kublr

Achieving True Reliability & Disaster Recovery for Mission Critical Apps

Oleg Chunikhin |CTO

Introductions

Oleg Chunikhin
CTO, Kublr

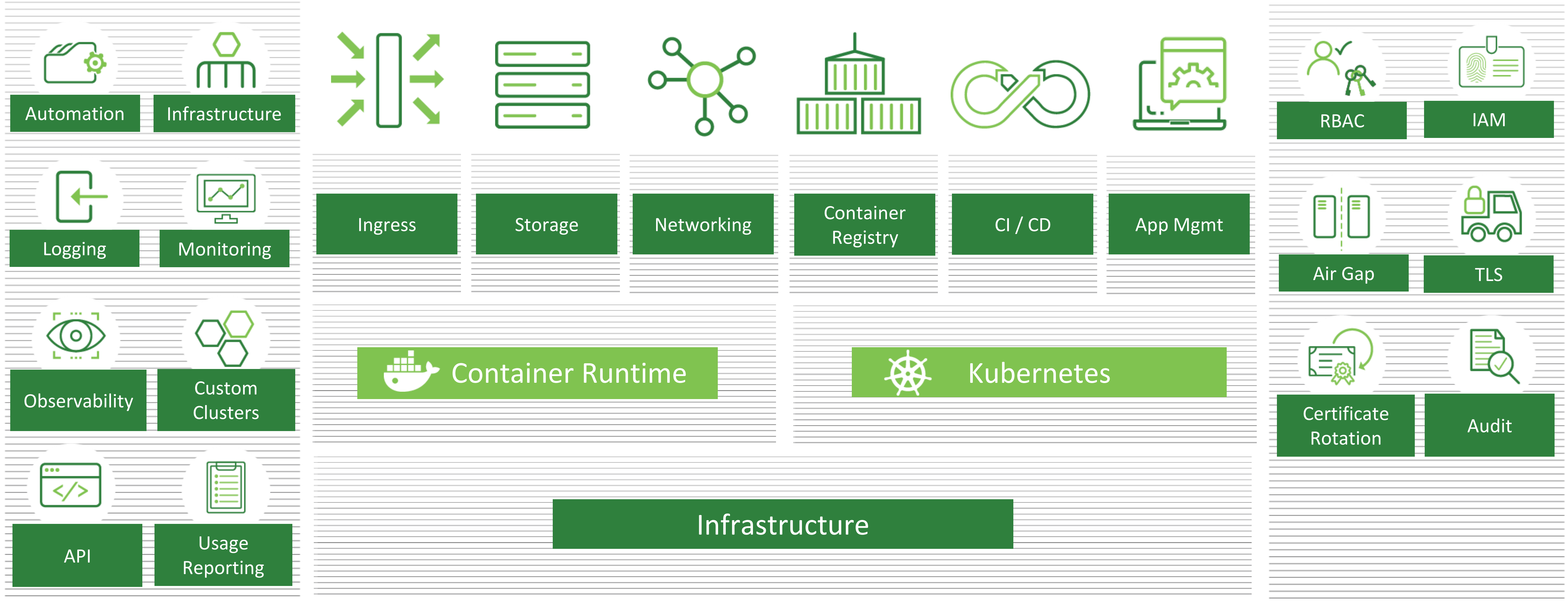
- ✓ 20+ years in software architecture & development
- ✓ Working w/ Kubernetes **since its release** in 2015
- ✓ **CTO at Kublr**—an enterprise ready container management platform
- ✓ Twitter **@olgch; @kublr**

Like what you hear? Tweet at us!

What’s Kublr?

