KubeCon | CloudNativeCon

Europe 2020

Virtual

ROOK-CEPH DEEP DIVE

Travis Nielsen, Red Hat
Sébastien Han, Red Hat

20 Aug 2020

ROOK

ceph

Project Status

- CNCF Incubating project since September 2018
- CNCF Graduation voting is in progress
  - Hopefully completed by now!
- Latest release: v1.4

What is Rook?

- Open Source
- Storage Operators for Kubernetes
- Automates Management of Ceph
  - Deployment
  - Configuration
  - Upgrading

What is Ceph?

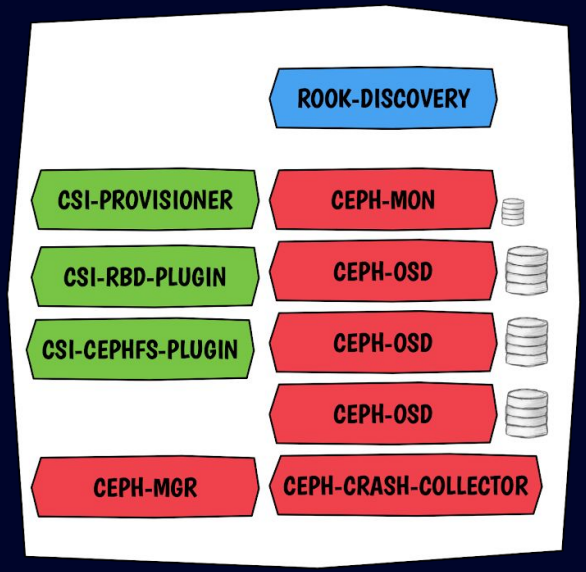
- Open Source
- Distributed storage software-defined solution
  - Block
  - Shared File System
  - Object Storage (S3 compliant)

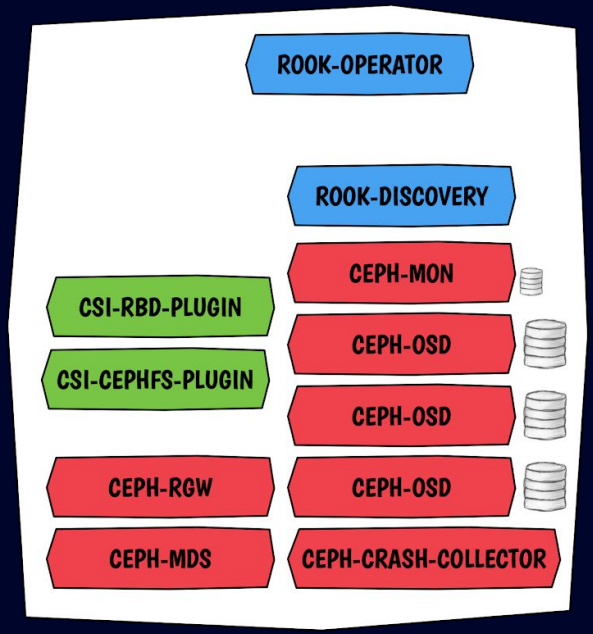
ARCHITECTURE

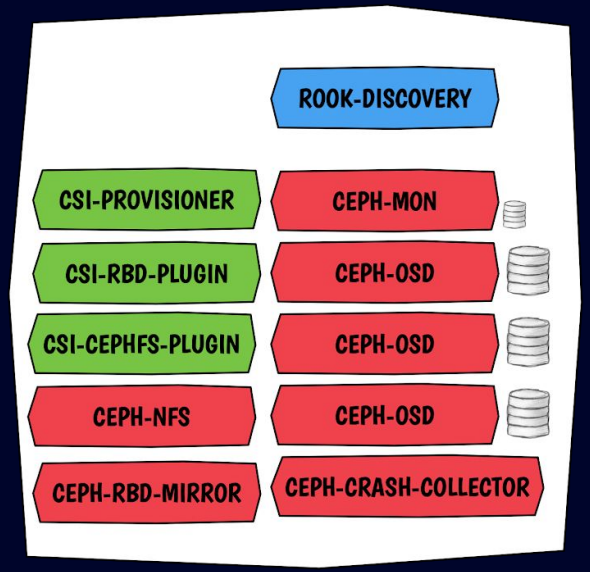
Architectural Layers

- Rook:
  - The operator owns the **management** of Ceph
- Ceph-CSI:
  - CSI driver dynamically **provisions** and **connects** client pods to the storage
- Ceph:
  - **Data** layer

