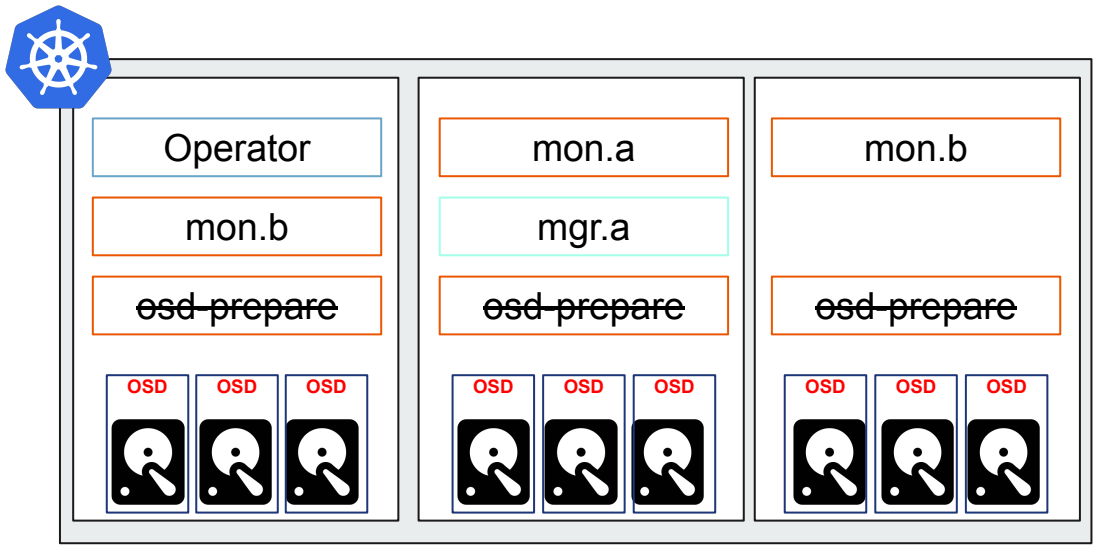
Create Ceph Cluster (cnt'd)



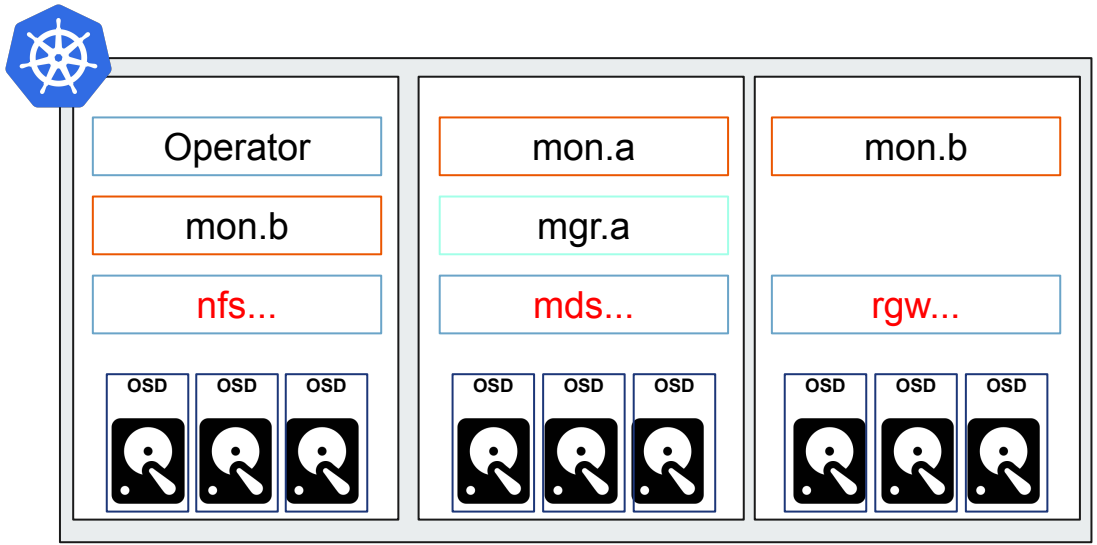
Create Ceph Cluster (cnt'd)



Create Ceph Cluster (cnt'd)



Create Ceph Cluster (cnt'd)



Claim a RBD Volume

Create a RBD storage class

Claim it in your Pod

```yaml
apiVersion: ceph.rook.io/v1
kind: CephBlockPool
metadata:
  name: replicapool
  namespace: rook-ceph
spec:
  failureDomain: host
  replicated:
    size: 1
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: rook-ceph-block
provisioner: rook-ceph.rbd.csi.ceph.com
parameters:
    # clusterID is the namespace where the rook cluster is running
    # If you change this namespace, also change the namespace below where the secret namespaces are d
    clusterID: rook-ceph

    # Ceph pool into which the RBD image shall be created
    pool: replicapool

    # RBD image format. Defaults to "2".
    imageFormat: "2"

    # RBD image features. Available for imageFormat: "2". CSI RBD currently supports only `layering`
    imageFeatures: layering
```

cluster/examples/kubernetes/ceph/csi/rbd/storageclass-test.yaml

```yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  storageClassName: rook-ceph-block
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
    tier: frontend
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.6.1-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          value: changeme
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```

cluster/examples/kubernetes/wordpress.yaml

Contribution

Community

slack https://rook-io.slack.com/ #ceph-dev

https://github.com/rook/rook

Flow: https://rook.io/docs/rook/v1.1/development-flow.html

Fork → Clone → Create local branch → Hack → Push branch→ Create PR → Review/CI → Merge

Most Rook codes are in Golang.

Some modules like orchestrator are in Ceph project (Python)

Demo

Deploy a Ceph cluster with Rook

Dashboard & Orchestrator

Persistent Volume Claim with RBD