Introduction to Kubernetes

SCALE 23x Workshop

Hands-On Workshop

SCaLE K8s Workshop

github.com/faisalcodesinfrastructure/scale23x-k8s-workshop

Tools: Docker Desktop · kind · kubectl · helm · Git

The Team

Faisal Afzal

Principal Technical Consultant, AHEAD

- CNCF & Platform Engineering Ambassador
- Cloud Native LA Organizer

linkedin.com/in/faisalafzal

Josh Berkus

Kubernetes Community Architect Manager, Red Hat

- K8s Community, OSI Board Member
- TAG Co-Chair, SIG Contributor

linkedin.com/in/josh-berkus

Justin Garrison

Head of Product, Sidero Labs

- Author
- Podcaster

linkedin.com/in/justingarrison

Fabrizio Sgura

Chief Engineer, Veritas Automata

- Golden Kubestronaut
- K8s Expert, CNCF Ambassador

linkedin.com/in/fabrizio-sgura

Cloud Native Los Angeles

community.cncf.io/cloud-native-los-angeles

Meetups · Talks, Workshops & Community

Workshop Agenda

1	**Core Concept Review**	Pods, Nodes, Control Plane, Architecture
2	**Tool Installation**	Docker Desktop, kind, kubectl, helm
3	**Create a kind Cluster**	Local K8s cluster on your laptop
4	**Namespaces**	Isolating workloads and environments
5	**Deployments & Replicas**	Running and scaling applications
6	**Services**	Exposing apps inside and outside the cluster
7	**Ingress**	HTTP routing to services
8	**ConfigMaps & Secrets**	Externalizing configuration
9	**Scaling & Updates**	Rolling updates and rollbacks

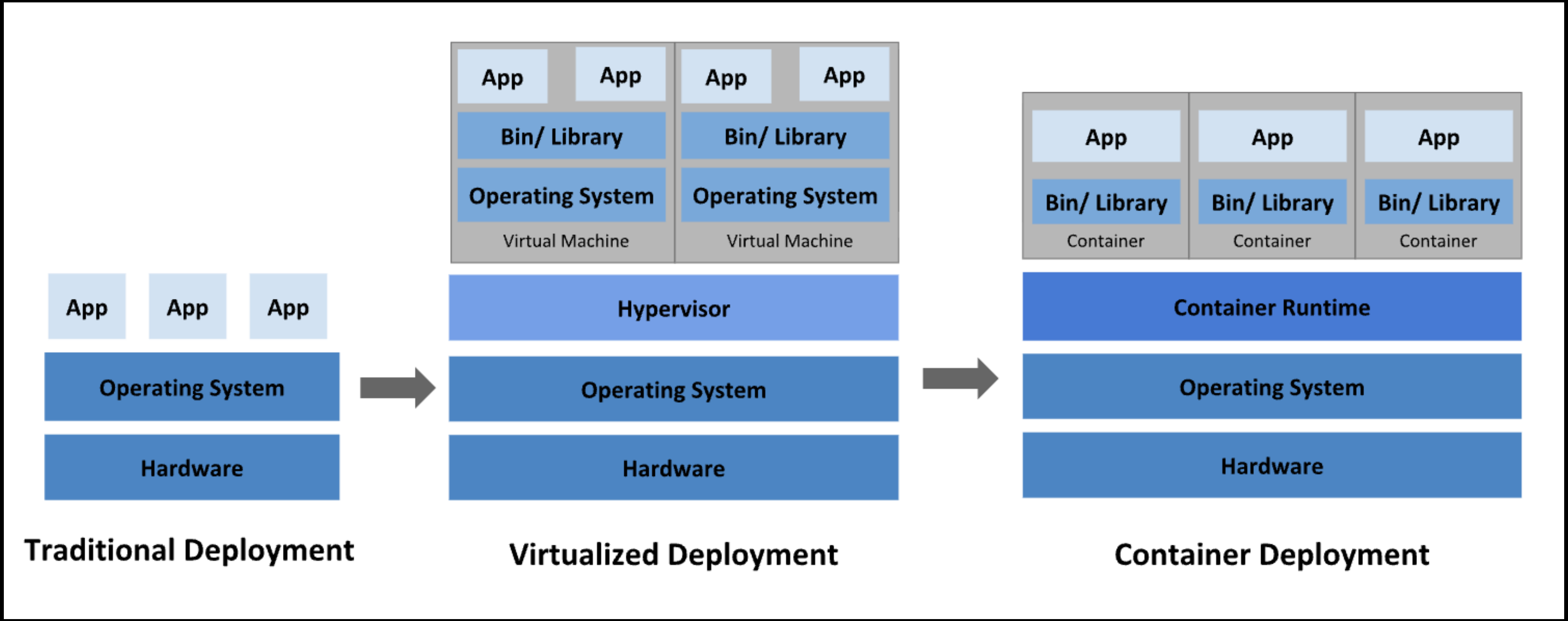
What is Kubernetes?

