Kubernetes多集群架构的思考、实践和探索

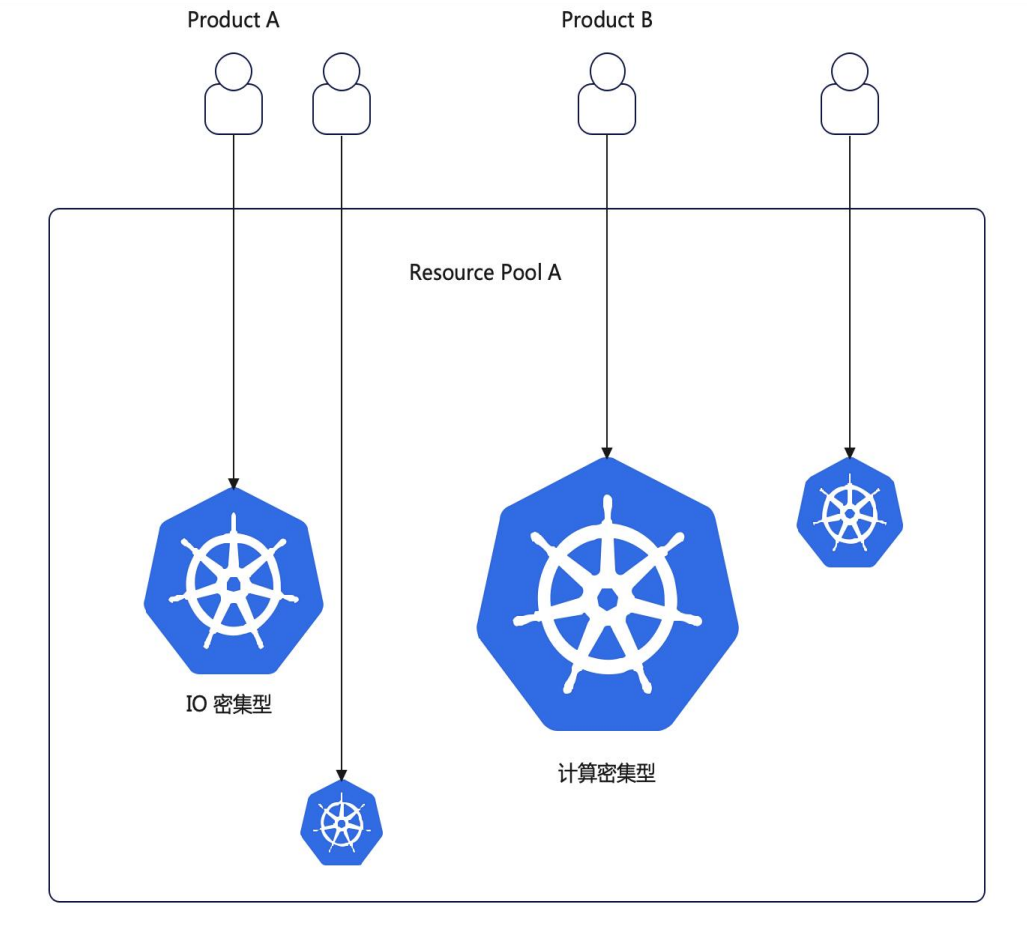
段朦 移动云

Content

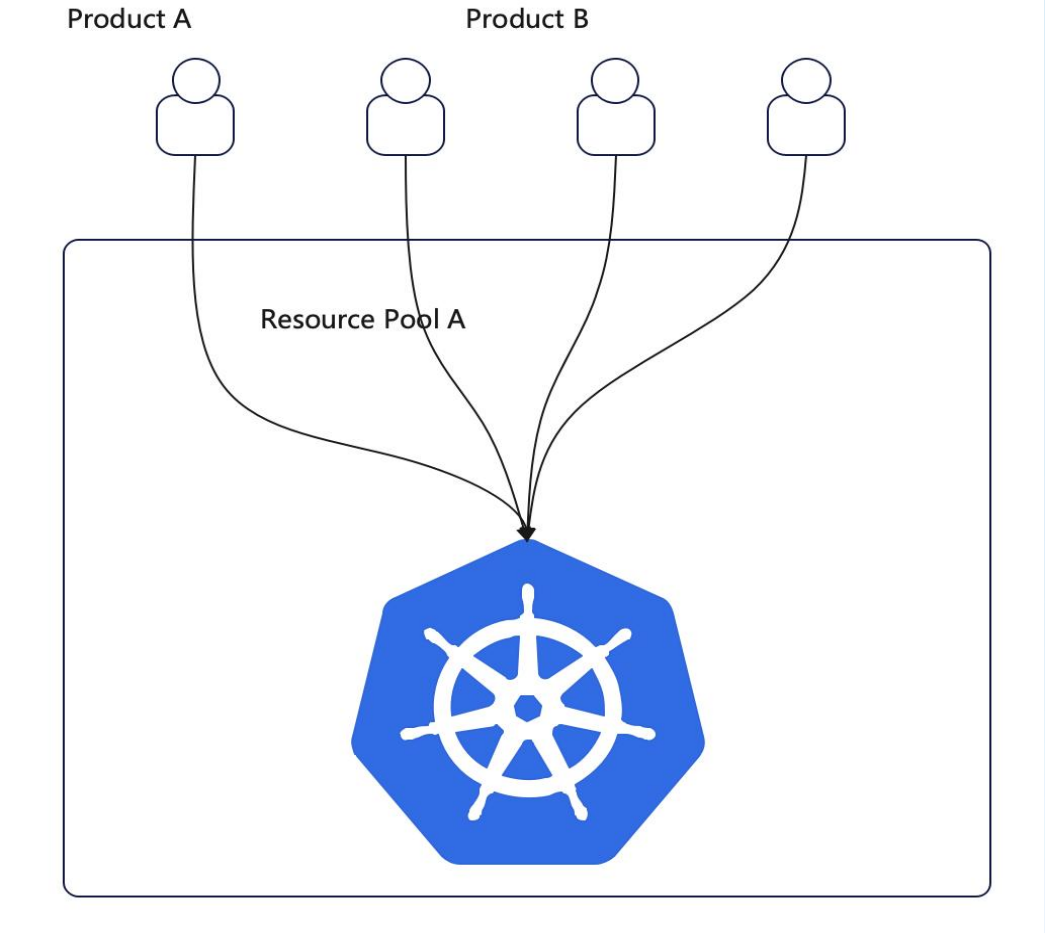
目录

Why do we need multicluster：Resource Utilization

初始态



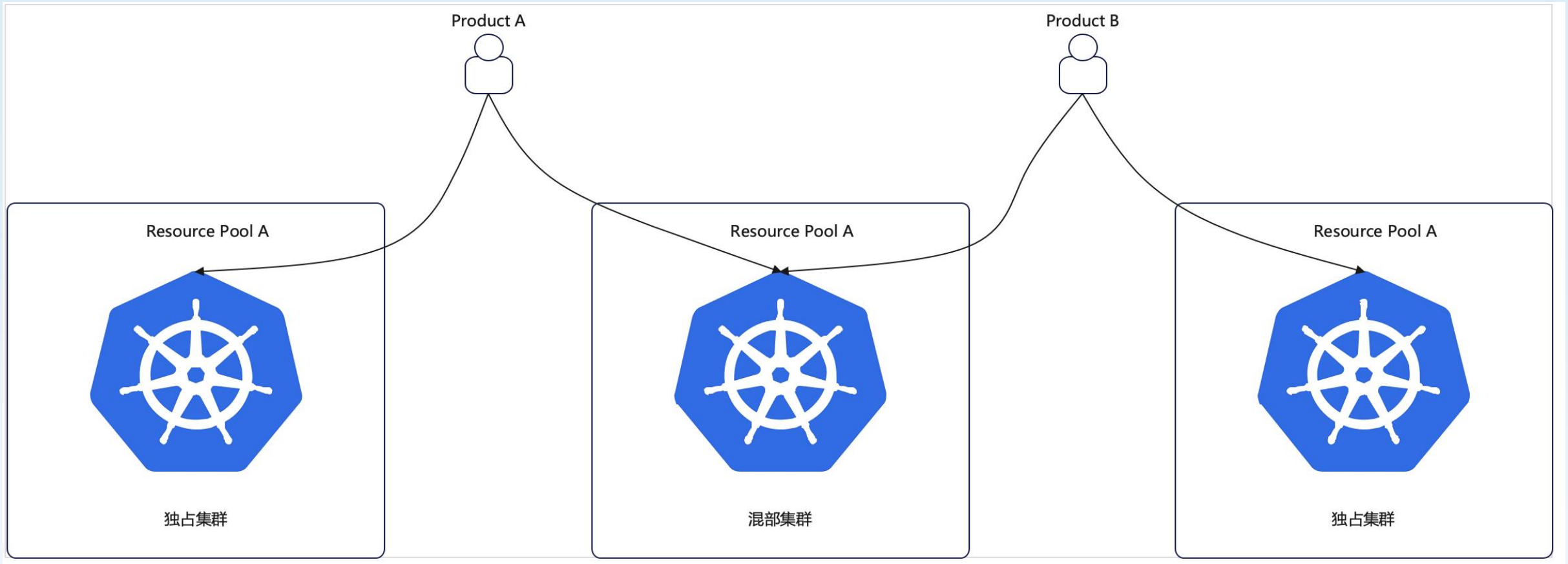
最终态



早期资源建设，产品条线单独申请建设k8s集群。带来的问题：集群的资源使用不均衡，带来资源浪费。

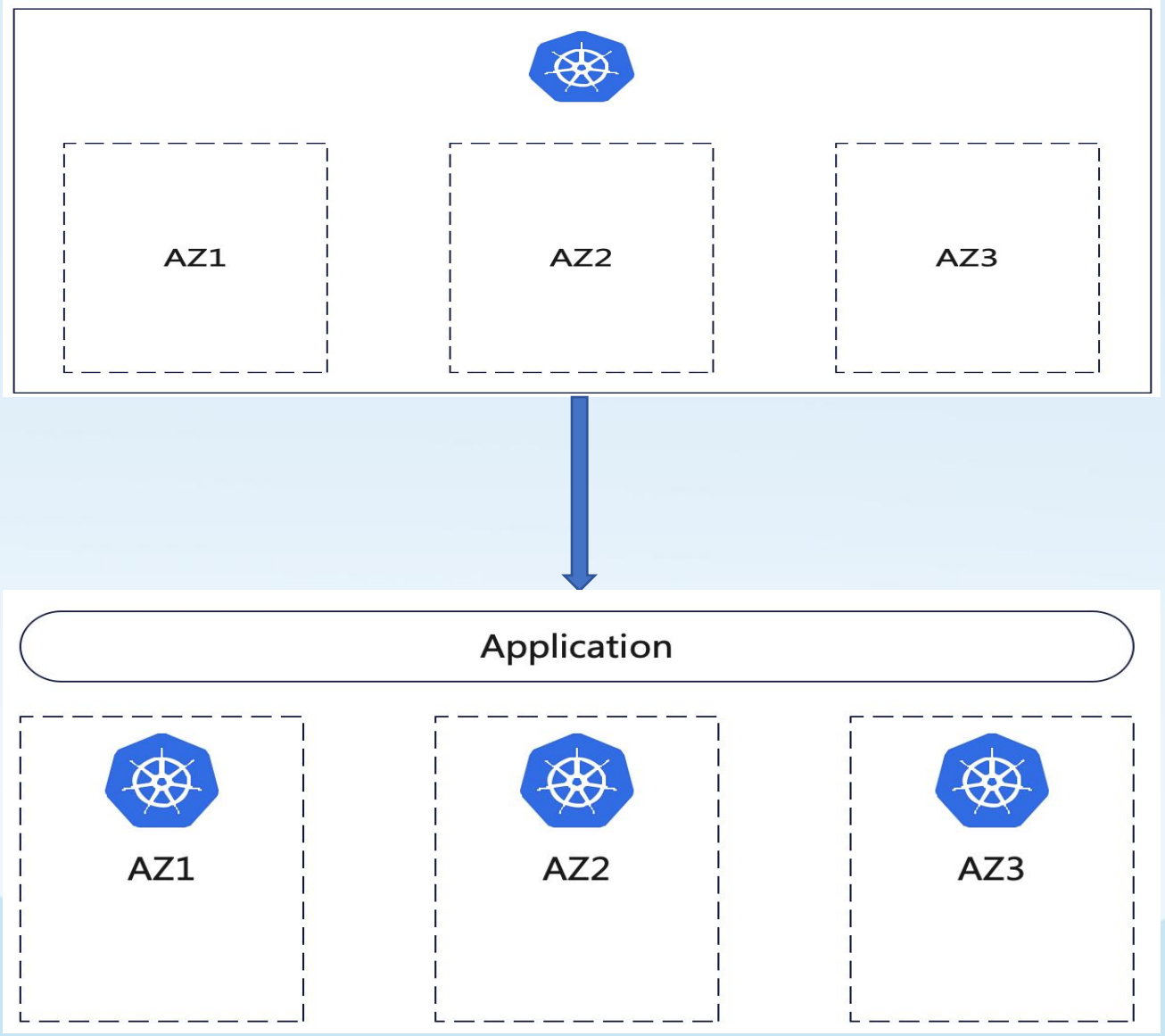
Why do we need multicluster：Resource Utilization

中间态



有状态实例无法平滑迁移，因此需要多集群管理的中间状态。将多集群下沉到基础设施，减小重复工作量

