
KubeCon
CloudNativeCon
North America 2021
RESILIENCE
REALIZED
Rook Intro &
Ceph Deep Dive
Rook maintainers:
Satoru Takeuchi, Cybozu, Inc.
Blaine Gardner, Red Hat
Travis Nielsen, Red Hat
Sébastien Han, Red Hat

Goals

- Kubernetes Storage Challenges
- What is Rook?
- Rook+Ceph background
- Rook+Ceph key features
- Rook v1.7 new features
- Demo
- Q&A

Kubernetes Storage Challenges

Kubernetes Storage Challenges

- Kubernetes is a platform to manage distributed apps
  - Ideally 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

This is what Rook hopes to solve

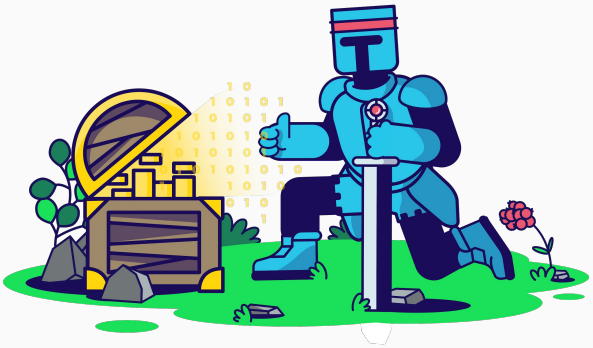
What is Rook?

What is Rook?

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

Storage Providers

- Stable
  - Ceph
- Alpha
  - Cassandra
  - NFS
- Storage providers now release independently!

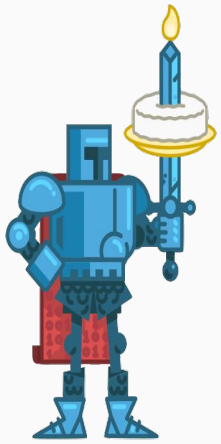


3 active storage providers

Ceph stable - thus, upcoming deep dive

Happy 5th Birthday!

- Rook went public with v0.1 in November 2016
  - KubeCon Seattle
- 111 releases in 5 years
- Thanks for all the community support!



Rook+Ceph background

- What is Ceph?
- Architectural layers
  - Rook : Management
  - CSI : Storage provisioning
  - Ceph : Data layer

What is Ceph?

- Open Source
- Scalable, fault-tolerant storage service
  - Block
  - Shared File System
  - Object (S3 compliant)
- Favors consistency
- First release in July 2012
- https://ceph.io/

"ceph"alopod

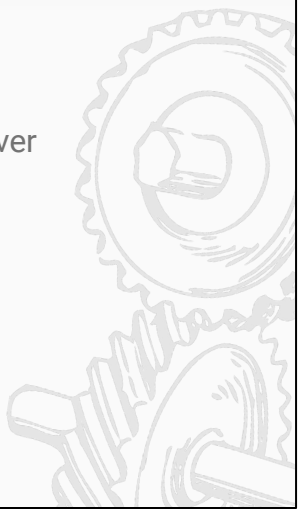
Cephalopod-themed storage service

A lot of words here … TL;DR:

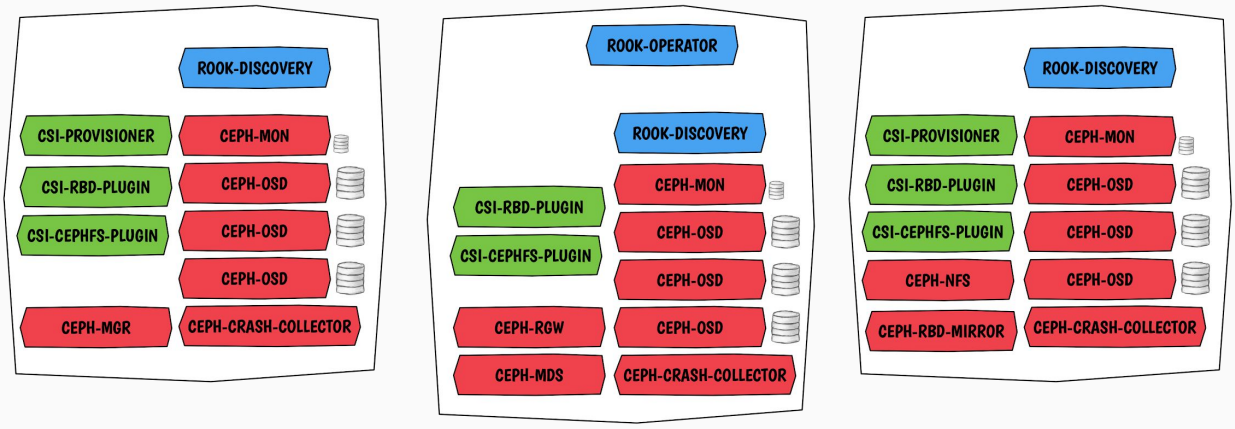
- keeps your data safe through scale
- Provides all 3 most common types of storage: block, shared file, and s3 object storage

