KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION

China 2024

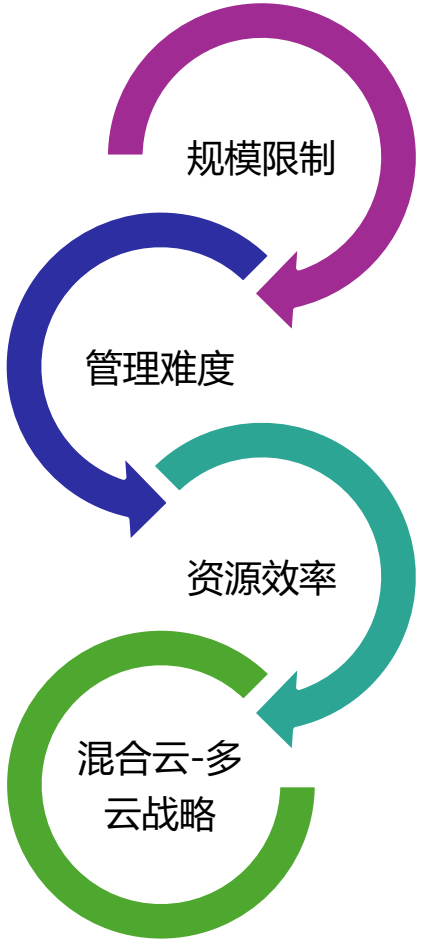
多集群助力小红书打造面向混合云的高可用弹性架构

熊峰@ 小红书

Agenda

- 为什么引入多集群&选型思考
- 应用如何分发与迁移
- Kubernetes API 兼容
- 多集群场景下弹性能力建设
- 控制面自动化运维
- 规划与展望

