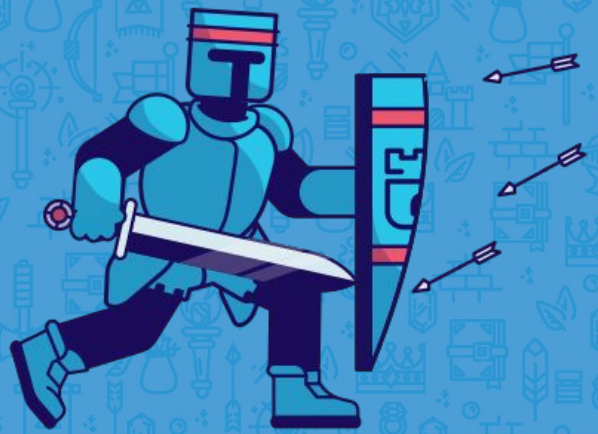
Rook Project Intro

Alexander Trost, Cloudical
Jared Watts, Upbound
Rook Maintainers

https://rook.io/
https://github.com/rook/rook



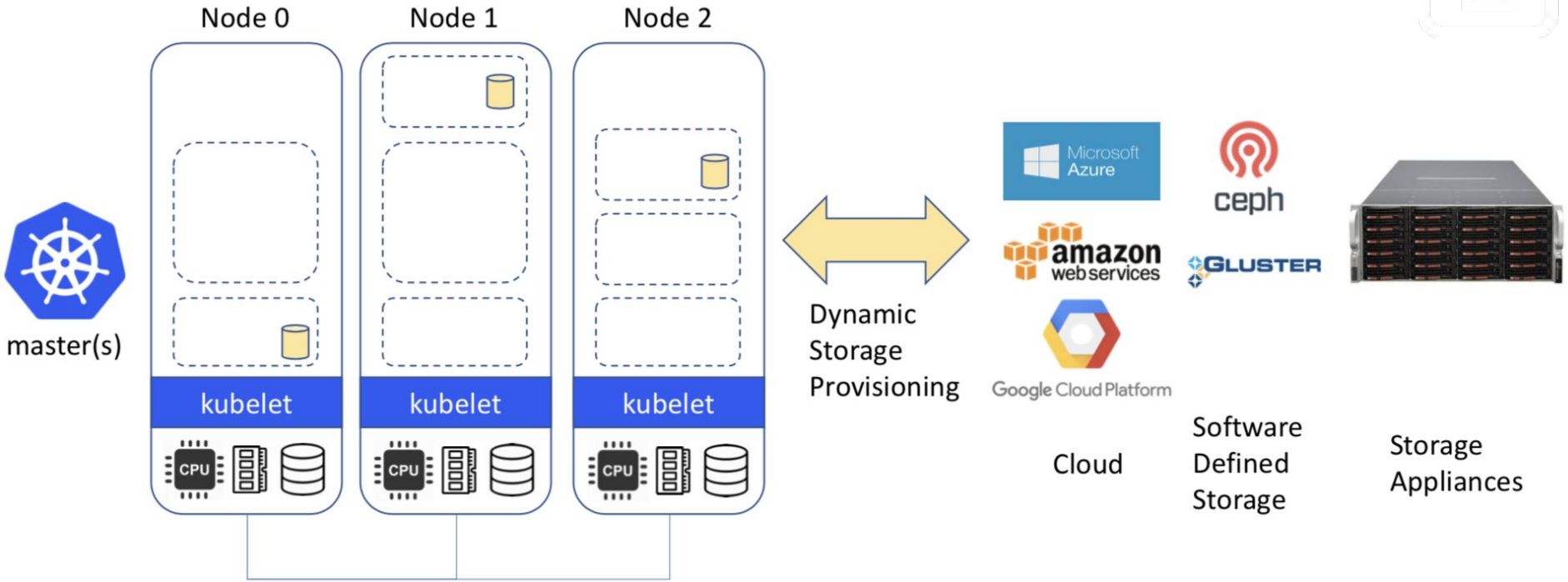
Agenda

- Storage Challenges in Kubernetes
- What is Rook and what does it solve?
- Rook Architecture
- Demo
- How to Get Involved
  - Deep-Dive Rook session on Thurs!
    https://sched.co/Zey5

