August 28

KEYSPACE

Amsterdam

[Tickets]

Effortless Platform Engineering: Multi-Cluster Valkey Deployments with k0rdent

Prithvi Raj
Senior Community Manager & Developer Advocate at Mirantis

Valkey

Background The Horsehead Nebula and its surroundings. The reflection nebula NGC 2023 in the bottom left corner. / Stephanh / License: CC BY 4.0.

About Me

Prithvi Raj

AMBASSADOR

Community Manager & Developer Advocate

- CNCF Ambassador
- Community Manager for **k0s** & **k0rdent**
- Ex- Community Leader for the **LitmusChaos project**.
- KCD Bengaluru co-organizer

(2020 - 2021) MayaData

(2021 - 2022) CHAOSNATIVE

(2022 - 2024) harness

(2024– present) MIRANTIS

k0s KØRDENT OPEN SDN

KCD Chennai

CHAOS carnival

Kubernetes Community Days ▸ Bengaluru

Platform Engineering & Resilience Engineering Meetup Group

LitmusChaosCon
The Chaos Engineering Community Conference

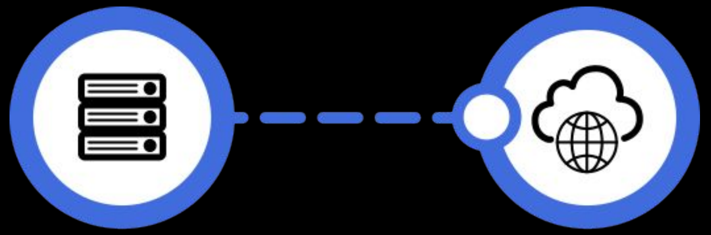
CLOUD NATIVE কলকাতা COMMUNITY GROUP

@prithvi137 /prithvi1307

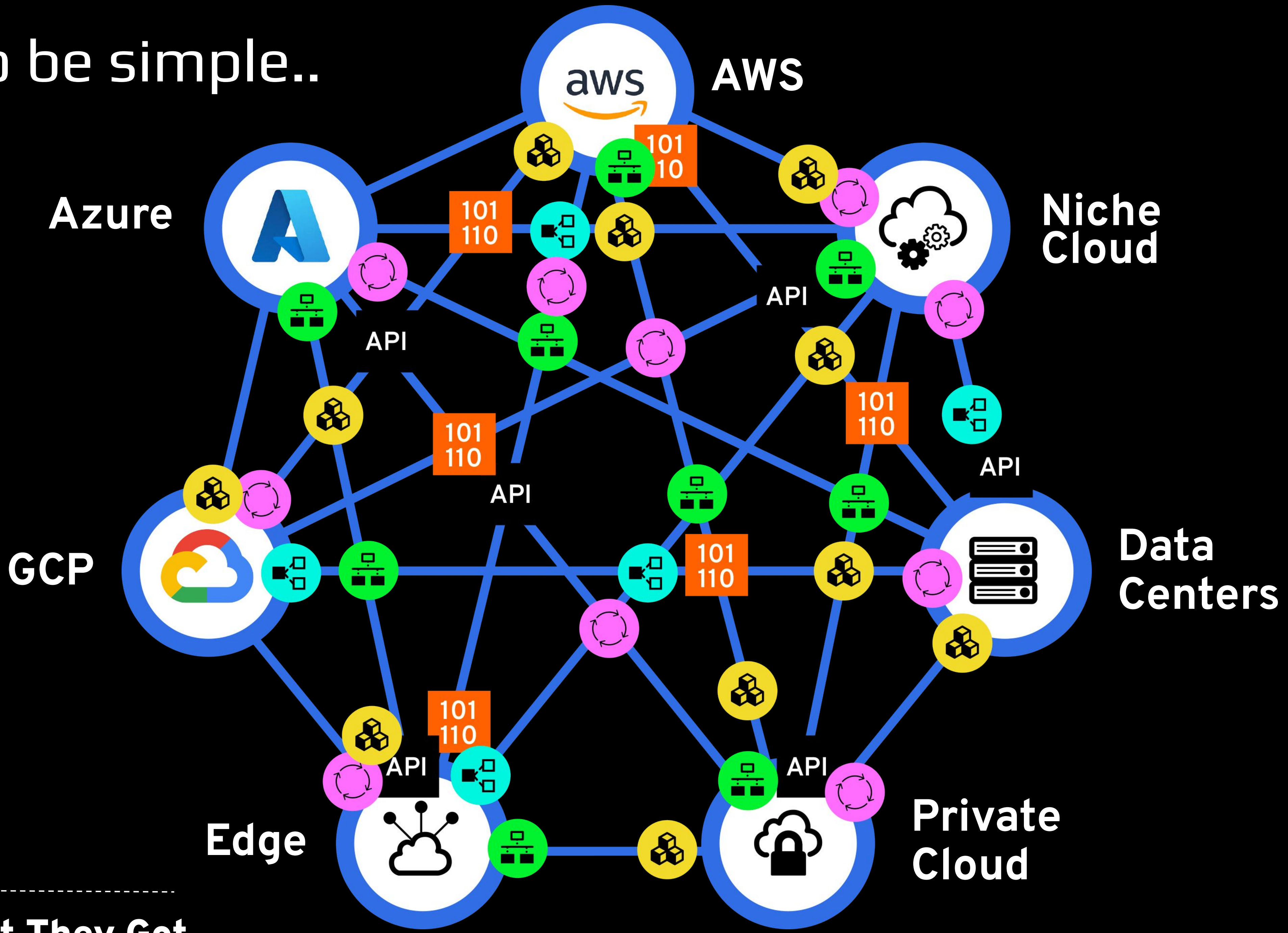
What's on the agenda?

Let’s be sure...

Cloud was meant to be simple..



What Customers Expected

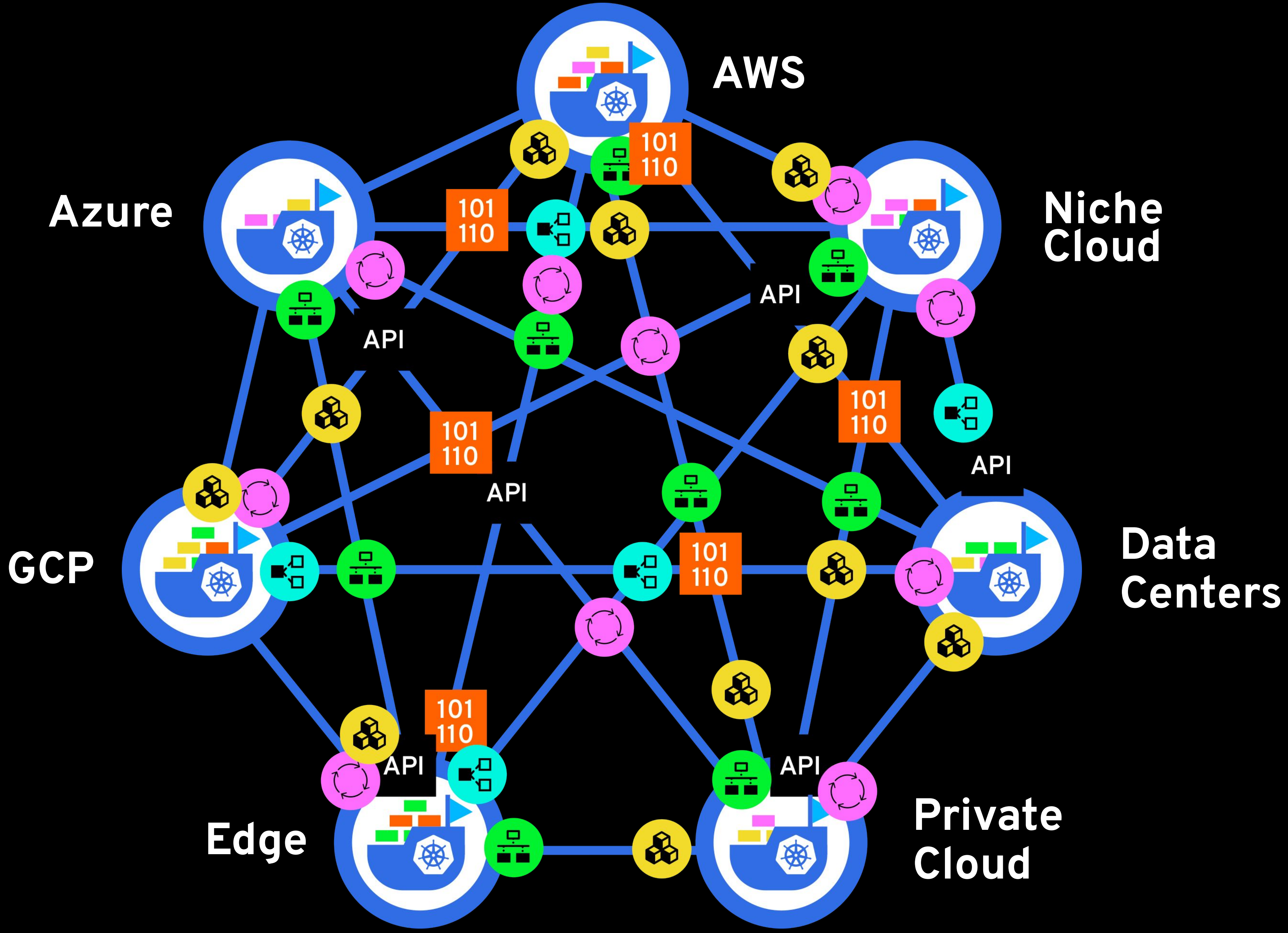


What They Got

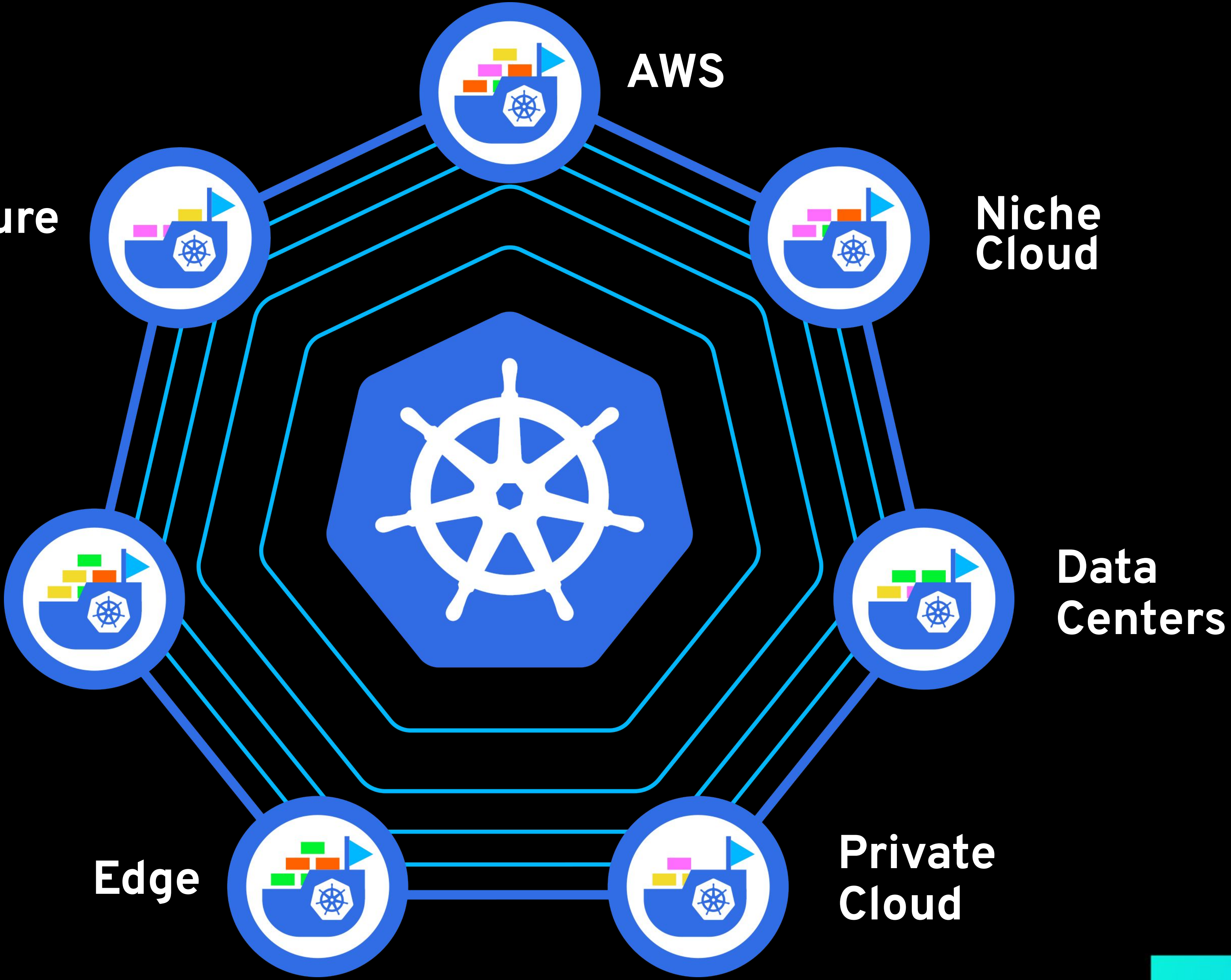
Container Orchestrators tried making it simpler

docker
SWARM
Apache
MESOS™
kubernetes

But K8s has matured..



