HUAWEI

CloudNative Lives

K8s初体验-快速部署第一个容器应用

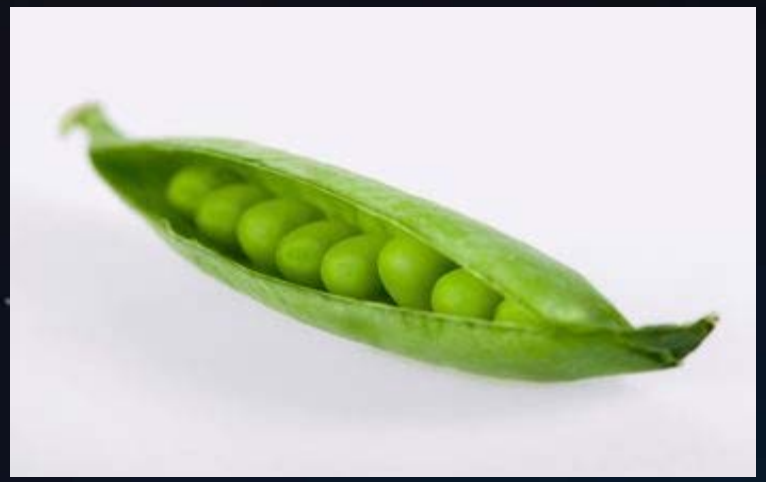
华为云容器团队倾心打造

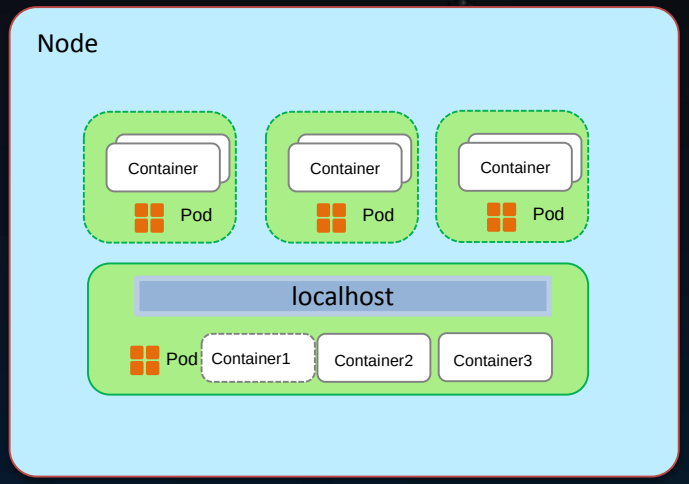
大 纲

- **Kubernetes Pod详解**
- Kubernetes工作负载与服务介绍
- 快速部署第一个容器应用

Kubernets概览

Kubernetes关键概念-Pod





- 在Kubernetes中，pods是能够创建、调度、和管理的最小部署单元，是一组容器的集合，而不是单独的应用容器
- 同一个Pod里的容器共享同一个网络命名空间，IP地址及端口空间。
- 从生命周期来说，Pod是短暂的而不是长久的应用。Pods被调度到节点，保持在这个节点上直到被销毁。

POD实例：

```
{
 "kind": "Pod",
 "apiVersion": "v1",
 "metadata": {
  "name": "redis-django",
  "labels": {
   "app": "webapp"
  }
 },
 "spec": {
  "containers": [
   {
    "name": "key-value-store",
    "image": "redis"
   },
   {
    "name": "frontend",
    "image": "django"
   }
  ]
 }
}
```

Pod详解-容器（Containers）

HUAWEI

容器分类

- Infrastructure Container：基础容器
  - 用户不可见，无需感知
  - 维护整个Pod网络空间
- InitContainers：初始化容器，一般用于服务等待处理以及注册Pod信息等
  - 先于业务容器开始执行
  - 顺序执行，执行成功退出（exit 0），全部执行成功后开始启动业务容器
- Containers：业务容器
  - 并行启动，启动成功后一直Running

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'until nslookup myservice; do echo waiting for myservice; sleep 2; done;']
  - name: init-mydb
    image: busybox
    command: ['sh', '-c', 'until nslookup mydb; do echo waiting for mydb; sleep 2; done;']
```

容器基本组成

镜像部分：

- 镜像地址和拉取策略
- 拉取镜像的认证凭据

启动命令：

- command：替换docker容器的entrypoint
- args：作为docker容器entrypoint的入参

计算资源：

- 请求值：调度依据
- 限制值：容器最大能使用的规格

```
spec:
  imagePullSecrets:
  - name: default-secret
  containers:
  - image: kube-dns:1.0.0
    imagePullPolicy: IfNotPresent
    command:
    - /bin/sh
    - -c
    - /kube-dns 1>>/var/log/skydns.log 2>&1 --domain=cluster.local. --dns-port=10053
      --config-dir=/kube-dns-config --v=2
    resources:
      limits:
        cpu: 100m
        memory: 512Mi
      requests:
        cpu: 100m
        memory: 100Mi
```

Pod详解-健康检查

健康检查：

分命令行方式、httpGet请求方式以及TCPSocket方式。

- 业务探针（readinessProbe）
  - 探测不正常后，不会重启容器，只会拿掉服务后端的endpoints
- 存活探针（livenessProbe）
  - 探测不正常后，会重启容器

