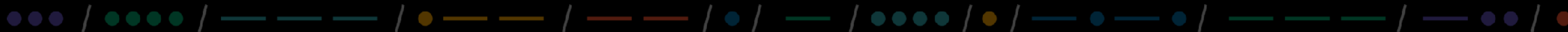


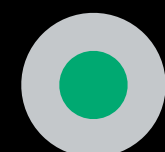
带你了解 KubeSphere

万宏明

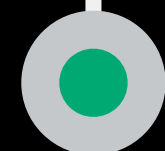
KubeSphere 研发工程师



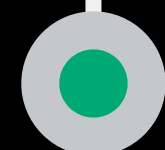
分享大纲



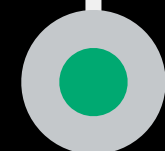
KubeSphere 项目简介



KubeSphere 的开源生态



KubeSphere 的核心功能及架构



KubeSphere 社区贡献方式

Part 1

KubeSphere 项目简介

1.1 什么是KubeSphere

面向云原生应用的容器混合云

KubeSphere 是在 Kubernetes 之上构建的以应用为中心的多租户容器平台，提供全栈的 IT 自动化运维的能力，简化企业的 DevOps 工作流。KubeSphere 提供了运维友好的向导式操作界面，帮助企业快速构建一个强大和功能丰富的容器云平台。

KubeSphere 愿景是打造一个以 Kubernetes 为内核的云原生分布式操作系统。



完全开源

通过 CNCF 一致性认证的
Kubernetes 平台，100% 开源，由
社区驱动与开发



简易安装

支持部署在任何基础设施环境，提
供在线与离线安装，支持一键升级
与扩容集群



功能丰富

在一个平台统一纳管 DevOps、云
原生可观测性、服务网格、应用生
命周期、多租户、多集群、存储与
网络



模块化 & 可插拔

平台中的所有功能都是可插拔与松
耦合，您可以根据业务场景可选安
装所需功能组件

1.2 主要功能



应用商店

提供基于 Helm 的应用商店与应用命周期管理



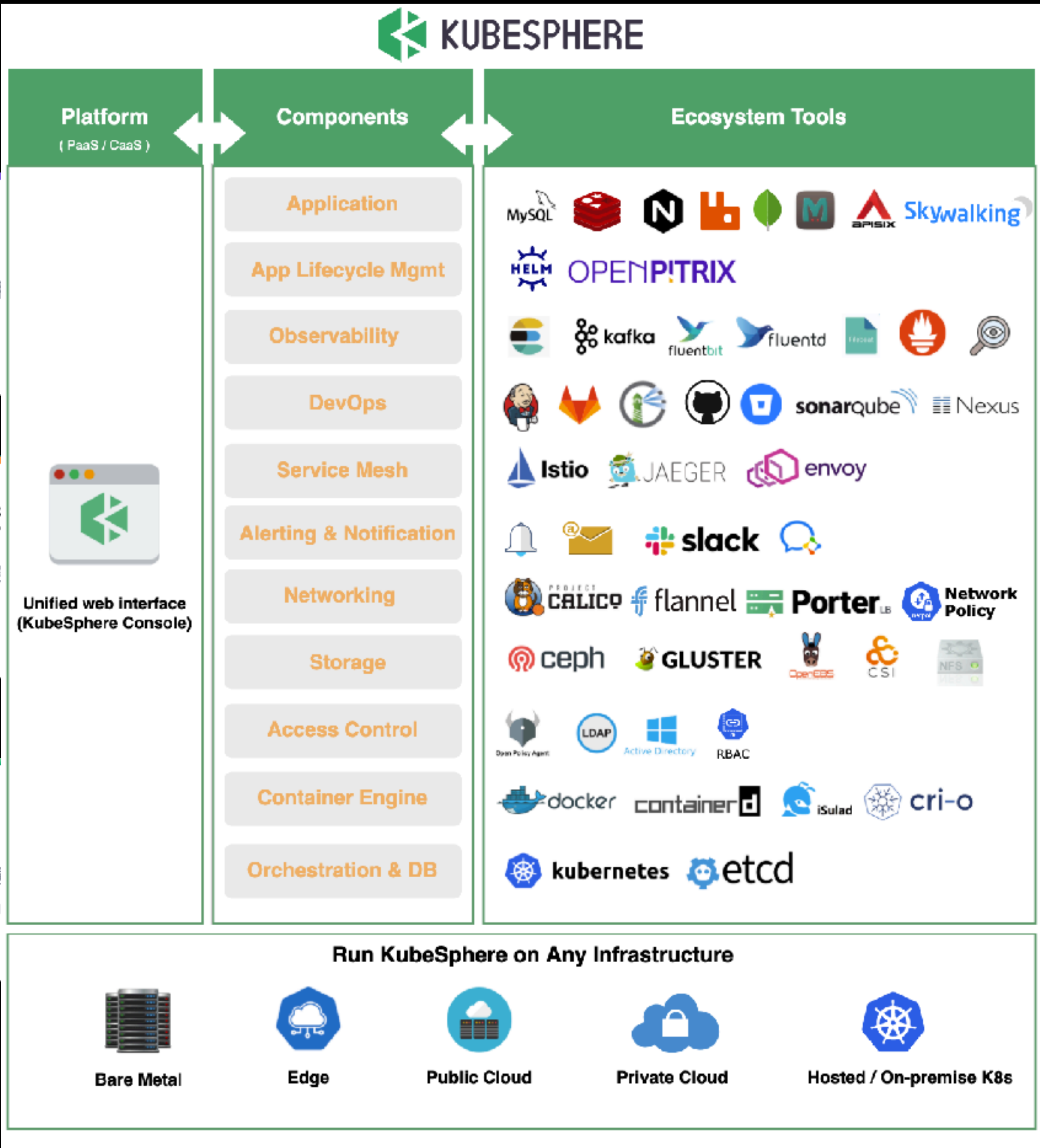
基于 Istio 的微服务

提供细粒度的流量管理、流量监控的流量拓扑



网络管理

提供面向物理机 Kubernetes 环境可视化管理，支持 Calico 与 Flannel



具备的最佳实践，支持

，支持多种告警策略

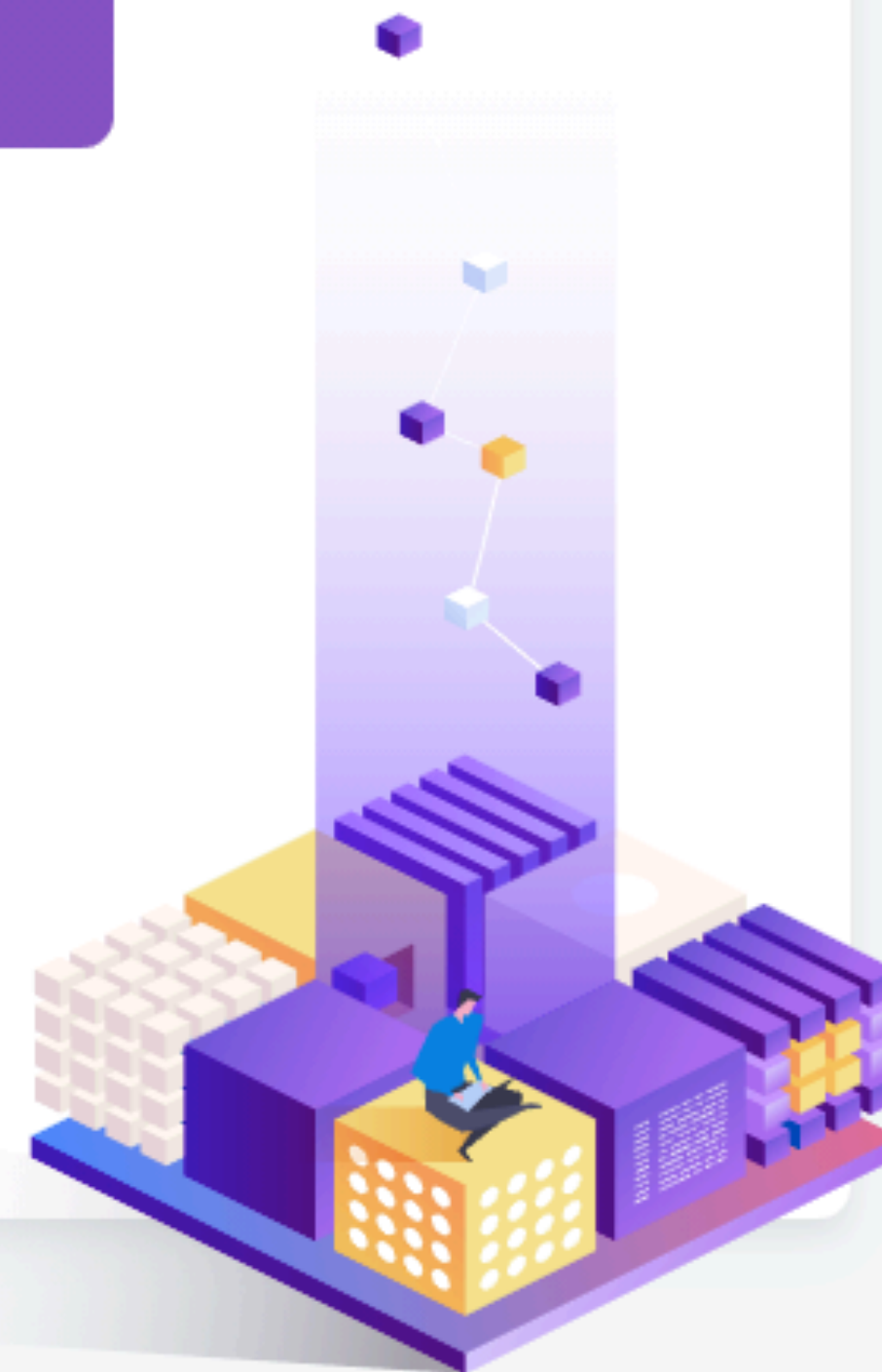
多个 CSI 插件对接使

1.3 核心价值

👤 基础设施团队

实现从云端到数据中心自动化部署、扩容与升级集群

- 提高资源利用率，减少内部基础设施的成本支出
- 提供安全增强，支持多种存储与网络方案
- 为企业交付一个通过 CNCF 认证和可信赖的 Kubernetes 发行版
- 支持 Kubernetes 的多云与多集群管理，提供多云与多可用区的高可用