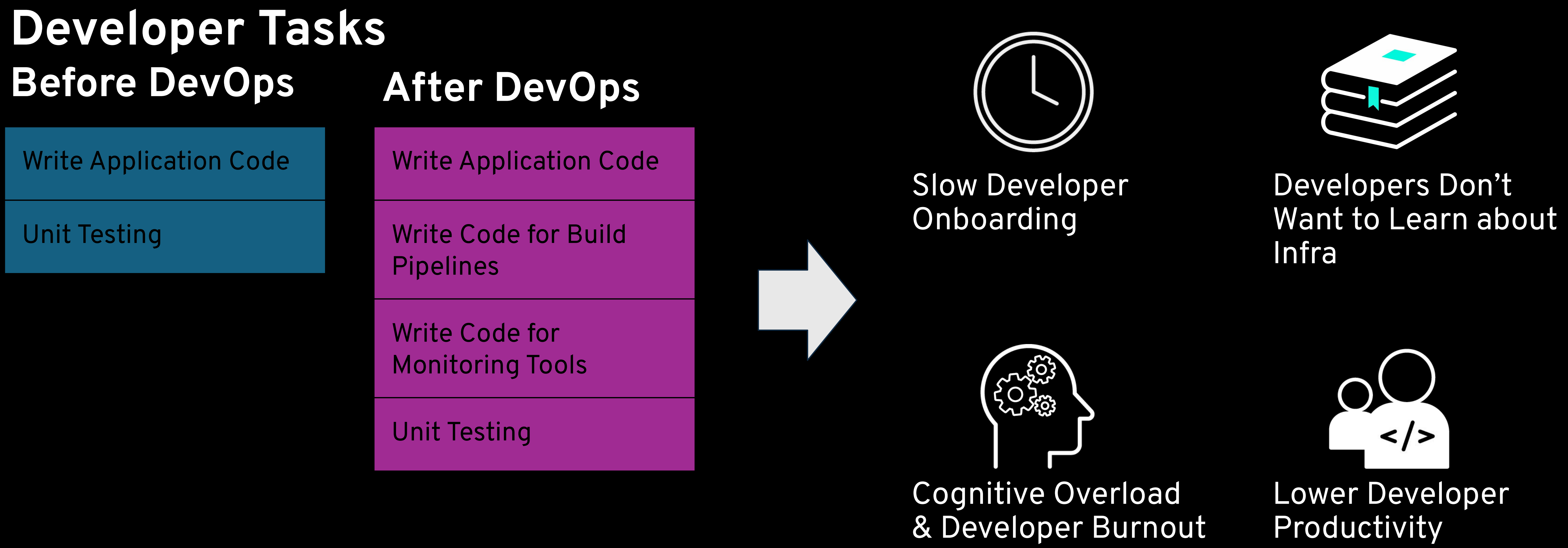
K8s is the common control plane...



Platform Engineering Challenges & Solutions

DevOps Challenges

Developer Tasks



Before DevOps

- Write Application Code
- Unit Testing

After DevOps

- Write Application Code
- Write Code for Build Pipelines
- Write Code for Monitoring Tools
- Unit Testing

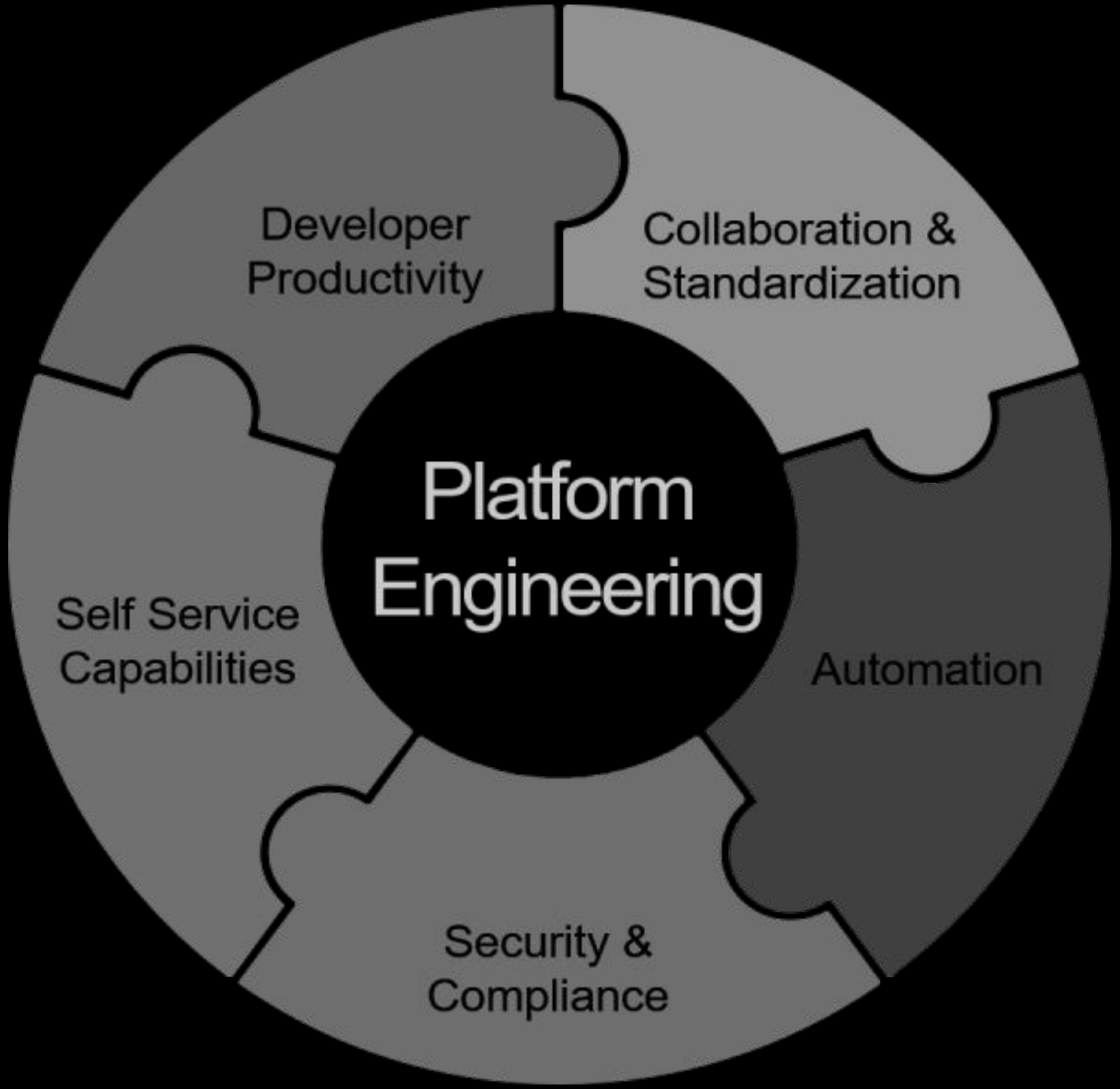
Slow Developer Onboarding

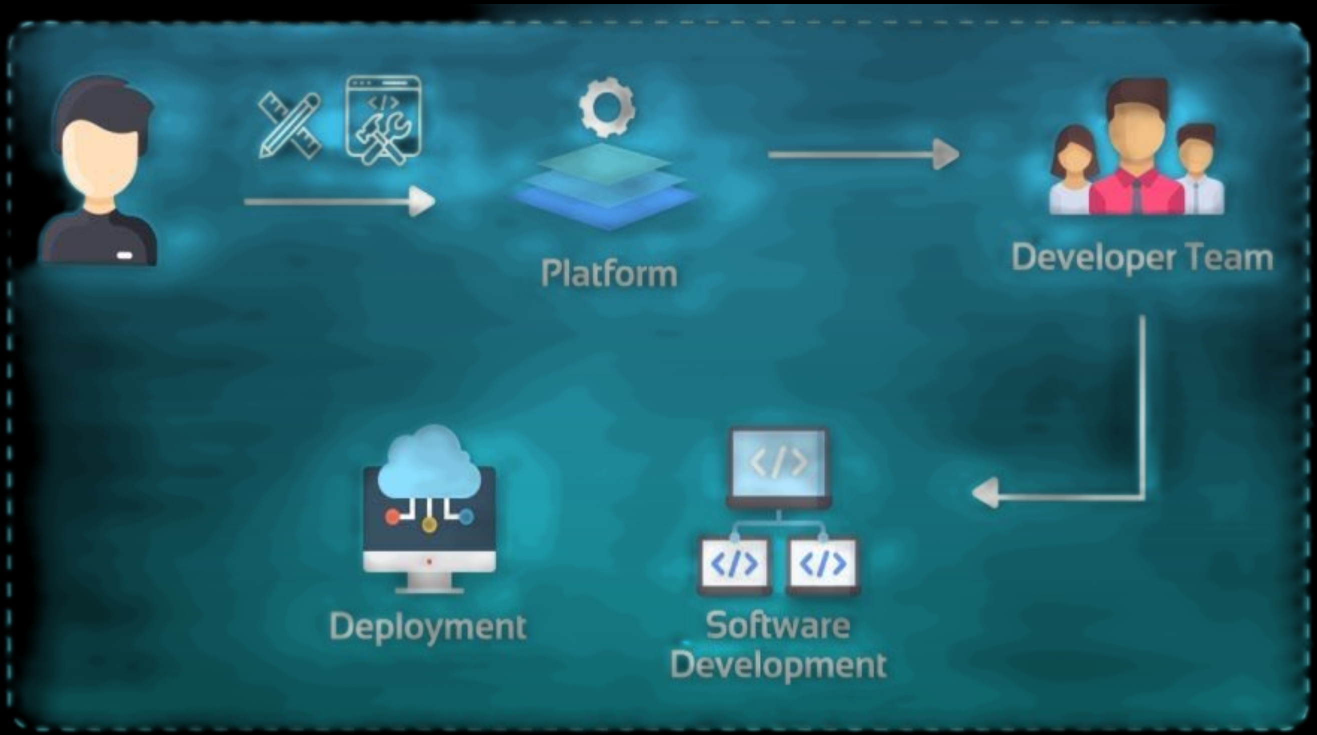
Developers Don’t Want to Learn about Infra

Cognitive Overload & Developer Burnout

Lower Developer Productivity

Platform Engineering Defined





Top Reasons for Considering Platform Engineering

1	Agility	Production releases happen only once a month.	We want to accelerate our cloud native journey.
2	Cost	Manual, repetitive tasks are costly to maintain.	We’re wasting money with underutilized resources.
3	Reliability	Too many critical incidents are causing application downtime.	It takes too long to find out when something goes wrong.
4	Consistency	It’s too hard to manage different pipelines for each product.	We need standardized processes for security and compliance.

Requirements for Platform Engineering

Developer Self-Service

Security & Compliance

Operational Simplicity

Visibility & Control

Ease of Customization

Platform Engineers need Multi-Cluster Configurations

AI/ML

Hybrid Multi-Cloud

Edge / IoT

High Availability

Multi-Tenancy

Data Sovereignty

Platforms need to be democratic

Infrastructure as Code Driving Platforms

Drives Automation and Consistency

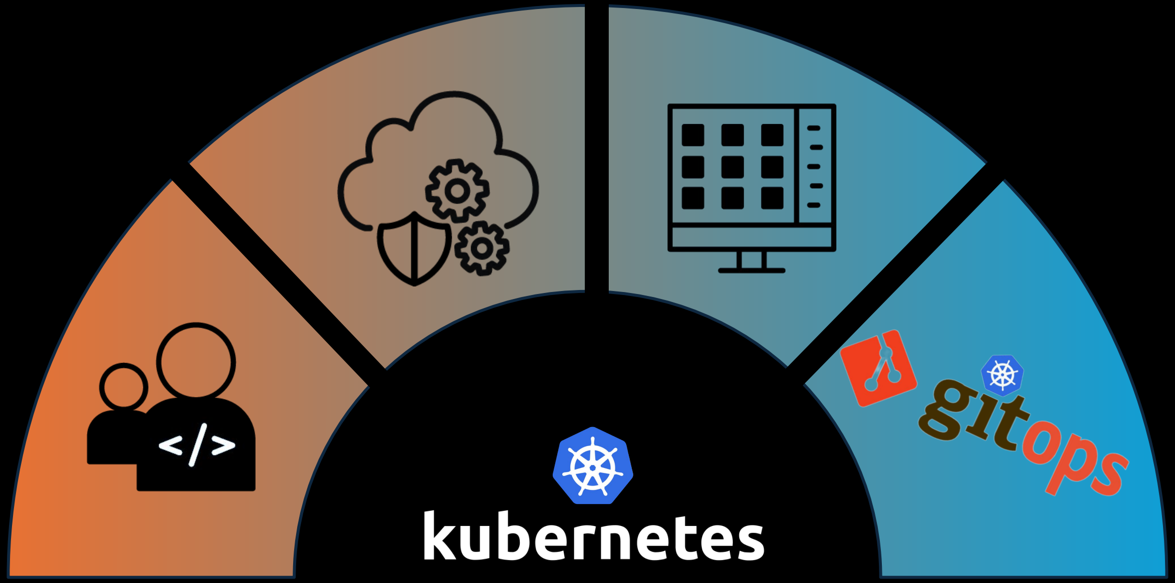
Effortless management with continuous reconciliation