OPERATIONS

SECURITY & GOVERNANCE



Automation

Infrastructure

Logging

Monitoring

Observability

Custom Clusters

API

Usage Reporting

Ingress

Storage

Networking

Container Registry

CI / CD

App Mgmt

Container Runtime

Kubernetes

Infrastructure

RBAC

IAM

Air Gap

TLS

Certificate Rotation

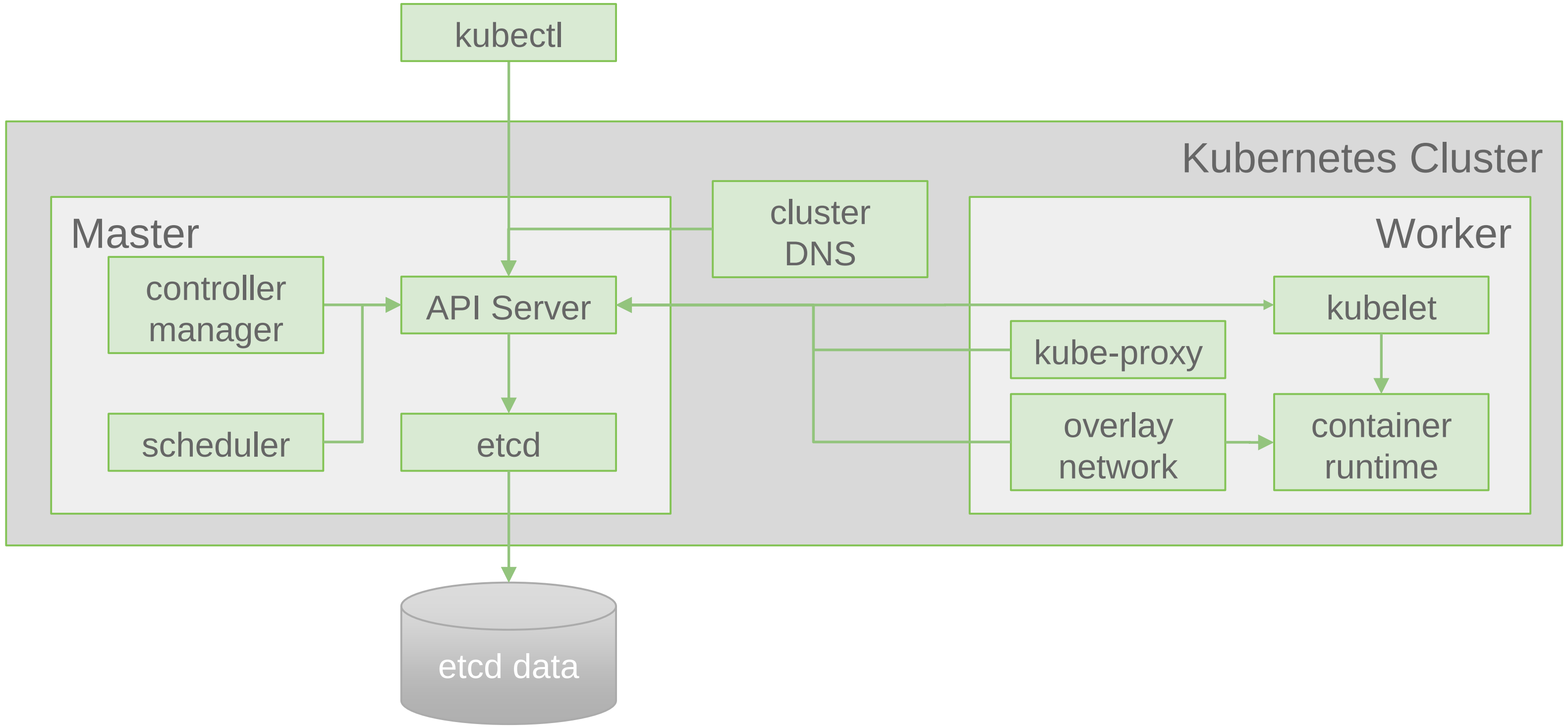
Audit

@olgch, @kublr

Building a Reliable System with Kubernetes

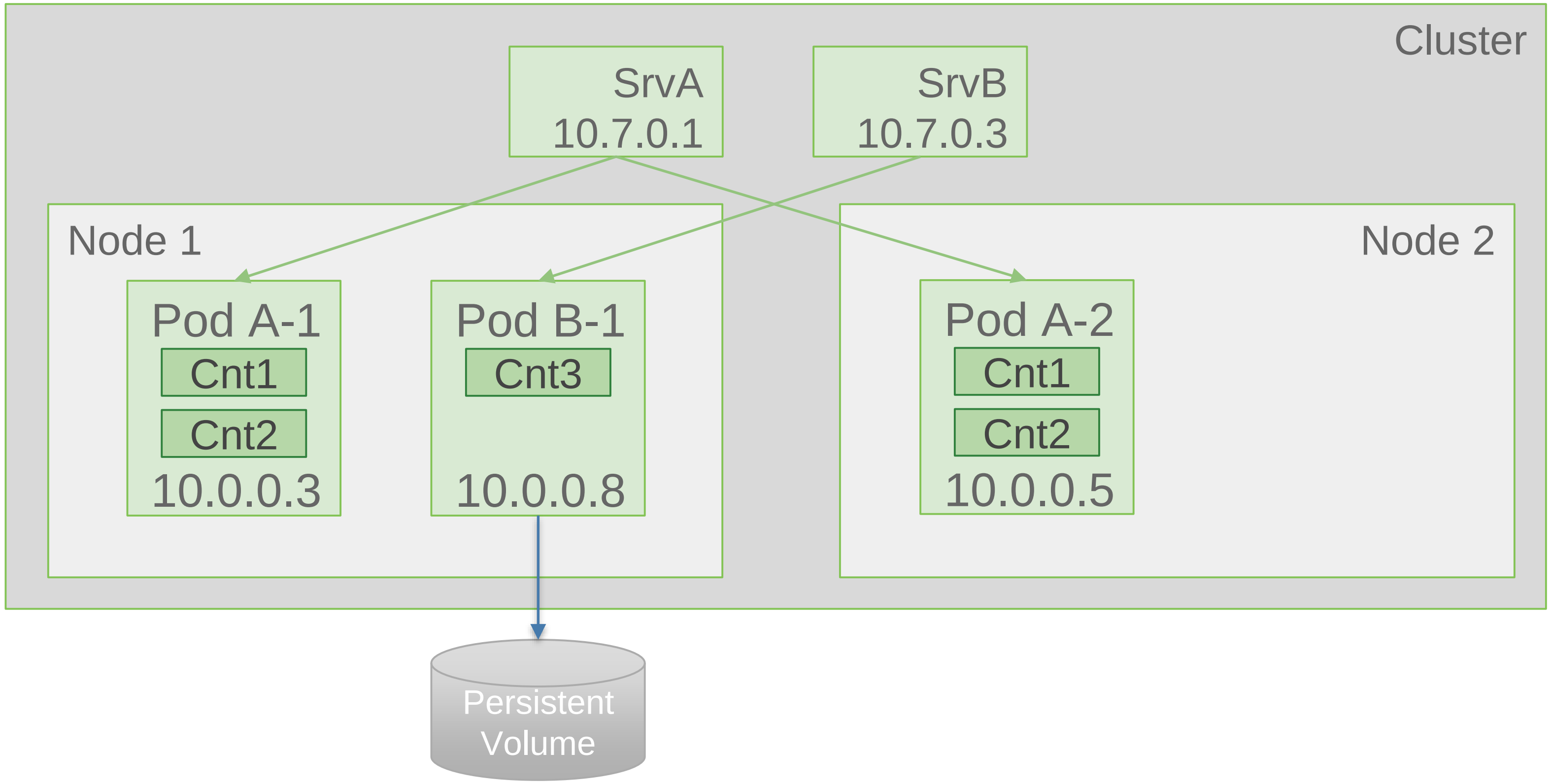
- **Day 1:** trivial, K8S will restart my pod!
- **Day 30:** this is not so simple...
- What Kubernetes does, can, and doesn’t do
- **Full stack reliability:** tools and approaches

K8S Architecture Refresher: Components



The master, agent, etcd, API, overlay network, and DNS

K8S Architecture Refresher: API Objects



Nodes, pods, services, and persistent volumes

K8S Reliability Tools: Probes & Controllers

Pod Probes

Liveness and readiness check

- TCP, HTTP(S), exec

Controllers

ReplicaSet

- Maintain specific number of identical replicas

Deployment

- ReplicaSet + update strategy, rolling update

StatefulSet

- Deployment + replica identity, persistent volume stability

DaemonSet

- Maintain identical replicas on each node (of a specified set)

Operators

K8S Reliability Tools: Resources & Scheduling

- Resource framework
  - Standard: CPU, memory, disk
  - Custom: GPU, FPGA, etc.
- Requests and limits
- Kube and system reservations
  - no swap
- Pod eviction and disruption budget (resource starving)
- Pod priority and preemption (critical pods)
- Affinity, anti-affinity, node selectors & matchers

K8S Reliability Tools: Autoscaling

Horizontal pod autoscaler (HPA)

Vertical pod autoscaler (VPA)

- In-place updates - WIP (issue #5774)

Cluster Autoscaler

- Depends on infrastructure provider - uses node groups
  AWS ASG, Azure ScaleSets, ...
- Supports AWS, Azure, GCE, GKE, Openstack, Alibaba Cloud

Full Stack Reliability

Applications/pods/containers

“Middleware”

- Operations: monitoring, log collection, alerting, etc.
- Lifecycle: CI/CD, SCM, binary repo, etc.
- Container management: registry, scanning, governance, etc.

