TRANSWARP 星环科技

# CTA/CTAD系列课程

# Kubernetes基础介绍

星环信息科技（上海）有限公司
www.transwarp.io

2019/8/28

content

# 目录

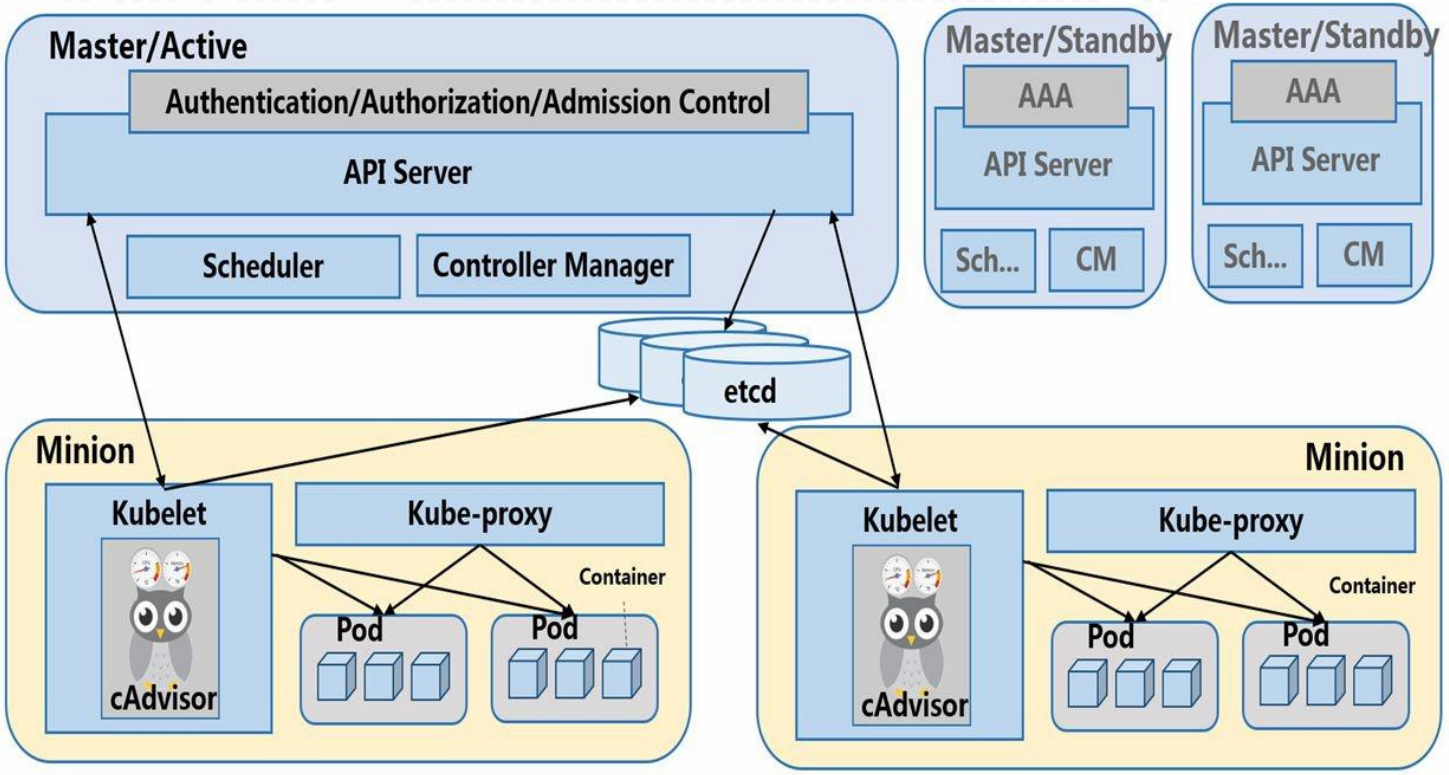
# /01

星环科技 Transwarp

# K8S架构与组件

# Kubernetes架构

- 核心组件
  - **Etcd** （持久化存储）
  - **ApiServer** （**k8s** 访问接口）
  - **Controller-manager** （管理控制）
  - **Scheduler** （调度）
  - **Kube-proxy** （网络转发）
  - **Kubelet** （任务执行）