```
spec:
  containers:
  - livenessProbe:
      failureThreshold: 5
      exec:
        command:
        - “/bin/sh”,
        - “-c”
        - “echo ‘iam ok’”
      initialDelaySeconds: 60
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 5
    readinessProbe:
      failureThreshold: 3
      httpGet:
        path: /readiness
        port: 8081
        scheme: HTTP
      initialDelaySeconds: 3
      periodSeconds: 10
      successThreshold: 1
      timeoutSeconds: 5
```

Pod详解-外部输入

Pod可以接收的外部输入方式：环境变量、配置文件以及密钥。

环境变量：使用简单，但一旦变更后必须重启容器。

- Key-value自定义
- From 配置文件（configmap）
- From 密钥（Secret）

以卷形式挂载到容器内使用，权限可控。

- 配置文件（configmap）
- 密钥（secret）

```
spec:
 containers:
 - env:
   - name: APP_NAME
     value: test
   - name: USER_NAME
     valueFrom:
       secretKeyRef:
         key: username
         name: secret
   envFrom:
   - configMapRef:
       name: config
   volumeMounts:
   - mountPath: /usr/local/config
     name: cfg
   - mountPath: /usr/local/secret
     name: sct
 volumes:
 - configMap:
     defaultMode: 420
     items:
     - key: age
       path: age
     name: config
   name: cfg
 - name: sct
   secret:
     defaultMode: 420
     secretName: secret
```

配置文件（ConfigMap）及密钥（Secret）介绍

HUAWEI

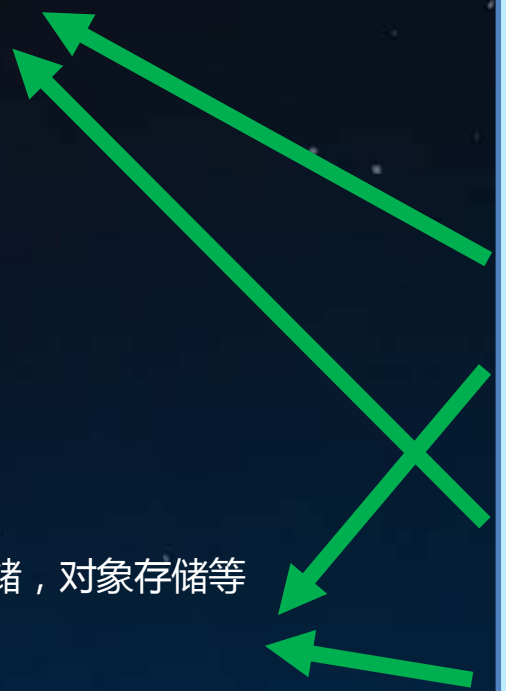
- 大小不能超过1M
- Secret有类型区分，不同类型有不同校验方式，输入的value需base64加密

```
apiVersion: v1
data:
  user: wangbo
  details: |
    xxx
kind: ConfigMap
metadata:
  name: config
```

```
apiVersion: v1
data:
  username: Y3Bvc3RncmVz
  password: cGNwcGNw
kind: Secret
metadata:
  name: secret
type: Opaque
```

Pod详解-持久化存储

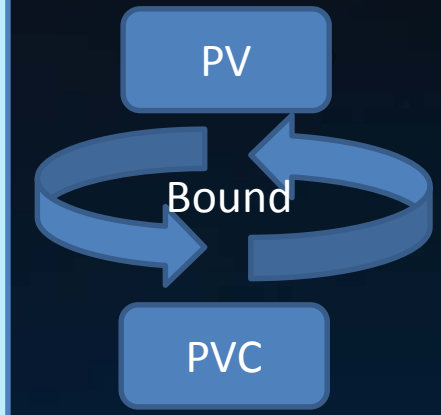
- 主机hostpath
  - 必须要求Pod调度到固定节点上
- 云存储
  - 多类型选择，如云硬盘，文件存储，对象存储等
  - 不用担心Pod迁移



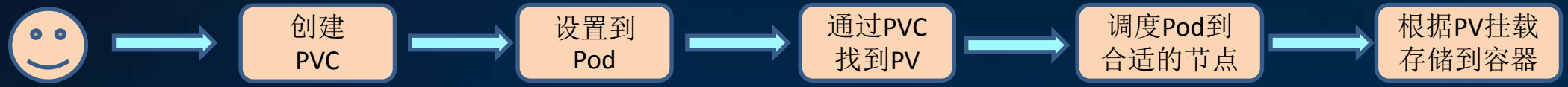
```
spec:
  containers:
  - image: tomcat:9-slim
    imagePullPolicy: IfNotPresent
    name: container-0
    volumeMounts:
    - mountPath: /usr/local/disk
      name: pvc-153085714479448980
    - mountPath: /usr/local/host
      name: host
  volumes:
  - name: pvc-153085714479448980
    persistentVolumeClaim:
      claimName: pvc-153085714479448980
  - name: host
    hostPath:
      path: /opt/
```

PV/PVC介绍

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    volume.beta.kubernetes.io/storage-class: sata
    volume.beta.kubernetes.io/storage-provisioner:
flexvolume-huawei.com/fuxivol
  labels:
    failure-domain.beta.kubernetes.io/region: southchina
    failure-domain.beta.kubernetes.io/zone: az1.dc1
  name: pvc-153085714479448980
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  volumeName: pv-b3377f21-80e2-11e8-ad55-
fa163e5a2cf6
```