Storage Challenges in Kubernetes

Storage Challenges

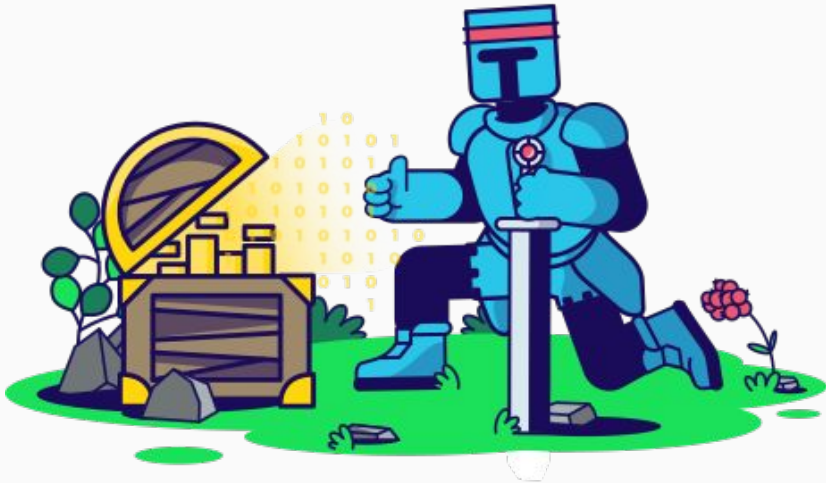
- Reliance on external storage
  - Not portable
  - Requires these services to be accessible
  - Deployment burden
- Reliance on cloud provider managed services
  - Vendor lock-in
- Day 2 operations - who is managing the storage?



Volume plugins allow external storage solutions to provide storage to your apps

What is Rook?

- Storage Operators for Kubernetes
- Automate
  - Deployment
  - Bootstrapping
  - Configuration
  - Upgrading
- Provision
  - Consume storage by PVCs



What is Rook?

- Framework for many storage providers and solutions
- Open Source (Apache 2.0)
- Cloud-Native Computing Foundation (CNCF)
  - Incubation project
  - TOC is now voting on Graduation!

Rook Community is Amazing!

v1.4 released

7.4K+ Github Stars

160M+ Downloads

275+ Contributors

CNCF Graduation vote

Operator Pattern

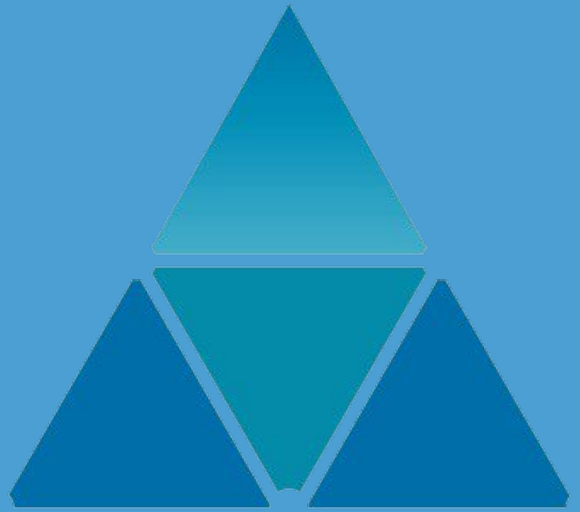
- Codifies domain expertise to deploy and manage an application
  - Automates actions a human would normally do
- Control loop that reconciles user's desired state and the actual system state
  - **Observe** - discover current actual state of cluster
  - **Analyze** - determine differences from desired state
  - **Act** - perform operations to drive actual towards desired

Rook Framework for Storage Solutions

- Rook is more than just a collection of Operators and CRDs
- **Framework** for storage providers to integrate their solutions into cloud-native environments
  - Storage resource normalization
  - Operator patterns & plumbing
  - Common policies, specs, logic
  - Testing effort
- Ceph, CockroachDB, NFS, Cassandra, EdgeFS, YugabyteDB...