# Etcd

- **Etcd是CoreOS开源的一个高可用强一致性的分布式存储服务**
- **Kubernetes使用Etcd作为数据存储后端，把需要记录的pod、rc、service等资源信息存储在Etcd中**
- **Etcd使用raft算法将一组主机组成集群，raft 集群中的每个节点都可以根据集群运行的情况在三种状态间切换：follower、candidate与leader。**
- **leader 和follower 之间保持心跳**
  - **如果follower在一段时间内没有收到来自leader的心跳，就会转为candidate，发出新的选主请求。**
  - **当一个节点获得了大于一半节点的投票后会转为leader节点**

# API Server

- API Server提供了k8s各类资源对象（pod,RC,Service等）的增删改查及watch等HTTP Rest接口，是整个系统的数据总线和数据中心。

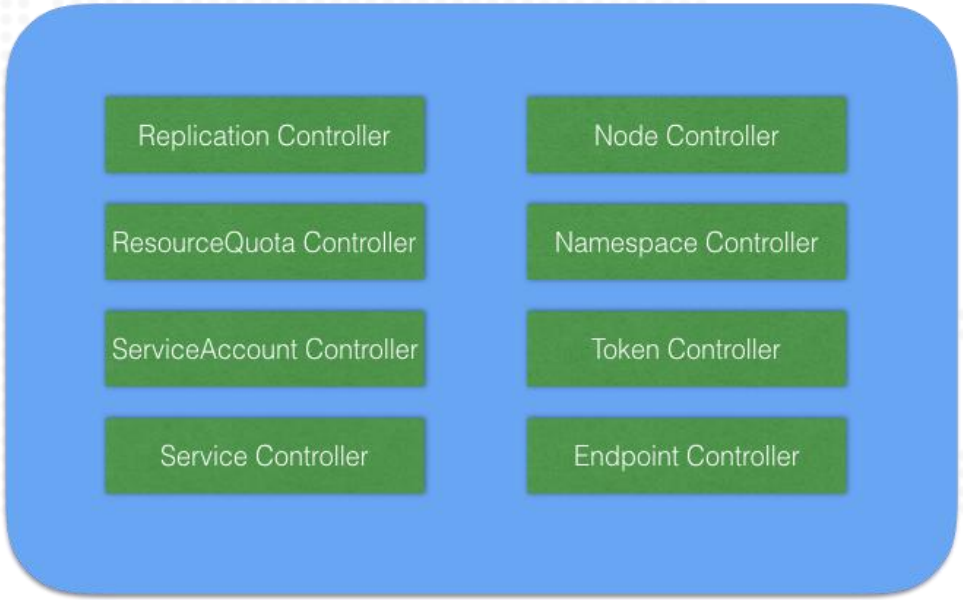
- 在 kubernetes 集群中，API Server 有着非常重要的角色。API Server负责和etcd交互（其他组件不会直接操作etcd，只有 API Server 这么做），是整个 kubernetes 集群的数据中心，所有的交互都是以API Server为核心的。简单来说，API Server 提供了以下功能：

  - 整个集群管理的*API*接口：所有对集群进行的查询和管理都要通过API Server来进行
  - 集群内部各个模块之间通信的枢纽：所有模块之之间并不会之间互相调用，而是通过和 API Server 打交道来完成自己那部分的工作
  - 集群安全控制：API Server 提供的验证和授权保证了整个集群的安全

