Rook Intro & Ceph Deep Dive

KubeCon | CloudNativeCon Europe 2021

Virtual

Rook maintainers:

Satoru Takeuchi, Cybozu, Inc.
Blaine Gardner, Red Hat
Travis Nielsen, Red Hat
Sébastien Han, Red Hat

April 2021

Kubernetes Storage Challenges

Kubernetes Storage Challenges

- Kubernetes is a platform to manage distributed apps
  - Traditionally stateless
- Reliance on external storage
  - Not portable
  - Deployment burden
  - Day 2 operations - who is managing the storage?
- Reliance on cloud provider managed services
  - Vendor lock-in

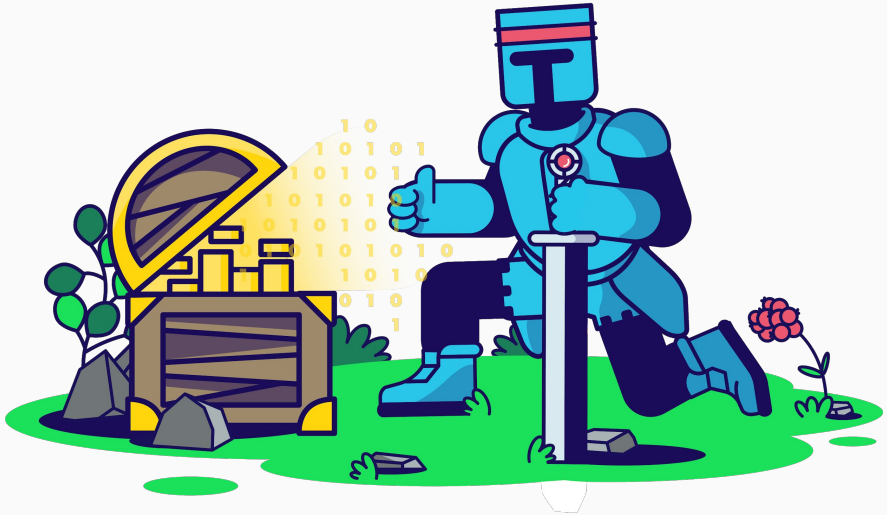
What is Rook?

What is Rook?

- Makes storage available inside your Kubernetes cluster
- Kubernetes Operators and Custom Resource Definitions
- Automated management
  - Deployment, configuration, upgrades
- Consume like any other K8s storage
  - Storage Classes, Persistent Volume Claims
- Open Source (Apache 2.0)

Storage Providers

- Stable
  - Ceph
- Alpha
  - Cassandra
  - NFS
  - YugabyteDB



🎉 v1.6 released

⭐ 8.5K+ Github Stars

💾 215M+ Downloads

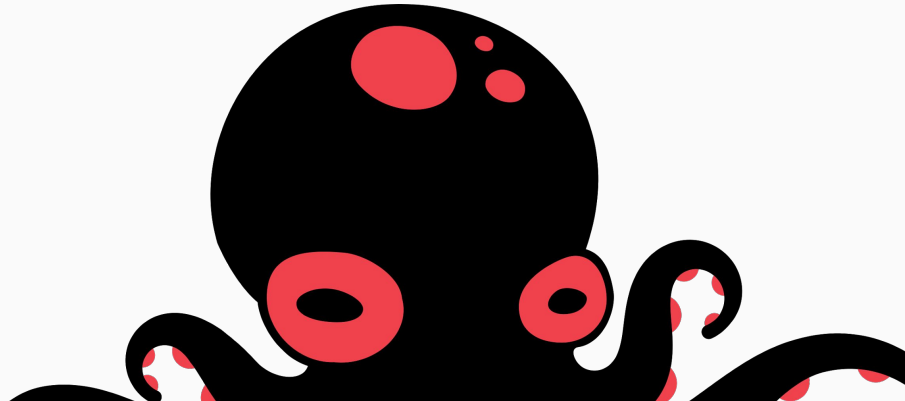
🤝 325+ Contributors

🎓 CNCF Graduated Project

Rook + Ceph

What is Ceph?

