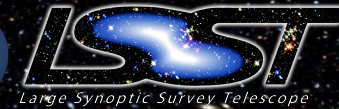




Google
Summer of Code



Embed an xrootd expert inside your Kubernetes cluster

Credits:

Andy Hanushevsky

Xrootd expert

SLAC

Yvan Calas

Large scale storage expert

CC-IN2P3

Shivansh Saini

B. Tech. Computer Science

Google summer of code student

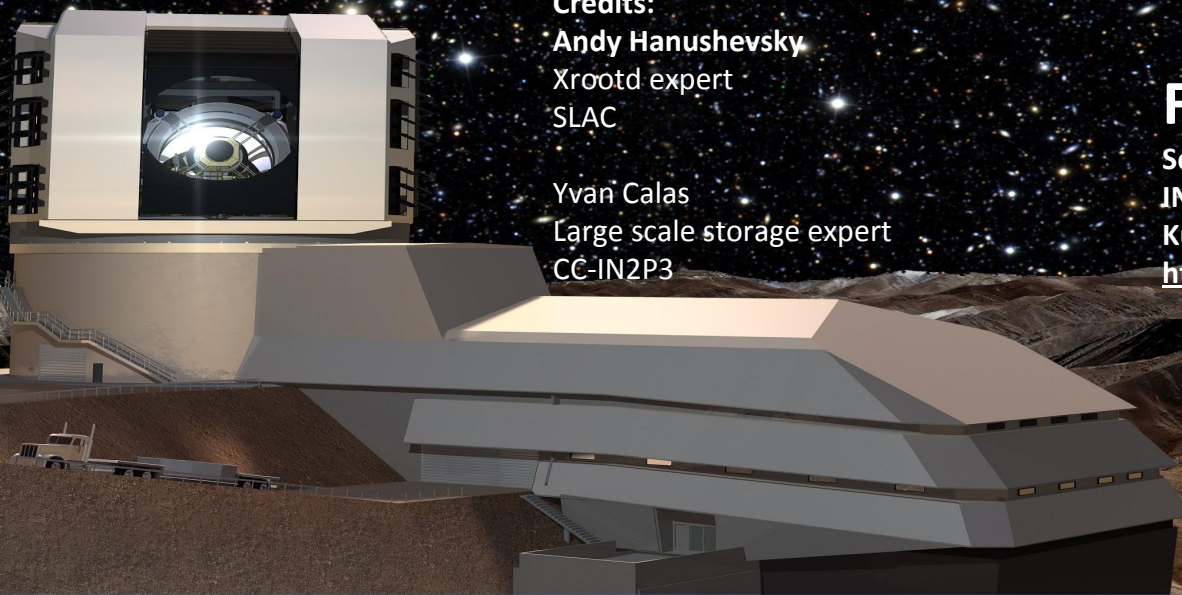
Fabrice Jammes

Scalable Data Systems Expert

IN2P3/LSST Corporation

Kubernetes expert and trainer

<https://k8s-school.fr>



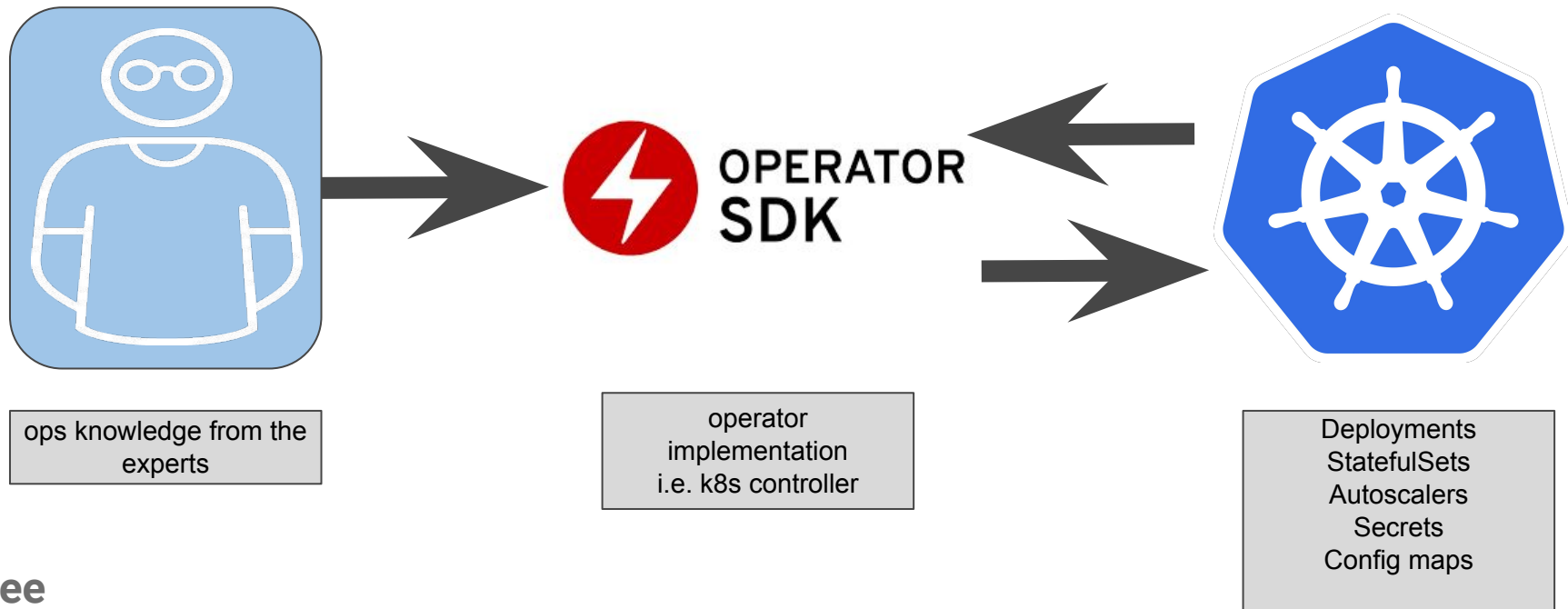


Agenda

- 1 What are Kubernetes operators?
- 2 Qserv operator
- 3 Xrootd operator
- 4 Demo

What is an Operator?

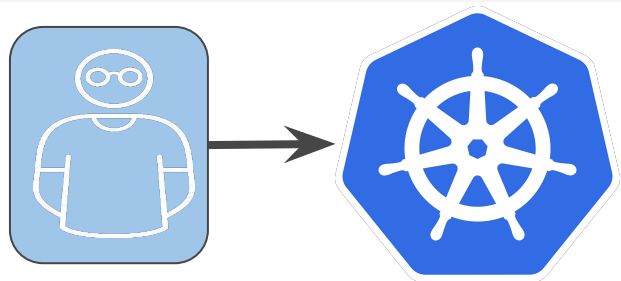
Operators embed ops knowledge from the experts



See

- <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>
- <https://cloud.google.com/blog/products/containers-kubernetes/best-practices-for-building-kubernetes-operators-and-stateful-apps>

How does an operator works?



Software Developer
Kubernetes user

K8s API

Custom resource

```
kind: CuttingEdgeDatabase
apiVersion:
  database.example.com /v1alpha1
metadata:
  name: my-important-database
spec:
  connectionPoolSize: 300
  readReplicas: 2
  version: v4.0.1
