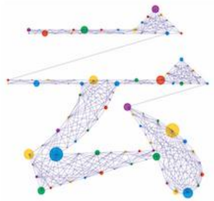
EasyStack
open cloud computing



第四届
全国高校
云计算
应用创新大赛

全国高校云计算应用创新大赛
官方培训班在线课程

Kubernetes

胡正川

EasyStack研发工程师

目录

Kubernetes架构

Kubernetes

Google 开源的容器集群管理系统，内部的borg的开源实现

CNCF项目

特点

具有完备的集群管理能力

多层次的安全防护和准入机制

多租户应用支撑能力

透明的服务注册和发现机制

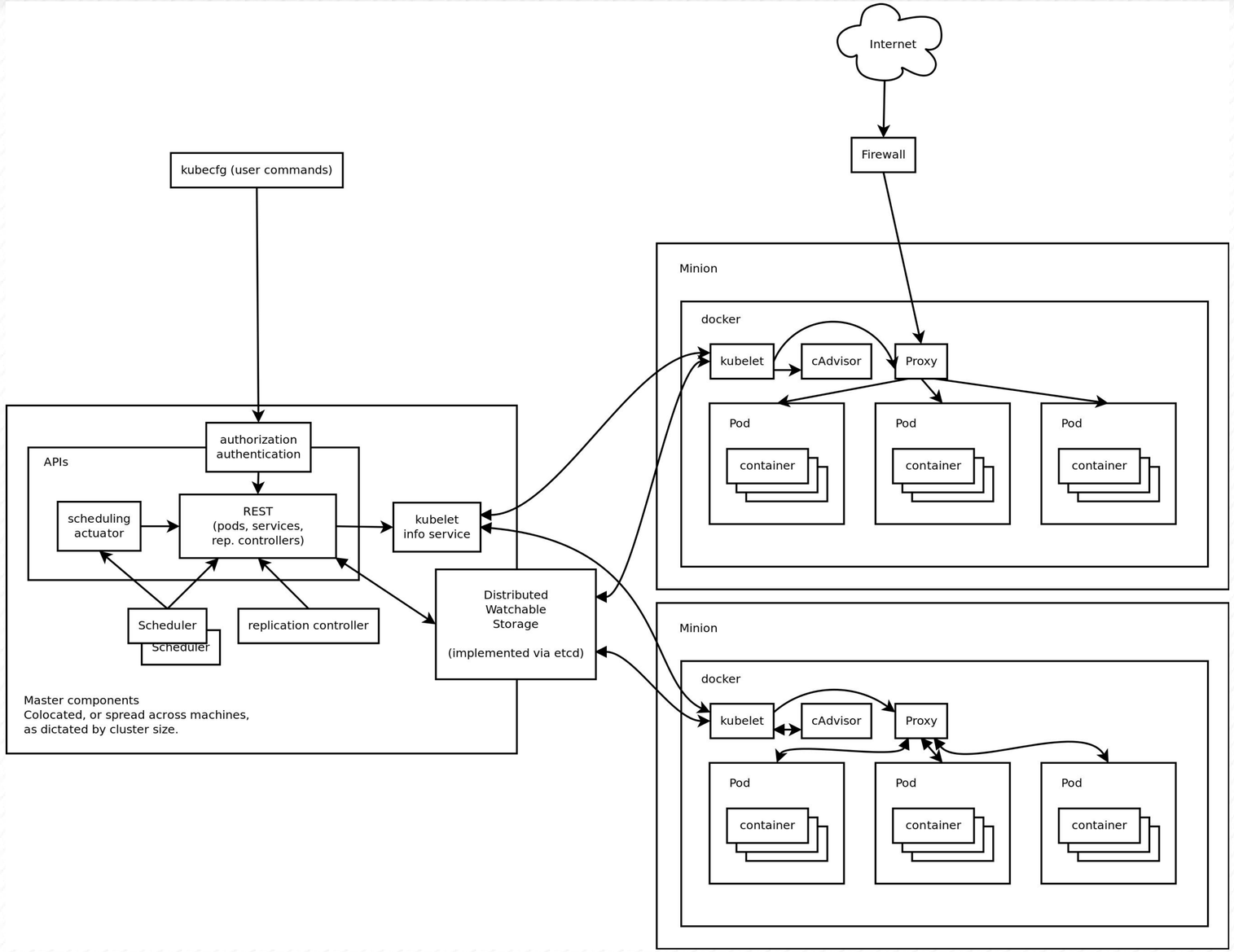
内建智能负载均衡器

强大的故障发现和自我修复能力

服务滚动升级和在线扩容能力

可扩展的资源自动调度机制以及多粒度的资源配额管理能力

Kubernetes架构



Kubernetes架构

Kubernetes组件

Master node

1. Kube/ apiserver
2. kube/ controller
3. kube/ schduler

Work node

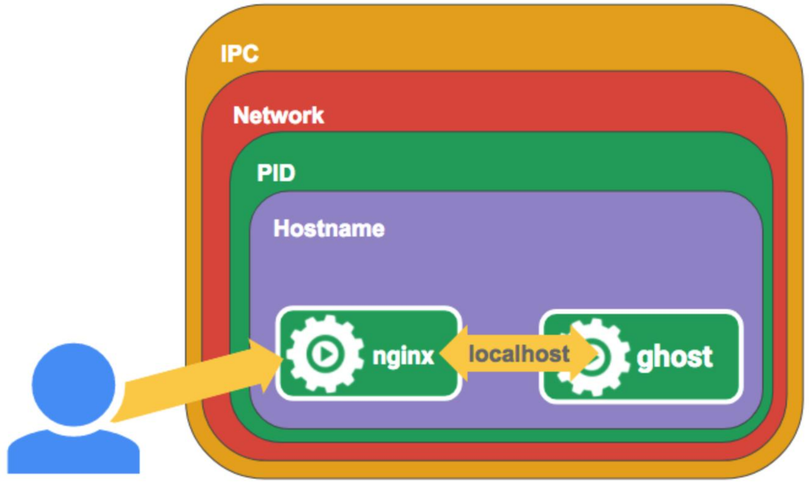
1. kubelet
2. kube/ proxy
3. Docker

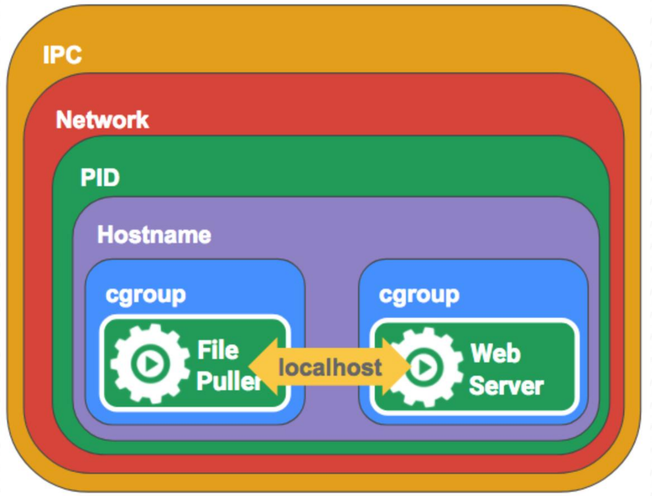
Support component

1. Etcd
2. Podnetwork(flannel...)

Kubernetes基础-app

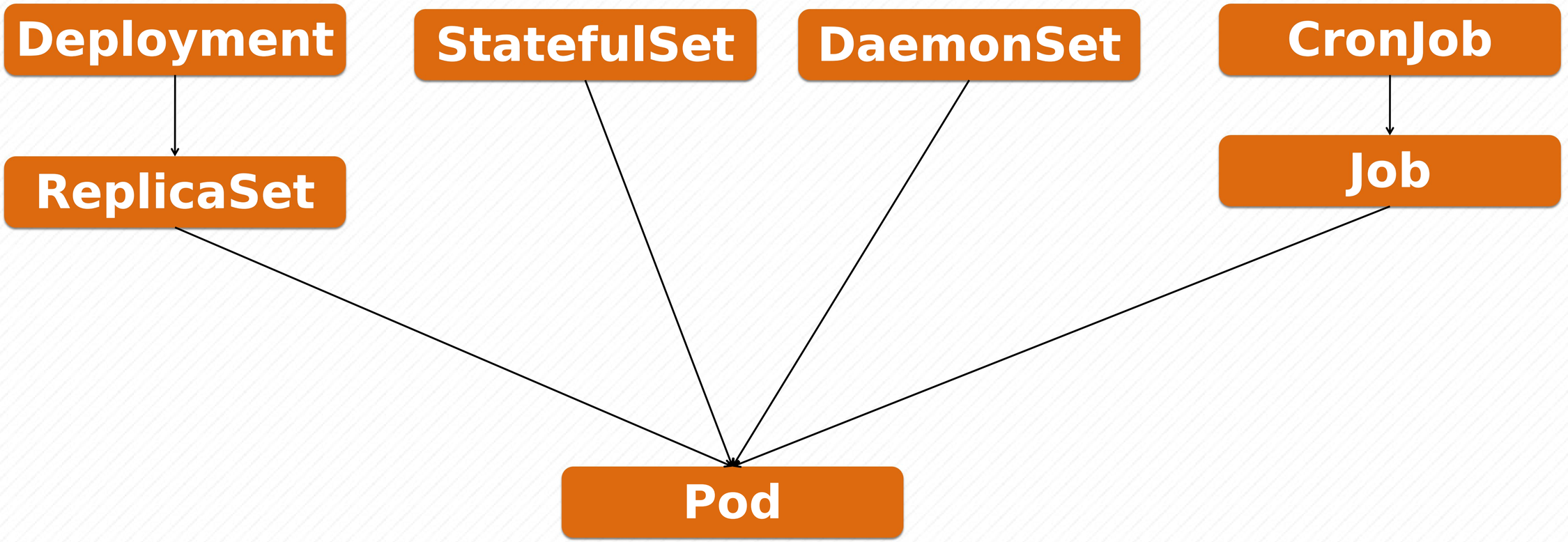
Pod: 嵌套的容器





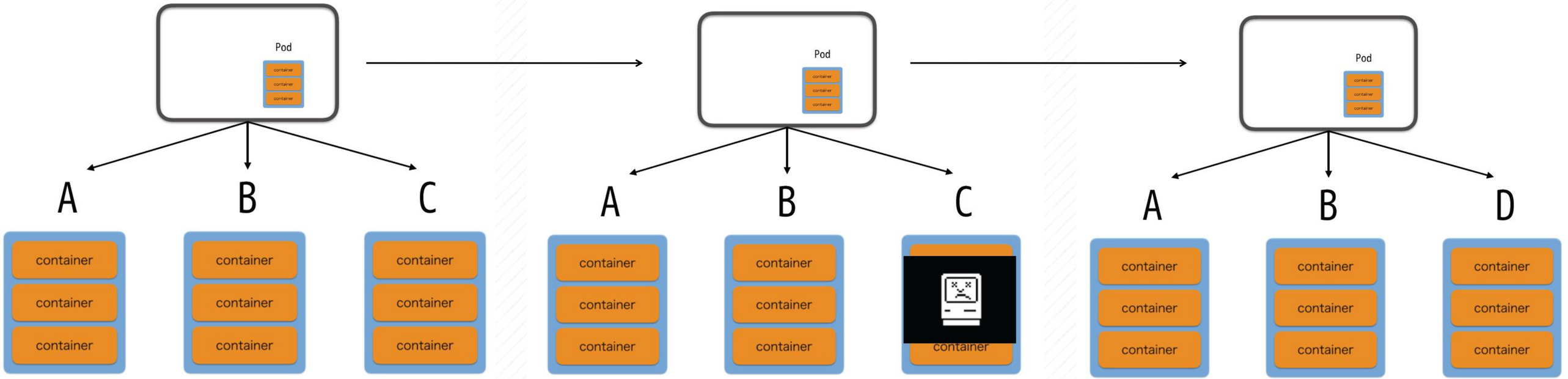
Pod是一组紧密关联的容器集合，它们共享IPC、Network和UTC namespace，是Kubernetes调度的基本单位。Pod的设计理念是支持多个容器在一个Pod中共享网络和文件系统，可以通过进程间通信和文件共享这种简单高效的方式组合完成服务。

Kubernetes基础-app



Kubernetes基础-app

Replica set: keeps track of pod replicas

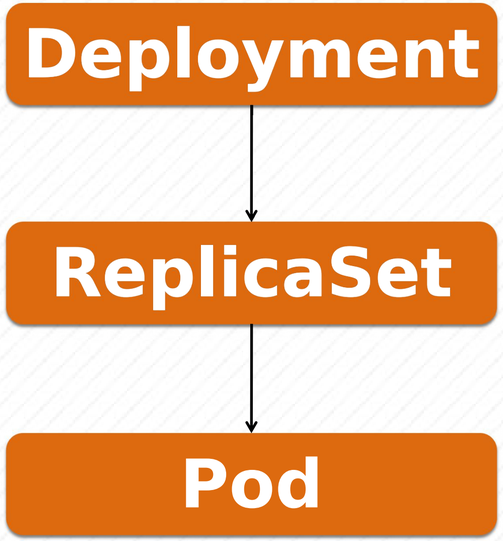


Kubernetes基础-app

Deployment:change the actual state to the desired state

特点

1. 定义Deployment来创建Pod和ReplicaSet
2. 滚动升级和回滚应用
3. 扩容和缩容(通过HPA根据CPU等指标实现自动扩缩容)
4. 暂停和继续Deployment




```yaml
apiVersion: apps/v1beta2 # fo
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Kubernetes基础-app

StatefulSets: Keeps unique IDs in replicas

Deployment	**StatefulSet**
Will roll out a Replica Set and can scale to facilitate more load.	Ordered, graceful deployment and scaling.