Architectural Layers

- Rook
  - Operator owns the **deployment** and **management** of Ceph and Ceph CSI (Container Storage Interface) driver
- Ceph-CSI
  - CSI driver dynamically **provisions** and **mounts** storage to user application Pods
- Ceph
  - **Data** layer



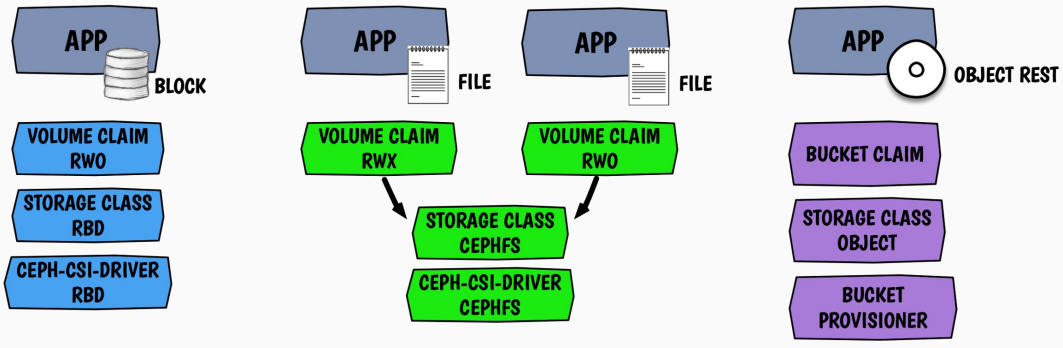
Rook : Management



Rook Operator (in blue) manages all of this other stuff

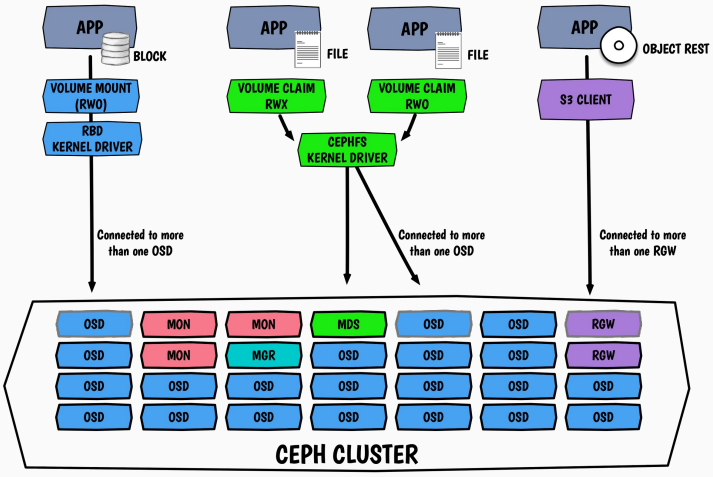
- Ceph (in red)
- CSI (in green)
- Even the Rook discovery daemons (also in blue)

CSI : Storage provisioning



CSI driver is the thing that provisions storage in Ceph
Then connects the storage to application pods

Ceph : Data path



A bit of a continuation of the CSI provisioning from earlier

Key takeaway: neither Rook nor Ceph-CSI are in the data path between application and storage

Rook+Ceph Key Features

- Installation is simple
- Ceph CSI driver features
- Support for bare metal and cloud environments
- Configure for any cluster topology
- Updates are automated
- Connect to an external Ceph cluster
- Provision object storage buckets