Container Orchestration Platform

- Originally designed by Google, donated to CNCF in 2016
- Automates deployment, scaling, and management of containerized apps
- Self-healing: restarts failed containers, reschedules on node failure
- Declarative configuration: you define desired state, K8s makes it real
- Works across bare metal, VMs, public/private/hybrid cloud

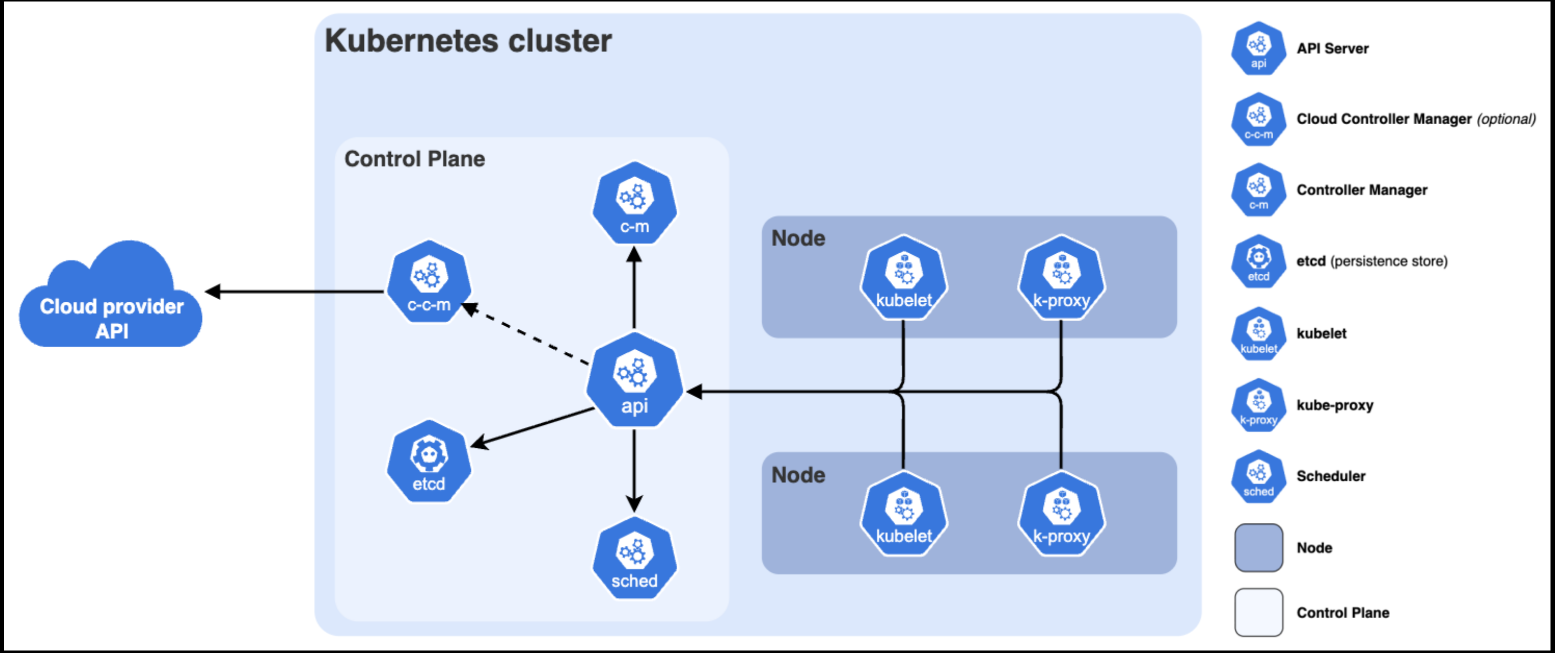
"K8s" = Kubernetes (8 letters between K and s)