Foundation for Self-Service Developer Portals

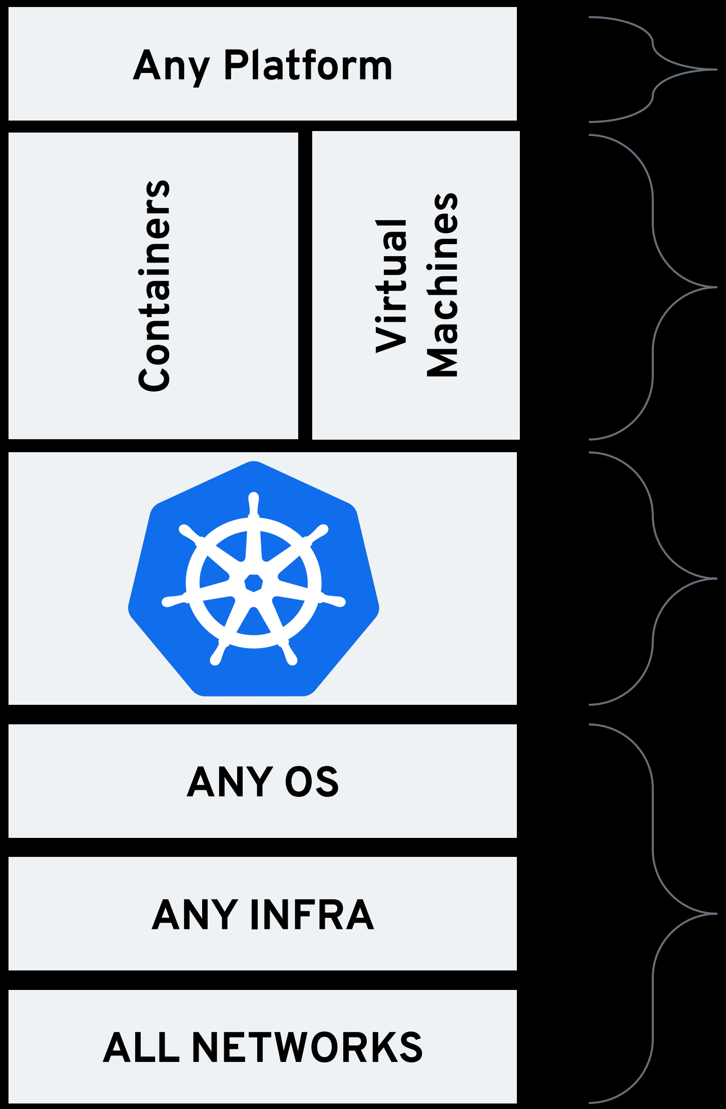
Enables Platform Standardization and Reusability

\

Supports GitOps and DevSecOps Practices



Kubernetes as the Orchestration Fabric of the Future



Platform engineering and Add-on services

All runtimes orchestrated in one place (containers, VMs)

Ubiquitous abstraction layer across all infrastructures

Open API driven

Orchestration Fabric for all workloads and resources

Unique underlying implementations

Value of Kubernetes-Native Approach

kubernetes

Greater developer productivity with declarative configuration and abstraction across infra & services

Automated updates, high availability, security & compliance via k8s continuous reconciliation

Use case-optimized IDPs for any workload, infrastructure

Greater reliability via replication controllers and self-healing

Platform Engineers

Reduced Complexity of Platform Configuration & Management

Developers

High Reliability & Availability of Platforms & End-User Services

Business

Single Point of Control & Visibility for Continuous Optimization

Common Valkey Platform Use Cases

- Read caches & sidecar/shared caches
- Session storage & feature flags
- Lightweight queues & streams
- Coordination primitives
- Edge or tiered caching
- **Rate limiting / request shaping**

Day-2 platform concerns

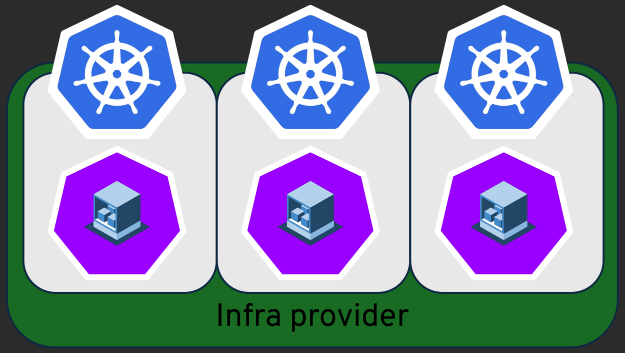
Durability Choice Per Use Case

Topology & HA

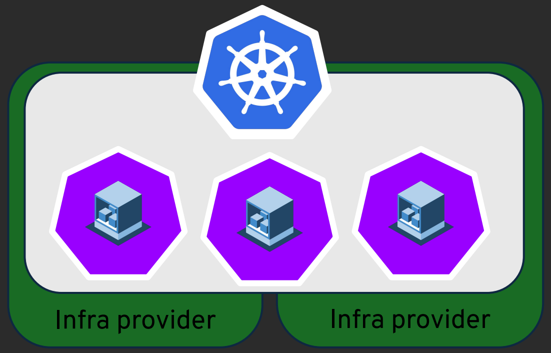
Security

Cost Controls

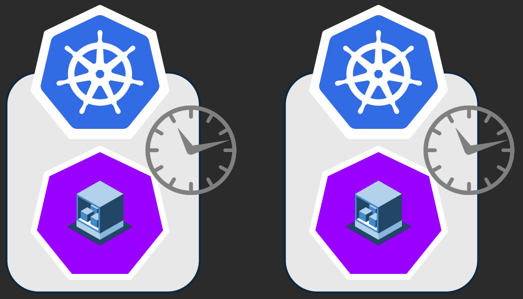
Challenges of Multi-Cluster Platform Engineering



Single Application per Cluster



Multi-Infra Provider



Dynamic On-Demand Clusters

CHALLENGES

Inconsistent Workloads

Inconsistent Policies

Operational Complexity

Kubernetes Sprawl

Options for Multi-Cluster Platform Engineering

Do Nothing

Expensive:
Infra and Ops costs will keep growing, and barriers to developer productivity continue.

DIY Open Source

Complex:
Huge learning curve and operational burden, unsupported open source tools. Hard to extend from on-premises to public cloud

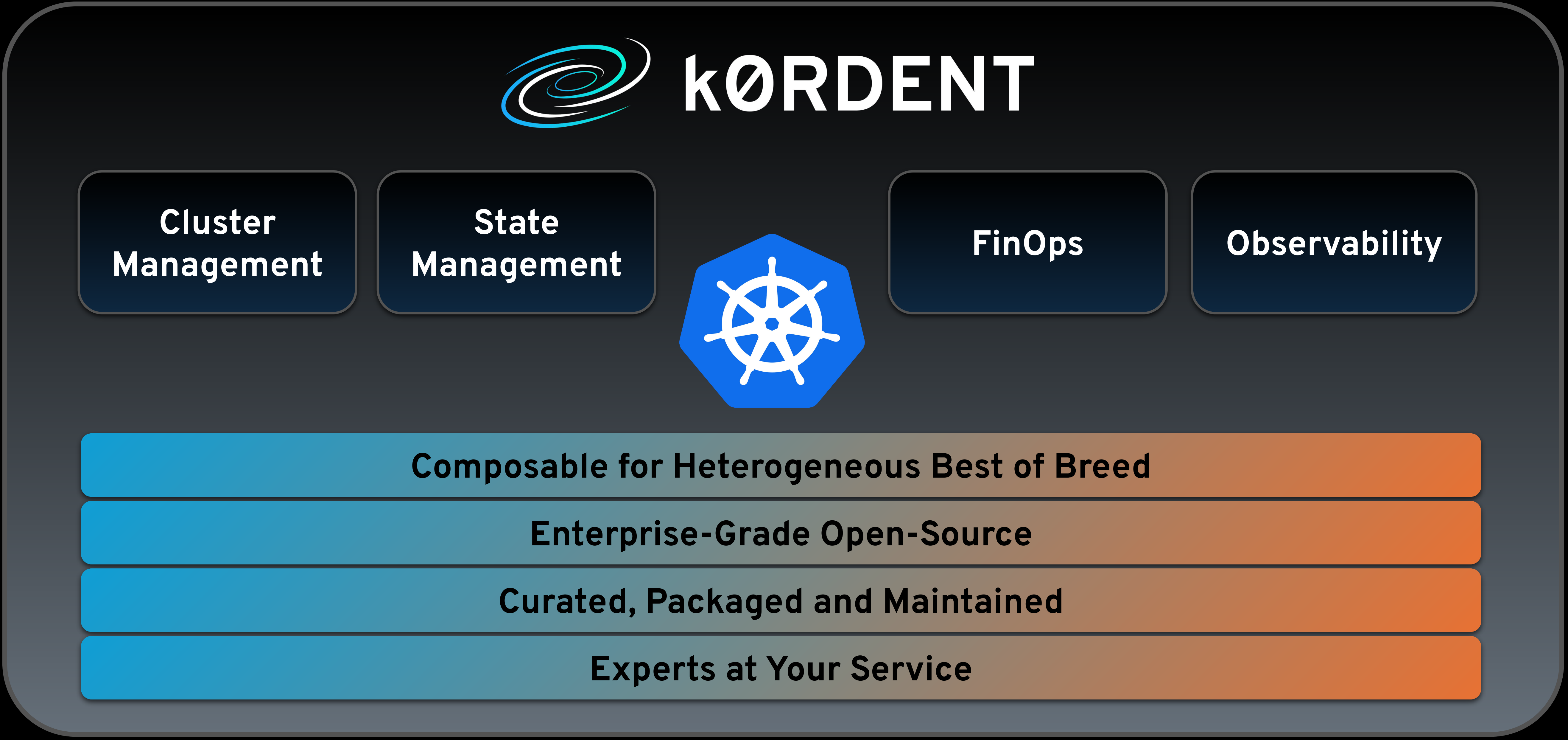
Proprietary Solution

Inflexible:
Limited integration options, expensive licenses and bundle requirements. Hard to extend from on-premises to public cloud

Enterprise-Grade Open Source Solution

Flexible:
Cost-efficient, highly extensible and customizable. Works on-prem and on public clouds

Introducing...



k0RDENT

Cluster Management

State Management

FinOps

Observability

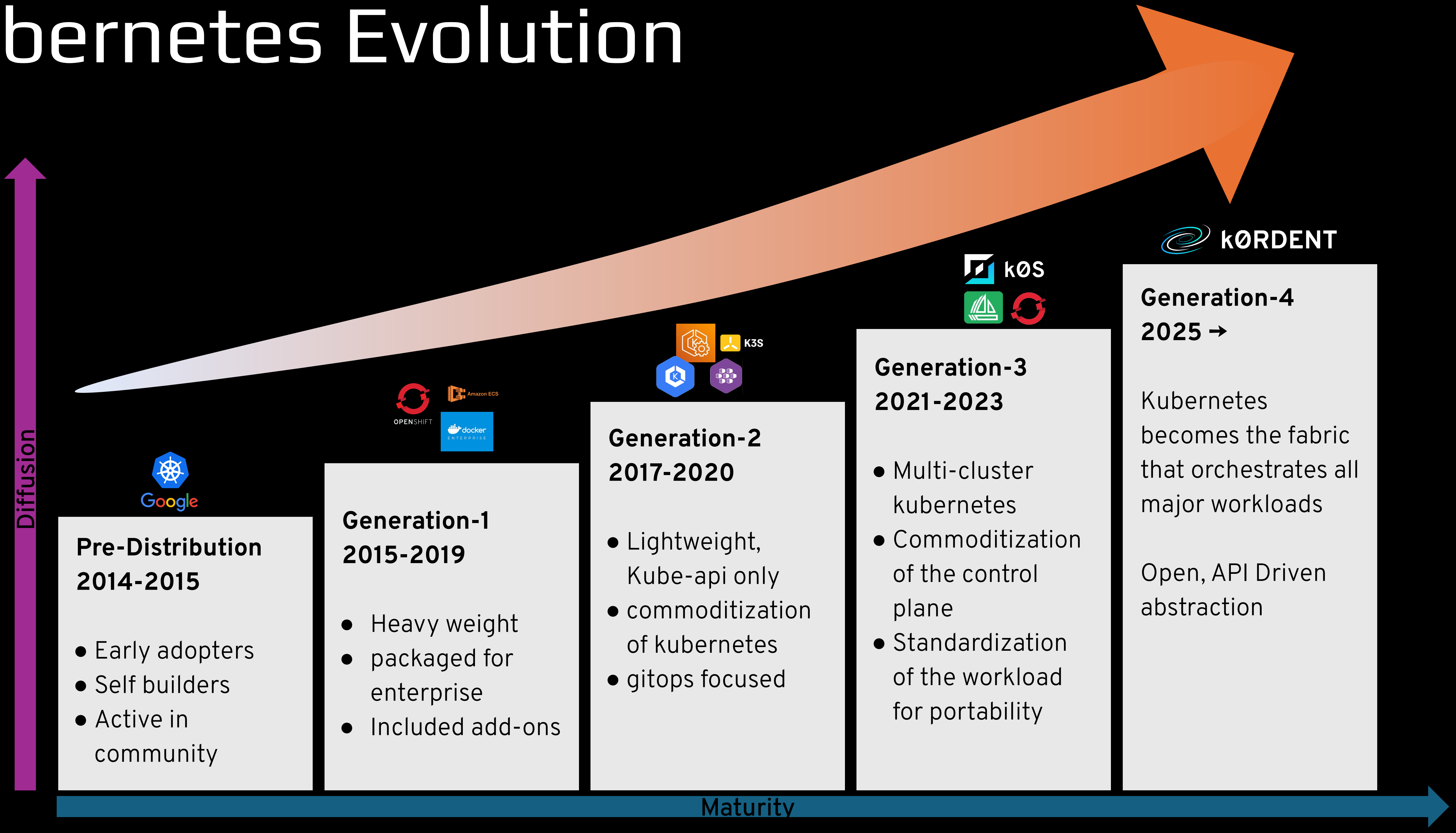
Composable for Heterogeneous Best of Breed