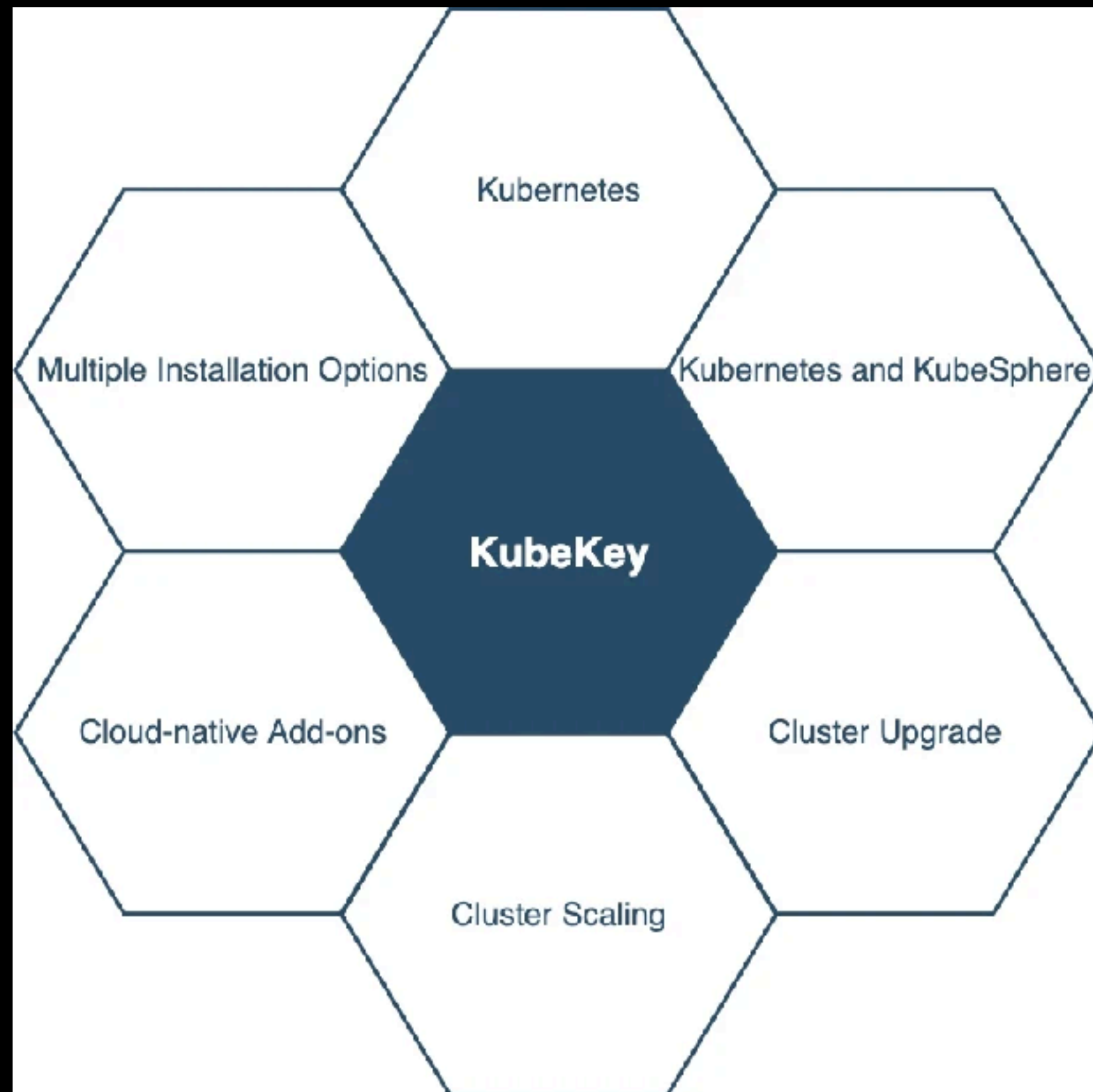


Part 2

KubeSphere 的开源生态

2.1 KubeKey

KubeKey 是由 KubeSphere 衍生的 K8s 集群安装工具。它提供了一个即插即用的架构，可以无缝集成第三方组件。



核心功能：

- 安装 Kubernetes
- 安装 KubeSphere
- Kubernetes 集群伸缩
- Kubernetes 集群升级
- 安装 Kubernetes 相关的插件（Chart 或 YAML）