Highlight features that are noteworthy and most commonly useful

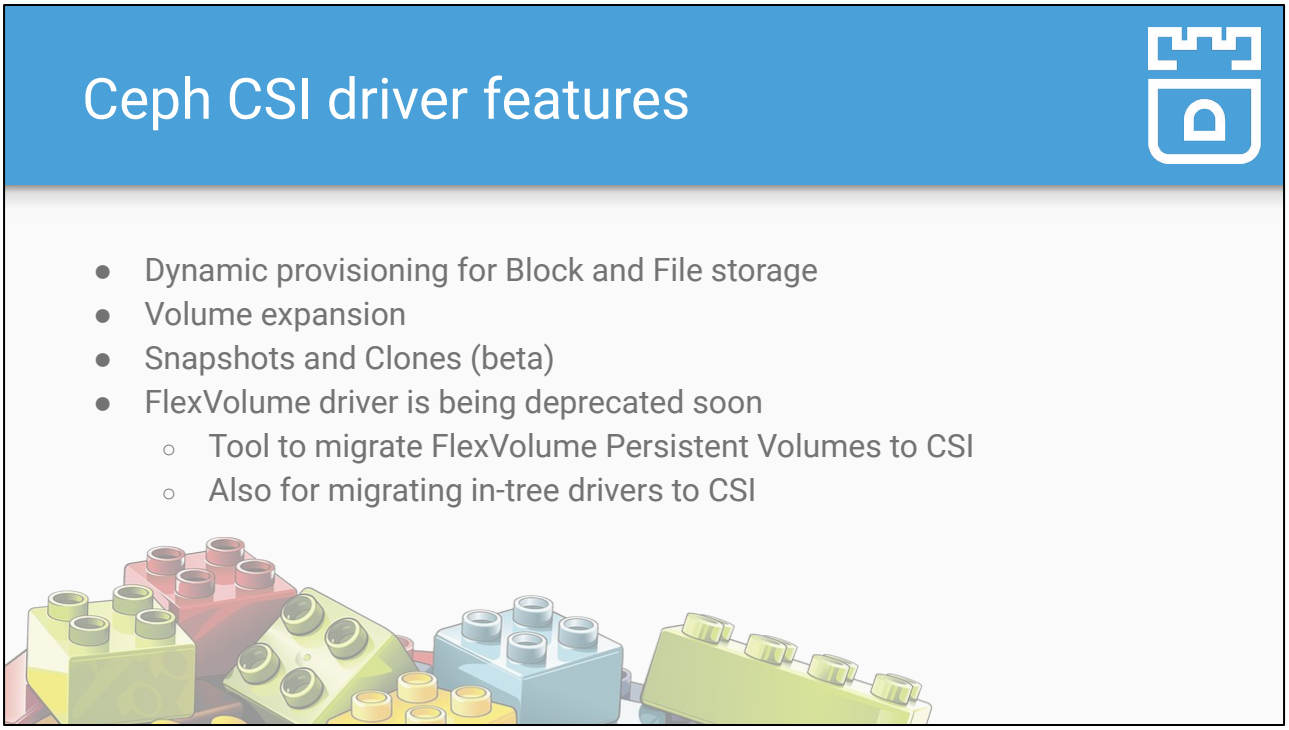
Installing Ceph is simple!

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

```yaml
apiVersion: ceph.rook.io/v1
kind: CephCluster
metadata:
  name: rook-ceph
  namespace: rook-ceph
spec:
  cephVersion:
    image: quay.io/ceph/ceph:v16.2.5
  mon:
    count: 3
  storage:
    useAllNodes: true
    useAllDevices: true
```

Minimal Ceph cluster spec to the right is 13 lines

Ceph CSI driver features



Ceph CSI driver is critical for allowing user applications to actually use the Ceph storage Rook creates

Environments

Bare metal

- Bring your own hardware
- Or shared hardware

Cloud providers

- Expand cloud provider storage with Rook capabilities

Rook in a Cloud Environment

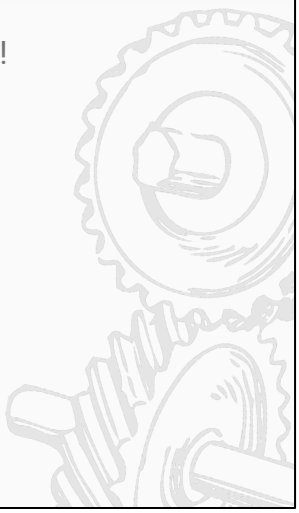
- Overcome shortcomings of the cloud provider's storage
  - Storage across availability zones (AZs)
  - Faster failover times (seconds instead of minutes)
  - Greater number of PVs per node (many more than ~30)
  - Use storage with better performance:cost ratio
- Consistent storage platform wherever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