解决什么问题



单集群约5000节点的规模上限

无法满足互联网超大规模应用

集群数量众多，持续快速增长

对业务发布与运维管理带来挑战

业务与集群深度绑定，资源分配效率低下

资源走查感知容量，自动化程度不高

资源腾挪、甚至人工调度

资源碎片多，跨集群资源碎片难以利用

面向应用提供统一的平台入口

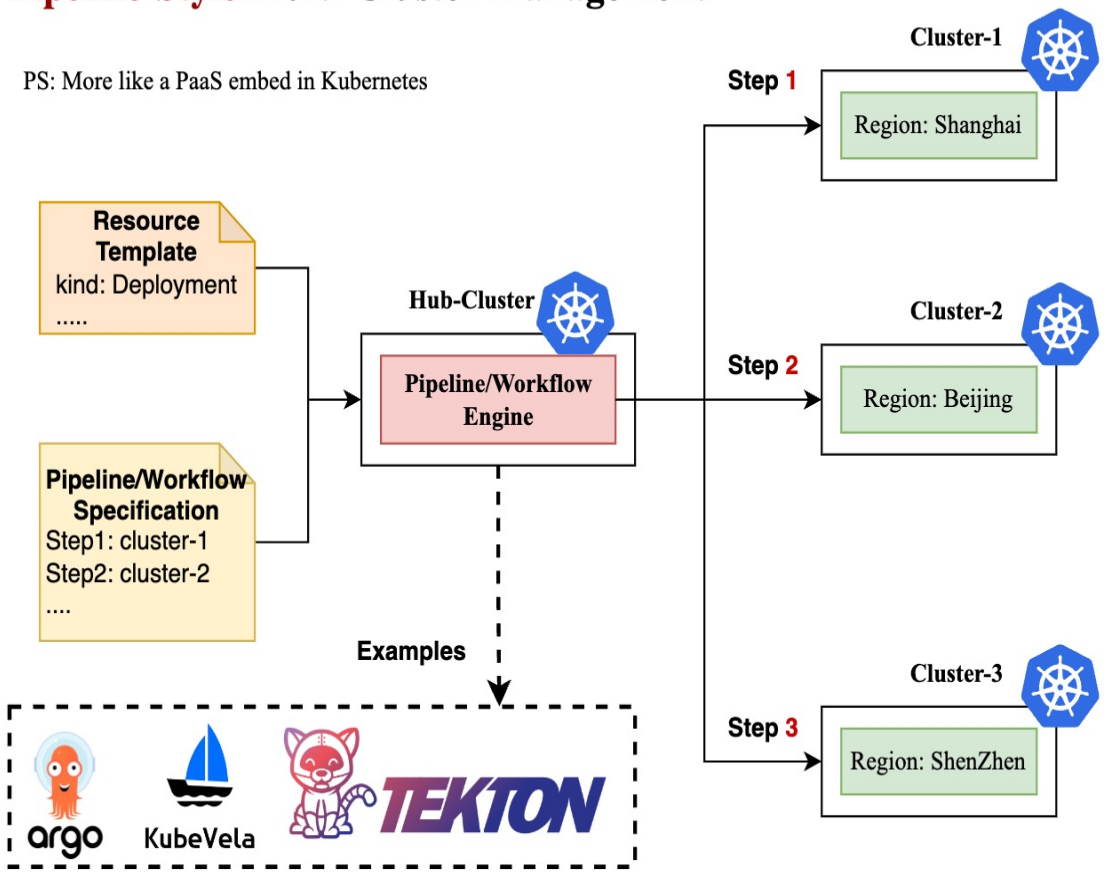
提升应用跨集群分发与调度能力

高效管理多云环境下的基础设施

选型思考

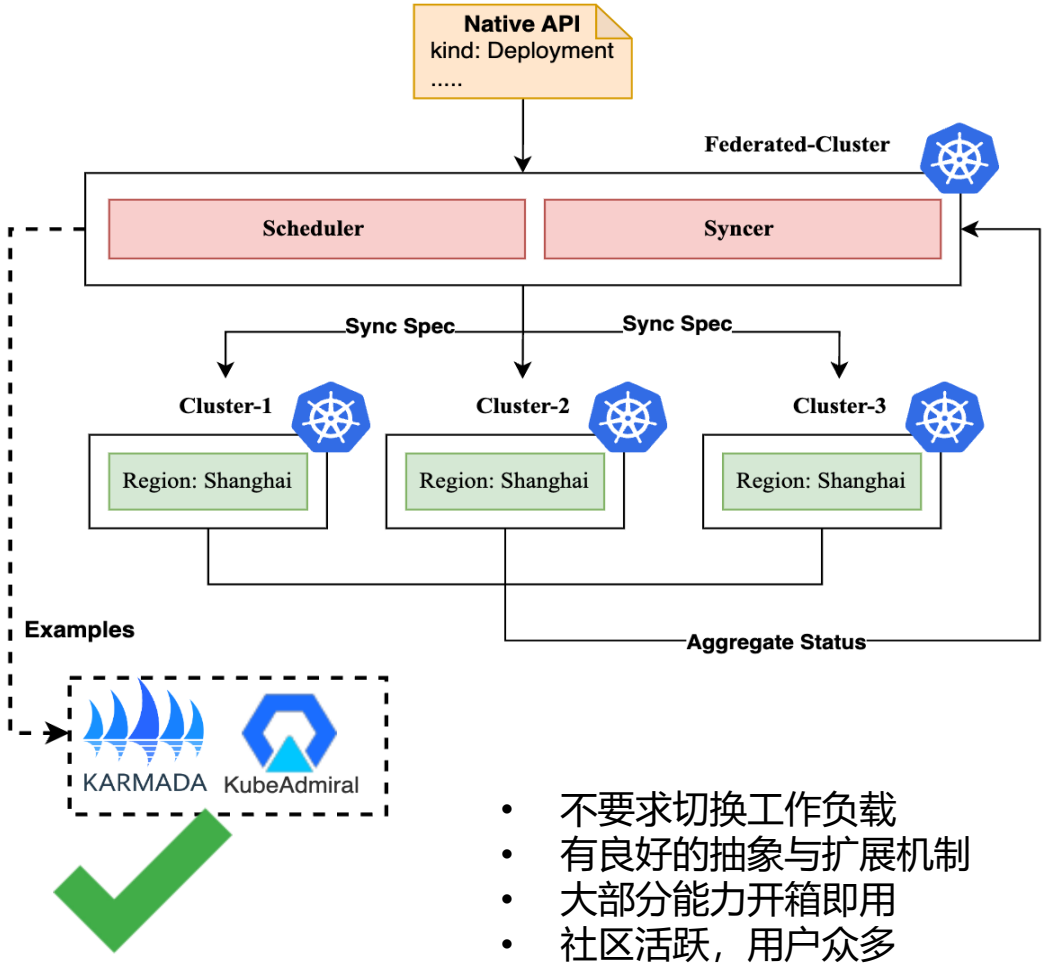
Pipeline-Style Multi-Cluster Management

PS: More like a PaaS embed in Kubernetes



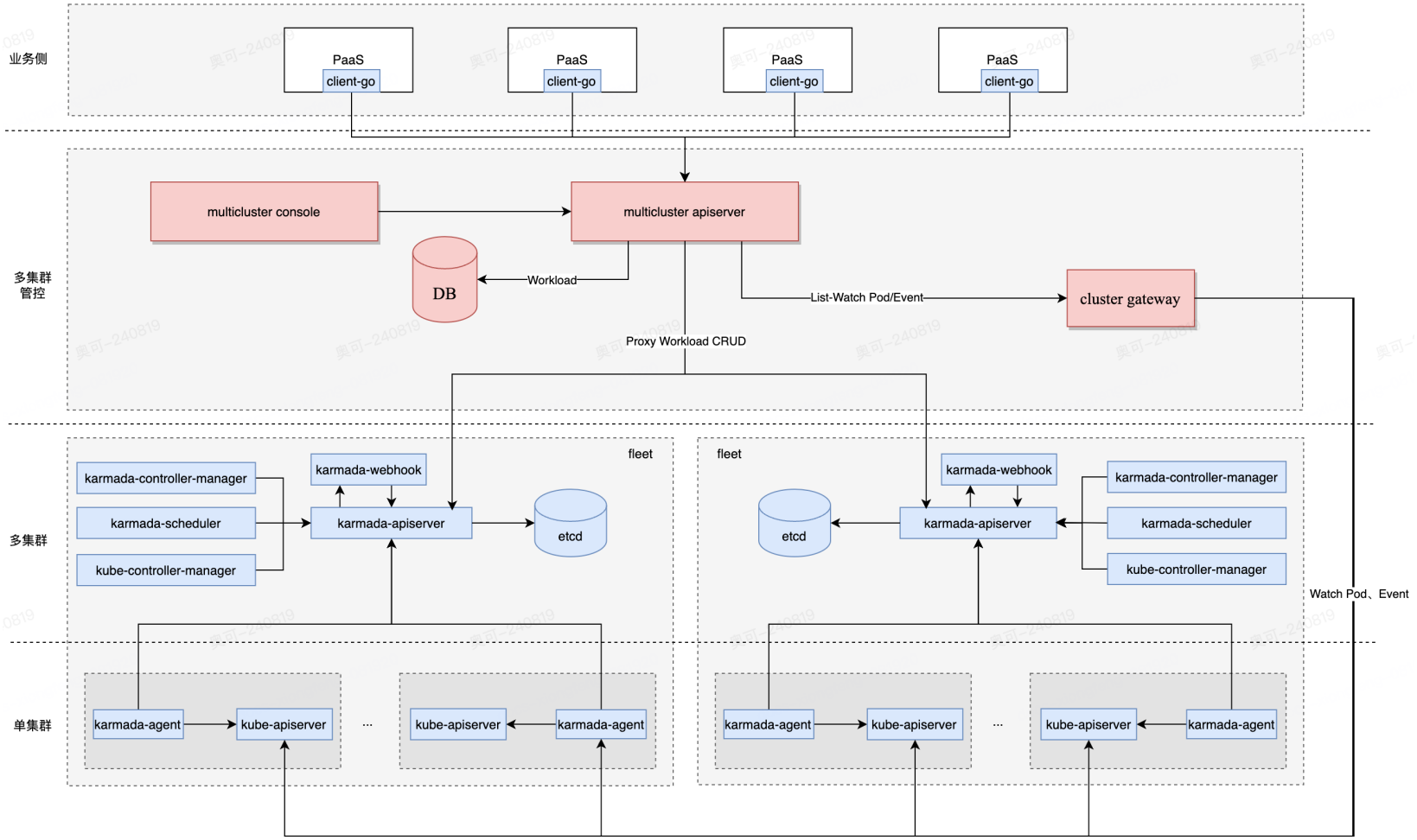
Federation-Style Multi-Cluster Management