# Controller Manager

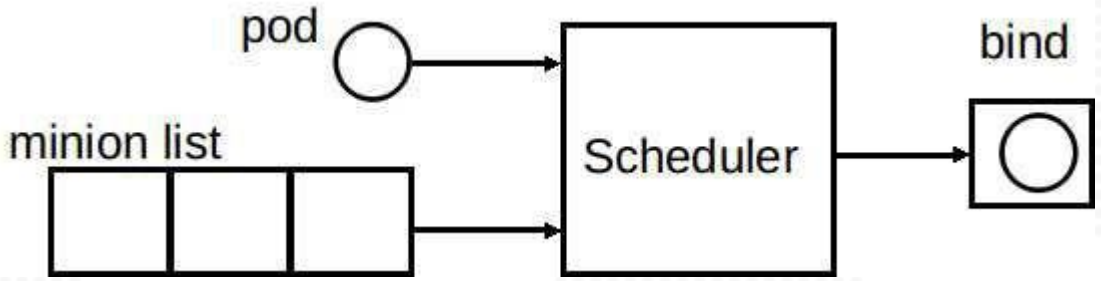
- Controller Manager 是一个集群内部的管理控制中心，有一组控制器构成，这组控制器负责集群内部的 Node、Pod、Endpoint、Namespace、ServiceAccount、ResourceQuota 等等资源的管理。

- 每个Controller通过API Server提供的接口实时监控整个集群的每个资源对象的当前状态，当发生各种故障导致系统状态发生变化时，会尝试将系统状态修 复到“期望状态”



# kube-scheduler

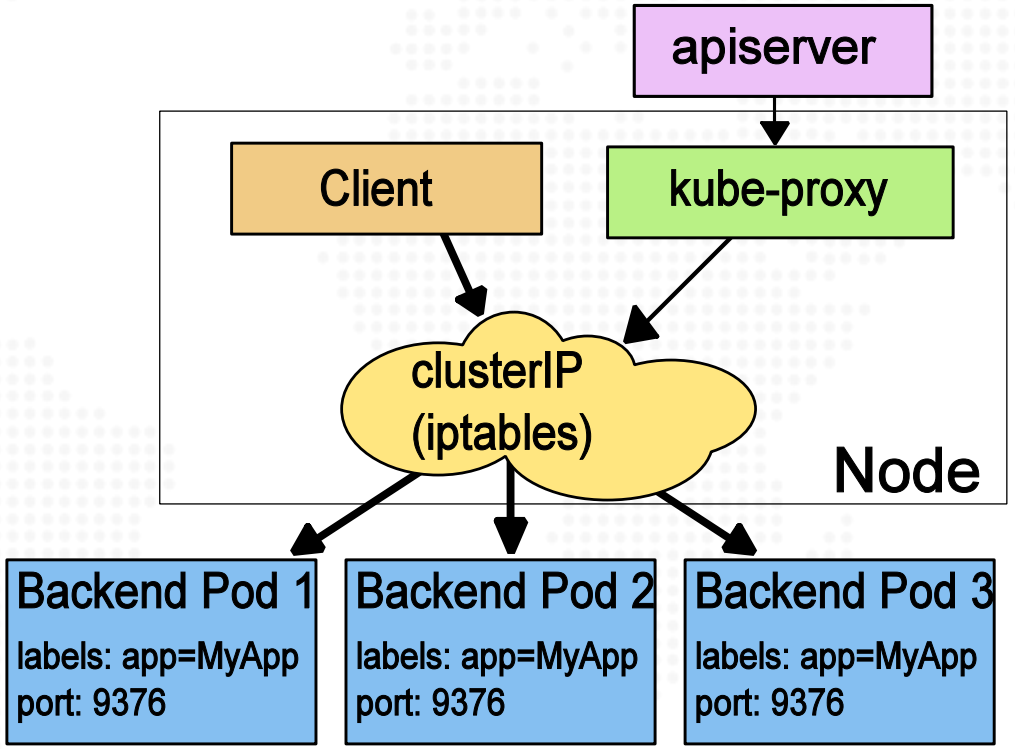
- 负责集群的资源调度，为新建的**pod**分配机器
- 根据特定的调度算法将**pod**调度到指定的工作节点（**Node**）上，这一过程也叫绑定（**bind**）。**Scheduler**的输入为需要调度的**Pod** 和可以被调度的节点**(Node)**的信息，输出为调度算法选择的**Node**，并将该**pod bind**到该**Node**