Rolling updates are done by updating the PodTemplateSpec of the Deployment. A new ReplicaSet is created and the Deployment manages moving the Pods from the old ReplicaSet to the new one at a controlled rate.	Ordered, automated rolling updates.
Rollback to an earlier Deployment.	No rollback, only deleting and scaling down.
Networking Service is separate.	StatefulSets currently require a Headless Service to be responsible for the network identity of the Pods.
Documentation doesn’t say it’s in beta but the api requires use of the `v1beta1` tag.	Currently in Beta.

Kubernetes基础-app

有状态应用拥有以下特点：

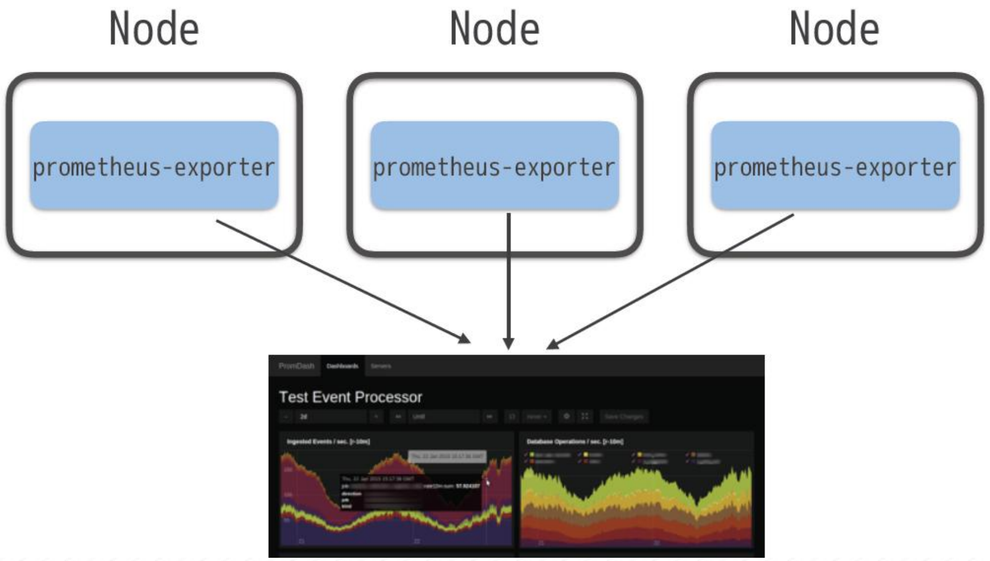
1. 稳定唯一的网络标识：Pod重新调度后，podName和hostName等信息保持不变
2. 稳定的持久化存储：Pod通过PVC来使用持久化存储，重新调度后，仍然能够访问之前的持久化数据.
3. 有序优雅的部署和扩展：pod是有序创建和扩展，在下一个pod创建的时候，之前创建的pod都是运行中的.
4. 有序优雅的删除和终止：statefulsets关联的pod，是根据创建pod的顺序相反依次进行删除.
5. 有序的自动滚动更新：pod更新是有序进行更新.

```
apiVersion: apps/v1beta1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: gcr.io/google_containers/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: my-storage-class
      resources:
        requests:
          storage: 1Gi
```

Kubernetes基础-app

DaemonSets: Ensure nodes run a copy of a pod

DaemonSet确保所有（或某些）节点运行pod的副本。 随着节点被添加到集群，pod被添加到它们。 当从集群中删除节点时，这些pod被垃圾回收。 删除DaemonSet将清理它创建的pod。




```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: gcr.io/google-containers/fluentd-elasticsearch:1.20
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

Kubernetes基础-app

job: creates one or more pods and ensures that a specified number of them successfully terminate.