Enterprise-Grade Open-Source

Curated, Packaged and Maintained

Experts at Your Service

Kubernetes Evolution



Pre-Distribution
2014-2015

- Early adopters
- Self builders
- Active in community

Generation-1
2015-2019

- Heavy weight
- packaged for enterprise
- Included add-ons

Generation-2
2017-2020

- Lightweight, Kube-api only
- commoditization of kubernetes
- gitops focused

Generation-3
2021-2023

- Multi-cluster kubernetes
- Commoditization of the control plane
- Standardization of the workload for portability

Generation-4
2025 →

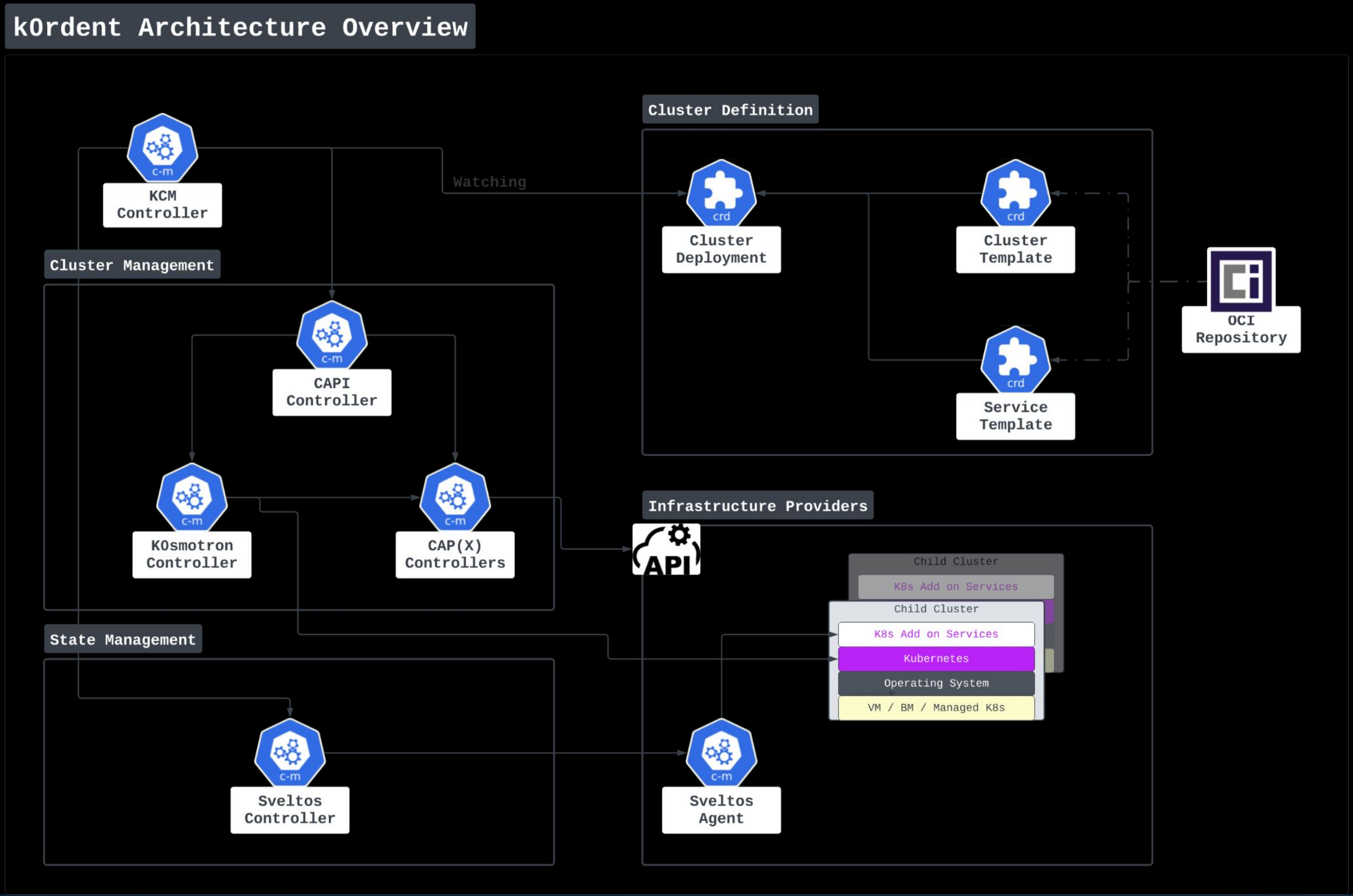
Kubernetes becomes the fabric that orchestrates all major workloads

Open, API Driven abstraction

What is k0rdent

- A declarative and composable platform for managing Kubernetes clusters.
- Key benefits for platform engineers and internal developer platforms:
    - Simplifies multi-cluster operations.
    - Provides automation and scalability.
    - Enhances developer experience and accelerates application delivery.

k0rdent Architecture



k0rdent Architecture Overview

k0rdent components: combined they together manage the full end to end lifecycle of the kubernetes clusters across the estate/fleet.

- **KCM -** k0rdent cluster management

  Role: Responsible for provisioning and managing the lifecycle of Kubernetes clusters using CAPI (Cluster API) and infrastructure providers.

- **KSM** - k0rdent state management

  Role: Sits on top of provisioned clusters and manages the deployment and lifecycle of runtime state, applications and services.

- **KOF -** k0rdent observability finops/framework

  Role: Aggregates metrics, logs, and traces from all managed clusters and stores the data with configurable retention, enabling observability at scale and supporting cost governance.

kØRDENT Open Source Components

CLUSTER MANAGEMENT

kØS

kØSMOTRON

Cluster API

aws

Azure

vmware

openstack

STATE MANAGEMENT

sveltos

CERT MANAGER

NGINX

flux

OBSERVABILITY & FINOPS

VictoriaMetrics

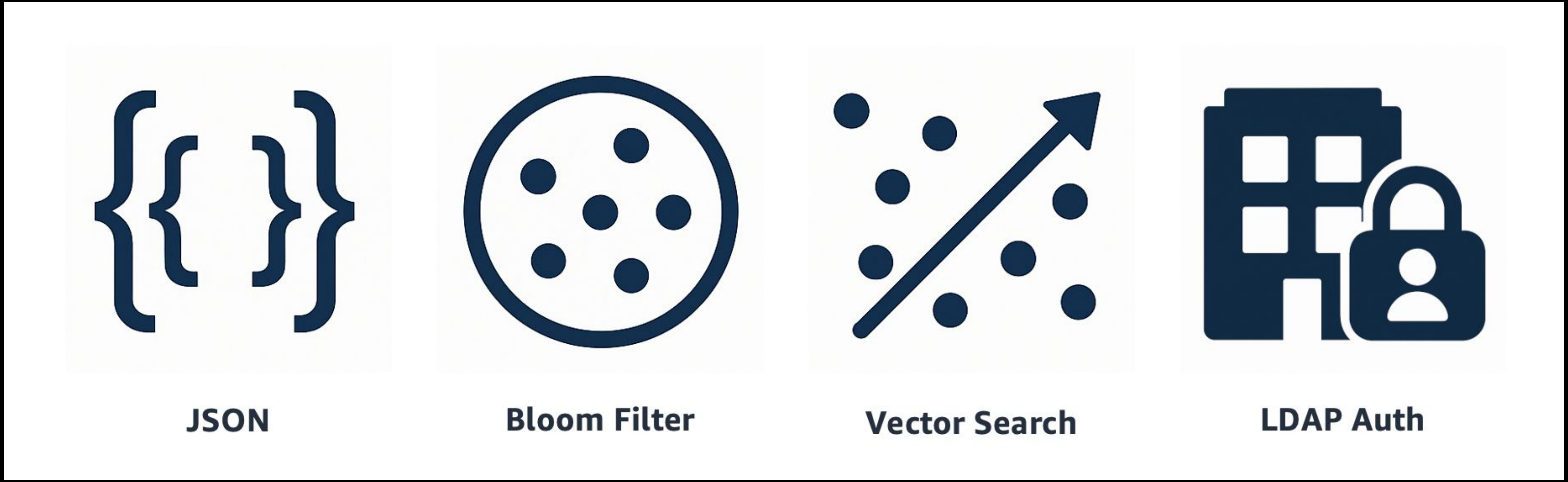
Grafana

OpenTelemetry

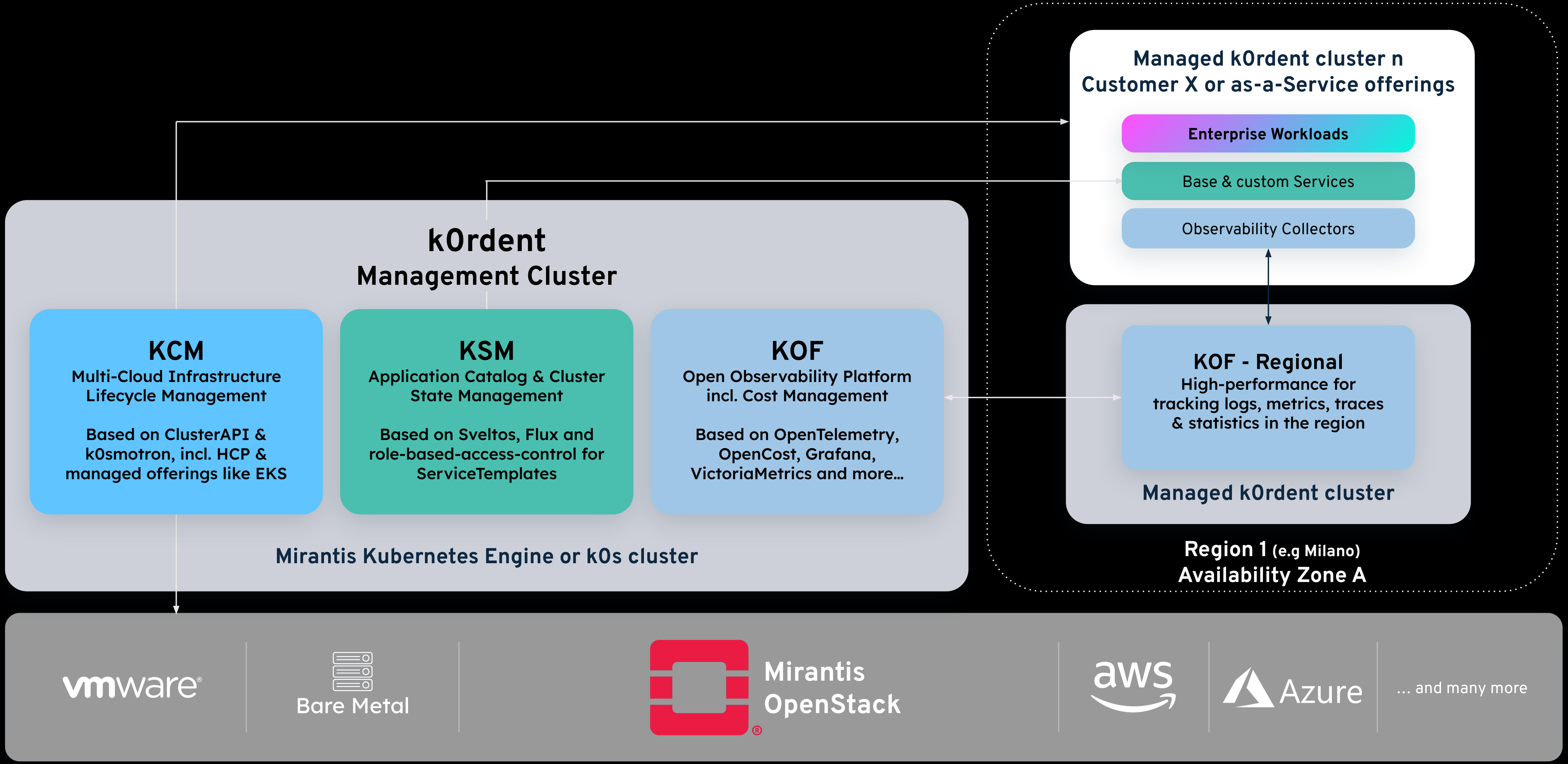
OpenCost

Promxy

Valkey Bundle Deployed using k0rdent



k0rdent
Architecture



Managed k0rdent cluster n
Customer X or as-a-Service offerings

Enterprise Workloads

Base & custom Services

Observability Collectors

KOF - Regional
High-performance for tracking logs, metrics, traces & statistics in the region

Managed k0rdent cluster

Region 1 (e.g Milano)
Availability Zone A

k0rdent
Management Cluster

KCM
Multi-Cloud Infrastructure Lifecycle Management

Based on ClusterAPI & k0smotron, incl. HCP & managed offerings like EKS

KSM
Application Catalog & Cluster State Management

Based on Sveltos, Flux and role-based-access-control for ServiceTemplates

KOF
Open Observability Platform incl. Cost Management

Based on OpenTelemetry, OpenCost, Grafana, VictoriaMetrics and more…

Mirantis Kubernetes Engine or k0s cluster

k0rdent pre-defined ServiceTemplates

k0rdent provides a set of pre-defined ServiceTemplates for commonly used beach-head services. These templates make it easy for platform teams to deploy essential services across their managed clusters.