Kubernetes: Historical Context



Source: https://kubernetes.io/docs/concepts/overview/

Kubernetes: Essential Components



Source: https://kubernetes.io/docs/concepts/overview/components/

Core Concepts

Pod

Smallest deployable unit. One or more containers sharing network [ip address] & storage.

Node

A worker, or control plane, machine or more (VM or physical) running your pods.

Cluster

A set of nodes managed by a control plane.

Namespace

Virtual cluster for resource isolation and multi-tenancy.

Deployment

Manages ReplicaSets to maintain desired pod count.

Service

Stable network endpoint to access a set of pods. Share with other pods or with external network.

Ingress

HTTP/S routing rules exposing services externally. Requires controller & DNS, Supports TLS/HTTPS.

ConfigMap

Non-sensitive config data injected into pods.

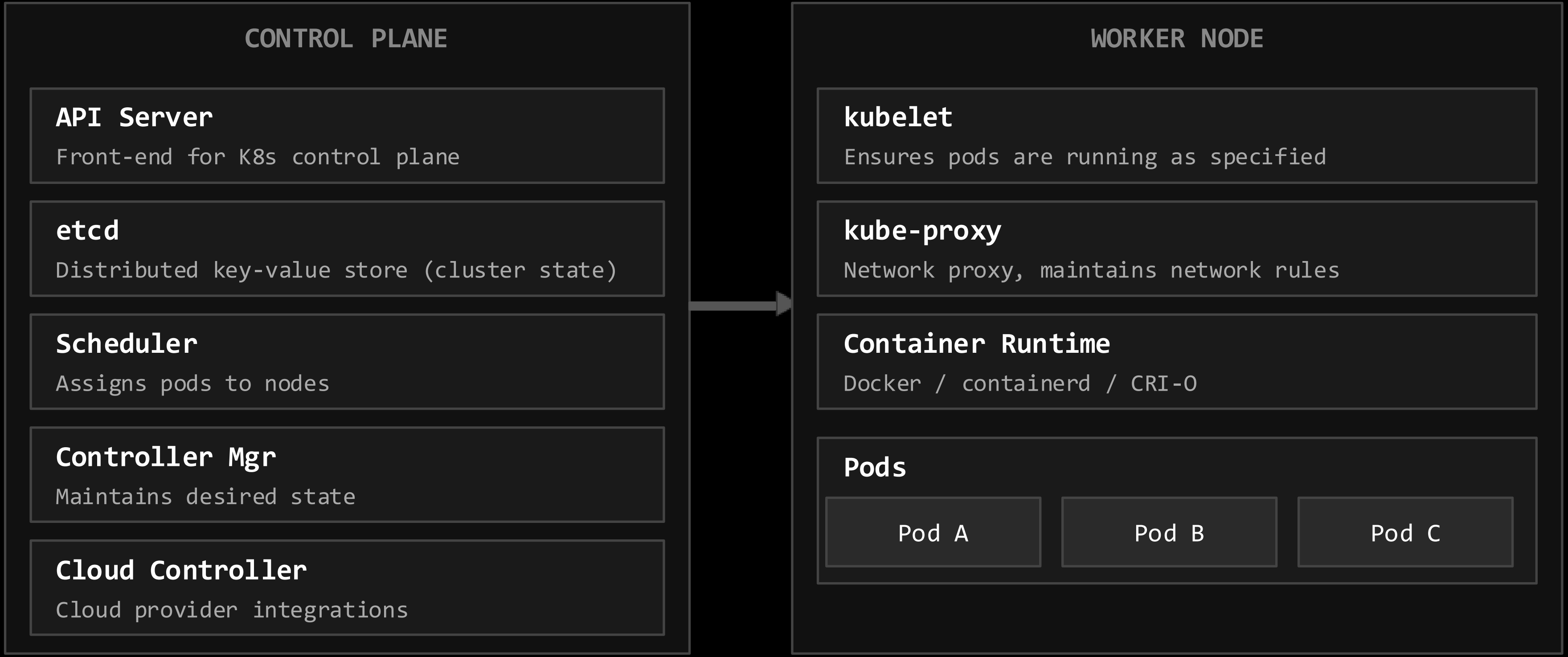
Secret

Base64-encoded sensitive data (passwords, tokens, certs).

Volumes

Local storage / node. PersistentVolumes: shared storage on network storage.

Kubernetes Architecture



CONTROL PLANE

API Server
Front-end for K8s control plane

etcd
Distributed key-value store (cluster state)

Scheduler
Assigns pods to nodes

Controller Mgr
Maintains desired state

Cloud Controller
Cloud provider integrations

WORKER NODE

kubelet
Ensures pods are running as specified

kube-proxy
Network proxy, maintains network rules

Container Runtime
Docker / containerd / CRI-O

Pods
Pod A | Pod B | Pod C

Multiple Worker Nodes form a Cluster · Control Plane manages them all

Tool Installation

Docker Desktop & Git *docker.com/products/docker-desktop & https://git-scm.com/install/*	Container runtime + includes kubectl. Required for kind.
kind *kind.sigs.k8s.io*	Kubernetes IN Docker. Runs K8s clusters using Docker containers.
kubectl *kubernetes.io/docs/tasks/tools*	CLI to communicate with the Kubernetes API server.
helm *helm.sh/docs/intro/install*	Kubernetes package manager for deploying charts.

Creating a kind Cluster

Create your first cluster

```
# Create a single-node cluster
kind create cluster --name workshop