Why do we need multicluster：Hign Availability



初始态

最终态

Content

目录

Problems

业务诉求

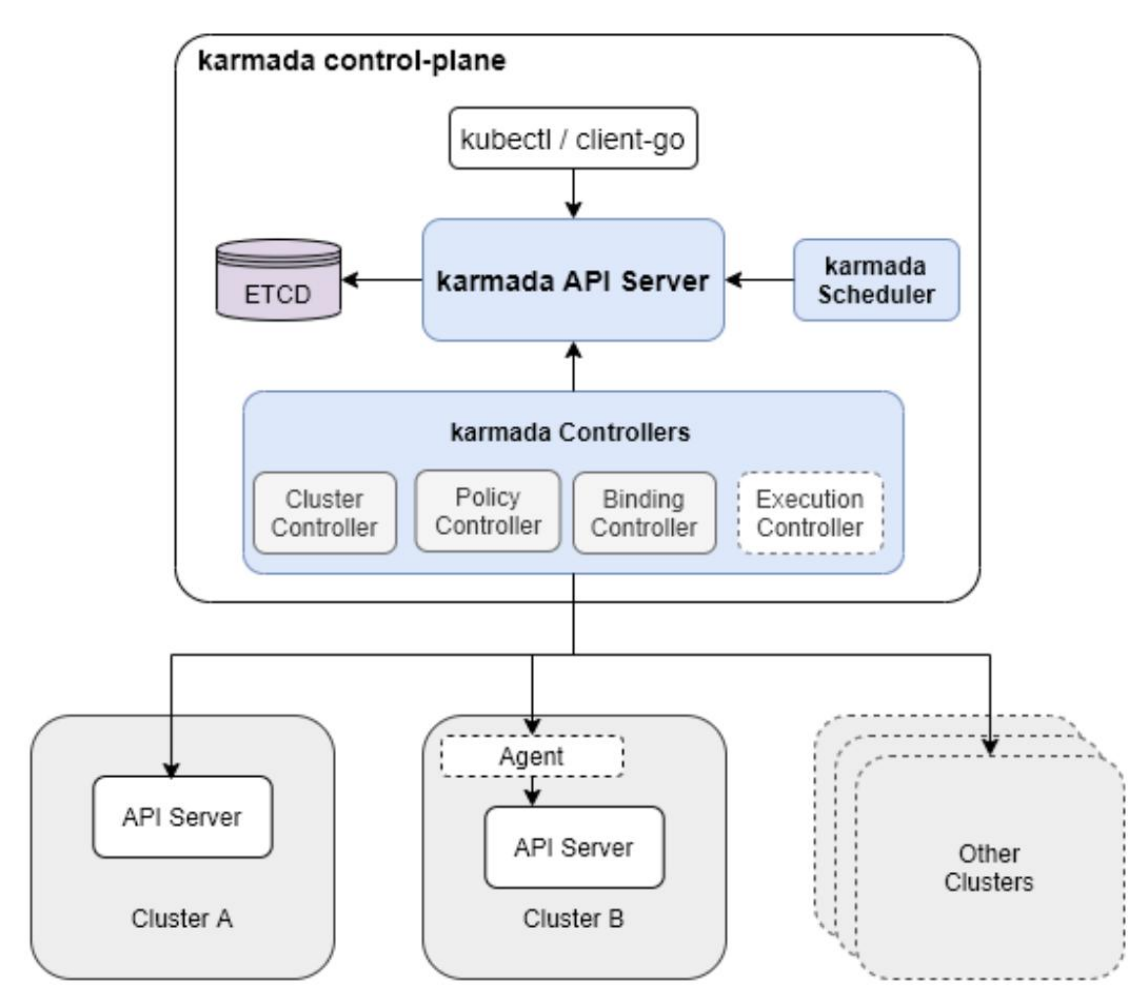
- ✓ 让用户像使用单集群一样使用多集群，业务侧多集群无感知
- ✓ 业务侧无需改造或改造成本分钟级，业务侧产品无需改动代码，或者改造代码量很少
- ✓ 业务侧平滑升级，业务侧从单集群到多集群的改造无需停服，业务流量平滑切换

多集群要解决的问题

- ✓ 多集群服务的分发部署
- ✓ 跨集群自动迁移与调度
- ✓ 多集群网络通信
- ✓ 多集群服务发现

Problems：Scheduling & Orchestration

按工作负载为维度调度： Karmada，clusternet



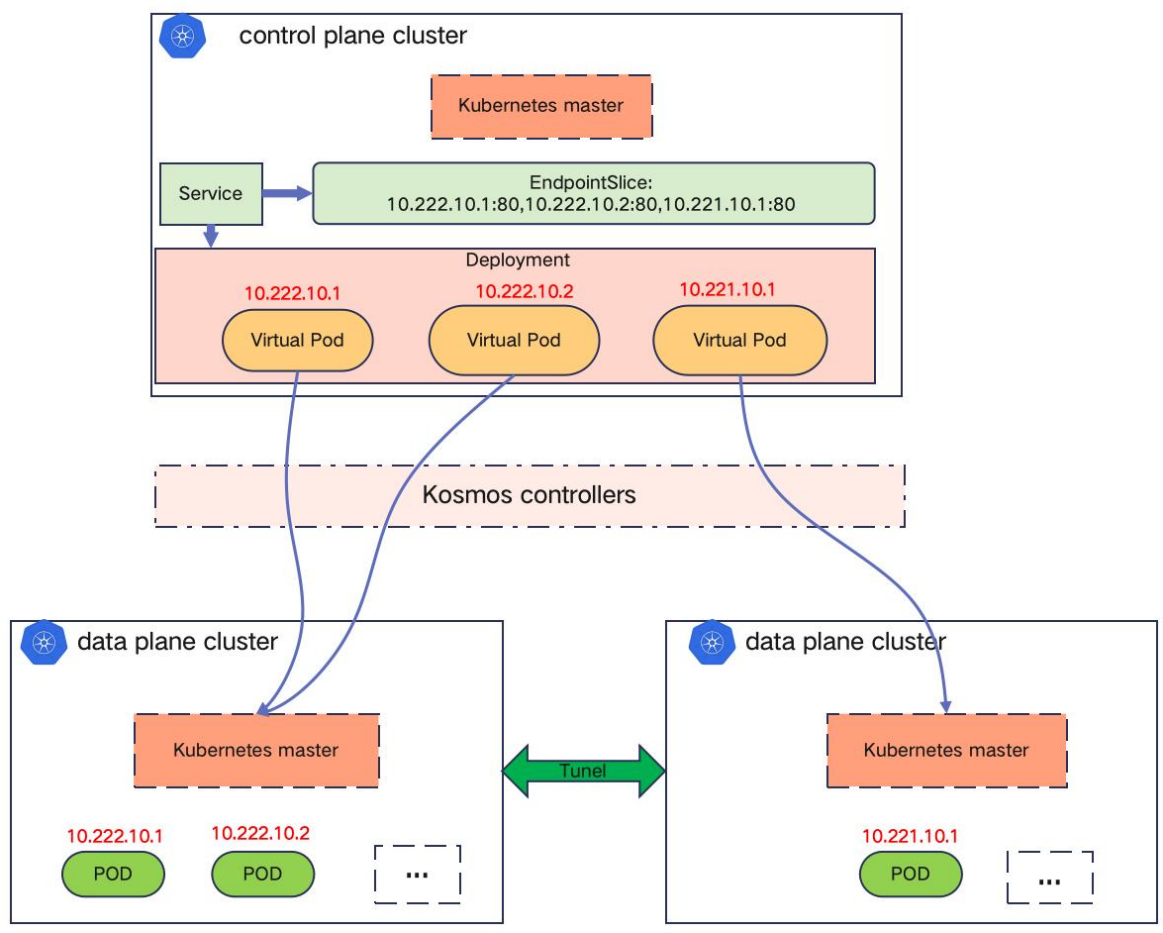
社区多集群的探索历程：kubefed v1，kubefed v2，karmada