Job创建一个或多个pod，并确保其指定数量的成功终止。 随着pod成功完成，Job将跟踪Pod是否成功的完成，适用于批处理的任务。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      name: pi
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
```

Kubernetes基础-app

cronjob: manages time based Jobs.

在指定的时间点执行的任务或者需要指定的时间点反复执行的定时任务 一个CronJob对象就像一个crontab（cron表）文件的一行。 它以给定的时间表定期运行，以Cron格式写入。

典型的用例是：

1. 在给定时间点安排Job执行。
2. 创建定期工作，例如 数据库备份，发送电子邮件

```
apiVersion: batch/v2alpha1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

Kubernetes基础-config

ConfigMap: 配置数据

1. 应用镜像与配置分离
2. 数据类型与键值对

Pod使用ConfigMap的场景

1. 环境变量
2. 数据卷文件

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: special-config
  namespace: default
data:
  special.how: very
  special.type: charm
```

```
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: gcr.io/google_containers/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: SPECIAL_LEVEL_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.how
        - name: SPECIAL_TYPE_KEY
          valueFrom:
            configMapKeyRef:
              name: special-config
              key: special.type
  restartPolicy: Never
```

Kubernetes基础-config

secret:解决了密码、token、密钥等敏感数据的配置问题

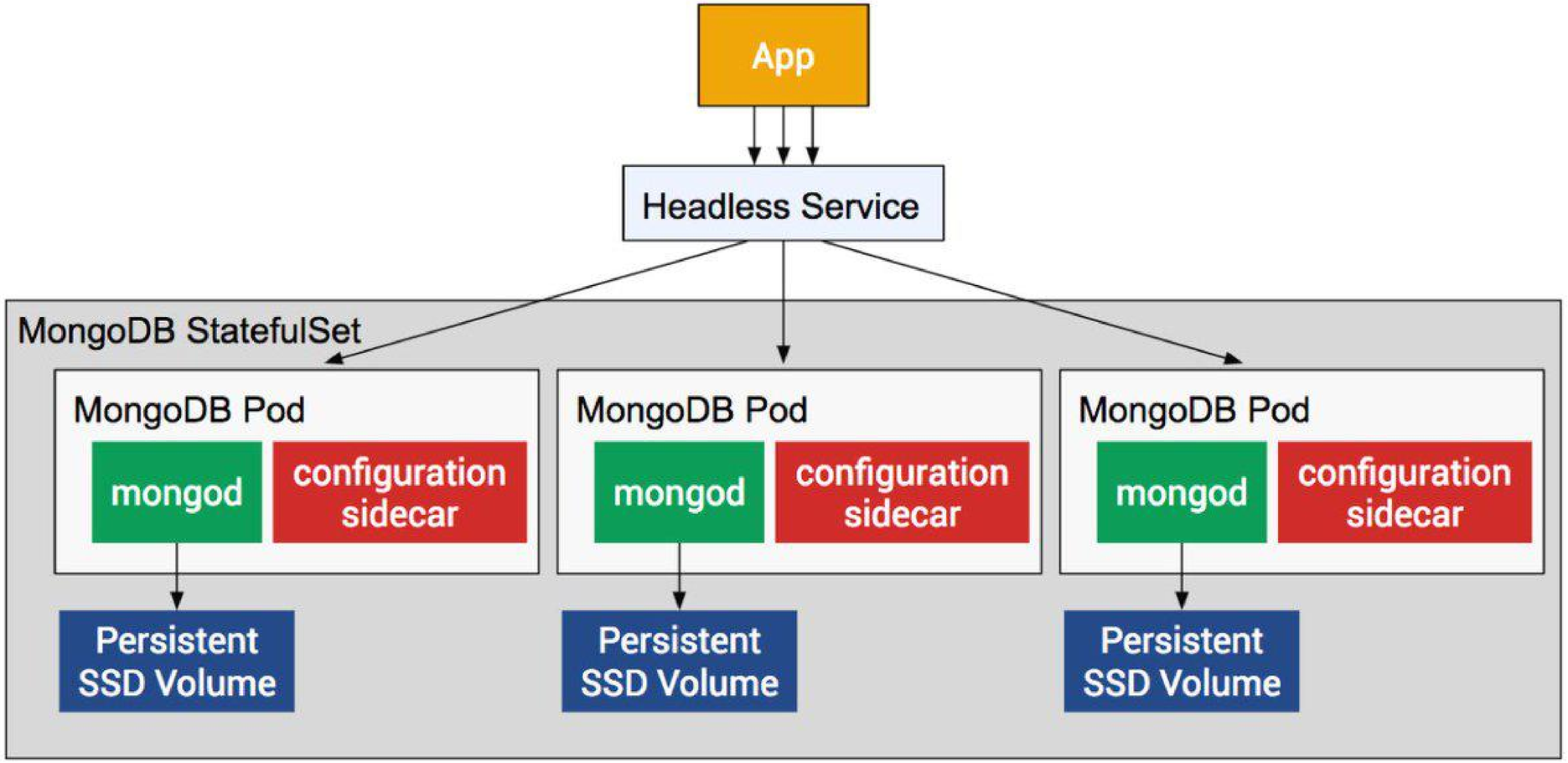
三种类型：

- Service Account：用来访问Kubernetes API，由Kubernetes自动创建，并且会自动挂载到Pod的/run/secrets/kubernetes.io/serviceaccount目录中；
- Opaque：base64编码格式的Secret，用来存储密码、密钥等；
- kubernetes.io/dockerconfigjson：用来存储私有docker registry的认证信息。

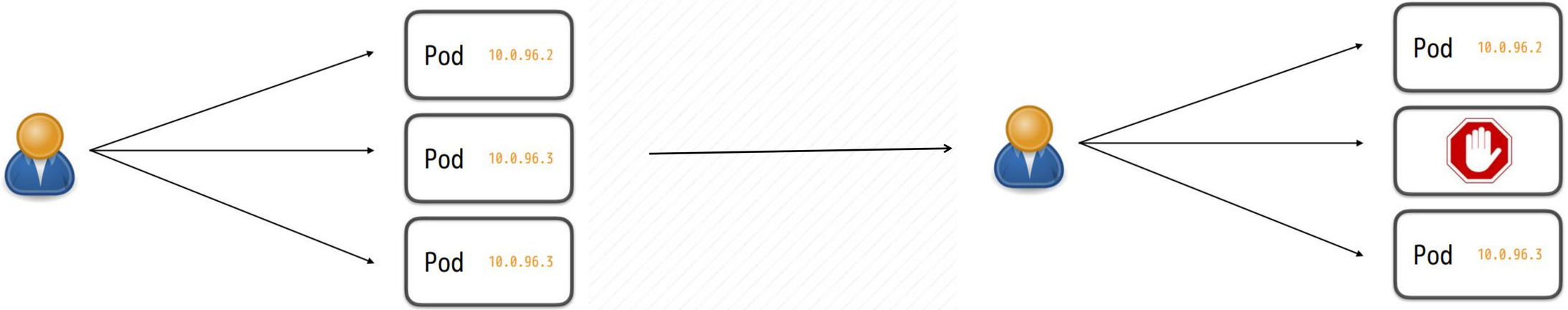
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  username: YWRtaW4=
  password: MWYyZDFlMmU2N2Rm
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
    - name: mycontainer
      image: busybox
      imagePullPolicy: IfNotPresent
      command:
        - sleep
        - "3600"
      env:
        - name: SECRET_USERNAME
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: username
        - name: SECRET_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysecret
              key: password
```