Consistent storage platform - good for multi-cloud

Configure for any cluster topology

- 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



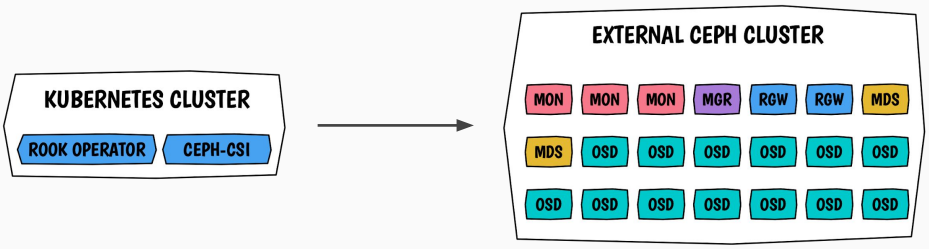
- Ceph updates and even major upgrades are fully automated!
  - Rook handles *everything*
- Rook patch up*dates* are fully automated
- Rook minor up*grades* sometimes require manual work
  - Take advantage of latest features
  - Occasional K8s/Ceph/CSI/Rook feature deprecations
  - https://rook.io/docs/rook/master/ceph-upgrade.html

Deprecation e.g., FlexVolume → CSI

Link: ceph upgrade guide

Connect to an external Ceph cluster

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



Ceph has been around for a long time
Many users may already have a Ceph cluster
Rook allows connecting K8s cluster to pre-existing Ceph cluster easily

Provision object storage buckets

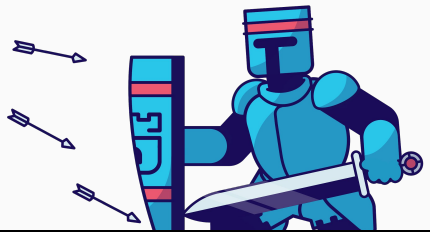
- Define a Storage Class for Ceph object storage
- Create an Object Bucket Claim (OBC)
  - Similar pattern to a Persistent Volume Claim (PVC)
  - Rook operator creates a bucket when requested
  - Give access via K8s Secret
- Container Object Storage Interface (COSI)
  - Kubernetes Enhancement Proposal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

Rook v1.7 features

Aug 2021

Notable updates

- Stretch cluster is stable
- Protect user data when deleting Ceph cluster
  - Don’t allow deleting a Ceph cluster if other resources exist
- Full support for mirroring a filesystem from one Ceph cluster to another
  - Newer Ceph feature still undergoing testing



(More on stretch cluster later)

Stretch Cluster

- Many users wish to run Ceph in multiple failure domains
- Most commonly either two or three zones

  🤔 With two zones, how do you maintain quorum during a zone failure?

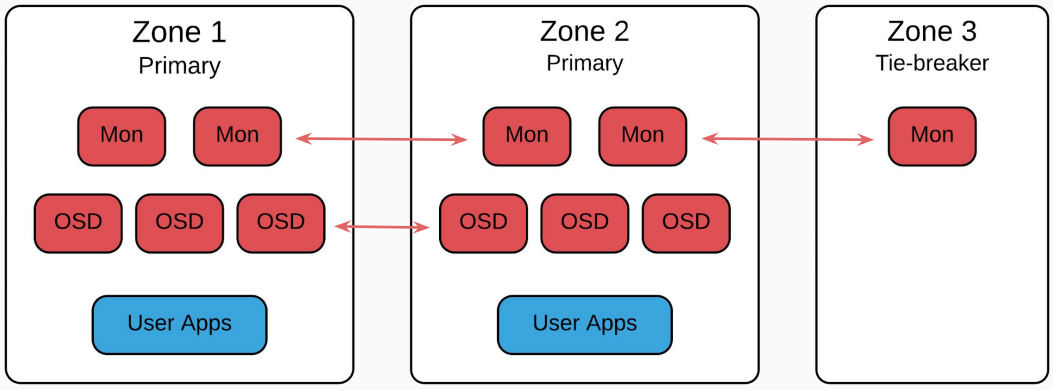
  A third zone acting as a minimal tie breaker

- https://docs.ceph.com/en/latest/rados/operations/stretch-mode/

Quorum is by definition **more** than 50%. If that's the case…
With two zones, how do you maintain quorum if a whole zone fails?