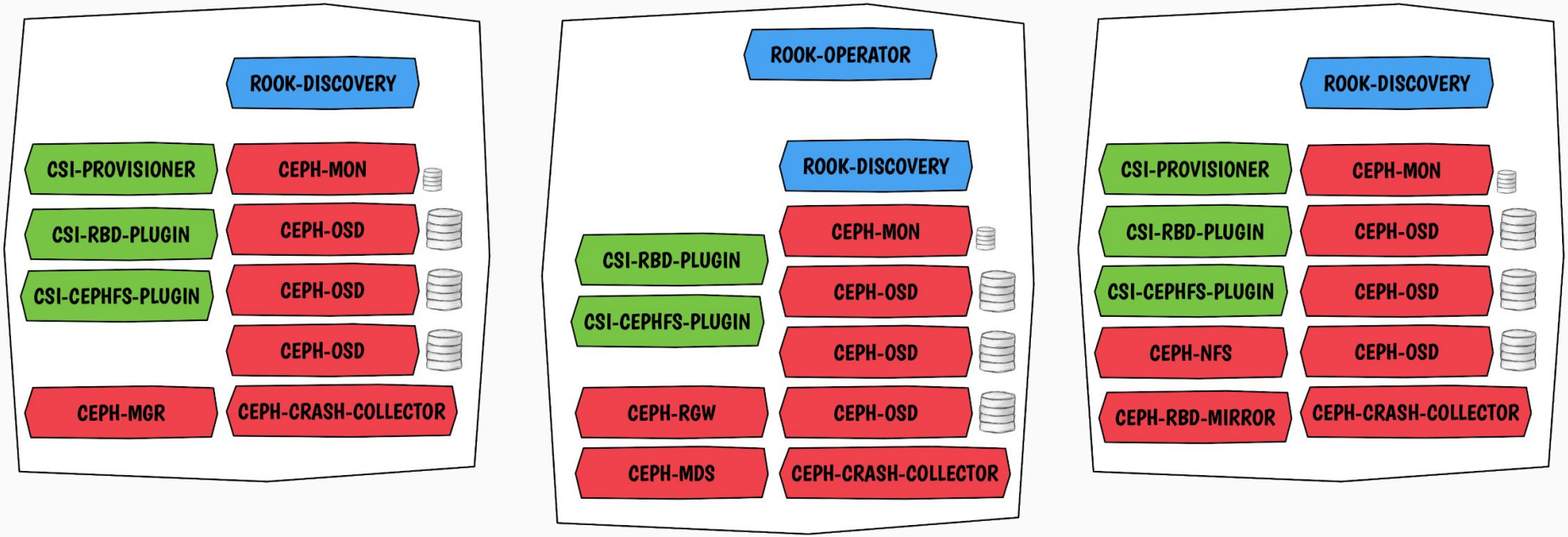
- Open Source
- Distributed Software-Defined Storage solution
  - Block
  - Shared File System
  - Object (S3 compliant)
- First release in July 2012
- https://ceph.io/



