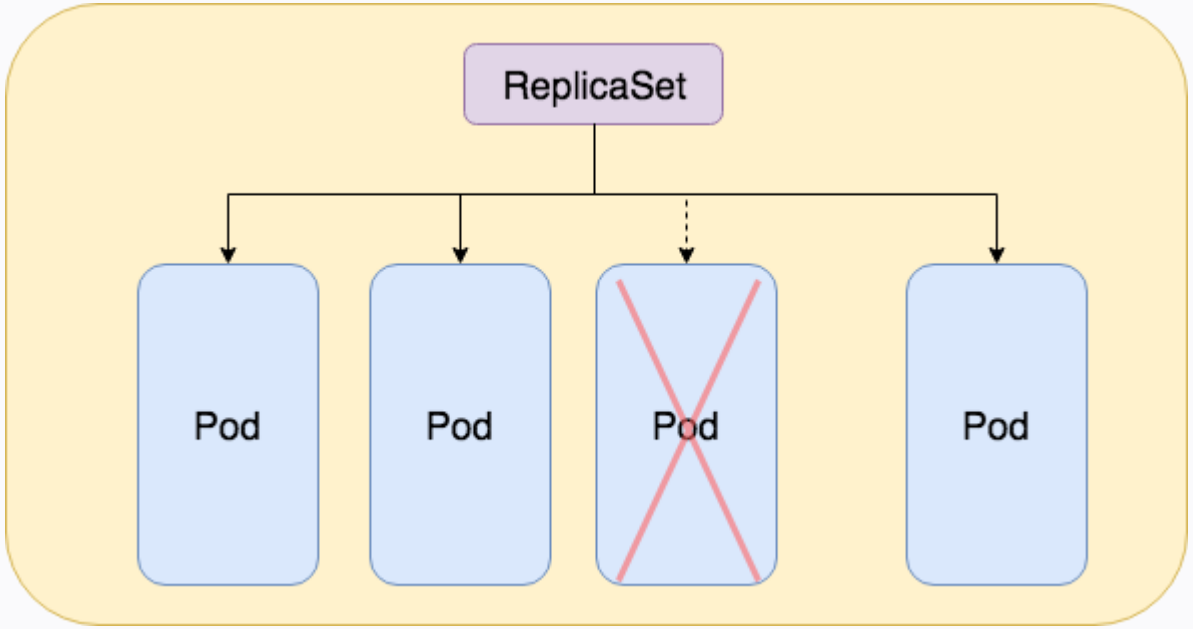
Kubernetes核心组件详解

目录

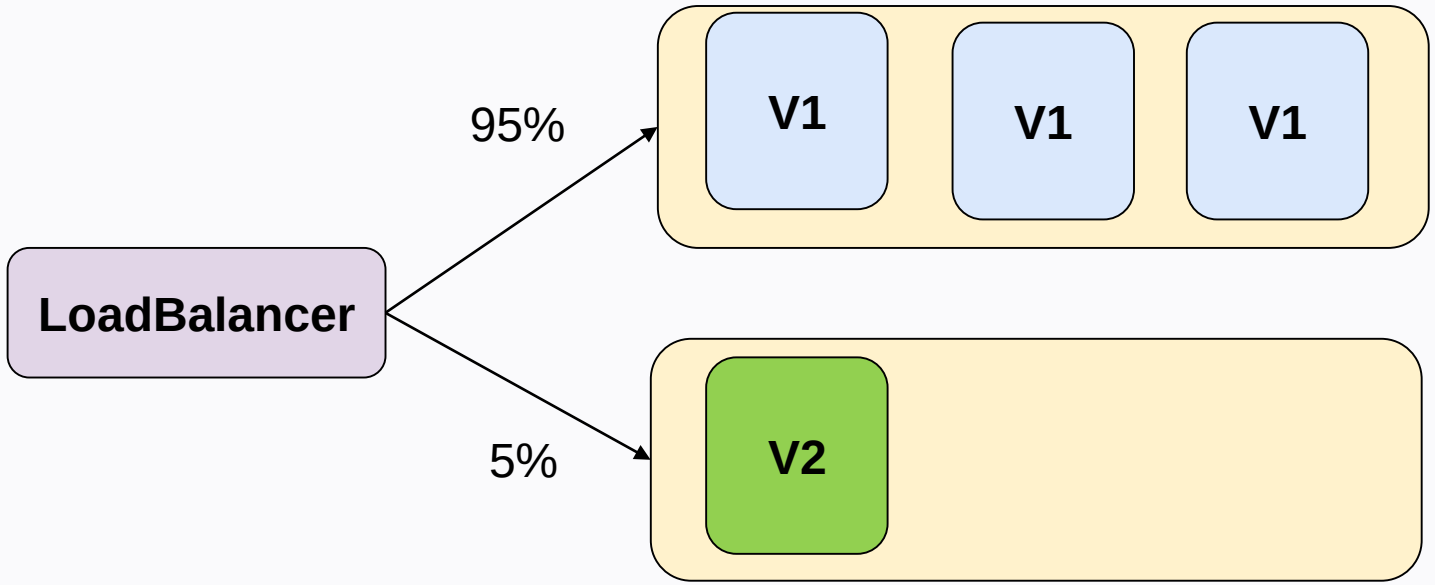
Kubernetes核心概念 – Deployments

- ReplicaSet
  - 确保Pod副本的数量
  - 支持自动扩容和收缩（scale)
  - 不能支持服务的滚动部署



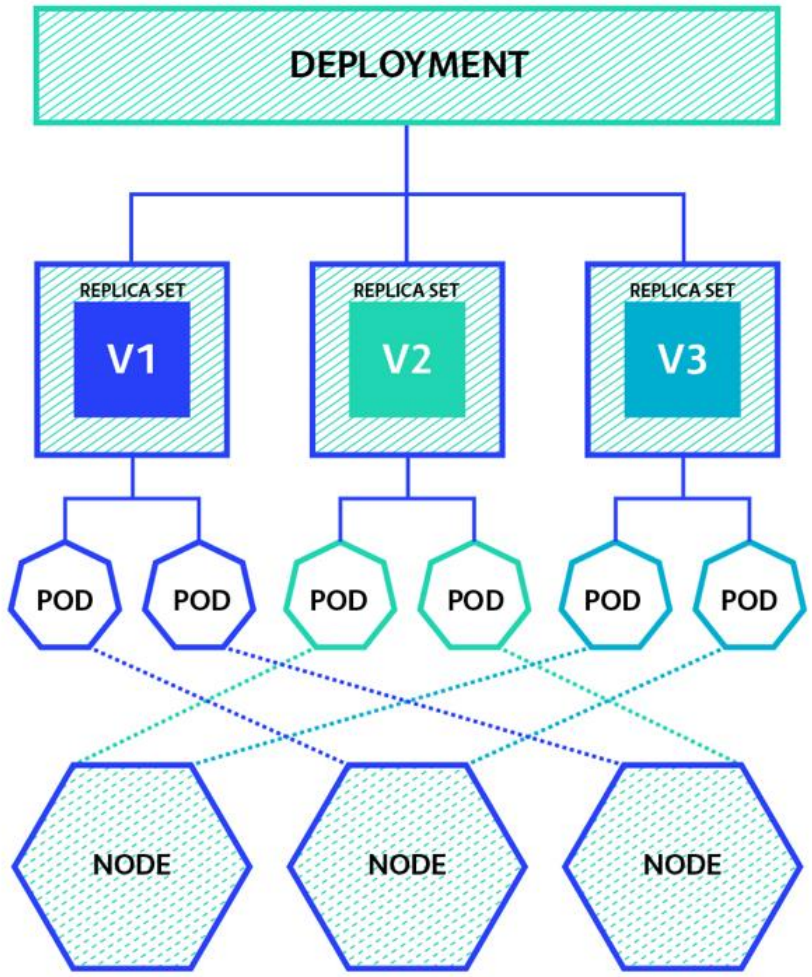
Kubernetes核心概念 – Deployments

- 红绿部署
- 滚动部署
- 灰度部署(canary)



Kubernetes核心概念 – Deployments

- Deployment
  - 提供了一种声明式的方法来通过ReplicationSet管理Pods
  - 支持Pod的RollingUpdate，并自动管理其背后的ReplicationSet
  - 支持roll back到之前的revision



Kubernetes核心概念 – Deployments Create

- kubectl apply – f nginx-deployment.yml --record
- kubectl rollout status deployment/nginx-deployment-demo
- kubectl get deployment
- Kubectl get rs
- kubectl describe deployment nginx-deployment