Certificate Management:

cert-manager: Automates the issuance and renewal of TLS certificates.
Template: cert-manager-<version>

Backup and Disaster Recovery:

Velero: Backup and restore solution for Kubernetes resources and persistent volumes.
Template: velero-<version>

Service Mesh:

Istio: Connect, secure, control, and observe services in a microservices architecture.
Template: istio-<version>

GitOps and Continuous Delivery:

FluxCD: Automates the deployment of applications using Git as the single source of truth.
Template: fluxcd-<version>

Ingress Controller:

NGINX Ingress Controller: Enables external access to services in the cluster.
Template: ingress-nginx-<version>

Security and Policy Enforcement:

Kyverno: Kubernetes-native policy engine for enforcing security best practices and governance.
Template: kyverno-<version>

Monitoring and Logging:

Prometheus: Powerful monitoring system and time series database.
Grafana: Interactive visualization and analytics platform for metrics.
Fluent Bit: Lightweight log processor and forwarder.
Templates: prometheus-<version>, grafana-<version>, fluent-bit-<version>

Platform teams can use these templates directly in their ClusterDeployment or MultiClusterService resources to consistently install and manage beach-head services across their clusters. k0rdent's beach-head service templates streamline the process of setting up essential infrastructure components, enabling teams to focus on delivering applications and value to their users.

KSM ServiceTemplates

KSM uses ServiceTemplates that define how services (runtime state) like ingress controllers, monitoring tools, etc, should be deployed across clusters or namespaces.

Deployment Methods:

Helm charts
Raw Kubernetes manifests
Operator-based deployments

Version Control:

Immutable versions
Upgrade paths
Dependency management

Configuration Management:

Templated values
Environment overrides
Default configurations

Health Management:

Readiness/liveness probes
Resource requirements
Monitoring integration

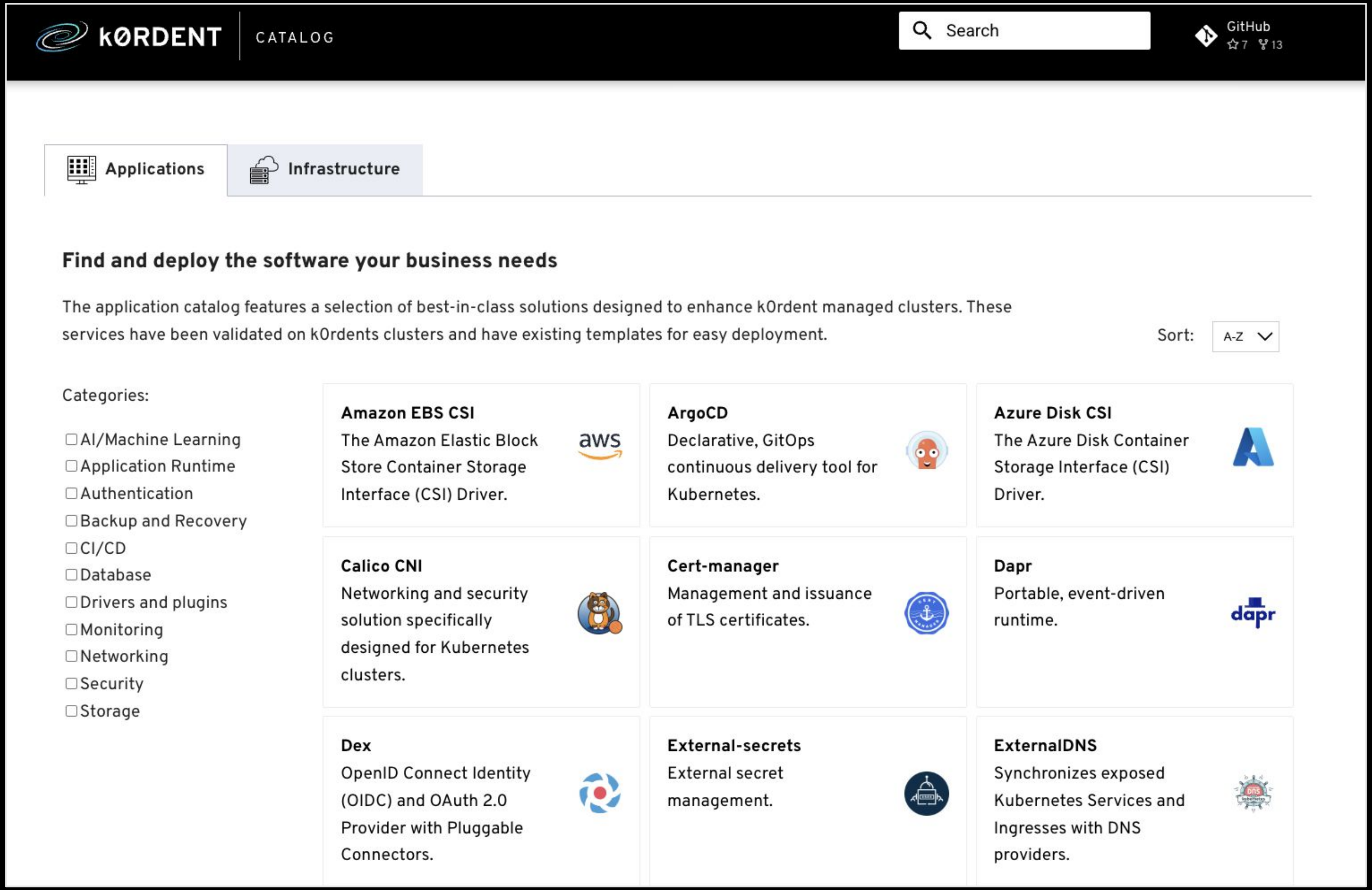
ServiceTemplates can be used in two main ways, via ClusterDeployment and MulticlusterService. This dual-use capability makes ServiceTemplates powerful for managing both cluster-wide and namespace-scoped services consistently across your k0rdent-managed infrastructure.

```yaml
# example Service Template yaml

apiVersion: templates.hmc.mirantis.com/v1alpha1
kind: ServiceTemplate
metadata:
  name: ingress-controller-template
spec:
  # Service identification and versioning
  serviceName: "nginx-ingress"
  version: "1.2.0"
  displayName: "NGINX Ingress Controller"
  description: "Production-grade Ingress Controller"

  # Template type and source
  type: "helm"  # Options: helm, manifest, operator
  source:
    helm:
      repository:
"https://kubernetes.github.io/ingress-nginx"
      chart: "ingress-nginx"
      version: "4.7.1"
      values:
        controller:
          replicaCount: 2
```

k0rdent Catalog provides choice of Applications



Find and deploy the software your business needs

The application catalog features a selection of best-in-class solutions designed to enhance k0rdent managed clusters. These services have been validated on k0rdents clusters and have existing templates for easy deployment.

Sort: A-Z

Categories:

- □AI/Machine Learning
- □Application Runtime
- □Authentication
- □Backup and Recovery
- □CI/CD
- □Database
- □Drivers and plugins
- □Monitoring
- □Networking
- □Security
- □Storage

Amazon EBS CSI
The Amazon Elastic Block Store Container Storage Interface (CSI) Driver.

ArgoCD
Declarative, GitOps continuous delivery tool for Kubernetes.

Azure Disk CSI
The Azure Disk Container Storage Interface (CSI) Driver.

Calico CNI
Networking and security solution specifically designed for Kubernetes clusters.

Cert-manager
Management and issuance of TLS certificates.

Dapr
Portable, event-driven runtime.

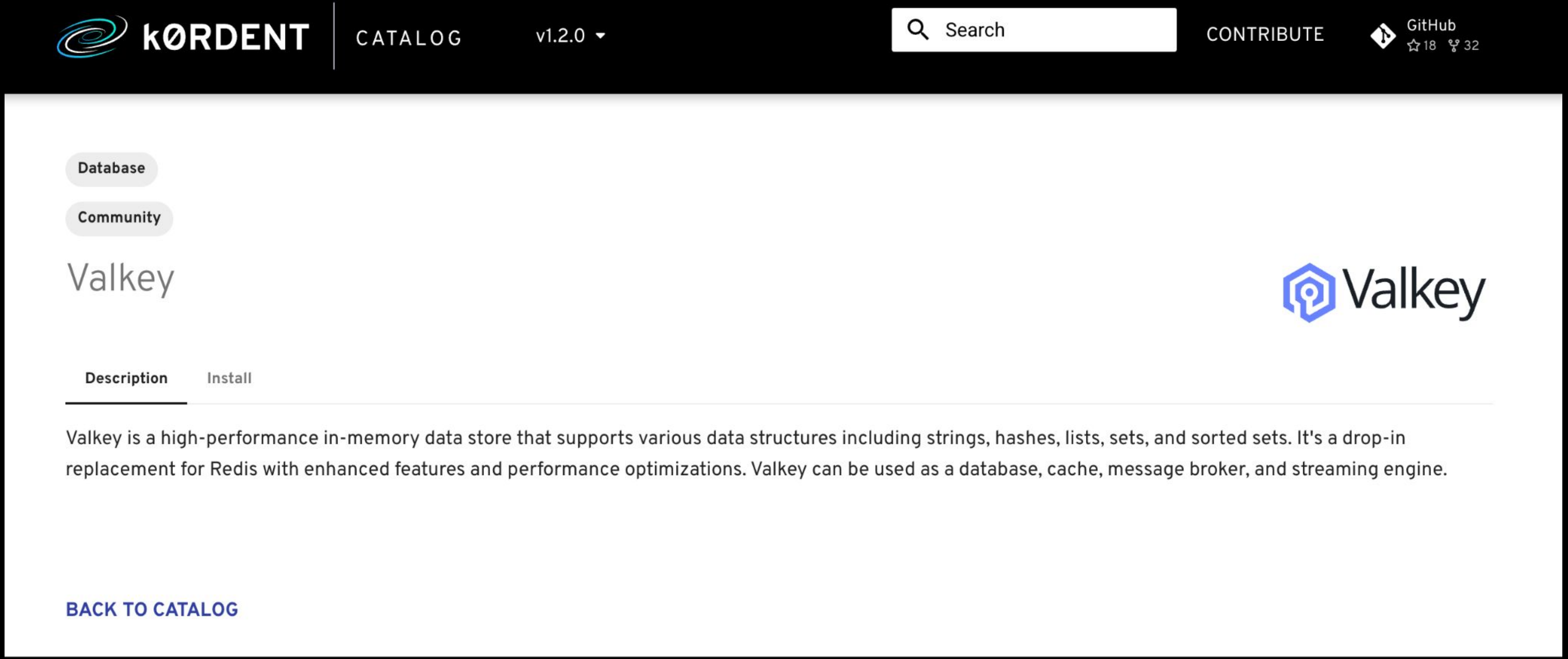
Dex
OpenID Connect Identity (OIDC) and OAuth 2.0 Provider with Pluggable Connectors.

External-secrets
External secret management.

ExternalDNS
Synchronizes exposed Kubernetes Services and Ingresses with DNS providers.