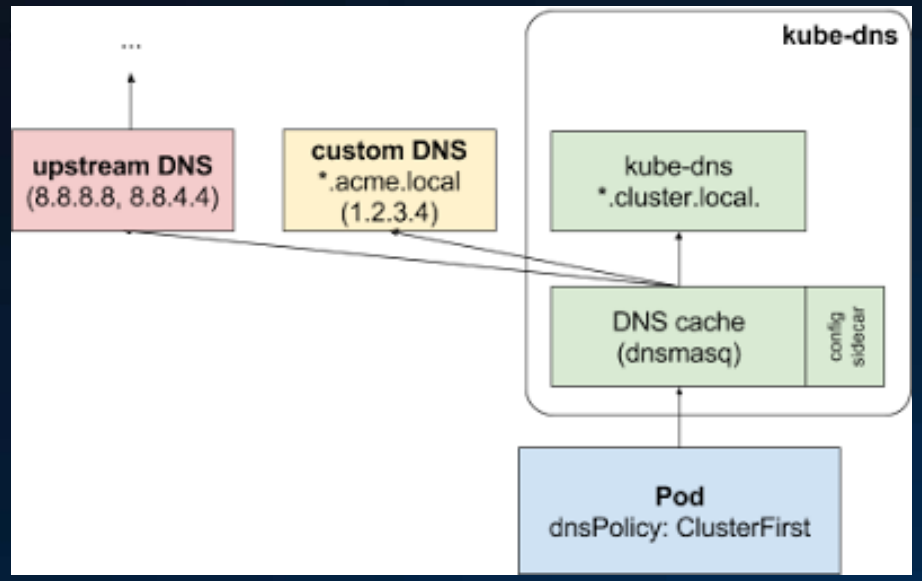

```
apiVersion: v1
kind: PersistentVolume
metadata:
  labels:
    failure-domain.beta.kubernetes.io/region: southchina
    failure-domain.beta.kubernetes.io/zone: az1.dc1
  name: pv-b3377f21-80e2-11e8-ad55-fa163e5a2cf6
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 10Gi
  claimRef:
    apiVersion: v1
    kind: PersistentVolumeClaim
    name: pvc-153085714479448980
flexVolume:
    driver: huawei.com/fuxivol
    fsType: ext4
    options:
      fsType: ext4
      kubernetes.io/namespace: default
      volumeID: 840d1d47-6303-4dd7-9a7f-41f4f80dbc1e
  storageClassName: sata
```

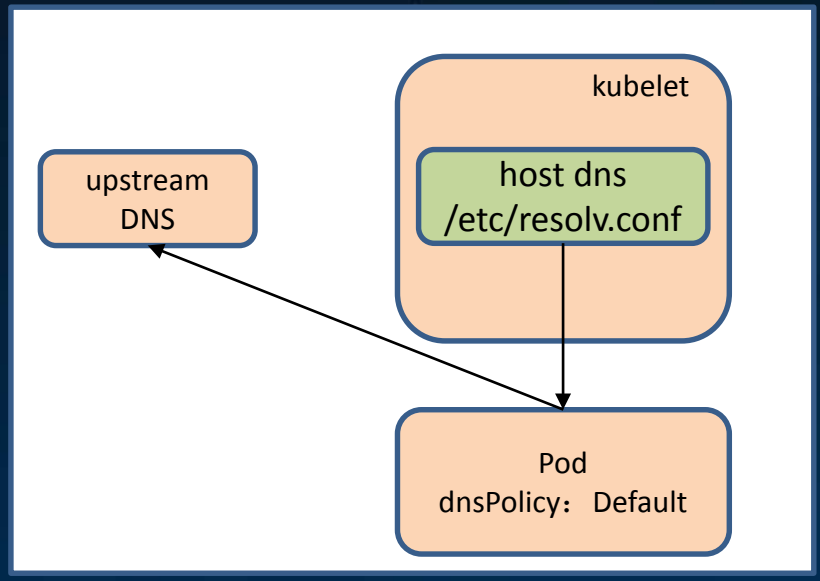


Pod详解-服务域名发现

- dnsPolicy：Pod内域名解析的策略
  - ClusterFirst：使用kube-dns作为域名解析服务器
  - Default：使用节点（kubelet）指定的域名服务器解析域名
  - ClusterFirstWithHostNet：当Pod使用主机网络部署时使用

```
spec:
  containers:
  - command:
    - /bin/sh
    - -c
    - stability --metrics=networkio#3000
    image: stability:1.0
    imagePullPolicy: IfNotPresent
    name: stability
  dnsPolicy: ClusterFirst
```



