link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=70 sentences=104, words=642

## 1. Kubernetes介绍 #

- Master是控制节点,负责编排、管理、调度用户提交的作业
    - 负责API服务的 kube-apiserver
    - 负责调度的 kube-scheduler
    - 负责容器编排的 kube-controller-manager
    - kube-apiserver会处理集群的持久化数据并保存在 etcd中
- Node是计算节点
    - CRI(Container Runtime Interface)的远程调用接口，这个接口定义了容器运行时的各项核心操作
    - OCI(Open Container Initiative) 容器运行时通过OCI同底层的Linux操作系统进行交互
    - 设备插件是用来管理宿主机物理设备的组件
    - gRPC (http://doc.oschina.net/grpc)是可以在任何环境中运行的现代开源高性能 RPC 框架
    - RPC是指远程过程调用，也就是说两台服务器A，B，一个应用部署在A服务器上，想要调用B服务器上应用提供的函数/方法，由于不在一个内存空间，不能直接调用，需要通过网络来表达调用的语义和传达调用的数据

### 1.1 Pod #

- Pod 是 K8S 中最小的可调度单元（可操作/可部署单元）
- 它里面可以包含1个或者多个 Docker 容器
- 在 Pod 内的所有 Docker 容器，都会共享同一个网络、存储卷、端口映射规则
- 一个 Pod 拥有一个 IP,但这个 IP 会随着Pod的重启，创建，删除等跟着改变，所以不固定且不完全可靠,这也就是 Pod 的 IP 漂移问题。这个问题我们可以使用下面的 Service 去自动映射
- Pod 是一个容器组，里面有很多容器，容器组内共享资源

### 1.2 deployment #

- 希望批量启动和管理多个Pod实例，就可以使用deployment

### 1.3 Service #

- 有了Pod实例后就需要以固定的IP地址以负载均衡的方式访问多个Pod实例，就有了Service

## 2.部署 #

### 2.1 编写配置文件 #

- Kubernetes 最核心的设计理念就是声明式 API
- 声明式 API可以用来描述容器化业务和容器间关系
- apiversion (https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-apiversion-definition-guide.html)

```
mkdir deployment && cd deployment
vim deployment-user-v1.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: user-v1
spec:
  selector:
    matchLabels:
      app: user-v1
  replicas: 3
  template:
    metadata:
      labels:
        app: user-v1
    spec:
      containers:
      - name: nginx
        image: registry.cn-beijing.aliyuncs.com/zhangrenyang/nginx:user-v1
        ports:
        - containerPort: 80
```

### 2.2 布署Pod #

- kubectl apply 代表准备对资源进行配置
- -f 等于 --filename 后面可以跟随多个配置文件

```
kubectl apply -f deployment-user-v1.yaml
deployment.apps/user-v1 created
```

- 想查看部署完毕后的 Pod 运行状态，当状态都是 Running 时，代表 Pod 运行正常
    - name 是 Pod的名称
    - READY 为容器状态，格式为可用容器/所有容器数量
    - STATUS 为 Pod 的运行状态
    - RESTARTS 为重启数量
    - AGE 为 Pod 运行时间

```
kubectl get pod
NAME                       READY   STATUS    RESTARTS   AGE
user-8445fbf8d7-6f6d7      0/1     ContainerCreating   0          13s
user-8445fbf8d7-nggzv      0/1     ContainerCreating   0          13s
user-8445fbf8d7-xfn52      0/1     ContainerCreating   0          13s
```

### 2.3 Service #

- deployment 是无状态的
- deployment 并不会对 pod 进行网络通信和分发
- Pod 的 IP 在运行时还会经常进行漂移且不固定
- 想访问服务需要使用 Service 组织统一的 Pod 访问入口
- 可以定义Service 来进行统一组织 Pod 服务访问
- 负责自动调度和组织deployment中 Pod 的服务访问，由于自动映射 Pod 的IP，同时也解决了 Pod 的IP漂移问题

#### 2.3.1 配置文件 #

- Kubernetes的三种外部访问方式 (http://www.dockerone.com/article/4884)
- NodePort服务是引导外部流量到你的服务的最原始方式
- NodePort在所有节点上开放一个特定端口,任何发送到该端口的流量都被转发到对应服务

字段 说明 protocol 通信类型（TCP/UDP） targetPort 原本 Pod 开放的端口 port Kubernetes容器之间互相访问的端口 type NodePort，Service的一种访问方式

user-service-v1.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: service-user-v1
spec:
  selector:
    app: user-v1
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: NodePort
```

#### 2.3.2 启动 #

```
kubectl apply -f user-service-v1.yaml
service/service-user-v1 created
```

- 查看当前的服务

```
kubectl get svc
```

可以在任何节点上访问

```
curl http:
curl http:
```

### 2.4 ingress #

- 我们可能会根据请求路径前缀的匹配，权重，甚至根据 cookie/header 的值去访问不同的服务
- 为了达到这种负载均衡的效果，我们可以使用kubernetes的另一个组件 ingress
- ingress-nginx 是基于 nginx 的一个 ingress 实现。
- 可以实现正则匹配路径，流量转发，基于 cookie header 切分流量（灰度发布）

