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
```

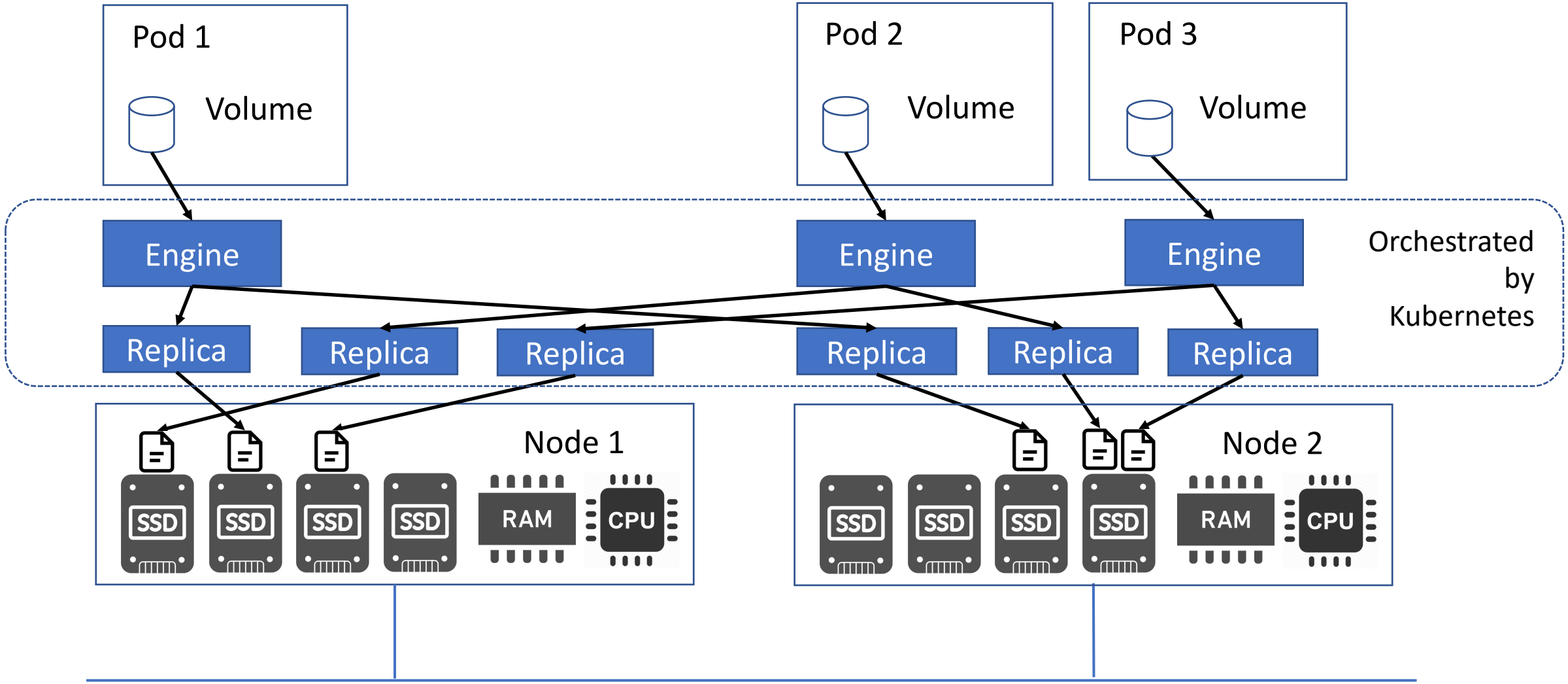
Compare Longhorn to legacy storage software

Legacy Storage Software	Longhorn
Implements sophisticated storage stack that includes disk management, storage virtualization, file systems and storage protocols	Leverages modern high-speed, high-capacity SSD/NVMe and proven Linux storage features (e.g. sparse files, and QoS via cgroups)