```



Kubernetes operator

Custom Kubernetes controller

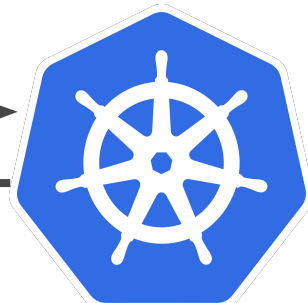
Watch Event

Reconcile

Custom resource definition

here CuttingEdgeDatabase

Native Kubernetes
resources



Deployments
StatefulSets
Autoscalers
Secrets
Config maps

Why should you use an operator?

Operators: both sysadmin + application experts

🔧 **Resize/Upgrade**

🔧 **Reconfigure**

🔧 **Backup**

🔧 **Healing**



The Sysadmin

Operator across the industry

Operator across the industry

OperatorHub.io | The registry for Kubernetes Operators



Multiple operator frameworks

Operators

- **kudo**: simple, no need to code, not so popular:
<https://github.com/kudobuilder/operators/tree/master/repository>
- **metacontroller**: simple, no need to code, started at Google

Based on [kubernetes-sigs/controller-runtime: Repo for the controller-runtime subproject of kubebuilder \(sig-apimachinery\)](#)

and [kubernetes-sigs/controller-tools: Tools to use with the controller-runtime libraries](#)

- **operator-framework**: complex, code in golang, popular, well-documented (book), from RedHat
- **kubebuilder**: complex, code in golang, popular, well-documented (book)