```
#wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.34.1/deploy/static/provider/baremetal/deploy.yaml
wget https://img.zhufengpeixun.com/deploy.yaml
```

vi deploy.yaml d$

```
  namespace: ingress-nginx
spec:
  type: NodePort
  ports:
    - name: http
      port: 80
      protocol: TCP
      targetPort: http
+     nodePort: 31234
    - name: https
      port: 443
      protocol: TCP
      targetPort: https
+     nodePort: 31235

+ image: registry.cn-hangzhou.aliyuncs.com/bin_x/nginx-ingress:v0.34.1@sha256:80359bdf124d49264fabf136d2aecadac729b54f16618162194356d3c78ce2fe
```

- 配置生效，拉取ingress镜像并自动布署ingress

```
kubectl apply -f deploy.yaml
```

- 查看pods的部署状态
    - -n 指定命名空间查询
    - -l 指定label名称查询

```
kubectl -n ingress-nginx get svc
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.108.109.94           80:31234/TCP,443:31235/TCP   16m
ingress-nginx-controller-admission   ClusterIP   10.106.43.59            443/TCP                      16m
```

```
kubectl -n ingress-nginx get svc
NAME                                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)                      AGE
ingress-nginx-controller             NodePort    10.108.109.94           80:31234/TCP,443:31235/TCP   17m
ingress-nginx-controller-admission   ClusterIP   10.106.43.59            443/TCP                      17m
```

- ingress 服务的配置也是使用 yaml 文件进行管理
- annotations (https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/annotations/) 是 ingress 的主要配置项目，可以用来修改这些配置来修改 ingress 的行为。我们可以通过修改这些配置来实现灰度发布，跨域资源，甚至将www.abc.com (http://www.abc.com) 重定向到 abc.com
- rules 是 ingress 配置路径转发规则的地方,当我们去访问 /front 时， ingress 就会帮我们调度到 front-service-v1 这个 service 上面
    - path 可以是一个路径字符串，也可以是一个正则表达式
    - backend 则是 k8s 的 service 服务， serviceName 是服务名称， servicePort 是服务端口
- backend 可以用来给 ingress 设置默认访问的 Service 服务。当请求不匹配 rules 中任何一条规则时，则会去走 backend 中的配置

vi ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: nginx-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - http:
      paths:
       - path: /user
         backend:
          serviceName: user-service-v1
          servicePort: 80
       - path: /pay
         backend:
          serviceName: pay-service-v1
          servicePort: 80
  backend:
     serviceName: user-service-v1
     servicePort: 80
```

```
kubectl apply -f ./ingress.yaml
```

```
curl http:
curl http:
curl http:
curl http:
```

```
kubectl  describe   ingress
```

## 3.参考 #

### 3.1 查看 #

```
kubectl get deploy
//删除deploy 删除后ReplicateSet和pod也没有了
kubectl delete deploy nginx

//查看Replication Controller
kubectl get rc
//删除Replication Controller,删除后Pod也没有了
kubectl delete rc mysql

//查看pod
kubectl get pod
//删除pod
kubectl delete pod mysql-77w7z

//查看服务
kubectl get svc
//删除服务
kubectl delete service nginx

//查看pod详情
kubectl describe pod fail-1034443984-jerry
```

### 3.2 发布镜像 #

```
sudo docker login --username=hongqishiq@126.com registry.cn-beijing.aliyuncs.com

docker pull registry.cn-beijing.aliyuncs.com/zhangrenyang/zhangrenyang:[镜像版本号]

docker login --username=hongqishiq@126.com registry.cn-beijing.aliyuncs.com
docker tag [ImageId] registry.cn-beijing.aliyuncs.com/zhangrenyang/zhangrenyang:[镜像版本号]
docker push registry.cn-beijing.aliyuncs.com/zhangrenyang/zhangrenyang:[镜像版本号]
```

```
docker login --username=hongqishiq@126.com registry.cn-beijing.aliyuncs.com
docker run -d -p 8080:80 nginx
docker exec -it 6764db063e37  bash
/usr/share/nginx/html
docker container commit -m"nginx-user-v1" -a"zhangrenyang" 6764db063e37  registry.cn-beijing.aliyuncs.com/zhangrenyang/nginx:user-v1
docker container commit -m"nginx-user-v2" -a"zhangrenyang" 6764db063e37  registry.cn-beijing.aliyuncs.com/zhangrenyang/nginx:user-v2
docker container commit -m"nginx-pay-v1" -a"zhangrenyang" 6764db063e37  registry.cn-beijing.aliyuncs.com/zhangrenyang/nginx:pay-v1
docker push registry.cn-beijing.aliyuncs.com/zhangrenyang/http-probe:1.0.0
```

### 3.3 链接 #

- Kubernetes中文社区 (http://docs.kubernetes.org.cn/)
- kubectl Cheat Sheet (http://docs.kubernetes.org.cn/783.html#i)
- kubernetes.cheatsheet (https://kubernetes.io/docs/reference/kubectl/cheatsheet/)
- namespaces (https://kubernetes.io/zh/docs/concepts/overview/working-with-objects/namespaces/)Kubernetes 支持多个虚拟集群，它们底层依赖于同一个物理集群。 这些虚拟集群被称为命名空间

```
kubectl get ns
```