Layer 1: Rook Management



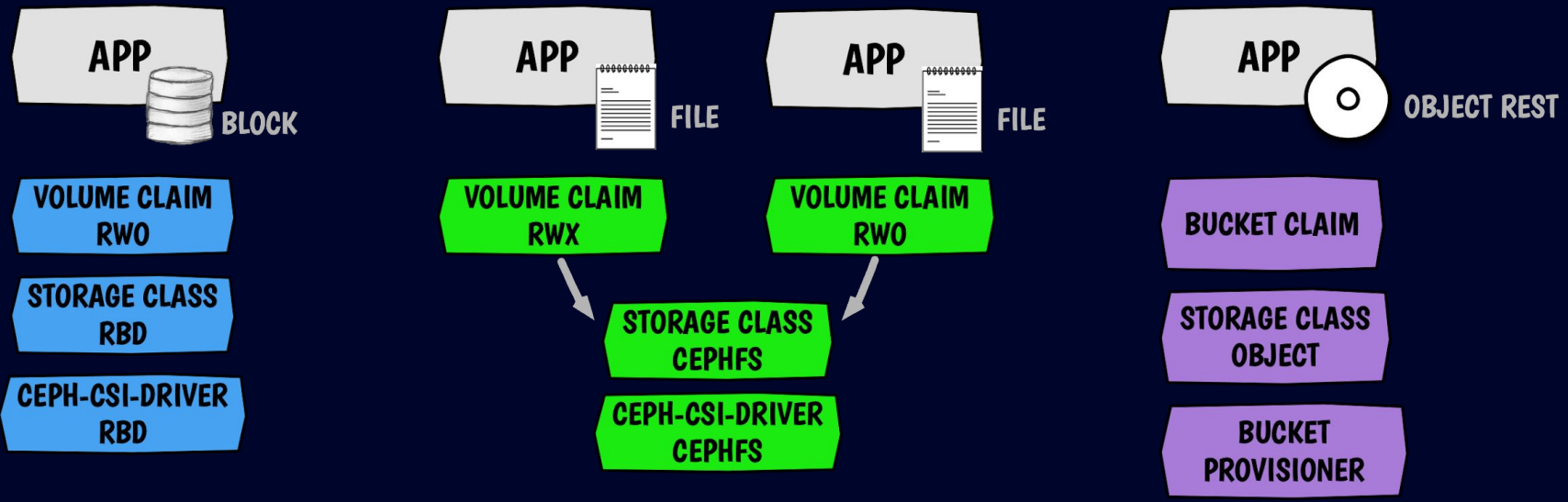




ROOK

ceph

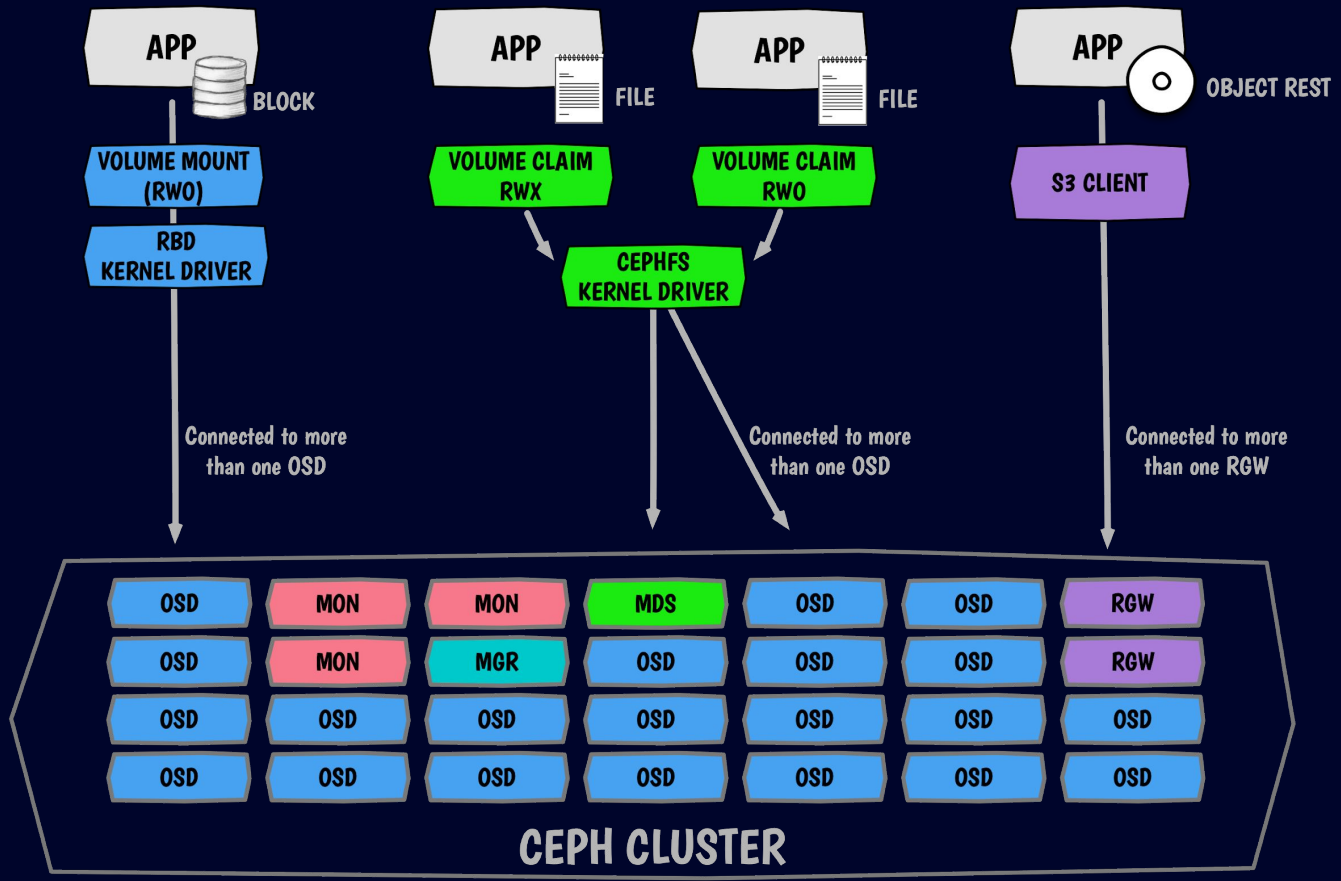
Layer 2: CSI Provisioning



ROOK

ceph

Layer 3: Ceph Data Path



ROOK

ceph

GETTING STARTED

Installing Ceph is easy!

- Create the authorization (RBAC) settings
  - `kubectl create -f common.yaml`
- Create the Operator
  - `kubectl create -f operator.yaml`
- Create the `CephCluster` CR
  - `kubectl create -f cluster.yaml`

```yaml
apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph
  namespace: rook-ceph
spec:
  cephVersion:
    image: ceph/ceph:v15.2.4
  dataDirHostPath: /var/lib/rook
  mon:
    count: 3
  storage:
    useAllNodes: true
    useAllDevices: true
```

ROOK

ceph

Application Storage

- Admin creates a storage class
- Create a PVC
- Create your application pod

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: csirbd-demo-pod
spec:
  containers:
  - name: web-server
    image: nginx
    volumeMounts:
      - name: mypvc
        mountPath: /var/lib/www/html
  volumes:
  - name: mypvc
    persistentVolumeClaim:
      claimName: rbd-pvc
      readOnly: false
```

ROOK

ceph

Storage Configuration

- Environments: Bare metal or Cloud
- Provision storage from a storage class (PV)
- Device management (non-PV):
  a. Use all available raw devices or partitions
  b. List all nodes and devices by name
  c. Ceph Drive Groups

Cluster Topology

- Failure domains: High availability and durability
  - Ceph Monitors should be spread across zones
  - OSD CRUSH hierarchy will be automatically populated based on node labels
  - Spread OSDs evenly with pod topology constraints
- Rook can be deployed on specific nodes if desired
  - Node affinity, taints/tolerations, etc

Ceph in a Cloud Environment

- Consistent Storage Platform wherever K8s is deployed
- Overcome shortcomings of the cloud provider's storage
  - Storage across AZs
  - Slow failover times (seconds instead of minutes)
  - Limitations of number of PVs per node (many more than ~30)
  - Perf characteristics of large volumes
- Ceph Monitors and OSDs run on PVCs
  - No need for direct access to local devices

KEY FEATURES

Upgrading is automated!

- To upgrade Rook, update the Operator version
  - Simply update the Operator version
  - Minor releases require steps as documented in the upgrade guide