- 调度过程分为两步， **predicate**以及**prioritize**
  - **predicate**筛选满足条件的**node**，
  - **prioritize**给剩下**node**打分，选择分数最高的**node**，作为**bind**的**node**

# kube-proxy

- kube-proxy负责service的实现，即实现了k8s内部从pod到service和外部从node port到service的访问。



# kubelet

TRANSWARP 星 环 科 技

- 集群中的每个 Node 都有 Kubelet 进程，该进程用于处理 Master 节点下发到本节点的任务，管理 Pod 以及 Pod 中的容器。
  - 节点管理：kubelet 启动时向 API Server 注册节点信息，并定时向 API Server 汇报节点状况；
  - Pod管理：创建/删除 Pod，下载容器镜像，用 Pause 创建容器，运行容器，校验容器是否正确等；
  - 容器健康检查：通过访问容器的 HTTP 接口（HTTP 状态码作为判断依据）来判断容器是否健康；
  - cAdvisor 资源监控：cAdvisor 集成到 kubelet 程序的代码之中，负责查找当前节点的容器，自动采集容器级别的 CPU、内存、文件系统和网络使用的统计信息。

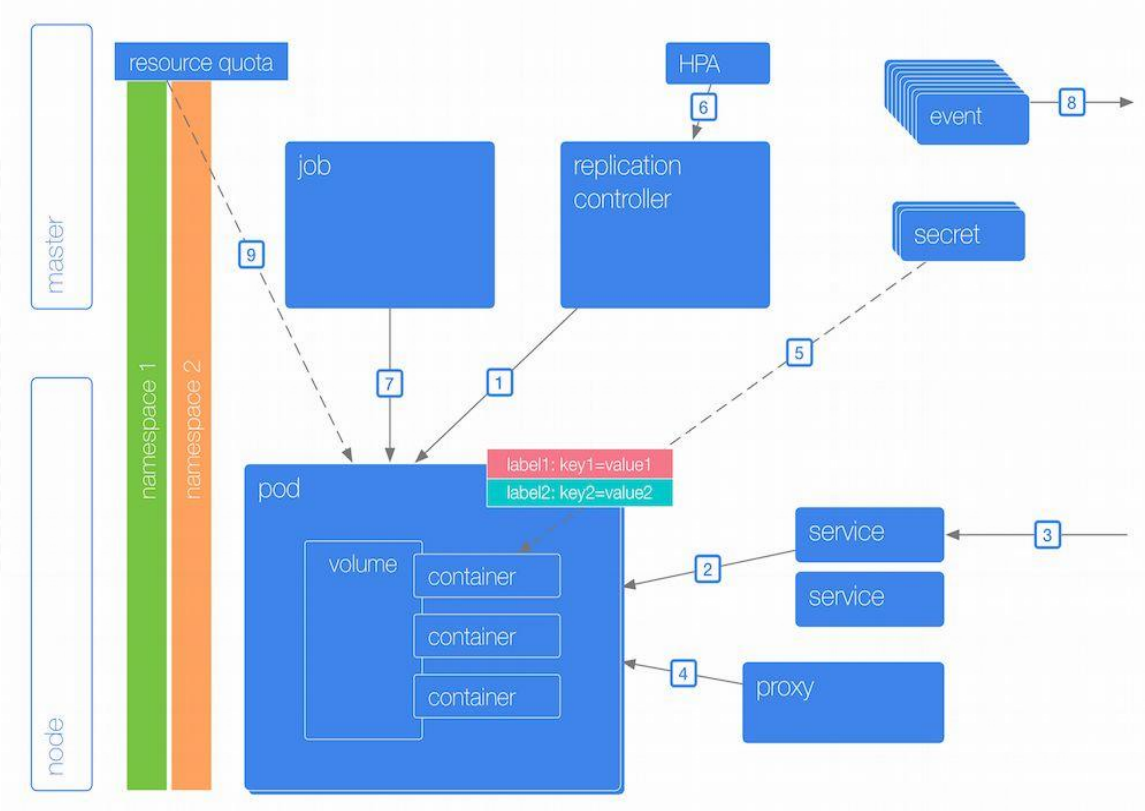
/02

星环科技 Transwarp

# K8S基本原语与操作

# K8S核心对象

- Namespace
- Resource Quota
- Pod
- Deployment
- Service
- Daemonset
- Secret
- ...