- ✓ 原生k8s接口
- ✓ 集群级别调度
- ✓ 分发策略对资源分发
- ✓ 控制器状态同步
- ✓ 统一资源汇聚

ref to：https://karmada.io/docs/core-concepts/architecture

Problems：Scheduling & Orchestration

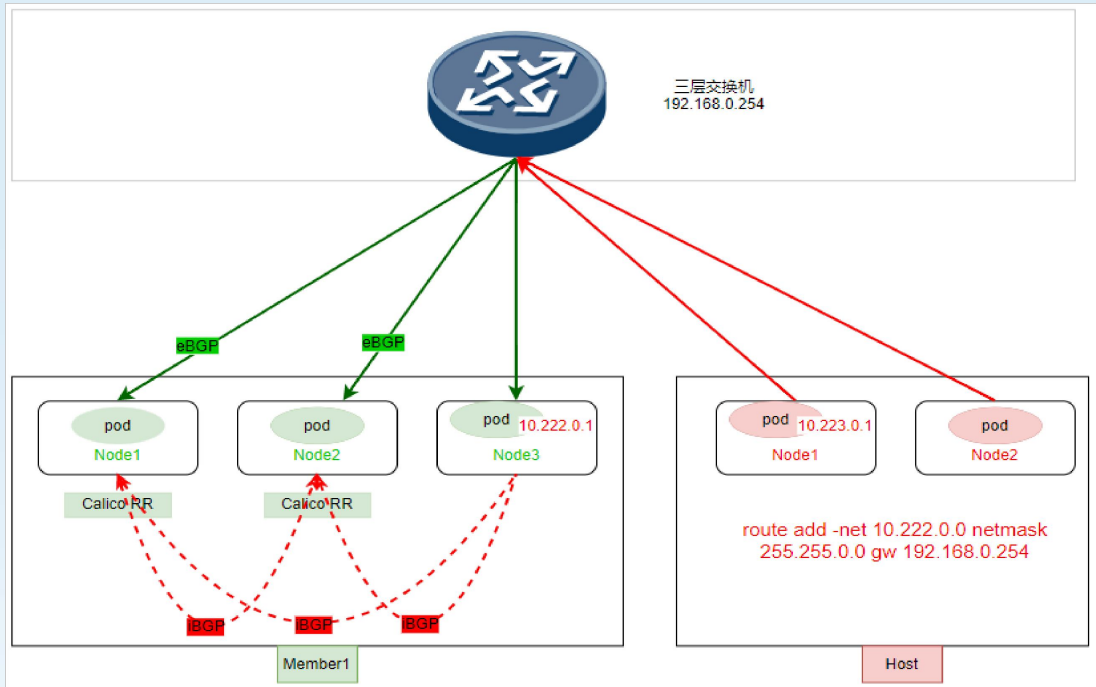
按pod为维度调度：virtual—Kubelet，Kosmos—clustertree



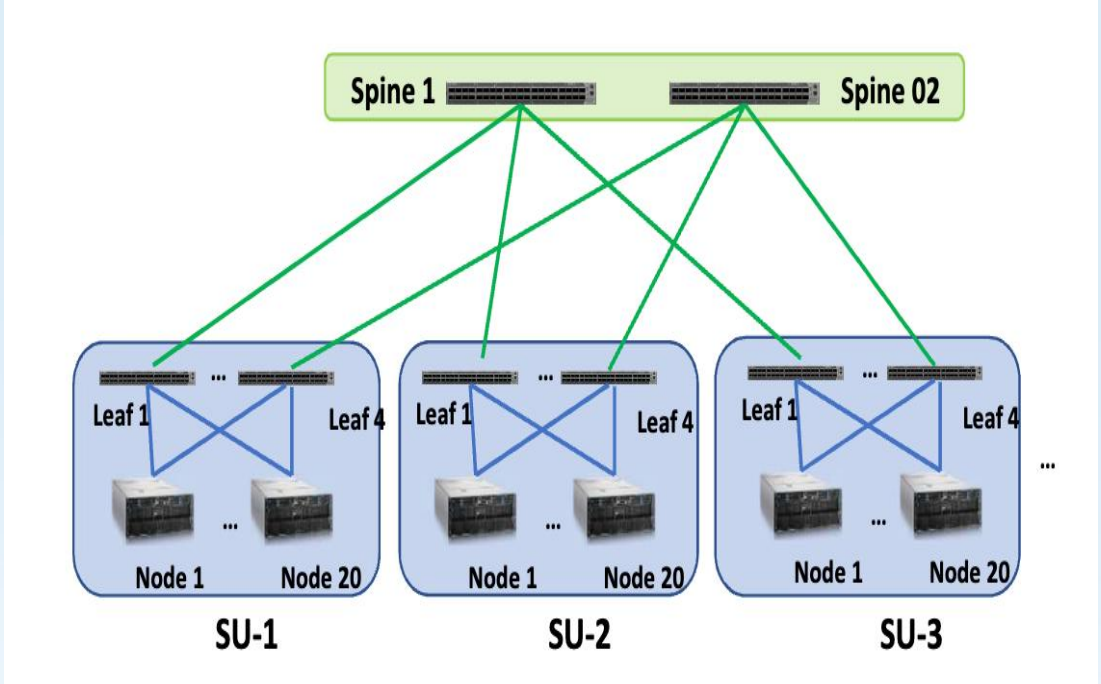
- ✓ 原生k8s接口，百分百兼容
- ✓ 天然支持有状态数据库应用和operator等应用

Problems：multi-cluster-network

直接路由方式打通集群网络：BGP



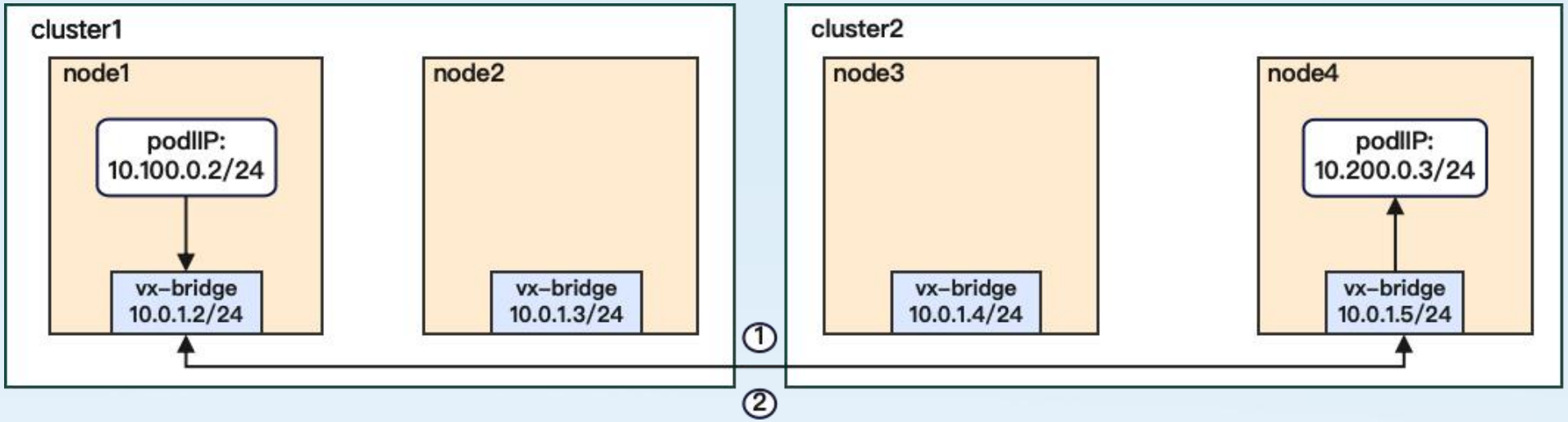
- 前提是网段不冲突
- 配置跨集群的路由
- 不在同一网络自治域 –> BGP
- 配置spine和leaf交换机对等



- 优点：走三层物理网络，性能上大于隧道
- 缺点：网络建设复杂，增大运维成本；不适合网络自治域较远的集群（网络链路太长）

Problems：multi-cluster-network

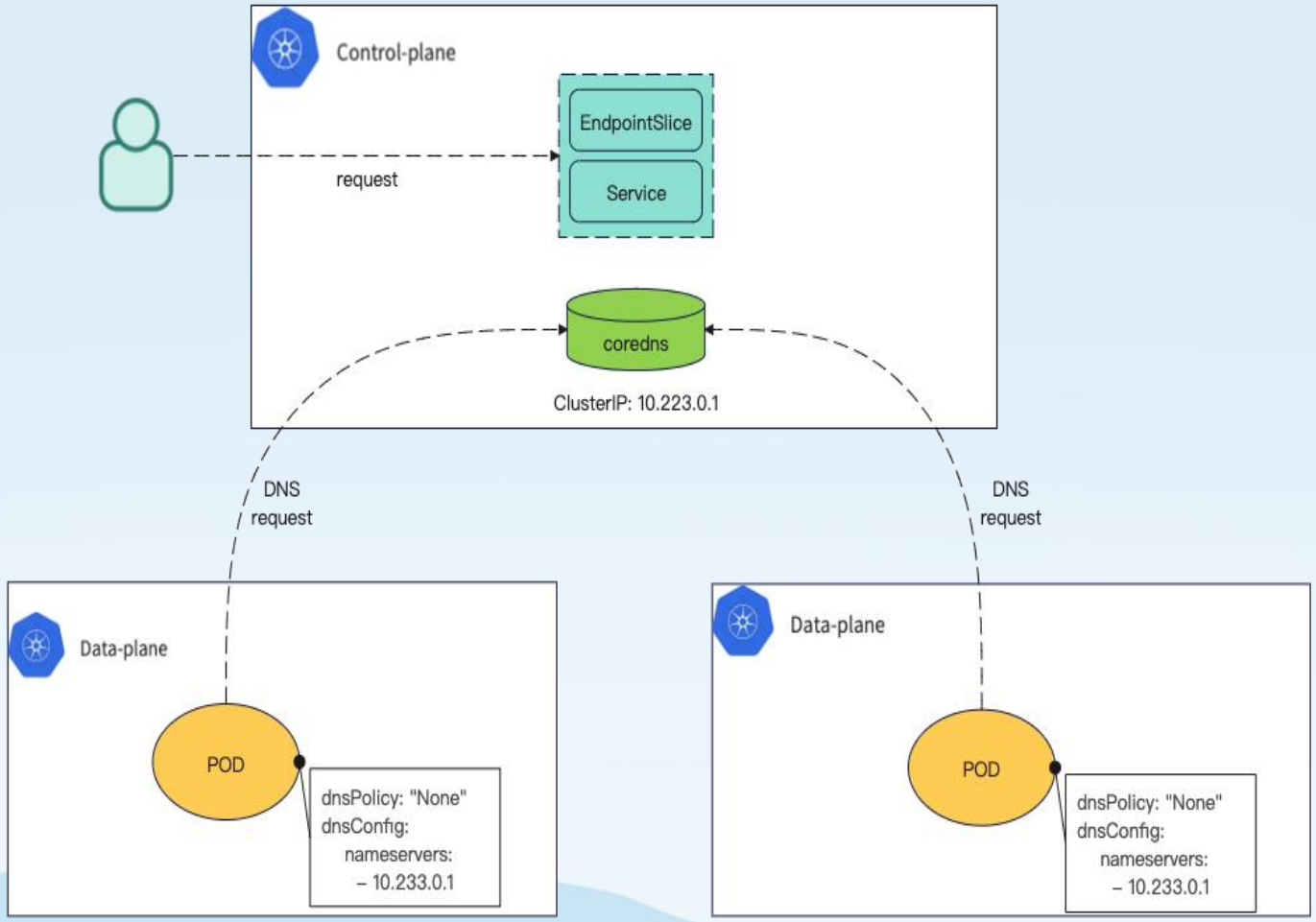
隧道方式打通集群网络：submariner，Kosmos—clusterlink



