USER GROUP TALK

Running Kubernetes (and Everything Under It) on Pure Open Source

Chris Reynolds — No vendors. No paywalls. Just upstreams you can *git clone* tonight.

User Group • 2026

whoami

Chris Reynolds

- K8s for 22+ years
- Rancher Government
- Ex DoD / IC contractor
- Kubecost — Global PubSec
- Red Hat — Federal SA/DA
- Penn State HPC

chris@creynold.net

Basics

- Every project I cover today is permissively-licensed open source. Apache-2.0, BSD, MPL, MIT.
- Architecture diagrams come straight from the project's official docs
- Citations on every slide

The Stack at a Glance

Everything below the dotted line is what we'll cover today.

- Pure CNCF and SUSE/Rancher upstream projects
- Each layer has a single-purpose, swappable upstream
- Same diagram appears at the end — fully wired
- Order is bottom-up: bare metal to apps

Apps / Workloads	*your stuff*
GitOps	*Fleet*
Policy	*Kubewarden*
Container Security	*NeuVector*
Multi-Cluster Mgmt	*Rancher*
Virtualization / HCI	*Harvester + KubeVirt*
Distributed Storage	*Longhorn*
Kubernetes Distro	*K3s / RKE2*
Immutable OS	*openSUSE MicroOS / SLE Micro*

Source: docs.k3s.io • docs.rke2.io • longhorn.io • harvesterhci.io

openSUSE MicroOS — Immutable Linux for K8s

- Read-only root filesystem. Updates never touch the running system.
- Btrfs snapshots + transactional-update wrap zypper — apply on next boot.
- Atomic rollback: if a snapshot boots bad, revert to the last good one.
- Designed for containers and K8s hosts: minimal package set, no GUI.
- Same lineage powers SLE Micro (enterprise) and Harvester's node OS.

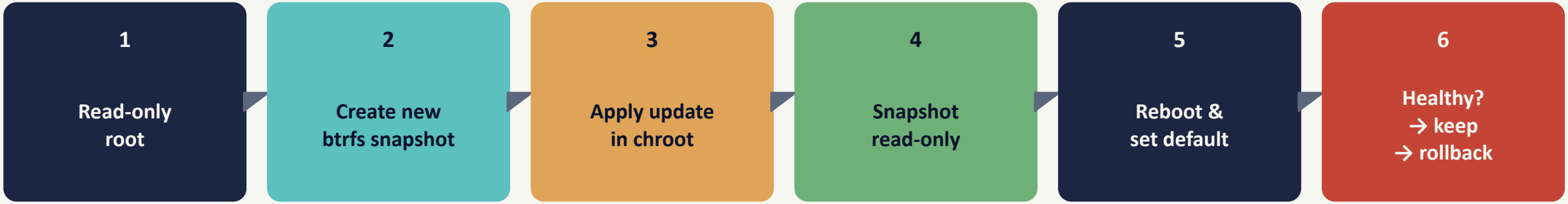
0

packages installed at runtime

snapshot →
update →
reboot →
commit OR rollback

Source: en.opensuse.org/Portal:MicroOS • github.com/openSUSE/transactional-update

MicroOS — Transactional Update Flow



What that buys you

Tamper resistance

Root is read-only at runtime. Malware can't persist a binary in /usr — it has no write path.

Automated rollback

Health-check hook fails after reboot → systemd flips the default snapshot back. Zero touch.

Repeatable images

Same bootable image on every host. Drift goes to zero. Easy to attest in an ATO package.

Source: en.opensuse.org/Portal:MicroOS • github.com/openSUSE/transactional-update

When (and Why) to Reach for MicroOS

GOOD FIT

K8s worker/control plane nodes • Edge compute • Air-gapped fleets • Anything you reflash more than you patch • Anywhere drift kills you

BAD FIT

Workstations that need GUI apps installed ad-hoc • Servers running long-lived legacy non-containerized workloads • Anything where you SSH in to install packages 'just for now'

How it stacks up

Project	License	Update model	Container runtime
openSUSE MicroOS	GPL/various	transactional-update + btrfs	podman / containerd
Fedora CoreOS / Flatcar	Apache-2.0	ostree-based image swap	containerd
Talos Linux	MPL-2.0	API-driven, no shell	containerd (K8s only)
Bottlerocket (AWS)	Apache/MIT	image swap via update agent	containerd

Source: get.opensuse.org/microos • flatcar.org • talos.dev

K3s — Lightweight Kubernetes

- Single ~75 MB binary. One file is the whole distro.
- Datastore options: embedded SQLite (single-node), embedded etcd (HA), or external (Postgres / MySQL / etcd).
- Embedded batteries: containerd, Flannel CNI, CoreDNS, Traefik (replacing servicelb/klipper), local-path-provisioner, Helm Controller.
- CNCF Sandbox project (Aug 2020). Now hosted by The Linux Foundation as K3s.
- Same Kubernetes API as upstream — kubectl, Helm, manifests all just work.