- kubectl get pod --show-labels

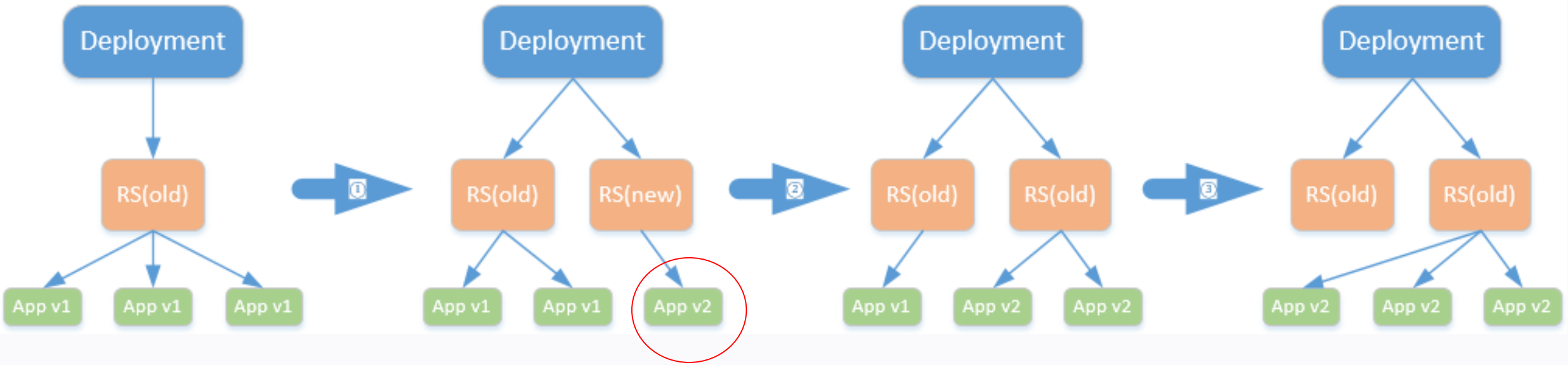
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-demo
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
     matchLabels:
        app: nginx
  strategy:
     type: RollingUpdate
     rollingUpdate:
        maxSurge: 1
        maxUnavailable: 0
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Kubernetes核心概念 – Deployments Upgrade

- kubectl apply – f nginx-deployment-v2.yml
-
- kubectl rollout **history** deployment nginx-deployment-demo
- kubectl rollout **history** deployment nginx-deployment-demo --revision=3
- kubectl rollout undo deployment nginx-deployment – demo --to-revision=1
- kubectl scale deployment nginx-deployment-demo --replicas=10

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment-demo
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.8
        ports:
        - containerPort: 80
~
```

Kubernetes核心概念 – Deployments Upgrade

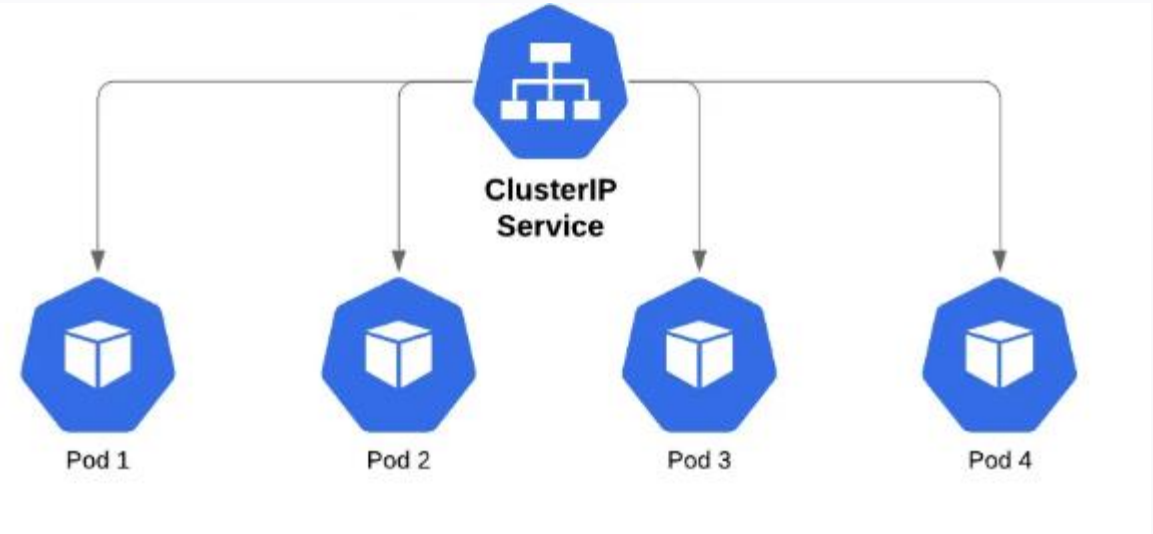


目录

Kubernetes核心概念 – Service

- Pod
  - Pod 的生命周期不是永久的
  - Pod IP随时发生变化

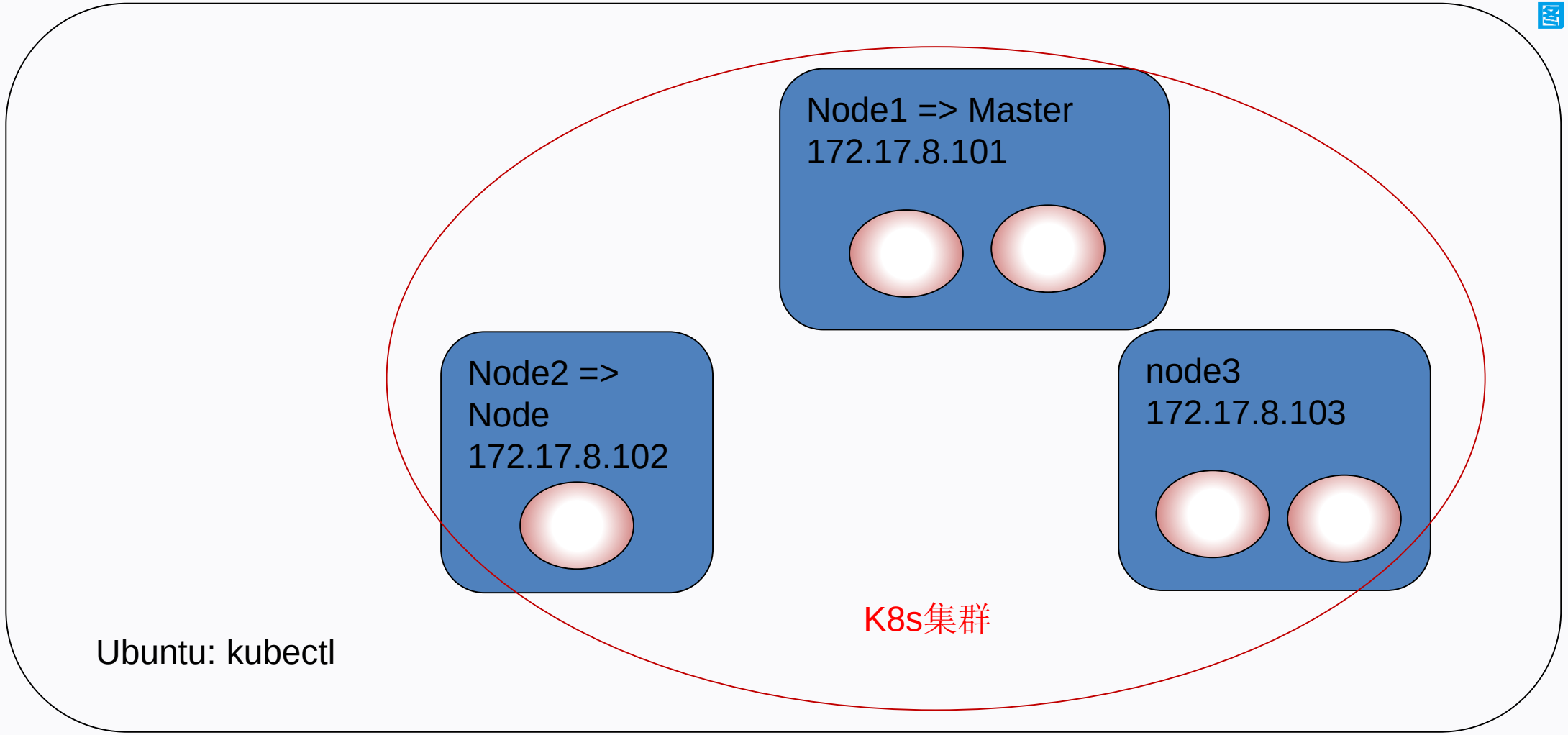
- Service
  - Pod之上的一个抽象层
  - Service 会被分配 一个 VIP(ClusterIP)，并在它销毁之前保持该 VIP 地址保持不变
  - 通过对它的访问，以代理的方式负载到对应的 Pod 上



Kubernetes核心概念 – Service Demo

- kubectl apply – f nginx-service.yml
- port：虚拟 ip 要绑定的 port，每个 service 会创建出来一个虚拟 ip，通过访问 vip:port 就能获取服务的内容
- targetPort：pod 中暴露出来的 port，这是运行的容器中具体暴露出来的端口
- 默认的 service 类型是 ClusterIP
  - 只能从集群内部访问这个 IP，不能直接从集群外部访问服务

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
```