- 屏蔽底层的物理设备
- overlay层面二层互通

Problems：multi-cluster-dns

全局多集群core—dns：Kosmos—multi—dns



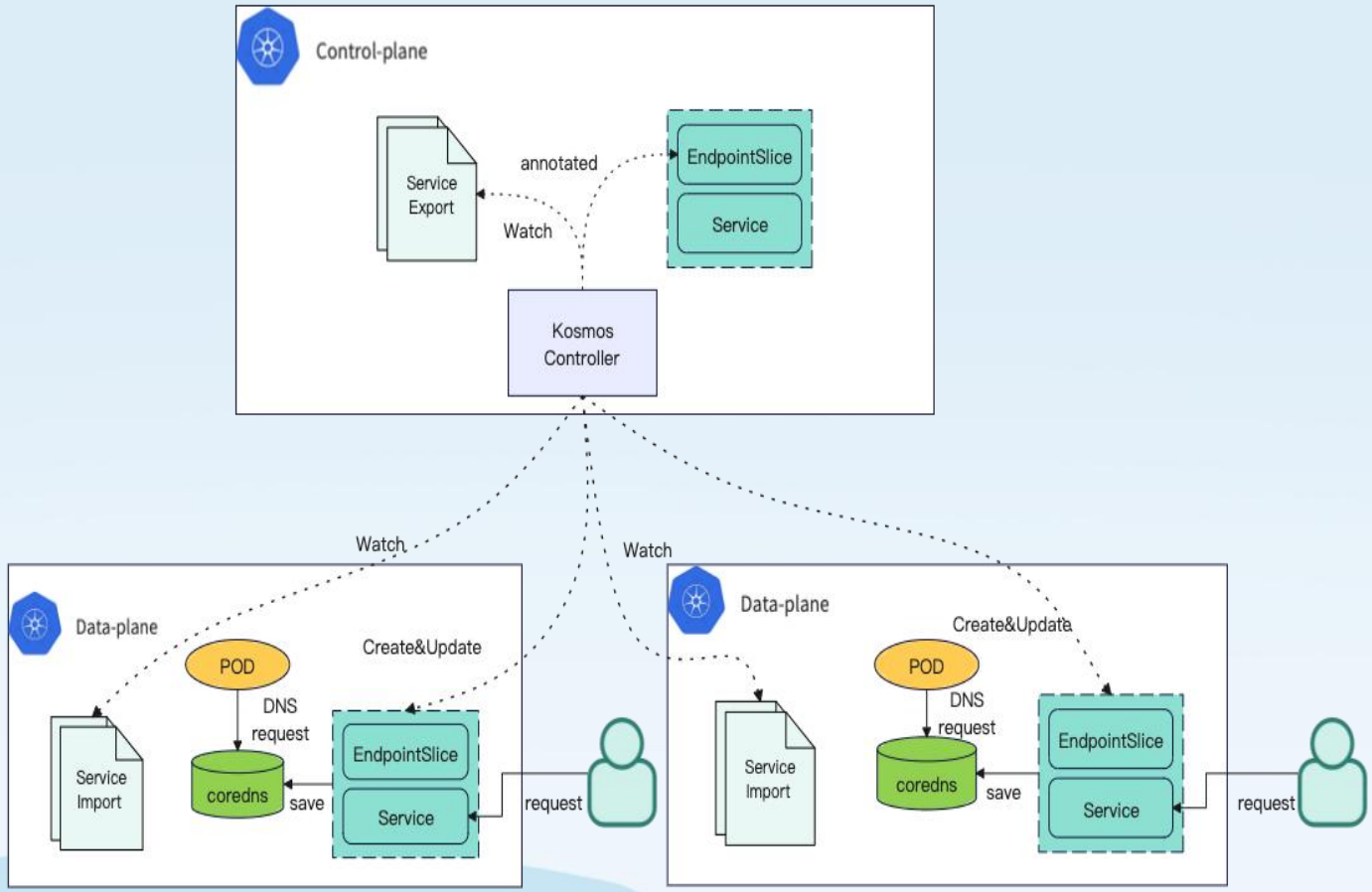
监听所有子集群的service和endpoint事件，维护所有子集群的域名解析记录

优势：速度快，不需要通过控制器下发资源，使用简单

缺点：集群规模增大，内存压力大

Problems：mcs

多集群MCS：Kosmos—clustertree



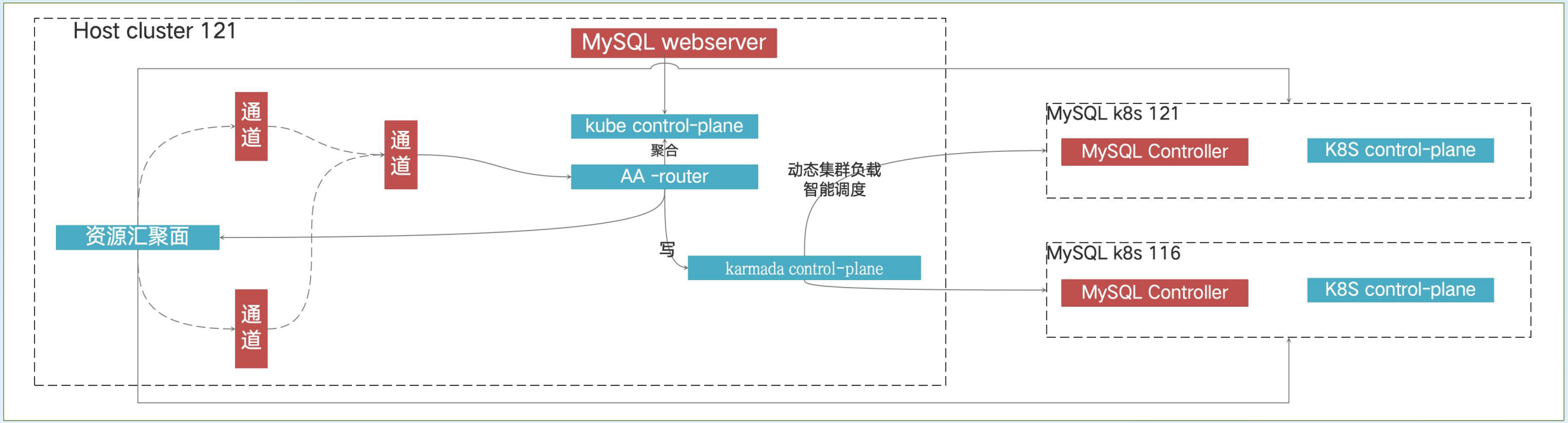
监听serviceImport和serviceExport资源，并通过控制器将service创建到子集群

优势：社区标准方案，已形成业界标准

Content

目录

多集群落地一 Karmada 支撑mysql落地



多集群事件汇聚，提供统一watch管道

智能路由，读写分离

根据不同集群负载智能调度

- 架构：控制面部署在主集群（控制台、operator等），mysql实例在子集群
- 集群网络：采用三层物理路由的方式（calico RR加交换机和节点BGP对等）
- 资源编排和调度：资源读写分离，写—> karmada—spiserver（通过控制器创建propagationPolicy和overridePolicy） 读—>karmada—search
- 要求：子集群资源名不能重名

多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

自研多集群网络连通工具cluster-link

为什么采用隧道方式：

- 物理路由打通集群网络建设复杂
- 跨集群网络主要是管理流量，就算隧道的方式有性能衰减（10%-25%），跨集群的隧道流量可能只占整个管理服务流程的一小部分，影响较小
- 考虑到后续公网混合集群的情况，隧道或VPN的方式更加方便

→ 通过隧道的方式打通集群网络

为什么要自研：

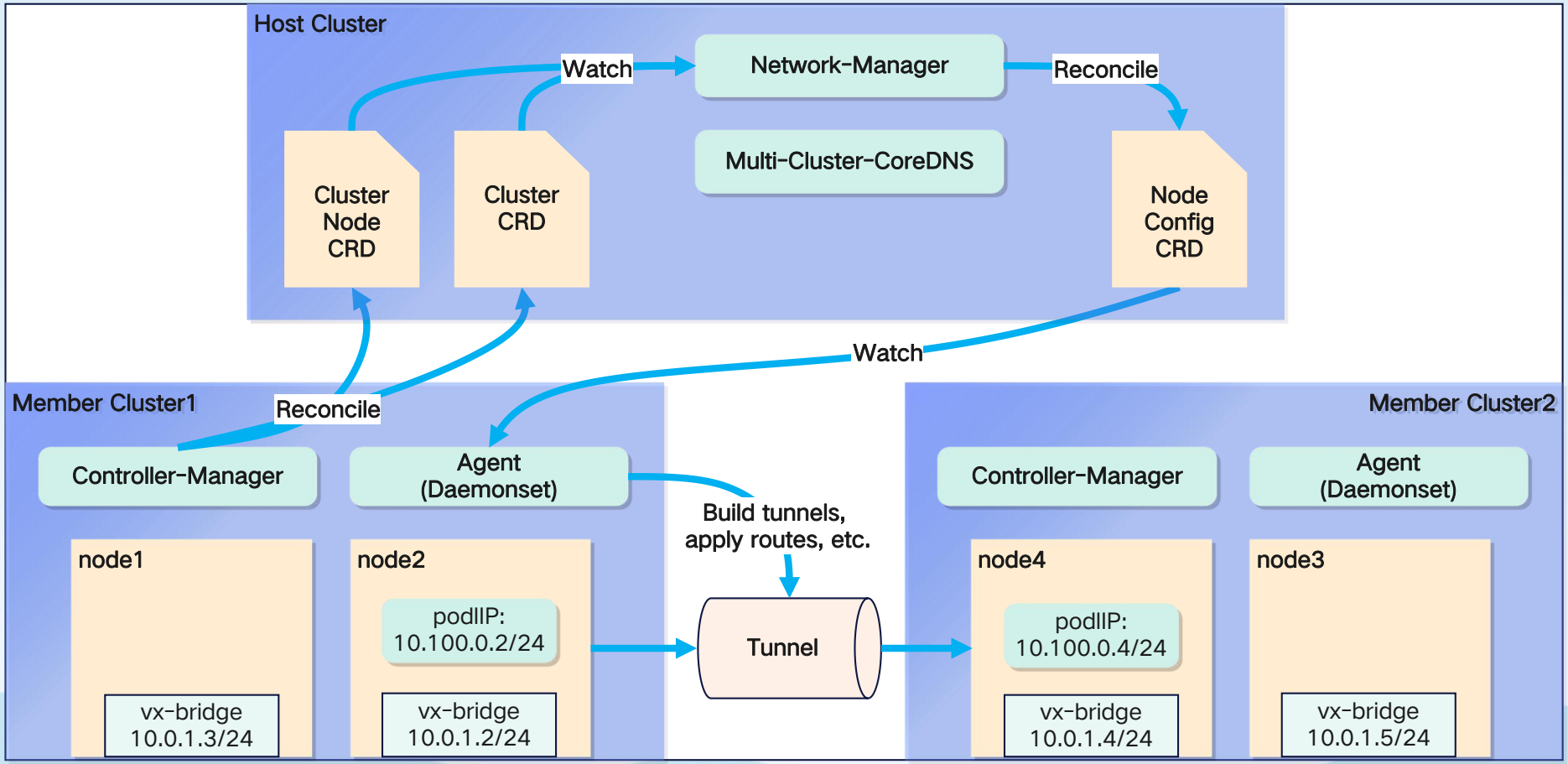
- 多集群网络没有形成标准化的产品
- 业界产品submariner的针对场景不同
- 性能问题