~75 MB

single static binary

512 MB

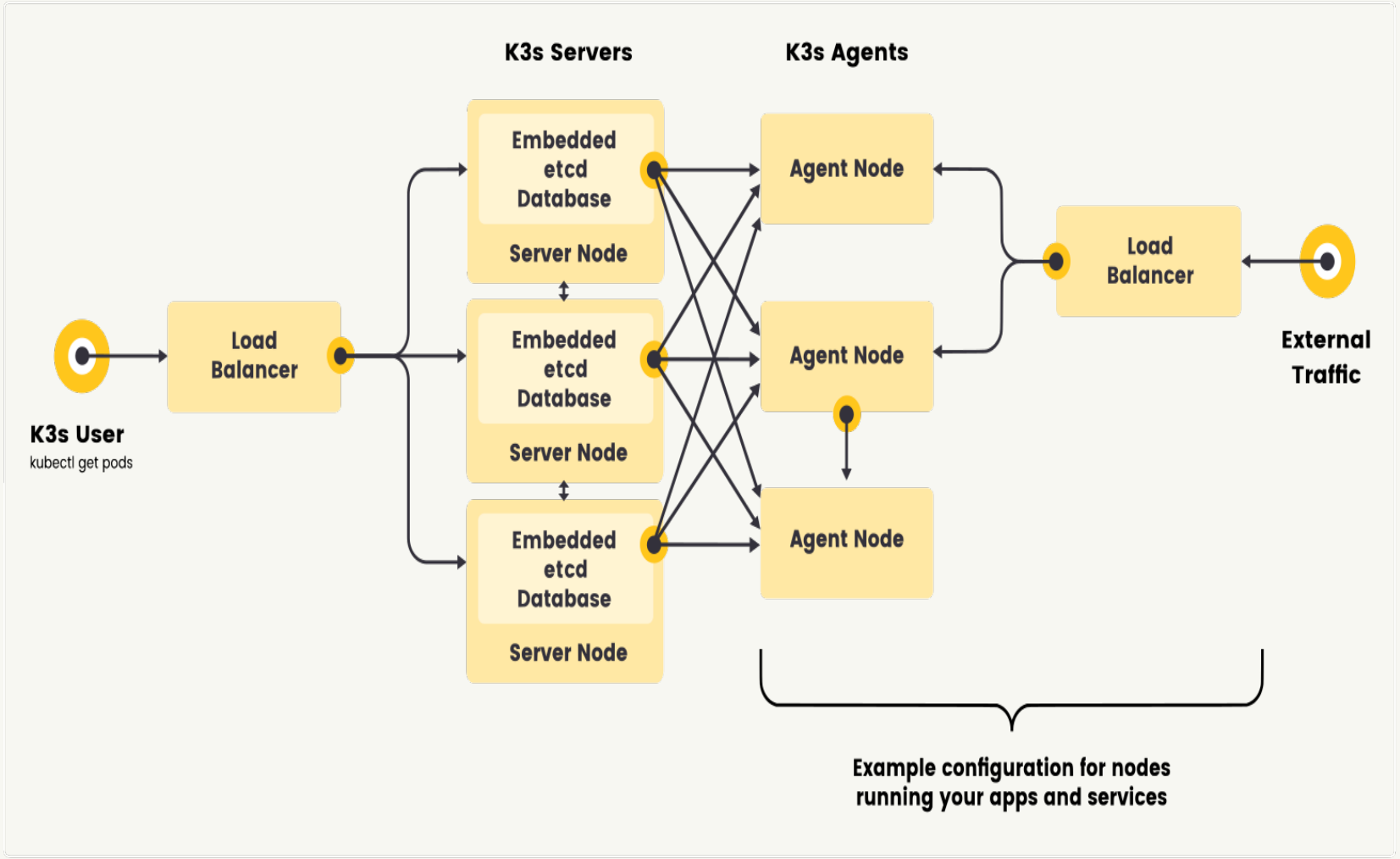
min RAM (server), can run ARM

v1.36.x

current K8s version (May 2026)

Source: docs.k3s.io • github.com/k3s-io/k3s

K3s Architecture — HA with Embedded etcd



Source: docs.k3s.io/architecture

Components per node

k3s server

kube-apiserver • controller-manager • scheduler • kubelet • containerd • CNI • embedded etcd (HA) or SQLite

k3s agent

kubelet • containerd • CNI • client-side load-balancer to all servers • node-password registration

K3s — Real-World Use Cases

EDGE & IOT

Retail kiosks, sensors, ruggedized field kits. Boots in seconds, sips RAM, survives flaky links. The K3s 'hello world' is literally a Raspberry Pi.

CI / DEV

Spin up real Kubernetes per PR. K3s in Docker (k3d) gives you a multi-node cluster in 10 seconds for integration tests.

EMBEDDED

Ship K8s inside an appliance. K3s is the K8s engine inside Rancher Desktop, k3d, k3os, and the upstream of RKE2.

Known production deployments (publicly cited)

- Chick-fil-A — ~3 nodes per restaurant store, thousands of stores nationwide
- Roblox, Apple, GitLab — internal CI fleets
- US DoD — runs in the upstream of RKE2, deployed on PlatformOne / Big Bang stacks
- Cisco IoT — built into industrial edge appliances