Ceph

Highly scalable distributed storage solution for block, object and file system storage.



EdgeFS

Multi-cloud era distributed storage system for geo-transparent data access.

Cassandra

CockroachDB

NFS

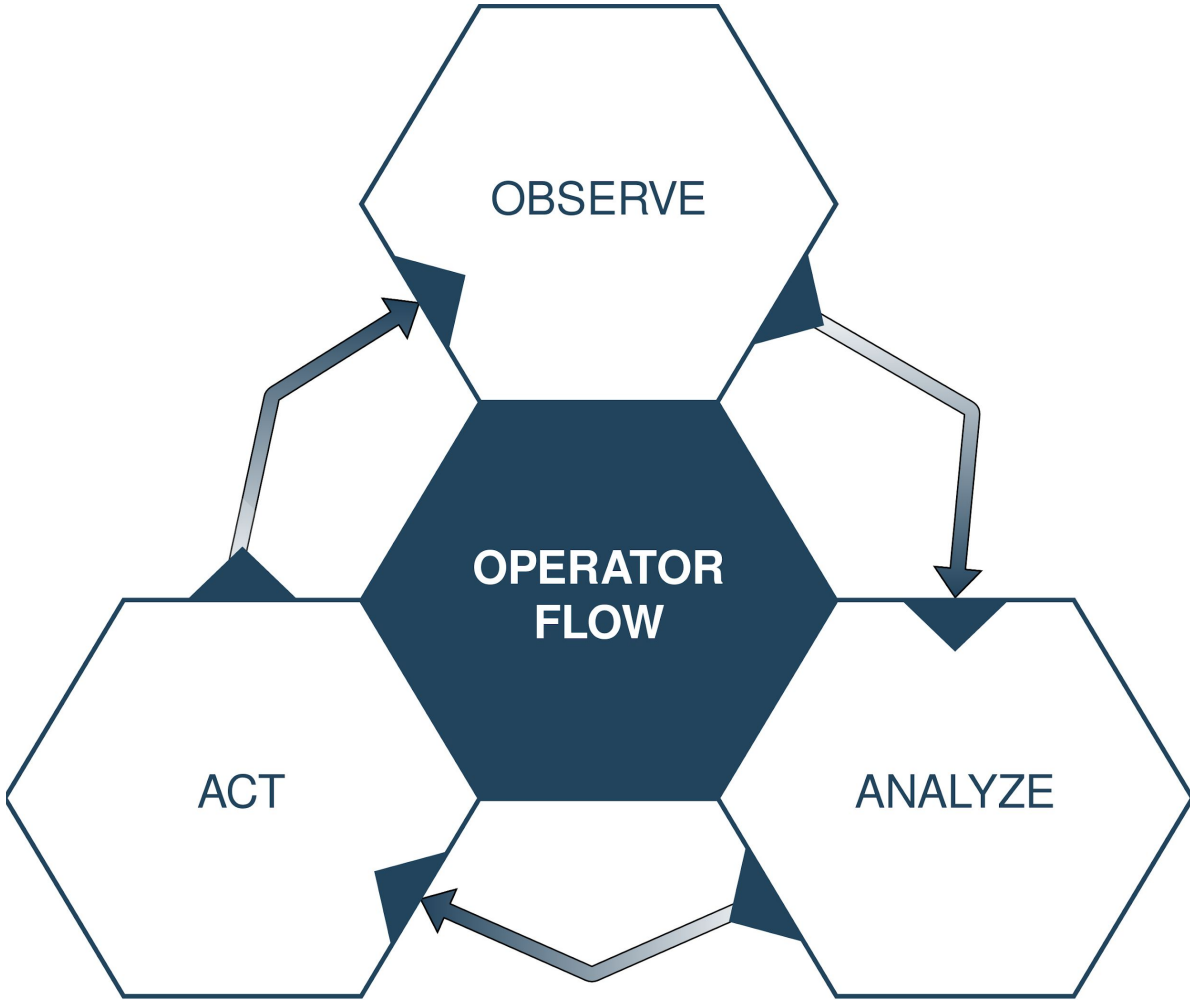
YugaByte

NFS
Network File System

The Goal is Persistence for Applications!