→ kosmos-cluster-link

https://github.com/kosmos-io/kosmos/tree/main/cmd/clusterlink

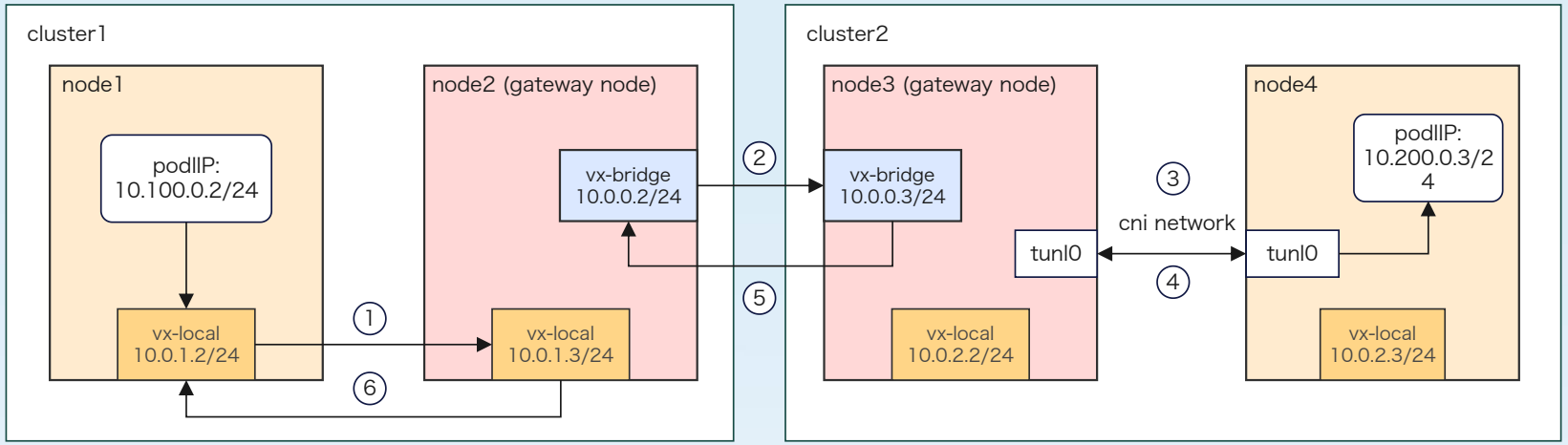
多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

ClusterLink：使用隧道技术（VxLAN/IPSec）打通跨集群网络。在CNI上层实现，用户无需卸载或重启已经安装的CNI插件，不会对正在运行的pod产生影响。



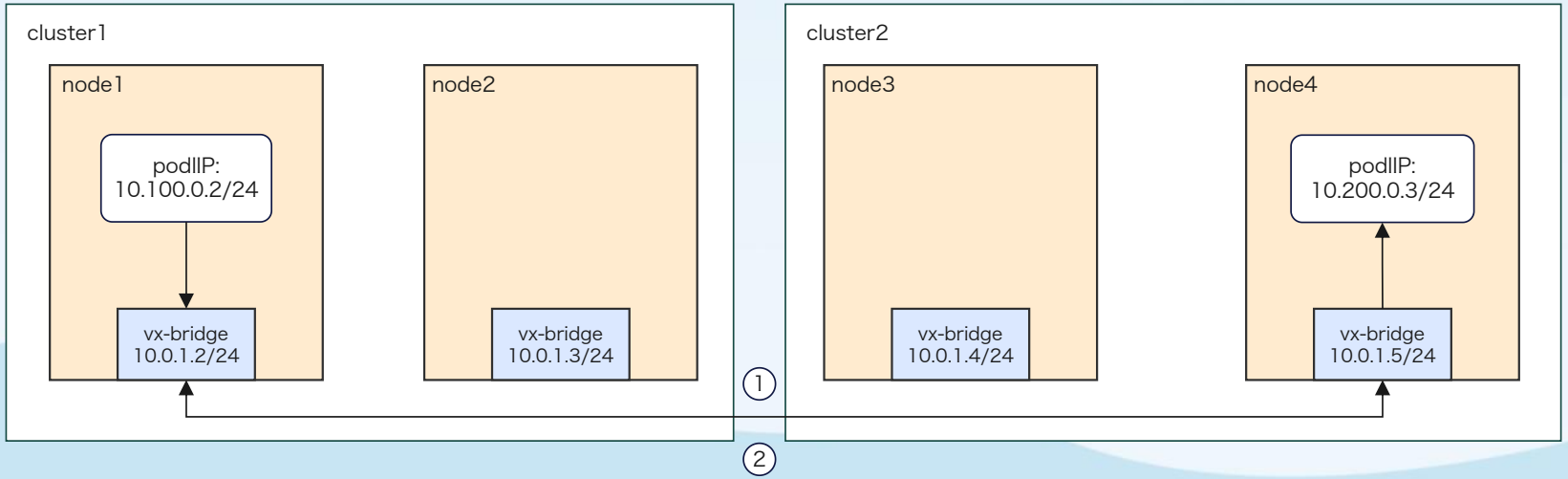
多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

Gateway Mode



- 兼容性强，每个集群只需1个节点对外提供访问（考虑HA时需要2个），适合跨云场景

P2P Mode



- 网络路径短，性能更优。适用于全节点underlay互通场景

p2p和gateway模式支持混合使用

多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

纳管存量集群时，挑战之一是需要考虑集群之间的网段冲突场景。即，集群联邦中存在多个pod拥有相同的IP。对此，ClusterLink支持为Pod或者Service配置全局IP。

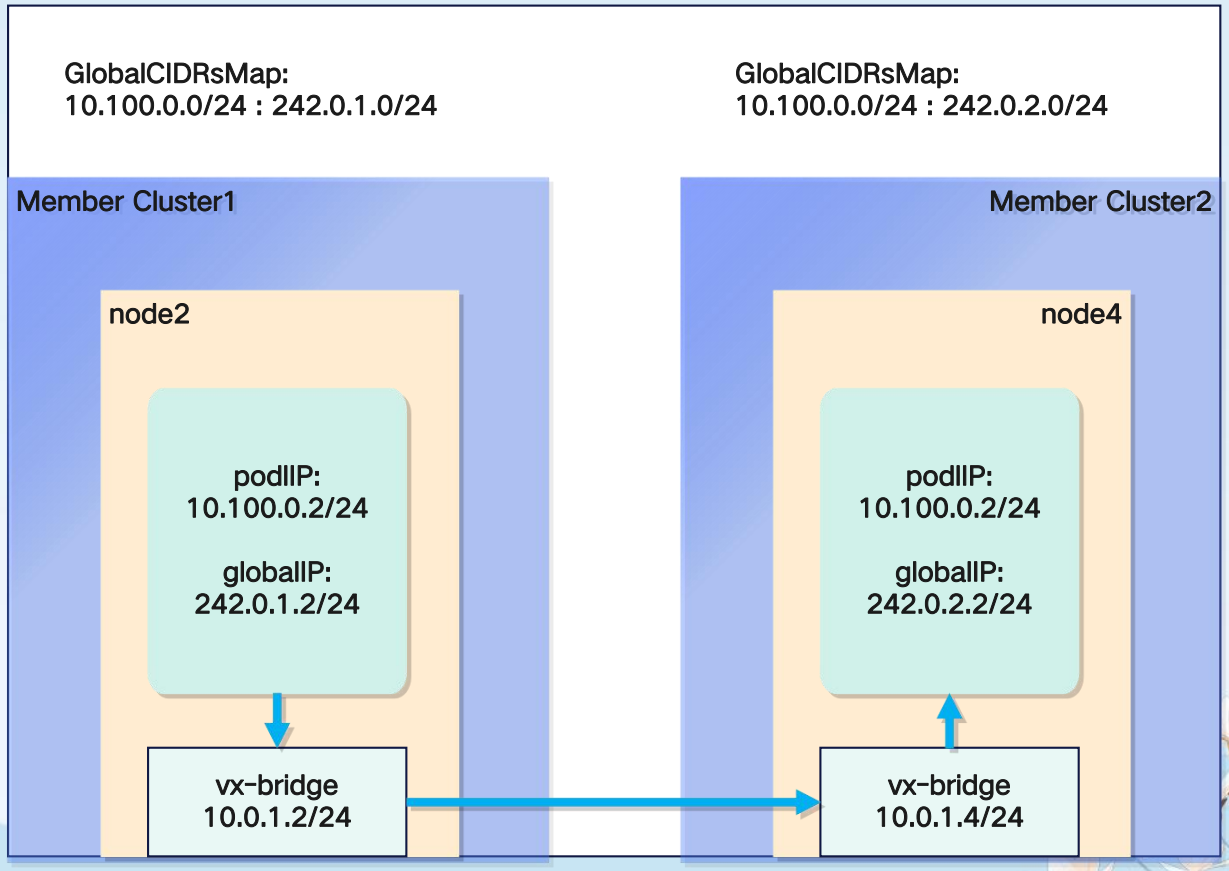
优势：iptables 的 NETMAP target 可以大大减少 iptables 数量