- Open source ecosystem
- Pre-validated integrations
- Easy to deploy with available templates

Valkey Template on k0rdent Catalog



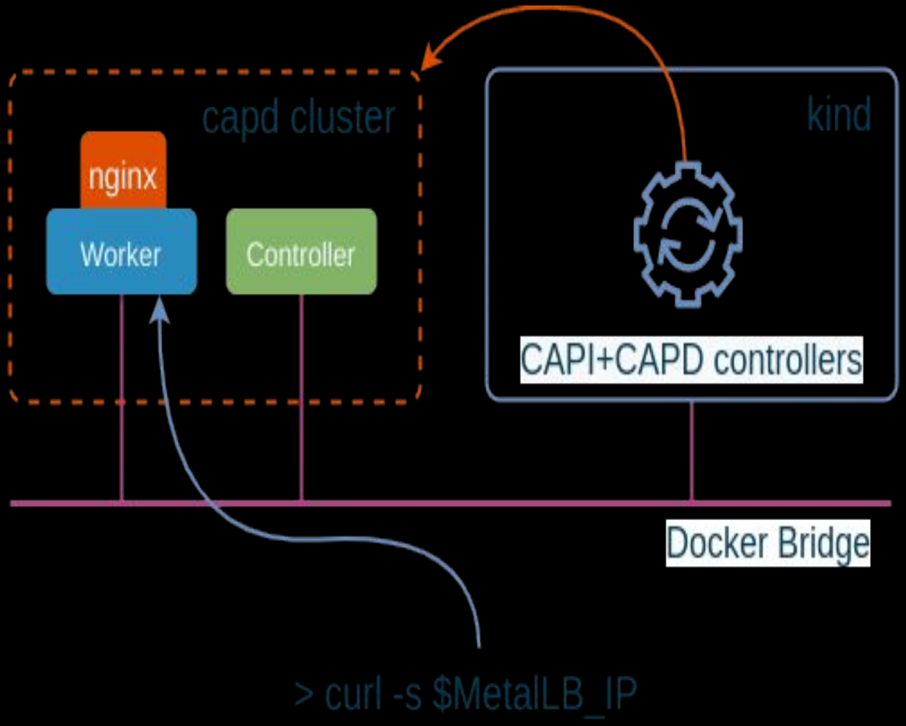
Valkey is a high-performance in-memory data store that supports various data structures including strings, hashes, lists, sets, and sorted sets. It's a drop-in replacement for Redis with enhanced features and performance optimizations. Valkey can be used as a database, cache, message broker, and streaming engine.

kØRDENT CATALOG

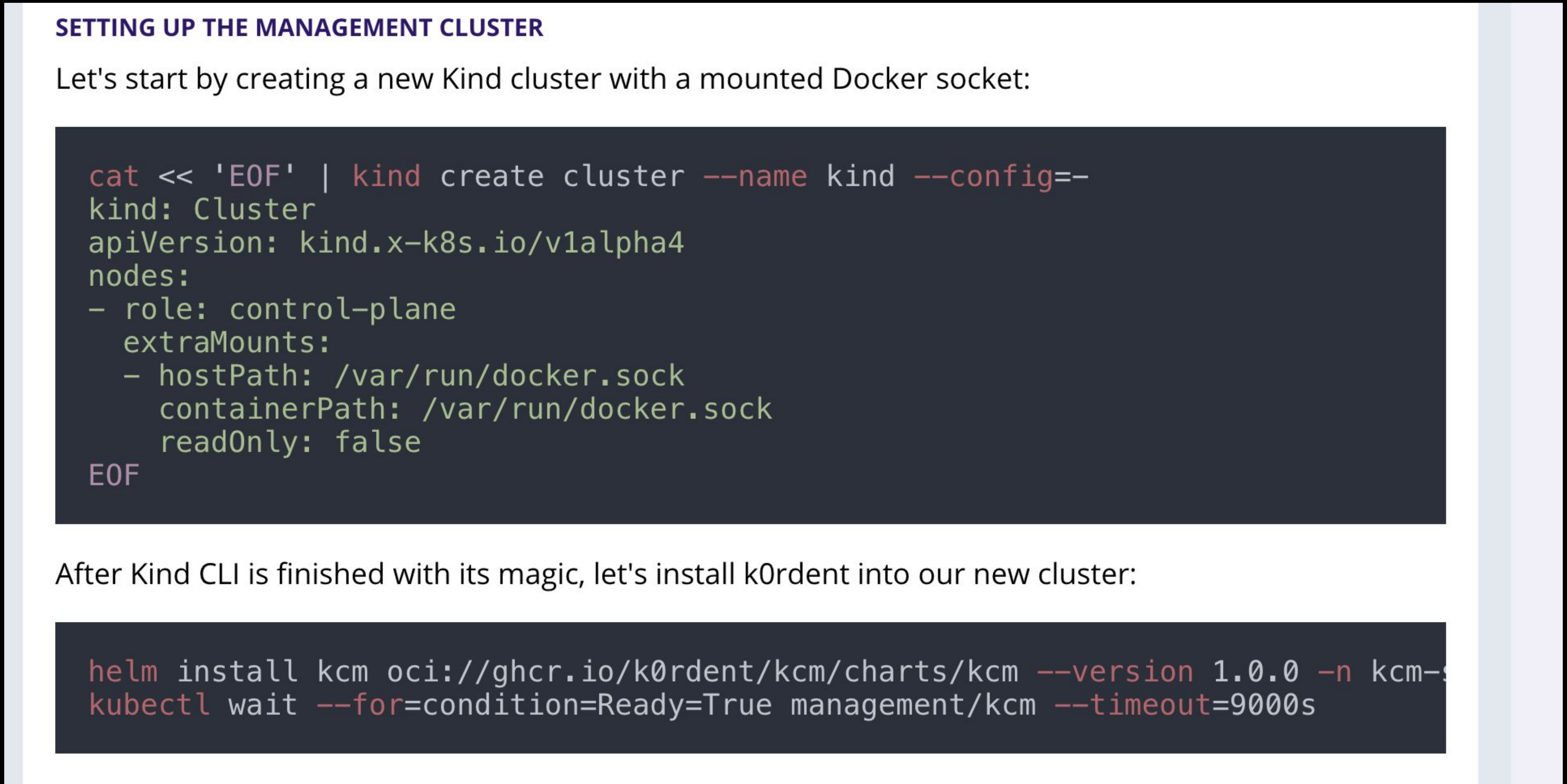
Pre-Requisites

- Have a kind Cluster Handy
- Deploy using CAPI Provider for Docker

- Use Hyperspike's Valkey Operator



Set up the Management Cluster



SETTING UP THE MANAGEMENT CLUSTER

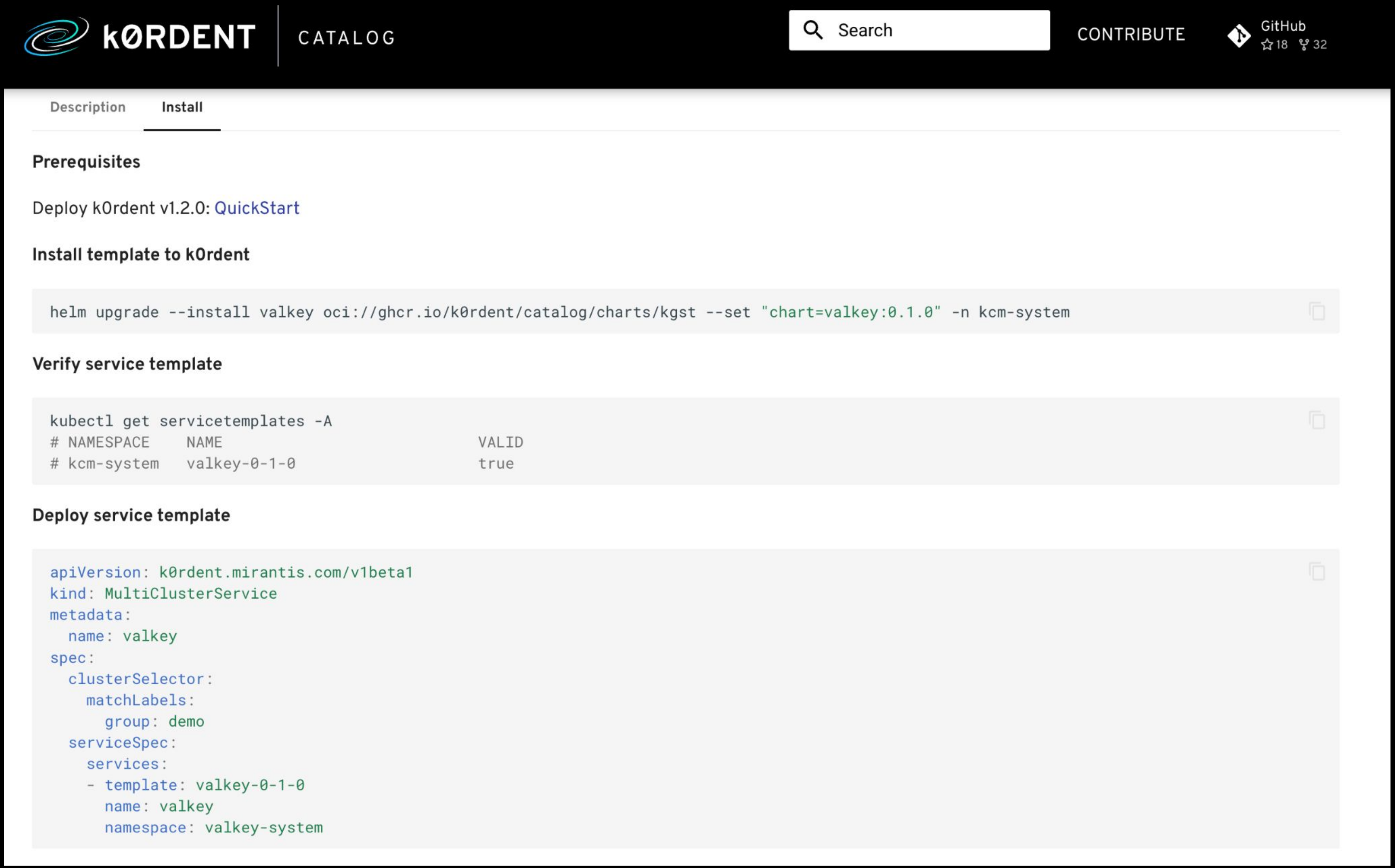
Let's start by creating a new Kind cluster with a mounted Docker socket:

```
cat << 'EOF' | kind create cluster --name kind --config=-
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraMounts:
  - hostPath: /var/run/docker.sock
    containerPath: /var/run/docker.sock
    readOnly: false
EOF
```

After Kind CLI is finished with its magic, let's install k0rdent into our new cluster:

```
helm install kcm oci://ghcr.io/k0rdent/kcm/charts/kcm --version 1.0.0 -n kcm-s
kubectl wait --for=condition=Ready=True management/kcm --timeout=9000s
```

Install the servicetemplate



Prerequisites

Deploy kOrdent v1.2.0: QuickStart

Install template to kOrdent

```
helm upgrade --install valkey oci://ghcr.io/k0rdent/catalog/charts/kgst --set "chart=valkey:0.1.0" -n kcm-system
```

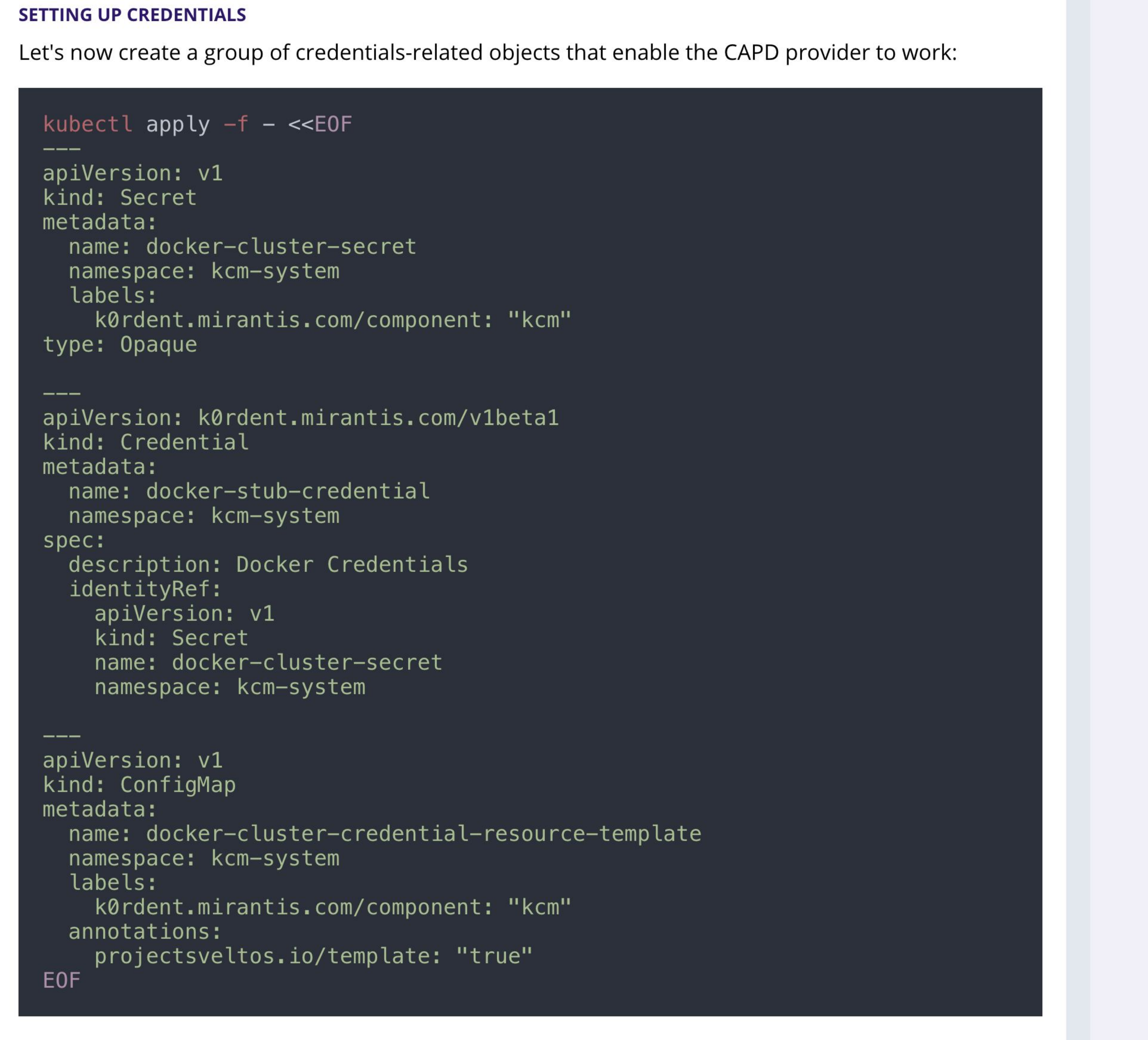
Verify service template

```
kubectl get servicetemplates -A
# NAMESPACE    NAME                             VALID
# kcm-system   valkey-0-1-0                     true
```