Source: k3s.io • github.com/k3s-io/k3s/blob/master/ADOPTERS.md

RKE2 — K3s, Grown Up for Regulated Workloads

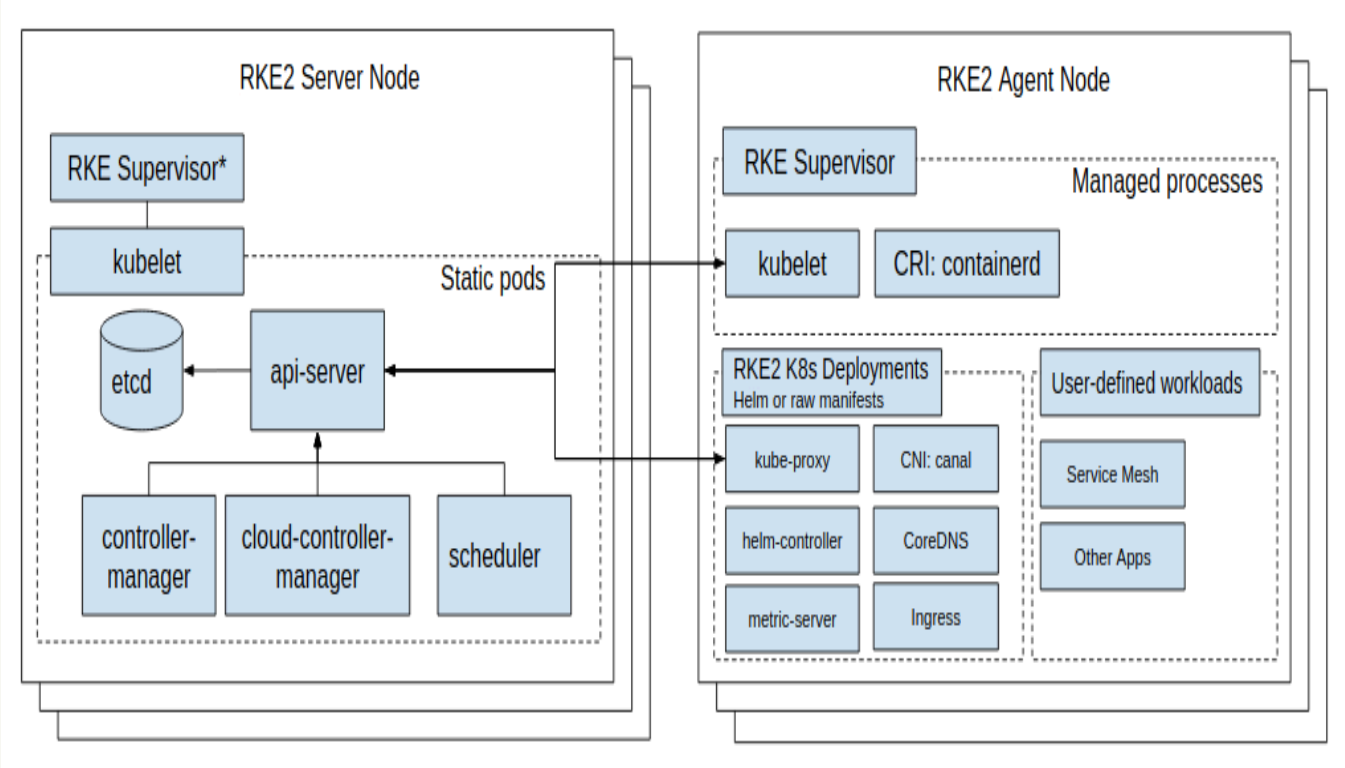
- Built on the K3s engine — same single-binary deploy, same simplicity.
- Control-plane components run as STATIC PODS (not host processes) — kube-apiserver, etcd, scheduler, controller-manager.
- No Docker. containerd + runc only. Smaller attack surface.
- Compiled with Go + BoringCrypto for FIPS-validated cryptographic modules.
- CIS Kubernetes Benchmark hardened by default. SELinux supported and enforced.
- CNI choice baked in: Canal (Flannel+Calico), Calico, Cilium, or Flannel. Multus for multi-NIC.

WHY DOD LIKES IT

- FIPS-140 capable build in paid version
- CIS Benchmark by default
- Static-pod control plane
- SELinux enforcing
- No Docker, no socket
- Air-gap friendly
- Same upstream K8s API

Source: docs.rke2.io • github.com/rancher/rke2

RKE2 Architecture — What's in the Single Binary



Bundled upstreams

- kube-apiserver • controller-manager • scheduler • kubelet • kube-proxy
- etcd • containerd • runc
- Canal / Cilium / Calico / Flannel + Multus
- CoreDNS
- Ingress NGINX / Traefik
- Metrics Server • Helm Controller
- Snapshot Controller + validation webhook
- All compiled with Go+BoringCrypto

RKE2 = single binary → spawns containerd → loads kubelet → kubelet brings up static-pod control plane

Source: docs.rke2.io/architecture

RKE2 — Security Posture Out of the Box

CIS BENCHMARK

Hardened defaults: anonymous-auth disabled, audit logs on, secure cipher suites, restricted kubelet, encrypted secrets at rest, PSA baseline/restricted.

FIPS 140

Go + BoringCrypto build available as a first-class CI target. Cryptographic modules certified under NIST CMVP. Same source, different compile flag.

SELINUX

Native support on RHEL/SLE family. Container processes confined. RKE2-specific policy ships in container-selinux package.

Enabling the profile is one line

```
# /etc/rancher/rke2/config.yaml
profile: cis
selinux: true
kube-apiserver-arg:
  - 'audit-log-path=/var/lib/rancher/rke2/server/logs/audit.log'
  - 'audit-log-maxage=30'
```

Source: docs.rke2.io/security • csrc.nist.gov/projects/cryptographic-module-validation-program

Rancher — Multi-Cluster K8s Control Plane

- Apache-2.0 open-source project at github.com/rancher/rancher — yes, the whole thing.
- Itself a Kubernetes app — runs as pods inside a 'local' cluster.
- Manages downstream clusters via an agent-based reverse-tunnel model.
- Provides: unified UI, RBAC + auth provider proxy (AD/LDAP/SAML/OIDC), Helm catalog, monitoring (Prometheus/Grafana), logging, backup-restore-operator.
- Works with ANY conformant K8s — RKE2, K3s, EKS, AKS, GKE, kubeadm