```
image: rook/ceph:v1.4.2
```

- To upgrade Ceph, simply update the `CephCluster` CR version
  - Rook handles intricacies of Ceph version upgrades

```
image: ceph/ceph:v15.2.6
```

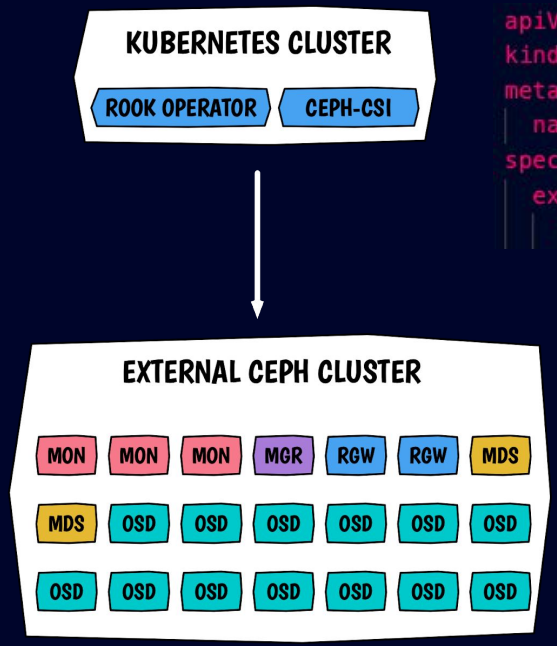
Ceph CSI Driver

- Ceph CSI 3.0 Driver is deployed by default with v1.4
  - Dynamic provisioning of RWO/RWX/ROX (RBD)
  - Dynamic provisioning of RWO/RWX/ROX (CephFS)
- Snapshots and clones are beta
  - Not backward compatible with alpha
- Flex driver is still available, but support is limited

External Cluster Connection

Connect to a Ceph cluster that you've configured separately from Kubernetes

- Inject the following in Kubernetes:
  - Monitors list
  - Keyring
  - Cluster FSID
- Create the cluster-external CR