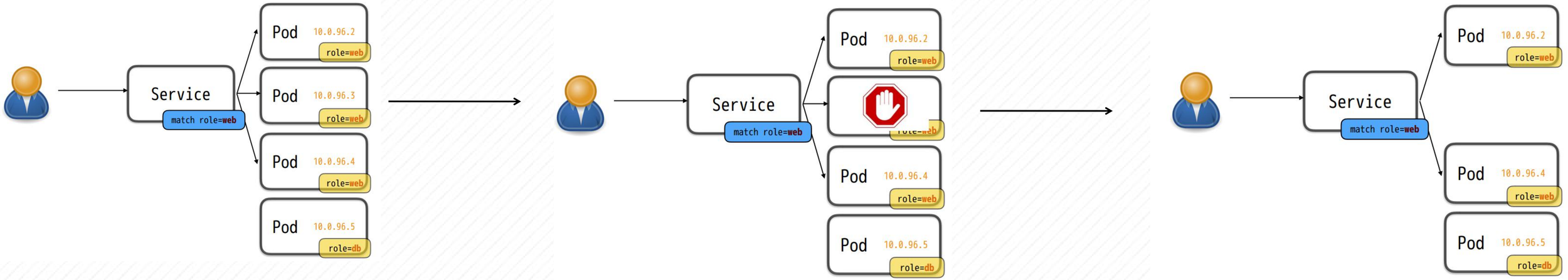
Kubernetes基础-demo



Kubernetes基础-service



Kubernetes基础-service



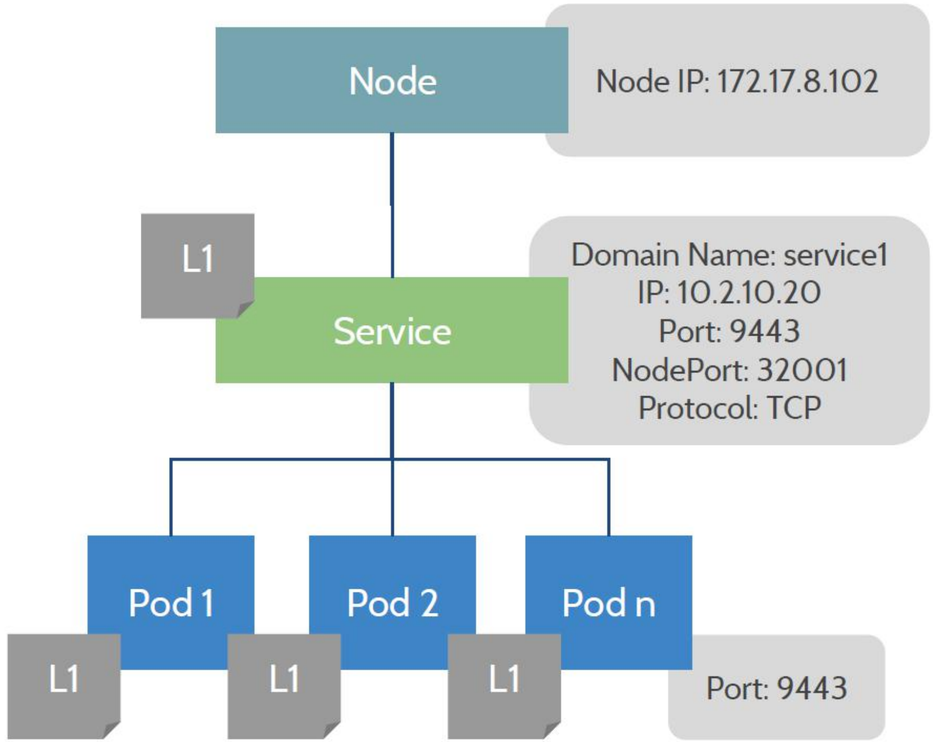
Kubernetes基础-service

service types

- clusterIP
- NodePort
- LoadBalancer
- Ingress

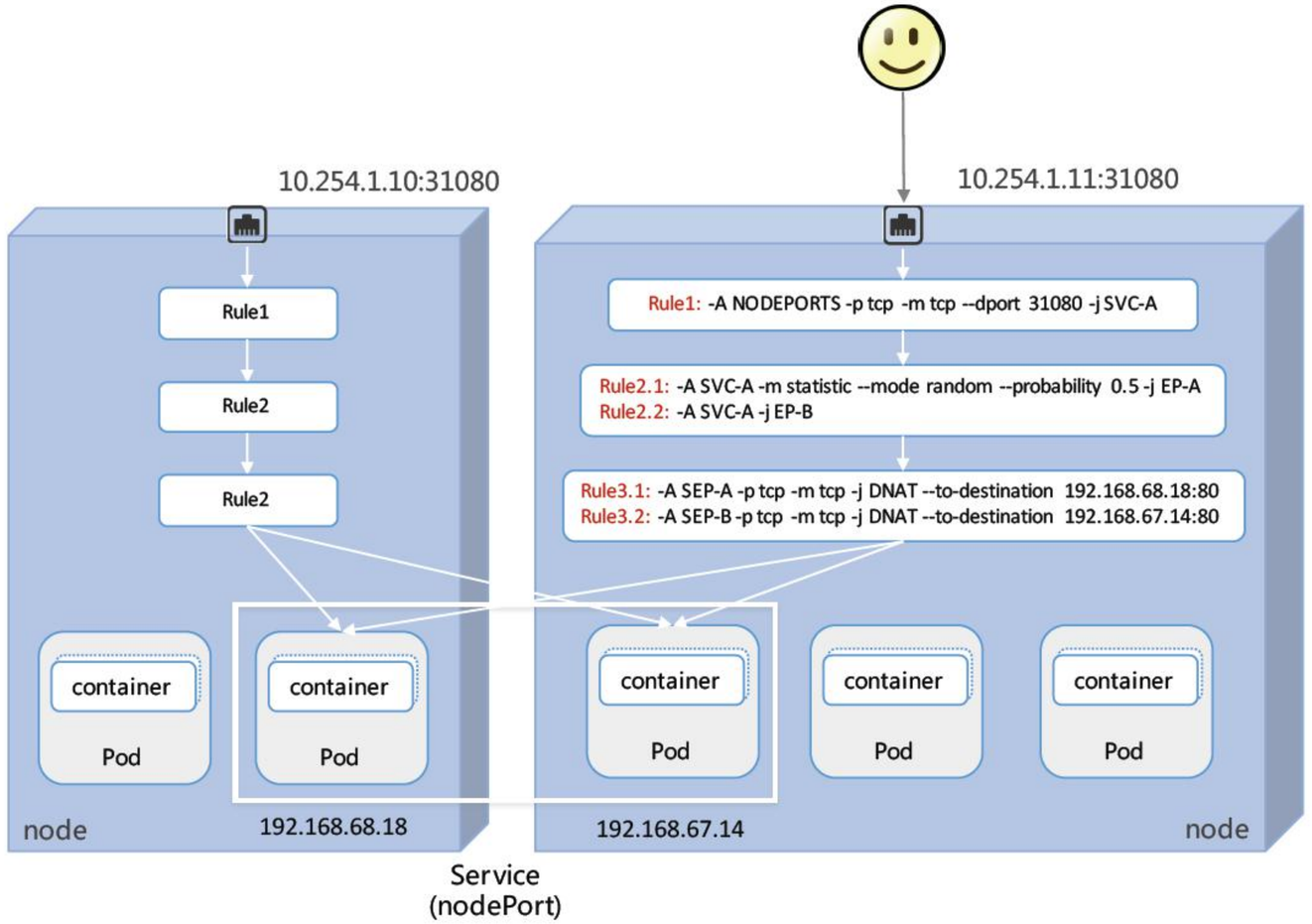
Service discovery

- DNS
- Environment variable



Kubernetes基础-service

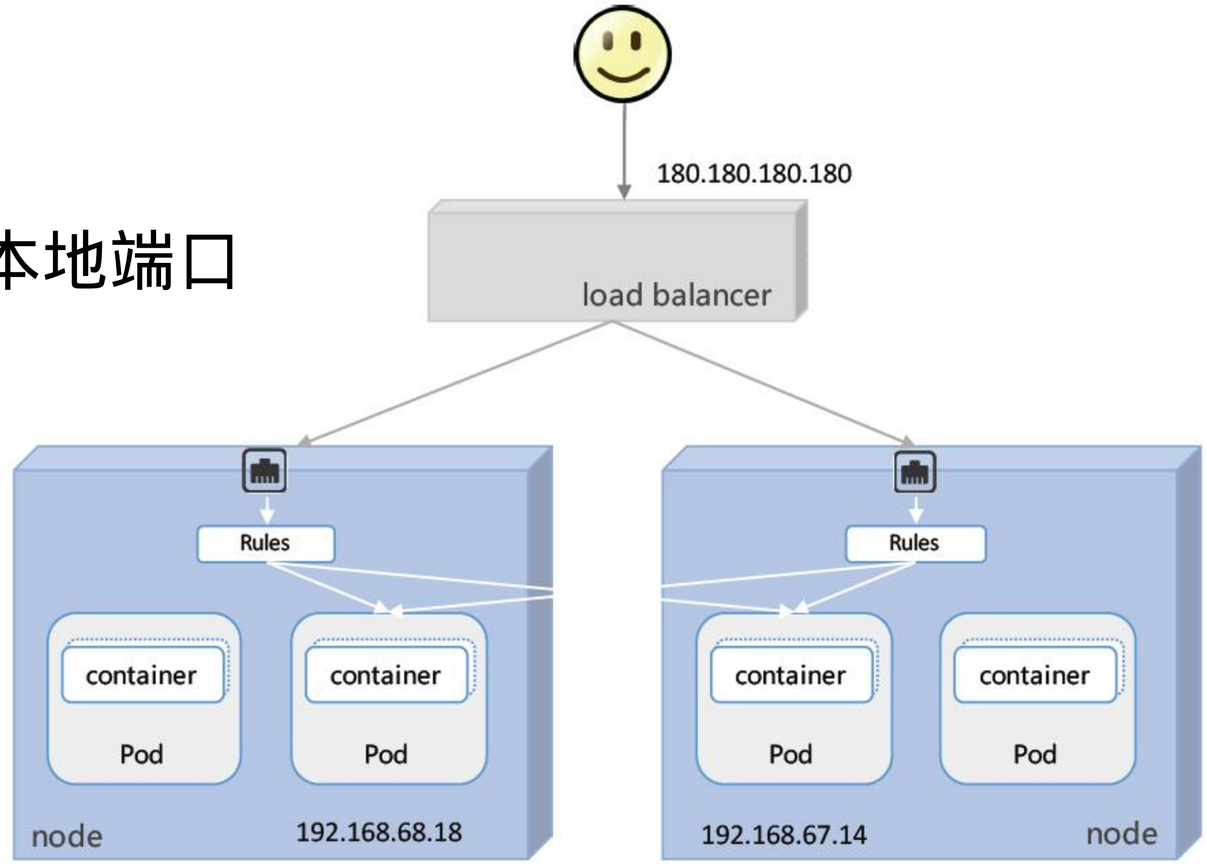
NodePort
1．对外提供服务暴露
2．每个service占用每个node一个本地端口



Kubernetes基础-service

LoadBalance

1．对外提供服务暴露
2．每个service占用每个node一个本地端口
3．需要cloud provider支持
4．可以使用cloud provider提供的loadBalancer的特性