1000s

downstream clusters per Rancher

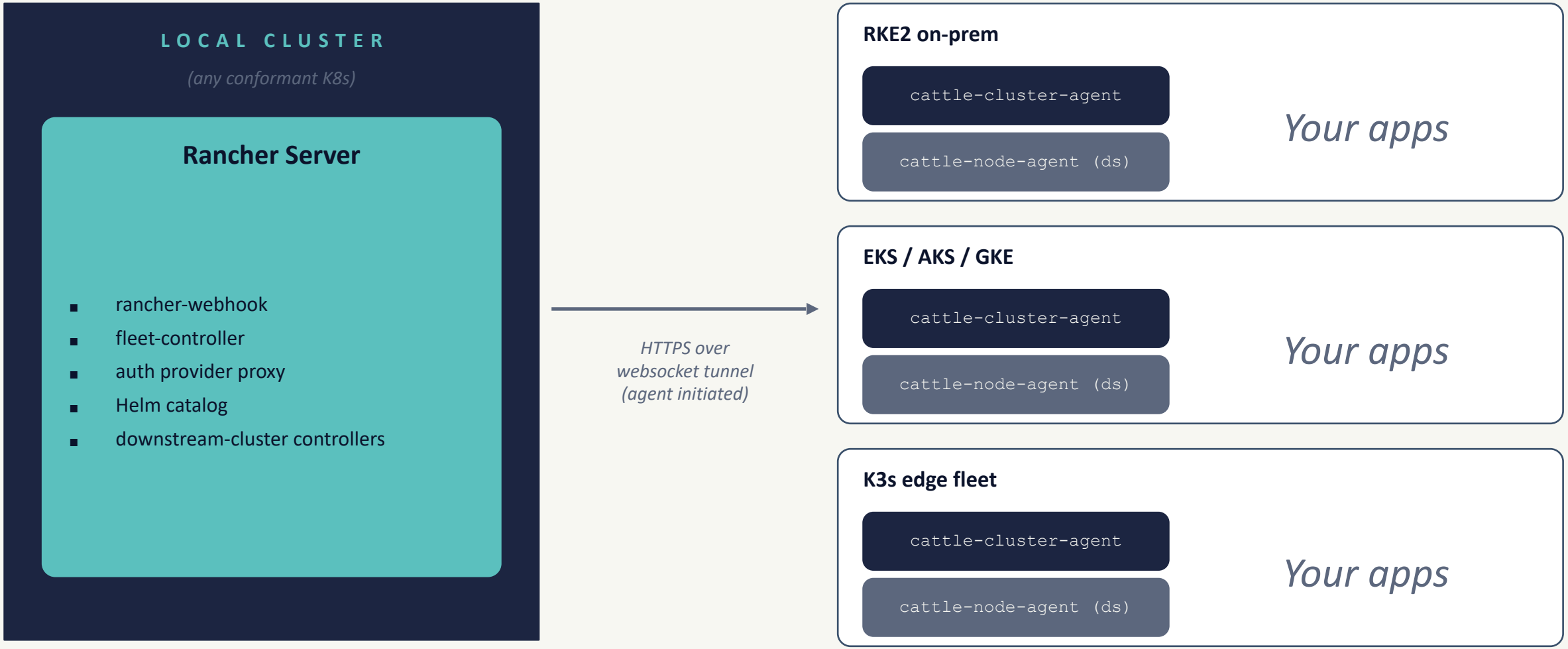
APACHE-2.0

Including the UI.
Including the catalog.
Including the multi-cluster controllers.

`github.com/rancher/rancher`

Source: github.com/rancher/rancher • ranchermanager.docs.rancher.com

Rancher Architecture — Agent + Tunnel



Source: ranchermanager.docs.rancher.com/reference-guides/rancher-manager-architecture

What Rancher Gives You Without Paying Anyone

Single-pane UI

Cluster list, node view, workload view, exec into pod from browser — works for every downstream cluster.

Auth provider proxy

Plug in AD, LDAP, SAML, OIDC, Okta, Azure AD. Rancher maps groups to RBAC across clusters.

App catalog (Helm)

Curated chart repos + your own. Same install UX for every cluster.

rancher-monitoring

Prometheus + Grafana + Alertmanager packaged as a chart, with built-in K8s dashboards.

rancher-logging

Fluentd/Fluent Bit operator. Ship logs to ES, Loki, Splunk, CloudWatch, S3.

rancher-backup

CRD-driven cluster-state backup operator. Stores to S3 or any PVC.

CIS scan & STIG scan

Run kube-bench and CIS profile scans against any cluster, on demand or on a schedule.

Fleet (built in)

GitOps engine for the multi-cluster fleet — covered in Section 8.

Source: ranchermanager.docs.rancher.com/integrations-in-rancher • github.com/rancher/charts

Longhorn — Cloud-Native Distributed Block Storage

- CNCF Incubating project (Nov 2021). Originally donated by Rancher Labs in Oct 2019.
- Distributed block storage built FOR Kubernetes, deployed AS Kubernetes — manager runs as a DaemonSet.
- Each volume has a dedicated controller (the Engine) → no shared blast radius.
- Synchronous replication across nodes (configurable replica count, default 3).
- Snapshots, incremental backups to S3 / NFS, DR volumes, ReadWriteMany via NFSv4 share-manager.
- V2 data engine uses SPDK for NVMe-class performance (still beta in 1.10).