PS: hope provide a consistent user experience with native Kubernetes



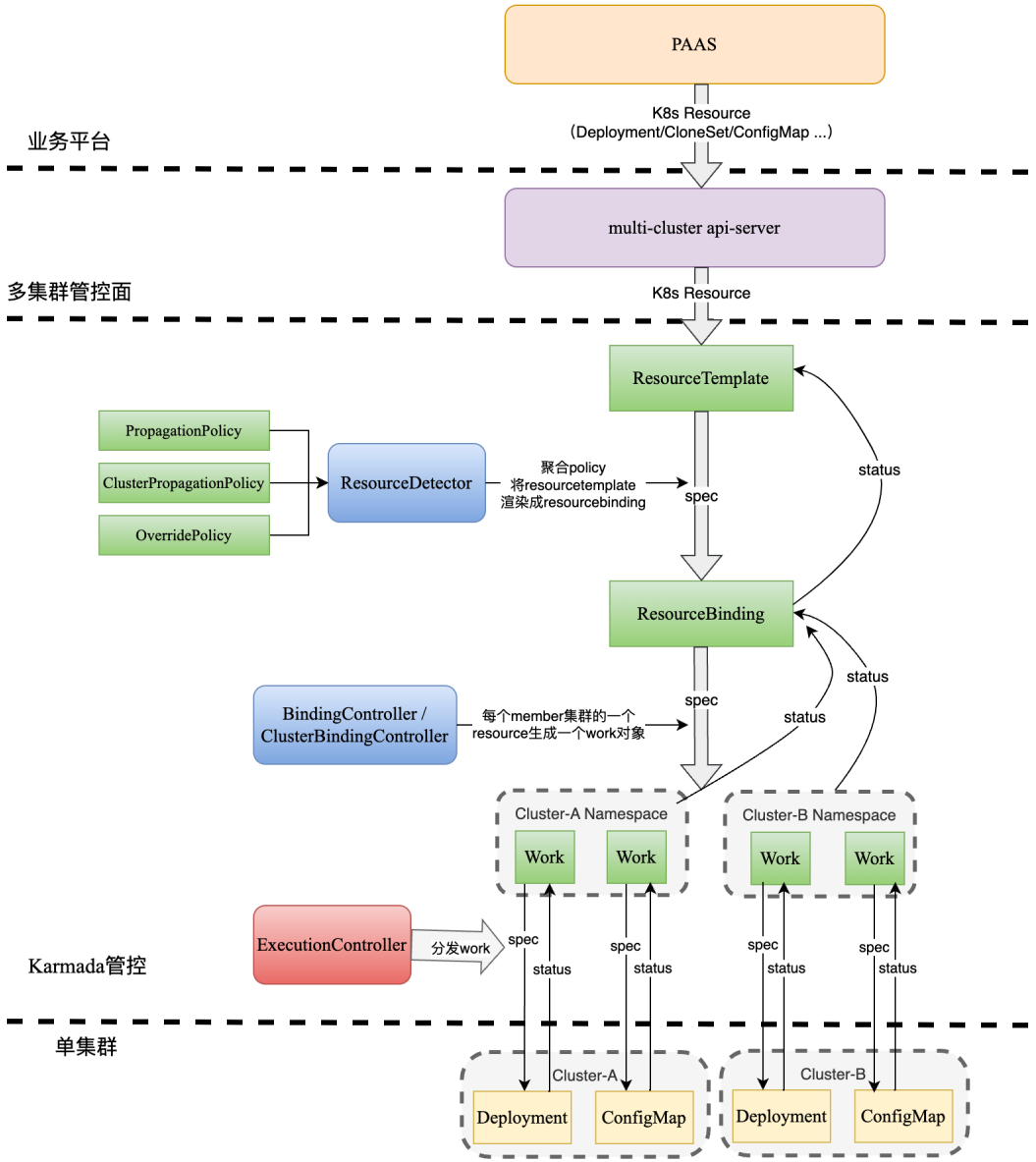
- 不要求切换工作负载
- 有良好的抽象与扩展机制
- 大部分能力开箱即用
- 社区活跃，用户众多

整体架构



- 业务层：不同业务板块的 PaaS 平台，通过K8S API实现业务应用的发布与管控运维
- 多集群管控：多集群apiserver 作为统一的 api 入口，将不同类型的请求转发到不同链路。
- 联邦层：联邦control-plane
- 单集群：成员集群，最重承载应用的 Pod 实体

如何实现资源分发



PaaS 交互原则：

- 像使用单集群一样使用多集群，不感知物理集群拓扑
- 不需要配置多集群资源分发策略和差异化策略、下发基础 K8S 资源

资源下发三个主要过程：

- ResourceDetector：聚合不同类型的分发策略，最终生成ResourceBinding对象；
- BindingController：解析placement rule和resource object，为每个member cluster生成一个work对象；
- ExecutionController：将work对象中包含的资源信息渲染成k8s resource，下发到member k8s cluster中；

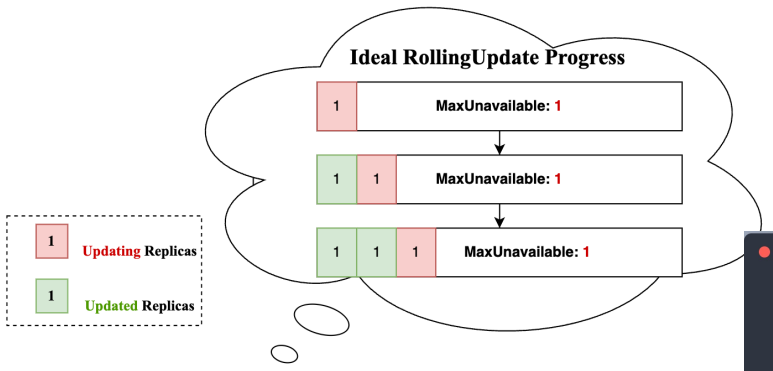
状态聚合同样经历三个过程：

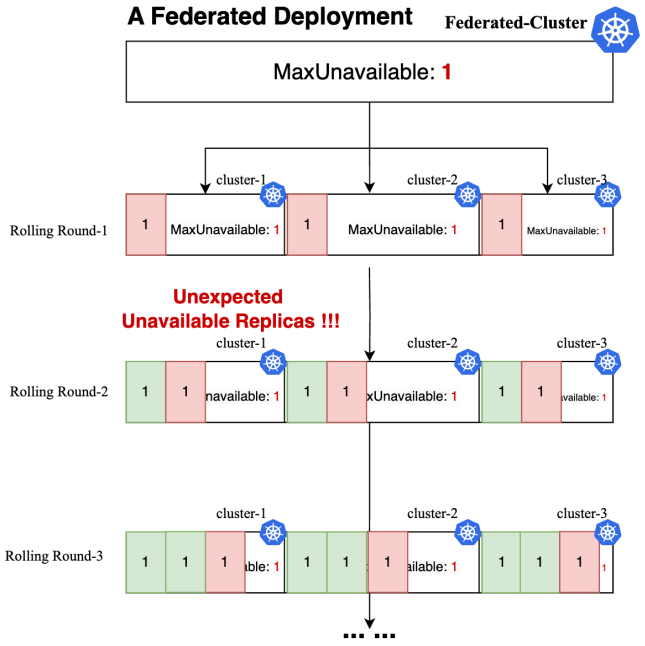
- 单集群k8s resource status -> work status
- 多个work status -> 聚合到一个resourcebinding status
- resourcebinding status -> 多集群k8s resource status

数据同步原则：

- spec自上而下，status自下而上；
- 数据更新原则：借助3-way-merge-patch ，与联邦有冲突以联邦为准，没有冲突的配置保留(以单集群为准)。