HA storage controllers that handle very high IOPS and throughput could become a bottleneck	Each volume is a set of independent microservices, orchestrated by Kubernetes
Complex code base	30k lines of Golang code, broken into clean modules

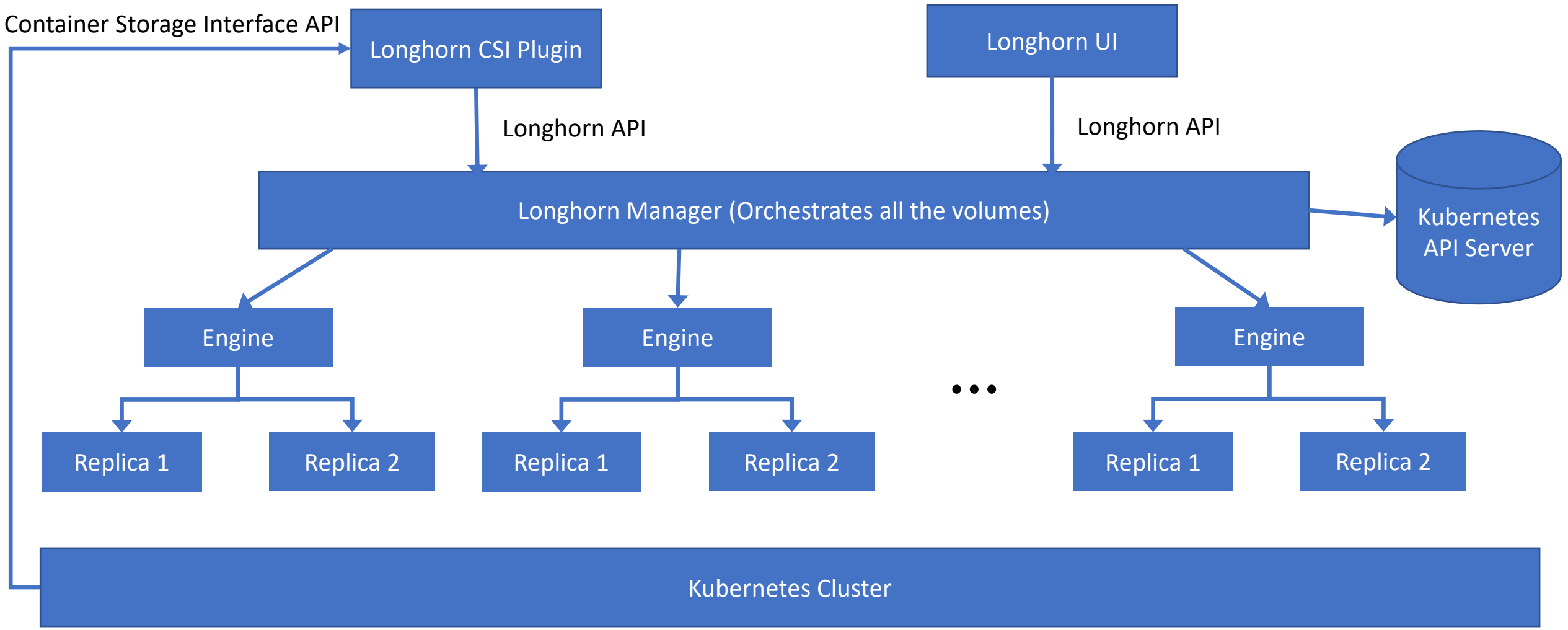
Latest release: Longhorn v0.5.0

- 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, et cetera.