Container Persistence: cloud native storage, DB, messaging

Container Orchestrator: Kubernetes

- “Essentials”: overlay network, DNS, autoscaler, etc.
- Core: K8S etcd, master, worker components
- Container engine: Docker, CRI-O, etc.

OS: kernel, network, devices, services, etc.

Infrastructure: “raw” compute, network, storage

Architecture 101

- Layers are separate and independent
- Disposable/“restartable” components
- Re-attachable dependencies (including data)
- Persistent state is separate from disposable processes
  Pets vs cattle - only data is allowed to be pets (ideally)

Infrastructure and OS

If a node has a problem...

- Try to fix it (pet)
- Replace or reset it (cattle)

Tools

- In-cluster: npd, weaveworks kured, ...
  - hardware, kernel, servicer, container runtime issues
  - reboot
- Infrastructure provider automation
  - AWS ASG, Azure Scale Set, ...
- External node auto recovery logic
  - Custom + infrastructure provider API
  - Cluster management solution
  - (Future) cluster API

Kubernetes Components: Auto-Recovery

Components

- **etcd**
- **Master:** API server, controller manager, scheduler
- **Worker:** kubelet, kube-proxy
- **Container runtime:** Docker, CRI-O

Already 12 factor

Monitor liveliness, automate restart

- Run as services
- Run as static pods

Dependencies to care about

- etcd data
- K8S keys and certificates
- Configuration

Kubernetes Components: Multi-Master

K8S multi-master

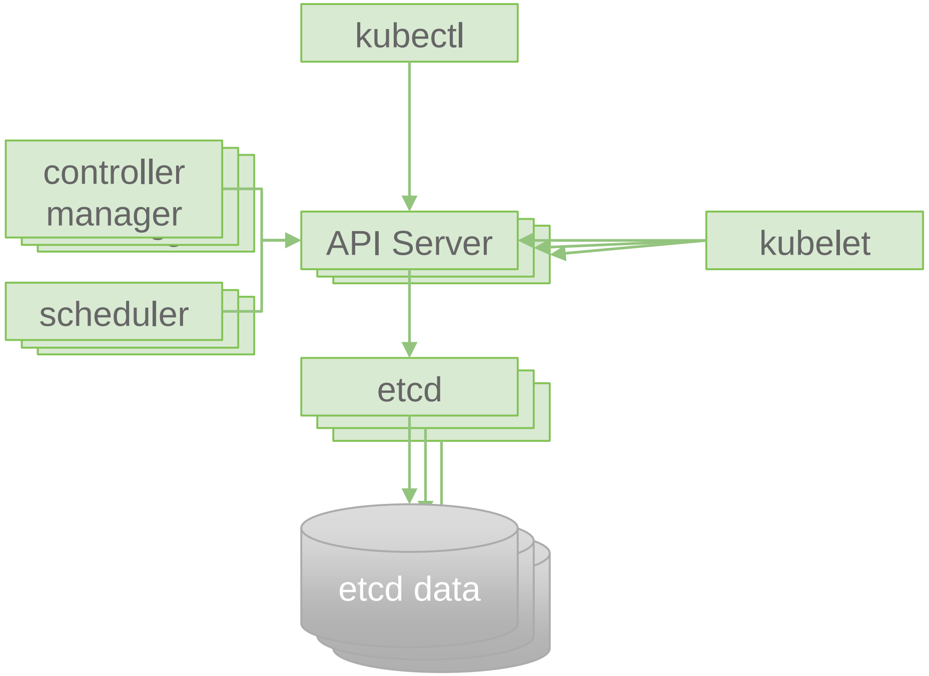
- **Pros**: HA, scaling
- **Cons:** need LB (server or client)

etcd cluster

- **Pros:** HA, data replication
- **Cons**: latency, ops complexity

etcd data

- Local ephemeral
- Local persistent (survives node failure)
- Remote persistent (survives node replacement)