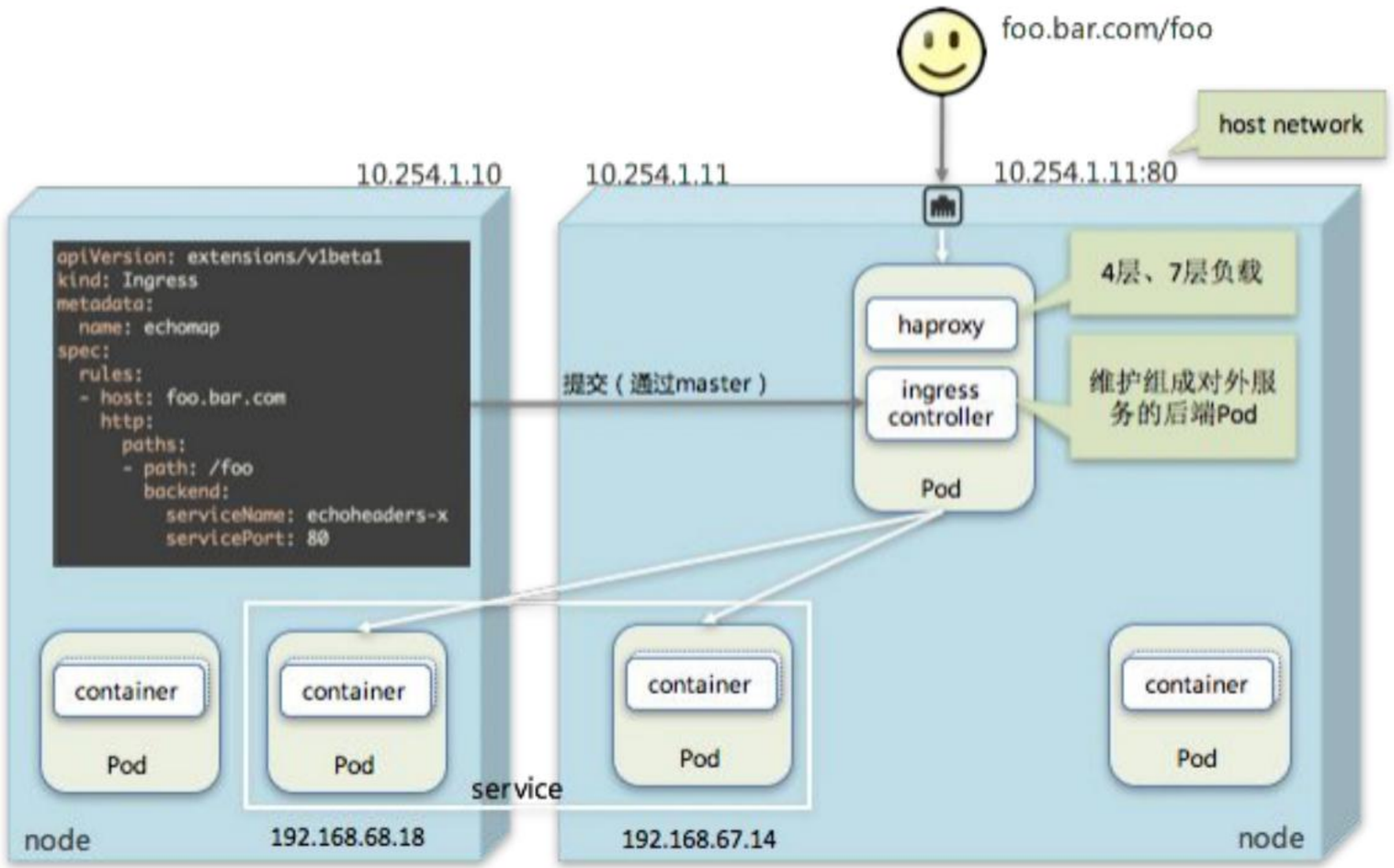
Kubernetes基础-service

Ingress

1．对外提供服务暴露

2．占用每个node一个本地端口

3．灵活性高，支持有限



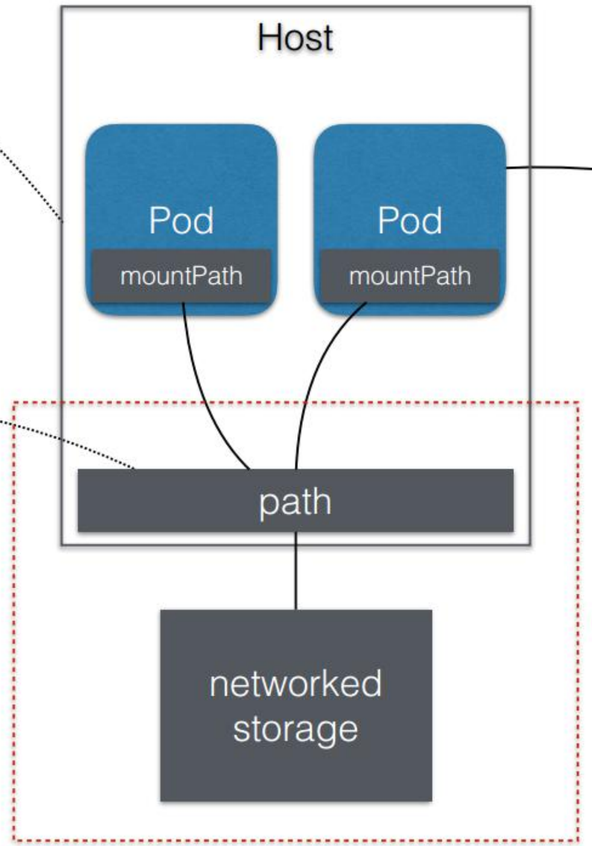
Kubernetes存储

PersistentVolumeClaims

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim-1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

Persistent Volumes

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0001
  labels:
    type: local
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  nfs:
    path: /tmp
    server: 172.17.0.2
```



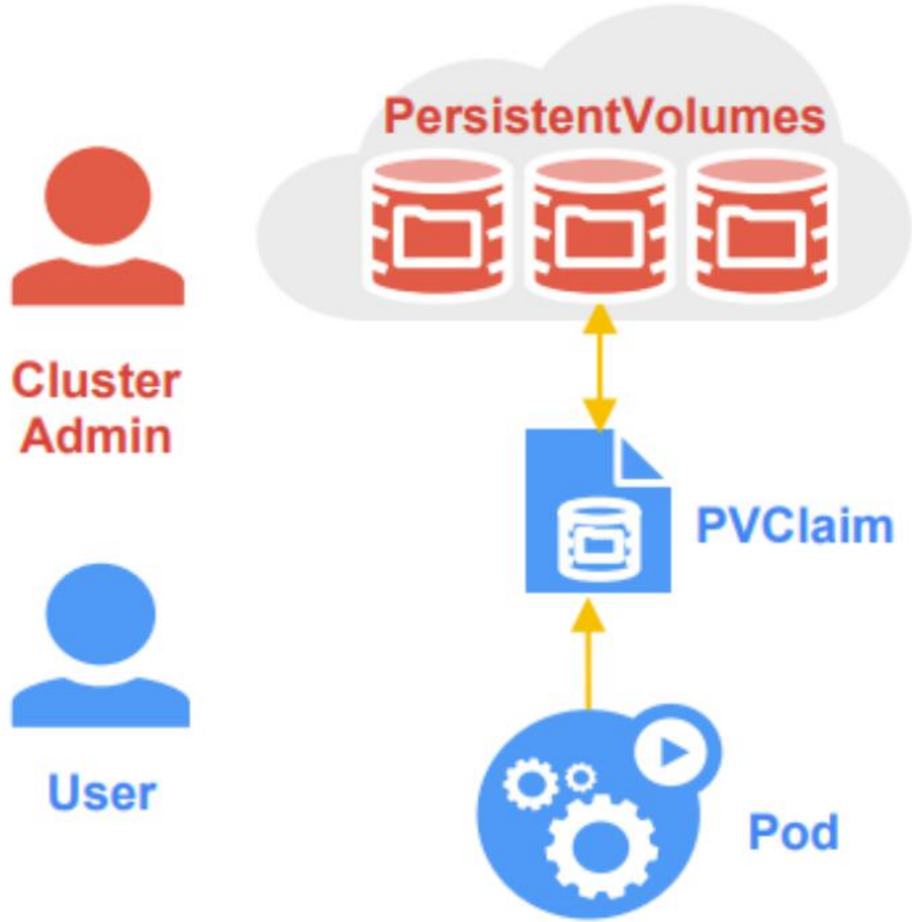
Pod

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
  labels:
    name: frontendhttp
spec:
  containers:
    - name: myfrontend
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: myclaim-1
```

Key point: 职责分离

Kubernetes存储

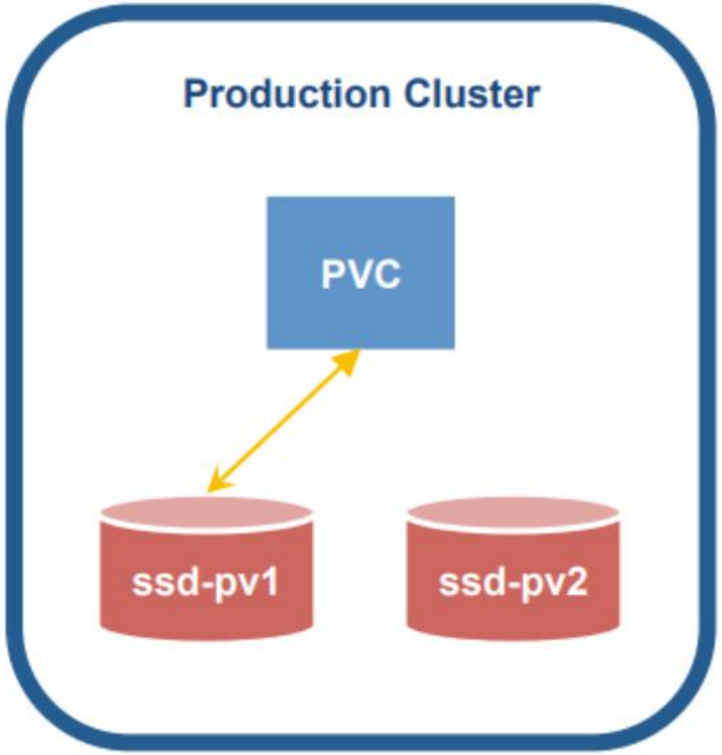
- PersistentVolumeClaim (PVC) API Object
  - Request for storage by a user
  - Claims request generic parameters
    - Capacity
    - Access modes
      - read many, write many, etc
  - Pods reference claims
  - Claim gets bound to a matching PV
  - User story is portable, cluster-agnostic



Kubernetes存储

StorageClass
Used when a PersistentVolume belonging to the class needs to be dynamically Provisioned.