理念： Cluster as an Object

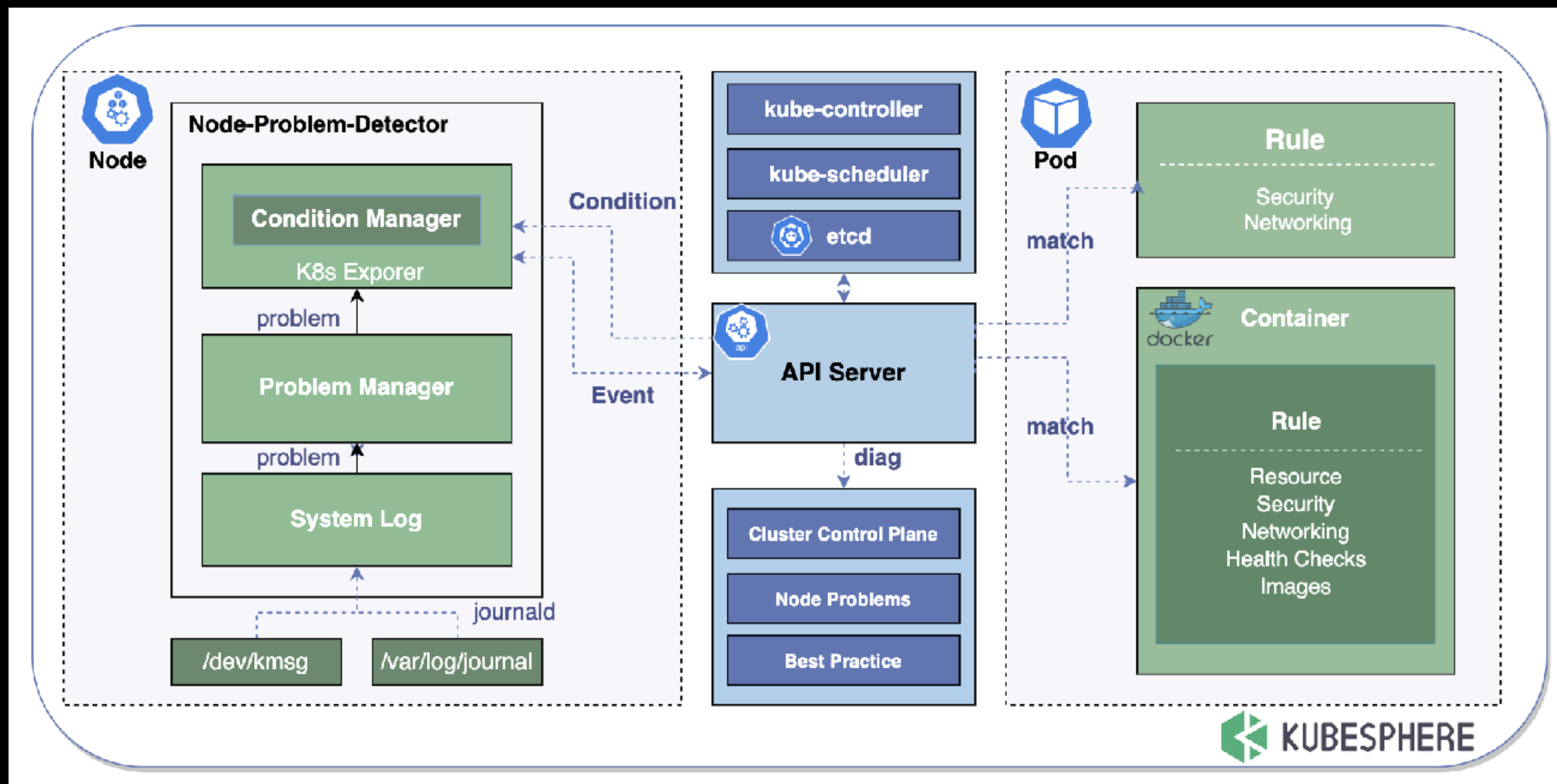
2.3 KubeEye



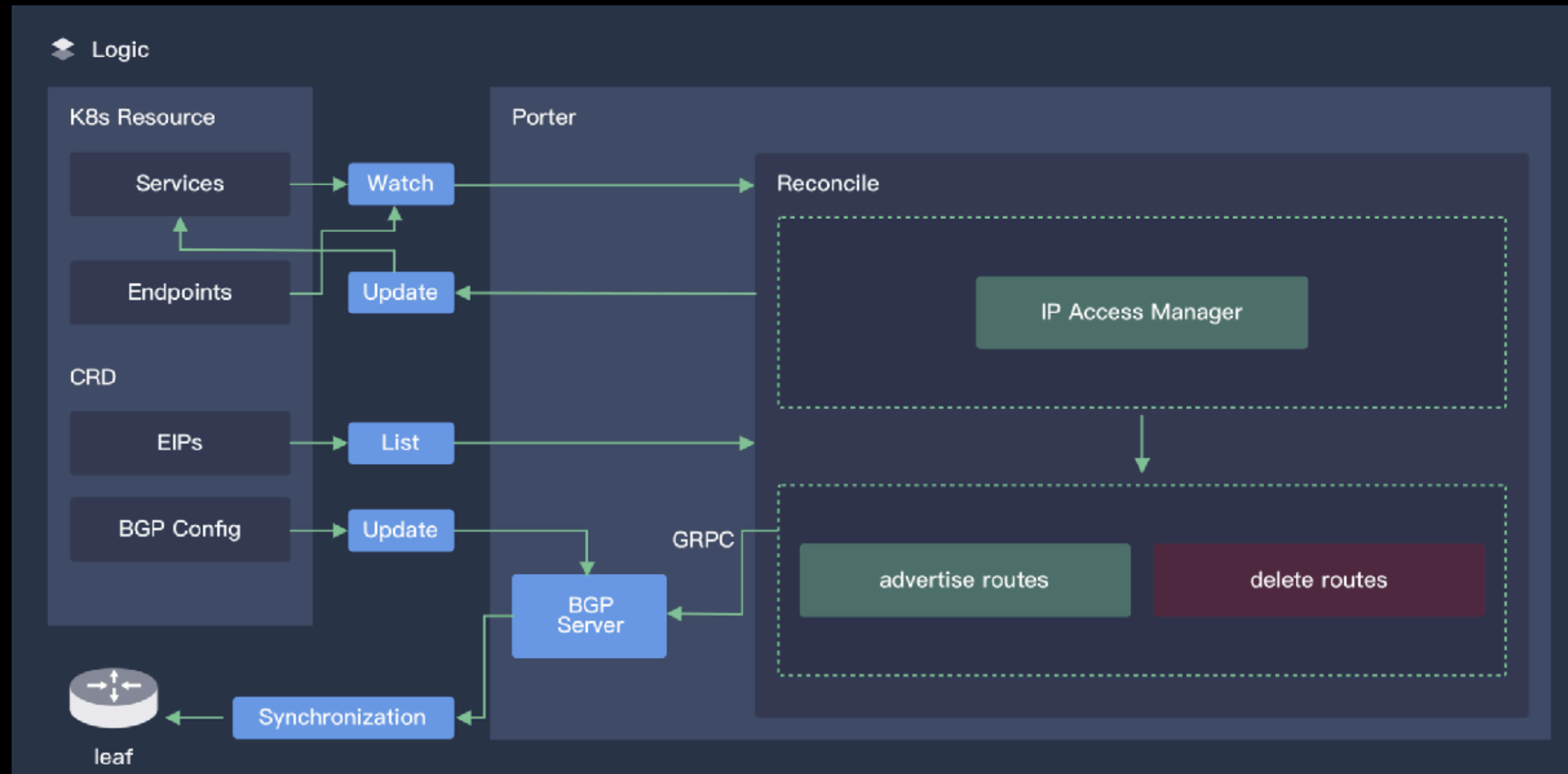
KubeEye 是一个 K8s 集群巡检工具

核心功能：

- 检测 K8s 控制平面的异常
- 检测节点异常
- 检测资源 YAML 规范



2.3 PorterLB



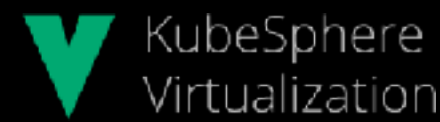
PorterLB

是一个裸金属环境负载均衡器实现

核心功能:

1. 基于 BGP 路由动态配置
2. 基于路由器 ECMP 的负载均衡
3. VIP 管理

2.4 产品家族



虚拟机&HCI

超轻量虚拟机管理平台，单节点起步
虚拟化业务支撑、边缘设备对接等



集群健康&安全巡检

细粒度插件化集群自动巡检



硬负载均衡

开源分布式硬负载均衡



跨平台云原生应用管理

支持跨云应用分发管理
支持 KubeEdge、EdgeWize 分发应用



KubeOcean

轻量化集群管理

支持基于 kind 快速和管理轻量化
容器节点集群

KubeKey

云原生环境交付引擎

插件化可插拔交付框架



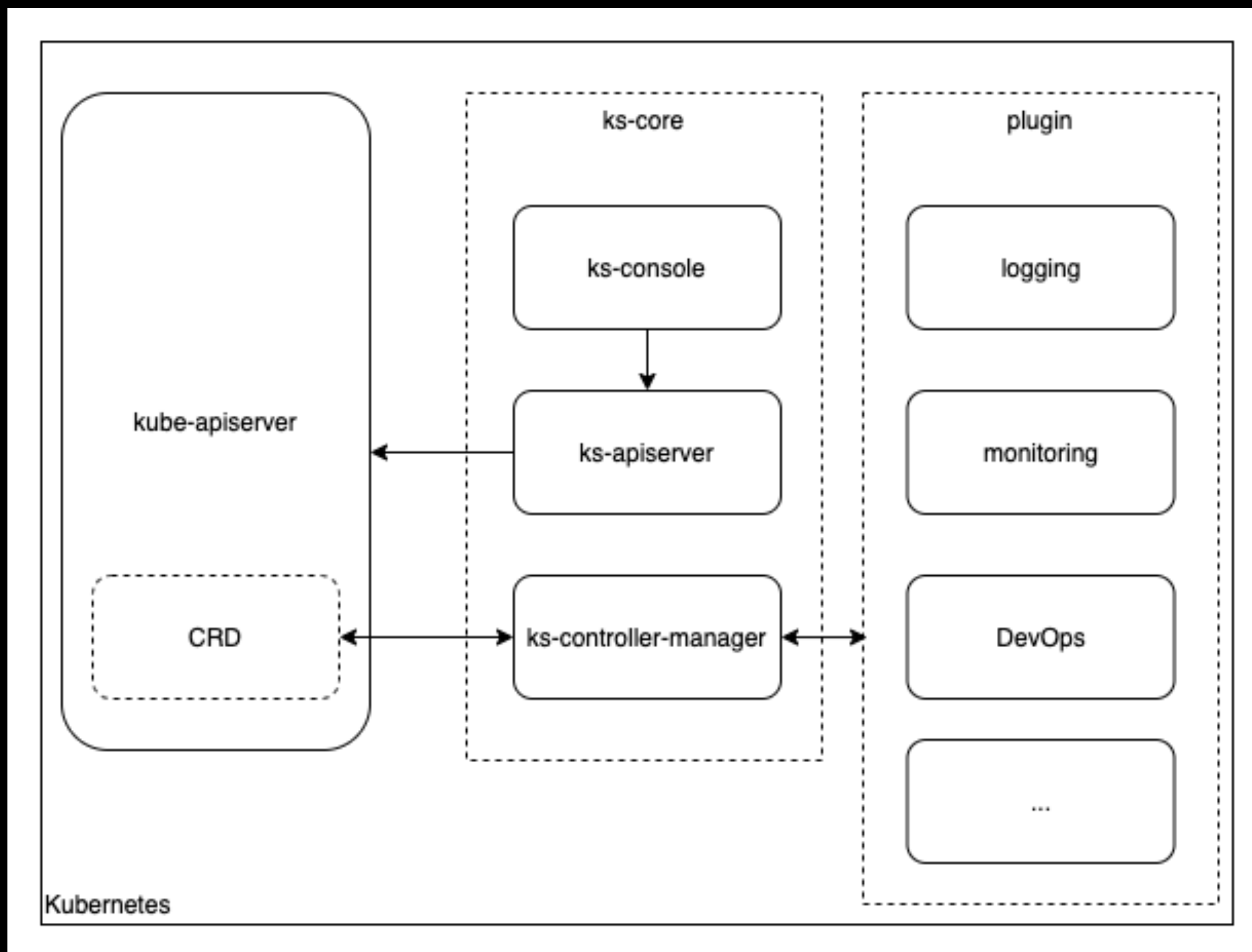
FaaS 开放框架

基于 Knative 的以事件驱动的、
跨基础设施的 FaaS 产品

Part 3

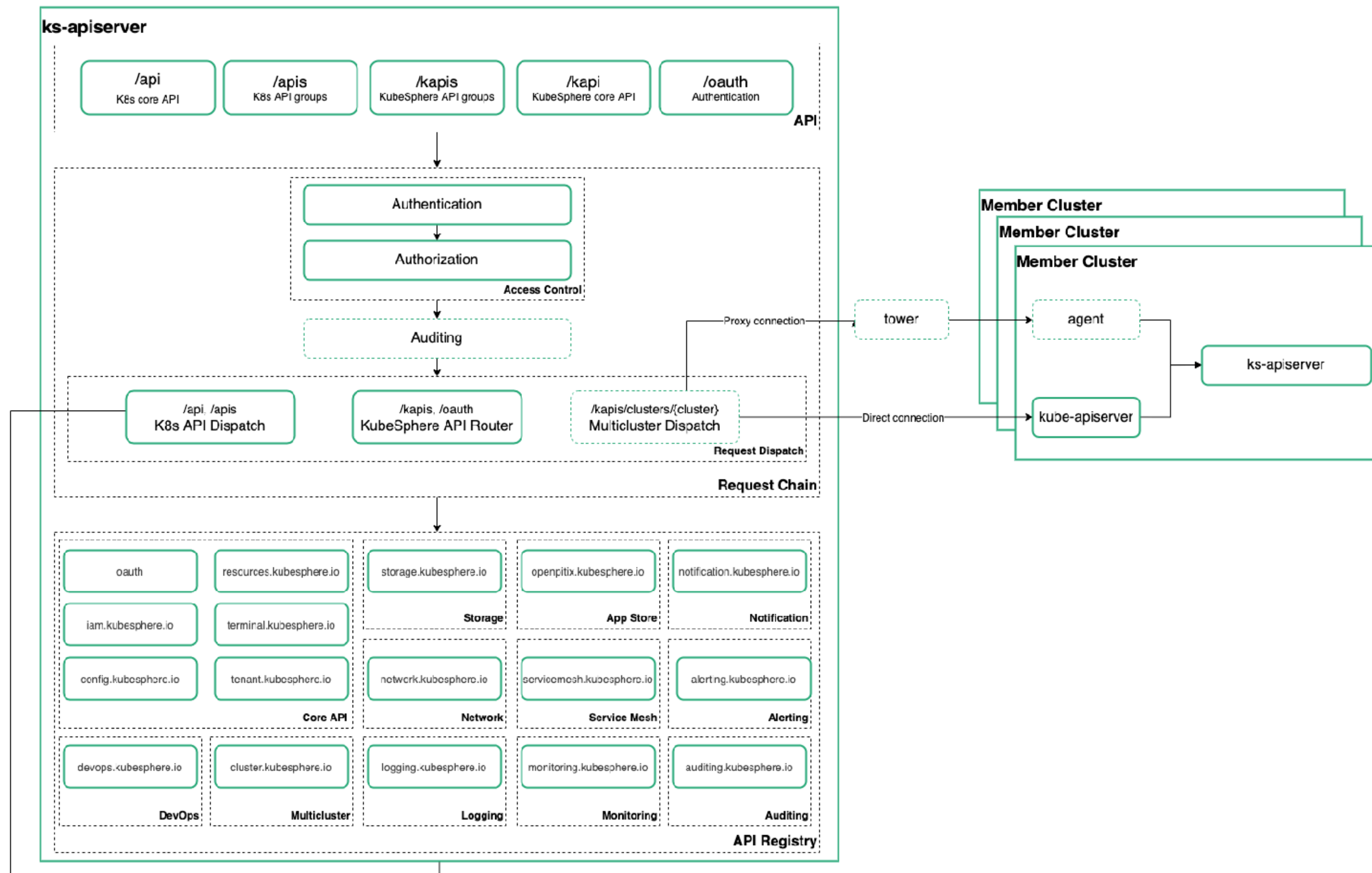
KubeSphere 的核心功能及架构

核心架构



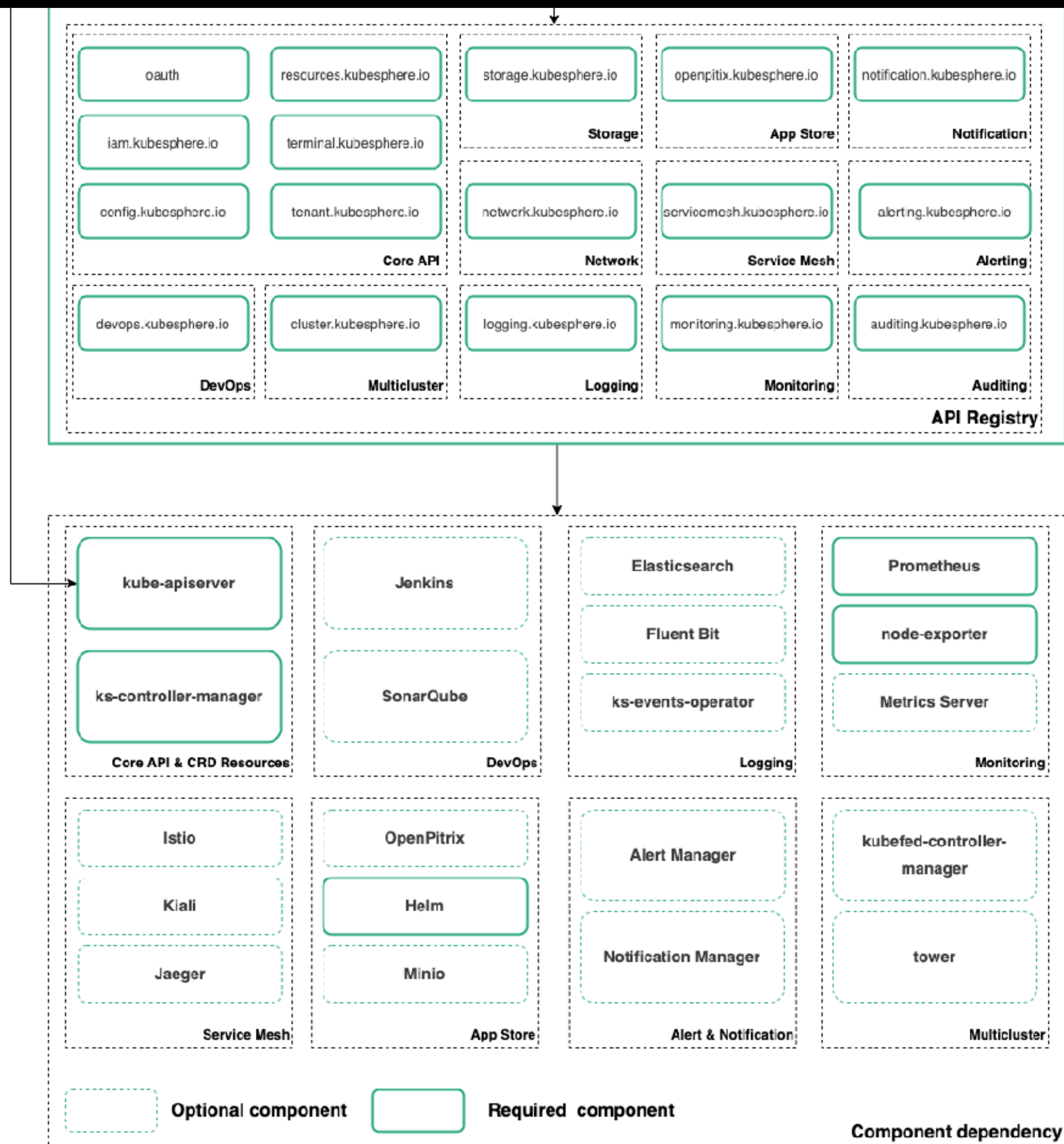
核心架构

- 认证、鉴权、审计
- 请求代理与分发
- API 聚合
- 资源（声明式）模型
- CRD
- controller runtime



核心架构

- 认证、鉴权、审计
- 请求代理与分发
- API 聚合
- 资源（声明式）模型
- CRD
- controller runtime



核心架构

CRD + Controller

Process main.go

One of these per cluster, or several if using HA.

Manager sigs.k8s.io/controller-runtime/pkg/manager

One of these per process.

Handles HA (leader election), exports metrics, handles webhook certs, caches events, holds clients, broadcasts events to Controllers, handles signals and shutdown.

Client

Communicates with API Server, handling authentication and protocols.

...

Cache

Holds recently watched or GET'ed objects. Used by Controllers. and Webhooks. Uses clients.

...

Controller sigs.k8s.io/controller-runtime/pkg/controller

One of these per Kind that is reconciled (i.e.. one per CRD).

Owns resources created by it.

Uses Caches and Clients and gets events via Filters.

Controller calls a Reconciler each time it gets an event.

Handles back-off and queuing and re-queuing of events.

Event

Filter

Predicate

sigs.k8s.io/controller-runtime/pkg/predicate

Filters a stream of events, passing only those that require action to the reconciler..

Reconciler

sigs.k8s.io/controller-runtime/pkg/reconciler

User-provided logic is added into the reconciler.Reconcile function.

Webhook sigs.k8s.io/controller-runtime/pkg/webhook

Zero or one Webhooks. One per Kind that is reconciled.