RollingUpdate失效





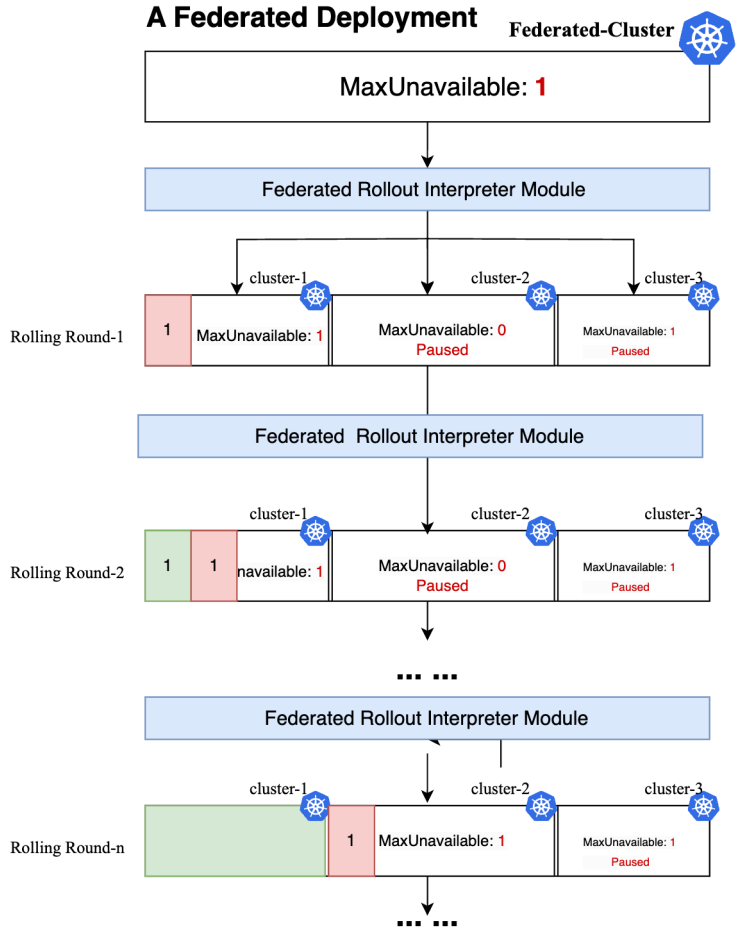
MaxUnavailable 被放大

```go
// ResourceInterpreter manages both default and customized webhooks to interpret custom resource structure.
type ResourceInterpreter interface {
    ... ...

    // GetRollingStrategy returns the rolling strategy of the object.
    GetRollingStrategy(object *unstructured.Unstructured) (
        rollingStrategy *configv1alpha1.RollingStrategy, err error)

    // ReviseRollingStrategy revises the rolling strategy of the given object.
    ReviseRollingStrategy(object *unstructured.Unstructured,
                          rollingStrategy *configv1alpha1.RollingStrategy) (
        revisedObject *unstructured.Unstructured, err error)

    // InterpretRollingStatus interprets the raw status of the object as unified rolling status.
    InterpretRollingStatus(object *unstructured.Unstructured,
                           rawStatus *runtime.RawExtension) (
        status *configv1alpha1.UnifiedRollingStatus, err error)
}

// Interpreter is the interface for aligning rolling strategy.
type Interpreter interface {
    AlignRollingStrategy(resourceTemplate *unstructured.Unstructured,
                         targetCluster []*workv1alpha2.TargetCluster,
                         works map[string]*workv1alpha1.Work) (
        map[string]*configv1alpha1.RollingStrategy, error)
}
```

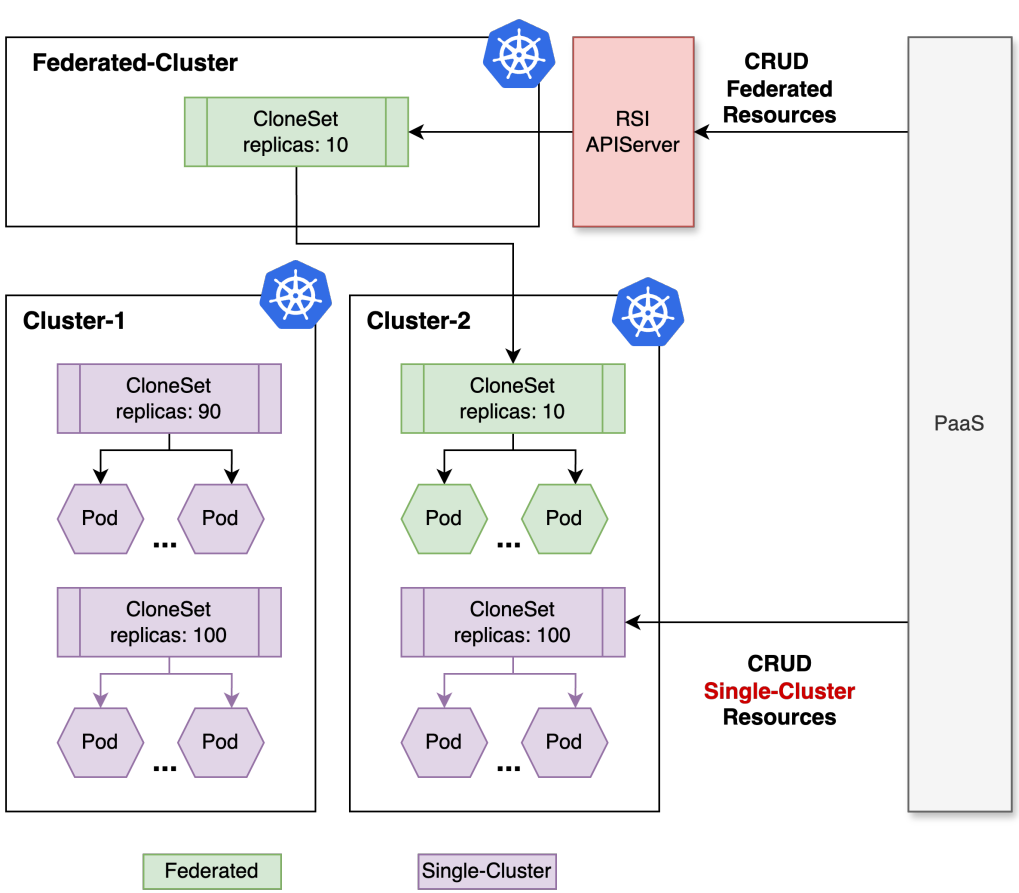
联合社区引入 Rollout-Interpreter

Federated Rollout Interpreter:

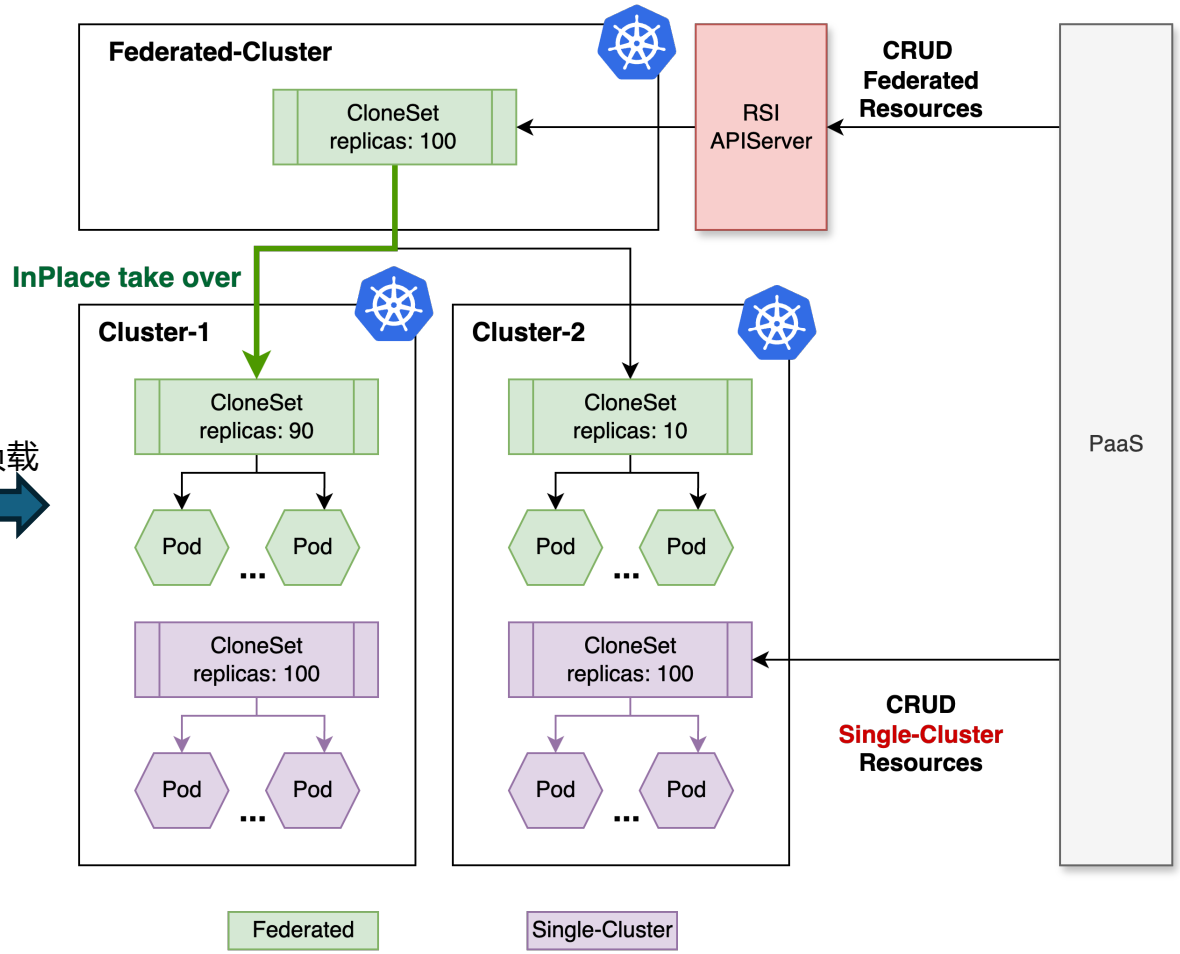
Align Federated RollingUpdate Behavior with Single-Cluster