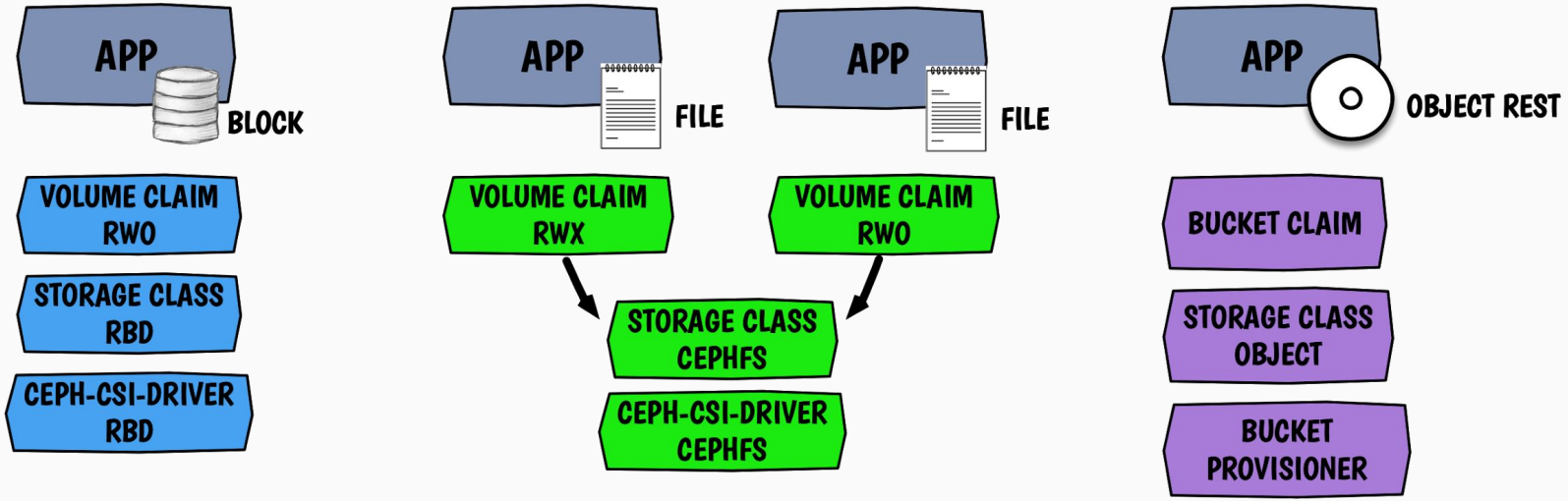
Architectural Layers

- Rook
  - Operator owns the **management** of Ceph
- Ceph-CSI
  - CSI driver dynamically **provisions** and **mounts** storage to user application Pods
- Ceph
  - **Data** layer

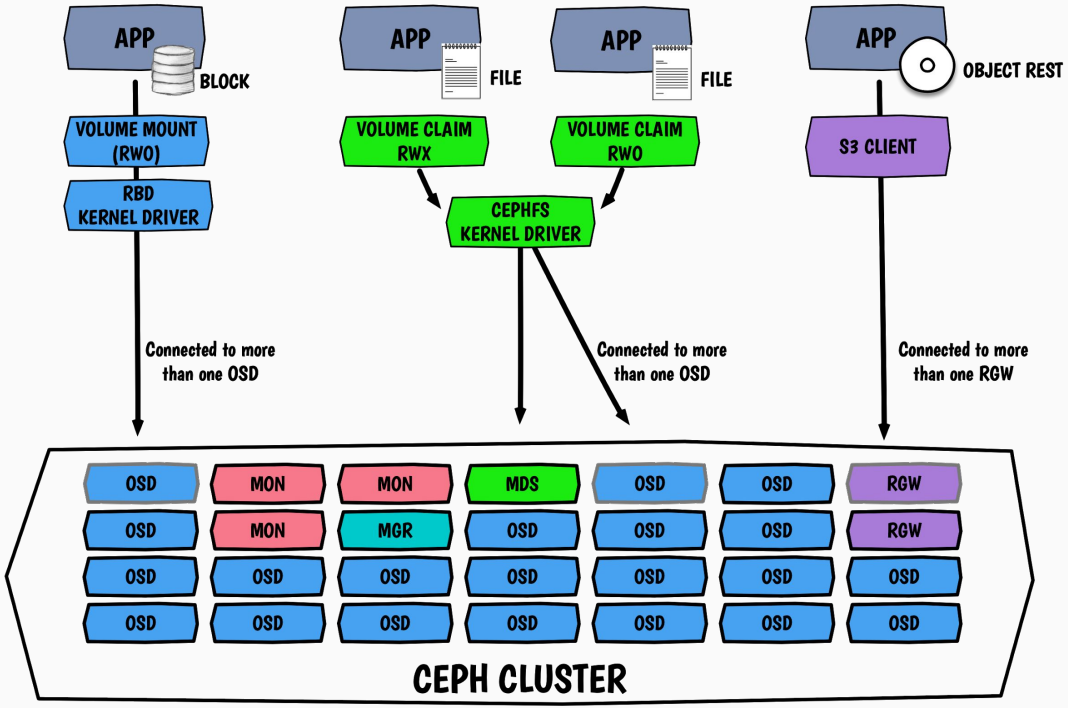
Layer 1: Rook Management



Layer 2: CSI Provisioning



Layer 3: Ceph Data Path



Key Features

Installing Ceph is simple!

