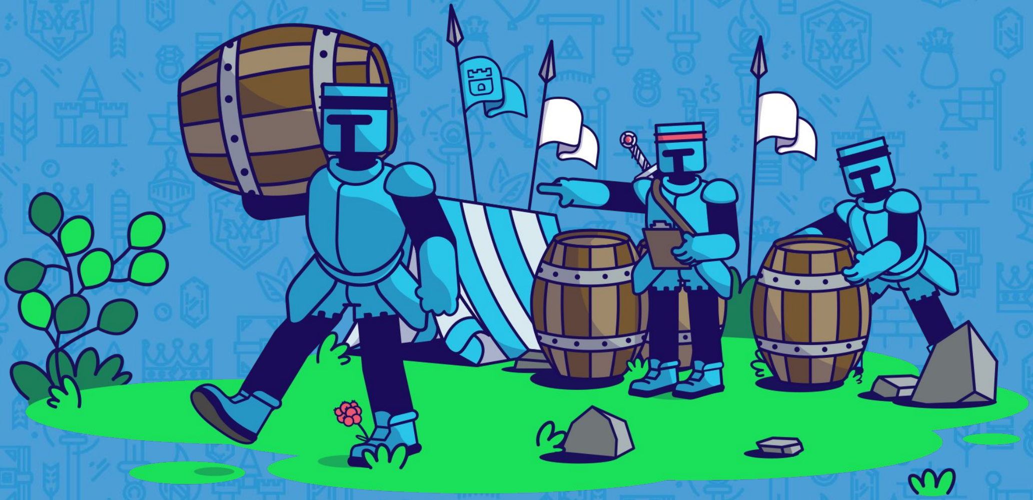
Data Without Borders

Using Rook Storage Orchestration at a Global Scale

Jared Watts
Rook Senior Maintainer
Upbound Founding Engineer

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



Why Deploy Globally (multi-cloud)?

- **Reliability**: Avoid downtime despite failures/outages
- **Performance**: Run as close to your users as possible
- **Cost:** Choosing the cheapest option, free credits
- **Innovation**: Taking advantage of a new features/offerings
- **Compliance**: Government policy, data sovereignty
- **Hybrid**: First foray into public cloud
- **Mergers**: Different businesses with different infrastructures
- **Policy**: multi-vendor organizational policy at all levels

Terminology

- **Availability** - resistance to downtime
- **Durability** - resistance to total loss
- **Latency** - delay in networking communication
- **Locality** - geographic topology awareness
- **Replication/Mirror** - copying the data in its entirety to another site
- **Snapshots** - point in time “complete” state
- **Disaster recovery** - BOOM...now what?

What is Rook?

- Cloud-Native Storage Orchestrator
- Extends Kubernetes with custom types and controllers
- Automates deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management
- Framework for many storage providers and solutions
- Open Source (Apache 2.0)
- Hosted by the Cloud-Native Computing Foundation (CNCF)

Control Planes & Data Planes

- Rook is an orchestrator - it’s the control plane, not the data plane
- **Data plane**: reading and writing of bytes
- **Control plane**: bootstraps, deploys, configures, manages the data plane
- Rook orchestrates multiple data planes including EdgeFS, Ceph, CockroachDB, and others
- Similar distinction between Istio (control) and Envoy (data)

Data Plane Approaches

Data plane architectures

- At a high level, there are two different architectures
- Storage systems that work at **global** scale (EdgeFS, CockroachDB)
- Storage systems that work at **local** scale, and **federate** at global scale (Ceph, MySql, etc.)
- The examples on the following slides are all orchestrated by Rook's control plane

Ceph

- Core architecture is for a single local cluster
- NOT designed to be natively global
- Strongly consistent storage model (<5ms latency between nodes)
    - All writes acknowledged by replicas before committed
- Highly scalable due to decentralized data placement
- Asynchronous replication (mirroring) of blocks and object storage across clusters
- Reliability, availability, disaster recovery

EdgeFS

- Natively designed to be global storage
- Based on immutable blocks similar to `Git`
  - modifications are globally unique and versioned
  - modification results in a new identity
  - caches are always in a consistent state
  - allows global fault tolerance, global scalability
- Segmented storage - stitching clouds into one single geo-namespace
  - ISGW - Inter-Segment GateWay

EdgeFS - metadata only

- Mode to (initially) transfer metadata only across segments
- Full file listings and info are available globally *fast*
- Data chunks will be fetched lazily (on demand)
- Critical for enabling a globally remote client to start consuming data as soon as it is created

EdgeFS - deduplication and recovery

- Global deduplication - multiple identical chunks are only stored once
- Built in disaster recovery: lost data chunks can be recovered from remote segments transparently
- Client sees a temporary loss in throughput, but no errors while fetching from remote
- Local site cache is repopulated with recovered data

CockroachDB

- Natively designed to be global storage
- Built in locality awareness
- Your data isn't bound by the data centers of just one cloud provider
- Design goal: minimize latency (data close to users) without sacrificing availability (high number of replicas across environments)