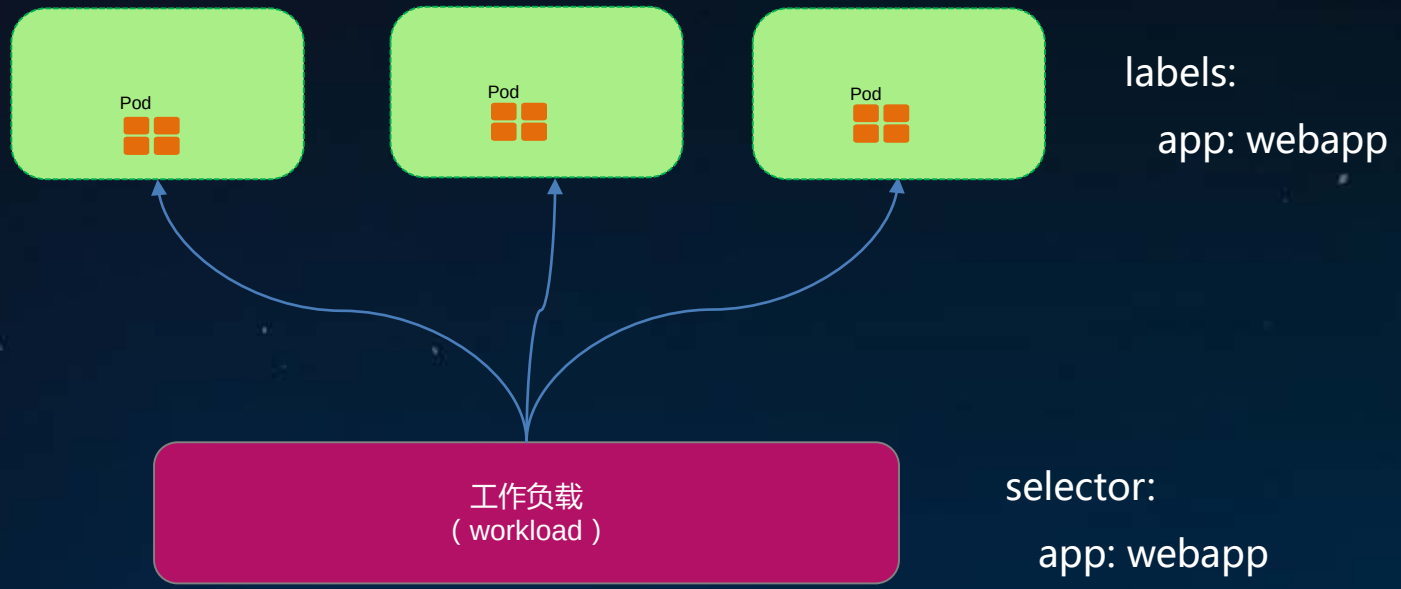


大纲

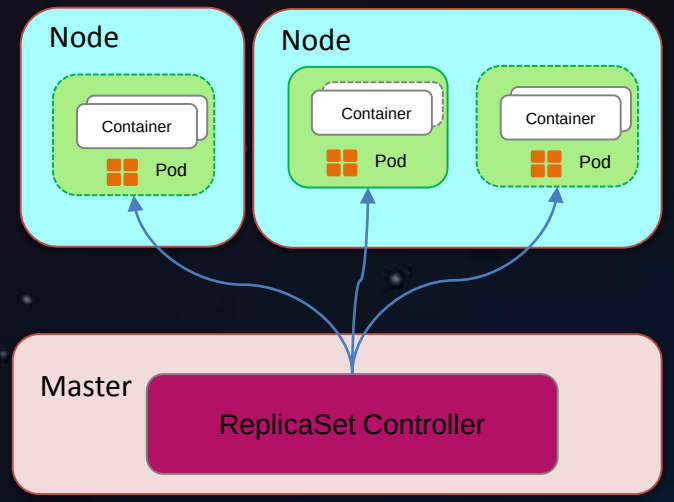
- Kubernetes Pod详解
- **Kubernetes工作负载与服务介绍**
- 快速部署第一个容器应用