- Create Custom Resource Definitions
  - `kubectl create -f crds.yaml`
- Create authorization (RBAC)
  - `kubectl create -f common.yaml`
- Create the Ceph Operator
  - `kubectl create -f operator.yaml`
- Create the CephCluster resource
  - `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
```

Environments

Bare metal

- Bring your own hardware
- Or shared hardware

Cloud providers

- Expand cloud provider storage with Rook capabilities

Rook in a Cloud Environment

- Ceph uses PVCs as underlying storage
  - No need for direct access to local devices
- Consistent storage platform wherever K8s is deployed
- Overcome shortcomings of the cloud provider's storage
  - Storage across availability zones (AZs)
  - Faster failover times (seconds instead of minutes)
  - Greater number of PVs per node (many more than ~30)
  - Use storage with better performance:cost ratio

Configurable for Cluster Topologies

- Customizable across/within cluster topologies
- High availability and durability
  - Spread Ceph daemons and data across failure domains
- Deployable on specific nodes if desired
  - Node affinity, taints/tolerations, etc.

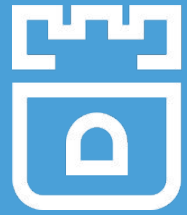
Updates are automated

- Ceph updates and even major upgrades are fully automated
  - Rook handles everything
- Rook patch up*dates* are fully automated
- Rook minor up*grades* sometimes require manual work
  - Take advantage of latest features
  - Occasional K8s/Ceph/CSI/Rook feature deprecations
  - Documented in Rook's upgrade guide

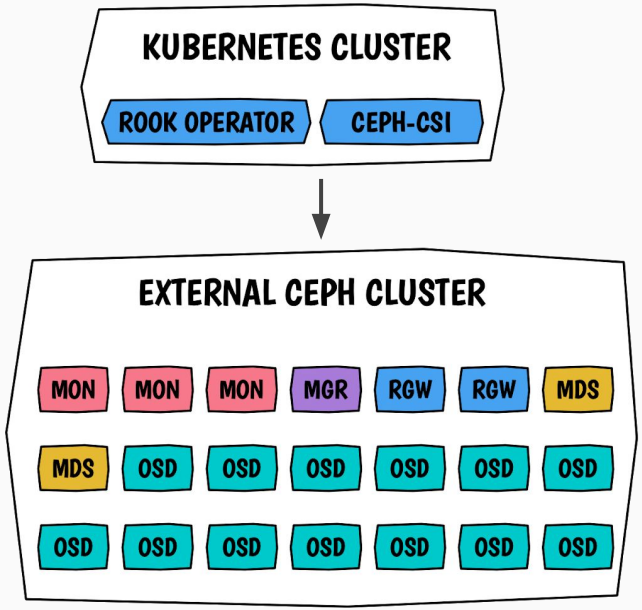
Ceph CSI Driver

- Dynamic provisioning of RWO/RWX/ROX modes for Block and Filesystem
- Volume expansion
- Snapshots and Clones (beta)
- FlexVolume driver is still available, but support is limited

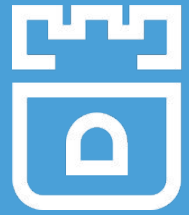
External Cluster Connection



- Connect to a Ceph cluster outside of the current K8s cluster
- Dynamically create Block/File/Object storage consumable by K8s applications



Object Storage Provisioning



- Define a Storage Class for object storage
- Create an Object Bucket Claim (OBC)
  - Similar pattern to a Persistent Volume Claim (PVC)
  - The operator creates a bucket when requested
  - Give access via K8s Secret
- Container Object Storage Interface (COSI)
  - Kubernetes Enhancement Proposal

Rook v1.6 Features

April 2021

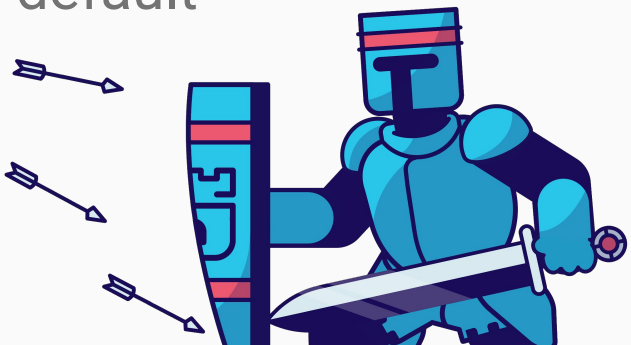
Ceph Pacific Support

Ceph versions supported:

- Nautilus (v14) – Support will be dropped in v1.7
- Octopus (v15)
- Pacific (v16) – New!

Notable new features

- Ceph Filesystem (CephFS)
  - Support multiple filesystems per Ceph cluster
  - Mirror a filesystem from one Ceph cluster to another
- Now support high-availability Ceph mgr daemons
- Pod Disruption Budgets are enabled by default



Ceph OSD Enhancements

Bulk OSD upgrades for large clusters

- Respects failure domains to retain high availability

LVM no longer used for most new OSDs

- Restore support for creating OSDs on partitions

Demo

Two types of clusters

- Host-based cluster
  - Specify hardware configurations directly in CephCluster CR
  - Persistent data is on hostPaths
- PVC-based cluster
  - Specify VolumeClaimTemplates for OSDs
  - Persistent data is on PVC

Host-based cluster

- Suitable for simple cluster
- CR gets complicated if...
  - Not all nodes are used
  - Various hardware configurations for each node