CNCF

Incubating since 2021

3x

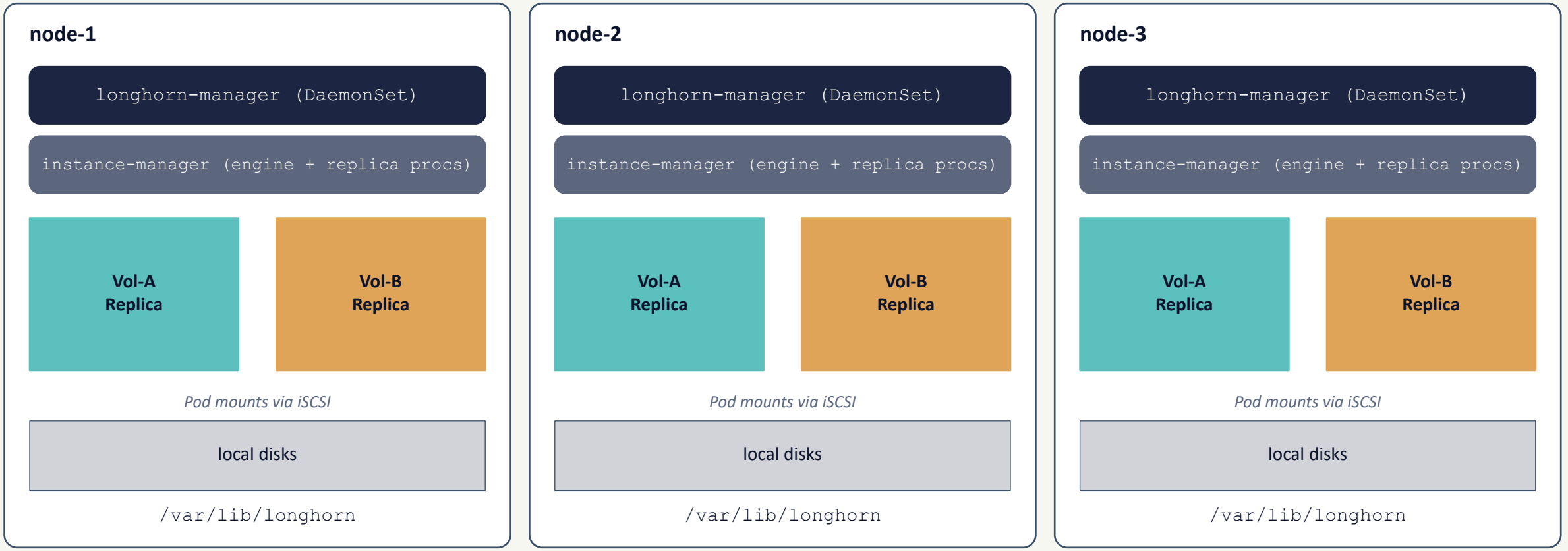
default replica count

RWO / RWX / ROX

via iSCSI (V1) or SPDK (V2)
+ NFSv4 share-manager for RWX

Source: longhorn.io • github.com/longhorn/longhorn • cncf.io/projects/longhorn

Longhorn Architecture — Engine per Volume



Each volume = 1 engine + N replicas spread across nodes. Synchronous writes; failed replica? Replace it, no app downtime.

Source: longhorn.io/docs/1.10/concepts

Harvester — Open-Source HCI Built on Kubernetes

- Bootable HCI appliance — ISO installs OS + K8s + HCI stack on bare metal.
- Built entirely from open-source upstreams: SLE Micro + RKE2 + KubeVirt + Longhorn + Multus + Grafana/Prometheus.
- VMs ARE Kubernetes resources (VirtualMachine, VirtualMachineInstance). kubectl them.
- Live migration, snapshots, backups, cloud-init, PCI passthrough, GPU passthrough, vGPU.
- Integrates with Rancher — manage VMs and containers side-by-side in one UI.
- Apache-2.0. Yes — including the UI.

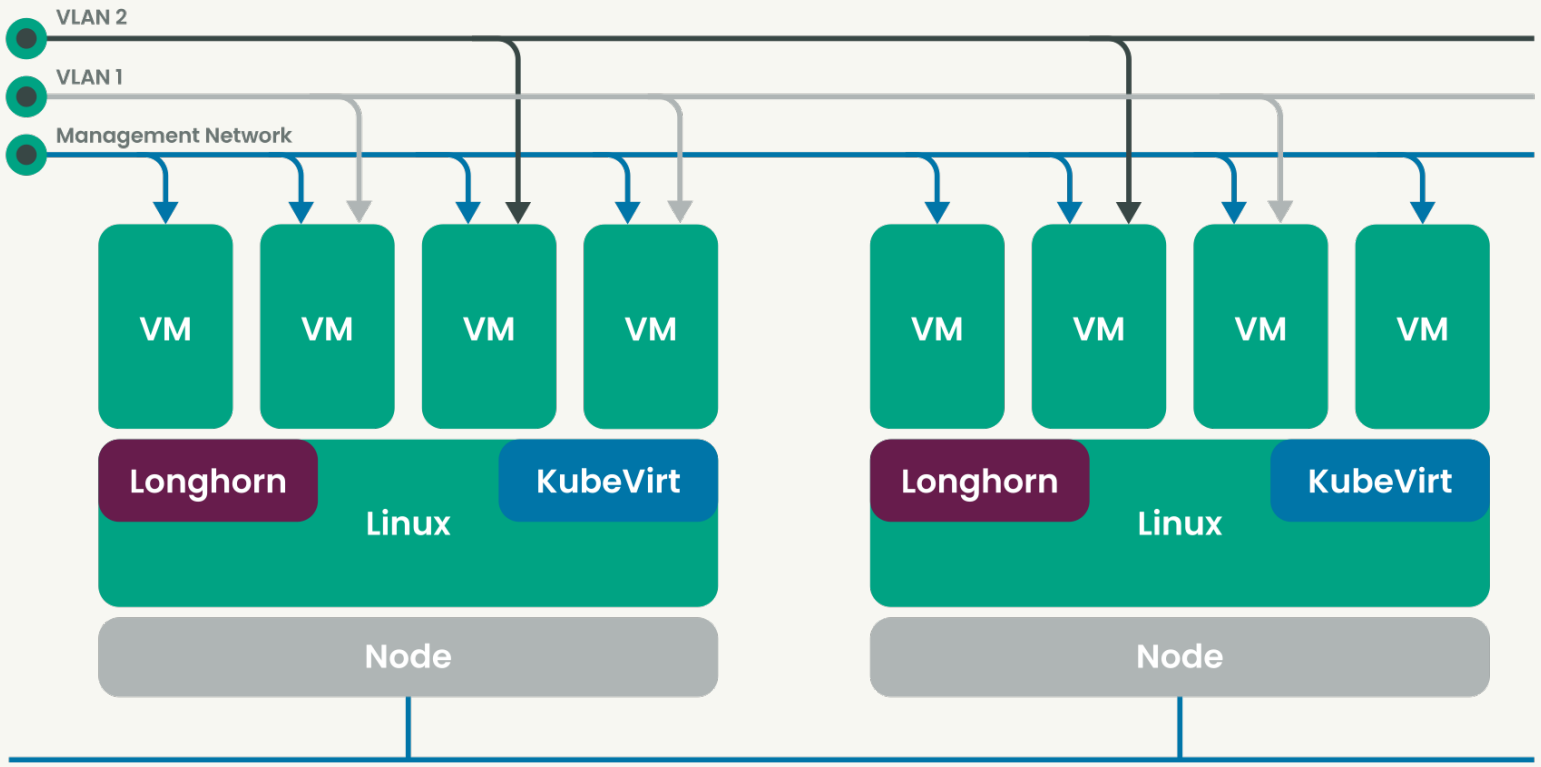
WHAT IT REPLACES

- VMware vSphere + vSAN
- Nutanix AHV + AOS
- Proxmox + Ceph
- OpenStack (the easy parts)

And keeps your VMs while you also schedule containers on the same nodes.