Zero or one Webhooks. One per Kind that is reconciled.

AdmissionRequest

Defaulter

Sets unset fields in spec.

Validator

Rejects poorly formed objects.

```
func main() {
```

```
mgr, err := manager.New(config.GetConfigOrDie(), manager.Options{})
if err != nil {
    log.Error(err, "could not create manager")
    os.Exit(1)
}
```

```
err = builder.
    ControllerManagedBy(mgr). // Create the ControllerManagedBy
    For(&appsv1.ReplicaSet{}). // ReplicaSet is the Application API
    Owns(&corev1.Pod{}).       // ReplicaSet owns Pods created by it
    Complete(&ReplicaSetReconciler{})
if err != nil {
    log.Error(err, "could not create controller")
    os.Exit(1)
}
```

```
if err := mgr.Start(signals.SetupSignalHandler()); err != nil {
    log.Error(err, "could not start manager")
    os.Exit(1)
}
```

// ReplicaSetReconciler is a simple ControllerManagedBy example implementation.

```
type ReplicaSetReconciler struct {
    client.Client
}
```

```
// Implement the business logic:
// This function will be called when there is a change to a ReplicaSet or a Pod with an OwnerReference
// to a ReplicaSet.
// * Read the ReplicaSet
// * Read the Pods
// * Set a Label on the ReplicaSet with the Pod count
func (a *ReplicaSetReconciler) Reconcile(ctx context.Context, req reconcile.Request) (reconcile.Result, error) {
```

```
rs := &appsv1.ReplicaSet{}
err := a.Get(ctx, req.NamespacedName, rs)
if err != nil {
    return reconcile.Result{}, err
}
```

```
pods := &corev1.PodList{}
err = a.List(ctx, pods, client.InNamespace(req.Namespace), client.MatchingLabels(rs.Spec.Template.Labels))
if err != nil {
    return reconcile.Result{}, err
}
```

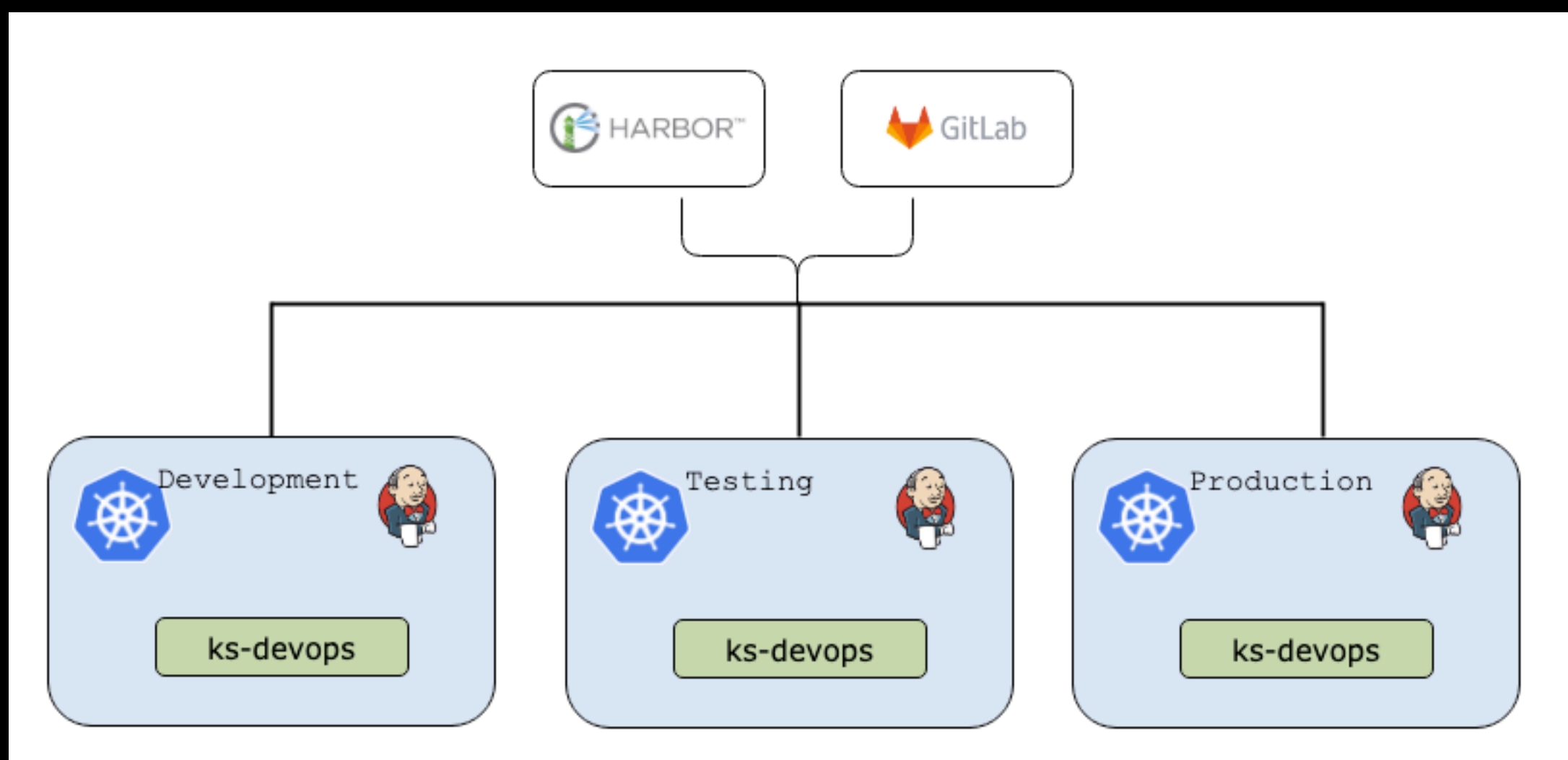
```
rs.Labels["pod-count"] = fmt.Sprintf("%v", len(pods.Items))
err = a.Update(ctx, rs)
if err != nil {
    return reconcile.Result{}, err
}
```

```
return reconcile.Result{}, nil
```