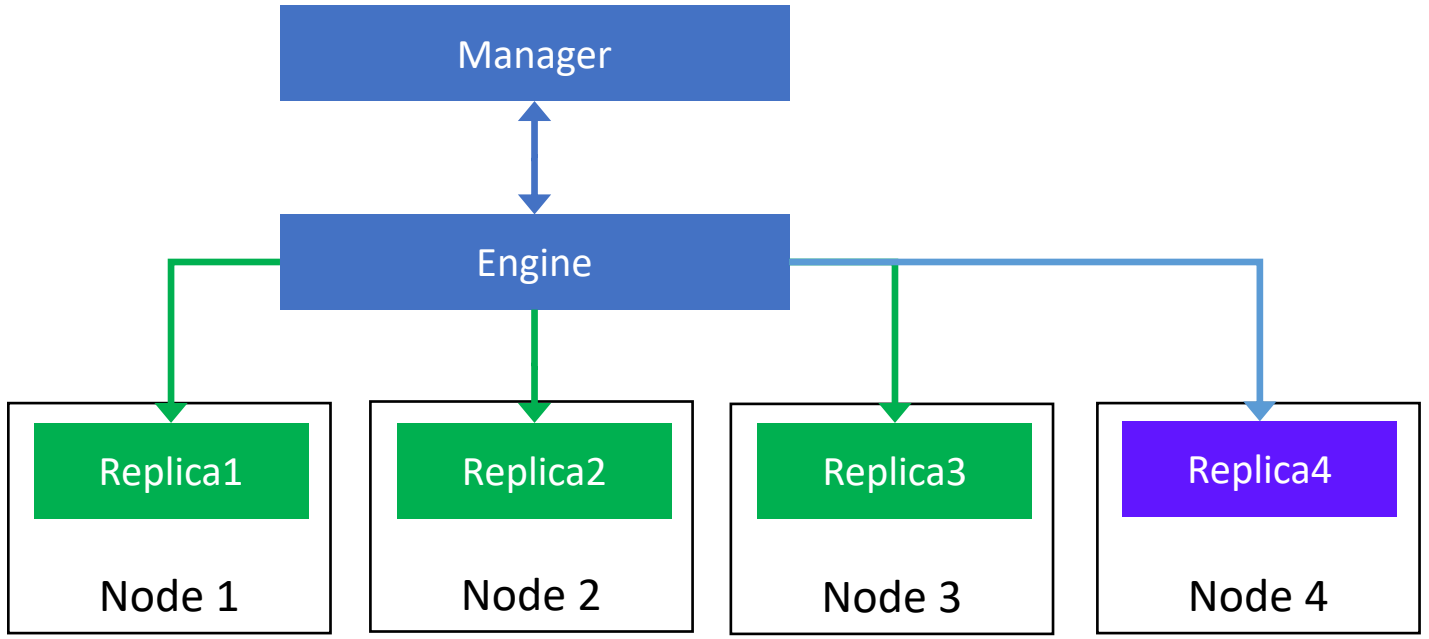
Longhorn Architecture - Engine



Longhorn Architecture - Manager



Cornerstone: Controller Pattern

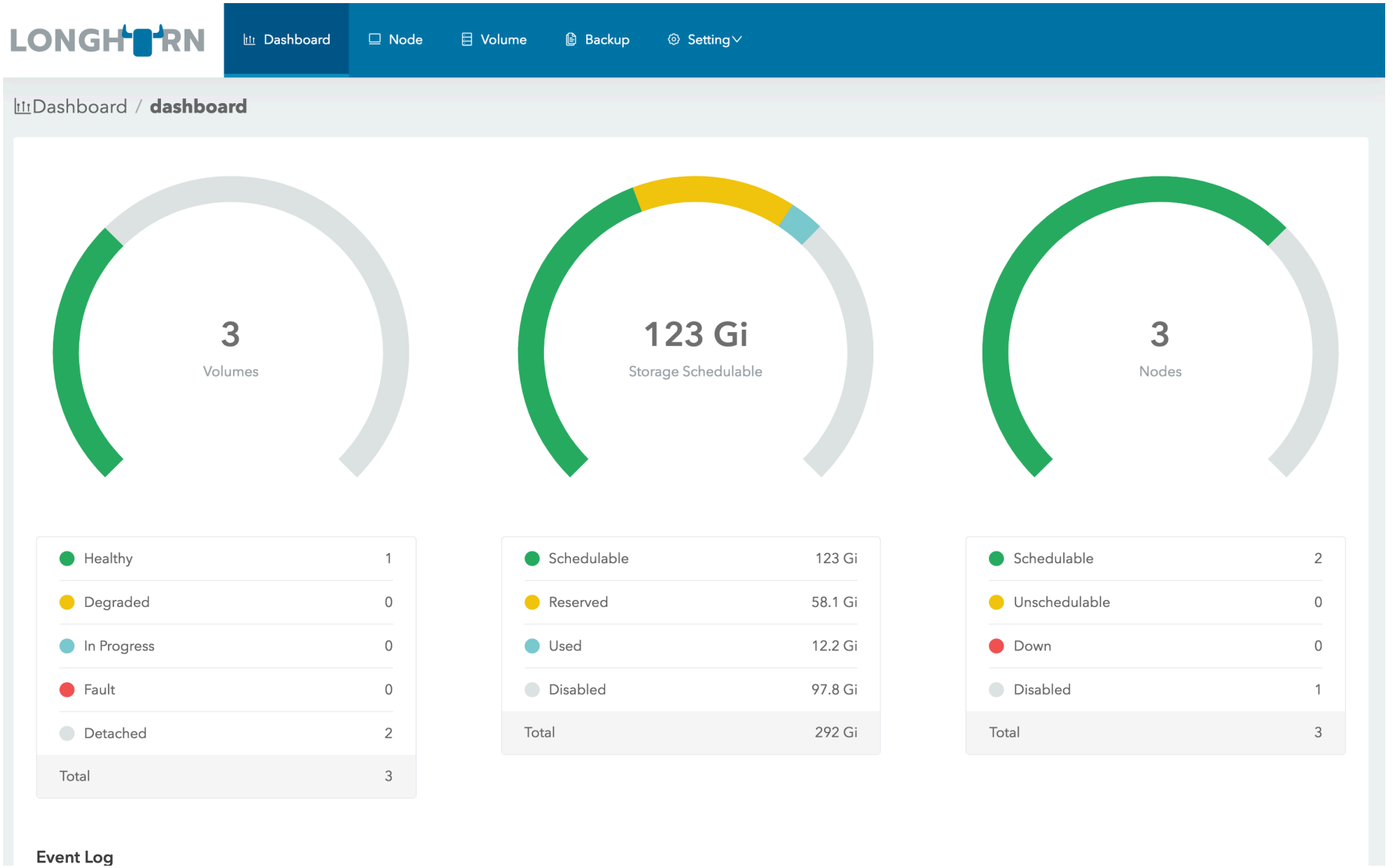



```
volume:
  spec:
    numberOfReplicas: 3
  status:
    currentHealthyReplicas: 3

engine:
  spec:
    replicaList:
      Replica1
      Replica2
      Replica3
  status:
    replicaList:
      Replica1
      Replica2