Source: harvesterhci.io • github.com/harvester/harvester • kubevirt.io

Harvester Architecture — Stacked Upstreams



The full stack

- Linux: SUSE Linux Micro 6.2 (Elemental-managed)
- K8s: RKE2
- VM: KubeVirt (KVM)
- Storage: Longhorn
- Network: Multus + Canal
- Observability: Grafana + Prometheus
- UI: Harvester Dashboard

Diagram from harvesterhci.io official docs (v1.8). Every block in the diagram is a project in this talk.

Source: docs.harvesterhci.io/v1.8

VM and Container Parity in One API

Same kubectl. Same RBAC. Same GitOps. Different workload.

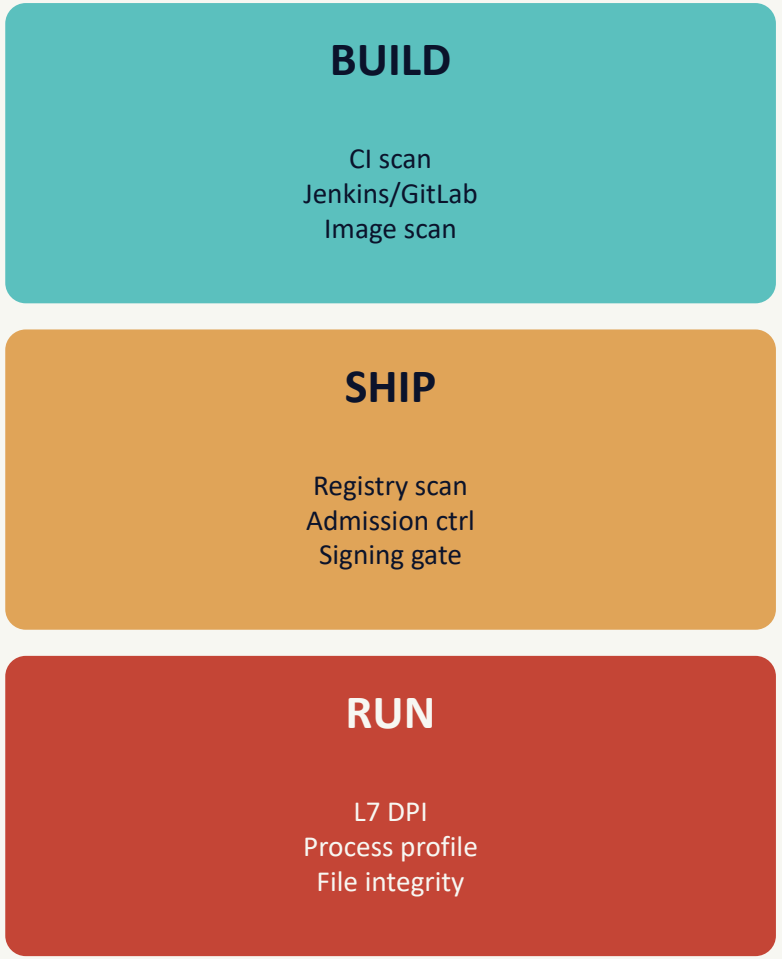
```yaml
# A container
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis:7
    ports:
    - containerPort: 6379
```

```yaml
# A virtual machine
apiVersion: kubevirt.io/v1
kind: VirtualMachine
metadata:
  name: ubuntu-jammy
spec:
  template:
    spec:
      domain:
        cpu: { cores: 2 }
        memory:
          guest: 4Gi
```

Source: kubevirt.io • harvesterhci.io/v1.8/vm/virtual-machines

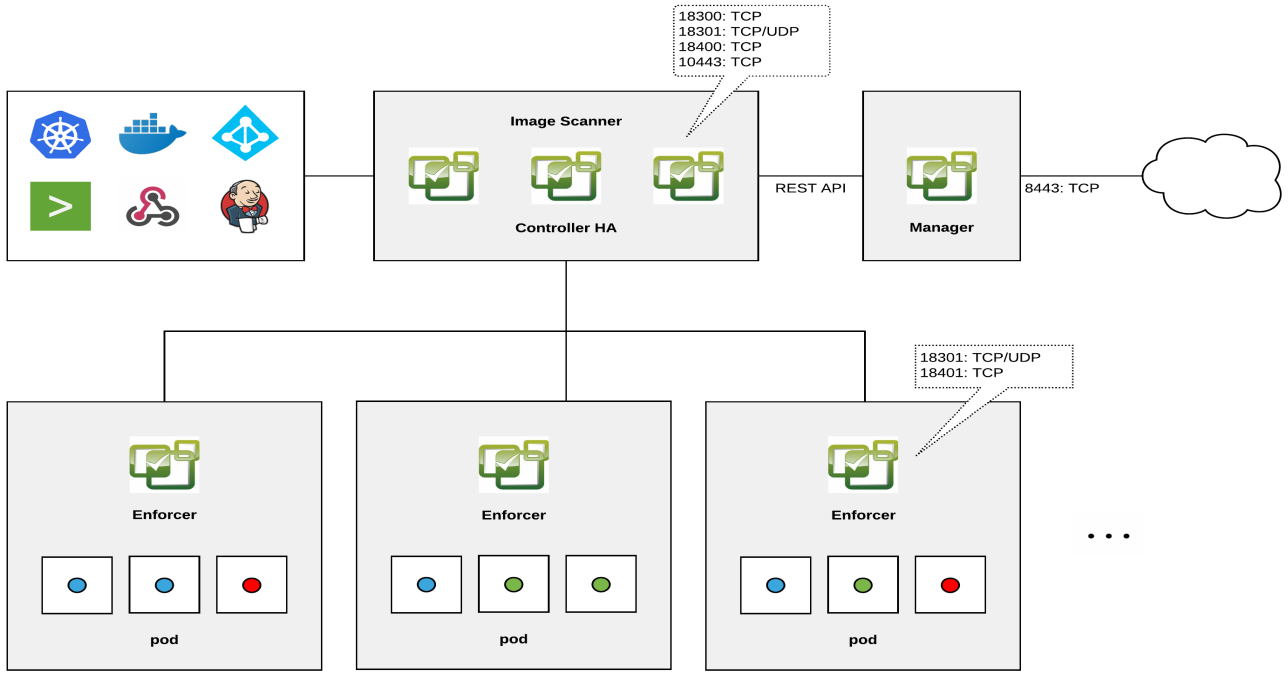
NeuVector — Full-Lifecycle Container Security

