KubeCon | CloudNativeCon

Europe 2019

Build a Kubernetes based cloud-native storage software from scratch

Sheng Yang, Rancher Labs

Project Longhorn

Open Source
Distributed Block Storage Software
For Kubernetes
https://github.com/rancher/longhorn/

Add persistent storage support to any Kubernetes cluster

```
kubectl apply –f longhorn.yaml
```

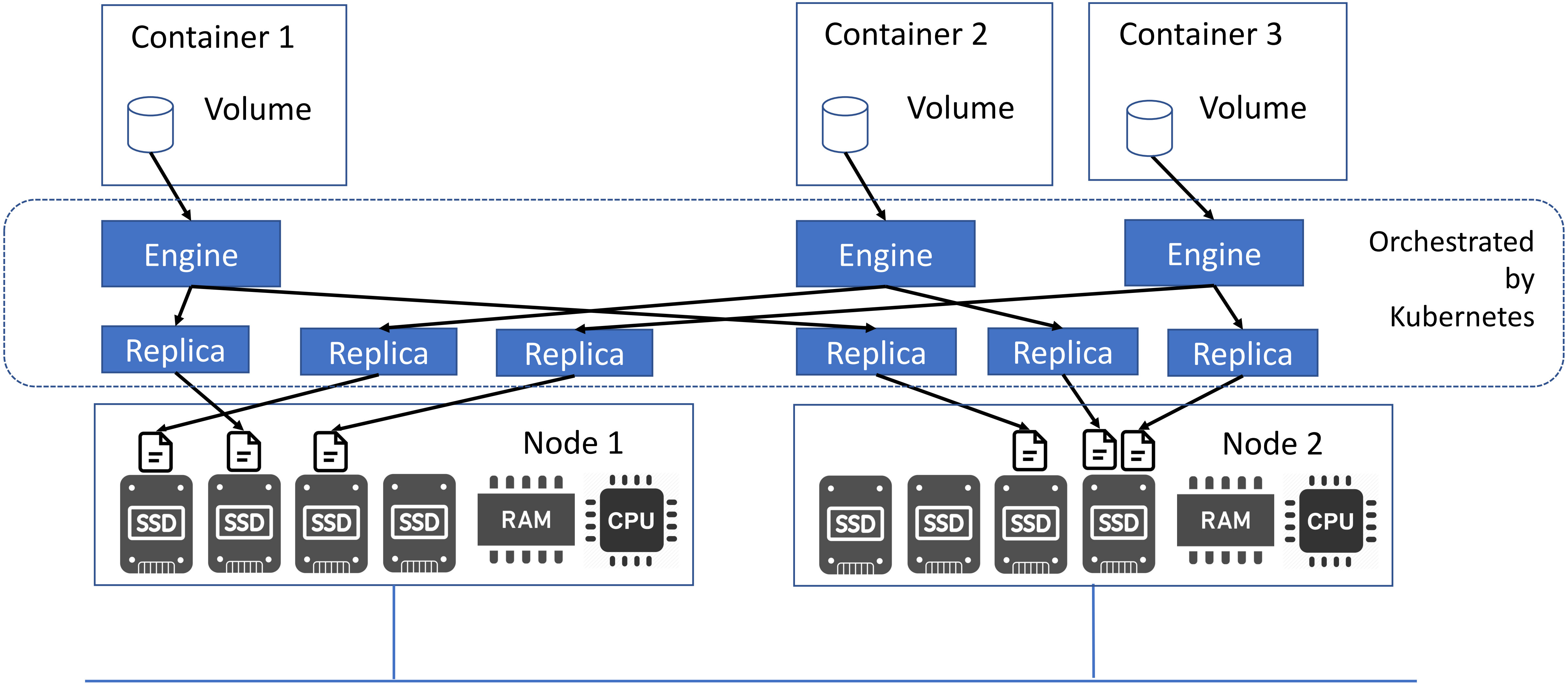
Compare Longhorn to legacy storage software

