## 开源容器应用自动化部署技术Kubernetes

主讲：Cat 老师

北京动力节点教育科技有限公司  
2020 • 北京

动力节点•版权所有•禁止传播

## 快速回顾

What is Kubernetes？

Kubernetes管理员认证（CKA）

Kubernetes整体架构

Kubernetes环境搭建方式(minikube、kind、kubeadm、二进制包、yum安装、第三方工具、花钱购买)

Kubeadm部署Kubernetes环境

Kubernetes部署容器化应用（Nginx、Tomcat、SpringBoot）

Kubernetes部署容器化应用（Yaml文件方式部署、命令方式部署）

Kubernetes Dashbaord UI管理后台部署

## Ingress暴露应用

### **NodePort**

NodePort服务是让外部请求直接访问服务的最原始方式，NodePort是在所有的节点（虚拟机）上开放指定的端口，所有发送到这个端口的请求都会直接转发到服务中的pod里；

NodePort服务的YAML文件如下：

apiVersion: v1

kind: Service

metadata:

name: my-nodeport-service

selector:

app: my-appspec:

**type: NodePort**

ports:

- name: http

  port: 80

  targetPort: 80

**nodePort: 30008**

  protocol: TCP

这种方式有一个“nodePort"的端口，能在节点上指定开放哪个端口，如果没有指定端口，它会选择一个随机端口，大多数时候应该让Kubernetes随机选择端口；

这种方式的不足：

1、一个端口只能供一个服务使用；

2、只能使用30000–32767之间的端口；

3、如果节点/虚拟机的IP地址发生变化，需要人工进行处理；

因此，在生产环境不推荐使用这种方式来直接发布服务，如果不要求运行的服务实时可用，或者用于演示或者临时运行一个应用可以用这种方式；

**三种端口说明**

ports:

- name: http

  port: 80

  targetPort: 80

**nodePort: 30008**

  protocol: TCP

**nodePort**

外部机器（在windows浏览器）可以访问的端口；

比如一个Web应用需要被其他用户访问，那么需要配置type=NodePort，而且配置nodePort=30001，那么其他机器就可以通过浏览器访问scheme://node:30001访问到该服务；

**targetPort**

容器的端口，与制作容器时暴露的端口一致（Dockerfile中EXPOSE），例如docker.io官方的nginx暴露的是80端口；

**port**

Kubernetes集群中的各个服务之间访问的端口，虽然mysql容器暴露了3306端口，但外部机器不能访问到mysql服务，因为他没有配置NodePort类型，该3306端口是集群内其他容器需要通过3306端口访问该服务；

kubectl expose deployment springboot-k8s --port=8080 --target-port=8080 --type=NodePort

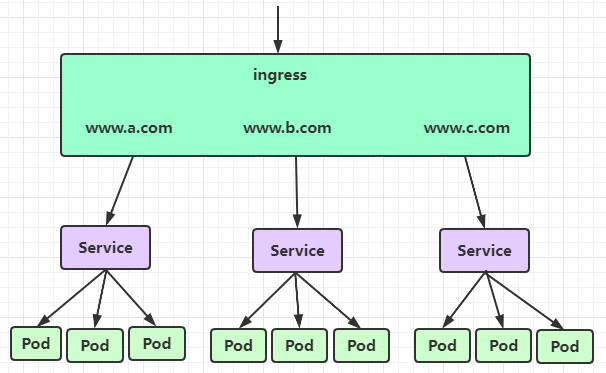
### LoadBalancer

LoadBlancer可以暴露服务，这种方式需要向云平台申请负载均衡器，目前很多云平台都支持，但是这种方式深度耦合了云平台；（相当于是购买服务）

从外部的访问通过负载均衡器LoadBlancer转发到后端的Pod，具体如何实现要看云提供商；

### **Ingress**

Ingress 英文翻译为：入口、进入、进入权、进食，也就是入口，即外部请求进入k8s集群必经之口，我们看一张图：



虽然k8s集群内部署的pod、service都有自己的IP，但是却无法提供外网访问，以前我们可以通过监听NodePort的方式暴露服务，但是这种方式并不灵活，生产环境也不建议使用；

Ingresss是k8s集群中的一个API资源对象，相当于一个集群网关，我们可以自定义路由规则来转发、管理、暴露服务(一组pod)，比较灵活，生产环境建议使用这种方式；