Kubernetes核心概念 – Service Demo

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

没有 selector 的服务

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 9376
    - name: https
      protocol: TCP
      port: 443
      targetPort: 9377
```

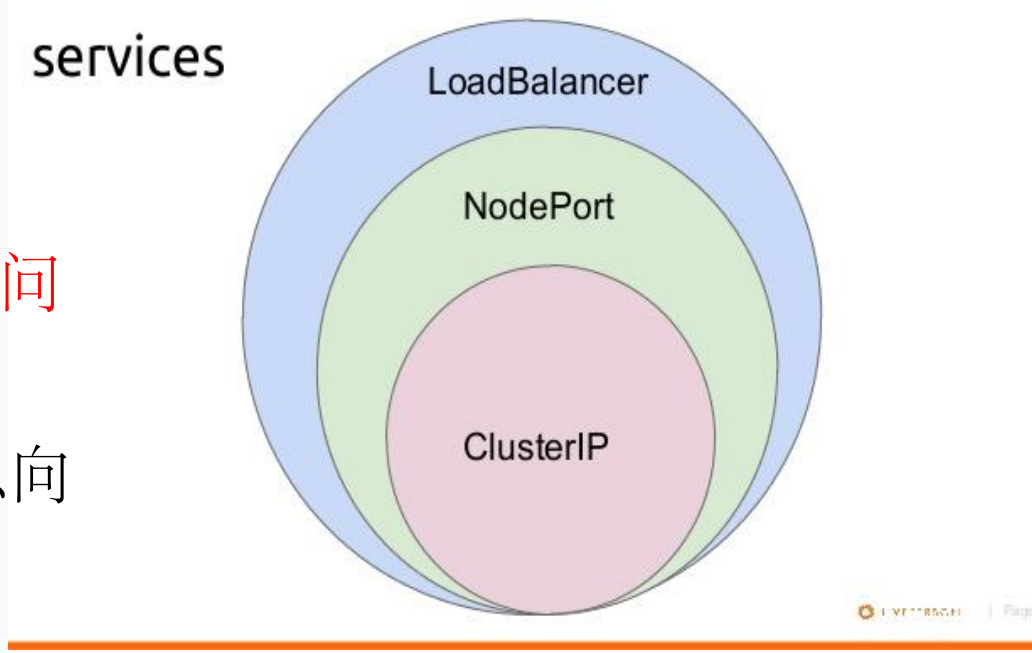
多端口服务

Kubernetes核心概念 – Service Discovery

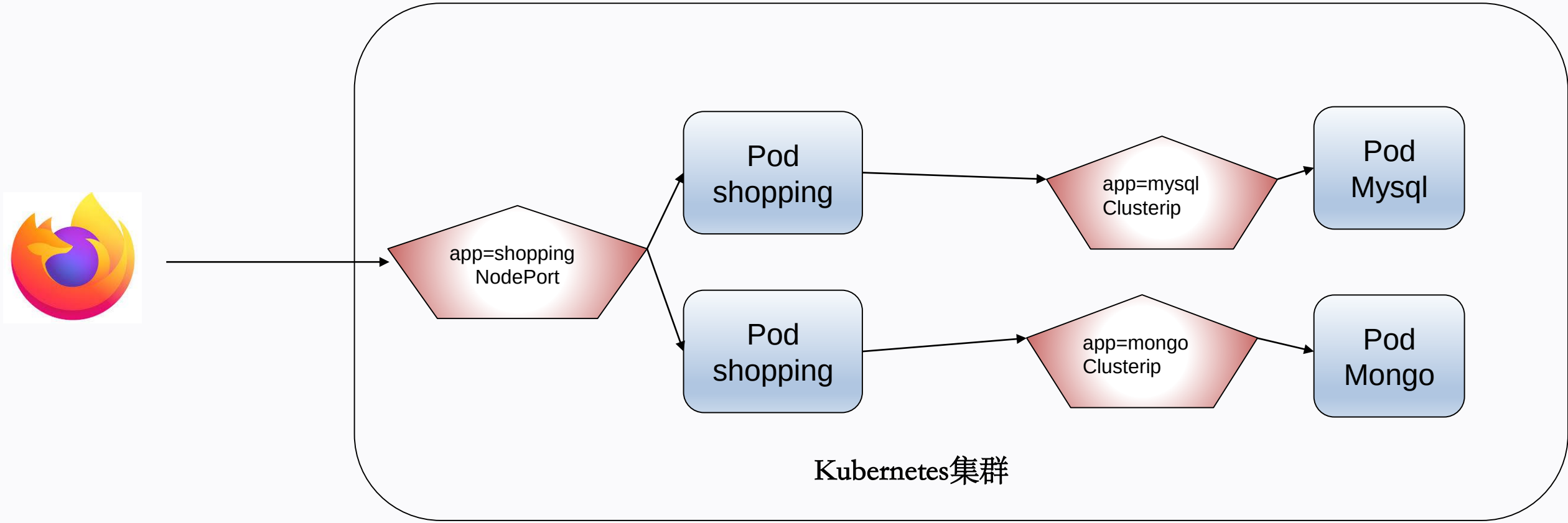
- 环境变量 - POD
  - Pod 运行在 Node 上，kubelet 会为每个活跃的 Service 添加一组环境变量。{SVCNAME}_SERVICE_HOST 和 {SVCNAME}_SERVICE_PORT 变量，这里 Service 的名称需大写，横线被转换成下划线。 Nignx-service