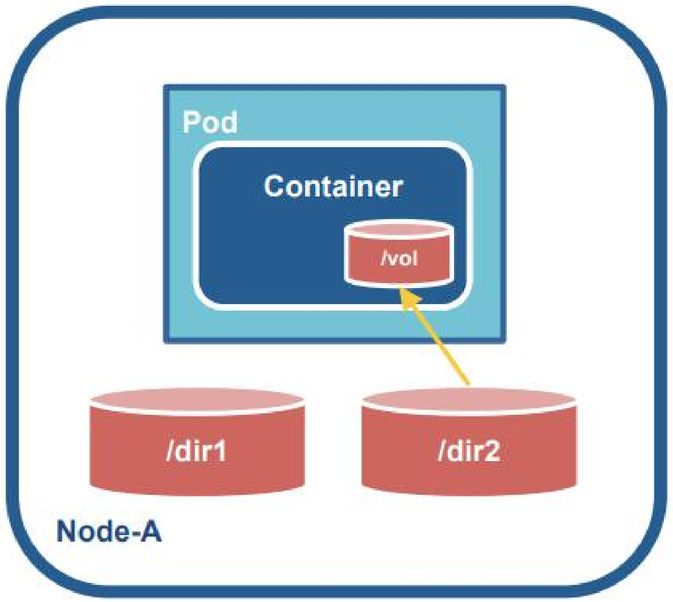
```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
```



Kubernetes存储

支持的存储插件

GCEPersistentDisk
AWSElasticBlockStore
AzureFile
AzureDisk
FlexVolume
FC
Flocker
NFS
iSCSI
RBD
CephFS
Cinder
Glusterfs
VsphereVolume
Quobyte Volume
HostPath
Wmware Photon




```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
  - name: busybox
    image: busybox
    volumeMounts:
    - name: my-vol
      mountPath: /vol
  volumes:
  - name: my-vol
    hostpath:
      path: /dir2
```

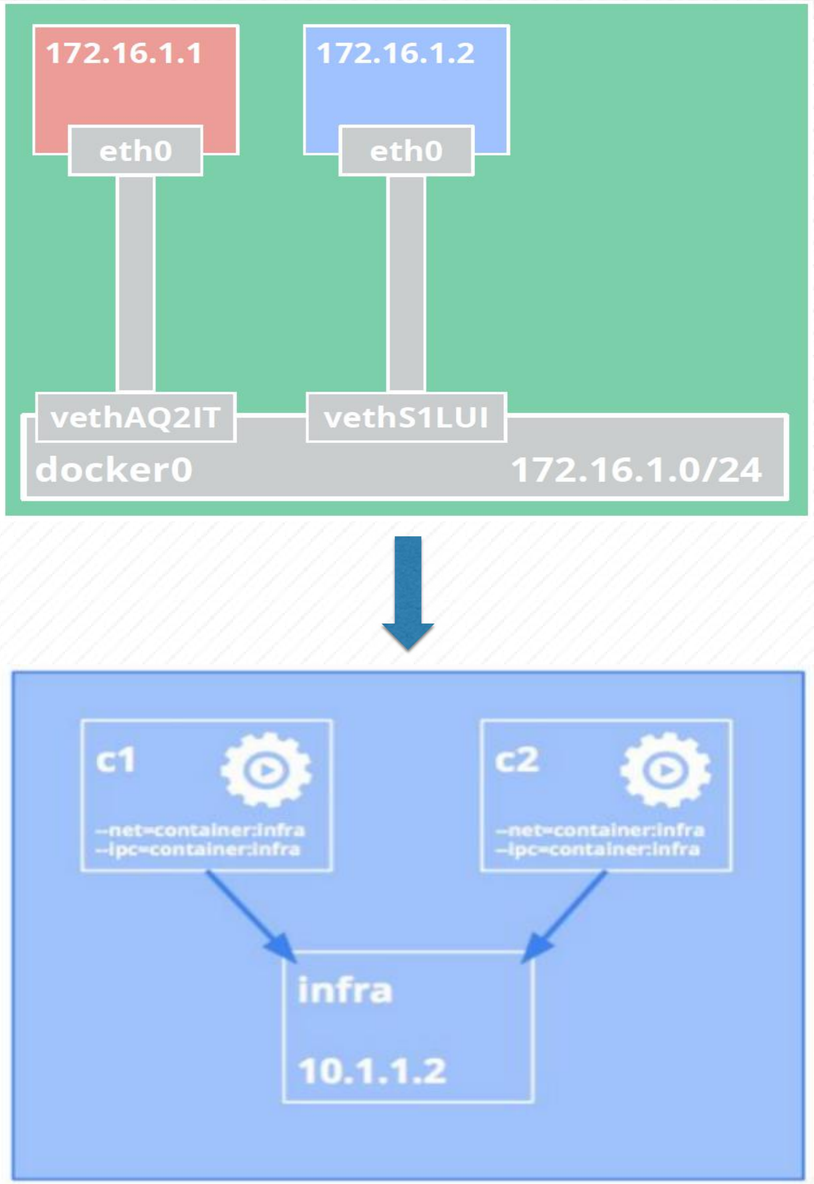
Kubernetes网络

Pods Networking

shared namespace

1. shared ip address & localhost
2. shared IPC, etc

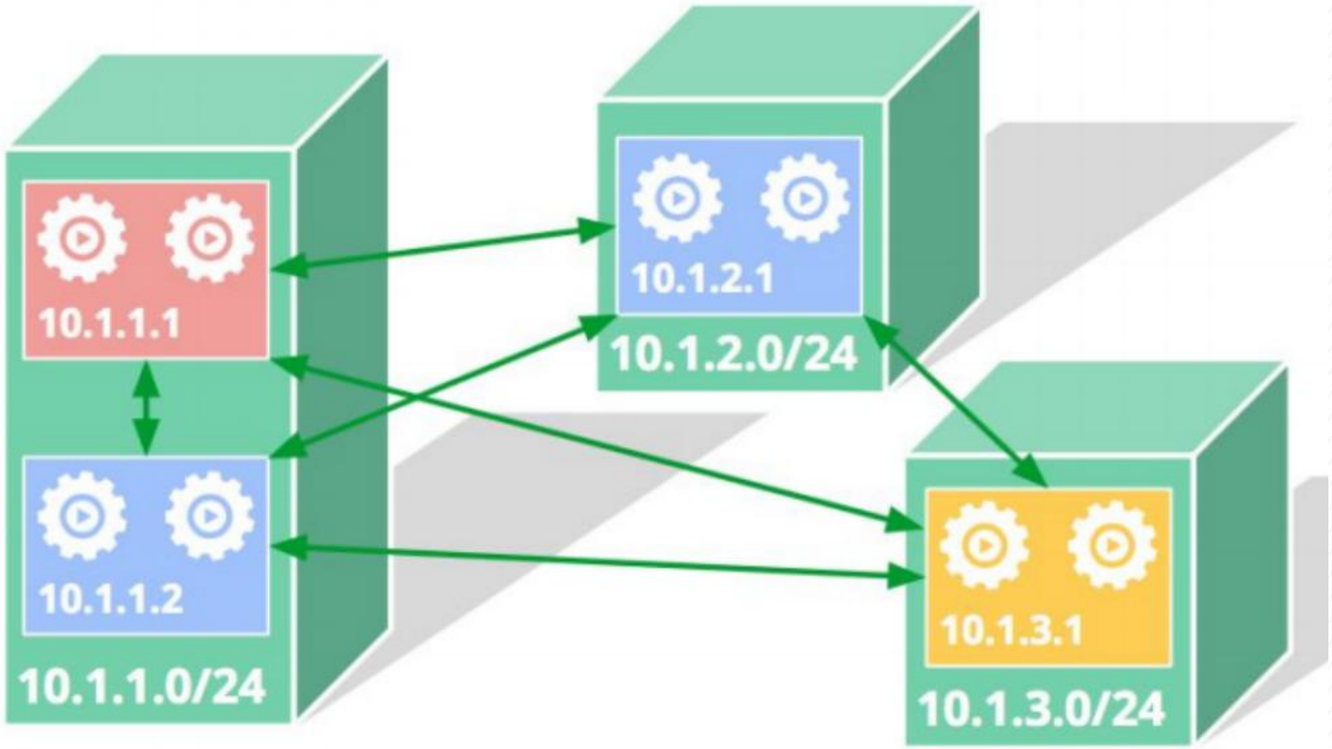
Pod as a small group of containers&volumes, the atom of scheduling & placement.