Legacy Storage Software	Longhorn
Complex code for storage stack and controller HA	30k Go code, leveraging proven Linux storage features (e.g. sparse file and cgroups QoS) and Kubernetes Orchestration

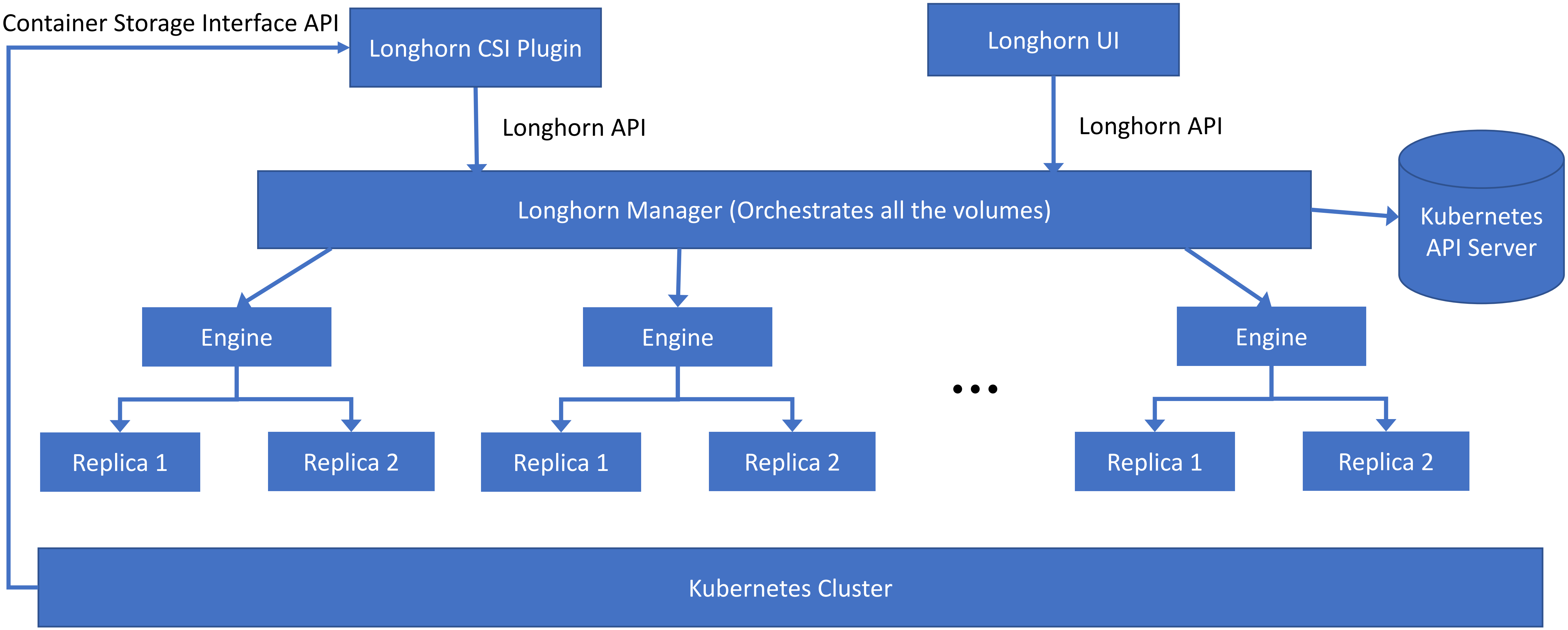
Latest release: Longhorn v0.5.0

- Enterprise-grade distributed block storage software for Kubernetes
- Volume snapshots
- Volume backup and restore
- Live upgrade of Longhorn software without impacting running volumes
- Cross-cluster disaster recovery volume with defined RTO and RPO
- Intuitive UI
- One click installation
- And more features are coming
  - QoS, volume resizing, real time performance monitoring, etc