  - 服务创建后的新建的Pod里才有
- DNS
  - 支持群集的DNS服务器（例如CoreDNS）监视 Kubernetes API 中的新服务，并为每个服务创建一组 DNS 记录
  - 例如：在 Kubernetes 命名空间 “my-ns” 中有一个名为 “my-service” 的服务， 则为"my-service.my-ns" 创建 DNS 记录

Kubernetes核心概念 – 发布服务：Service Type

- Service Type 的取值以及行为如下：
  - ClusterIP：**通过集群的内部 IP 暴露服务，选择该值，服务只能够在集群内部可以访问**
  - NodePort：通过每个 Node 上的 IP 和静态端口（NodePort）暴露服务。通过请求 <NodeIP>:<NodePort>，可以从集群的外部访问一个 NodePort 服务。
  - LoadBalancer：使用云提供商的负载局衡器，可以向外部暴露服务。外部的负载均衡器可以路由到 NodePort 服务和 ClusterIP 服务。
  - ExternalName：通过返回 CNAME 和它的值，可以将服务映射到 externalName 字段的内容. 没有任何类型代理被创建。

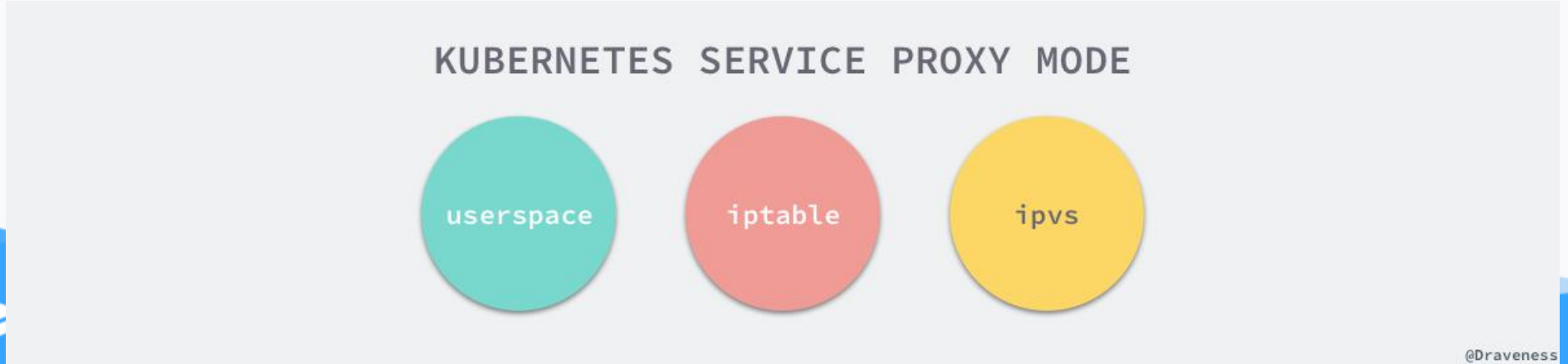


Kubernetes核心概念 – 发布服务：Service Type

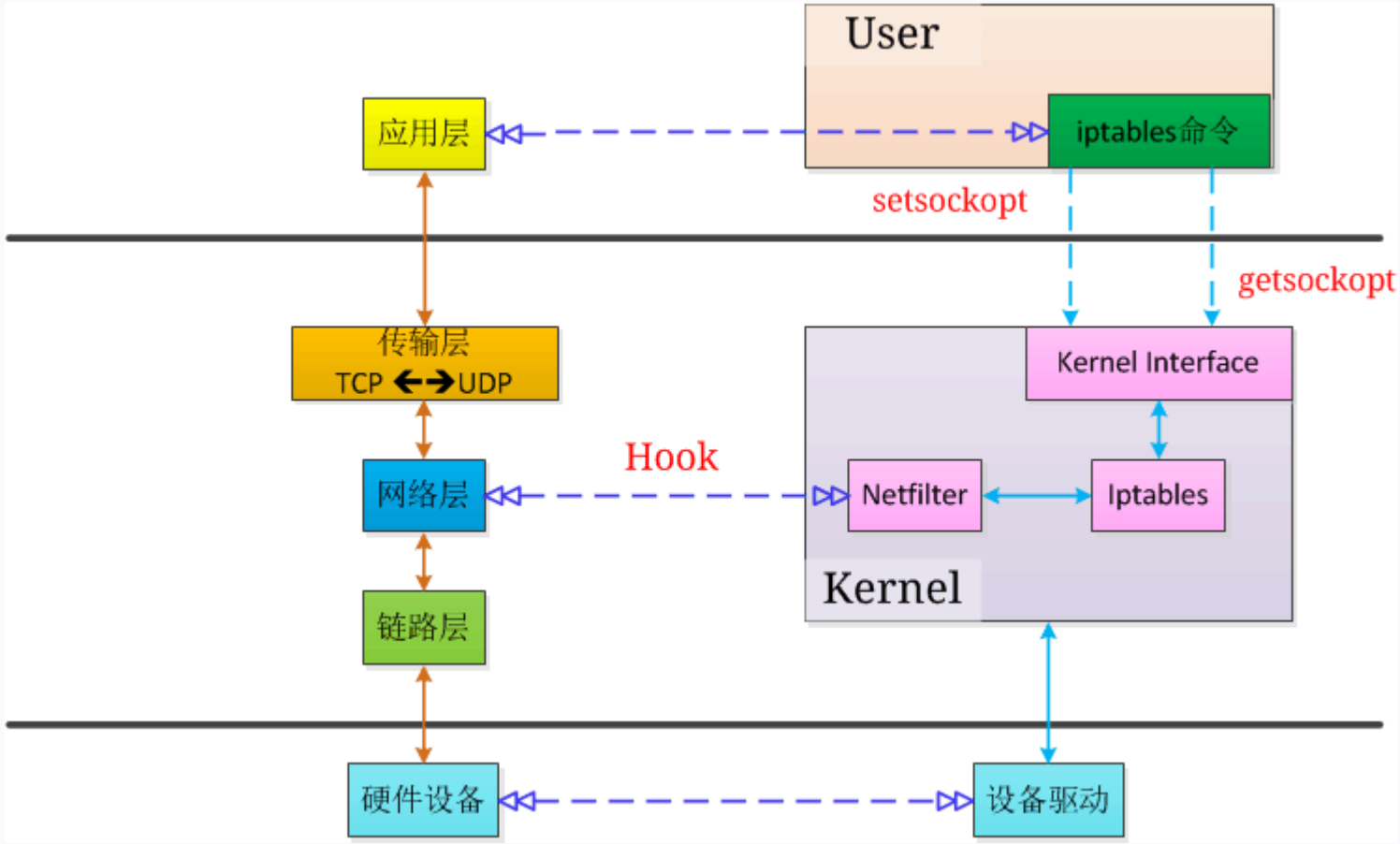


Kubernetes Service ClusterIP 网络实现

- Service ClusterIP (VIP)并不是真正的IP
- 关键角色：**kube-proxy**
  - 每一个节点都运行着一个 kube-proxy 进程
  - 负责监听 Kubernetes 主节点中 Service 的增加和删除事件并修改运行**代理的配置**
  - 为节点内的客户端提供**流量的转发和负载均衡**等功能

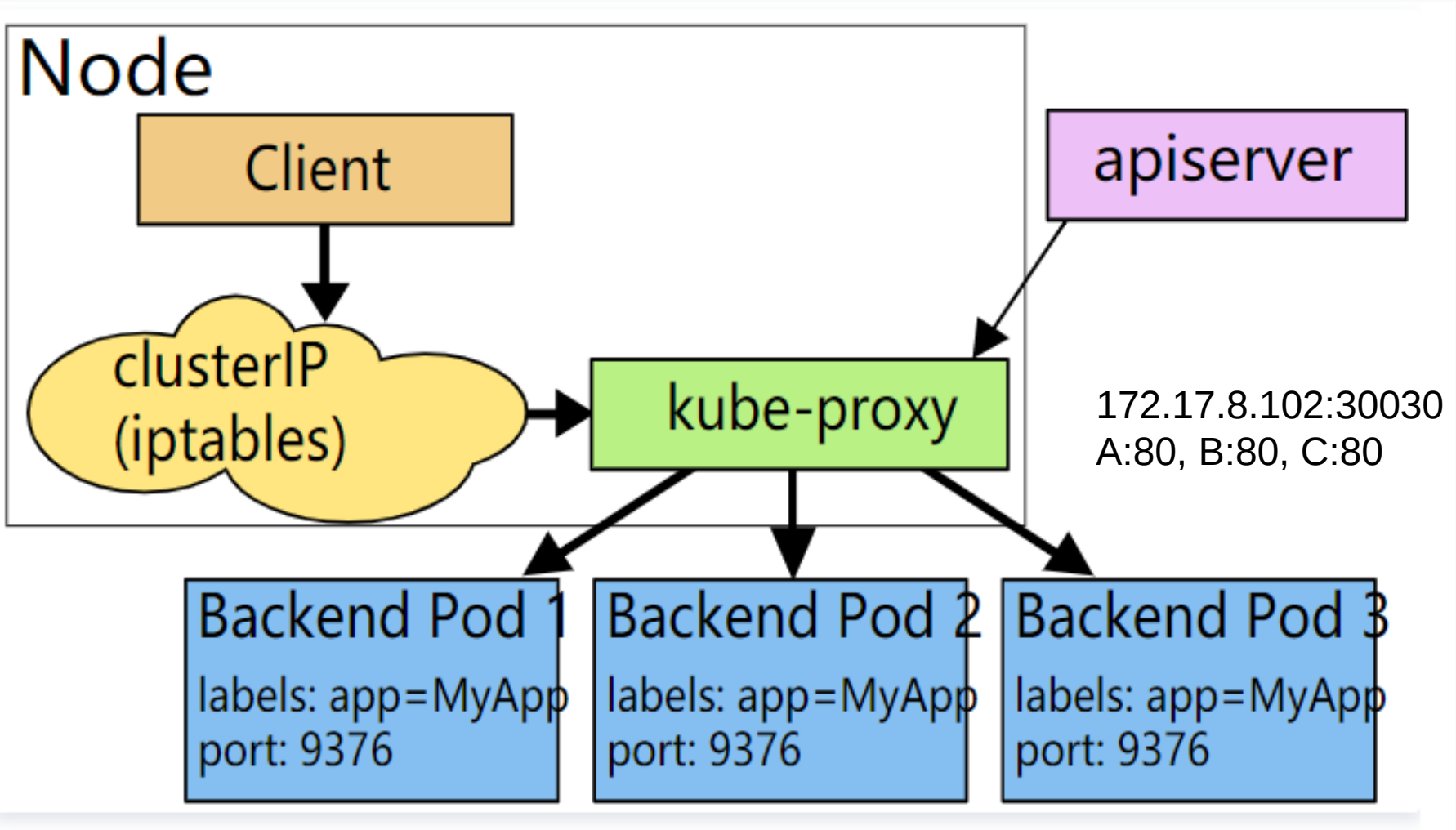


Iptable/netfilter



userspace代理模式