存量应用迁移



纳管Cluster-1 工作负载



- 原地纳管：纳管过程 Pod、Workload 不重建、不重启
- 双入口并存：优先联邦入口，紧急情况回退单集群入口。

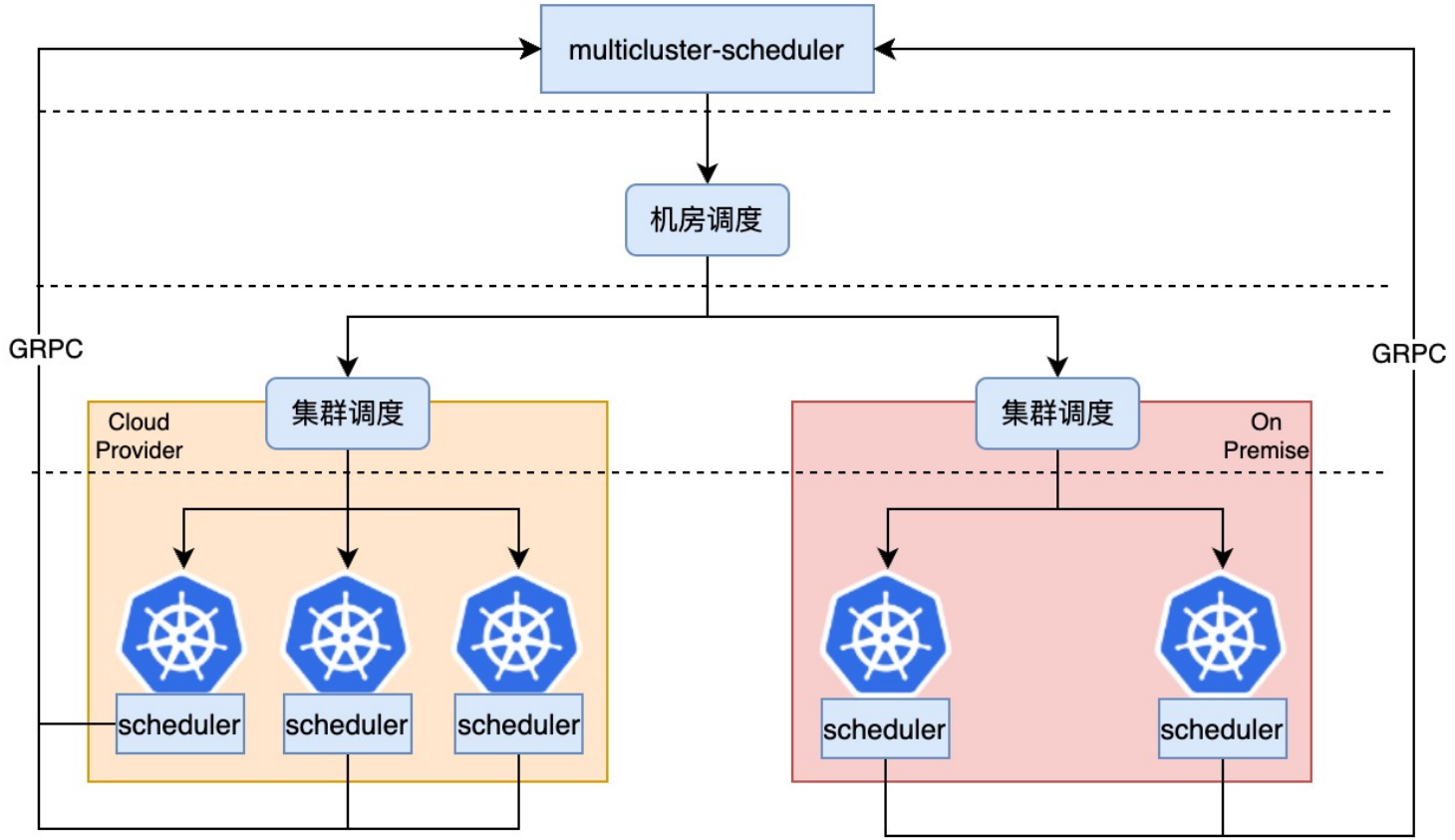
更灵活的多集群调度

多集群的多层调度结构：

机房间调度：
- 基于成本、弹性实现机房间调度
- 迁移辅助

集群间调度：
- 调度到已经分发 workload 的集群，避免过于分散
- 基于节点规模的调度，有效管控集群节点规模
- 优先基于集群可用资源，Pending 尽量集中在一个集群，便于节点伸缩