Kubernetes网络

Kubernetes Networking

1. Ips are routable
2. Every host is assigned a subnet

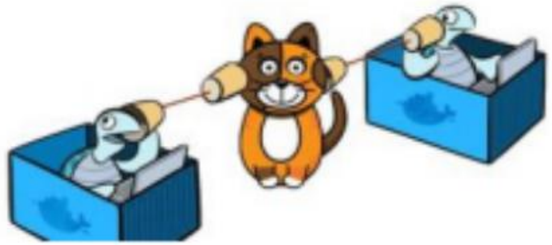


Kubernetes网络

- On GCE/GKE
  - GCE advanced routes (program the fabric)
  - Everything to “10.1.1.0/24”, send to this VM
- On AWS
  - Route table
- Plenty of other ways
  - Weave
  - Calico
  - Flannel
  - OVS
  - OpenContrail
  - Cisco Contiv
  - Canal
  - Others
  - ...

Google Cloud Platform

weave



CISCO

OvS Open vSwitch

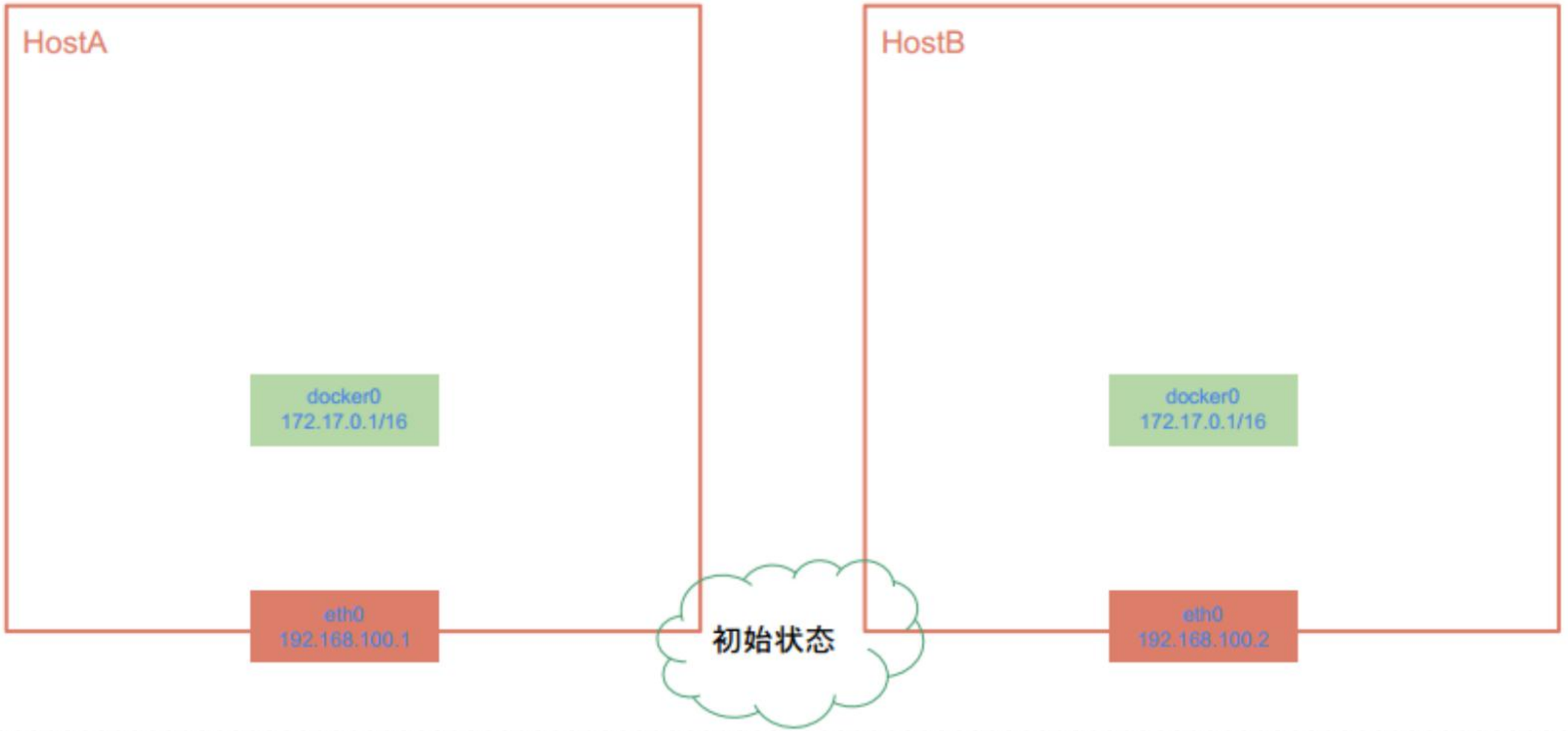
OPENCONTRAIL

Kubernetes网络

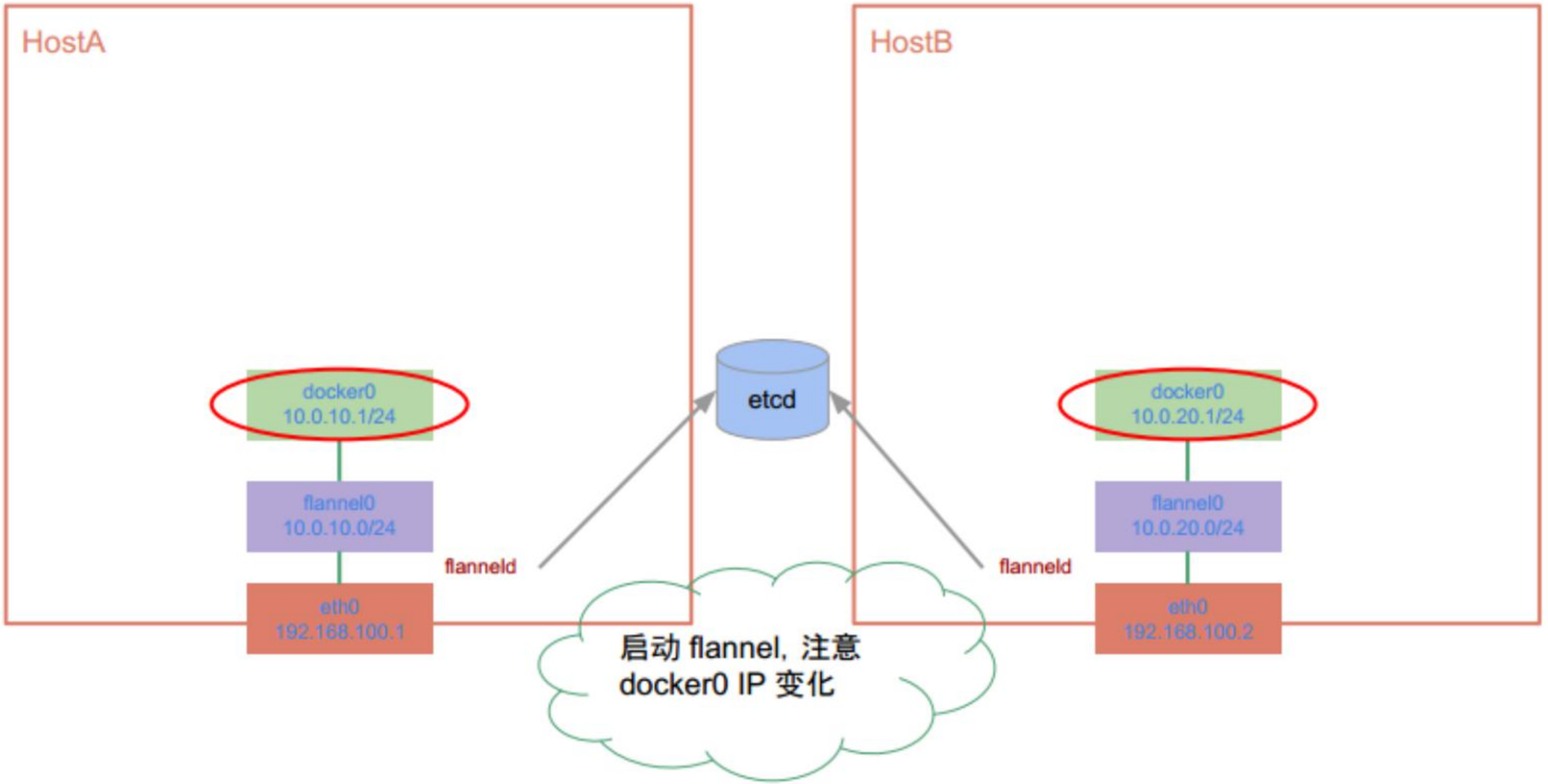
Flannel

1. 解决docker容器直接跨主机的通信
2. 为每个主机分配一个IP段，然后主机上的容器被分配该IP网络的不同的IP
3. 所有的容器在一个虚拟的二层网络，底层通过UDP/VxLAN等进行报文的封装和转发