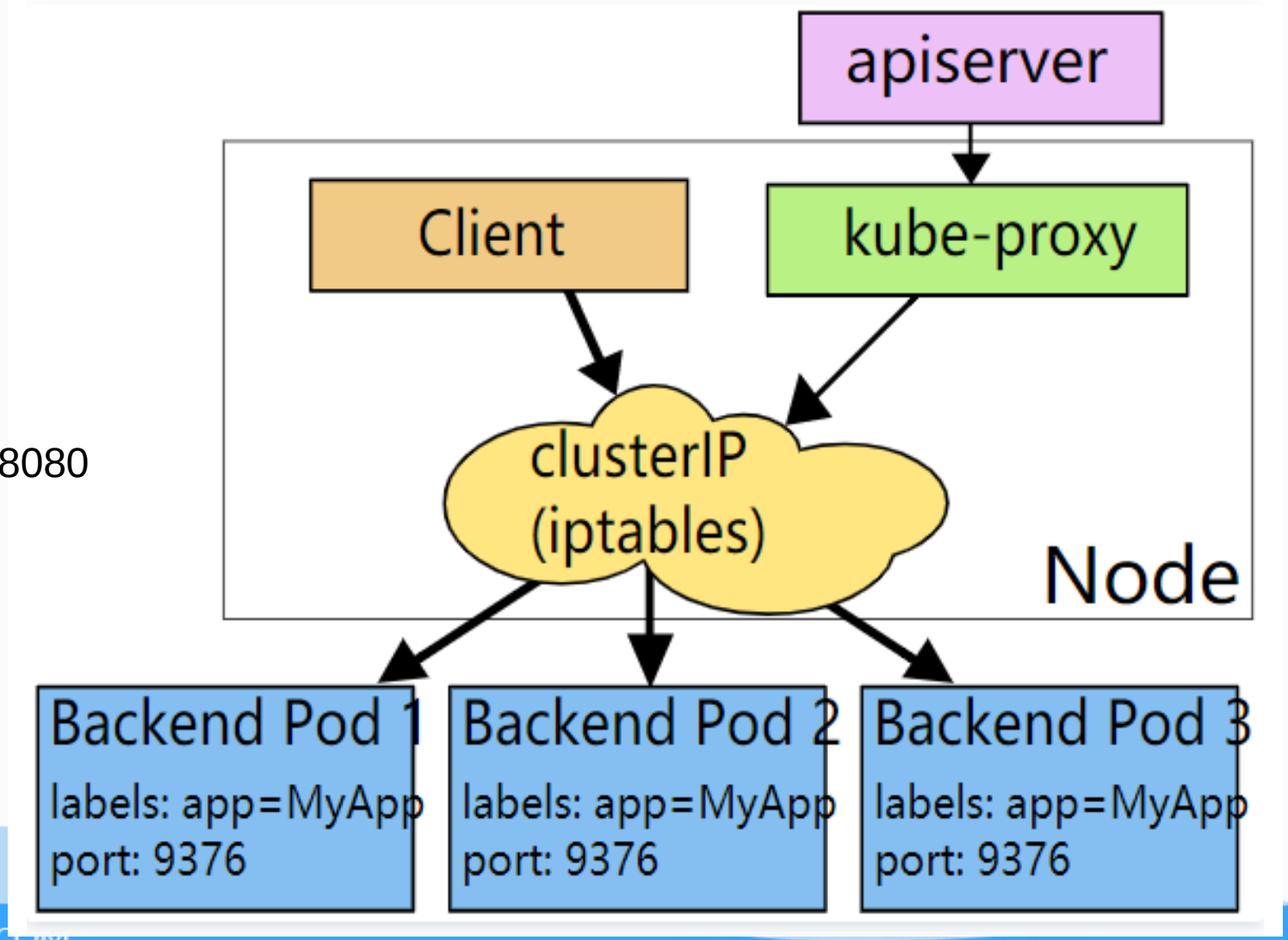
10.254.164.247:8080
-> 172.17.8.102:30030



10.254.164.247:8080
->A:80, B:80, C:80

172.17.8.102:30030 ->
A:80, B:80, C:80

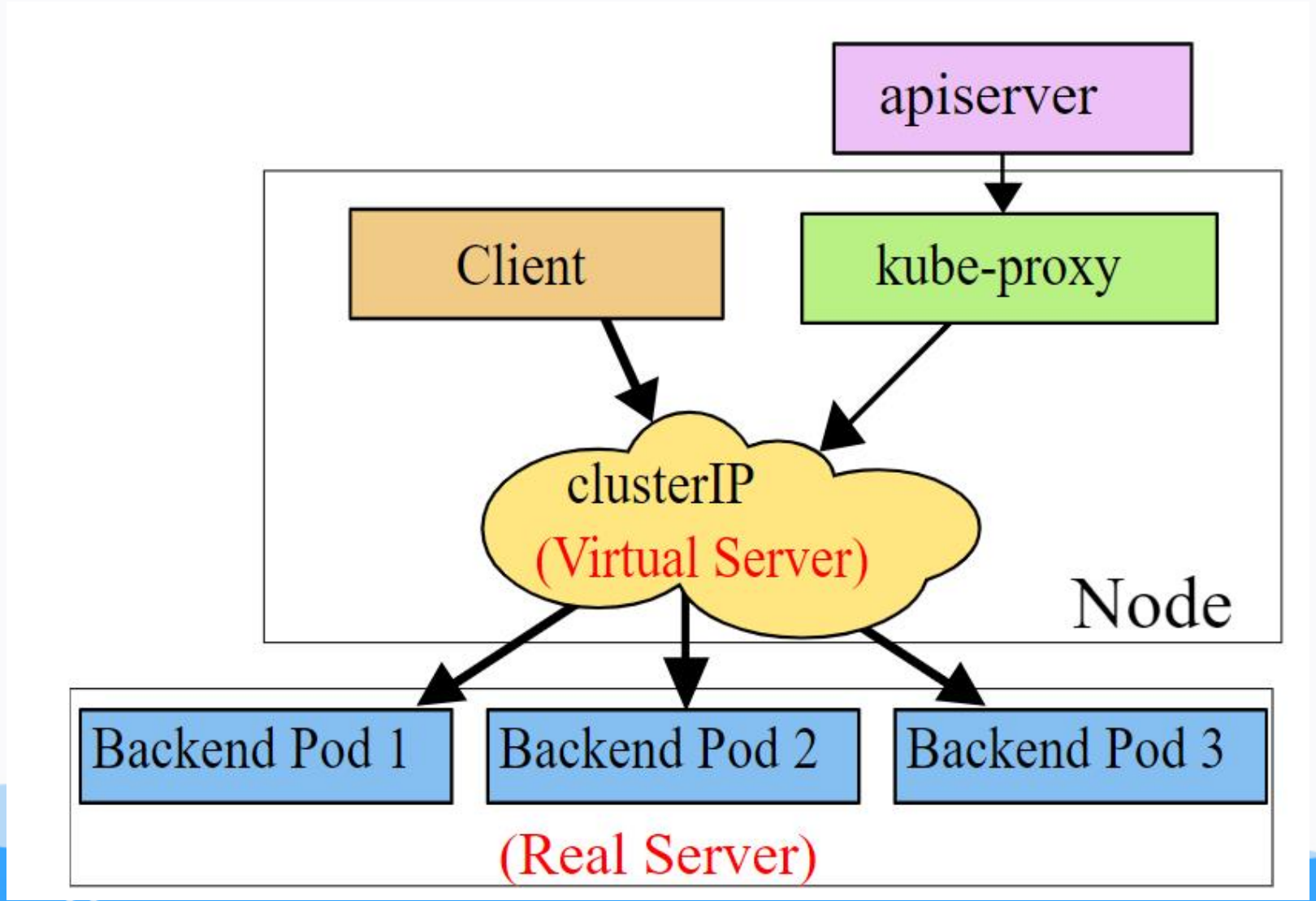
iptables代理模式



10.254.164.247:8080
->A:80, B:80, C:80

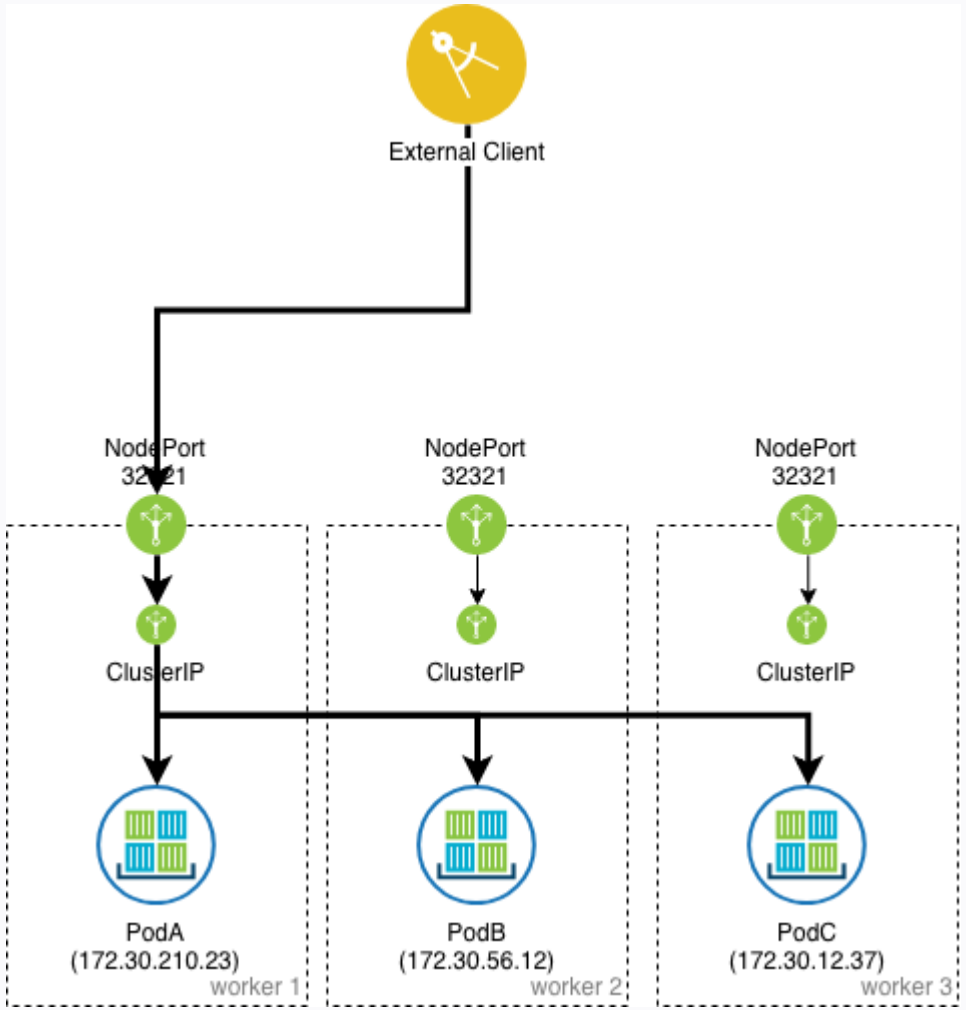
10.254.164.247:8080
-> A:80

IPVS代理模式



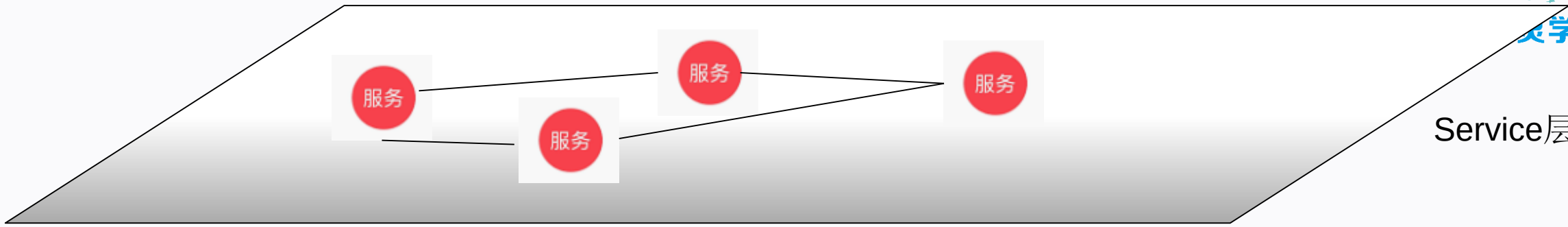
Kubernetes Service NodePort

- NodePort示例与原理
  - 端口转发
- NodePort不足
  - 每个端口只能提供一个服务
  - 只能使用端口 30000–32767