Kubernetes Native Integration

Image by Julius Silver from Pixabay


OBSERVE
OPERATOR
FLOW
ACT
ANALYZE

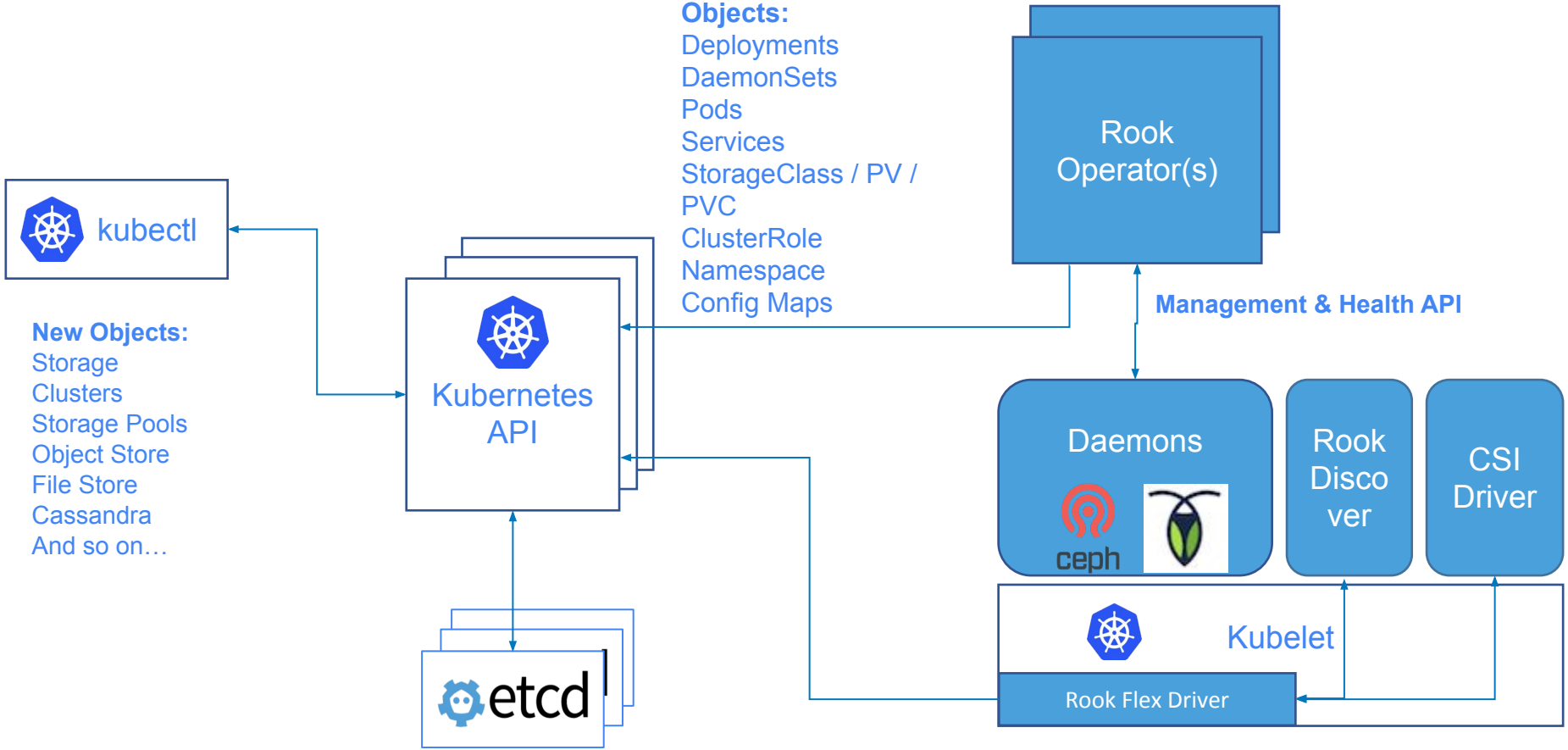

Objects:
Deployments
DaemonSets
Pods
Services
StorageClass / PV / PVC
ClusterRole
Namespace
Config Maps
Rook Operator(s)
kubectl
Management & Health API
New Objects:
Storage
Clusters
Storage Pools
Object Store
File Store
Cassandra
And so on…
Kubernetes API
Daemons
ceph
Rook Discover
CSI Driver
Kubelet
Rook Flex Driver
etcd

Image by Julius Silver from Pixabay

Rook Operators

- Implements the **Operator Pattern** for storage solutions
- Defines *desired state* for the storage resource
  - Storage Cluster, Pool, Object Store, etc.
- The Operator runs reconciliation loops
  - Watches for changes in desired state
  - Watches for changes in the cluster
  - Applies changes to the cluster to make it match desired

Rook Operators

- The Operators leverages the full power of K8S
  - Services, ReplicaSets, DaemonSets, Secrets, …
- Manage storage systems at scale
  - Stateful upgrades
  - Health and monitoring tasks
- Not on the data path – can be offline for minutes

Configuration through CustomResourceDefinitions

Custom Resource Definitions (CRDs)

- Teaches Kubernetes about new first-class objects
- Custom Resource Definition (CRDs) are arbitrary types that extend the Kubernetes API
  - look just like any other built-in object (e.g. Pod)
  - Enabled native `kubectl` experience
- A means for user to describe their desired state