      Replica3
```

Demo

KubeCon CloudNativeCon Europe 2019



Kubernetes orchestrates the data plane

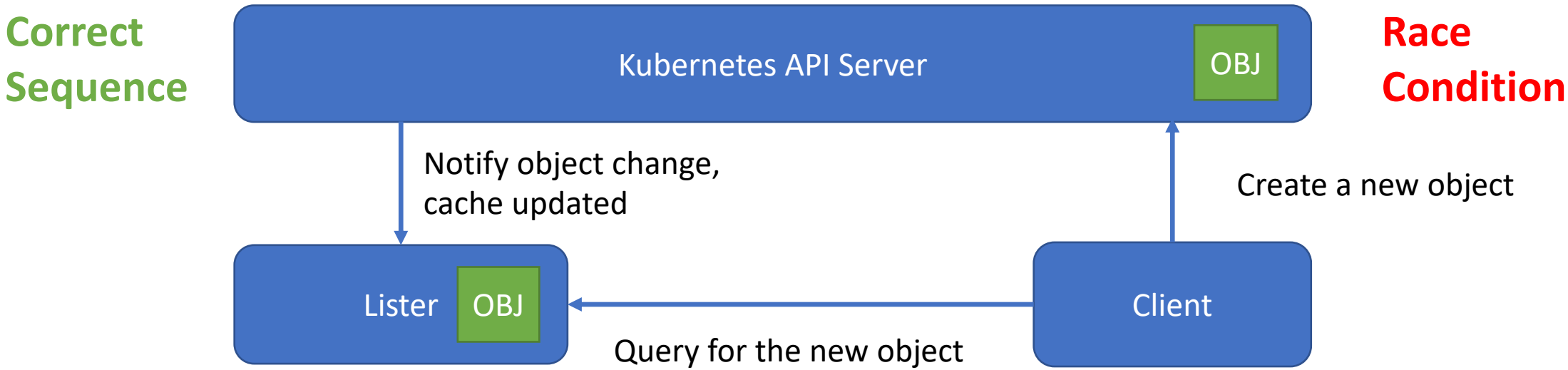
- Engine and replica pods are scheduled to its associated nodes by Kubernetes
- Each replica contains a full copy of the volume
- The raw data can be accessed using the replica directory on the node
  - E.g. `/var/lib/rancher/longhorn/replicas/volname-1234abcd` on node 1