  - 如果节点 / VM IP 地址更改，则需要处理

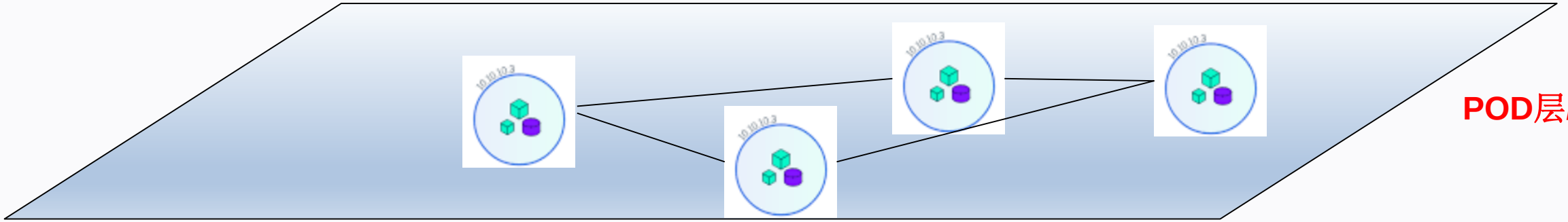


目录

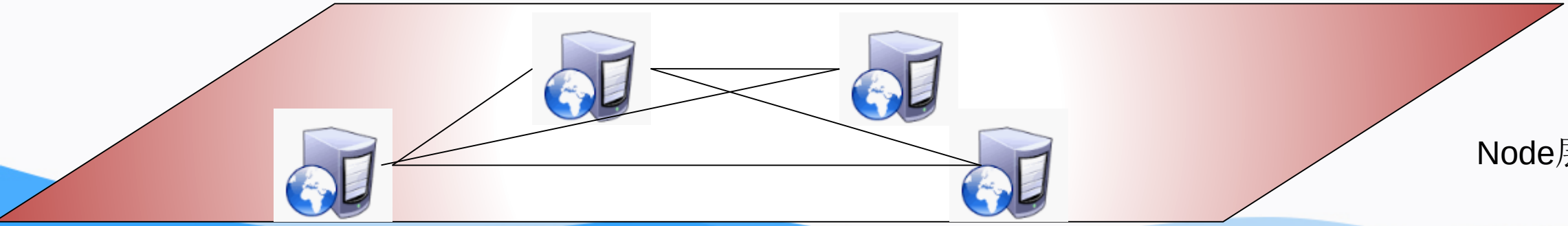
Kubernetes网络



Service层/Clust



POD层/POD IP



Node层/Node IP

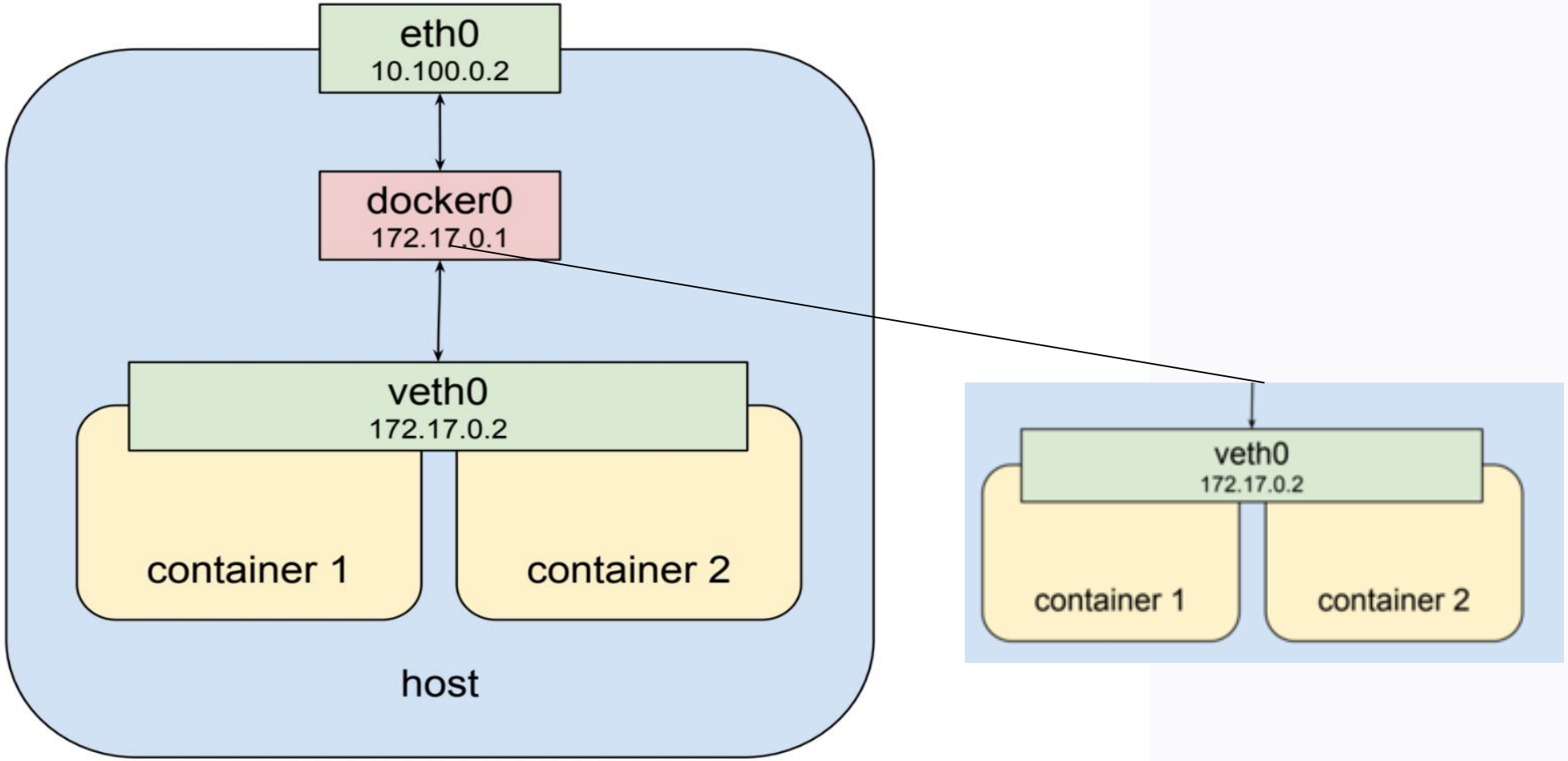
Kubernetes Pod 网络模型

- Every Pod gets its own IP address.
- Pod on a node can communicate with all pods on all nodes **without NAT**
- Agents on a node (e.g. system daemons, kubelet) can communicate with all pods on that node

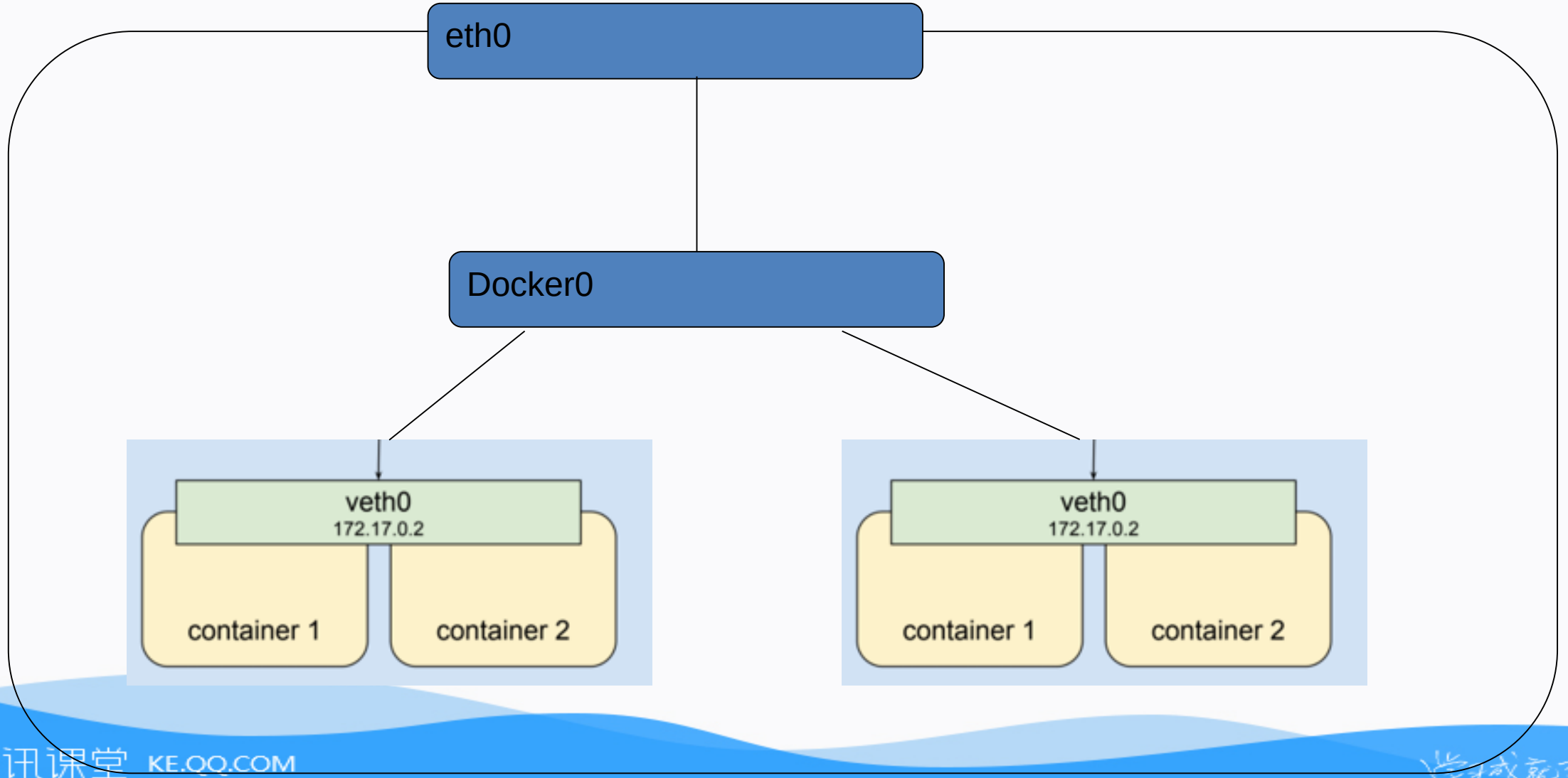
Kubernetes Pod 网络