Deploy service template

```
apiVersion: k0rdent.mirantis.com/v1beta1
kind: MultiClusterService
metadata:
  name: valkey
spec:
  clusterSelector:
    matchLabels:
      group: demo
  serviceSpec:
    services:
    - template: valkey-0-1-0
      name: valkey
      namespace: valkey-system
```

Setup Credentials



SETTING UP CREDENTIALS

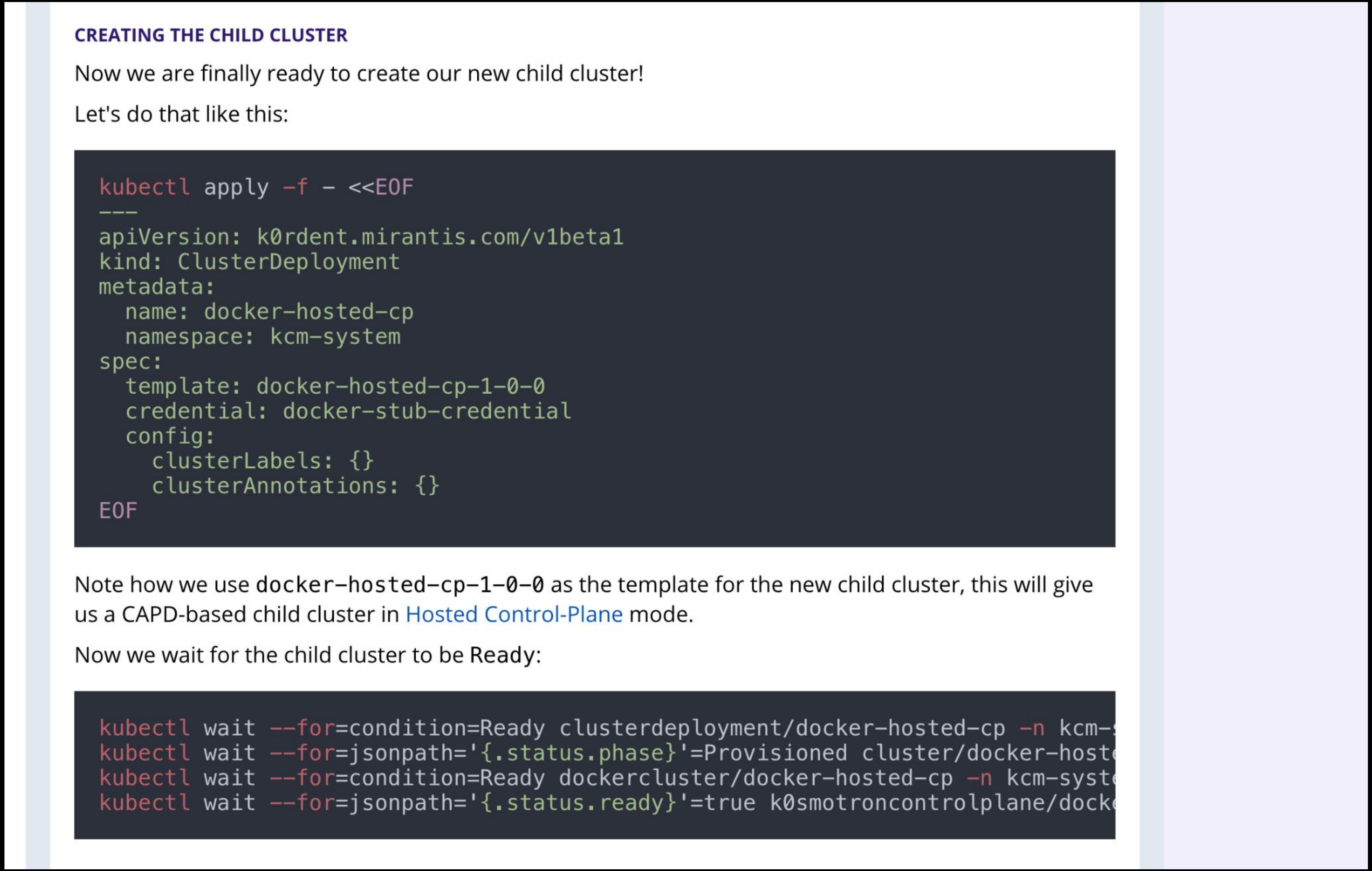
Let's now create a group of credentials-related objects that enable the CAPD provider to work:

```
kubectl apply -f - <<EOF
---
apiVersion: v1
kind: Secret
metadata:
  name: docker-cluster-secret
  namespace: kcm-system
  labels:
    k0rdent.mirantis.com/component: "kcm"
type: Opaque

---
apiVersion: k0rdent.mirantis.com/v1beta1
kind: Credential
metadata:
  name: docker-stub-credential
  namespace: kcm-system
spec:
  description: Docker Credentials
  identityRef:
    apiVersion: v1
    kind: Secret
    name: docker-cluster-secret
    namespace: kcm-system

---
apiVersion: v1
kind: ConfigMap
metadata:
  name: docker-cluster-credential-resource-template
  namespace: kcm-system
  labels:
    k0rdent.mirantis.com/component: "kcm"
  annotations:
    projectsveltos.io/template: "true"
EOF
```

Creating and Verifying the Child Cluster



CREATING THE CHILD CLUSTER

Now we are finally ready to create our new child cluster!

Let's do that like this:

```
kubectl apply -f - <<EOF
---
apiVersion: k0rdent.mirantis.com/v1beta1
kind: ClusterDeployment
metadata:
  name: docker-hosted-cp
  namespace: kcm-system
spec:
  template: docker-hosted-cp-1-0-0
  credential: docker-stub-credential
  config:
    clusterLabels: {}
    clusterAnnotations: {}
EOF
```

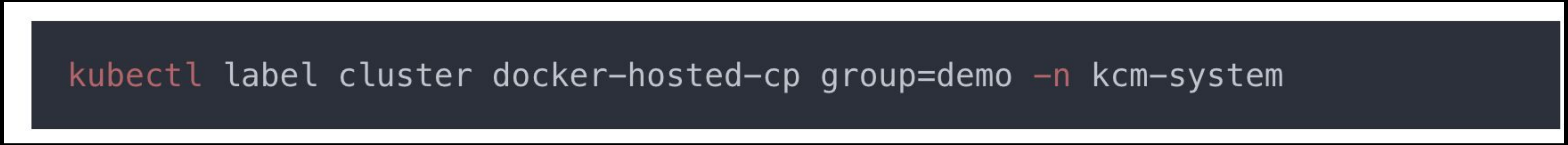
Note how we use `docker-hosted-cp-1-0-0` as the template for the new child cluster, this will give us a CAPD-based child cluster in Hosted Control-Plane mode.

Now we wait for the child cluster to be Ready:

```
kubectl wait --for=condition=Ready clusterdeployment/docker-hosted-cp -n kcm-
kubectl wait --for=jsonpath='{.status.phase}'=Provisioned cluster/docker-host
kubectl wait --for=condition=Ready dockercluster/docker-hosted-cp -n kcm-syst
kubectl wait --for=jsonpath='{.status.ready}'=true k0smotroncontrolplane/dock
```

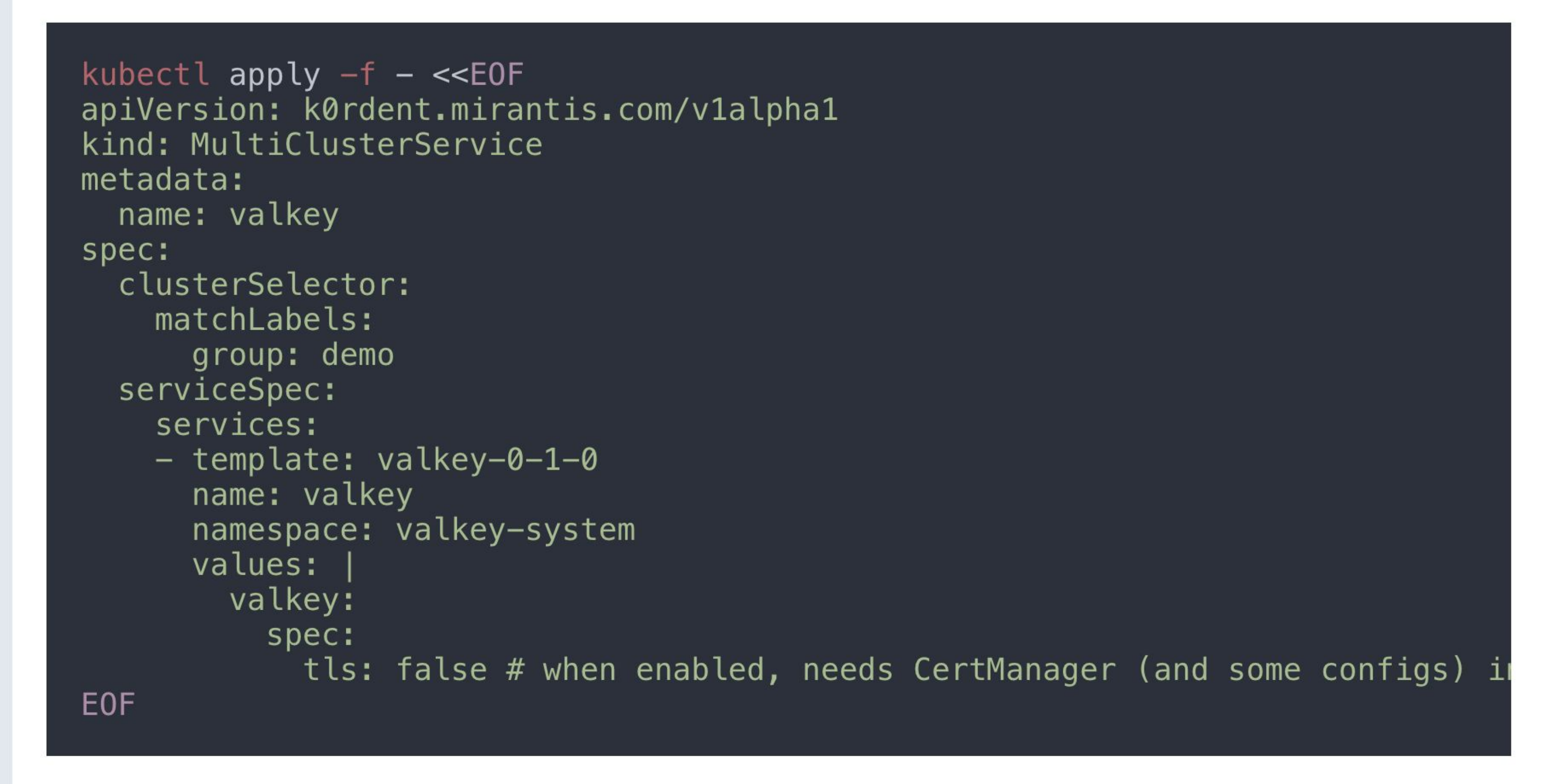
Deploying Valkey using MultiClusterService