- Open-sourced January 2022 (Apache-2.0). Code at github.com/neuvector/neuvector.
- Covers BUILD (CVE scan in CI) → SHIP (registry scan, admission control) → RUN (zero-trust runtime).
- The unique sauce: L7 deep packet inspection in user space — sees actual HTTP, gRPC, Kafka traffic.
- Behavioral learning: NeuVector profiles 'normal' (process + network + file) and alerts on drift.
- Compliance scanning: CIS Docker Benchmark, CIS K8s Benchmark, DISA STIG, PCI v4, NIST 800-53/171.
- Includes WAF and DLP sensors at the pod-network boundary.



Source: open-docs.neuvector.com • github.com/neuvector/neuvector

NeuVector Architecture — Controllers, Enforcers, Scanners



Four pieces

Controller

Cluster brain. REST API + state. Deploy 3 for HA.

Enforcer

DaemonSet on every node. Inline traffic + process + file inspection.

Manager

Stateless web UI (HTTPS). Scale as desired.

Scanner

Replicaset for CVE scans. Scales horizontally. CVE DB updated by Updater.

Source: open-docs.neuvector.com/basics/overview

What 'Runtime Protection' Actually Means

L7 PACKET INSPECTION

Enforcer sees decrypted HTTP/gRPC/Kafka/Redis at the pod boundary. Detects SQL injection, XSS, abnormal API paths, lateral movement. Generates per-service allow/deny rules from observed traffic.

PROCESS & FILE PROFILING

Records every process exec and file write inside the container. Alerts on a shell spawned where there should be none. Quarantines containers that violate the learned profile.

DLP & WAF SENSORS

Pattern-match outbound traffic for SSNs, credit cards, sensitive headers. Block OWASP Top 10 attacks (XSS, SQLi, RCE, deserialization). Per-namespace, per-service policy.

COMPLIANCE & CVE

Daily CIS K8s + CIS Docker + DISA STIG scans across the fleet. Image CVE scans on push, on schedule, on demand. Export reports to SYSLOG, webhook, S3.

Source: open-docs.neuvector.com/policy • open-docs.neuvector.com/scanning

Kubewarden — Policy as WebAssembly

- CNCF Sandbox project. Apache-2.0. github.com/kubewarden.
- Kubernetes Dynamic Admission Controller — same interface as OPA/Gatekeeper.
- Policies are WebAssembly modules. Write them in Rust, Go (TinyGo), Swift, AssemblyScript, or Rego.
- Each policy runs in its own Wasm sandbox — memory-safe, language-agnostic, no host escape.
- Built-in policy catalog: PSA replacements, image signature verification (cosign), RBAC linters, etc.
- Single binary on the host side; policies are OCI artifacts (you 'docker pull' a policy).

WRITE POLICIES IN

- Rust
- Go (TinyGo)
- Swift
- AssemblyScript
- Rego (OPA)
- Python (experimental)

All compiled to Wasm

Source: kubewarden.io • github.com/kubewarden

Fleet — GitOps for ONE Cluster or a THOUSAND

- Apache-2.0. github.com/rancher/fleet. Bundled with Rancher but runs standalone.
- Built specifically for SCALE — designed for managing thousands of clusters from one Git repo.
- Watches Git → produces Bundles → BundleDeployments on each target cluster.
- Supports raw manifests, Kustomize overlays, and Helm charts in the same workflow.
- Single-cluster style (Fleet runs IN the cluster it manages) OR multi-cluster style (central manager).
- Agent on each downstream cluster pulls instructions via the same tunnel pattern as Rancher.

1 repo → N clusters

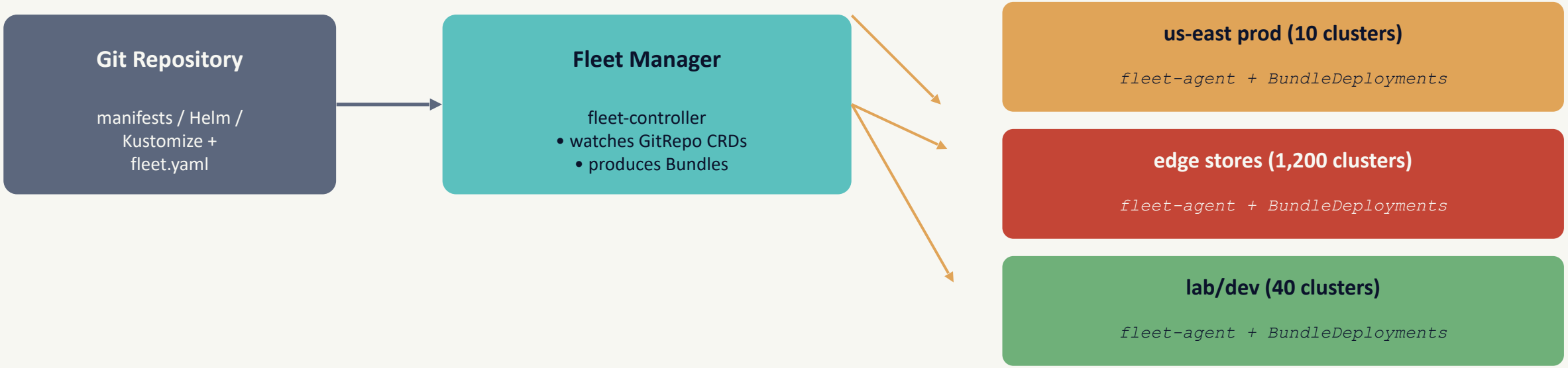
with per-cluster overlays

vs. ArgoCD?