Link: read in-depth about the topic from Ceph's documentation

Stretch Cluster

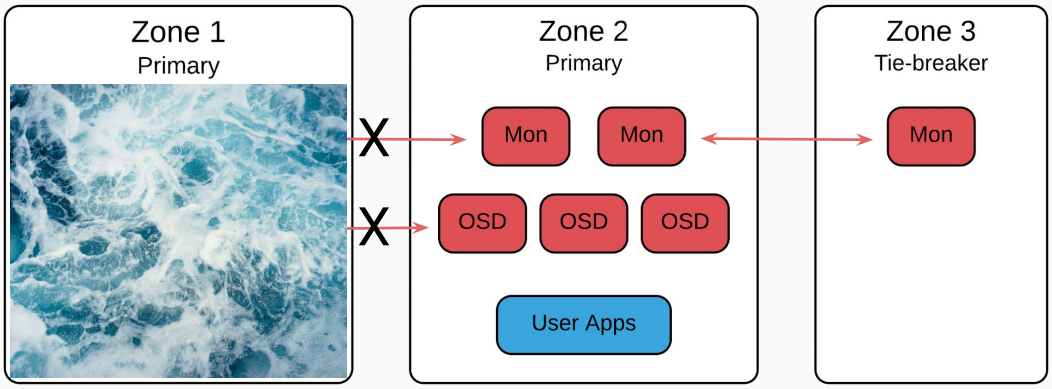


Two primary zones where all user applications and Ceph storage lives

A third zone that is as minimal as possible, serving only as a tie-breaker monitor
Keeps >50% quorum during zone failure

This is what it looks like during normal operation …
… but what we care about is failure

Stretch Cluster



Imagine one of your primary zones goes offline
… maybe because you live in Houson, TX, and there was a hurricane, and now your data center is under 3 feet of water (true story)

With the tie-breaker zone, there is still a majority quorum of 3/5 mons

Ceph has protected all your data, and it's still available

Kubernetes will schedule any apps from zone 1 into zone 2 to keep running

Demo

Software versions

- Kubernetes v1.22.4
- Rook v1.7.2

Two types of Rook/Ceph clusters

- Host-based cluster
- PVC-based cluster

Host-based cluster

- Suitable for simple cluster
- CephCluster CR gets complicated if...
  - Not all nodes/devices are used

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

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

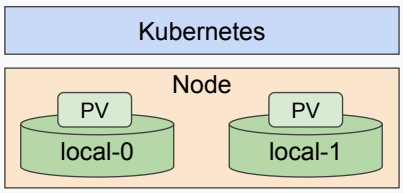
PVC-based cluster

- Free from describing hardware configurations
- Easy to expand
  - Just increase the `count` field!

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

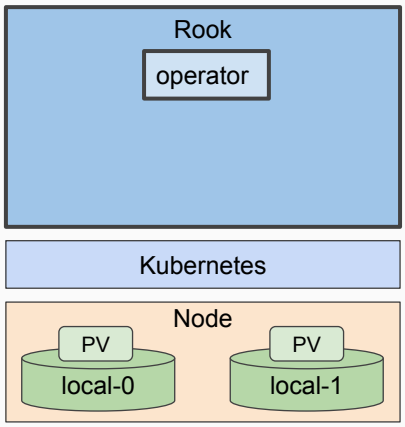
Create a PVC-based cluster (1/3)

- Environment
  - 1 node Kubernetes cluster
  - 2 local empty block devices
- Steps
  1. Create a Rook operator
  2. Create a Rook/Ceph cluster
  3. Expand this cluster
- https://github.com/satoru-takeuchi/rook-demos