CockroachDB - Distributed Algorithms

- The nodes self-organize via a Gossip protocol and how the cluster replicates data via the Raft consensus algorithm
- Nodes self-organize with Gossip protocol
  - every node has up-to-date details of other nodes in the cluster (e.g., location of data, storage capacity)
- Cluster replicates data with Raft consensus algorithm
  - ensures that every “range” of data is replicated and all replicas are consistent
  - Raft also used by `etcd`

CockroachDB - Locality Awareness

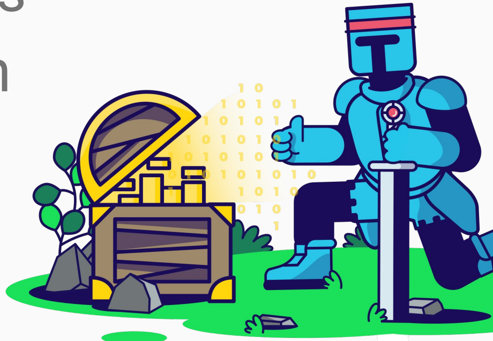
-
  ```
  cockroach start --locality region=us-west
  ```
- Locality information to increase “replica diversity”
  - data copies stored on machines in different localities
- During failures, diversity can be sacrificed in favor of replica count
- Replication constraints

  -
    ```
    ALTER DATABASE mydb CONFIGURE ZONE USING
    constraints='[+region=EU]'
    ```
  - Applied at any level (cluster, database, table, row)

Control Plane Approaches

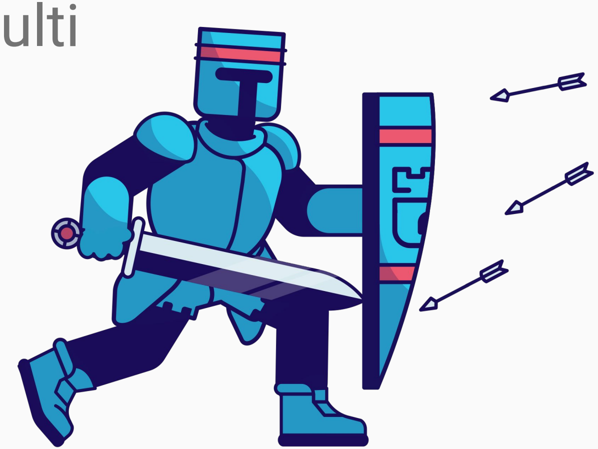
Global Storage Orchestration

- Rook orchestrates at the level of a single cluster
- We still need a “global orchestrator” that spans clusters and clouds to enable true global data
- Could deploy all components of a global data plane like EdgeFS across clusters
- Could setup replication/mirroring/federation across clusters for local cluster storage systems like Ceph



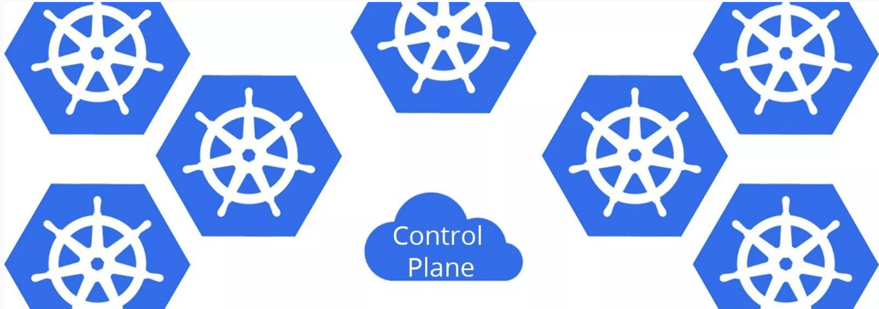
Global Control Plane

- This would be the control plane for global storage systems
- Could orchestrate movement of data and automate disaster recovery and failover
- A critical piece of infrastructure for this is a multi cluster and multi-cloud control plane



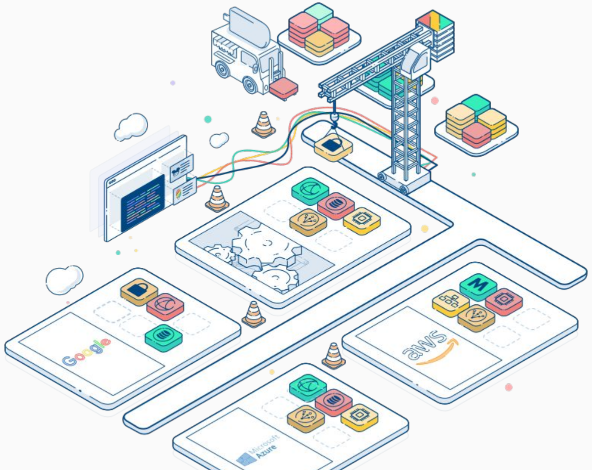
Kubefed (Federation v2)

- **Templates**: representation of a resource common across clusters