Ingress不是kubernetes内置的（安装好k8s之后，并没有安装ingress），ingress需要单独安装，而且有多种类型Google Cloud Load Balancer，[Nginx](https://www.baidu.com/s?wd=Nginx&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)，Contour，Istio等等，我们这里选择官方维护的Ingress Nginx；

**使用Ingress Nginx的步骤：**

1. 部署Ingress Nginx；
2. 配置Ingress Nginx规则；

### 采用Ingress暴露容器化应用（Nginx）

1. 部署一个容器化应用（pod），比如Nginx、SpringBoot程序；

kubectl create deployment nginx --image=nginx

1. 暴露该服务；

kubectl expose deployment nginx --port=80 --target-port=80 --type=NodePort

1. 部署Ingress Nginx

<https://github.com/kubernetes/ingress-nginx>

ingress-nginx是使用NGINX作为反向代理和负载均衡器的Kubernetes的Ingress控制器；

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.41.2/deploy/static/provider/baremetal/deploy.yaml>

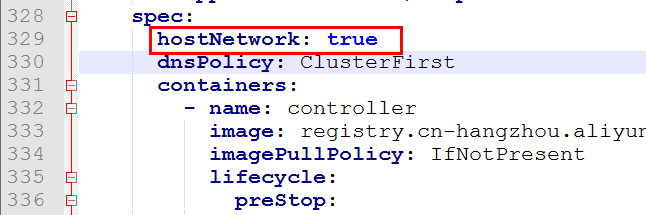
332行修改成阿里云镜像：

阿里云镜像首页：<http://dev.aliyun.com/>

修改镜像地址为：

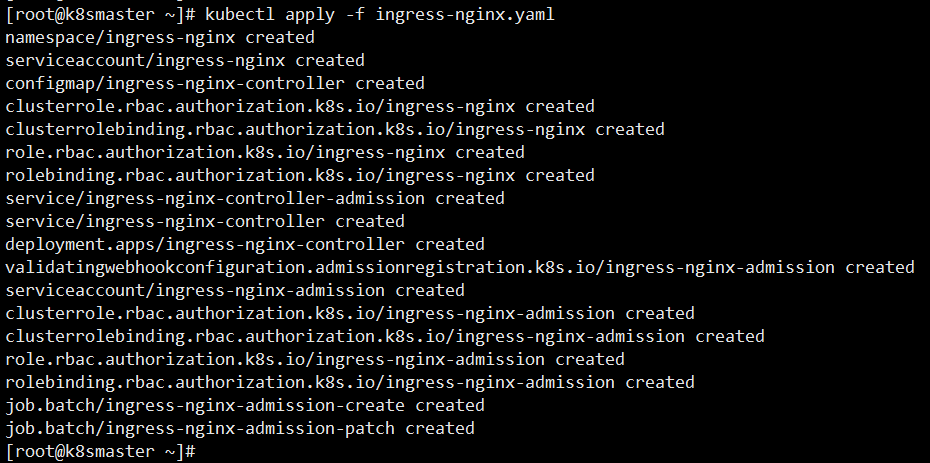
registry.cn-hangzhou.aliyuncs.com/google\_containers/nginx-ingress-controller:0.33.0