模拟调度：
- 集群调度层面结合单集群模拟调度与资源预留能力
- 全局资源账本
- 更有效的资源利用



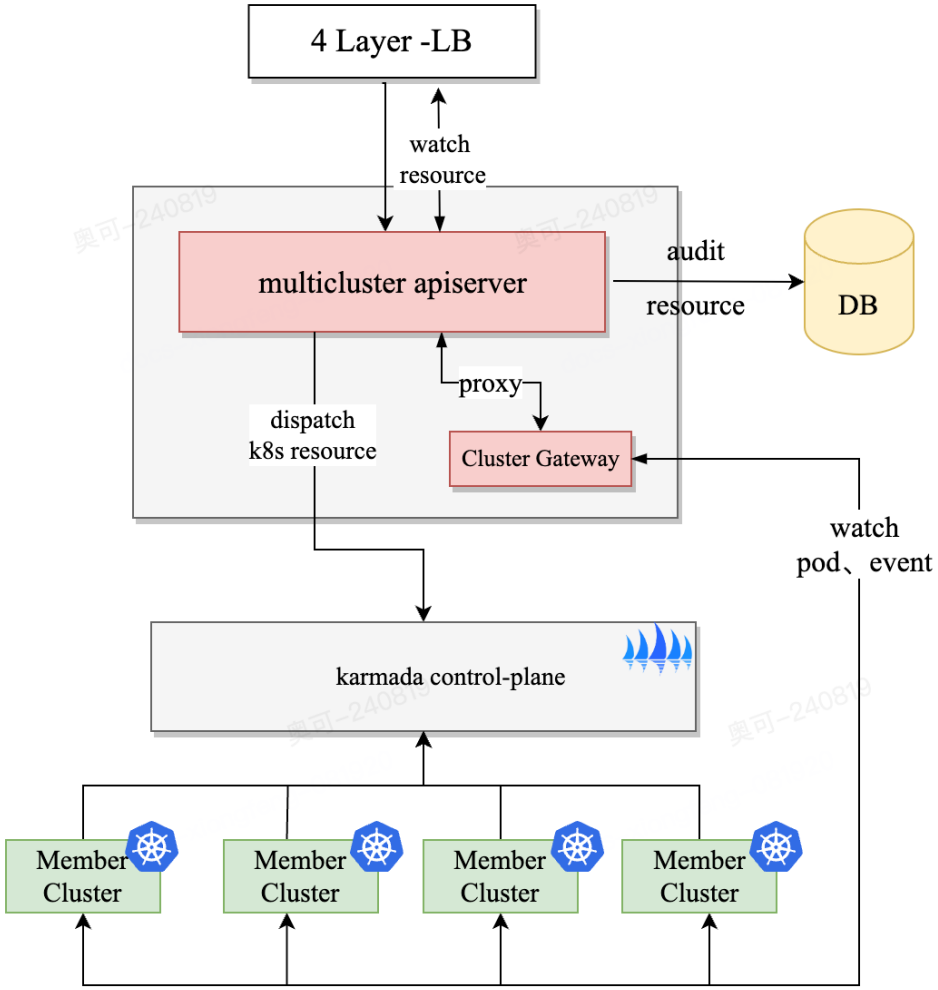
Kubernetes API兼容

背景：

1. Pod、Event等单集群资源无法通过Karmada 管控面直接获取；
2. 公司内部多个PaaS平台直接使用client-go和k8s交互，期望多集群方案能够兼容单集群的使用模式，降低接入门槛，同时提供贴近单集群的使用体验，屏蔽多集群

架构：

1. 基于 k8s.io/apiserver 库构建兼容 K8S API的 7 层网关 multicluster-apiserver 和缓存多集群 Pod、Event资源的多集群缓存组件 cluster-gateway
2. PaaS 平台通过client-go直接和 multicluster-apiserver 进行交互，对联邦化资源的请求会转发到 karmada-apiserver，对Pod、Event资源的请求被转发到 cluster-gateway



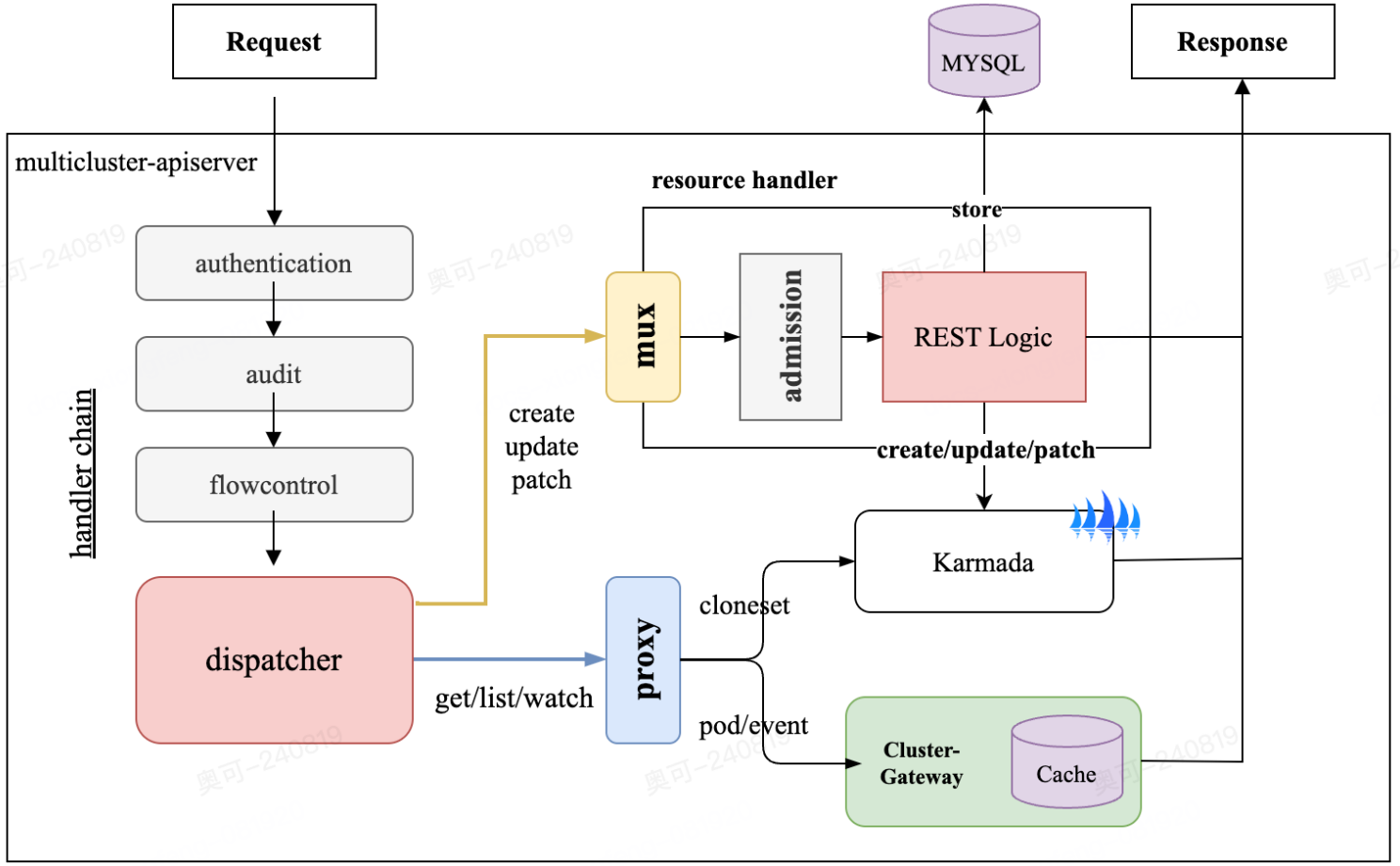
Kubernetes API 兼容

1. 基于 k8s.io/apiserver 构建的网关服务multicluster-apiserver提供兼容k8s api的服务，复用了k8s.io/apiserver 中的 handler 处理链，支持对请求的身份验证、审计以及全局的请求流控

2. 在handler处理链的最后一步，根据请求verb决定Proxy请求还是经由网关进一步处理

3. 所有的变更动作，会经过内部的admission校验变更动作合理性，比如删除保护、缩容保护，经过准入校验后会原子性的审计变更和应用变更到karmada control-plane。

4. 所有的查询动作都会被转发，联邦资源的请求会转发到karmada control-plane；Pod/Event单集群资源会被转发到cluster-gateway