@olgch, @kublr

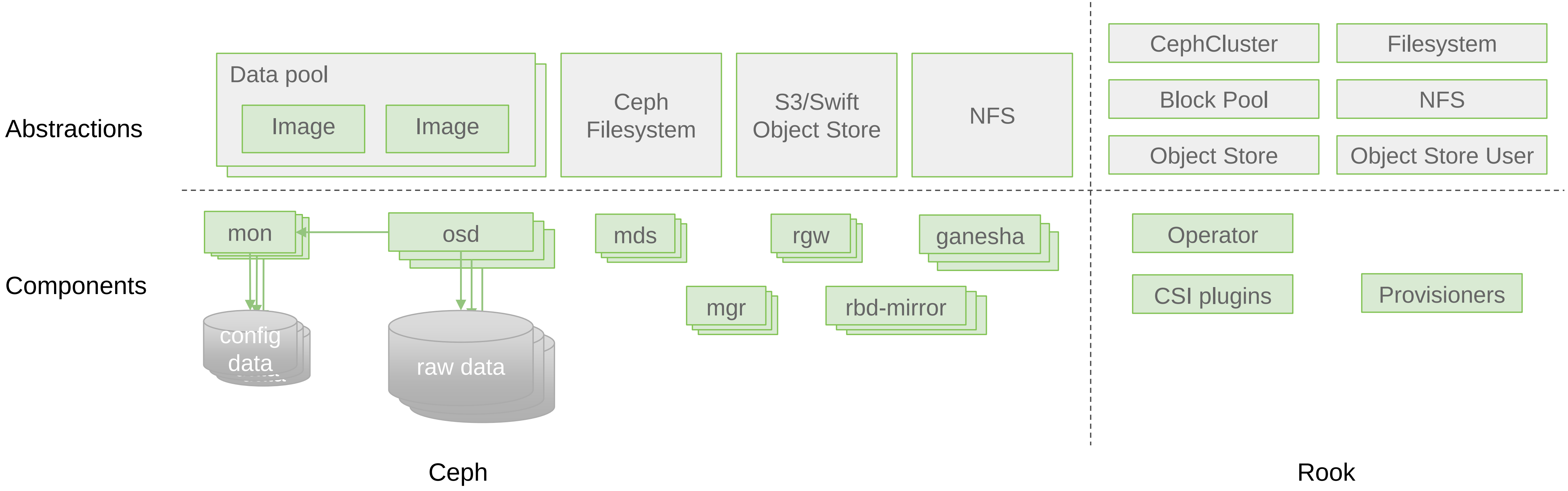
Container Persistence

- **Persistent volumes**
- **Volume provisioning**
- **Storage categories**
  - Native block storage: AWS EBS, Azure Disk, vSphere volume, attached block device, etc.
  - HostPath
  - Managed network storage: NFS, iSCSI, NAS/SAN, AWS EFS, etc.
- **Some of the idiosyncrasies**
  - Topology sensitivity (e.g. AZ-local, host-local)
  - Cloud provider limitations (e.g. number of attached disks)
  - Kubernetes integration (e.g. provisioning and snapshots)

Cloud Native Storage

- **Integrates with Kubernetes**
  - CSI, FlexVolume, or native
  - Volume provisioners
  - Snapshots support
- **Runs in cluster or externally**
- **Approach**
  - Flexible storage on top of backing storage
  - Augmenting and extending backing storage
- **Backing storage:** local, managed, Kubernetes PV
- **Examples:** Rook/Ceph, Portworx, Nuvoloso, GlusterFS, Linstor, OpenEBS, etc.

Cloud Native Storage: Rook/Ceph



@olgch, @kublr

Middleware

Operations: monitoring, log collection, alerting, etc.

Lifecycle: CI/CD, SCM, binary repo, etc.

Container management: registry, scanning, governance, etc.

Deployment options:

- Managed service
- In Kubernetes
- Deploy separately

Something Missing? Multi-Site

- Region to region; cloud to cloud; cloud to on-prem (hybrid)
- One cluster (⚠) vs cluster per location (✓)

Tasks

- Physical network connectivity: VPN, direct
- Overlay network connectivity: Calico BGP peering, native routing, …
- Cross-cluster DNS: CoreDNS
- Cross-cluster deployment: K8S federation
- Cross-cluster ingress, load balancing: K8S federation, DNS, CDN
- Cross-cluster data replication
  - native: e.g. AWS EBS, Snapshots inter-region transfer
  - CNS level: e.g. Ceph geo-replication
  - database level: e.g. Yugabyte geo-replication, sharding, ...
  - application level

To Recap...

- Kubernetes provides robust tools for application reliability
- Underlying infrastructure and Kubernetes components recovery is responsibility of the cluster operator
- Kubernetes is just one of the layers
- Remember *Architecture 101* and assess all layers accordingly
- Middleware, and even CNS, can run in Kubernetes and be treated as regular applications to benefit from K8S capabilities
- Multi-site HA, balancing, failover is much easier with K8S and the cloud native ecosystem. Still requires careful planning!

Q&A

Thank you!

Take Kublr for a test drive!
kublr.com/deploy

Free non-production license

@olgch, @kublr

Oleg Chunikhin
CTO | Kublr
oleg@kublr.com