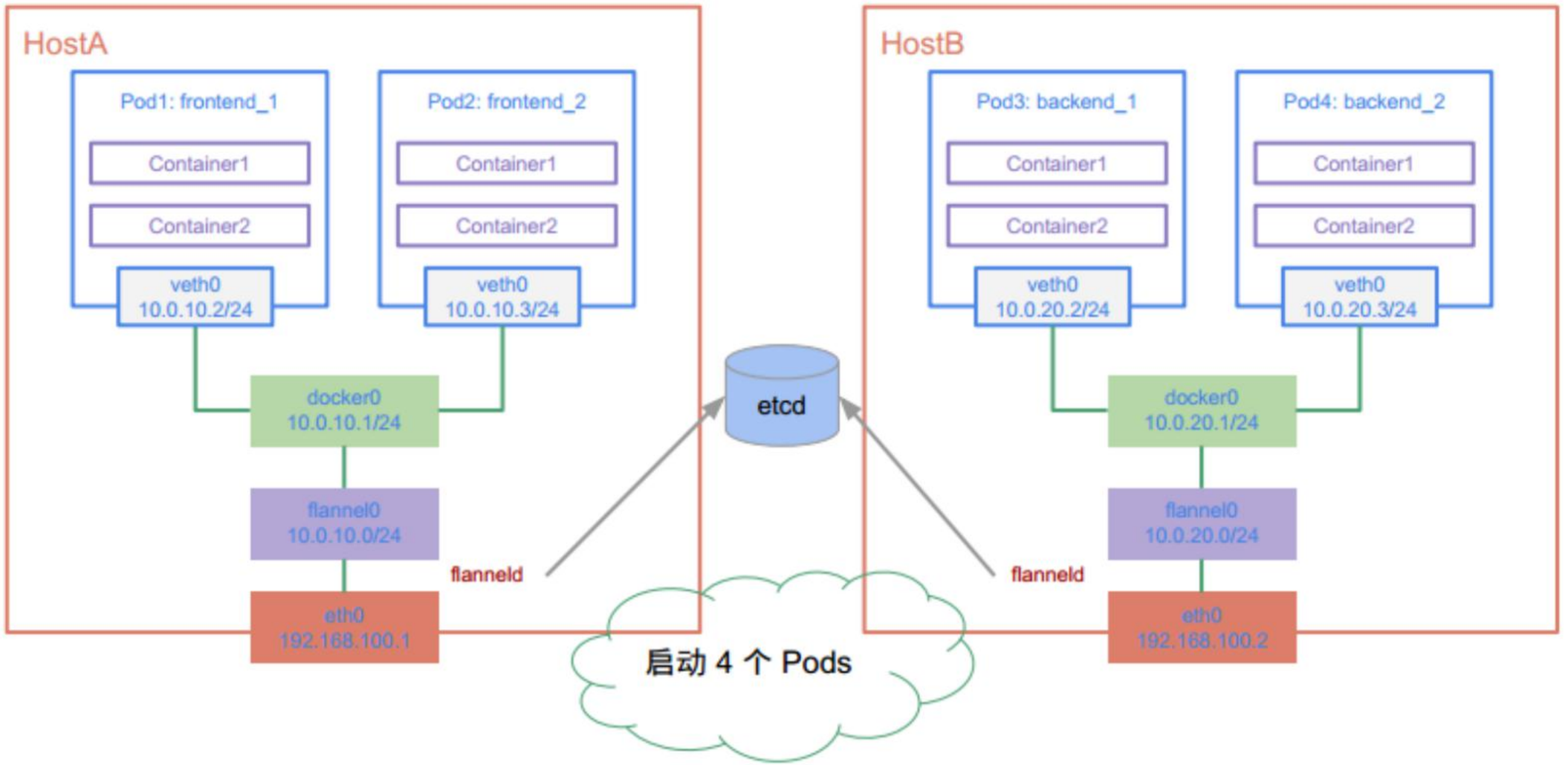
Kubernetes网络



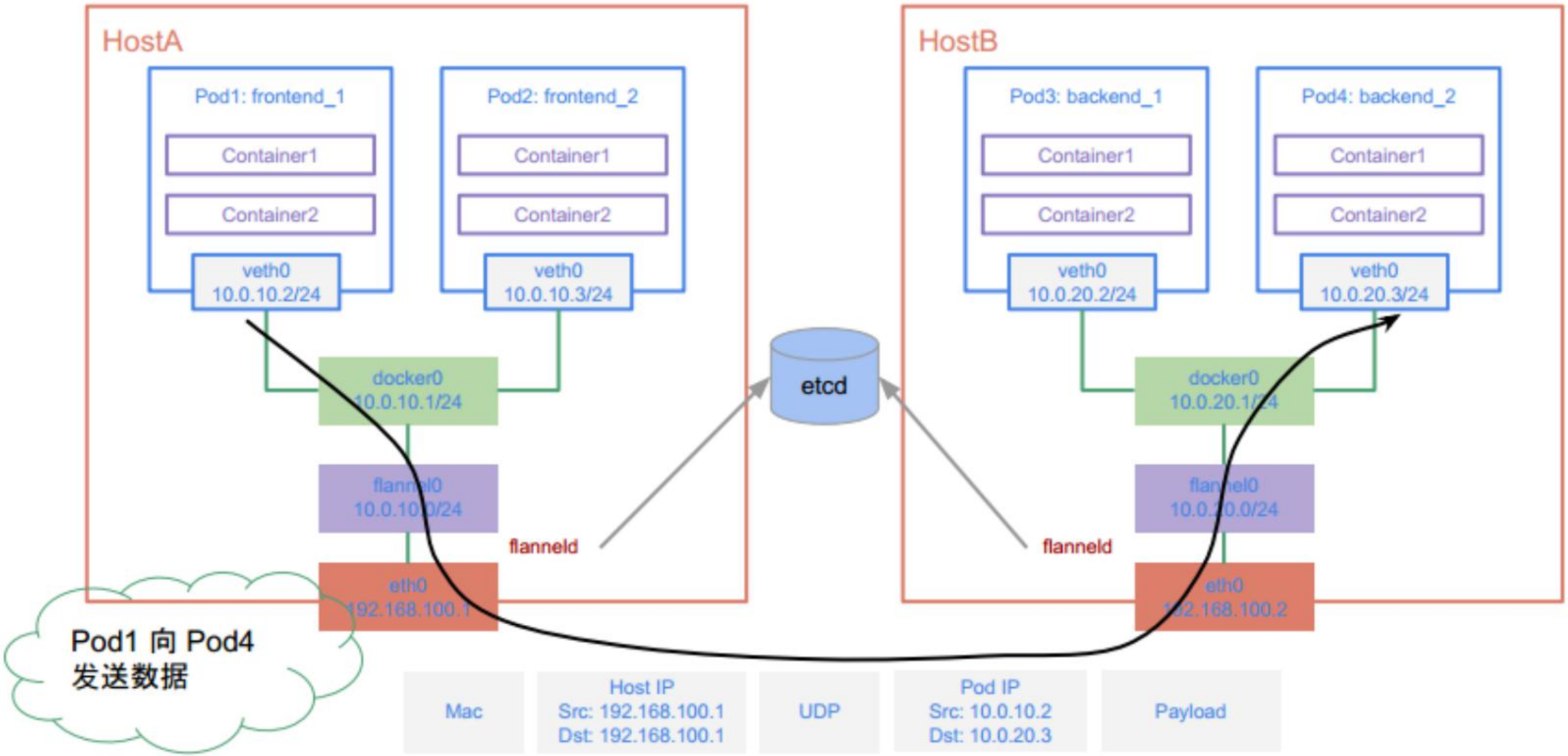
Kubernetes网络



Kubernetes网络



Kubernetes网络



Kubernetes部署

社区部署工具

minkube
  https://kubernetes.io/docs/getting/ started/ guides/minikube/
kubeadm
kops
kubespray
Run public cloud(AWS/GCE/Rancher)
Custom Solutions

Kubernetes部署

部署工具对比

名称	特点
minikube	适合新手入门，启动虚拟机allinone运行kubernetes，灵活性低
kubeadm	Go语言，提供Kubernetes的安装，扩展，升级
kops	生产级别的Kubernetes部署
kubespray	Ansible脚本，适合于aws公有云场景下部署kubernetes
Run public cloud	使用云提供商提供的kubernetes部署工具
custom solution	定制化，灵活性高

Kubernetes部署

Custom Solution

https://kubernetes.io/docs/setup/

Kubernetes部署

常见问题解决

- 部署Kubernetes组件的过程中pod的状态为Pending

kubectl describe pod xxx查看event事件，可能是image在国外无法下载，可以通过国内的镜像市场来下载，之后docker load在本地。

- 集群状态不正常

https://kubernetes.io/docs/tasks/debug/_application/_cluster/debug/_cluster/

- 集群部署成功之后应用不正常

https://kubernetes.io/docs/tasks/debug/_application/_cluster/debug/_application/

THANK YOU