# Create multi-node cluster from config
kind create cluster --name workshop --config kind-
config.yaml

# List clusters
kind get clusters

# Set kubectl context
kubectl cluster-info --context kind-workshop

# Delete cluster
kind delete cluster --name workshop
```

kind-config.yaml

```
kind: Cluster
apiVersion: kind.x-
k8s.io/v1alpha4
nodes:
- role: control-plane
- role: worker
- role: worker
```

kind nodes = Docker containers

Each node gets its own kubelet + runtime

Namespaces

Virtual clusters within a cluster for isolation

default	kube-system	kube-public	your-app
Default namespace for objects with no other namespace	Objects created by K8s (scheduler, dns, etc.)	Readable by all users, used for cluster info	Your workloads, naming is up to you

```
# Create namespace
kubectl create namespace workshop

# List namespaces
kubectl get namespaces

# Work in a namespace
kubectl get pods -n workshop
kubectl get all -n workshop

# Set default namespace
kubectl config set-context --current \
  --namespace=workshop
```

Resource Quota and LimitRange can be applied per namespace

RBAC permissions are scoped

to namespaces

Deployments & ReplicaSet

Manages a set of pods to run an application workload

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-app
  namespace: workshop
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

```
# Apply the deployment
kubectl apply -f deployment.yaml

# Check deployment status
kubectl get deployments -n workshop
kubectl rollout status deployment/nginx-app

# Scale manually
kubectl scale deployment nginx-app --replicas=5

# View pods
kubectl get pods -n workshop -l app=nginx

# Describe a deployment
kubectl describe deployment nginx-app
```

Services

Stable virtual IP and DNS name for accessing a set of pods

ClusterIP	NodePort	LoadBalancer	ExternalName
Internal only (default)	Exposed on each node's IP	Cloud load balancer	DNS alias for external service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
  namespace: workshop
spec:
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
  type: ClusterIP
```