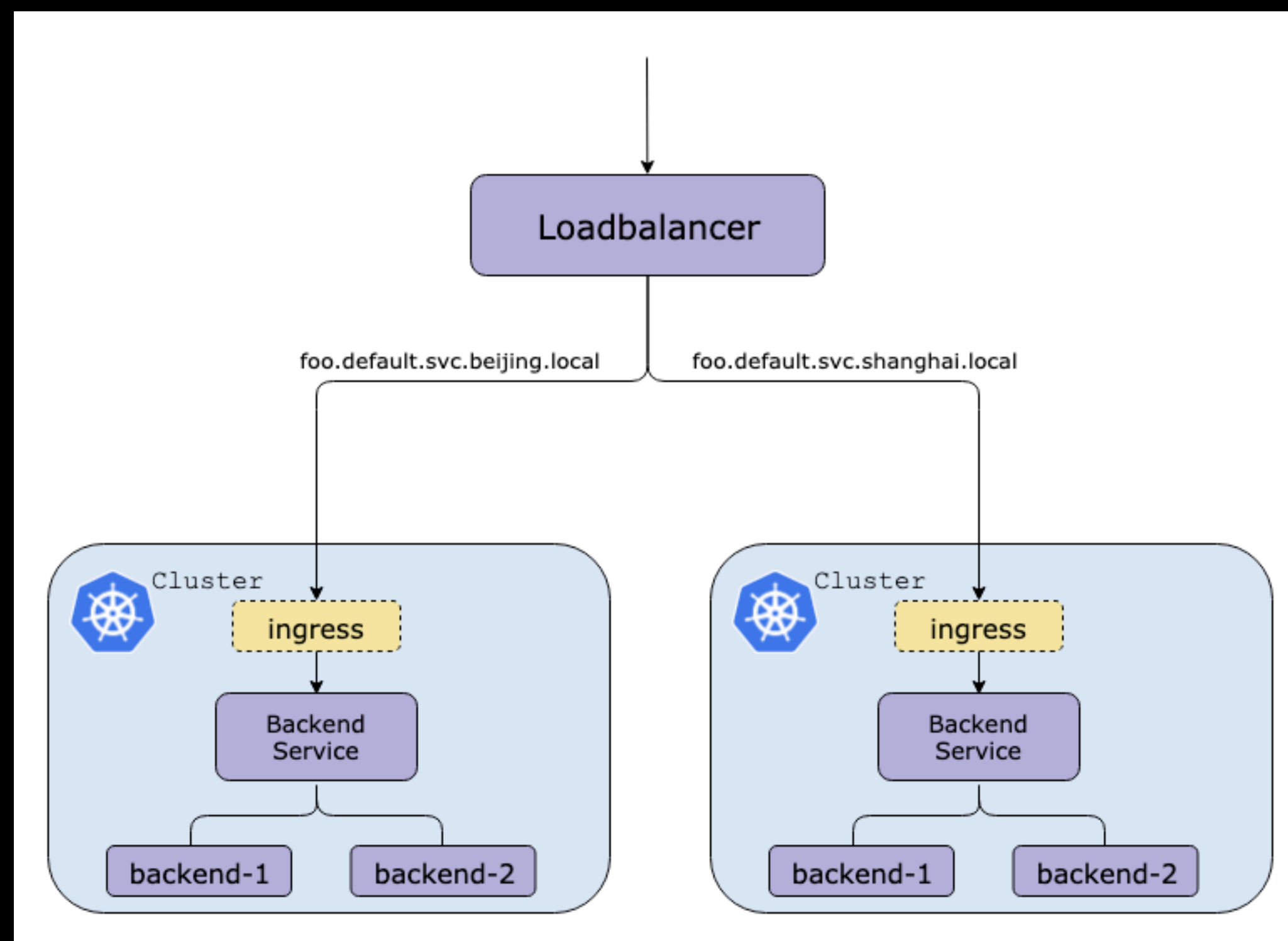
```
}
```

3.1 多集群管理

环境隔离



高可用

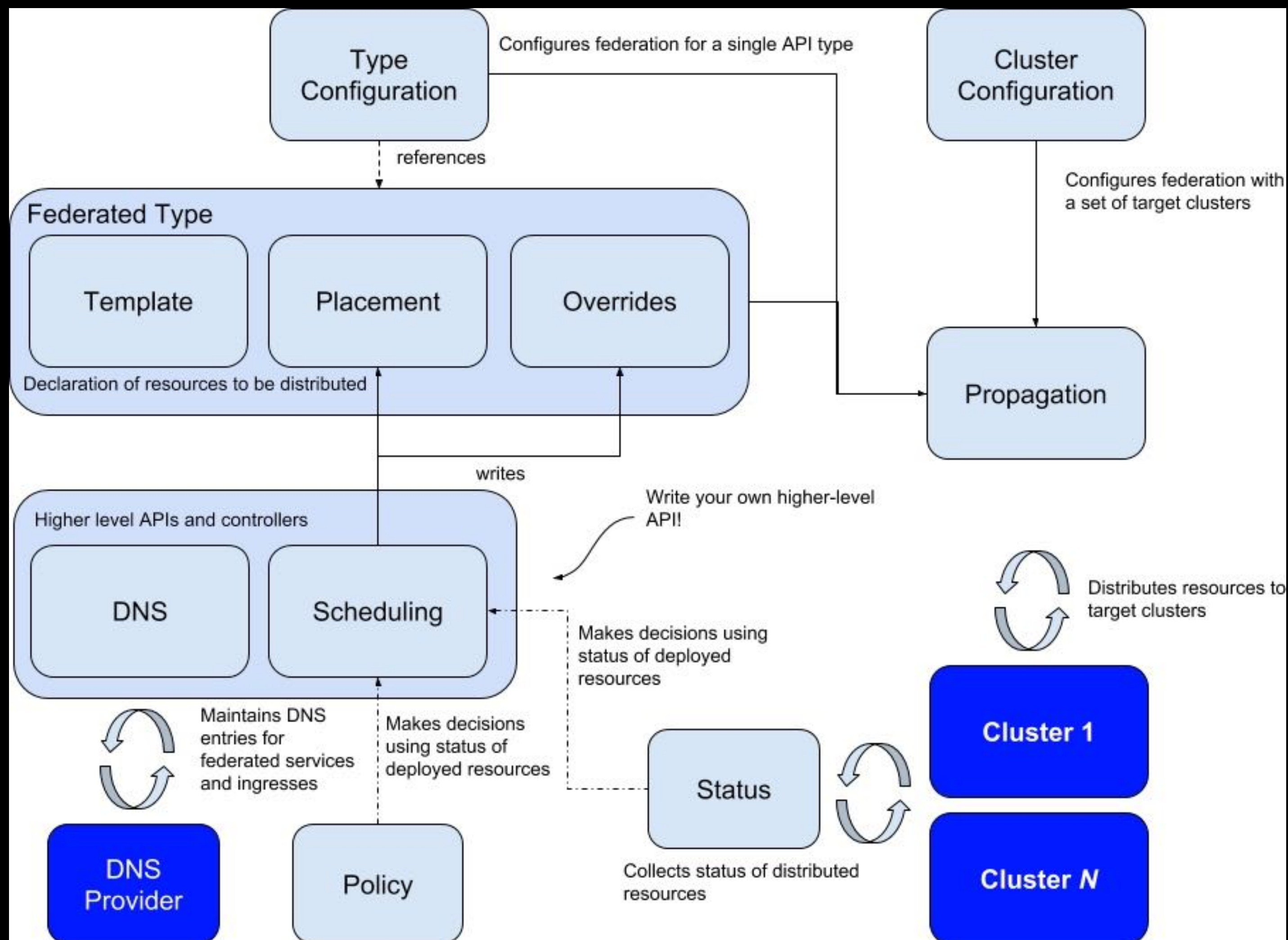


3.1 多集群管理

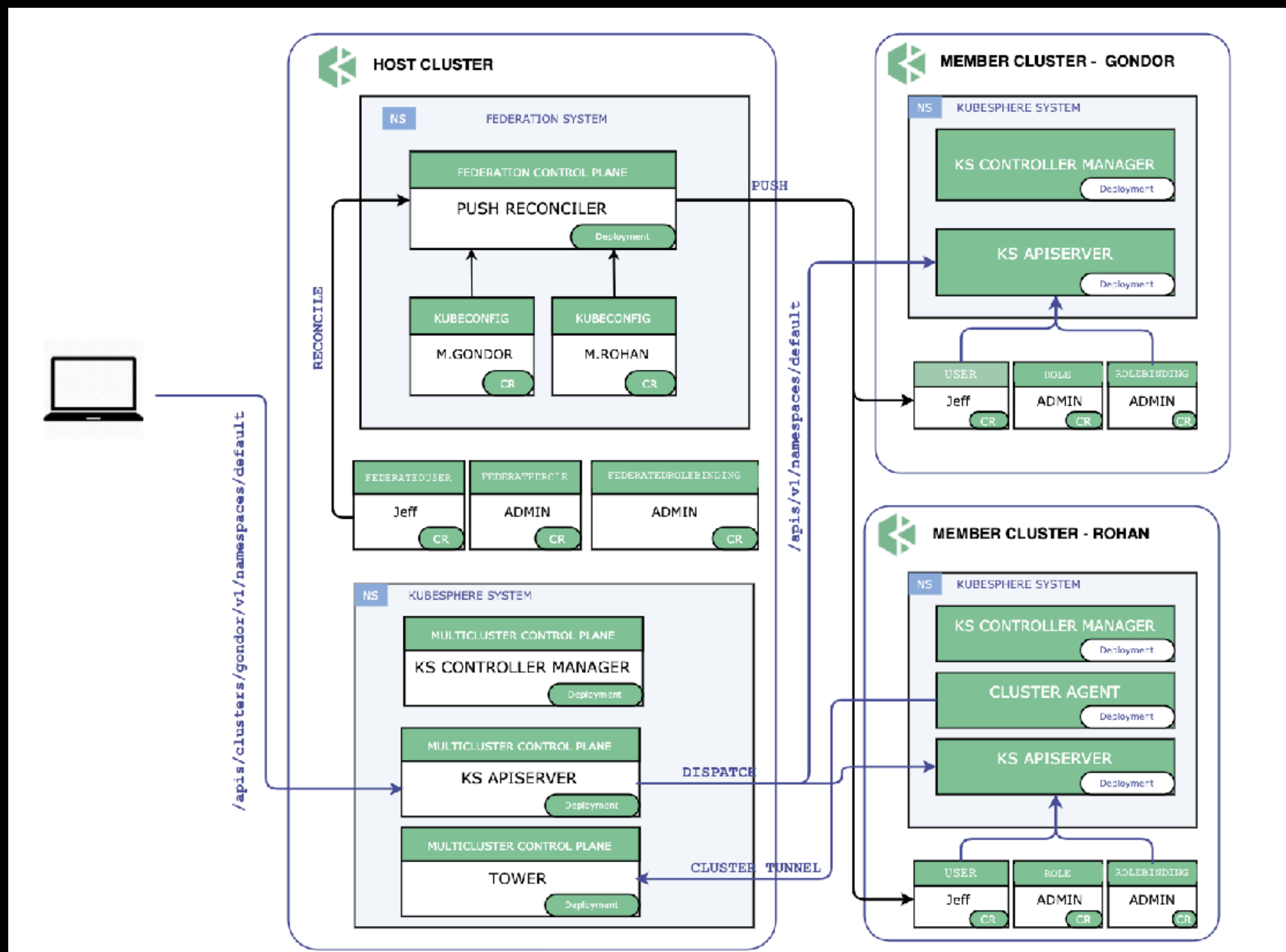
KubeSphere 多集群管理采用了
K8s 社区集群联邦方案 (KubeFed v2)

```

apiVersion: types.kubefed.io/v1beta1
kind: FederatedDeployment
metadata:
  labels:
    app: nginx
  name: nginx
  namespace: jeff
spec:
  overrides:
    - clusterName: gondor
      clusterOverrides:
        - path: /spec/replicas
          value: 4
  placement:
    clusters:
      - name: gondor
      - name: rohan
      - name: shire
  template:
    metadata:
      labels:
        app: nginx
      namespace: jeff
    spec:
      replicas: 1
      template:
        metadata:
          labels:
            app: nginx
        spec:
          containers:
            - image: nginx:latest
              name: nginx
  