① ip route add 242.0.2.0/24 via 10.0.1.4 dev vx-bridge

② iptables -t nat -A POSTROUTING -s 10.100.0.0/24 -j SNAT --to-source 242.0.1.0/24

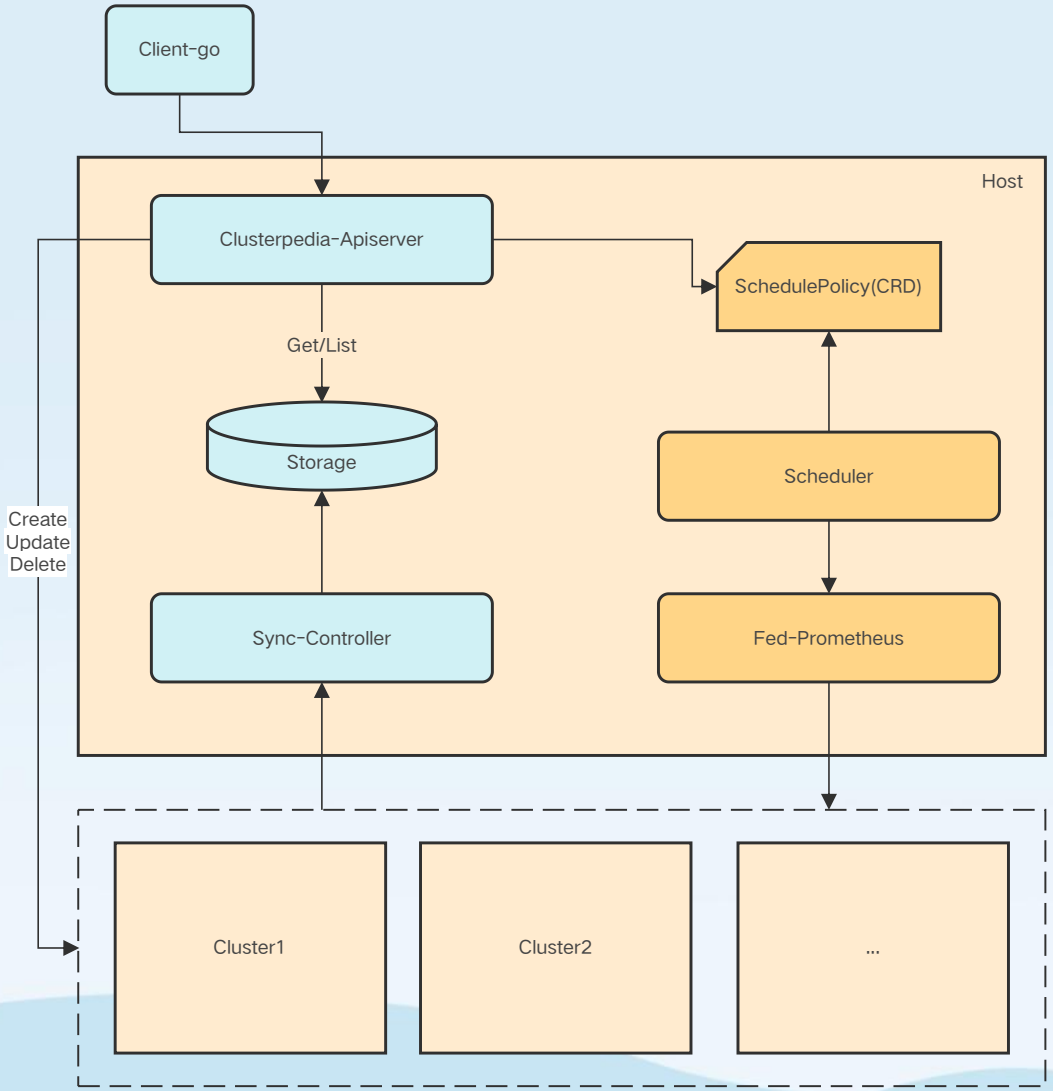
③ iptables -t nat -A PREROUTING -d 242.0.2.0/24 -j NETMAP --to 10.100.0.0/24

④ ip route add 242.0.1.0/24 via 10.0.1.2 dev vx-bridge



多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

基于Clusterpedia的多集群控制面，以工作负载为维度调度，将工作负载动态分发到不同集群



多集群控制面基于Clusterpedia开发，新增了以下能力：

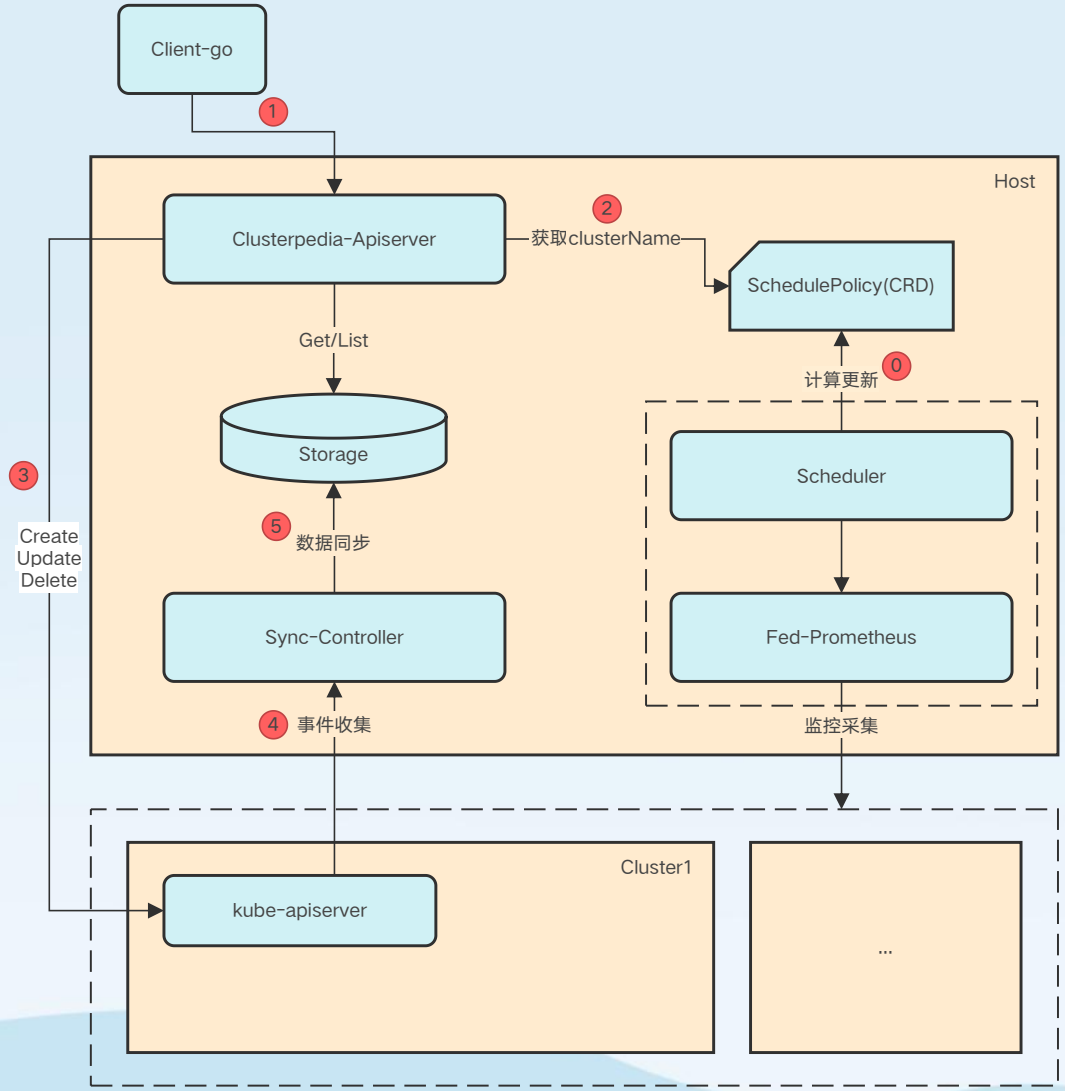
- Apiserver 现在可以处理 Create、Delete、Update、Watch请求。产品控制服务可以通过 Client-go 与 apiserver 交互，支持 informer。
- 新增调度模块，支持多种调度策略：静态调度、动态调度、伴随调度。动态调度结合Prometheus联邦，将实例下发到最合适的集群。

为什么选择Clusterpedia：

- 兼容多k8s版本
- get/list 性能优于kube-apiserver
- 丰富的检索条件

多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

资源下发过程



0、定时任务，获取当前集群监控指标，计算更新最优集群列表到 schedulepolicy

1、client-go发起post请求：

```
// kubeconfig
apiVersion: v1
kind: Config
current-context: cluster1
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server:
https://{ip:port}/apis/clusterpedia.io/v1beta1/resources/schedulepolicies/{name}/
  name: cluster1
preferences: {}
...
```

2、结合调度策略取得最优 clusterName

3、clusterpedia-apiserver 将请求转发至子集群 kube-apiserver

4、synchro 获取资源 create 事件

5、写入 storage

多集群落地二 自研多集群网络，基于Clusterpedia的多集群编排

schedulePolicy设计

```
apiVersion: schedule.clusterpedia.io/v1alpha1
kind: SchedulePolicy
metadata:
  name: product-a
spec:
  clusterNames:
    - member1
    - member2
    - member3
  resources:
    ...
  policies:
    ...
```