See <https://gist.github.com/tiewei/d98c663cf76b61bf835c1ebf87b36999>

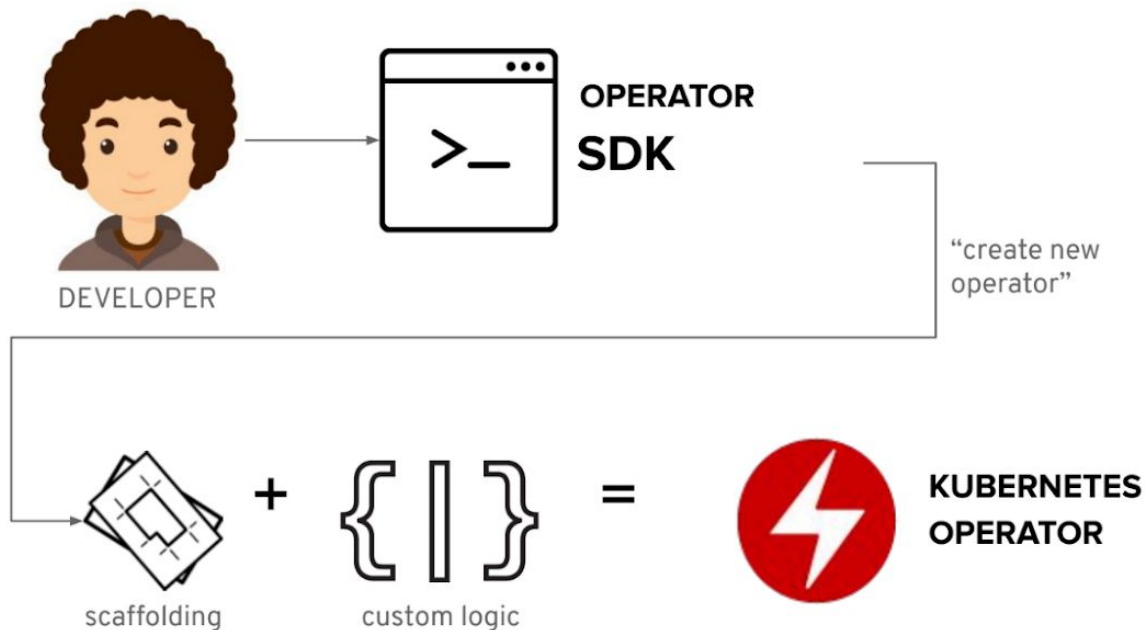
operator-framework

Operator framework in action

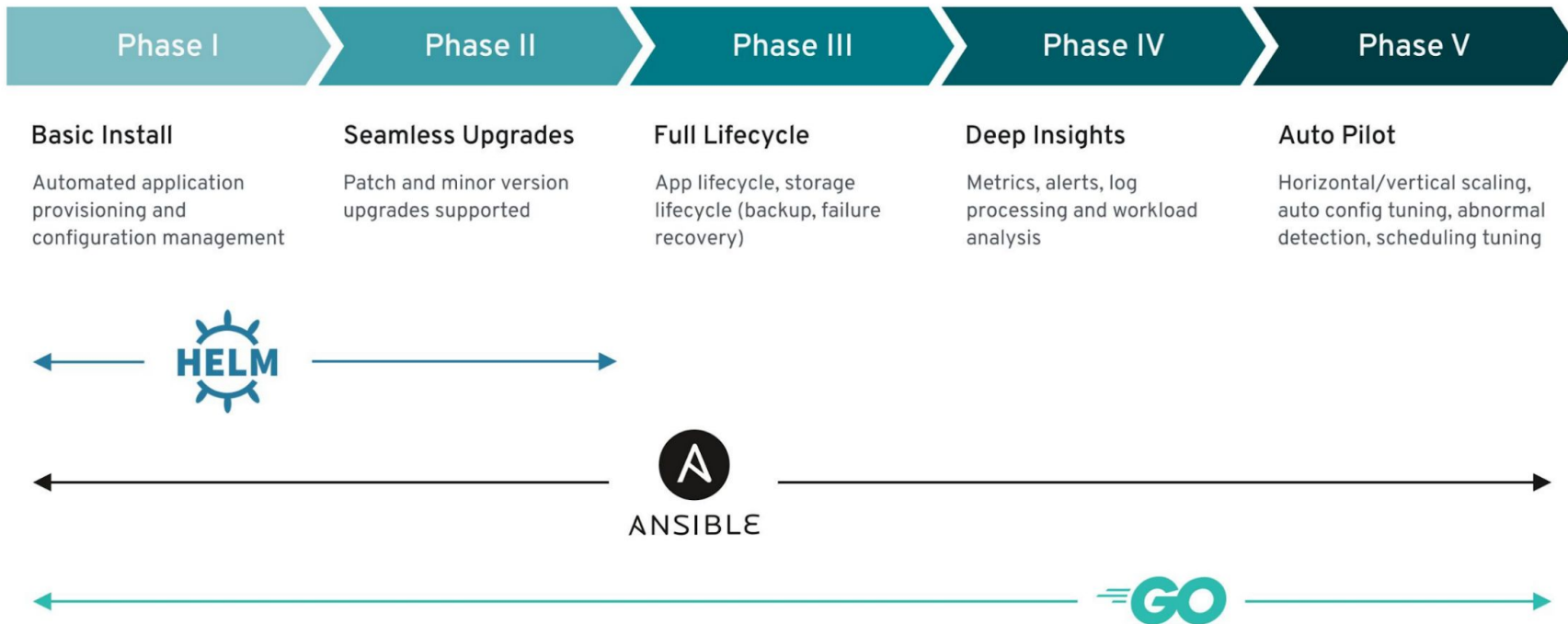
Based on KubeBuilder libraries:

[kubernetes-sigs/controller-runtime](https://github.com/kubernetes-sigs/controller-runtime): Repo for the controller-runtime subproject of kubebuilder (sig-apimachinery)

[kubernetes-sigs/controller-tools](https://github.com/kubernetes-sigs/controller-tools): Tools to use with the controller-runtime libraries



Operator SDK: types of operators



OperatorHub: the MongoDB example

OperatorHub.io

Search OperatorHub...

Contribute ▾

[Home](#) > [MongoDB](#)

MongoDB

The MongoDB Enterprise Kubernetes Operator enables easy deploys of MongoDB into Kubernetes clusters, using our management, monitoring and backup platforms, Ops Manager and Cloud Manager.