# Namespace （命名空间）

TRANSWARP 星 环 科 技

- 通过将系统内部的对象“分配”到不同的Namespace中，形成逻辑上分组的不同项目、小组或用户组，便于不同的分组在共享使用整个集群的资源的同时还能被分别管理。
- 与**Resource Quota**（配额）一起提供多租户管理
- Kubernetes集群在启动后，会创建一个名为“default”的Namespace，如果不特别指明Namespace，则用户创建的Pod、RC、Service都被系统创建到“default”的Namespace中
- **kube-system**是预留的命名空间，系统服务在**kube-system**下运行
- 可以通过配置**RBAC,** 每个用户只能使用自己命名空间下的资源

# Resource Quota （配额）

- 用于限制命名空间下资源的使用
  - 命名空间下如果没有配额， 则视为无限制
  - 可以限制**cpu**，**memory**，存储卷等
  - 目前主要是限制**pod**的创建
- **pod**无法在带**quota**的命名空间下创建
  - 超出配额
  - **pod**中的容器没有资源使用声明

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "4"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.nvidia.com/gpu: 4
```

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-counts
spec:
  hard:
    configmaps: "10"
    persistentvolumeclaims: "4"
    replicationcontrollers: "20"
    secrets: "10"
    services: "10"
    services.loadbalancers: "2"
```

# Pod （容器组）

- **K8S**中最小的调度单位
  - 一个**pod**只会调度到一台机器，不会横跨两台机器
  - 一个**pod**调度完成之后，不会移动到其他机器
- 容器组中包含一个或多个容器
  - 共享网络空间
    - **infra** 容器用于维持容器组**IP**
  - 共享存储卷
- 来源：
  - 用户可以创建**pod**
  - **Job**，**Rs**，**Deployment**，**Statefulset**产生

