Advancing clustered storage architecture with Kubernetes

Daniel Kerns
Quantum
Agenda

- Storage Systems Architecture
- Container Orchestration / Kubernetes
- The Rook Project
- Advantages and Disadvantages
- Demo
Storage Systems Architecture

- These are complex distributed systems
- > 10 distinct services
- Often 1,000s of instances
- Operationally complex
Kubernetes Features

- Self-healing
- Binpacking
- Horizontal Scaling
- Service Discovery
- Load Balancing
- Rollout and Rollback
- Secrets
- Configuration Mgt
- Batch Execution
Container Orchestration

- Declarative Automation for:
  - Placement
  - Scheduling
  - Deployment
  - Update
  - Health
  - Scalability
  - Failover
Kubernetes

- Declarative Model
- Diverse Services
- Extensible control plane
- Complete application lifecycle support
Storage in Kubernetes Clusters

- Storage is handled externally to the cluster

Dynamic Volume Provisioning

- Ceph
- OpenZFS
- Lustre
- GLUSTER
Rook

- Bring storage into the cluster and make storage systems self-managing
  - Not exclusively Ceph
- The Rook Operator leverages Kubernetes to manage Ceph
- Both appliance and converged applications

https://rook.io
Storage Services running inside the Cluster

Provisioning
Open Source
Multi-Cloud
Federation
Integrated Security
Integrated Scheduling
Integrated QoS
Availability
Backup and DR
Upgrades
Migrations
Monitoring
Rook Components

- Rook Operator
- Ceph (mon, osd, rgw, mds, mgr)
- NFS, CIFS, iSCSI
- Rook Agent
- Rook API
Rook Components - Singletons

- Rook Operator
- Ceph Manager
- Rook API

Generally run one instance of these. Multiple instances for HA configurations.
Rook Components – Ceph Mon’s

- Monitors do not like to be moved
  - Use service IP’s for monitors
- Must be able to fail
Rook Components – OSD’s

- One OSD pod per storage node
- Each pod contains N OSD’s – generally one per drive
- Add necessary metadata comes from the Operator
  - No user supplied ceph.conf files
- Can optionally write metadata to different devices
Rook Components – Other Services

- RGW and MDS Daemons
  - Scale as needed based on traffic and load
  - Can specify minimums / maximums
- SSL termination normally done at Kubernetes load balancer
What we get from Kubernetes

- Lifecycles are controlled by Kubernetes
  - node, pod, process
- Automatic failover
- Upgrade support
- Cluster-level metadata
- Secrets
- Extensible API
Demo: Rook in Kubernetes
Downsides

- We don’t own the software stack
- Stack is rapidly evolving
- We don’t have total control
  - Trading opportunity cost for total control
Conclusion

- Kubernetes turns lifecycle code into lifecycle declarations
  - Developers have more time to spend on logic
  - Operationally complex systems can be simplified
- Overall systems reliability is increased
Questions – Shameless Plug

- Rook is taking pull requests at http://github.com/rook
- We’re hiring Rook developers:
  - info@rook.io
  - http://quantum.com/jobs