The Operator has beta support for a containerized Ops Manager with the MongoDB0psManager custom resource.

Install

CHANNEL

stable

VERSION

1.4.1 (Current) ▾

CAPABILITY LEVEL ⓘ

☒ Basic Install

☒ Seamless Upgrades

☒ Full Lifecycle

☒ Deep Insights

☐ Auto Pilot

PROVIDER

MongoDB, Inc

LINKS

[Documentation](#) ↗

REPOSITORY

<https://github.com/mongodb/mongodb-kubernetes-operator>

Before You Start

To start using the operator you'll need an account in MongoDB Cloud Manager or a MongoDB Ops Manager deployment.

- [Create a Secret with your OpsManager API key](#)
- [Create a ConfigMap with your OpsManager project ID and URL](#)

By installing this integration, you will be able to deploy MongoDB instances with a single simple command.

Required Parameters

Qserv

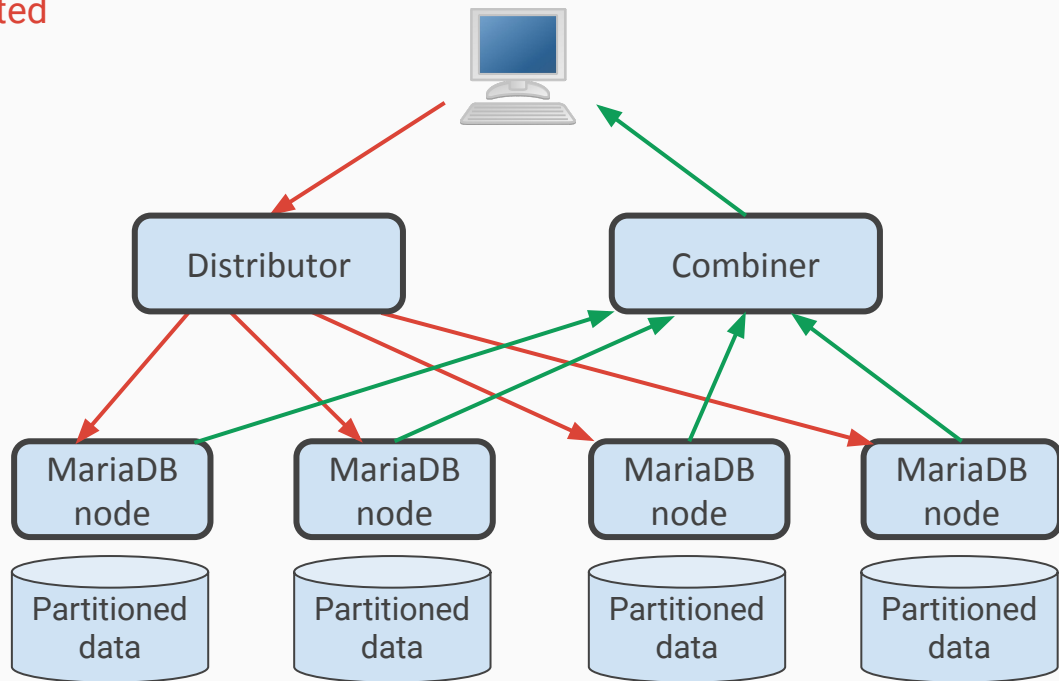
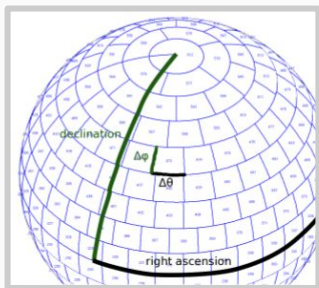
The LSST Petascale database

Qserv design

Relational database, 100% open source

Spatially-sharded with overlaps

Map/reduce-like processing, highly distributed



Kubernetes operator for Qserv

The Qserv custom resource: qserv.yaml

```
apiVersion: qserv.lsst.org/v1alpha1
```

```
kind: Qserv
```

```
metadata:
```

```
  name: qserv
```

```
spec:
```

```
  imagePullPolicy: "Always"
```

```
  storageclass: "standard"
```

```
  storagecapacity: "5Ti"
```

```
  czar:
```

```
    image: "qserv/qserv:dcbfff7"
```

```
  ingest:
```

```
    dbimage: "mariadb:10.2.16"
```

```
  worker:
```

```
    replicas: 30
```

```
    image: "qserv/qserv:dcbfff7"
```

```
  replication:
```

```
    image: "qserv/replica:tools-w.2018.16-1347-g5de8f05-dirty"
```

```
    dbimage: "mariadb:10.2.16"
```

```
  xrootd:
```

```
    image: "qserv/qserv:dcbfff7"
```

```
    replicas: 2
```

Describe how to install a custom Qserv cluster on **any** Kubernetes platform

Embed xrootd servers

Manage xrootd redirectors

What it does @CC-IN2P3

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
qserv-operator-694db99875-ss67r	1/1	Running	0	27d