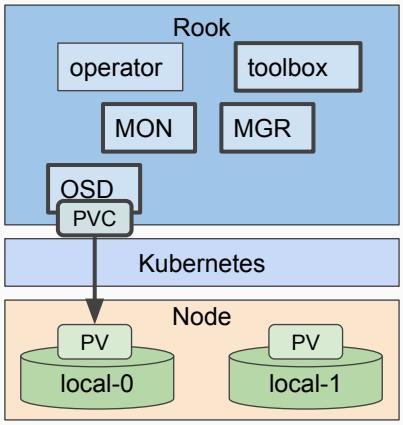
Create a PVC-based cluster (1/3)

- Environment
  - 1 node Kubernetes cluster
  - 2 local empty block devices
- Steps
  1. **Create a Rook operator**
  2. Create a Rook/Ceph cluster
  3. Expand this cluster



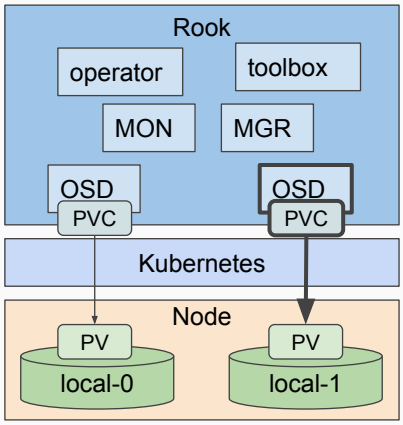
Create a PVC-based cluster (2/3)

- Environment
  - 1 node Kubernetes cluster
  - 2 local empty block devices
- Steps
  1. Create a Rook operator
  2. **Create a Rook/Ceph cluster**
  3. Expand this cluster



Create a PVC-based cluster (3/3)

- Environment
  - 1 node Kubernetes cluster
  - 2 local empty block devices
- Steps
  1. Create a Rook operator
  2. Create a Rook/Ceph cluster
  3. **Expand this cluster**



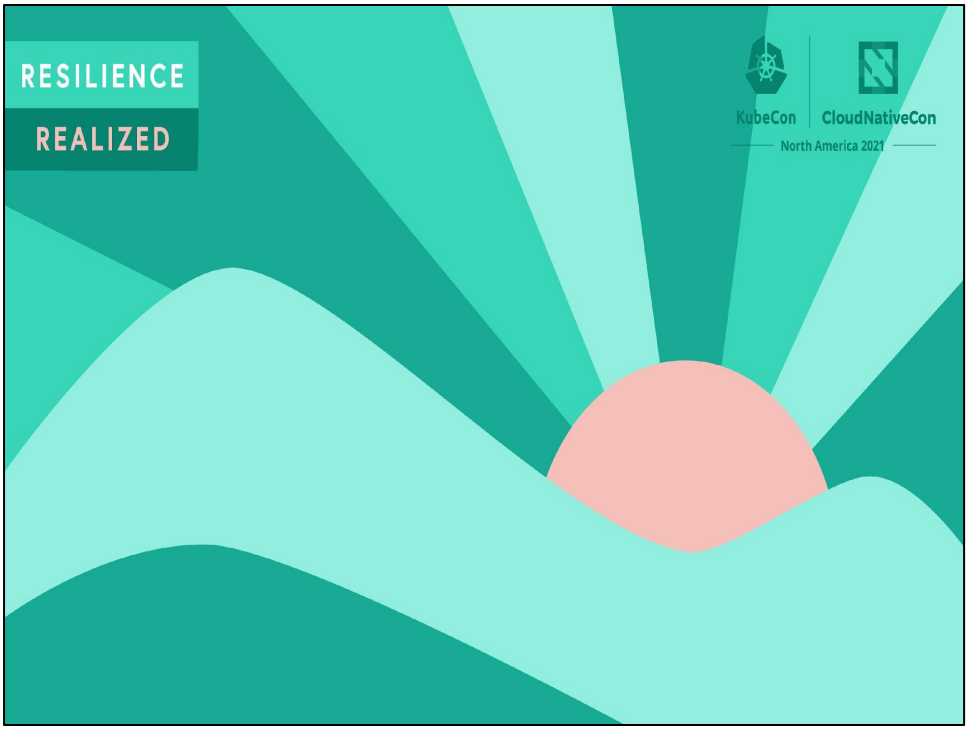
Advanced configurations

- Create PVs for OSDs on-demand
  - CSI drivers with dynamic volume provisioning
- Even OSD spreading among all nodes
  - 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


RESILIENCE
REALIZED
KubeCon
CloudNativeCon
North America 2021