```bash
# Apply service
kubectl apply -f service.yaml

# List services
kubectl get svc -n workshop

# Quick expose (imperative)
kubectl expose deployment nginx-app \
  --port=80 --type=NodePort

# Port-forward for local testing
kubectl port-forward svc/nginx-svc 8080:80
```

Ingress

HTTP/S routing rules — host-based and path-based routing to Services

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: workshop-ingress
  namespace: workshop
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
  - host: app.workshop.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-svc
            port:
              number: 80
```

Ingress Controller Required
Install nginx ingress for kind:

```
kubectl apply -f
https://raw.githubusercontent.com/
kubernetes/ingress-nginx/main/deploy/static/
provider/kind/deploy.yaml

# List ingress rules
kubectl get ingress -n workshop

# Add to /etc/hosts for local testing
127.0.0.1  app.workshop.local
```

Ingress ≠ Service
Ingress routes to Services, not Pods

ConfigMaps & Secrets

ConfigMap – Non-sensitive config data

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  LOG_LEVEL: debug
  APP_ENV: production
  config.json: |
    {"timeout": 30}
```

```yaml
# Inject as environment variables
envFrom:
- configMapRef:
    name: app-config
- secretRef:
    name: app-secret

# Mount as volume files
volumes:
- name: config-vol
  configMap:
    name: app-config
```

Secret – Sensitive data (base64 encoded)

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
type: Opaque
data:
  DB_PASS: cGFzc3dvcmQxMjM=
  API_KEY: bXlzZWNyZXRrZXk=
```

```
# Create from literals (imperative)
kubectl create secret generic app-secret \
  --from-literal=DB_PASS=password123


#   Secrets are only base64 encoded,
#   not encrypted. Use Sealed Secrets
#   or external-secrets for production.
```

Scaling & Rolling Updates

Scaling

```
# Manual scaling
kubectl scale deployment nginx-app --replicas=5 -n
workshop

# Autoscaling (HPA)
kubectl autoscale deployment nginx-app \
  --cpu-percent=50 --min=2 --max=10

# Check HPA status
kubectl get hpa -n workshop
```

RollingUpdate Strategy:

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 1
    maxSurge: 1
```

Rolling Updates & Rollbacks

```
# Update image (triggers rolling update)
kubectl set image deployment/nginx-app \
  nginx=nginx:1.26 -n workshop

# Watch rollout progress
kubectl rollout status deployment/nginx-app

# Rollout history
kubectl rollout history deployment/nginx-app

# Rollback to previous version
kubectl rollout undo deployment/nginx-app

# Rollback to specific revision
kubectl rollout undo deployment/nginx-app \
  --to-revision=2
```

The CNCF Landscape

Cloud Native Computing Foundation

The neutral home for Kubernetes, Prometheus, Envoy, and 200+ cloud native projects

Orchestration & Runtime

- Kubernetes
- containerd
- CRI-O
- KubeEdge

Observability

- Prometheus
- Grafana
- Jaeger
- OpenTelemetry

Networking & Service Mesh

- Istio
- Cilium
- Envoy
- Linkerd

Security

- Falco
- OPA/Gatekeeper
- cert-manager
- Vault

Storage

- Rook
- Longhorn
- OpenEBS
- Velero

CI/CD & GitOps

- ArgoCD
- FluxCD
- Tekton
- Argo Workflows

Explore all 1,000+ projects and products: landscape.cncf.io · cncf.io/projects

What's Next?

▸ **Helm Charts**	Package your apps for reuse and sharing	▸ **Persistent Storage**	PersistentVolumes, PVCs, StorageClasses
▸ **RBAC**	Role-based access control for security	▸ **Network Policies**	Control pod-to-pod traffic
▸ **Operators**	Extend K8s with custom controllers	▸ **GitOps**	ArgoCD / FluxCD for declarative deployments

kubernetes.io/docs · cncf.io · killercoda.com