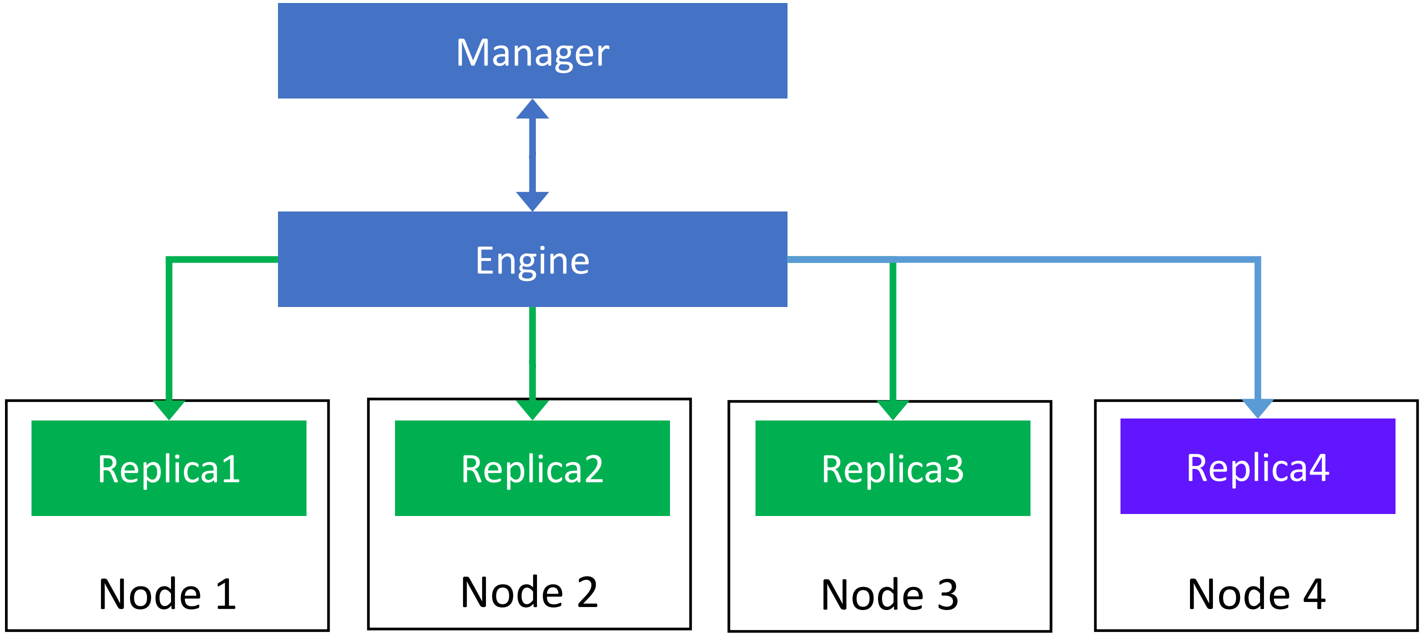
Longhorn Architecture - Engine



Longhorn Architecture - Manager



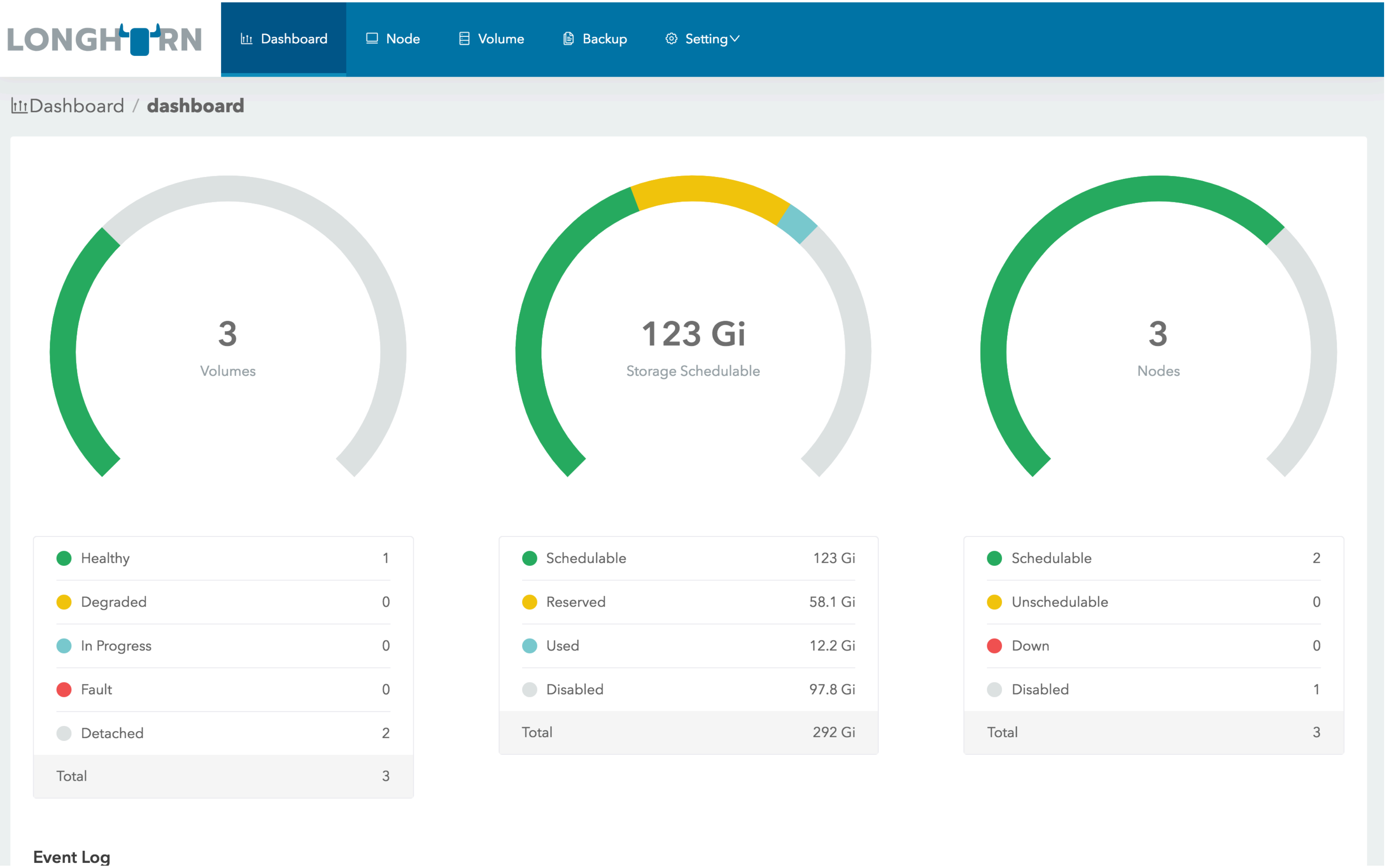
Cornerstone: Controller Pattern



```
volume:
   spec:
      numberOfReplicas: 3
    status:
       currentHealthyReplicas:    2

  engine:
     spec:
        replicaList:
          Replica1
          Replica2
          Replica4
     status:
        replicaList:
          Replica1
          Replica2
          Replica4
```

Demo



Dashboard / **dashboard**

3
Volumes

123 Gi
Storage Schedulable

3
Nodes

● Healthy	1
● Degraded	0
● In Progress	0
● Fault	0
● Detached	2
Total	3

● Schedulable	123 Gi
● Reserved	58.1 Gi
● Used	12.2 Gi
● Disabled	97.8 Gi
Total	292 Gi

● Schedulable	2
● Unschedulable	0
● Down	0
● Disabled	1
Total	3

Event Log

Kubernetes helps to increase resiliency

- Automatic node status update
  - Make it easier to deal with failed/pressured nodes
- Automatic pod status update
  - Log collection after pod failure
- Automatic reattach volume after node reboot