Prerequisite

```
$ kubectl apply -k overlays/in2p3/
```

```
secret/secret-mariadb-qserv-dev unchanged
secret/secret-repl-db-qserv-dev unchanged
secret/secret-wmgr-qserv-dev unchanged
qserv.qserv.lsst.org/qserv-dev created
```

Create a custom Qserv instance, based on qserv.yaml

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
Qserv-dev-czar-0	1/1	Running	0	5m48s
qserv-dev-repl-ctl-0	1/1	Running	0	5m48s
qserv-dev-repl-db-0	1/1	Running	0	5m48s
qserv-dev-worker-0	5/5	Running	0	5m48s
qserv-dev-worker-1	5/5	Running	0	5m48s
qserv-dev-worker-2	5/5	Running	0	5m47s
...				
qserv-dev-worker-29	5/5	Running	0	5m45s
qserv-dev-xrootd-redirector-0	2/2	Running	0	5m48s
qserv-dev-xrootd-redirector-1	2/2	Running	0	5m48s
qserv-operator-694db99875-ss67r	1/1	Running	0	27d


In seconds
Qserv+xrootd ssi plugin
is up and running on
CC-IN2P3
pre-production platform

Summary

What we have seen:

- Operators **ease application delivery and management** over Kubernetes.
- Operator goal is to **automate sysadmins tasks**.
- **Multiple operator frameworks** are competing right now.
- **Qserv operator** works fine and is build on top of RedHat operator-sdk

=> And what about pure xrootd?



Kubernetes operator for XRootD cluster

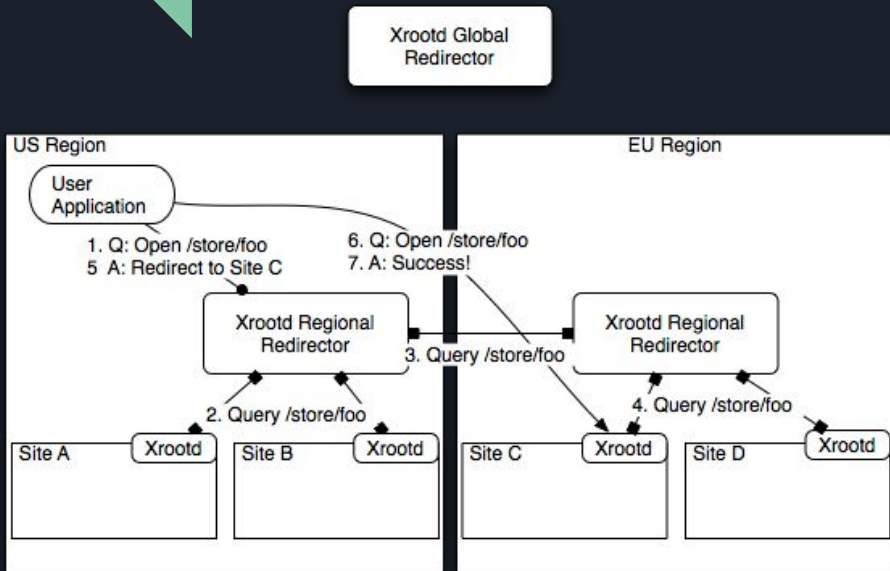
Hosted at [xrootd/xrootd-k8s-operator](https://github.com/xrootd/xrootd-k8s-operator)



Project Goals

- To develop a operator that:
 - eases and fully automate deployment and management of XRootD clusters
 - targeted for all clusters compliant with Kubernetes API
 - is intended for use by the XRootD community in order to scale-up worldwide XRootD clusters management
 - is easy-to-install and has seamless upgrades
 - provides deep insights to the cluster state and alerts on failure
- Write well-written documentation for the operator that:
 - describes the installation and update process
 - explains configuration options for XRootD cluster
 - describes how to extend the cluster
 - documents the contribution guidelines and development process