Kubernetes API 兼容

给用户签发访问 multicluster-apiserver 的证书，用于直接使用 kubectl 操作多集群资源

- kubectl api-resources
  展示多集群管控面支持的资源类型，目前我们只对用户提供有限的资源操作。

- kubectl get clone rsi-app
  和操作单集群资源一致，可以查看联邦资源 cloneset 的运行状态

- kubectl get pod
  列出被多集群纳管的pod列表，在cluster-gateway 组件中自定义 pod 资源的 Table 格式，支持展示 pod 所在集群的信息

```
➜  ~ kubectl api-resources
NAME              SHORTNAMES   APIVERSION                     NAMESPACED   KIND
configmaps        cm           v1                             true         ConfigMap
events            ev           v1                             true         Event
namespaces        ns           v1                             false        Namespace
pods              po           v1                             true         Pod
secrets                        v1                             true         Secret
clonesets         clone        apps.kruise.io/v1alpha1        true         CloneSet
workloadscalers   wls          crd.xiaohongshu.com/v1alpha1   true         WorkloadScaler
➜  ~ kubectl get clone rsi-app
NAME      DESIRED   UPDATED   UPDATED_READY   READY   TOTAL   AGE
rsi-app   100       100       68              68      100     71d
➜  ~ kubectl get pod
NAME            READY   STATUS    RESTARTS   CLUSTER   AGE
rsi-app-tqf6g   1/1     Running   0          test3     7m43s
rsi-app-xdtr8   1/1     Running   0          test3     7m43s
rsi-app-5mrnb   1/1     Running   0          test3     7m40s
rsi-app-zsdpq   1/1     Running   0          test3     7m39s
rsi-app-2g5n2   1/1     Running   0          test3     7m38s
rsi-app-n94nt   1/1     Running   0          test3     7m36s
rsi-app-m2vrl   1/1     Running   0          test3     7m34s
rsi-app-fdbhv   1/1     Running   0          test3     7m33s
rsi-app-6k97j   1/1     Running   0          test3     7m31s
rsi-app-q5b8g   1/1     Running   0          test3     7m30s
rsi-app-xhqcj   1/1     Running   0          test3     7m28s
rsi-app-g27wm   1/1     Running   0          test3     7m25s
rsi-app-s6l69   1/1     Running   0          test3     7m24s
rsi-app-5czqc   1/1     Running   0          test3     7m22s
rsi-app-zlnnp   1/1     Running   0          test3     7m19s
rsi-app-2j5jx   1/1     Running   0          test1     55s
rsi-app-2jtdg   1/1     Running   0          test3     55s
```