- **Placement**: which clusters the resource is intended to appear in
- **Overrides**: define per-cluster field-level variation to apply to the template
- **Propagation**: Distributes resources amongst federated clusters



Crossplane

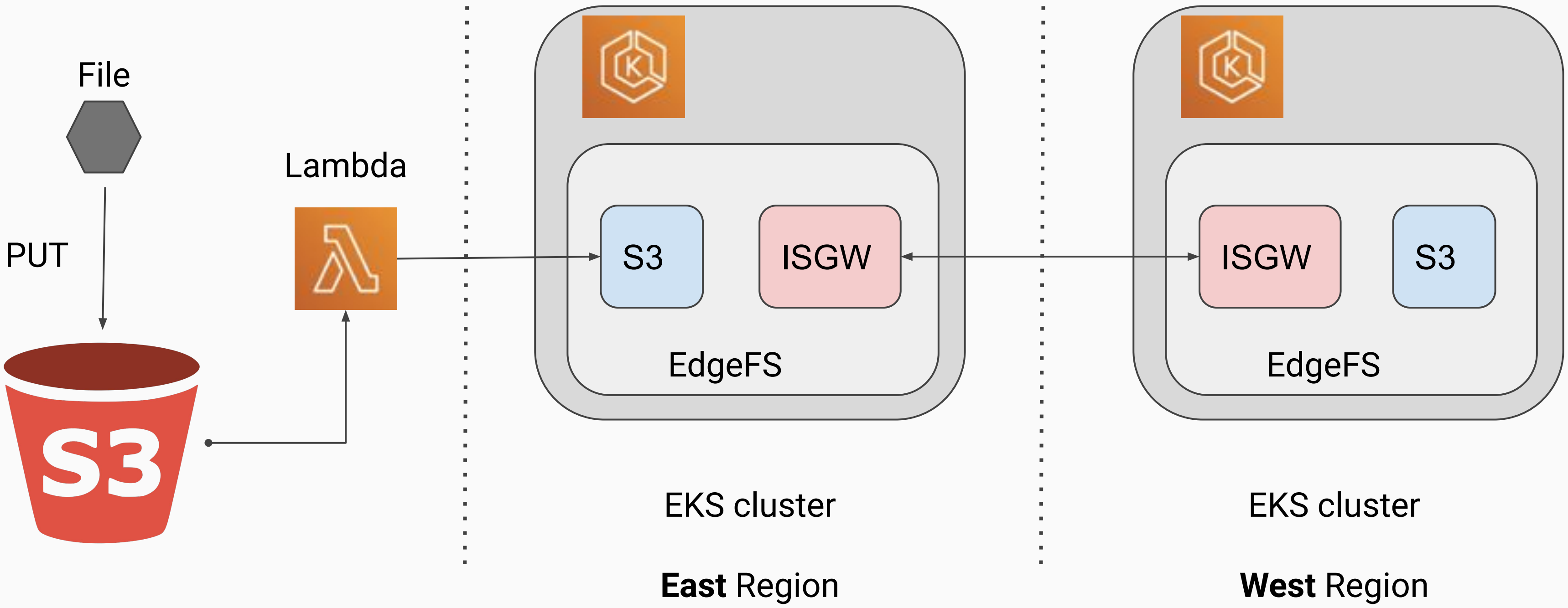
- Extends Kubernetes and spans multiple clouds and clusters
- Deploys infrastructure, platform services, and applications
- Smart scheduler to globally optimize placement
- Portable resource abstractions
- Open source and community driven
  - crossplane.io



Example Global Rook-EdgeFS

- EKS clusters in separate regions (global distribution)
- Rook creates EdgeFS cluster in both EKS
- Each EdgeFS cluster exposes S3 service and bucket
- ISGW link between EdgeFS clusters - bi-directional sync
- AWS S3 bucket with Lambda function for bucket events
    - Syncs event to EdgeFS bucket in 1 of the EKS clusters
    - ISGW syncs event to other EdgeFS in other cluster
- Google Cloud also supported, Azure is in progress
    - all Clouds could be stitched together seamlessly

Example Global Rook-EdgeFS



Demo

Globally Distributed Rook-EdgeFS clusters

Thanks to Dmitry Yusupov and Ilya Grafutko from EdgeFS team

How to get involved?

- Contribute to Rook and Crossplane
  - https://rook.io/
  - https://crossplane.io/
- Slack
  - https://slack.rook.io/
  - https://slack.crossplane.io/
- Twitter - @rook_io & @crossplane_io
- Forums - rook-dev & crossplane-dev on google groups
- Community Meetings

Rook sessions at Kubecon

Rook Deep Dive
Wednesday, **11:55** @ Hall 8.1 G3

Meet the Maintainers
Wednesday, **12:30** @ CNCF Answer Bar

Keep the Space Shuttle Flying: Writing Robust Operators
Wednesday, **15:55** @ Hall 8.1 G2

Rook, Ceph, and ARM: A Caffeinated Tutorial
Wednesday, **16:45** @ Hall 8.0 D2

Thank you!

https://rook.io/

https://crossplane.io/