Pod与工作负载的关系

- 通过label-selector相关联
- Pod通过工作负载实现应用的运维，如伸缩、升级等



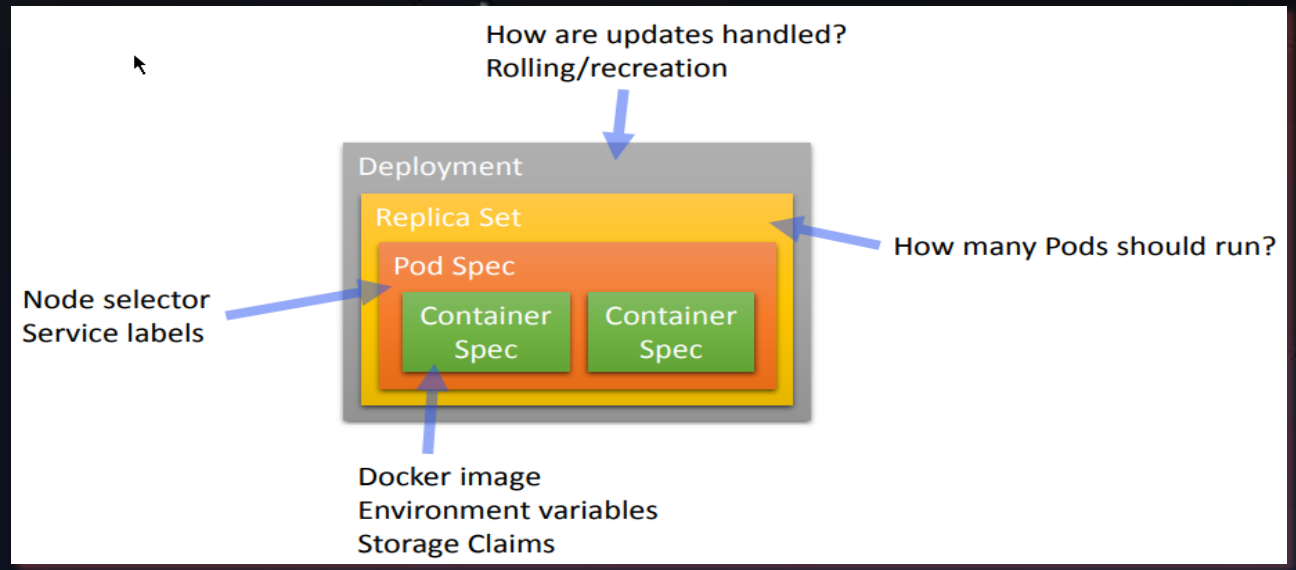
关键工作负载-ReplicaSet



- ReplicaSet—副本控制器
- 确保Pod的一定数量的份数(replica)在运行。如果超过这个数量，控制器会杀死一些，如果少了，控制器会启动一些。
- ReplicaSet用于解决pod的扩容和缩容问题。
- 通常用于无状态应用

```
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
  name: frontend
spec:
replicas: 3
selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: tier, operator: In, values: [frontend]}
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-
frontend:v3
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

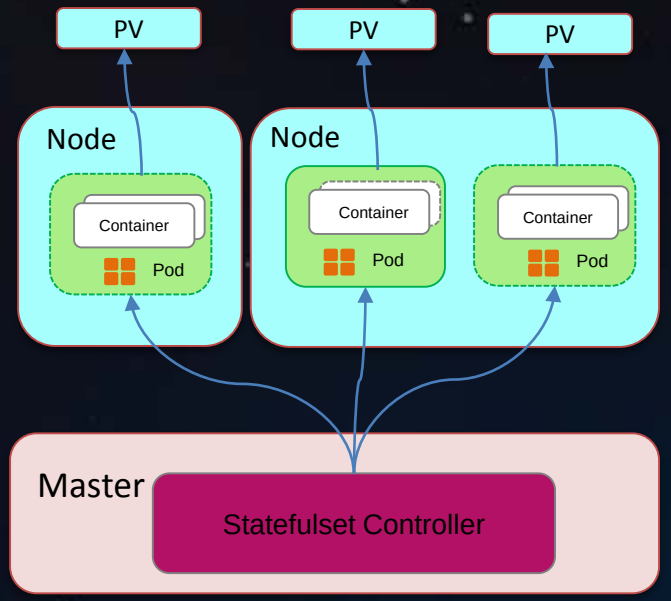
关键工作负载-Deployment



- Kubernetes Deployment提供了官方的用于更新Pod和Replica Set（下一代的Replication Controller）的方法，您可以在Deployment对象中只描述您所期望的理想状态（预期的运行状态），Deployment控制器为您将现在的实际状态转换成您期望的状态;
- Deployment集成了上线部署、滚动升级、创建副本、暂停上线任务，恢复上线任务，回滚到以前某一版本（成功/稳定）的Deployment等功能，在某种程度上，Deployment可以帮我们实现无人值守的上线，大大降低我们的上线过程的复杂沟通、操作风险。
- Deployment的典型用例：
  - 使用Deployment来启动（上线/部署）一个Pod或者ReplicaSet
  - 检查一个Deployment是否成功执行
  - 更新Deployment来重新创建相应的Pods（例如，需要使用一个新的Image）
  - 如果现有的Deployment不稳定，那么回滚到一个早期的稳定的Deployment版本

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

关键工作负载-StatefulSet

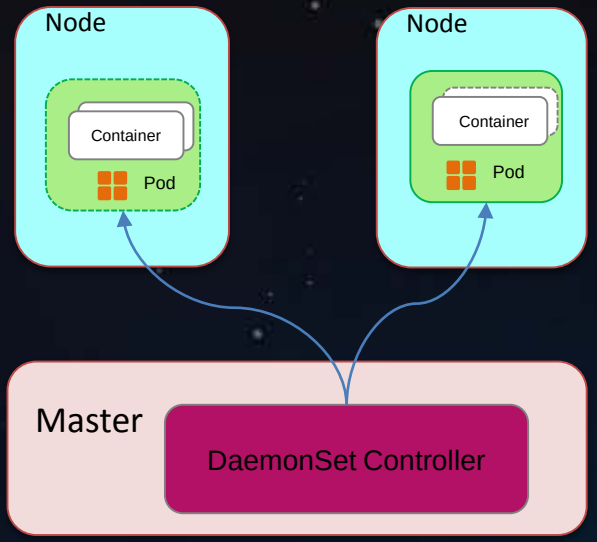



```
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
```

- StatefulSet—有状态应用
- 用于解决各个pod实例独立生命周期管理，提供各个实例的启动顺序和唯一性
  - 稳定，唯一的网络标识符。
  - 稳定，持久存储。
  - 有序的，优雅的部署和扩展。
  - 有序，优雅的删除和终止。
  - 有序的自动滚动更新。

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to
match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to
match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: k8s.gcr.io/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"
```

关键工作负载-DaemonSet




```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: tomcat-ds
spec:
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
      - name: tomcat
        image: tomcat:8.0.30-jre8
        ports:
        - containerPort: 8080
```

- DaemonSet能够让所有（或者一些特定）的Node节点运行同一个pod。当节点加入到kubernetes集群中，pod会被（DaemonSet）调度到该节点上运行，当节点从kubernetes集群中被移除，被（DaemonSet）调度的pod会被移除，如果删除DaemonSet，所有跟这个DaemonSet相关的pods都会被删除。
- 在使用kubernetes来运行应用时，很多时候我们需要在一个区域（zone）或者所有Node上运行同一个守护进程（pod），例如如下场景：
  - 每个Node上运行一个分布式存储的守护进程，例如glusterd，ceph
  - 运行日志采集器在每个Node上，例如fluentd，logstash
  - 运行监控的采集端在每个Node，例如prometheus node exporter，collectd等

关键工作负载-Job

分为普通任务（Job）和定时任务（CronJob）

- 一次性执行，非堵塞
- 适合CI，视频解码，基因测序等业务

```
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
      backoffLimit: 4
```