- One correct replica is all that’s need to recover the whole volume

Kubernetes helps to increase resiliency

- Node status monitoring
  - Make it easier to deal with failed/pressured nodes
- Pod status monitoring
  - 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
- Informer/Lister cache race condition
  - 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

Project Longhorn

Open Source

Distributed Block Storage Software

For Kubernetes

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

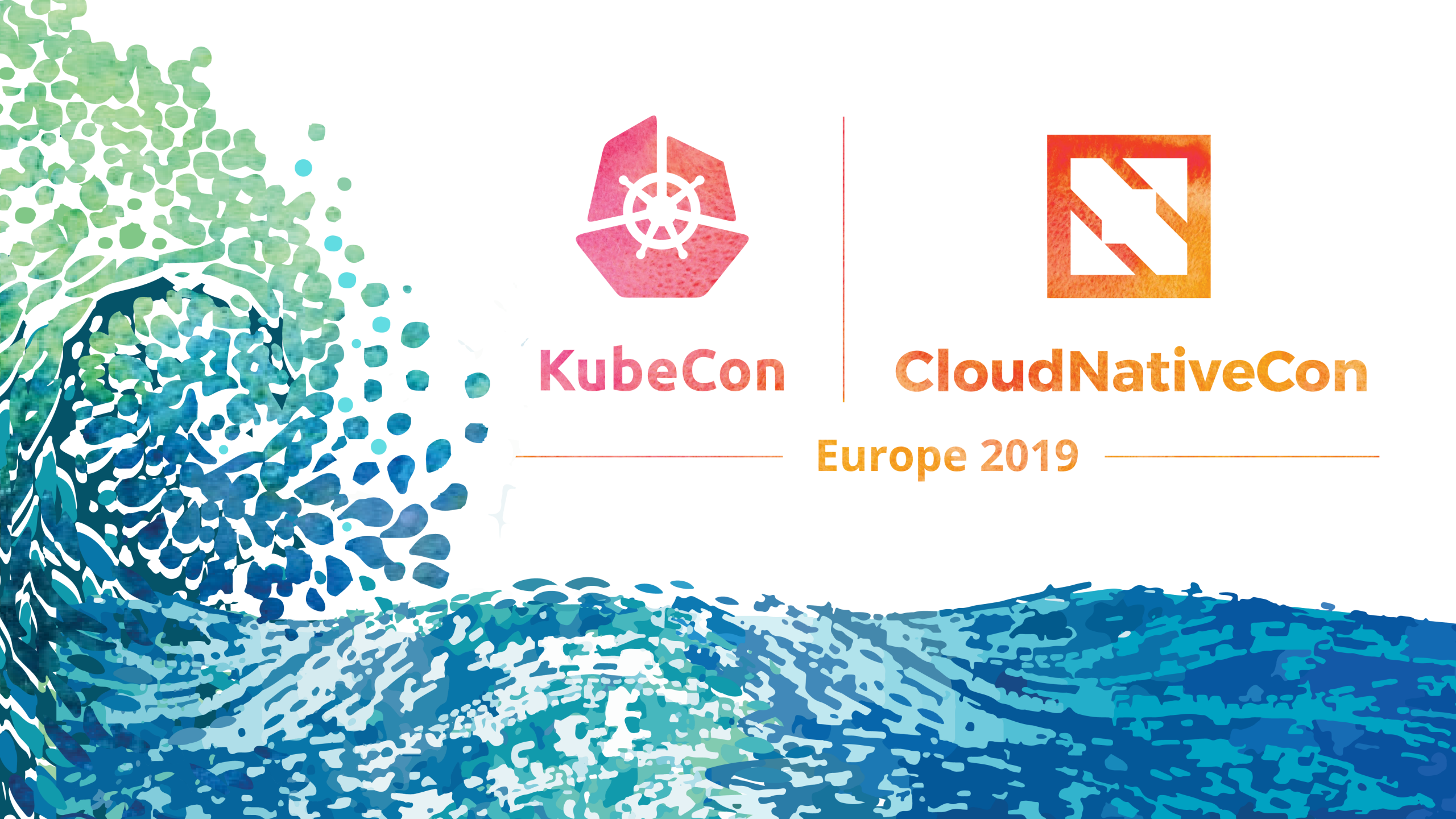
Thank you!

Sheng Yang

Software Architect, Rancher Labs

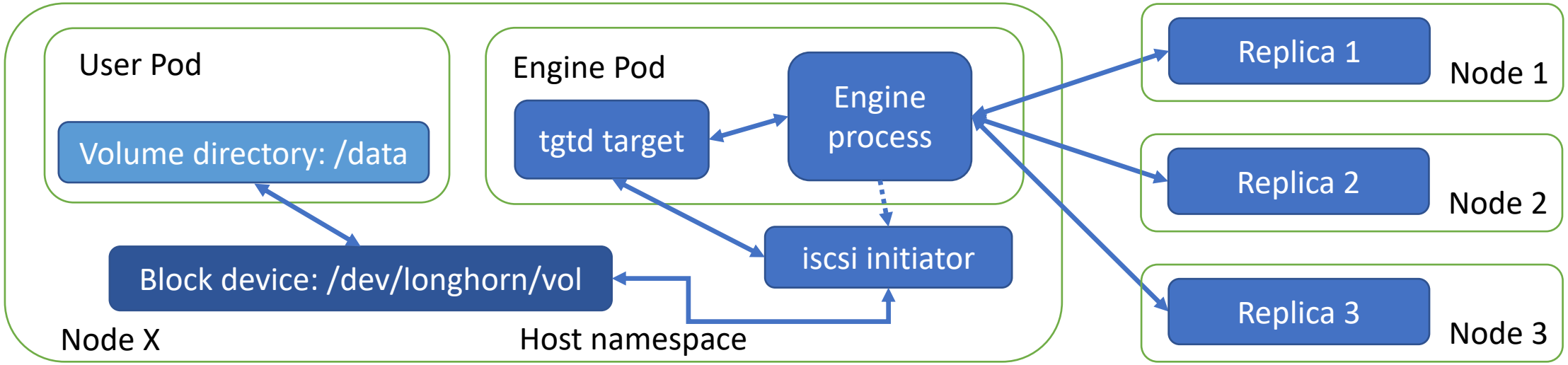
/ / : @yasker

sheng.yang@rancher.com


KubeCon
CloudNativeCon
Europe 2019

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



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