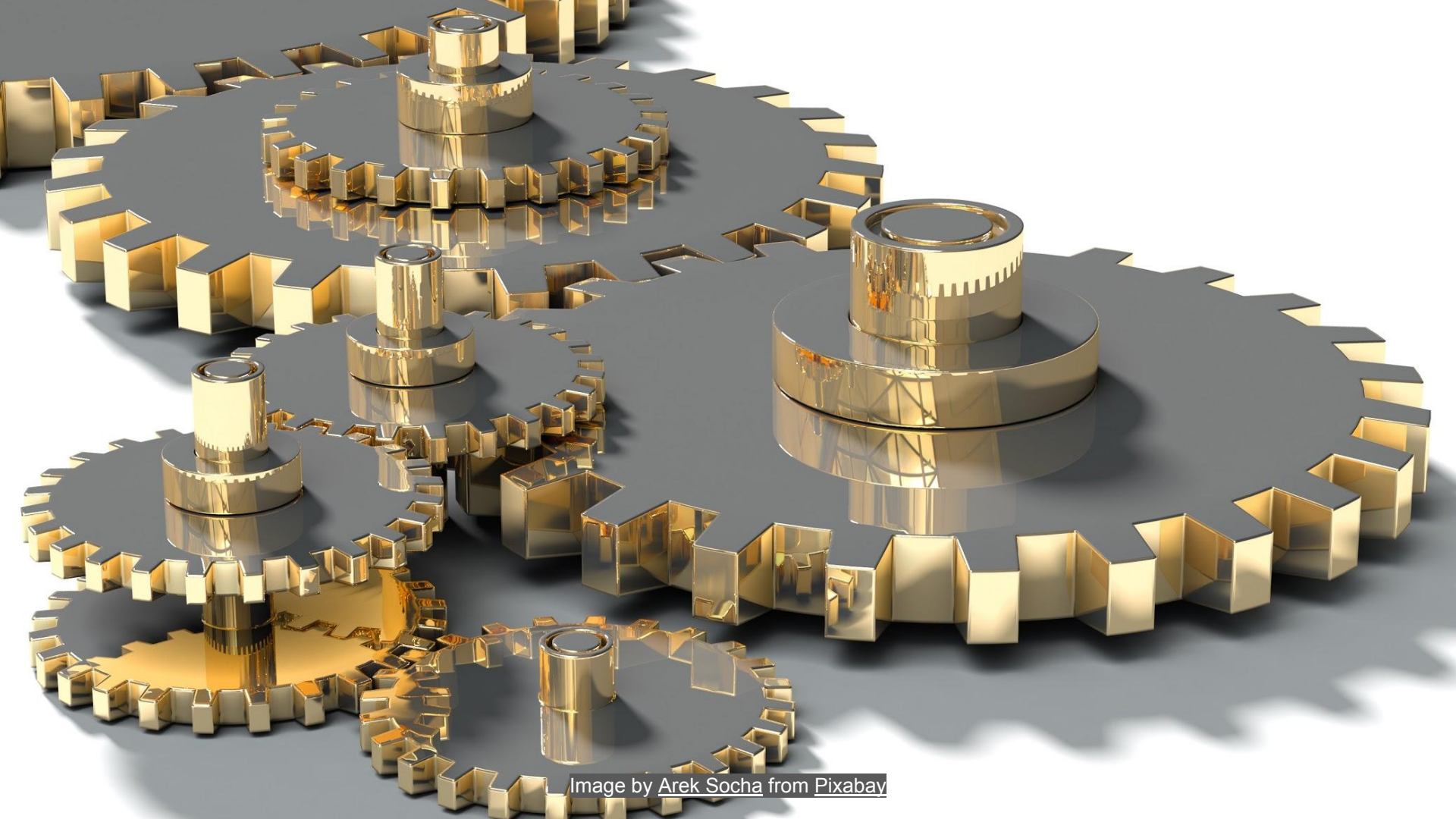


Image by Arek Socha from Pixabay

kind: CephCluster

```
storage:
  useAllNodes: true
  useAllDevices: true
  deviceFilter:
  location:
    config: [ ... ]
```

```
config:
  storeType: "bluestore"
  metadataDevice: "md0"
  databaseSizeMB: "1024"
  journalsizeMB: "1024"
  osdsPerDevice: "1"
  encryptedDevice: "true"
```

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

```yaml
apiVersion: ceph.rook.io/v1
kind: CephBlockPool
metadata:
  name: replicapool
  namespace: rook-ceph
spec:
  failureDomain: host
  replicated:
    size: 3
    requireSafeReplicaSize: true
```

Consuming the Storage

```
1 apiVersion: storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4     name: rook-ceph-block