Kubernetes关键概念-CustomResourceDefinition

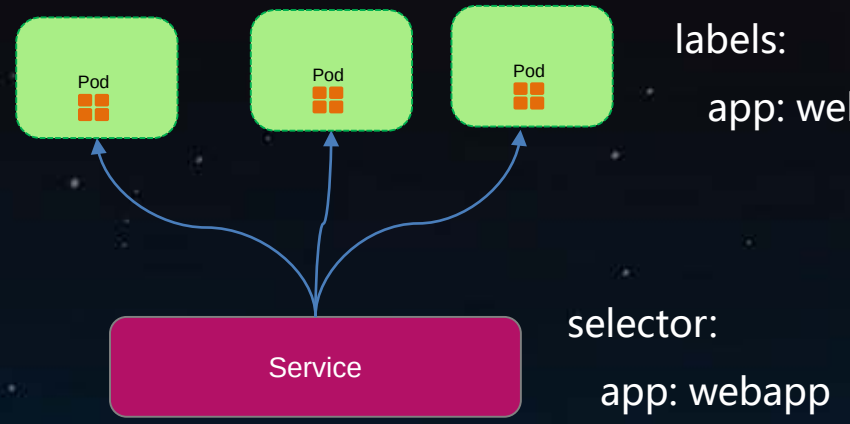
HUAWEI

用户可自定义资源模型，可扩展行强：

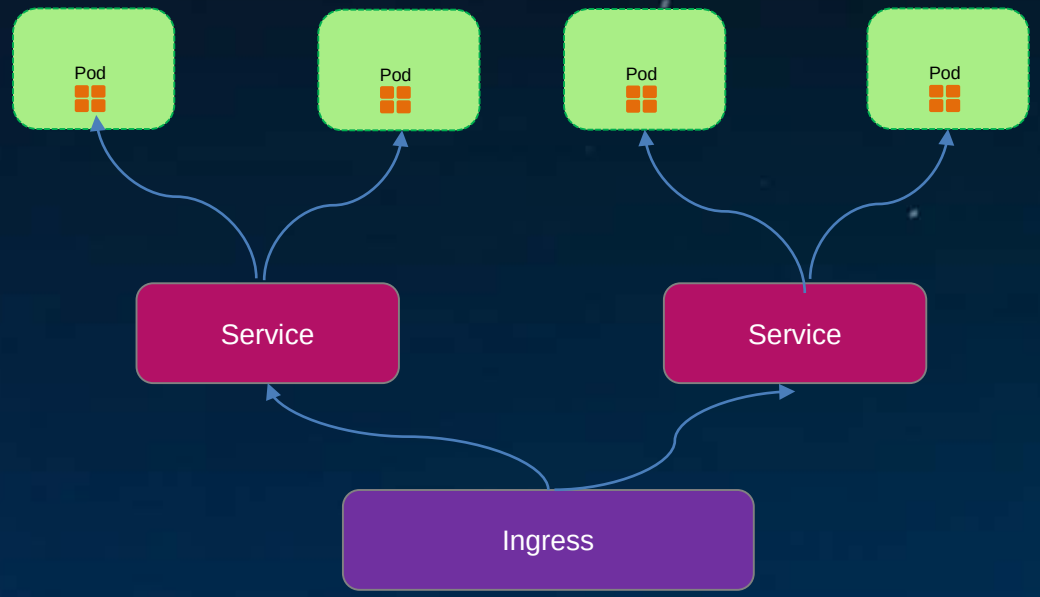
- 需实现自己的controller进行自定义资源的管理

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  # name must match the spec fields below, and be in the form: <plural>.<group>
  name: crontabs.stable.example.com
spec:
  # group name to use for REST API: /apis/<group>/<version>
  group: stable.example.com
  # list of versions supported by this CustomResourceDefinition
  versions:
    - name: v1
      # Each version can be enabled/disabled by Served flag.
      served: true
      # One and only one version must be marked as the storage version.
      storage: true
  # either Namespaced or Cluster
  scope: Namespaced
  names:
    # plural name to be used in the URL: /apis/<group>/<version>/<plural>
    plural: crontabs
    # singular name to be used as an alias on the CLI and for display
    singular: crontab
    # kind is normally the CamelCased singular type. Your resource manifests use this.
    kind: CronTab
    # shortNames allow shorter string to match your resource on the CLI
    shortNames:
    - ct
```

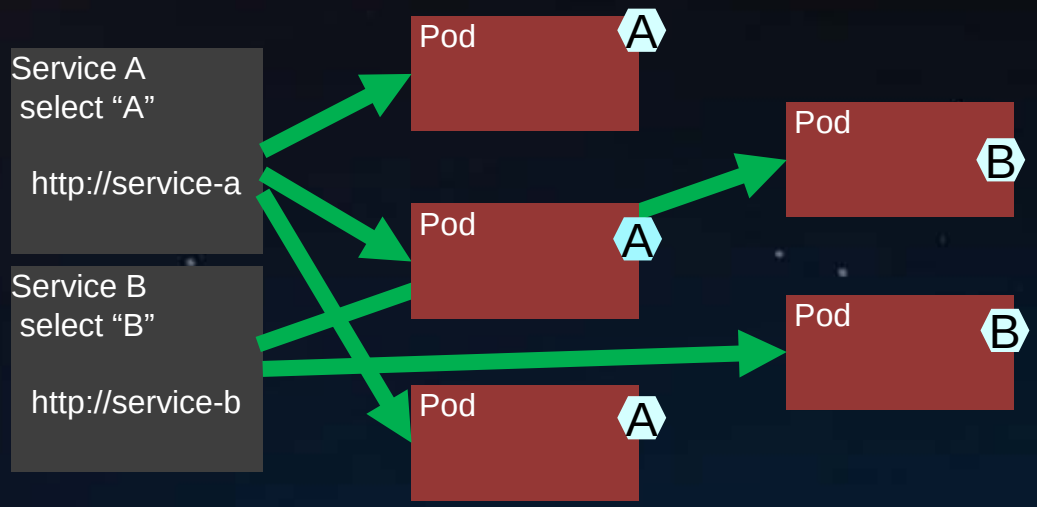
Pod与服务的关系



- 通过label-selector相关联
- 通过服务实现Pod的负载均衡能力：
  - Service：实现TCP/UDP 4层负载均衡能力
  - Ingress：实现HTTP 7层负载均衡能力



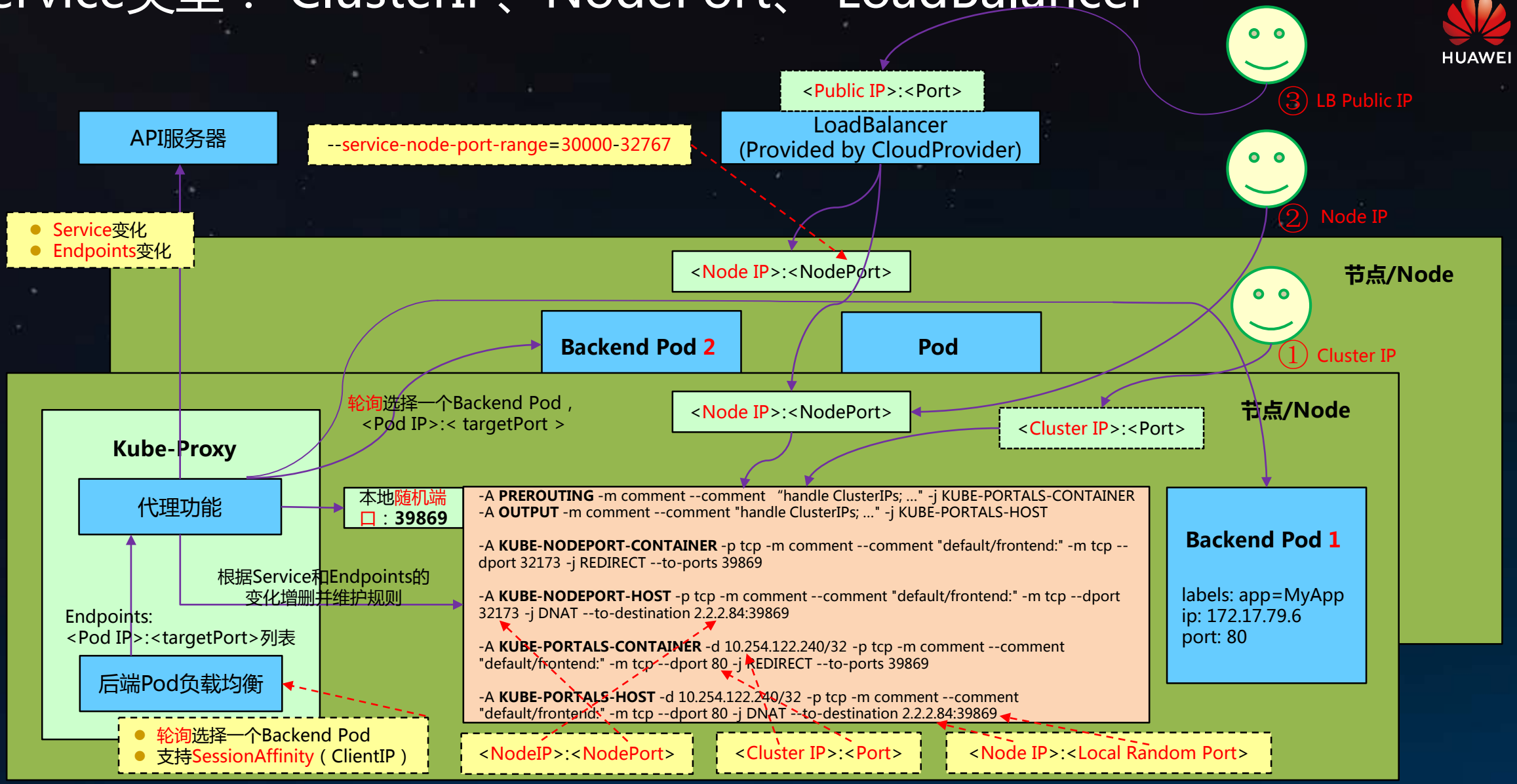
Service



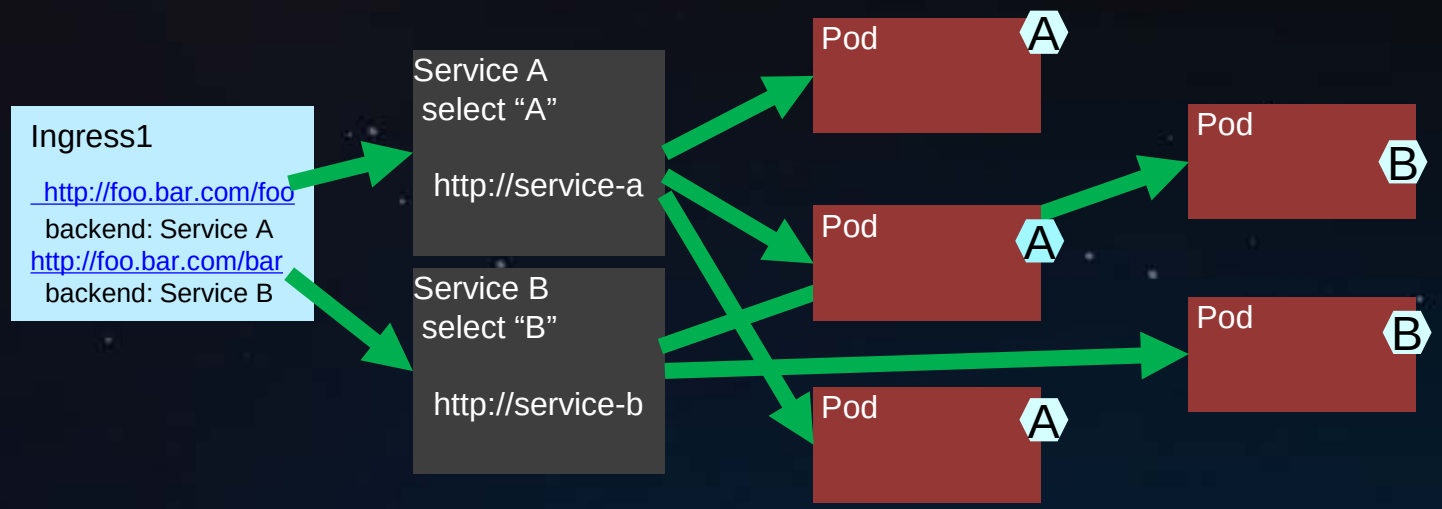

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: A
  ports:
  -  protocol: TCP
     port: 80
     targetPort: 9376
     nodePort: 30061
  type: ClusterIP
```