```

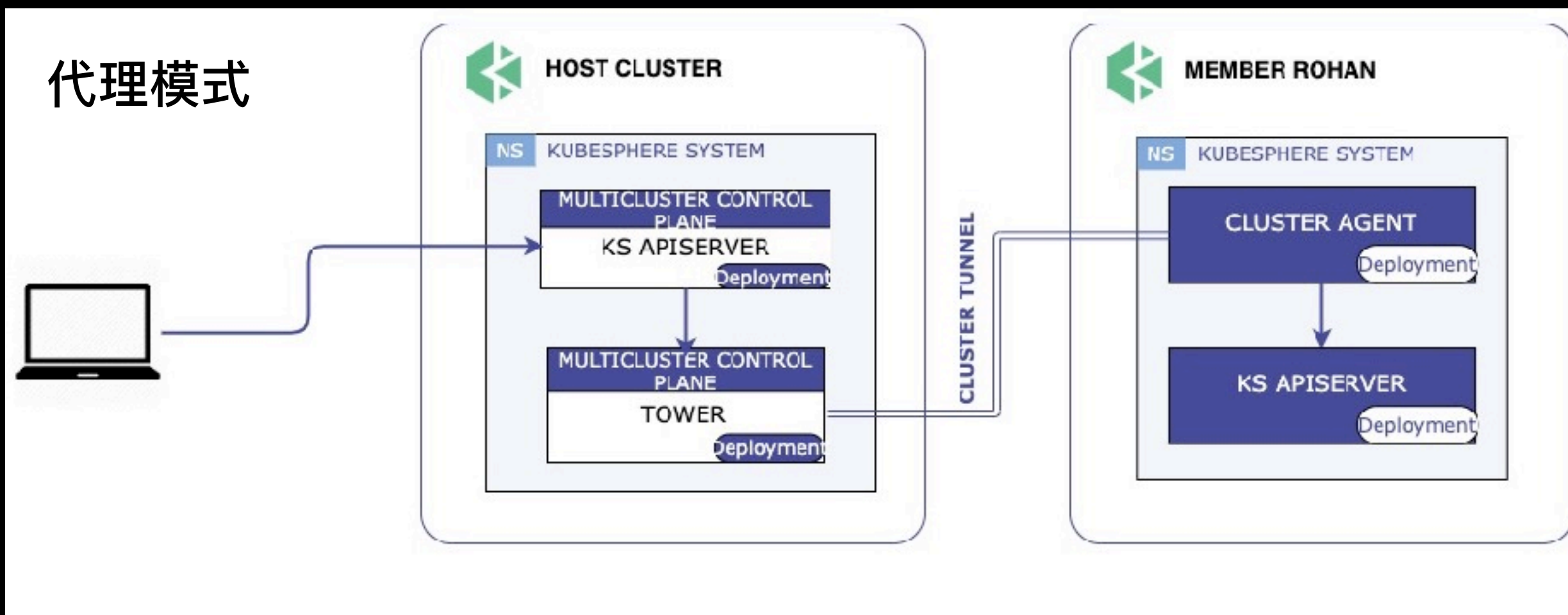


3.1 多集群管理

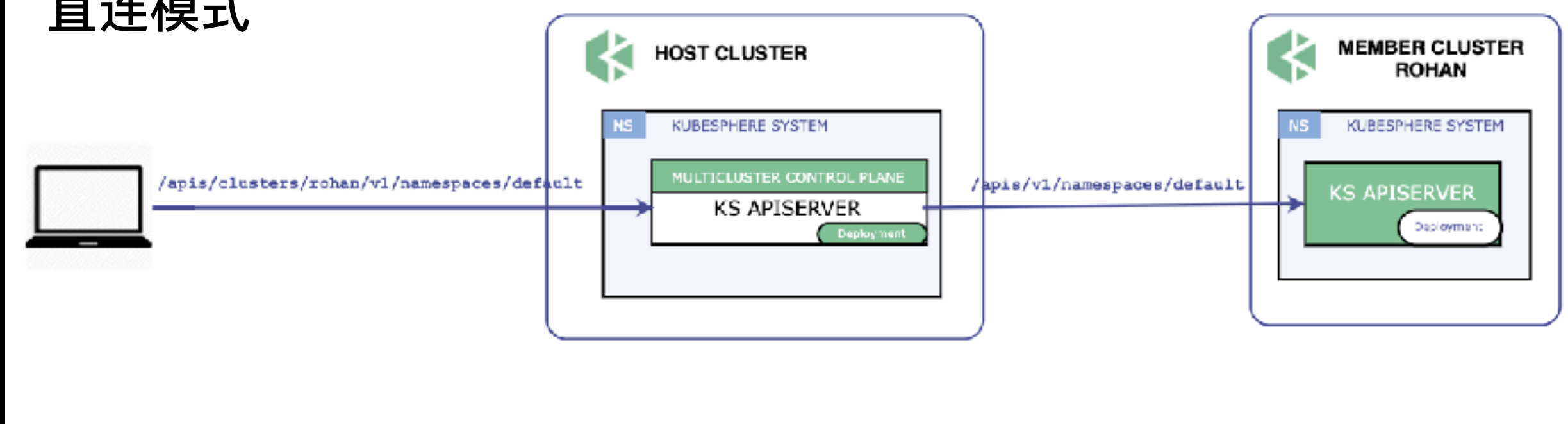


3.1 多集群管理

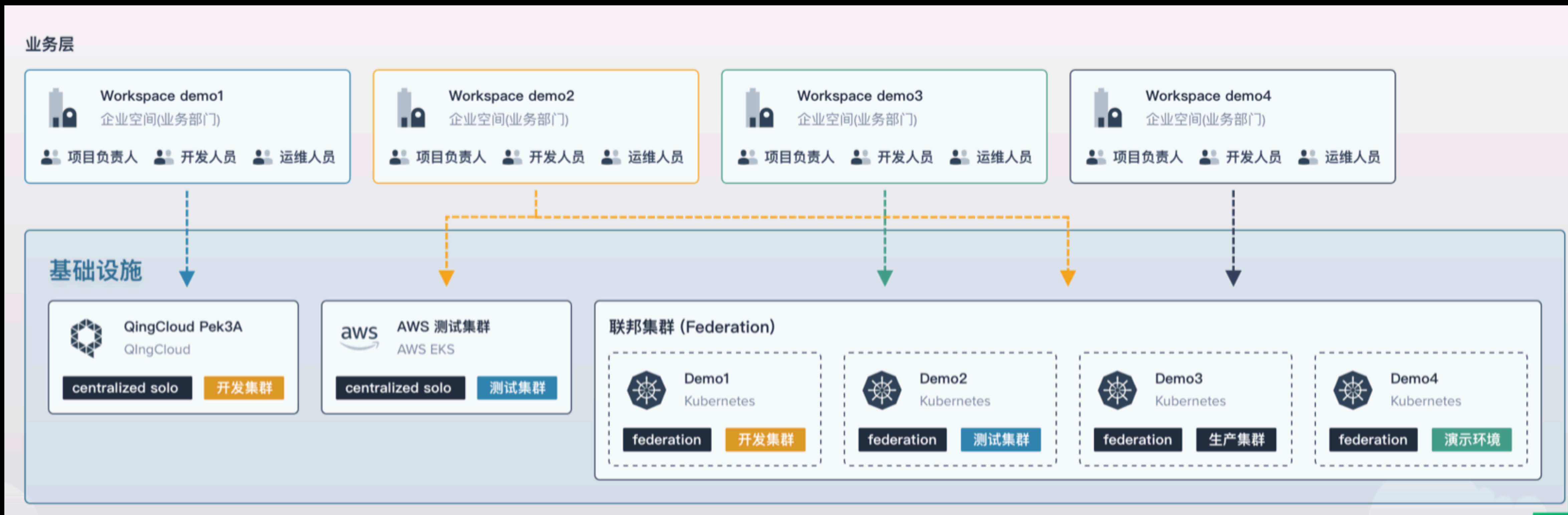
代理模式



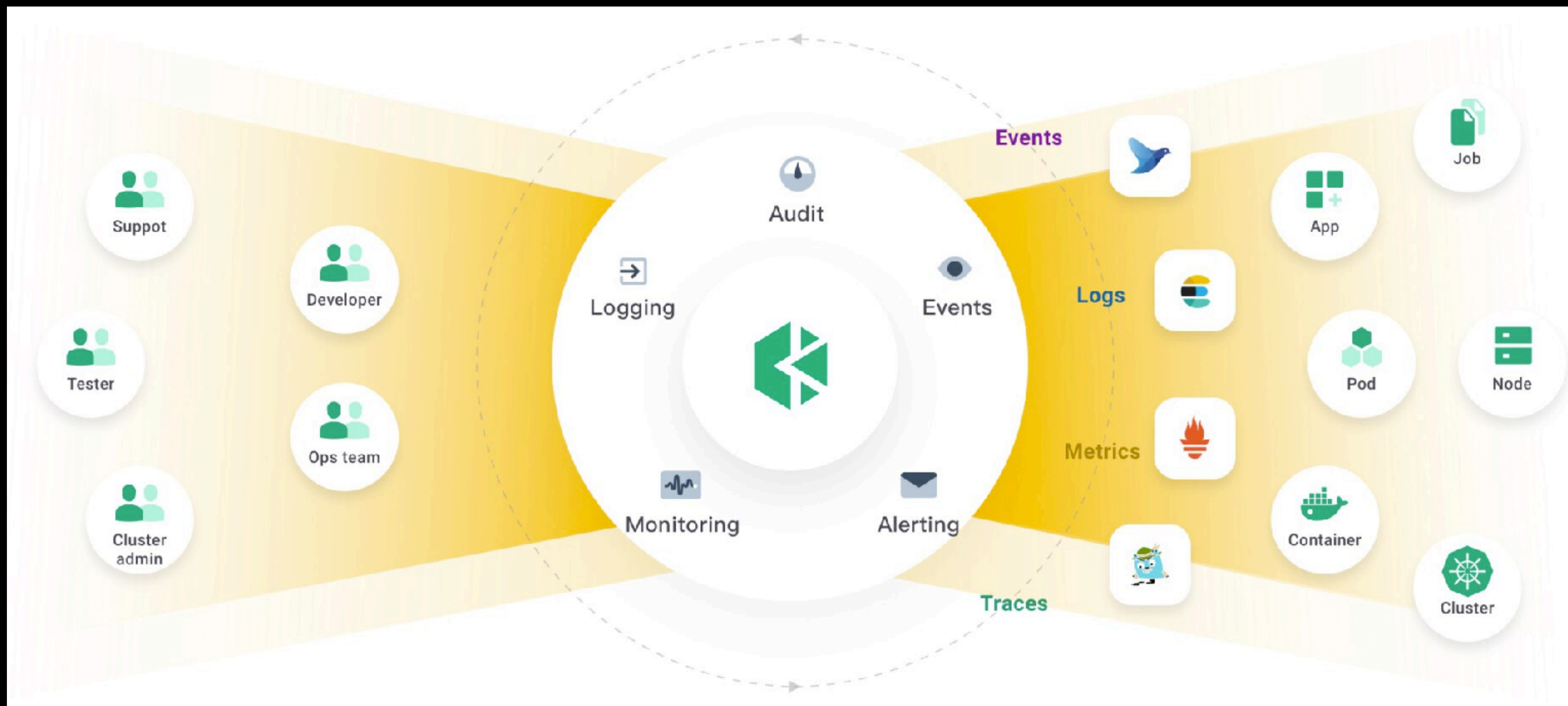
直连模式



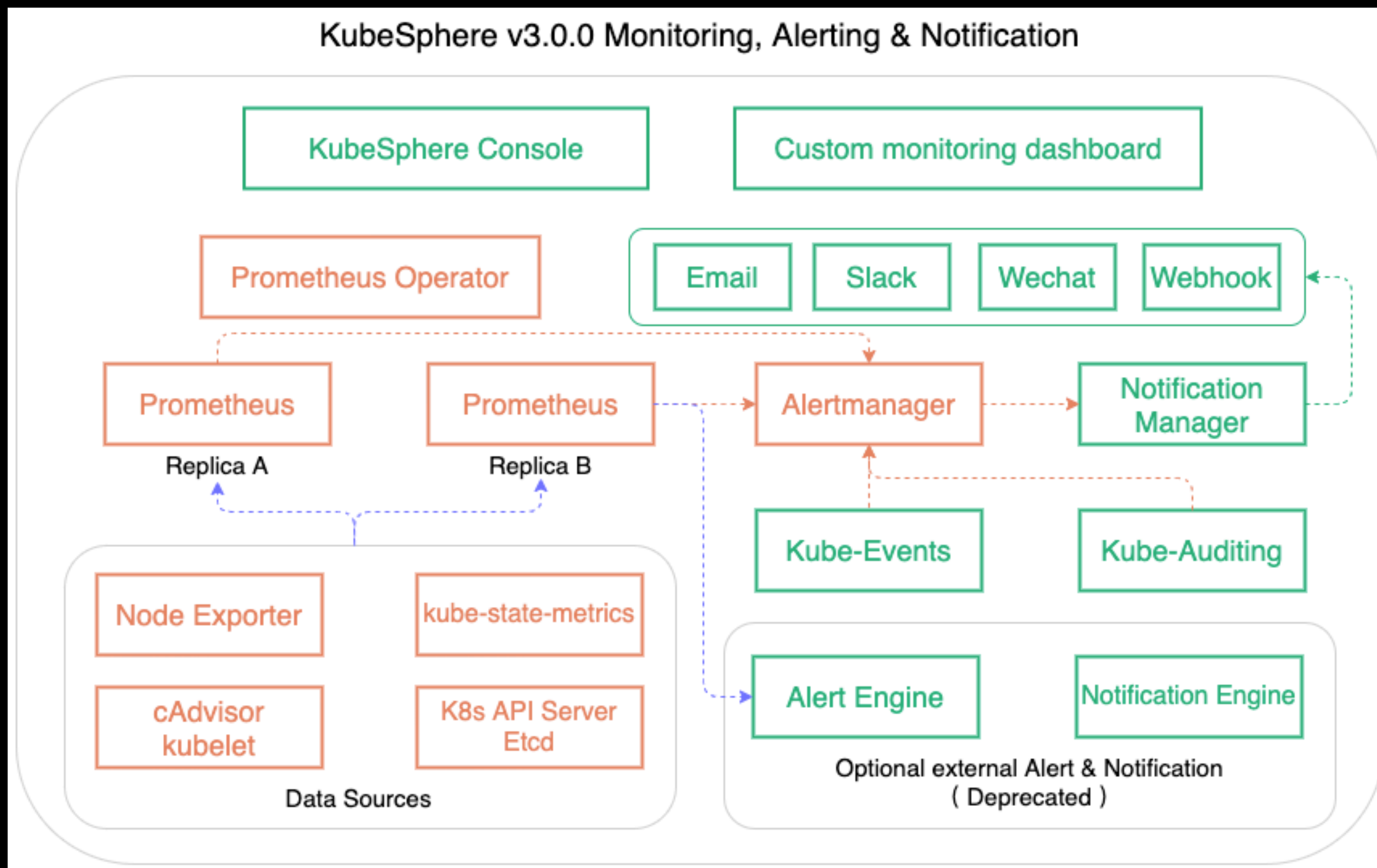
3.1 多集群管理



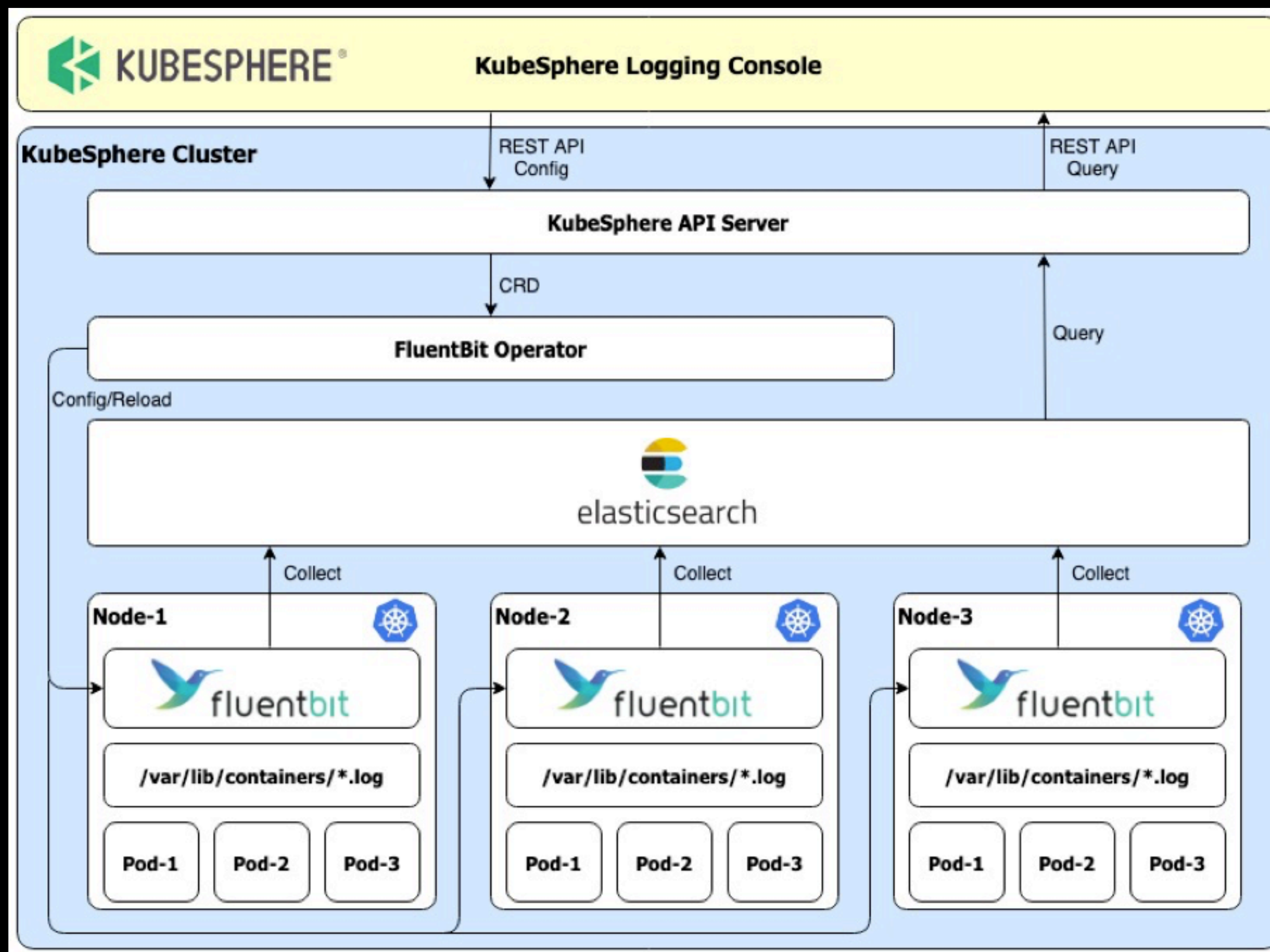
3.2 可观测性



3.2 监控告警



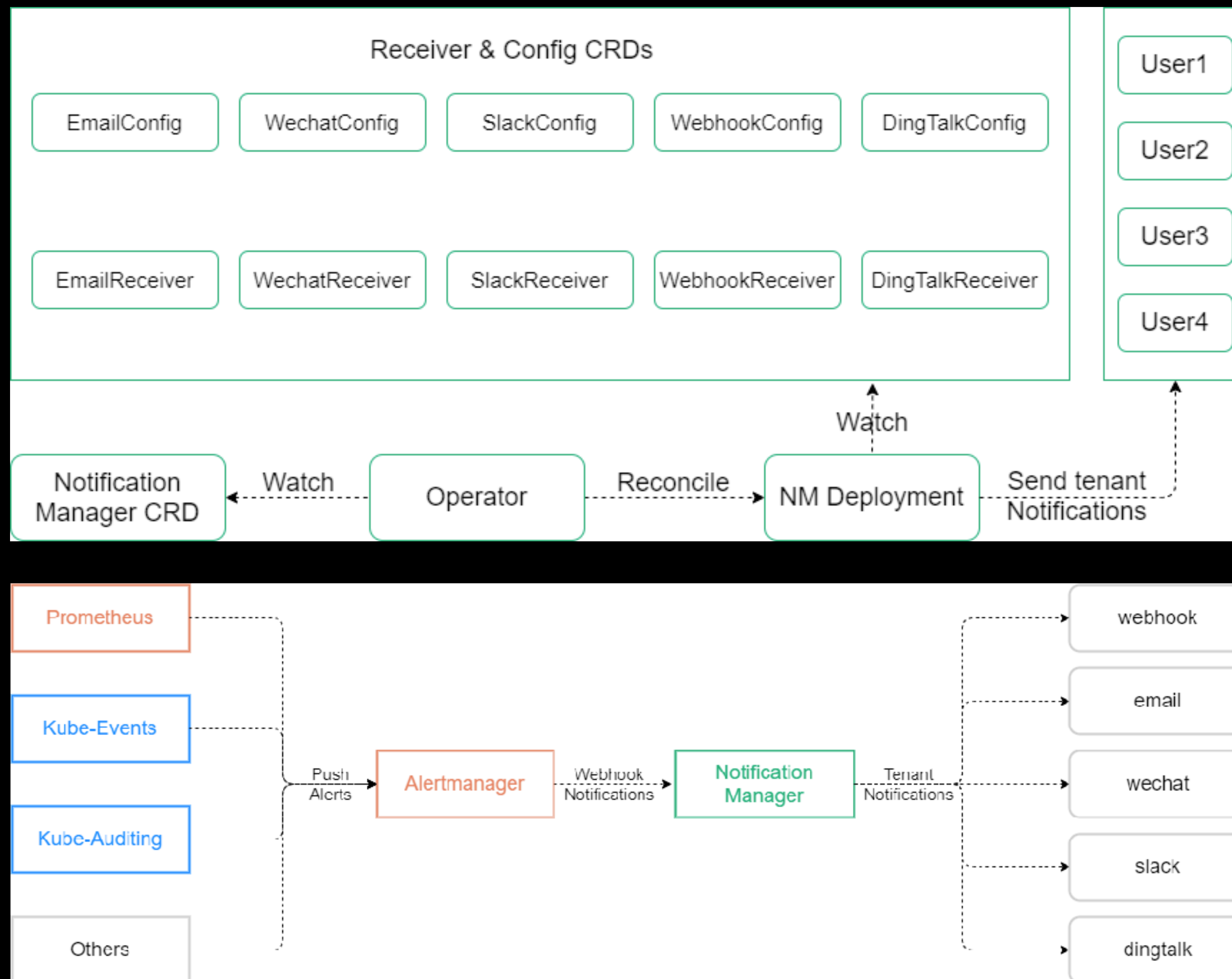
3.2 日志



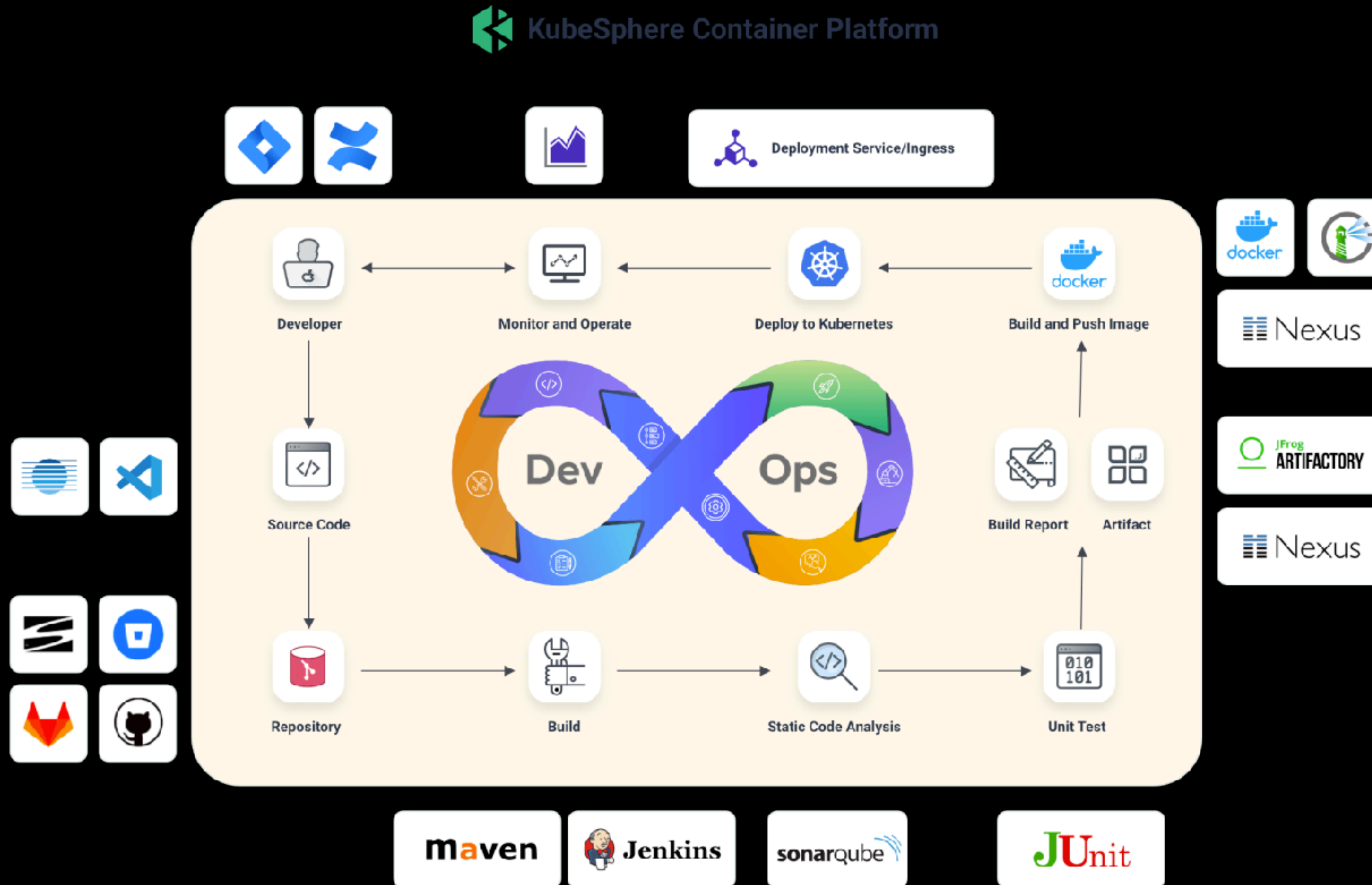
3.2 通知



```
apiVersion: notification.kubesphere.io/v2beta1
kind: Receiver
metadata:
  labels:
    app: notification-manager
    type: global
    name: global-email-receiver
spec:
  email:
    enabled: true
    to:
      - receiver1@xyz.com
      - receiver2@xyz.com
```



3.3 DevOps



3.5 微服务

任务状态

灰度组件

将一部分实际流量引入一个新版本进行测试，测试新版本的性能和表现，在保证系统整体稳定运行的前提下，尽早发现新版本在实际环境中的问题。

reviews v2

副本: 1/1

reviews-v2-7495795f84-frk4q

6 m

106.55 Mi

reviews v1

副本: 1/1

reviews-v1-68bb7b8c4f-gzlb1

5 m

106.12 Mi

实时流量分布

将所有的流量按比例分配给灰度组件

v2 流量 50%

v1 流量 50%

流量监控

请求成功率

请求延迟

暂无数据

暂无数据

流量监控 (RPS)

关闭

Part 4

KubeSphere 社区贡献方式

- KubeSphere 社区 <https://github.com/kubesphere/community>
- 官网与文档 <https://github.com/kubesphere/website>
- KubeSphere 后端 <https://github.com/kubesphere/kubesphere>
- KubeKey <https://github.com/kubesphere/kubekey>
- KubeEye <https://github.com/kubesphere/kubeeye>
- PorterLB <https://github.com/kubesphere/porterlb>
- KubeDesign <https://github.com/kubesphere/kube-design>
- NotificationManager <https://github.com/kubesphere/notification-manager>

欢迎大家以各种形式参与社区，issue、doc、proposal

Thank you!



github.com/kubesphere



twitter.com/kubesphere



[UP/KubeSphere](https://www.bilibili.com/up/KubeSphere)



kubesphere.io



kubesphere.slack.com