```yaml
apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph-external
spec:
  external:
    enable: true
```

ROOK

ceph

Object Bucket Provisioning

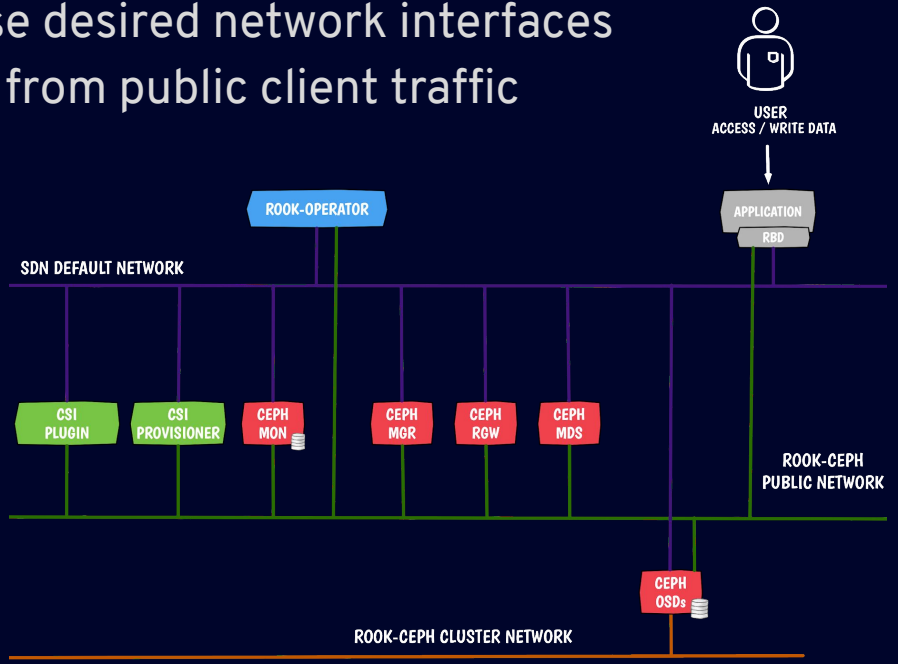
- Define a Storage Class for object storage
- Create an “object bucket claim”
  - The operator creates a bucket when requested
  - Similar pattern to a Persistent Volume Claim (PVC)

ROOK v1.4 FEATURES

Aug 2020

Multus Networking

- Multus is supported - not experimental anymore
- “Whereabouts” IPAM is preferred
- Increased security, only expose desired network interfaces
- Separate internal Ceph traffic from public client traffic
- Lack of Services support



ROOK

ceph

Object Multisite Replication (Experimental)

- Geographically replicate objects across Rook-Ceph clusters
  - Region
  - Datacenter
- New CRDs:
  - Realm
  - Zone group
  - Zone

ROOK

ceph

Admission Controller

- Validates the creation of Custom Resources
- Reject incorrect CR before the Operator reconciles
- Not enabled by default (yet)

Toolbox Job

- Execute Ceph commands in a Kubernetes Job
- Examples:
  - Periodically collect information in the cluster
  - Remove failed OSDs from the cluster
- No manual intervention

Improved external mode

- More stable
- Gather External cluster metrics and put them in Prometheus
- External `CephObjectStore`:
  - use external gateways and integrate them as Kubernetes Service

ROOK

ceph

And much more!

- Encryption for OSD on PVC
- Health checks and livenessprobe configuration
- All Rook CRDs have been converted to use the controller-runtime library
- Cluster cleanup during uninstall enhancements (sanitize drives)
- Ceph Drive Groups can be specified in the CephCluster CR

Thanks!

https://rook.io/

tnielsen@redhat.com
seb@redhat.com


KubeCon
CloudNativeCon
Europe 2020
Virtual
HELM
KEEP CLOUD NATIVE
CONNECTED

Storage: All Devices

- Use all available devices that Rook discovers on nodes in the cluster
- Filter with a node selector where the nodes have a label role=storage-node

```yaml
storage:
  useAllNodes: true
  useAllDevices: true
```

```yaml
storage:
  useAllNodes: true
  useAllDevices: true
placement:
  osd:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: role
            operator: In
            values:
            - storage-node
```

ROOK

Storage: Device Sets

1. Provision storage from a storage class
2. Native K8s solution: No need for direct access to hardware
3. OSDs can failover across nodes
4. Scenarios:
   a. Cloud environments
   b. Local PVs

```yaml
storage:
  storageClassDeviceSets:
  - name: set1
    count: 3
    portable: true
    volumeClaimTemplates:
      - metadata:
          name: data
        spec:
          resources:
            requests:
              storage: 100Gi
          storageClassName: <name>
          volumeMode: Block
          accessModes:
            - ReadWriteOnce
    placement:
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - rook-ceph-osd
            - rook-ceph-osd-prepare
```

ROOK

Storage: Ceph Drive Groups

- Use hdds for data and ssds for metadata

```yaml
driveGroups:
  - name: data_rotational
    spec:
      data_devices:
        rotational: 1
      db_devices:
        rotational: 0
```

- Use max of 6 devices between 10-50TB with separate db and wal devices

```yaml
driveGroups:
  - name: sizes_and_models
    spec:
      data_devices:
        limit: 6
        size: "10TB:50TB"
      db_devices:
        model: SSD-123-foo
      wal_devices:
        model: NVME-QQQQ-987
```

Storage: Named Nodes and Devices

- List all nodes and devices by name
- Scenarios:
  - Absolute control rather than relying on discovery

```
storage:
  useAllNodes: false
  useAllDevices: false
  nodes:
  - name: "172.17.4.201"
    devices:
    - name: "sdb"
    - name: "/dev/disk/by-id/ata-ST4000DM004-XXXX"
    - name: "nvme01"
      config:
        osdsPerDevice: "5"
  - name: "172.17.4.301"
    deviceFilter: "^sd."
```

ROOK

ceph