XRootD Protocol



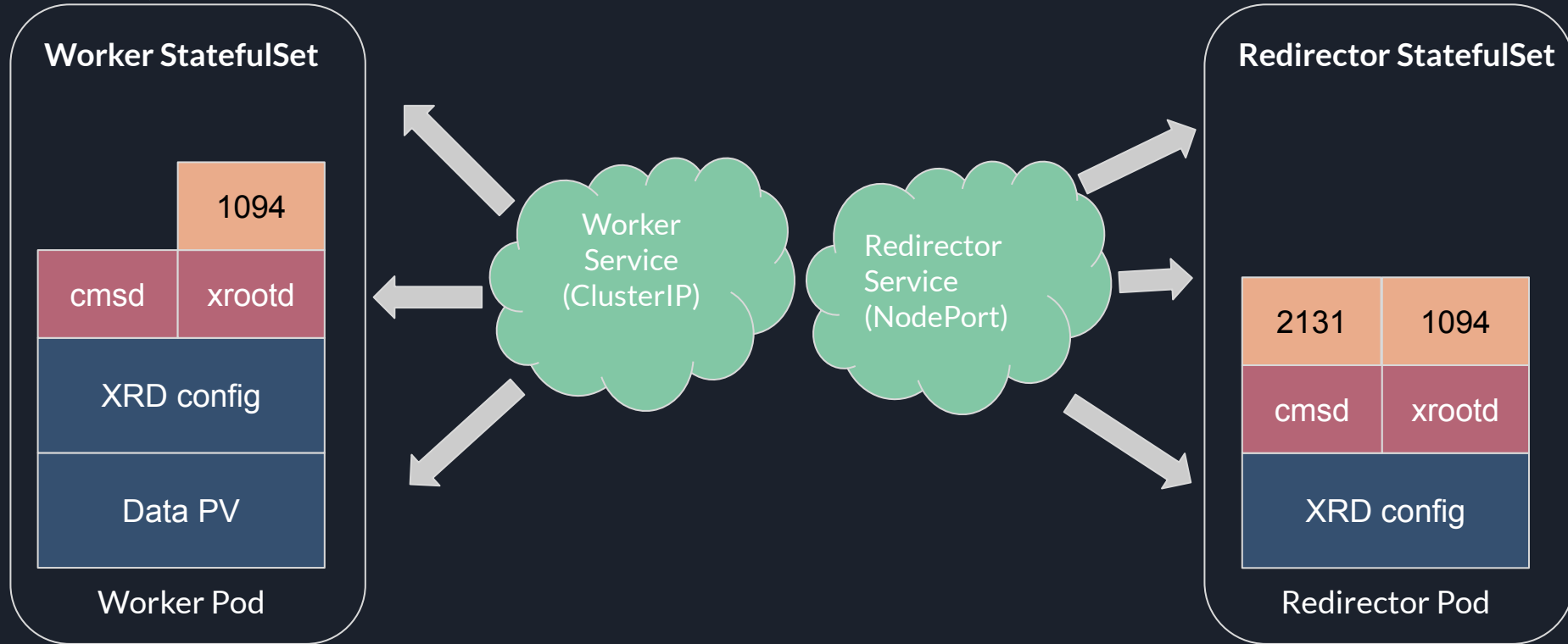
XRootD protocol enables high performance, scalable fault-tolerant access to data repositories of various kinds, including EOS.

It is meant to solve the **Any Data, Anytime, Anywhere (AAA)** requirement to access the remote files regardless if they are present in your region or halfway around the world!

It's possible by abstracting two types of nodes in any XRootD cluster:

1. **Redirectors** - These nodes coordinate the function of the cluster and enable communication via Intra-region and Cross-region redirection
2. **Workers** - These nodes are actually the ones storing and providing the data to the client

XRootD Cluster Architecture





Installation

OLM via OperatorHub

- Install OLM in your cluster
- Install **Subscription CR** for Xrootd operator
- OLM will now fetch the latest operator bundle image, belonging to the specified channel
- OLM will install the required CRDs, permissions, role and operator deployment
- Updating operator is seamlessly handled by OLM

Manually via script

- Deploy the operator using [installation script](#)
- Updating operator version requires manual re-installation



Cluster Configuration via CRDs

XRootD CRD

```
apiVersion: xrootd.org/v1alpha1
kind: Xrootd
metadata:
  name: base-xrootd
spec:
  version: 4.11.2
  redirector:
    replicas: 2
  worker:
    replicas: 3
    storage:
      capacity: "1Gi"
      class: "default"
```

XRootD Version Catalog CRD

```
apiVersion: catalog.xrootd.org/v1alpha1
kind: XrootdVersion
metadata:
  name: 4.11.2
spec:
  version: 4.11.2
  deprecated: false
  image: "qserv/xrootd:v4.11.2"
```