- Service定义了pods的逻辑集合和访问这个集合的策略。Pods集合是通过定义Service时提供的Label选择器完成的
- Service的引入旨在保证pod的动态变化对访问端透明，访问端只需要知道service的地址，由service来提供代理
- Service的抽象使得前端客户和后端Pods进行了解耦
- 支持ClusterIP，NodePort以及LoadBalancer三种类型
- Service的底层实现有userspace、iptables和ipvs三种模式

Service类型： ClusterIP、NodePort、 LoadBalancer



Ingress




```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: service-a
          servicePort: 80
      - path: /bar
        backend:
          serviceName: service-b
          servicePort: 80
```

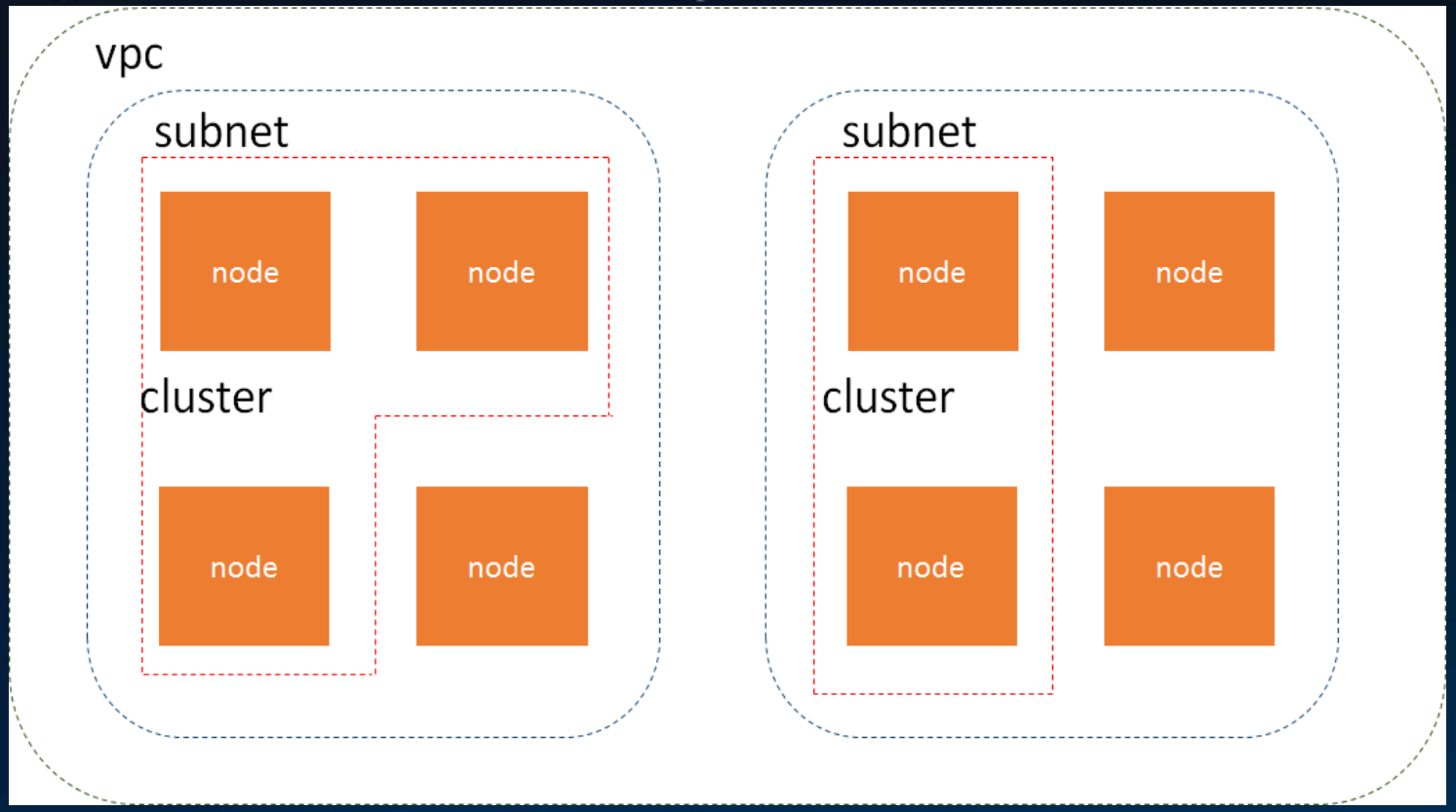
- Ingress基于service实现7层路由转发能力

大 纲

- Kubernetes Pod详解
- Kubernetes工作负载与服务介绍
- **快速部署第一个容器应用**

在云上快速部署一个容器应用

需新建一个k8s私有集群



HUAWEI

Thank You

http://zhibo.huaweicloud.com/watch/2174406

直播 每周四 晚20:00