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dapi-envars-fieldref
  annotations:
    build: two
    builder: john-doe
spec:
  containers:
    - name: test-container
      image: busybox
      command: [ "sh", "-c"]
      args:
      - while true; do
          echo -en '\n';
          printenv MY_NODE_NAME MY_POD_NAME
          printenv MY_ANNOTATIONS
          sleep 10;
        done;
      env:
#         - name: MY_ANNOTATIONS
#           valueFrom:
#             fieldRef:
#               fieldPath: metadata.annotations
        - name: MY_NODE_NAME
          valueFrom:
            fieldRef:
              fieldPath: spec.nodeName
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
  restartPolicy: Never
```

# Label & Annotation（标签&注释）

TRANSWARP 星 环 科 技

- 所有k8s元素共有属性
- **label**
  - **key=value，value是简单字符串**
  - 用于资源选择以及简单的数据存放
    - **kubectl get resource_type -l key=value**
    - **kubectl get resource_type -l environment in (production),tier in (frontend)**
    - **kubectl label resource_type resource_name key=value**
- **annotation**
  - **key=value， value可以是复杂的字符串**
  - 用于存复杂数据
  - 不支持资源选择

# Secret & ConfigMap

- 编码
  - **secret 中的数据用base64编码，而configmap无编码**
- 场景
  - **secret一般用于存放敏感数据，而configmap一般是配置文件**

- 使用**kubectl 创建 secret**
  - ***kubectl create secret generic secret-hello --from-file=abc=hello.txt***

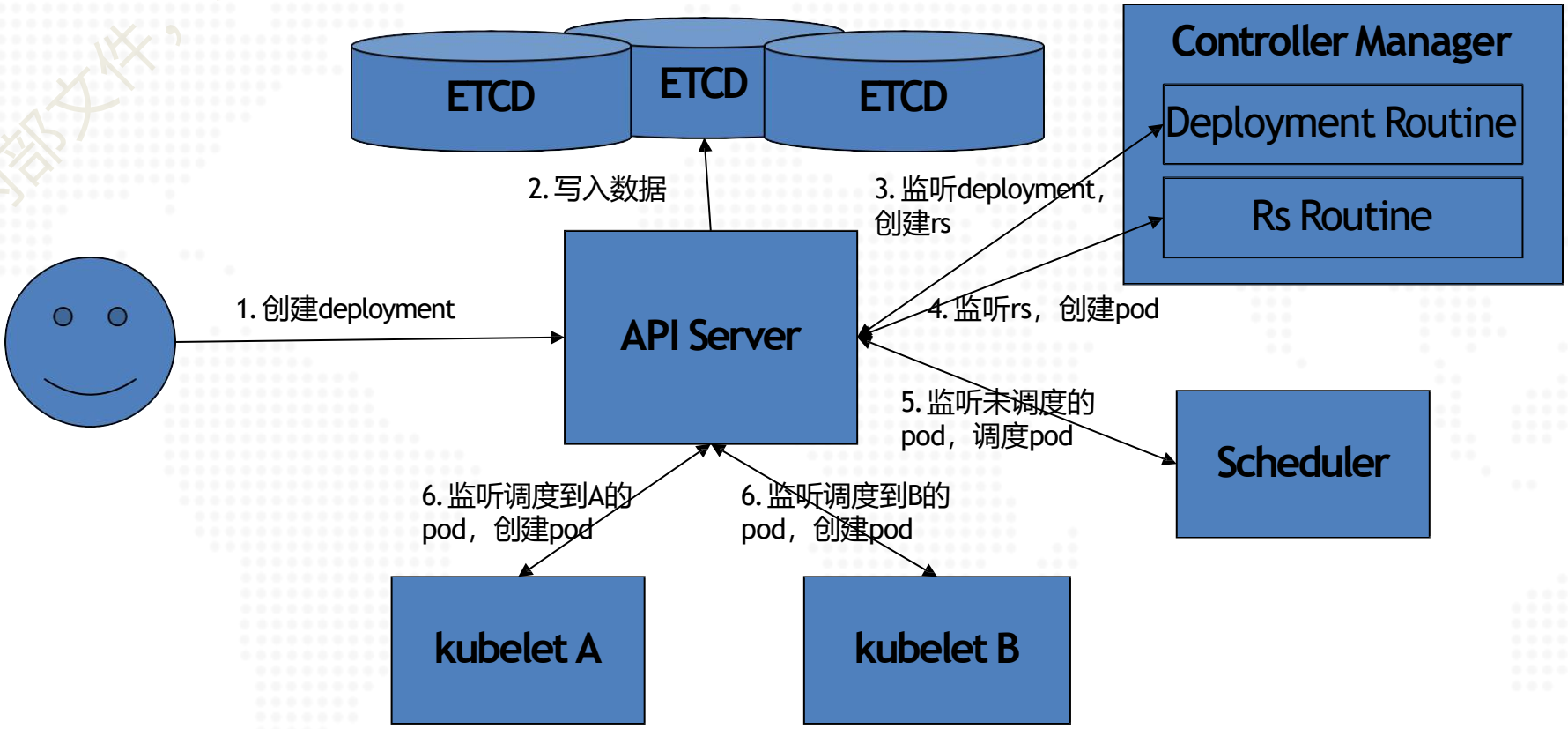
- 使用**kubectl 创建configmap**
  - **kubectl create cm cm-hello --from-file=abc=hello.txt**

# Deployment

- **Deployment** 包含如下：
  - **update strategy** （升级策略，rolling update / recreate）
  - **Pod template** （pod模板，用于创建pod）
  - **replicas** （将要创建的pod数目）
  - **label selector** （如何选中**pod**）
- **Controller-manager**会负责保持**$replica**份满足**label selector**的**pod**
  - 多删，少增
  - 当**pod template** 发生变化，会创建出新的**RS**
- 支持**rolling update**以及**roll back**
  - 当**pod template** 发生变化，会创建新的**RS**
  - 老的**RS**选中的**pod**数目减少，新的**RS**选中的**pod**数目增多

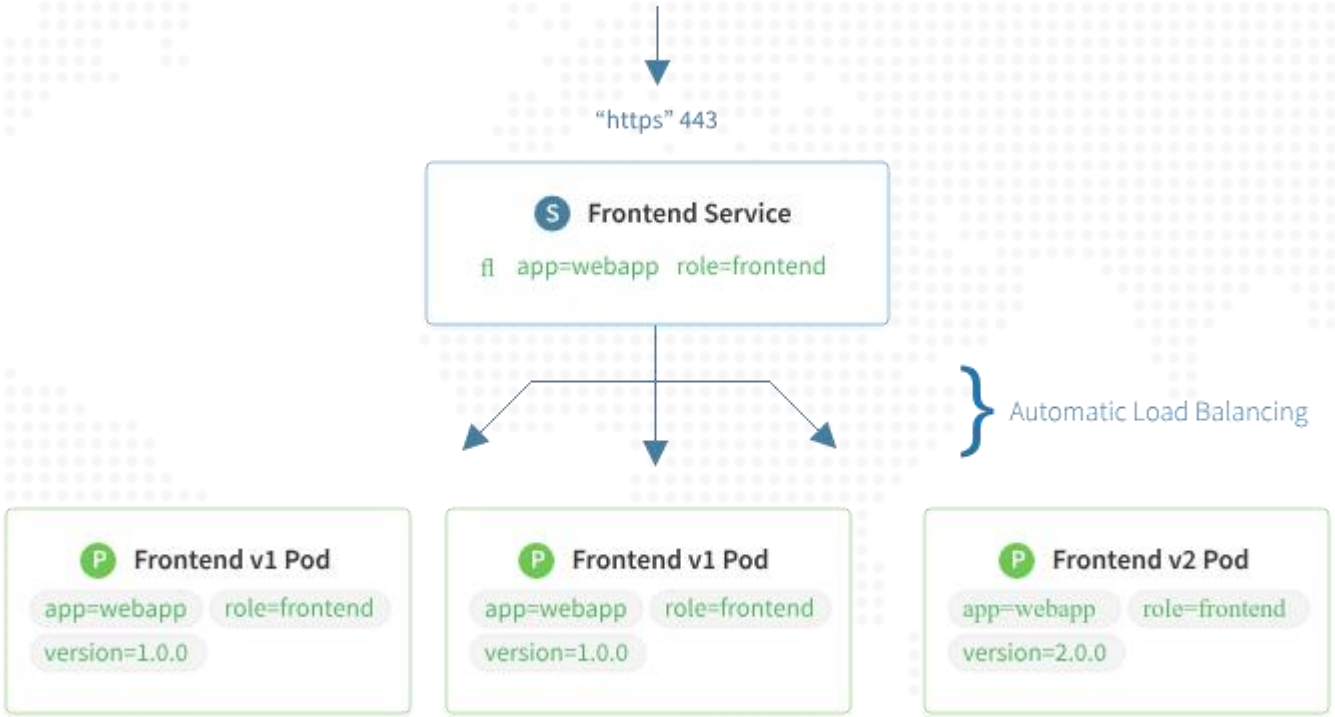
# Deployment

- Deploy workflow



# Service

- 将一组pod抽象出来， 提供访问接口



# Node Port Service

- **kubernetes**从主机上一个区段的端口中分配出端口用来提供对外服务 **(default: 30000-32767)**
- 每个节点都会将对那个端口（每个节点上的端口是一样的）的访问请求，转发到对应的**service**上

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx-pod
spec:
  containers:
  - name: nginx
    image: 172.16.3.220:5000/transwarp/nginx:live
```