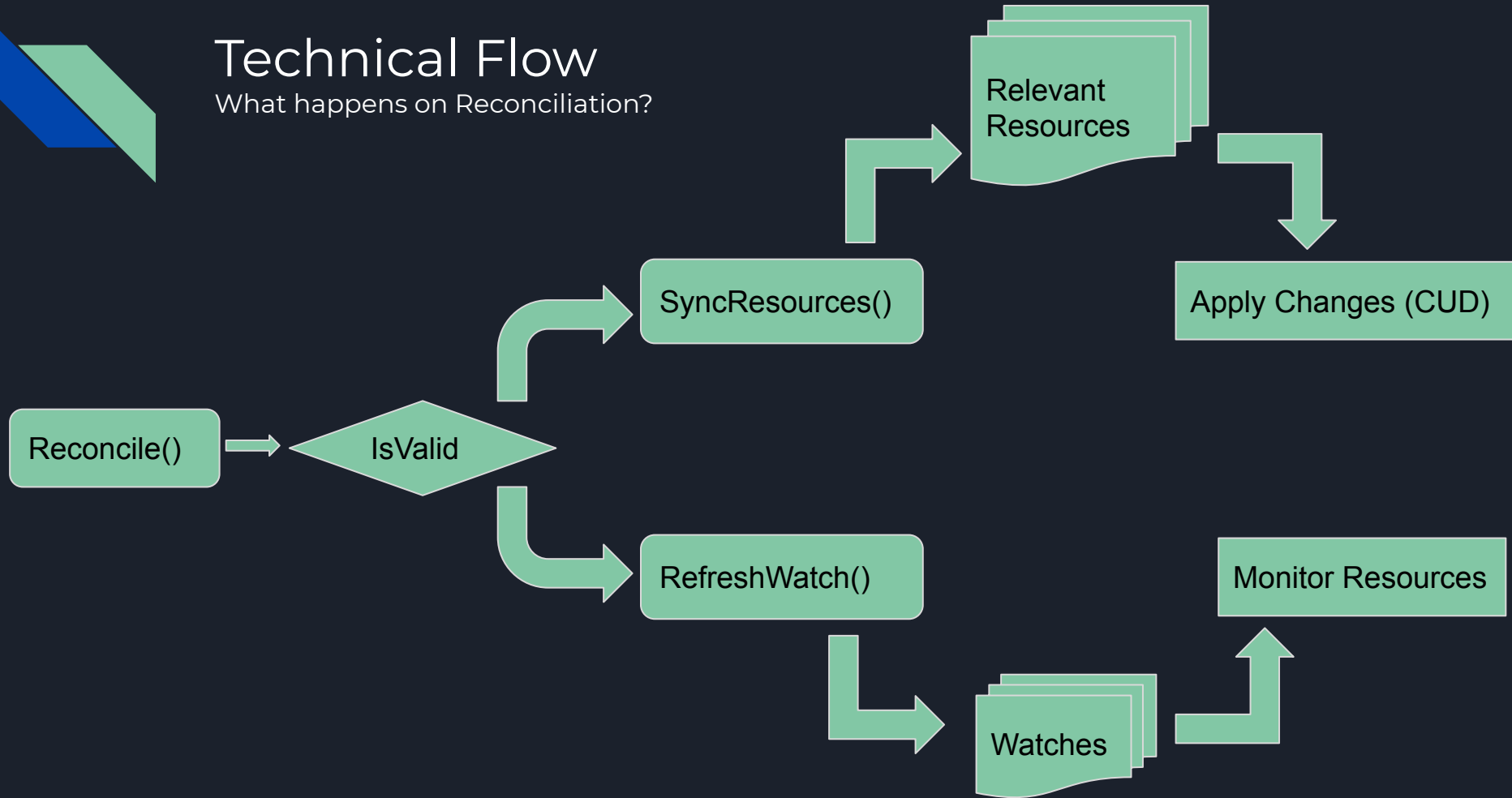
Example

kubectl apply -k manifests/base

```
xrootd-operator-2418c92f8c-pqczj 1/1 Running 0 23s
A ~ proj... xrootd-k8s-operator P master 0 kubectl get po
NAME READY STATUS RESTARTS AGE
base-xrootd-xrootd-redirector-0 2/2 Running 0 19s
base-xrootd-xrootd-redirector-1 2/2 Running 0 18s
base-xrootd-xrootd-worker-0 2/2 Running 0 19s
base-xrootd-xrootd-worker-1 2/2 Running 0 18s
base-xrootd-xrootd-worker-2 2/2 Running 0 14s
xrootd-operator-54f8c4978c-pqczj 1/1 Running 0 26s
A ~ proj... xrootd-k8s-operator P master 0 kubectl logs xrootd-operator-54f8c4978c-pqczj
{"level":"info","ts":1595486965.3281264,"logger":"cmd","msg":"Operator Version: 0.0.1"}
{"level":"info","ts":1595486965.3281717,"logger":"cmd","msg":"Go Version: go1.14.4"}
{"level":"info","ts":1595486965.3281865,"logger":"cmd","msg":"Go OS/Arch: linux/amd64"}
{"level":"info","ts":1595486965.328196,"logger":"cmd","msg":"Version of operator-sdk: v0.18.2"}
{"level":"info","ts":1595486965.3286107,"logger":"leader","msg":"Trying to become the leader."}
{"level":"info","ts":1595486965.9872425,"logger":"leader","msg":"No pre-existing lock was found."}
{"level":"info","ts":1595486965.9890609,"logger":"leader","msg":"Became the leader."}
{"level":"info","ts":1595486966.6504905,"logger":"controller-runtime.metrics","msg":"metrics server is starting to listen","addr":"0.0.0.0:8383"}
{"level":"info","ts":1595486966.6511803,"logger":"cmd","msg":"Registering Components."}
{"level":"info","ts":1595486968.005825,"logger":"metrics","msg":"Metrics Service object created","Service.Name":"xrootd-operator-metrics","Service.Namespace":"default"}
{"level":"info","ts":1595486968.6588824,"logger":"cmd","msg":"Could not create ServiceMonitor object","error":"no ServiceMonitor registered with the API"}
{"level":"info","ts":1595486968.6589682,"logger":"cmd","msg":"Install prometheus-operator in your cluster to create ServiceMonitor objects","error":"no ServiceMonitor registered with the API"}
{"level":"info","ts":1595486968.6590314,"logger":"cmd","msg":"Starting the Cmd."}
{"level":"info","ts":1595486968.6598232,"logger":"controller-runtime.manager","msg":"starting metrics server","path":"/metrics"}
{"level":"info","ts":1595486968.6600018,"logger":"controller-runtime.controller","msg":"Starting EventSource","controller":"xrootd-controller","source":"kind source: /, Kind="}
{"level":"info","ts":1595486968.7611315,"logger":"controller-runtime.controller","msg":"Starting Controller","controller":"xrootd-controller"}
{"level":"info","ts":1595486968.761208,"logger":"controller-runtime.controller","msg":"Starting workers","controller":"xrootd-controller","worker count:1"}
{"level":"info","ts":1595486971.1588905,"logger":"controller_xrootd","msg":"Reconciling Xrootd","Request.Namespace":"default","Request.Name":"base-xrootd"}
{"level":"info","ts":1595486971.1589148,"logger":"controller_xrootd","msg":"Started syncing resources...","Request.Namespace":"default","Request.Name":"base-xrootd"}
{"level":"info","ts":1595486971.1590047,"logger":"objects.scanDir","msg":"Scanning file...","path":"/configmaps/xrootd/etc"}
{"level":"info","ts":1595486971.1590245,"logger":"objects.scanDir","msg":"Scanning file...","path":"/configmaps/xrootd/etc/xrootd.cf"}