集群范围限定

记录资源-调度策略的映射关系：

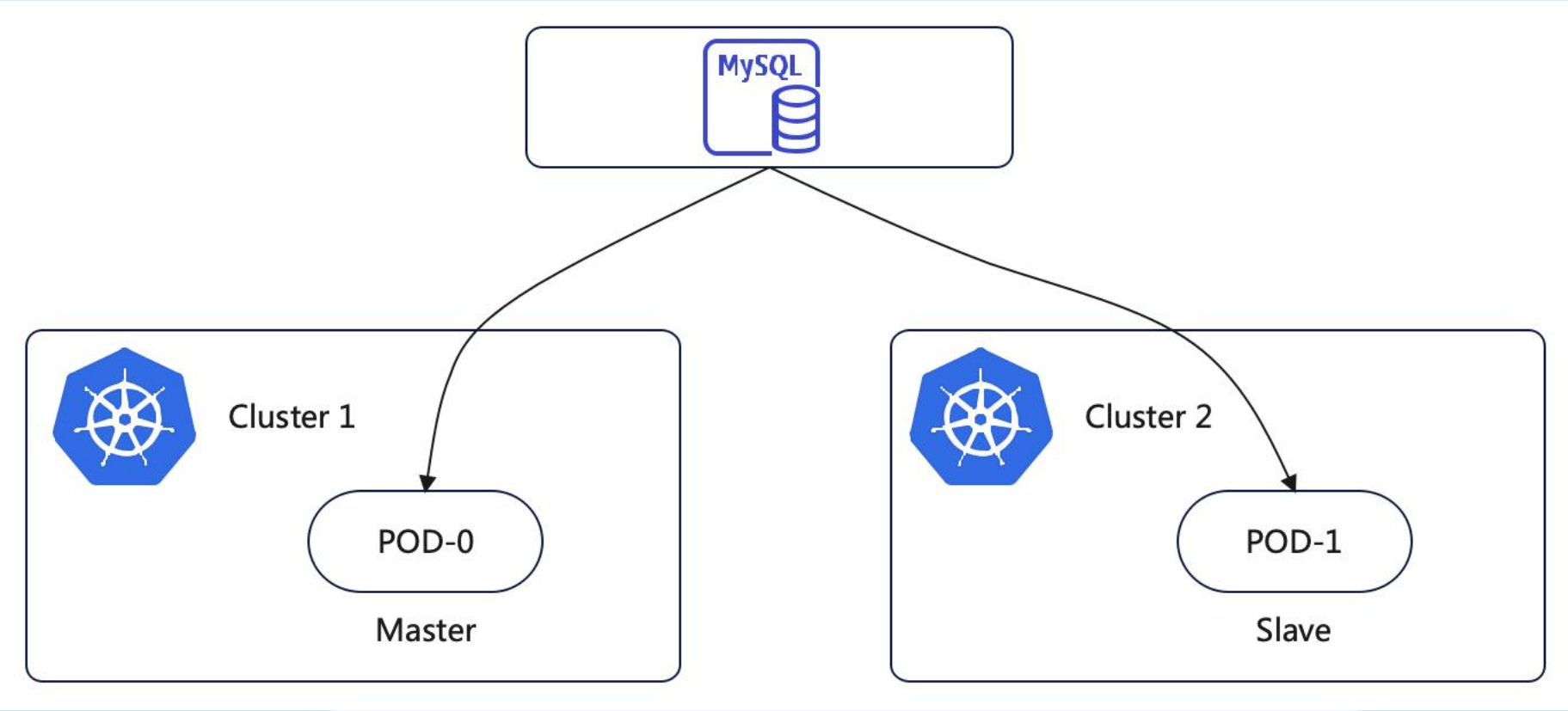
```
- group: product.group.io
    versions:
      - v1
    resources:
      - productCRs
    policy: dynamic
- group: apps
    versions:
      - v1
    resources:
      - jobs
    policy: follow-crd
```

具体的策略，策略共分为3种类型：

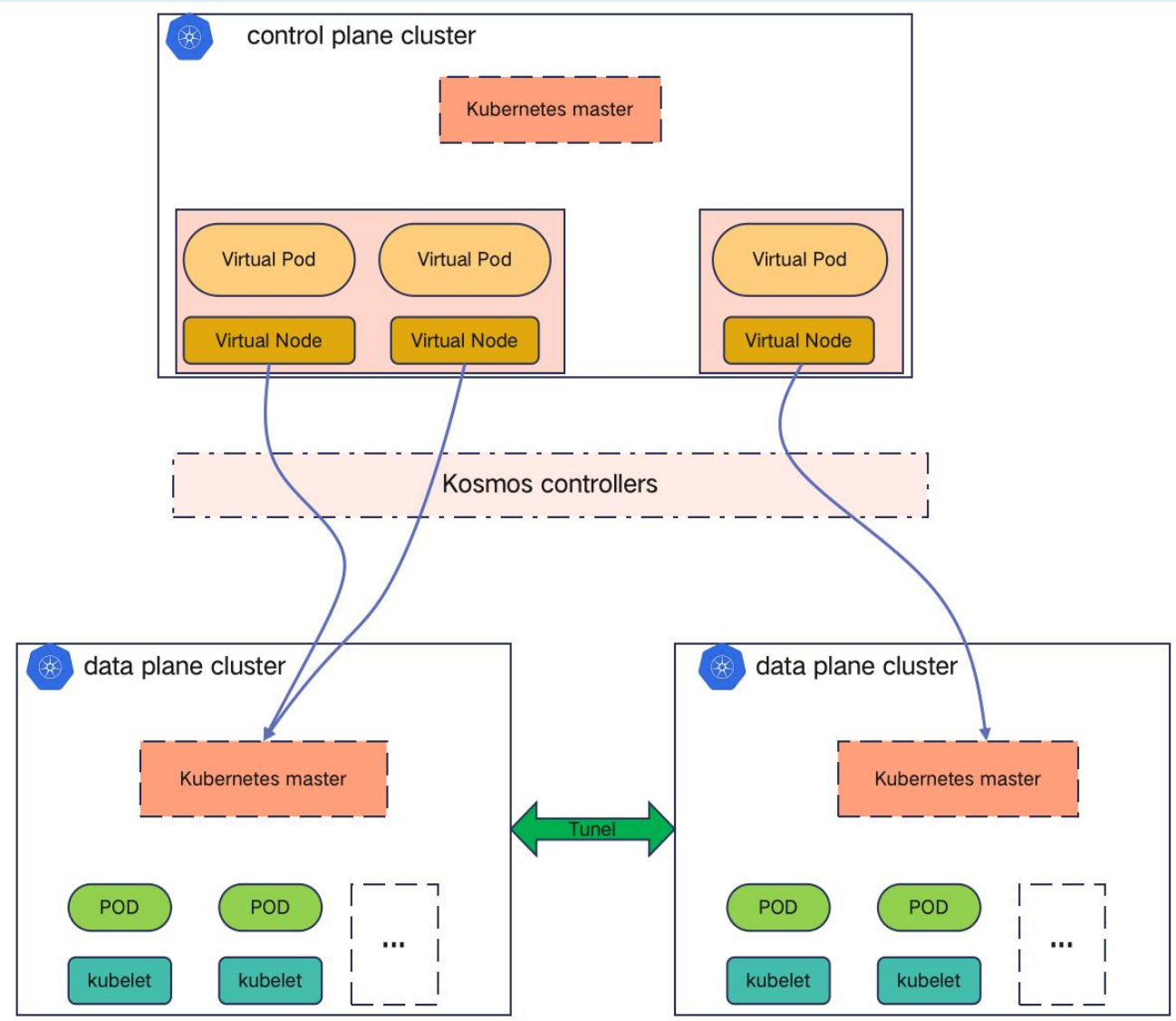
1. 静态调度（static）：指定下发的clasterName、labelSelector等
2. 动态调度（dynamic）：根据集群资源情况调度、可以指定具体的调度因子（scheduleFactor），比如：cpu、gpu、memory等
3. 伴随调度（follow）：跟随某个已经存在的资源下发，比如：job跟随cr

多集群落地三 自研有状态服务的多集群编排调度

- 产品组提出了数据库实例不同副本分布到不同集群的需求
- 集群算力的互相使用



多集群落地三 自研有状态服务的多集群编排调度

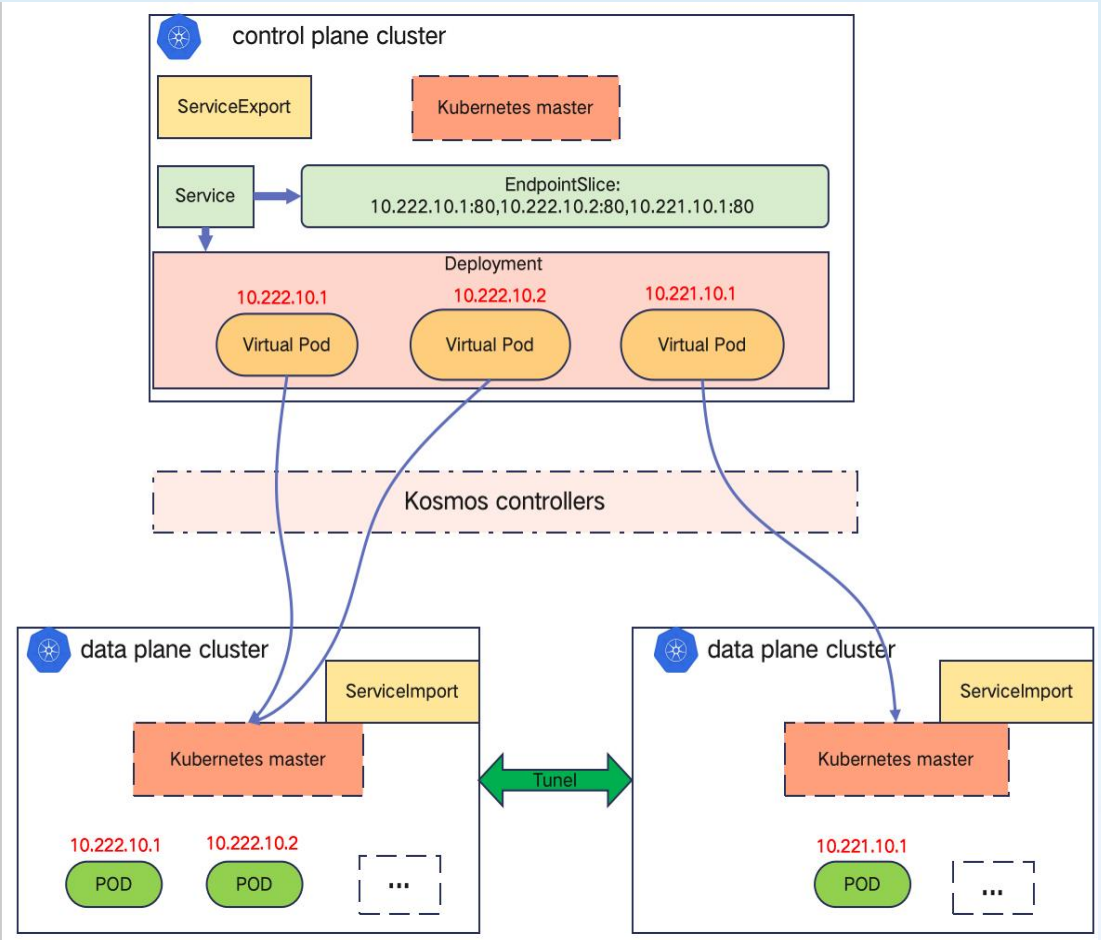


Kosmos跨集群编排：

- 收到VK的启发而设计
- 原生的k8s API
- 多种集群纳管模式（一对一，一对多，多对多）
- 支持有状态应用、operator的跨集群编排

- node-controller：节点资源计算；节点状态维护；节点lease更新
- pod-controller：监听host集群pod创建，调用leaf集群kube-apiserver进行pod创建；维护pod状态；环境变量转换；权限注入
- storagecopy-controller：pv/pvc资源同步和状态管理
- mcs-controller：service资源同步和状态管理