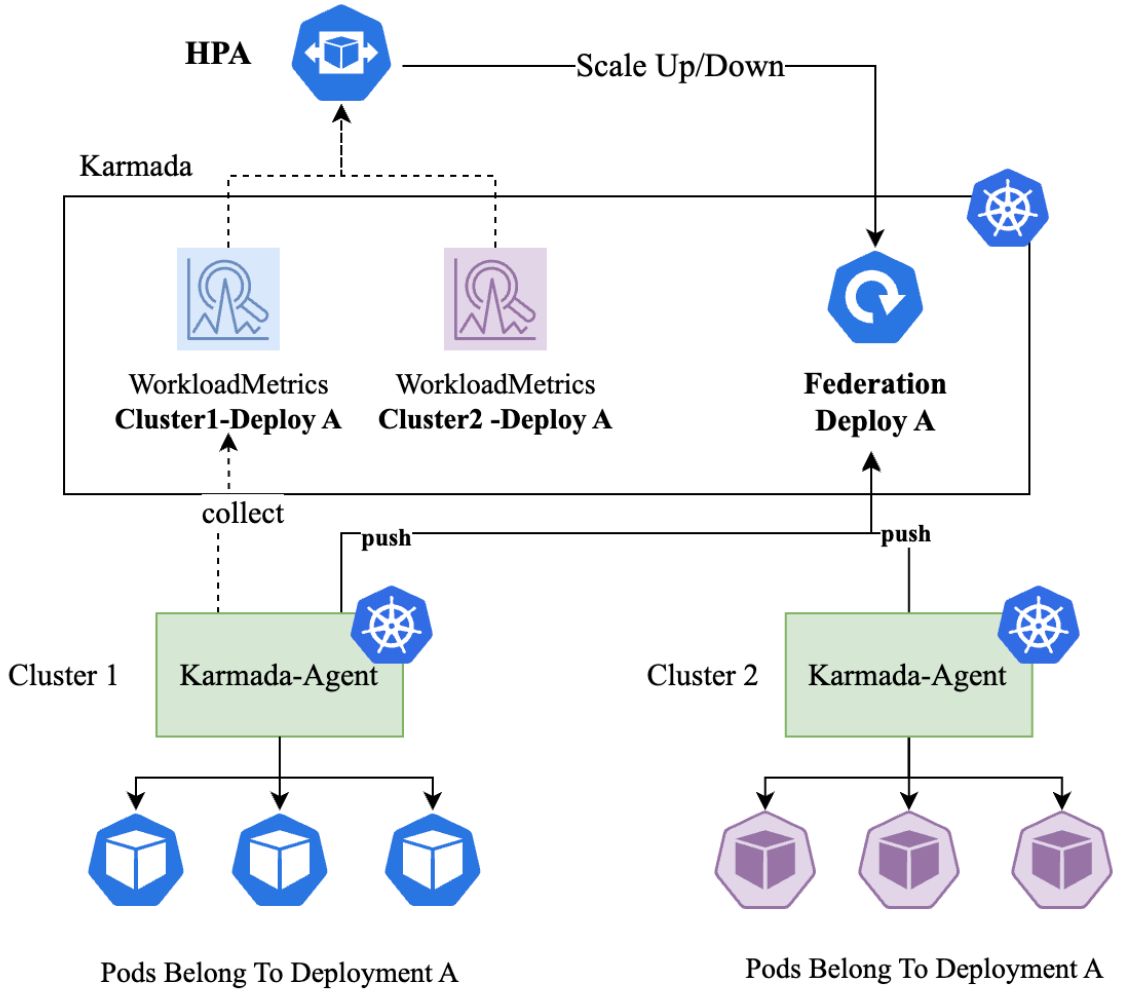
多集群场景下的弹性建设

1. Karmada-agent 以工作负载的粒度push各自集群管理的负载的指标信息到自定义WorkloadMetrics

2. 联邦侧的HPA组件整合多个集群的 WorkloadMetrics 决定联邦负载的副本数

3. 副本修改及对应的分发复用副本伸缩的链路及相应的调度策略。

4. 减少联邦侧组件对单集群控制面的直接访问

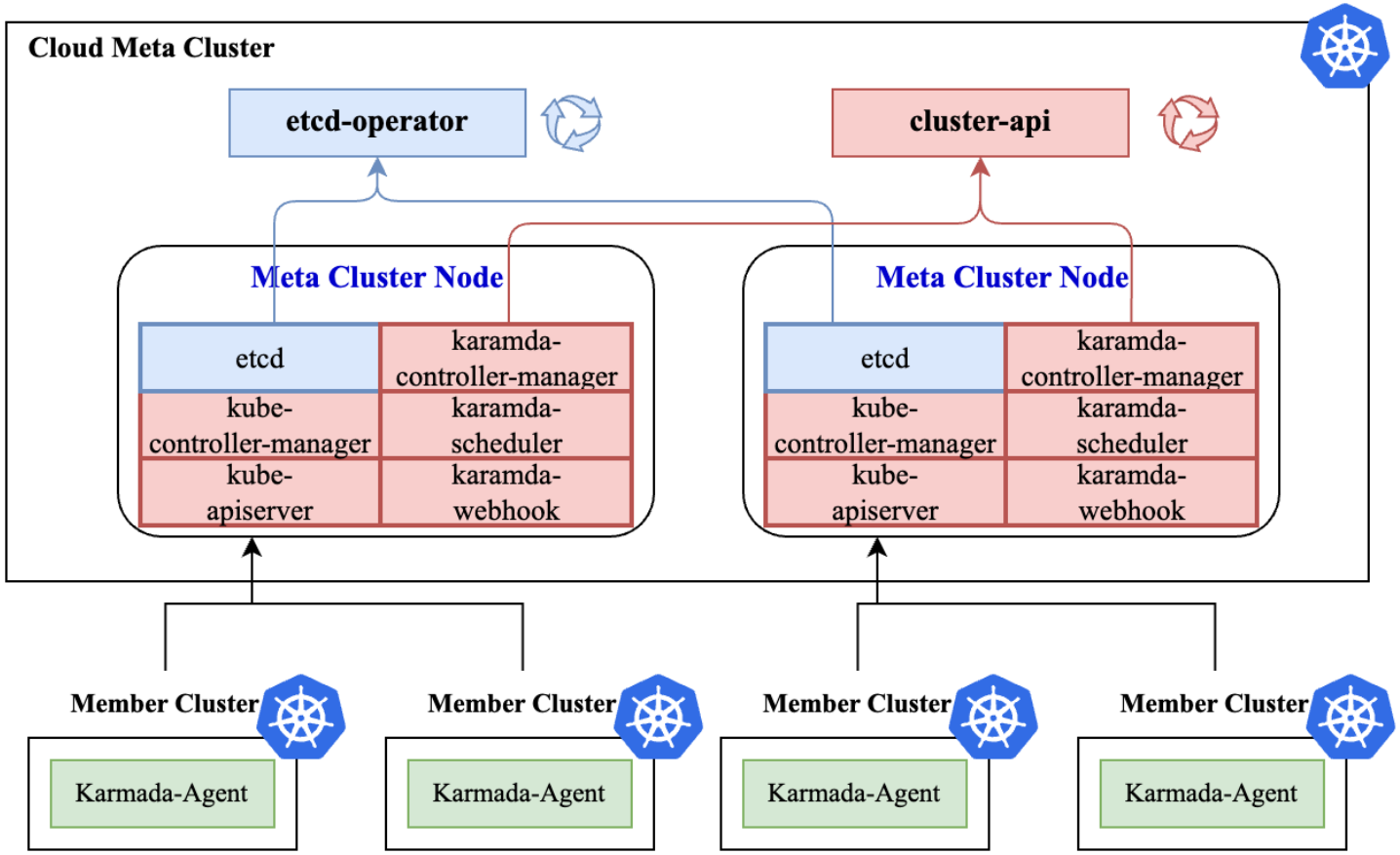


控制面自动化运维

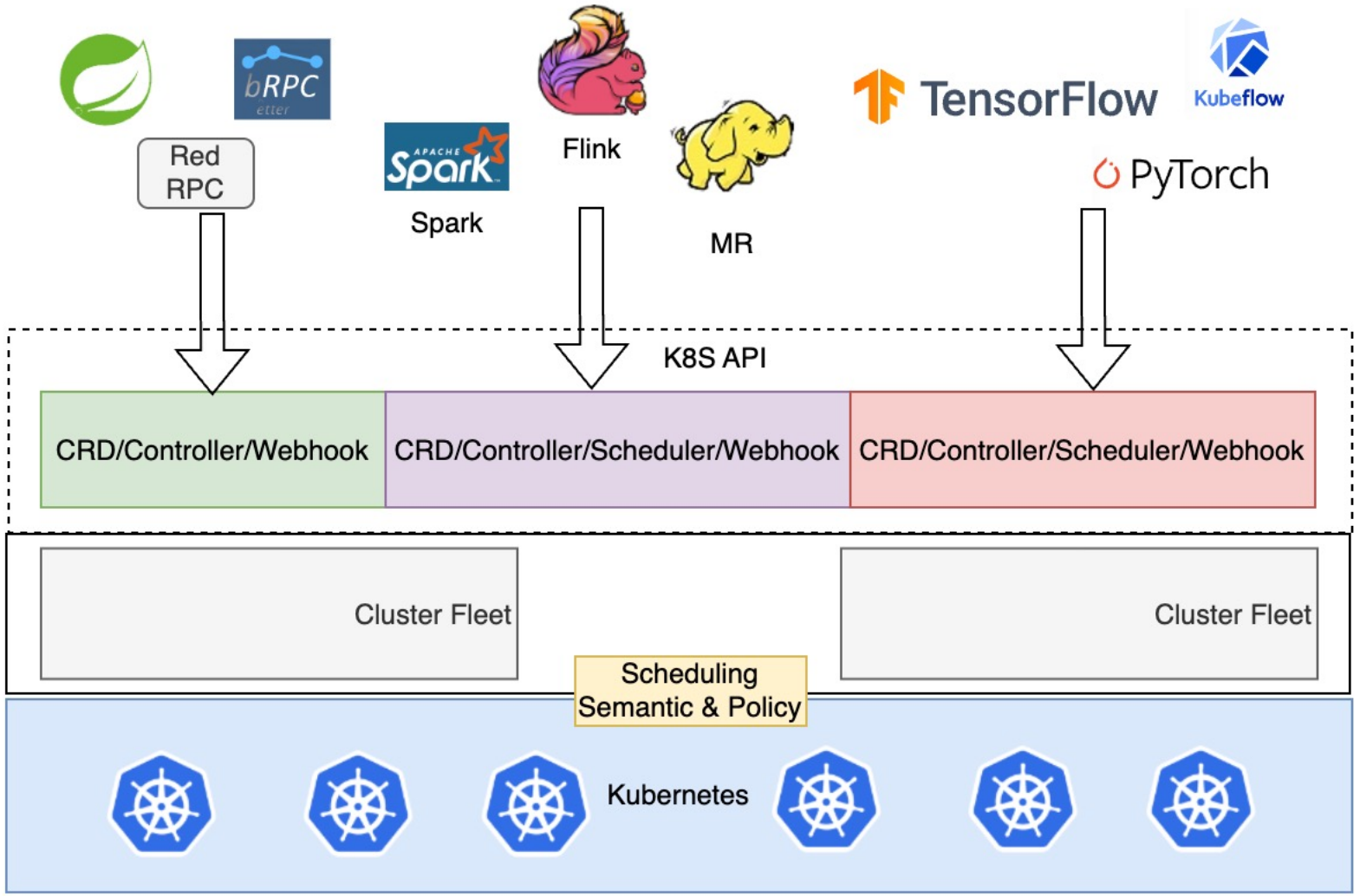
1. 多集群控制面使用KoK的部署模式，基于cluster-api声明式的管理管控组件

2. 多集群控制面部署在云上元集群中，在元集群内部署了相应的 etcd-operator 和 cluster-api operator 负责管控组件的创建和生命周期管理

3. karmada-agent 使用内部的插件机制部署到成员集群



规划与展望



1. 更全面的应用形态覆盖：无状态/有状态应用、大数据应用、AI 工作负载

2. 解决运维、管控问题 => 帮助提升资源效能