5 provisioner: rook-ceph.rbd.csi.ceph.com
6 parameters:
7   clusterID: rook-ceph
8   pool: replicapool
9 [ ... ]
```

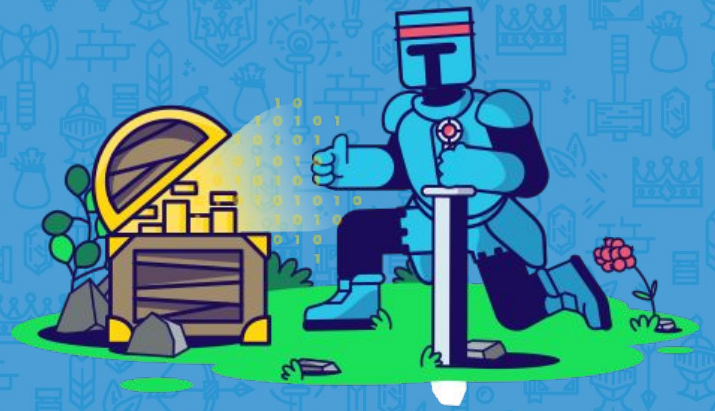
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: my-cool-app
spec:
  storageClassName: rook-ceph-block
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
   name: my-cool-app
spec:
  [ ... ]
  template:
      [ ... ]
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: my-cool-app
```

Demo

Deploying a Ceph cluster with a Stateful Application

How to get involved?



Documentation - https://rook.io/
Slack - https://rook-io.slack.com/ #conferences
Contribute to Rook - https://github.com/rook/rook
Twitter - @rook_io
Community Meetings

Thank you!

https://github.com/rook/rook

https://rook.io/