```
...
storage:
  useAllNodes: true
  useAllDevices: true
```

```
...
storage:
  nodes:
  - name: “foo”
    - devices:
      - name: “sdb”
      ...
```

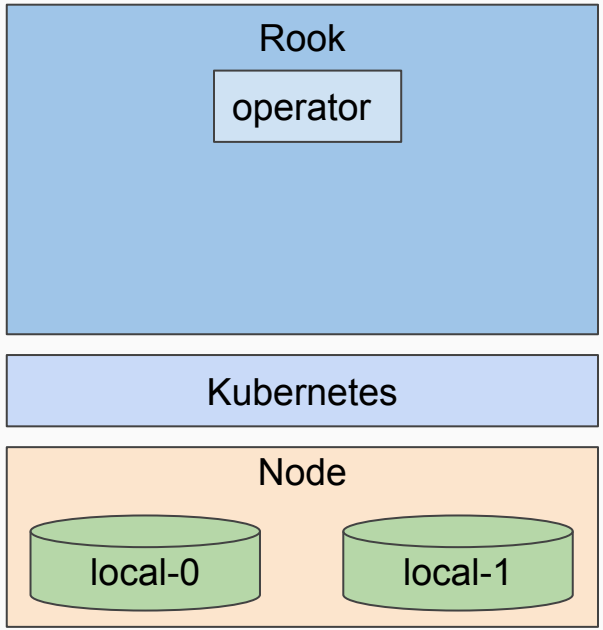
PVC-based cluster

- Free from describing hardware configurations
- Not intuitive