- Pod内Container的通信
  - 共享网络空间
- POD和POD的通信
  - 同一个NODE
  - 跨不同的NODE
- Pod和Service

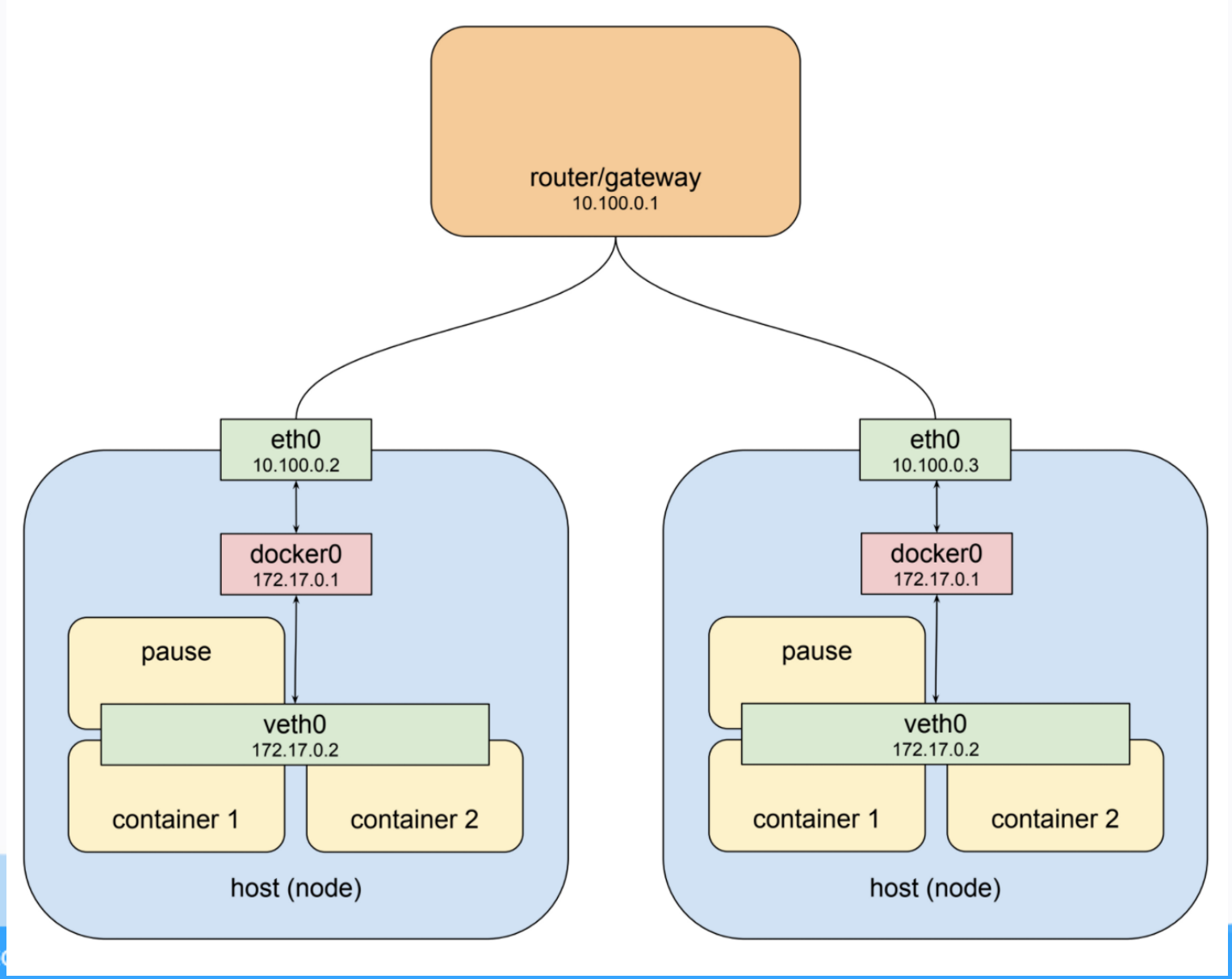
Kubernetes Pod 网络



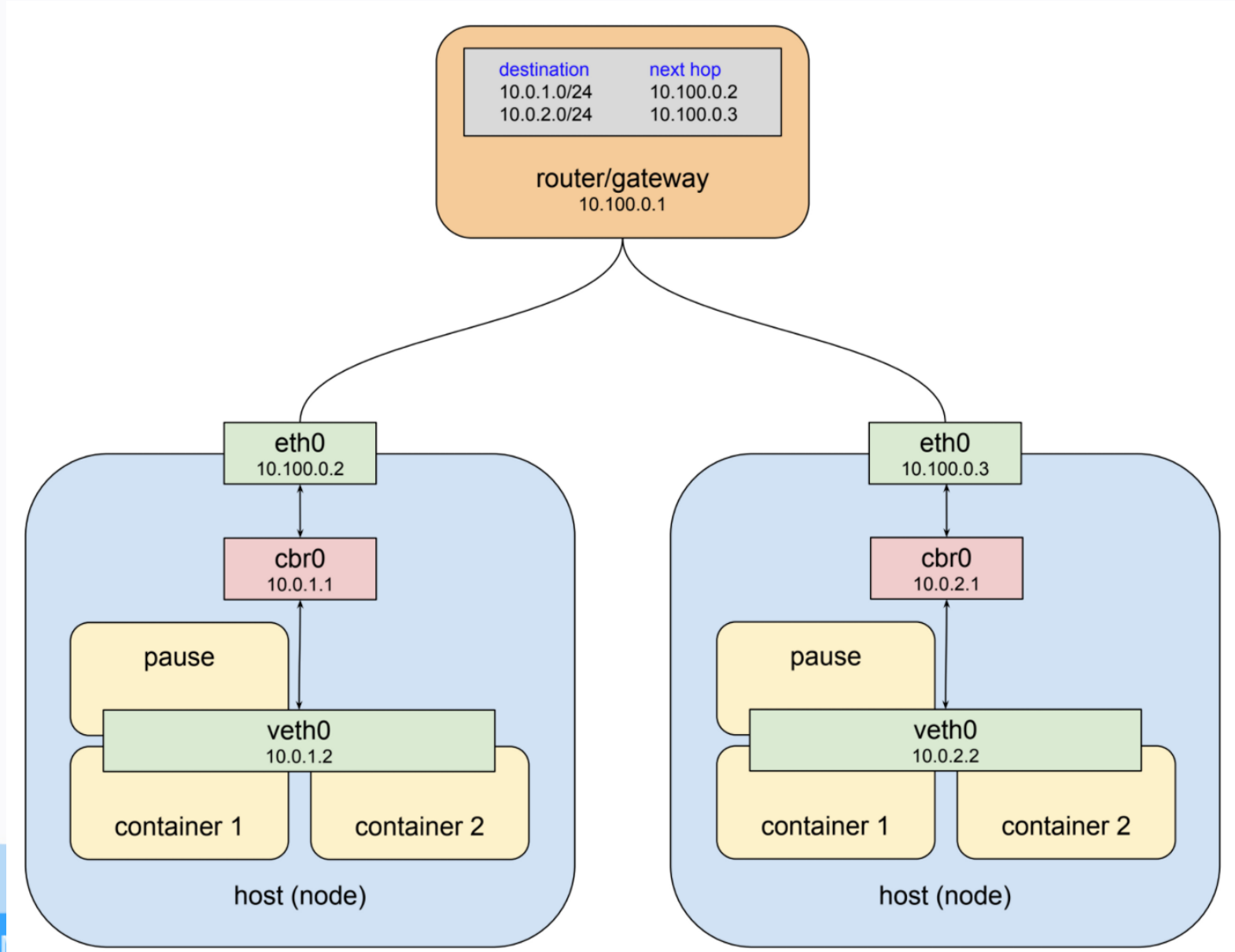
Kubernetes Pod 网络



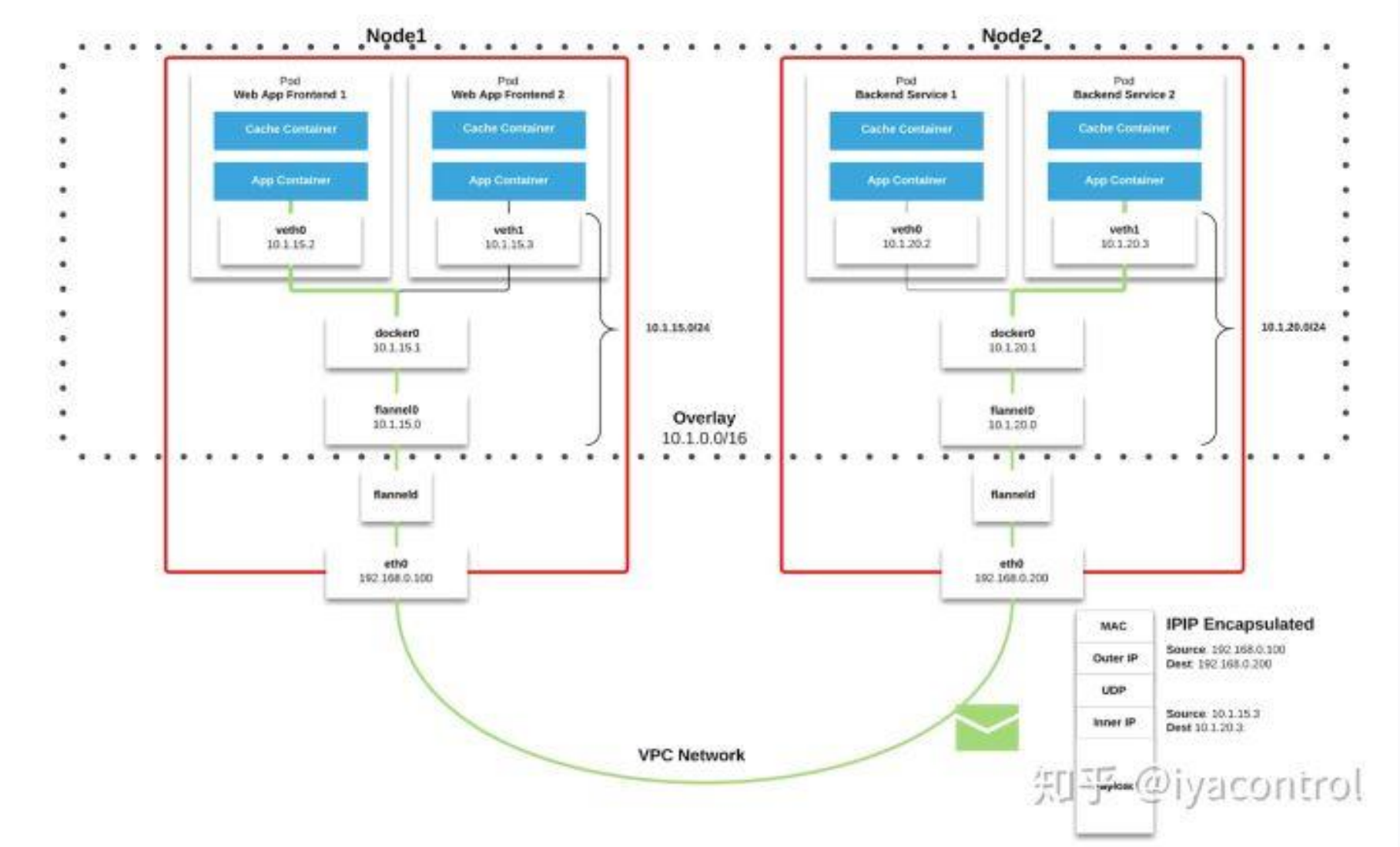
Kubernetes Pod 网络



Kubernetes Pod 网络



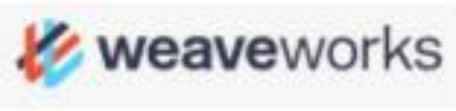
Kubernetes Pod - Overlay容器网络

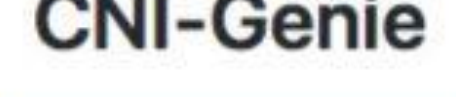


Kubernetes Pod 网络

CNI标准

- Bridge
- PTP
- IPVLAN
- MACVLAN
- VLAN
- PORTMAP





谢谢！