{"level":"info","ts":1595486971.1591117,"logger":"objects.scanDir","msg":"Scanning file...","path":"/configmaps/xrootd/run"}
{"level":"info","ts":1595486971.1591249,"logger":"objects.scanDir","msg":"Scanning file...","path":"/configmaps/xrootd/run/start.sh"}
{"level":"info","ts":1595486971.4609008,"logger":"controller_xrootd.syncResources","msg":"Processing delta","create":2,"update":0,"delete":0,"type":"*v1.Service"}
{"level":"info","ts":1595486971.4903827,"logger":"controller_xrootd.syncResources","msg":"Executed changes","added":true,"updated":false,"removed":false}
{"level":"info","ts":1595486971.490548,"logger":"controller_xrootd.syncResources","msg":"Processing delta","create":2,"update":0,"delete":0,"type":"*v1.StatefulSet"}
{"level":"info","ts":1595486971.518051,"logger":"controller_xrootd.syncResources","msg":"Executed changes","added":true,"updated":false,"removed":false}
{"level":"info","ts":1595486971.5182593,"logger":"controller_xrootd.syncResources","msg":"Processing delta","create":2,"update":0,"delete":0,"type":"*v1.ConfigMap"}
{"level":"info","ts":1595486971.523892,"logger":"controller_xrootd.syncResources","msg":"Executed changes","added":true,"updated":false,"removed":false}
{"level":"info","ts":1595486971.5239158,"logger":"controller_xrootd","msg":"Started watching resources...","Request.Namespace":"default","Request.Name":"base-xrootd"}
{"level":"info","ts":1595486971.5239248,"logger":"controller_xrootd","msg":"Watching Xrootd resources...","Request.Namespace":"default","Request.Name":"base-xrootd"}
{"level":"info","ts":1595486971.523931,"logger":"controller_xrootd","msg":"Reconciled successfully!","Request.Namespace":"default","Request.Name":"base-xrootd"}
{"level":"info","ts":1595486971.5239756,"logger":"Watcher.GroupedRequestWatcher.Watch","msg":"Refreshing watch...","request":"default/base-xrootd"}
{"level":"info","ts":1595486971.5241103,"logger":"XrootdLogsWatcher","msg":"Started monitoring xrootd cluster...","request":"default/base-xrootd","component":"xrootd-worker"}
{"level":"info","ts":1595486971.5247476,"logger":"Watcher.GroupedRequestWatcher.Watch","msg":"Refreshing watch...","request":"default/base-xrootd"}
{"level":"info","ts":1595486971.5248973,"logger":"XrootdLogsWatcher","msg":"Started monitoring xrootd cluster...","request":"default/base-xrootd","component":"xrootd-redirector"}
{"level":"info","ts":1595486971.626039,"logger":"XrootdLogsWatcher","msg":"Fetched pods...","request":"default/base-xrootd","component":"xrootd-redirector","pods":1}
{"level":"info","ts":1595486971.6262398,"logger":"XrootdLogsWatcher","msg":"Fetched pods...","request":"default/base-xrootd","component":"xrootd-worker","pods":1}
{"level":"info","ts":1595486972.7933846,"logger":"XrootdLogsWatcher","msg":"Greping and reading...","pod":"base-xrootd-xrootd-redirector-0","component":"xrootd-redirector","regex":"Protocol:
redirector.+ logged in.$"}
{"level":"info","ts":1595486972.7939768,"logger":"XrootdLogsWatcher","msg":"Greping and reading...","pod":"base-xrootd-xrootd-worker-0","component":"xrootd-worker","regex":"Protocol: logged i
nto .+$"}
A ~ proj... xrootd-k8s-operator P master 0
```

Technical Flow

What happens on Reconciliation?





Monitoring and Insights

- Integration with OLM provides Declarative UI controls
- Status of Xrootd CR is updated with connected workers and redirectors

DEMO