first



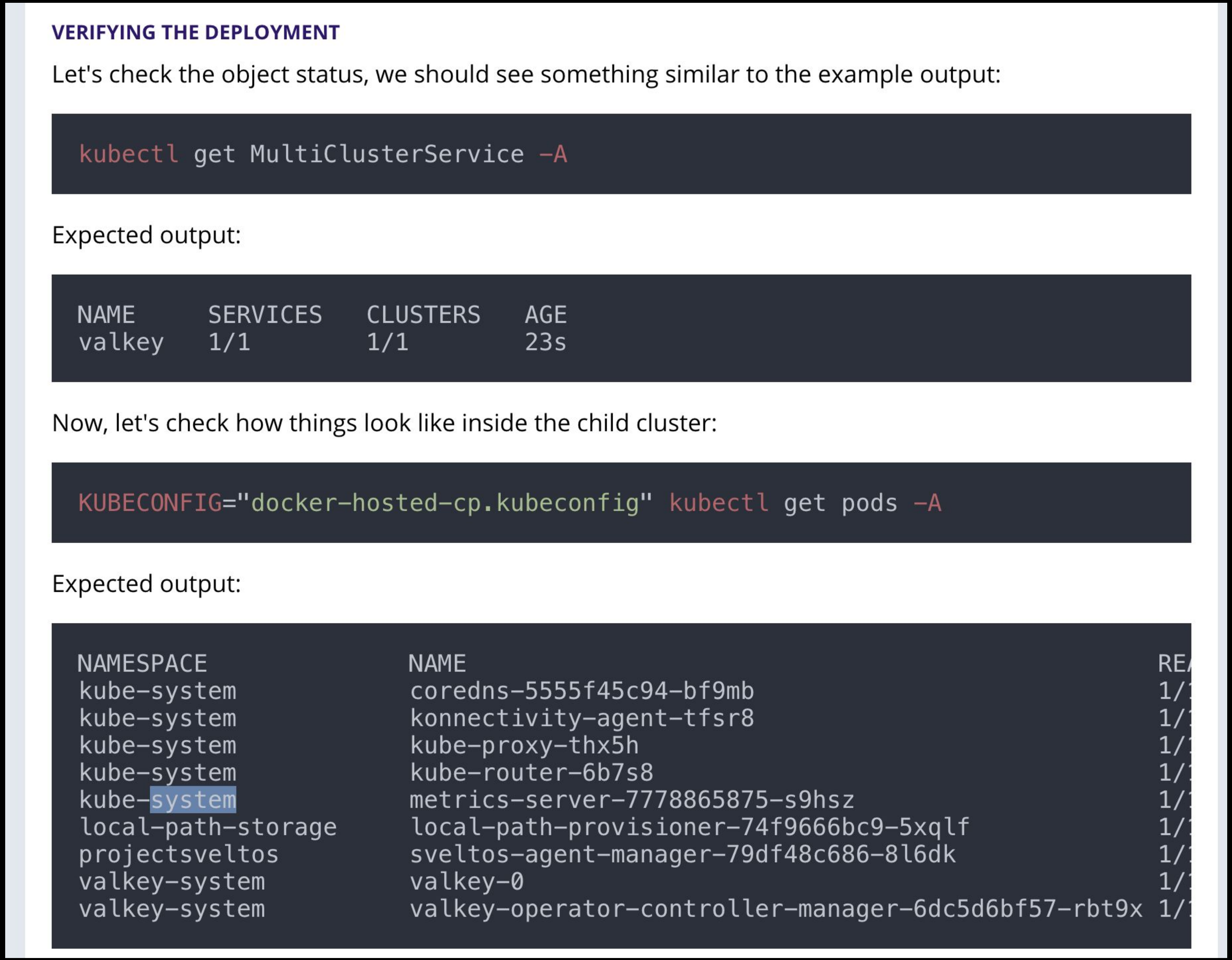
```
kubectl label cluster docker-hosted-cp group=demo -n kcm-system
```

then



```
kubectl apply -f - <<EOF
apiVersion: k0rdent.mirantis.com/v1alpha1
kind: MultiClusterService
metadata:
  name: valkey
spec:
  clusterSelector:
    matchLabels:
      group: demo
  serviceSpec:
    services:
    - template: valkey-0-1-0
      name: valkey
      namespace: valkey-system
      values: |
        valkey:
          spec:
            tls: false # when enabled, needs CertManager (and some configs) i
EOF
```

Once it is deployed



VERIFYING THE DEPLOYMENT

Let's check the object status, we should see something similar to the example output:

```
kubectl get MultiClusterService -A
```

Expected output:

```
NAME      SERVICES    CLUSTERS    AGE
valkey    1/1         1/1         23s
```

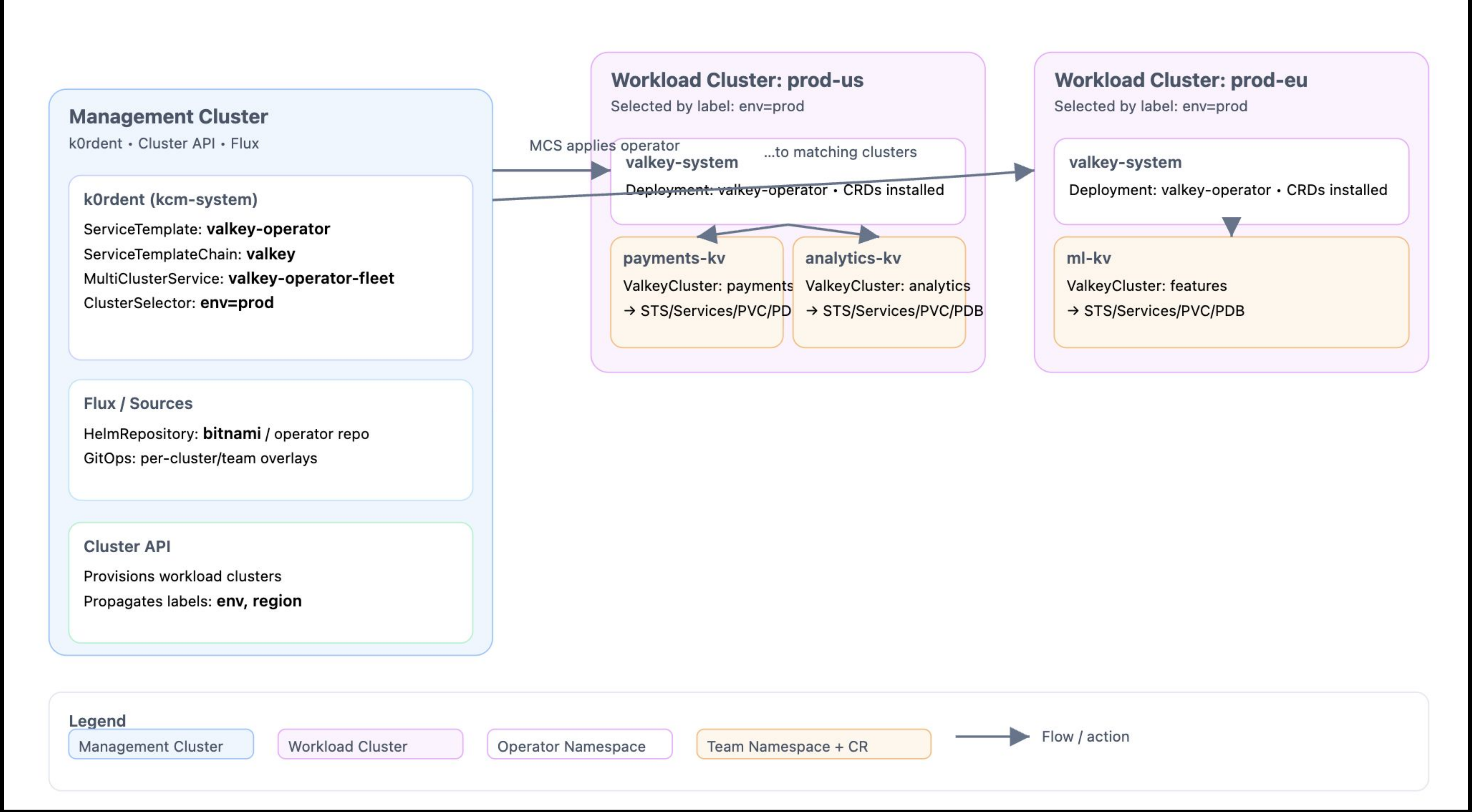
Now, let's check how things look like inside the child cluster:

```
KUBECONFIG="docker-hosted-cp.kubeconfig" kubectl get pods -A
```

Expected output:

```
NAMESPACE              NAME                                              RE
kube-system            coredns-5555f45c94-bf9mb                          1/
kube-system            konnectivity-agent-tfsr8                          1/
kube-system            kube-proxy-thx5h                                  1/
kube-system            kube-router-6b7s8                                 1/
kube-system            metrics-server-7778865875-s9hsz                   1/
local-path-storage     local-path-provisioner-74f9666bc9-5xqlf           1/
projectsveltos         sveltos-agent-manager-79df48c686-8l6dk            1/
valkey-system          valkey-0                                          1/
valkey-system          valkey-operator-controller-manager-6dc5d6bf57-rbt9x 1/
```

How it works



Management Cluster

k0rdent • Cluster API • Flux

k0rdent (kcm-system)

ServiceTemplate: **valkey-operator**
ServiceTemplateChain: **valkey**
MultiClusterService: **valkey-operator-fleet**
ClusterSelector: **env=prod**

Flux / Sources

HelmRepository: **bitnami** / operator repo
GitOps: per-cluster/team overlays

Cluster API

Provisions workload clusters
Propagates labels: **env, region**

MCS applies operator

...to matching clusters

Workload Cluster: prod-us

Selected by label: env=prod

valkey-system

Deployment: valkey-operator • CRDs installed

payments-kv

ValkeyCluster: payments
→ STS/Services/PVC/PD

analytics-kv

ValkeyCluster: analytics
→ STS/Services/PVC/PDB

Workload Cluster: prod-eu

Selected by label: env=prod

valkey-system

Deployment: valkey-operator • CRDs installed

ml-kv

ValkeyCluster: features
→ STS/Services/PVC/PDB

Legend

Management Cluster

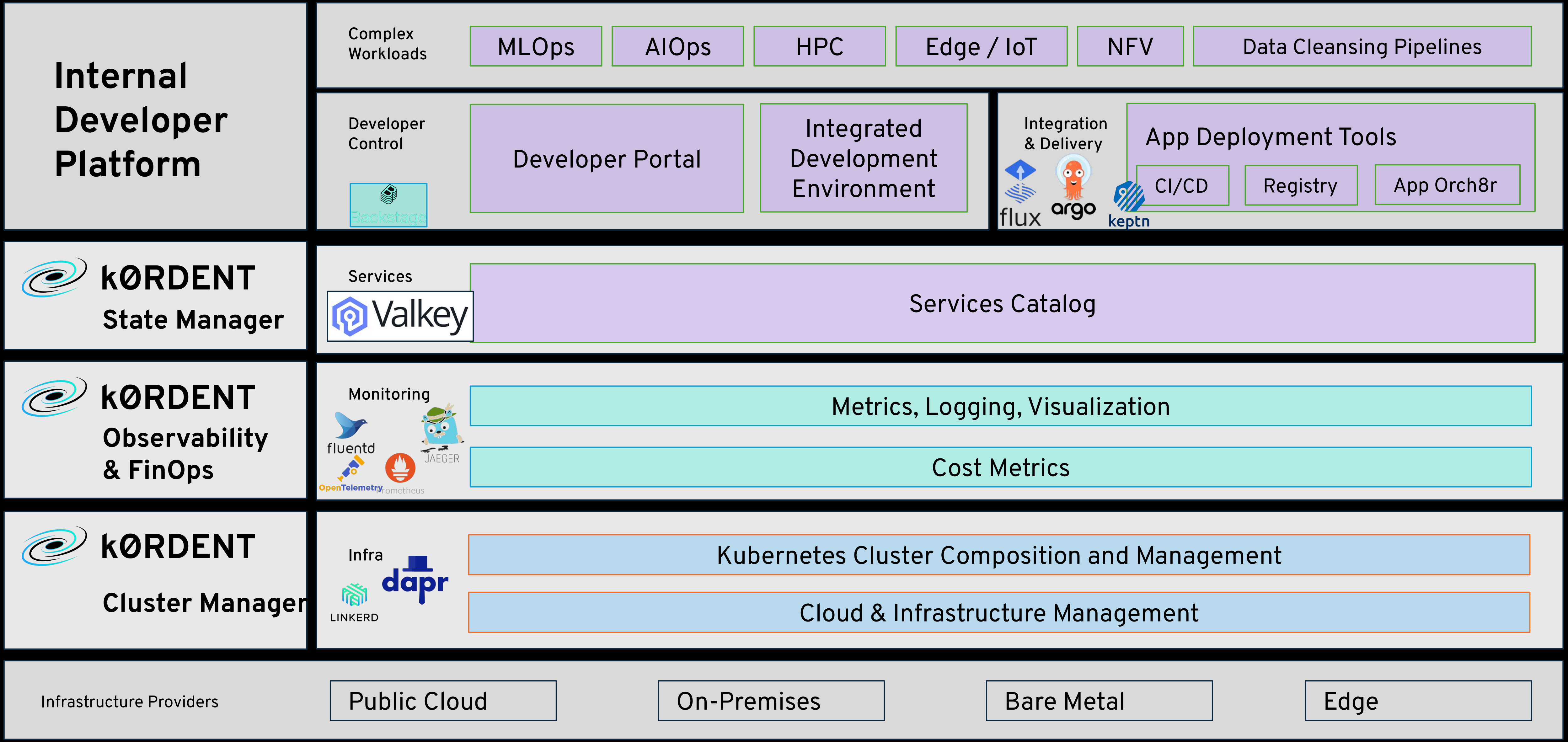
Workload Cluster

Operator Namespace

Team Namespace + CR

→ Flow / action

Streamline Creation & Maintenance of IDPs



Internal Developer Platform

Complex Workloads: MLOps | AIOps | HPC | Edge / IoT | NFV | Data Cleansing Pipelines

Developer Control: Developer Portal | Integrated Development Environment

Backstage

Integration & Delivery: App Deployment Tools — CI/CD | Registry | App Orch8r

flux argo keptn

kØRDENT State Manager

Services: Services Catalog

Valkey

kØRDENT Observability & FinOps

Monitoring: Metrics, Logging, Visualization | Cost Metrics

fluentd JAEGER OpenTelemetry Prometheus

kØRDENT Cluster Manager

Infra: Kubernetes Cluster Composition and Management | Cloud & Infrastructure Management

LINKERD dapr

Infrastructure Providers: Public Cloud | On-Premises | Bare Metal | Edge

IT'S TIME TO GO

MIRANTIS

Drop a “⭐” on Github:
https://github.com/k0rdent/k0rdent

Thank You!

Copyright © 2024 Mirantis, Inc. All rights reserved