Argo is great for one cluster.
Fleet is built for the fleet.
Use what fits.

Source: fleet.rancher.io • github.com/rancher/fleet

Fleet Architecture — GitRepo → Bundle → Cluster



CRDs you'll actually touch

GitRepo

Points at a repo + branch + path. Generates Bundles. You write these.

Bundle

Auto-generated unit of deployment. Helm chart rendered server-side. You usually don't write these.

BundleDeployment

Instance of a Bundle on a specific cluster, with cluster-specific overlays applied.

Source: fleet.rancher.io/explanations/architecture • fleet.rancher.io/explanations/concepts

Hauler — The Air-Gap Swiss Army Knife

- Apache-2.0. github.com/hauler-dev/hauler. Originally Rancher Federal, now standalone.
- Single Go binary. Fetch every artifact your cluster needs into a single tarball. Move tarball over the wire.
- Stores everything as OCI artifacts: container images, Helm charts, raw files, and more.
- Embedded OCI registry + file server: serve the bundle in your high-side network with one command.
- Declarative manifests (haul YAMLs) — list what you need, hauler grabs it.
- Works with Sigstore signatures + cosign verification — supply chain survives the airgap.

TYPICAL FLOW

```
# low-side
hauler store sync \
  -f haul.yaml

# burn tarball
hauler store save \
  -f hauler.tar.zst

# walk it across
# the airgap...

# high-side
hauler store load \
  -f hauler.tar.zst

hauler store serve \
  registry
```

Source: docs.hauler.dev • github.com/hauler-dev/hauler

The Sigstore + Trivy + SBOM Combo

COSIGN / SIGSTORE

Keyless image signing using OIDC identities (or your own keys). Signatures stored in the registry as OCI artifacts. Verify with one flag in Kubewarden, NeuVector, or `cosign verify`.

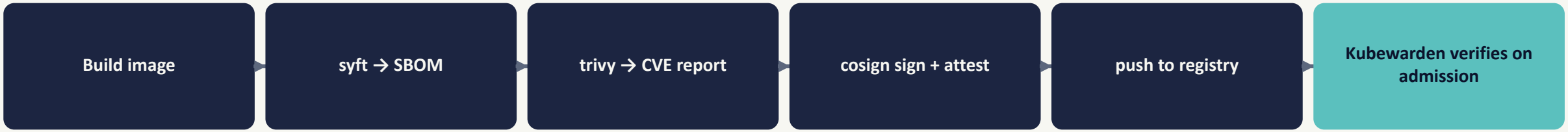
SYFT / SBOM

Generate SPDX or CycloneDX SBOM at build time. Attach as an OCI attestation. Now every image carries a 'parts list' a customer/auditor can validate.

TRIVY

Open-source vulnerability scanner. Scans container images, IaC, SBOMs. CNCF-friendly. Used inside NeuVector, Carbide, and a hundred CI pipelines.

Put it together



Source: sigstore.dev • github.com/anchore/syft • aquasecurity.github.io/trivy

The Stack Reassembled — Every Box Has a Repo Now

Everything in this talk runs at home, in your homelab, in production. All Apache/MIT/MPL/BSD.

Apps / Workloads	your-org/whatever
GitOps	rancher/fleet
Policy	kubewarden
Container Security	neuvector/neuvector
Multi-Cluster Mgmt	rancher/rancher
Virtualization / HCI	harvester/harvester + kubevirt/kubevirt
Distributed Storage	longhorn/longhorn
Kubernetes Distro	k3s-io/k3s + rancher/rke2
Immutable OS	openSUSE/MicroOS (SLE Micro upstream)
+ Supply Chain & Airgap	sigstore/cosign • hauler-dev/hauler • aquasecurity/trivy • anchore/syft

Source: github.com — all of the above

CLOSING

Where to Start Tonight

10-minute lab

curl -sfL https://get.k3s.io | sh - → you have a single-node K8s cluster in 30 seconds. kubectl just works.

30-minute lab

Run RKE2 in HA on three VMs, install Longhorn from its Helm chart, install NeuVector. Now you have a hardened K8s cluster with storage and security.

Saturday morning lab

Burn the Harvester ISO to a USB, install on bare metal, add it to Rancher. Run a VM and a container side-by-side, see them in the same UI.

Week-long lab

Set up Fleet against a Git repo. Have it deploy your dev/staging/prod clusters with overlays. Add Kubewarden policies for signature verification.

Community channels

- Slack: rancher-users.slack.com — channels for k3s, rke2, rancher, longhorn, harvester, fleet, neuvector
- Kubernetes Slack: #kubewarden, #kubevirt • CNCF Slack: #longhorn, #kubewarden
- GitHub issues — most projects respond within a day

Questions?

And links for further reading.

`docs.k3s.io` k3s.io • github.com/k3s-io/k3s

`ranchermanager.docs.rancher.com` github.com/rancher/rancher

`docs.harvesterhci.io` github.com/harvester/harvester

`kubewarden.io` github.com/kubewarden

`docs.hauler.dev` github.com/hauler-dev/hauler

`en.opensuse.org/Portal:MicroOS` github.com/openSUSE/transactional-update

`docs.rke2.io` github.com/rancher/rke2

`longhorn.io` github.com/longhorn/longhorn

`open-docs.neuvector.com` github.com/neuvector/neuvector

`fleet.rancher.io` github.com/rancher/fleet

`sigstore.dev` github.com/sigstore/cosign

`kubevirt.io` github.com/kubevirt/kubevirt

Chris Reynolds • chris@creynold.net