添加一个配置项：



应用：

kubectl apply -f deploy.yaml

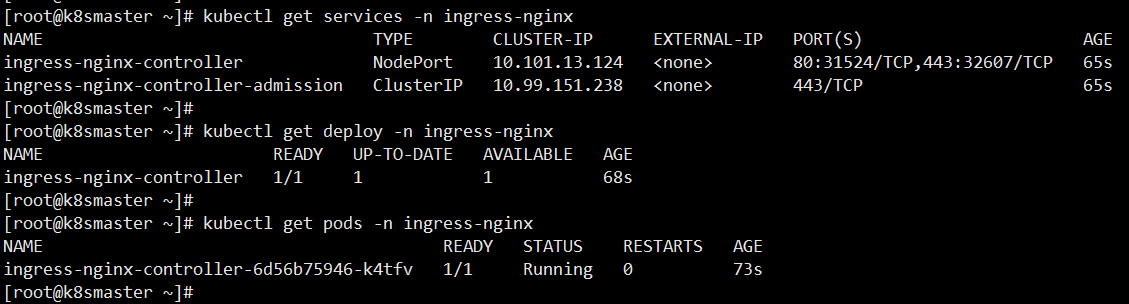


1. 查看Ingress的状态

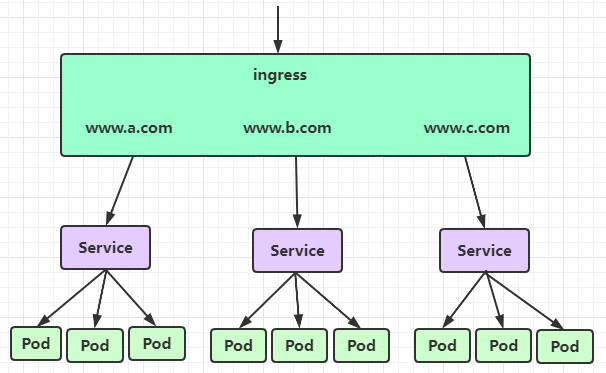
kubectl get service -n ingress-nginx

kubectl get deploy -n ingress-nginx

kubectl get pods -n ingress-nginx

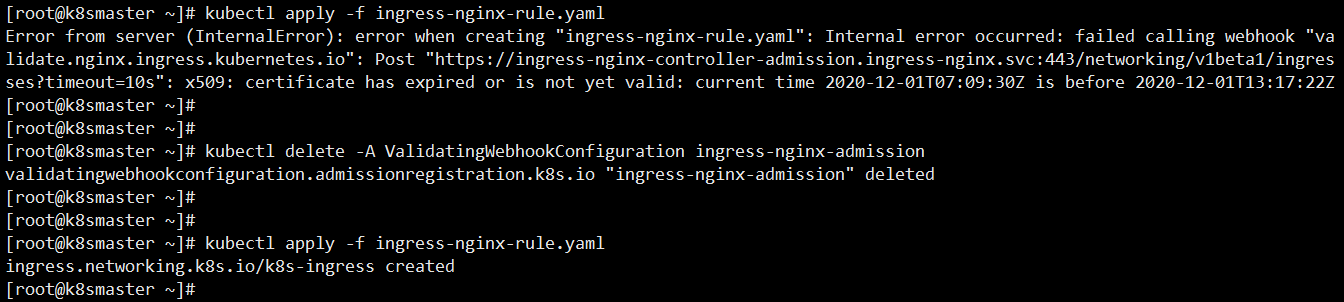


1. 创建Ingress规则



kubectl apply -f ingress-nginx-rule.yaml

报如下错误：



解决：kubectl delete -A ValidatingWebhookConfiguration ingress-nginx-admission

然后再次执行：kubectl apply -f ingress-nginx-rule.yaml

检查一下：

kubectl get ing(ress) --查规则

## kubernetes部署Spring Cloud微服务

1. 项目本身打成jar包或者war包；
2. 制作项目镜像（写Dockerfile文件）；
3. 用k8s部署镜像（命令方式、yaml方式）；
4. 对外暴露服务；

生成镜像：

docker build -t spring-cloud-alibaba-consumer -f Dockerfile-consumer .

docker build -t spring-cloud-alibaba-provider -f Dockerfile-provider .

docker build -t spring-cloud-alibaba-gateway -f Dockerfile-gateway .

部署提供者：

（1）

kubectl create deployment spring-cloud-alibaba-provider --image=spring-cloud-alibaba-provider --dry-run -o yaml > provider.yaml

（2）

kubectl apply -f provider.yaml

部署消费者：

（1）

kubectl create deployment spring-cloud-alibaba-consumer --image=spring-cloud-alibaba-consumer --dry-run -o yaml > consumer.yaml

（2）

kubectl apply -f consumer.yaml

（3）

kubectl expose deployment spring-cloud-alibaba-consumer --port=9090 --target-port=9090 --type=NodePort

部署网关：

（1）

kubectl create deployment spring-cloud-alibaba-gateway --image=spring-cloud-alibaba-gateway --dry-run -o yaml > gateway.yaml

（2）

kubectl apply -f gateway.yaml

（3）

kubectl expose deployment spring-cloud-alibaba-gateway --port=80 --target-port=80 --type=NodePort

在网关上面部署ingress，统一入口；

按照上面讲解ingress暴露nginx的步骤进行就可以；

kubectl get service -n ingress-nginx

kubectl get deploy -n ingress-nginx

kubectl get pods -n ingress-nginx

注意：deploy.yaml文件里面镜像从本地拉取；

containers:

- image: 38-springboot-k8s-1.0.0-jar

name: 38-springboot-k8s-1-0-0-jar-8ntrx

imagePullPolicy: Never

把镜像拉取策略改为Never；

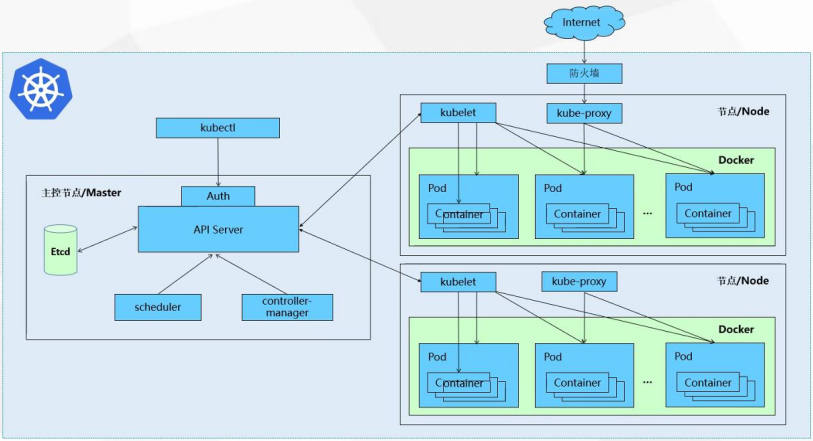
看pod详情：

kubectl describe pods spring-cloud-alibaba-consumer-5d557f4765-d27d9

看pod运行日志：

kubectl logs -f spring-cloud-alibaba-consumer-8697896544-g6rsf

## Kubernetes架构及和核心组件



API server是所有请求的唯一入口；

api server管理所有的事务，并把信息记录到etcd数据库中，etcd有一个自动服务发现的特性机制，etcd会搭建有三个节点的集群，实现三副本；

scheduler 调度器用来调度资源，查看业务节点的资源情况，确定在哪个node上创建pod，把指令告知给api server；

控制管理器controller-manager管理pod；

pod可以分为有状态和无状态的pod，一个pod里最好只放一个容器；

api server把任务下发给业务节点的kubelet去执行；

客户访问通过kube-proxy去访问pod；

pod下面的不一定是docker，还有别的容器；

一般pod中只包含一个容器，除了一种情况除外，那就是elk，elk会在pod内多放一个logstash去收集日志；

**kubernetes各个组件及功能**

### 1、master组件

### kube-apiserver

kubernetes api，集群的统一入口，各组件之间的协调者，以restful api提供接口服务，所有对象资源的增删改查和监听操作都交给apiserver处理后在提交给etcd存储记录；

### kube-controller-manager

处理集群中常规的后台任务，一种资源对应一个控制器，controller-manager就是负责管理这些控制器的；

### kube-scheduler

根据调度算法为新创建的pod选择一个node节点，可以任意部署，可以部署在同一个节点上，也可以部署在不同节点上；

### etcd

分布式键值存储系统，用户保存集群状态数据，比如pod、service等对象信息；

### 2、node组件

### kubelet

kubelet时master在node节点上的代理agent，管理本node运行容器的生命周期，比如创建容器、pod挂载数据卷、下载sercet、获取容器和节点状态等工作，kubelet将每个pod转换成一组容器；

### kube-proxy

在node节点上实现pod的网络代理，维护网络规则和四层的负载均衡工作；

### docker

容器引擎，运行容器；

### kubernetes核心概念

### pod

最小部署单元；

一组容器的集合；

一个pod中的容器共享网络命名空间；

pod是短暂的；

### controllers

replicaset：确保预期的pod副本数量；

deployment：无状态应用部署，比如nginx、apache，一定程度上的增减不会影响客户体验；

statefulset：有状态应用部署，是独一无二型的，会影响到客户的体验；

daemonset：确保所有node运行同一个pod，确保pod在统一命名空间；

job：一次性任务；

cronjob：定时任务；

### service

防止pod失联；

定义一组pod的访问策略；

确保了每个pod的独立性和安全性；

### storage

volumes

persistent volumes

### pollcies策略

resource quotas

### 其他

label：标签，附加到某个资源上，用户关联对象、查询和筛选；

namespaces：命名空间，将对象从逻辑上隔离；

annotations：注释；

Kubectl：k8s提供的终端控制命令；

Kubeadm：可以用来初始化或加入一个k8s集群；