Problems we encountered

- The driver interface is keep changing
  - Flexvolume, CSI v0.3, CSI v0.4, CSI v1.0
- Finalizers can result in the namespace stuck in `terminating` state
- Informer/Lister cache issue with the Controller Pattern
  - Lister can return stale information even with one node

Upcoming Longhorn v0.6.0 (Beta)

- Re-architecture
  - Engines and replicas would be run as processes inside the DaemonSet Pods
    - Instead of one pod for each engine or replica
- Result
  - Speed up volume attach/detach process
  - No more worry about Pod per node limitation
  - Guaranteed resource for DaemonSet Pods without the risk of scheduling failure

Project Longhorn

LONGHORN

Open Source

Distributed Block Storage Software

For Kubernetes

https://github.com/rancher/longhorn/

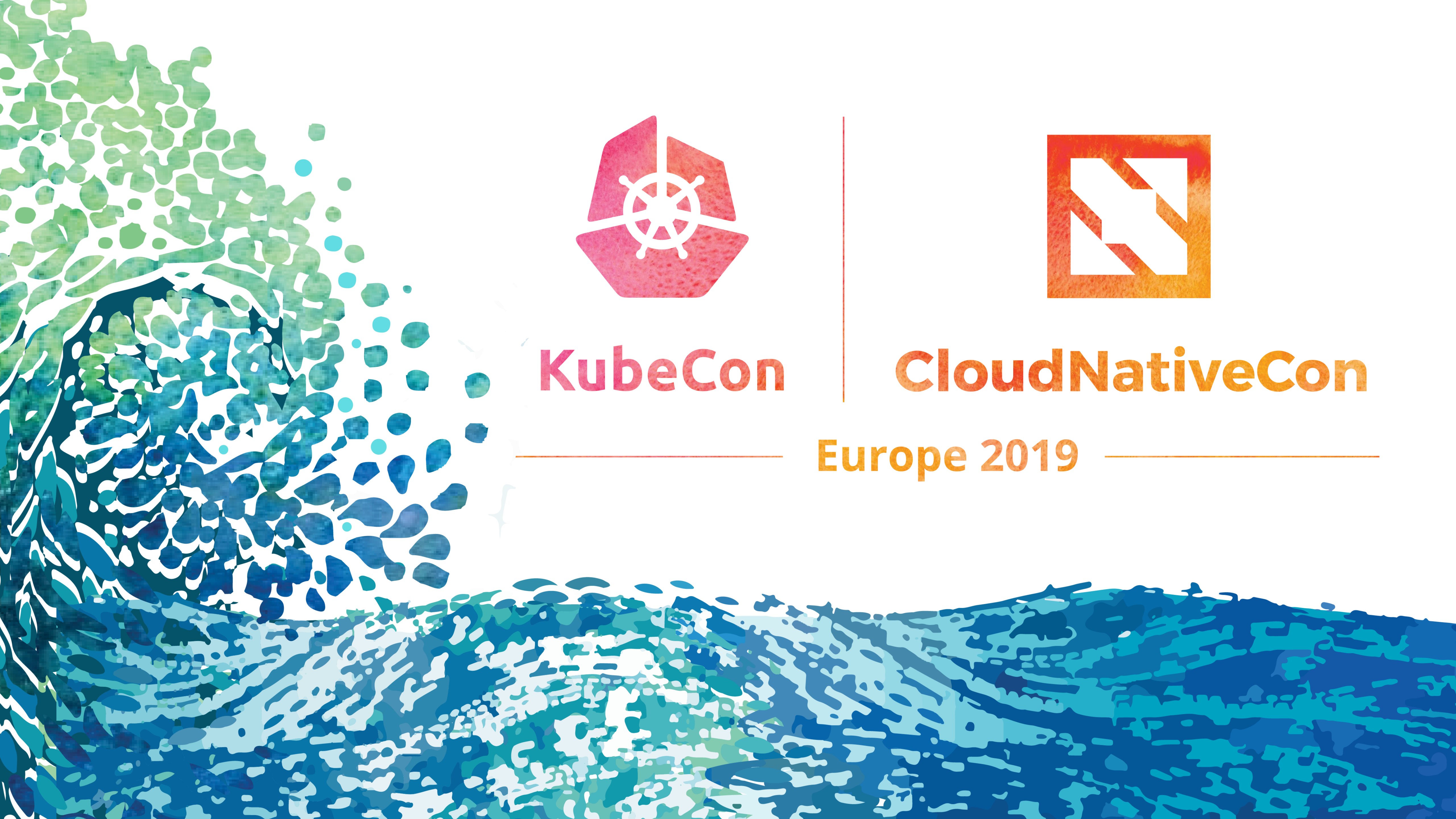
Thank you!