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
spec:
 type: NodePort
 ports:
 - port: 30080
   targetPort: 80
   nodePort: 30001
 selector:
  name: nginx-pod
```

# Daemonset

- 每个节点一个， 或者某类节点一个
- 包含如下:
  - **pod template** （pod模板，用于创建pod）
  - **label selector** （如何选中pod）
- **Controller-manager**会根据**Daemonset**的定义筛选出符合条件的节点，在节点上启停**pod**
  - 符合条件创建
  - 不符合删除

# K8S应用管理



- 应用创建
  - 通过yaml文件

    kubectl create -f xx.yaml
  - 通过kubectl命令

    kubectl run demo-backend --image=172.16.1.41:5000/library/example-  guestbook-php-redis:20151205

    kubectl label deployment demo-backend name=demo

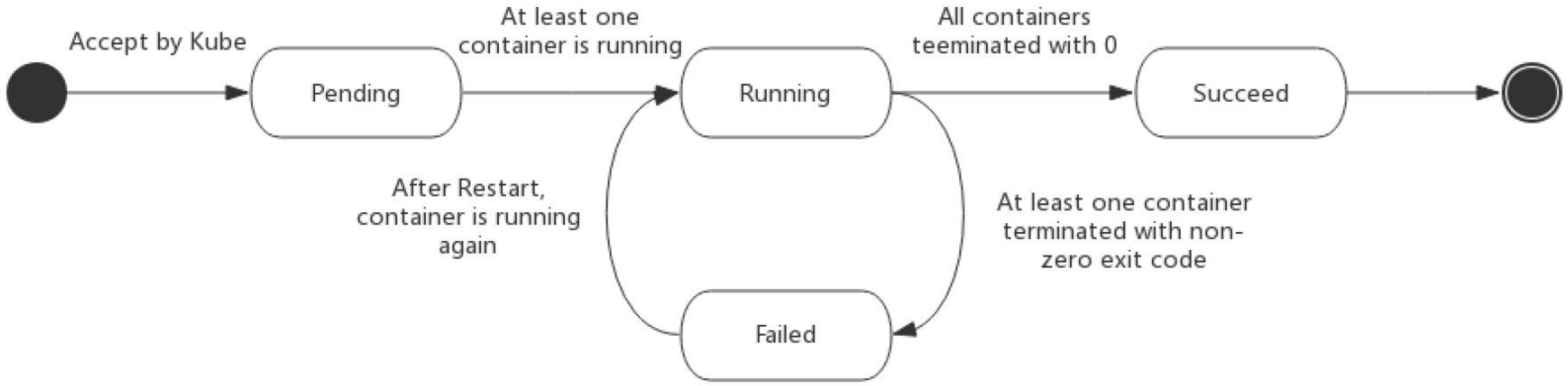
    kubectl annotate deployment demo-backend version="1.0"

    kubectl set env deployment demo-backend java_home=/etc/jdk

- 应用扩展
  - *kubectl scale --replicas=2 deployment backend*
- 应用升级
  - *kubectl set image deployment backend php-redis=172.16.1.41:5000/jenkins/redis:3.0.5*

# Pod生命周期

- PodPhase
  - Pending （pod中至少还有一个容器还没有启动）
  - Running （pod中所有容器都启动了，并且至少一个容器还在运行中）
  - Succeed （pod中所有容器都退出了，并且都成功退出）
  - Failed （pod中所有容器都退出了，并且至少一个容器失败退出）
  - Unknow （无法或者容器状态）





# Thanks

www.transwarp.io



公司地址 / Our Office

上海：徐汇区虹漕路88号B座11F&12F&15F，A座9F
北京：海淀区西直门北大街甲43号金运大厦B座1101室
广州：天河区体育东路140-148号南方证券大厦1015-1016室
郑州：郑东新区龙子湖湖心岛卫华研究院科研楼13层
南京：雨花台区宁双路19号云密城J栋10楼
联系电话：4007-676-098