```
...
storage:
 storageClassDeviceSets:
  - name: set1
    count: 1
    volumeClaimTemplates:
    - spec:
        resources:
          requests:
            storage: 5Gi
        storageClassName: foo
        ...
```

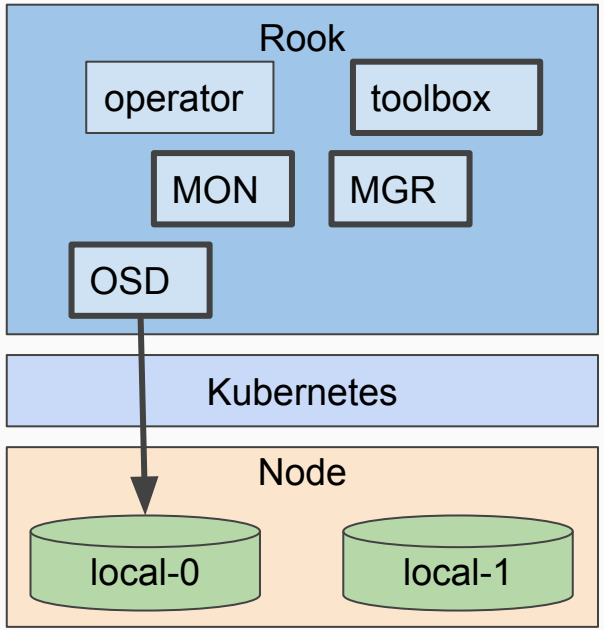
Create a PVC-based cluster (1/3)

- Environment
  - 1 node Kubernetes cluster (v1.20.5)
  - 2 local PVs on this node
  - A Rook operator (v1.5.9)
- Steps
  1. Create a simple cluster
     - Apply sample-cluster.yaml & toolbox.yaml
     - https://github.com/satoru-takeuchi/kce21-sample
  2. Expand this cluster
     - Increase `count` field in CR



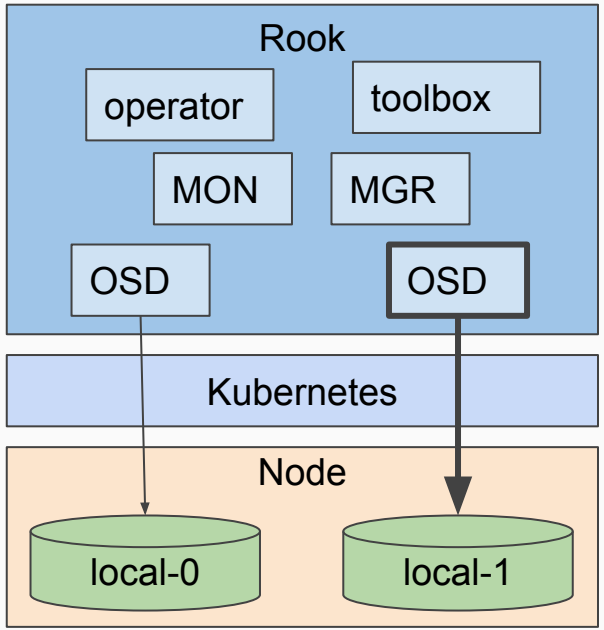
Create a PVC-based cluster (2/3)

- Environment
  - 1 node Kubernetes cluster (v1.20.5)
  - 2 local PVs on this node
  - A Rook operator (v1.5.9)
- Steps
  1. **Create a simple cluster**
     - **Apply sample-cluster.yaml & toolbox.yaml**
     - **https://github.com/satoru-takeuchi/kce21-sample**
  2. Expand this cluster
     - Increase `count` field in CR



Create a PVC-based cluster (3/3)

- Environment
  - 1 node Kubernetes cluster (v1.20.5)
  - 2 local PVs on this node
  - A Rook operator (v1.5.9)
- Steps
  1. Create a simple cluster
     - Apply sample-cluster.yaml & toolbox.yaml
     - https://github.com/satoru-takeuchi/kce21-sample
  2. **Expand this cluster**
     - **Increase `count` field in CR**



Advanced configurations

- Create PVs for OSDs on-demand
  - CSI drivers with dynamic volume provisioning
- Even OSD spreading
  - TopologySpreadConstraints feature in Kubernetes

- 📝 Production-grade Deployment of PVC-based Rook/Ceph Cluster
  - https://blog.kintone.io/entry/2020/09/18/175030

Thank you!

Website	https://rook.io/
Documentation	https://rook.io/docs/rook/v1.6/
Slack	https://rook-io.slack.com/
Contributions	https://github.com/rook/rook
Twitter	@rook_io
Community Meeting	https://github.com/rook/rook#community-meeting


KubeCon
CloudNativeCon
Europe 2021
Virtual
Forward Together »