Sheng Yang

Software Architect, Rancher Labs

/ / M : @yasker

sheng.yang@rancher.com



KubeCon | CloudNativeCon

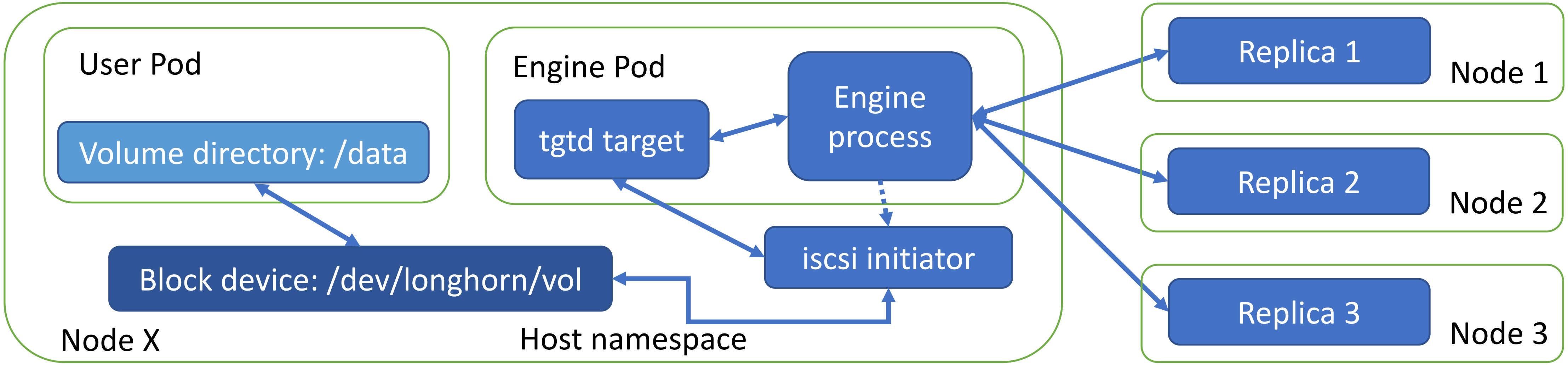
Europe 2019

Workload use RWO volume cannot self-healing if the node is down

- Currently if you want self-healing with Read-Write-Once volume in Kubernetes, you will have a problem
- Stateful Set uses different volumes for each Pod
- But it will not automatically create a new pod if the node of the old pod is down
- Deployment can automatically starts a new pod on a new node if the old pod's node failed
- but it won't detach the volume from the old node, which will result in error for RWO volume since the volume can only be attached to one node

Choice of implementing the block device

- We’ve tried different ways to implement the user-facing block device
  - NBD – Unreliable, easily cause kernel panic
  - TCMU – Kernel patch contributed, require on-going maintaince, not mature enough
  - FUSE – Too slow
- In the end, we choose to use tgtd/iscsi to implement the block device


User Pod
Volume directory: /data
Engine Pod
tgtd target
Engine process
Replica 1
Node 1
Replica 2
Node 2
Replica 3
Node 3
iscsi initiator
Block device: /dev/longhorn/vol
Node X
Host namespace