多集群落地三 自研有状态服务的多集群编排调度



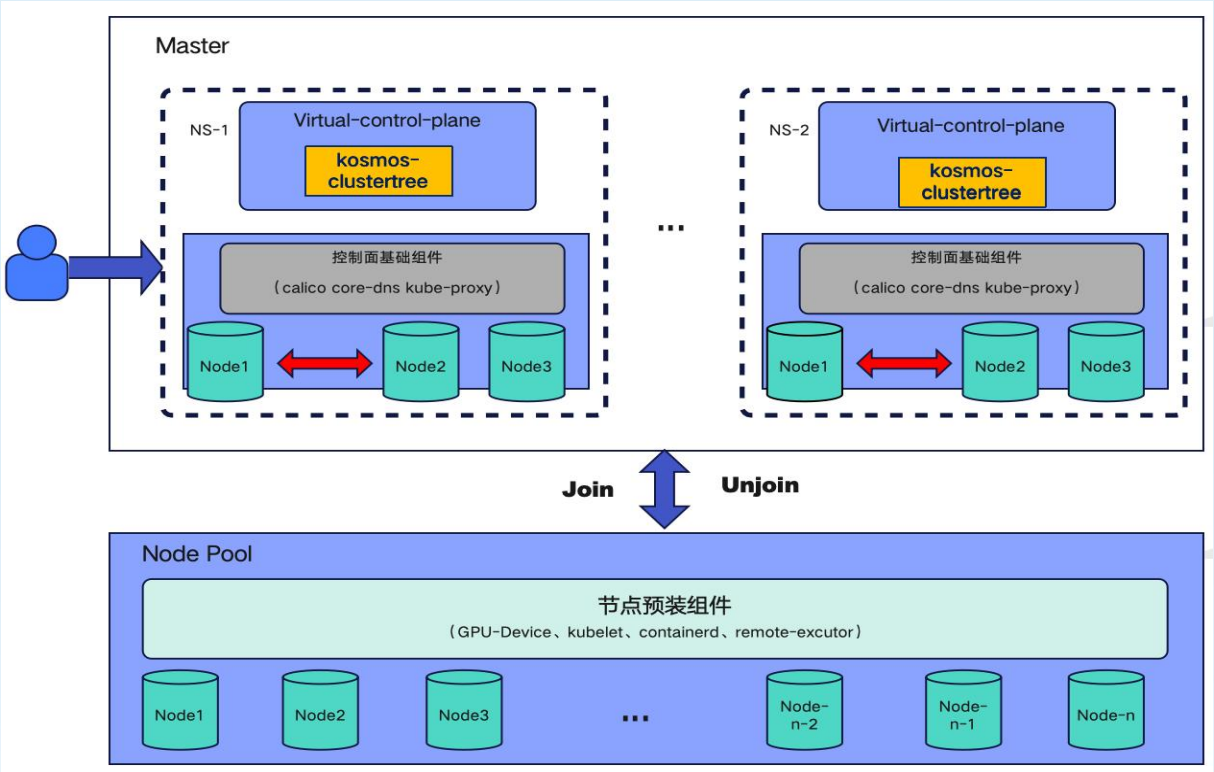
算力纳管业务流程：

- Kosmosctl join cluster，主集群创建 cluster资源
- Kosmos-cluster-tree 根据cluster资源的配置在host集群创建多个node资源，并通过控制器维护node的心跳，保持node的ready状态
- Kosmos-cluster-link 根据cluster的配置创建clusternode和nodeconfig资源，并根据cluster对象在相应的集群创建agent，agent根据clusternode和nodeconfig创建对应的虚拟设备和跨集群的路由

Kosmos跨集群编排业务流程：

- 主集群创建mysql CR
- 主集群mysql-operator创建statefulset，service
- 主集群kube-Controller 监听到sts创建pod 和 pvc，监听到service创建EndpointSlice
- 主集群scheduler将pod调度到上面子集群映射的虚拟节点
- Kosmos控制器将pod和pvc同步到子集群，子集群创建真实pod，csi创建pv完成pvc和pv的绑定，并将pv同步到主集群
- Kosmos不断更新子集群pod和pv状态同步到主集群，并更新node资源使用量
- mcs控制器将主集群service和eps同步到子集群
- 子集群主从pod之间通过域名同步和备份资源

拓展：如何基于Kosmos实现单集群多租户（kube in kube）



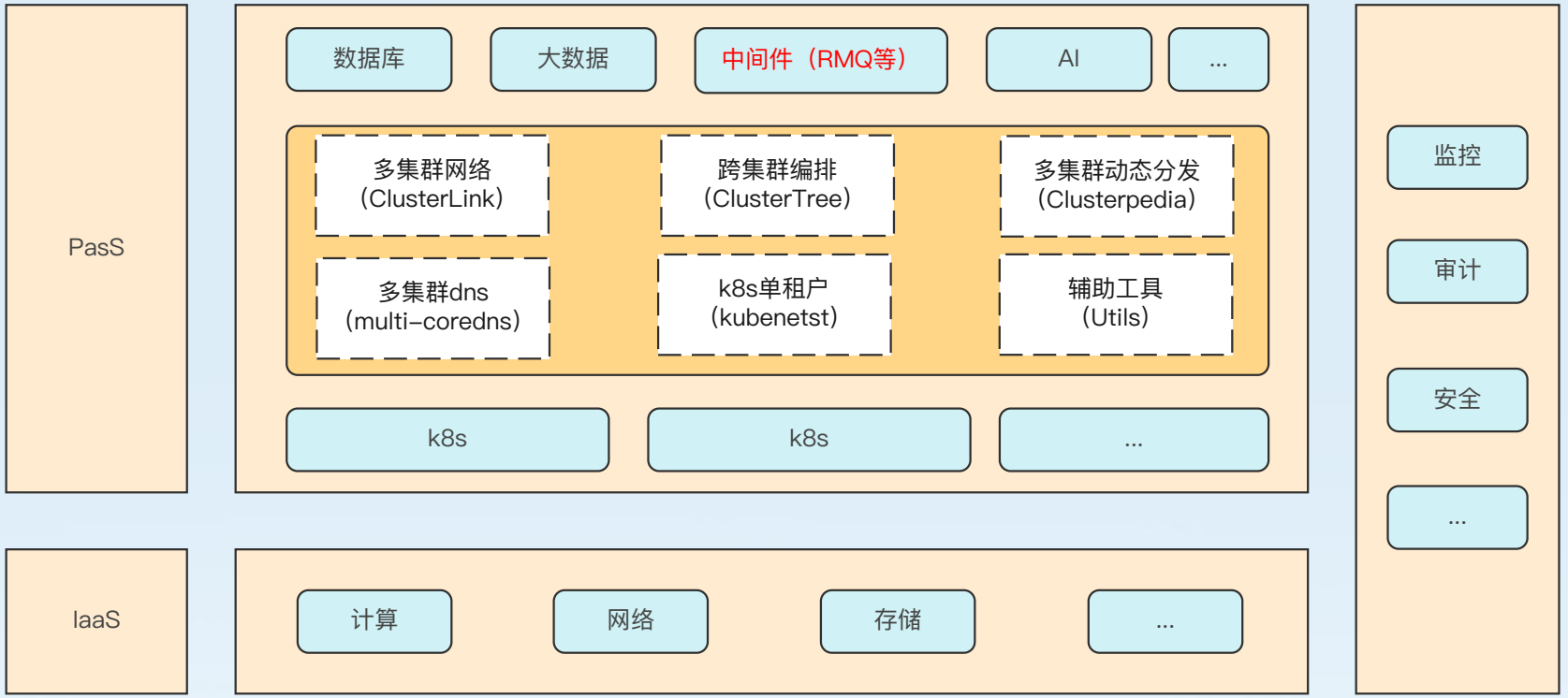

```yaml
apiVersion: kosmos.io/v1alpha1
kind: VirtualCluster
metadata:
  namespace: test
  name: test
spec:
  kubeconfig: xxxxxxxx(虚拟控制面的，base64加密)
  promoteResources:
    nodeInfos:
      - nodeName: xxxxxxx
        address: xxxxxx
        lastVirtualCluster: xxxxxx
      - nodeName: xxxxxxx
        address: xxxxxx
        lastVirtualCluster: xxxxxx
status:
  phase: Completed
```

KubeNest实现架构解析：

- 租户独占虚拟控制面，VirtualCluster配置租户需要纳管的node资源
- 租户虚拟控制面部署Kosmos-clustertree纳管host集群的对应的node
- 租户使用独占的apiserver下发资源

→ 像水和电一样使用算力

总结



■ ClusterLink，作用是打通多个k8s集群之间的网络

- ✓ 跨集群PodIP、ServiceIP互访
- ✓ P2P、Gateway多模式支持
- ✓ 支持全局IP分配
- ✓ 双栈支持

■ ClusterTree，作用是实现应用的跨集群编排

- ✓ 跨集群应用
- ✓ 兼容k8s api，用户零改造
- ✓ 有状态应用
- ✓ k8s-native应用支持

■ 工具，辅助运维

- kosmos-operator
  提供快速部署kosmos能力
- kosmosctl
  命令行工具，提供网络连通性检测、安装部署kosmos、快速纳管集群等功能

后续发展：	已上生产版本：
多集群AI训练	v0.2.0-lts
多集群分布式推理	v0.3.0

总结

基于POD调度的多集群编排和调度：

- 代码库：https://github.com/kosmos-io/kosmos/tree/main/cmd/clustertree
- 介绍：https://mp.weixin.qq.com/s/dVzINCZyv3SicX0Ig0LH5w

基于隧道的多集群网络通信：

- 代码库：https://github.com/kosmos-io/kosmos/tree/main/cmd/clusterlink
- 介绍：https://mp.weixin.qq.com/s/0L3ZRrMWBJe0gtsfJAubwA

集群多租户实现：

- 代码库：https://github.com/kosmos-io/kosmos/tree/main/cmd/kubenest/
- 介绍：https://kosmos-io.github.io/website/v0.2.0/proposals/k8s-in-k8s

集群网络连通性校验工具：

- 代码库：https://github.com/kosmos-io/netdoctor
- 介绍：https://mp.weixin.qq.com/s/yKfhjjvAh7EHUxv679Ah4g

多集群Core-DNS：

- 代码库：https://github.com/kosmos-io/coredns

多集群资源分发和汇聚和调度、多集群watch：

- 代码库：https://github.com/kosmos-io/clusterpedia

公众号：畅聊云原生
（可私聊）

Thanks.