用k8s部署微服务应用

以我们之前用docker部署过的eureka应用为例，首先添加配置文件eureka-app-deployment.yaml用于**创建Deployment**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eureka-app-deployment
  labels:
    app: eureka-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: eureka-app
  template:
    metadata:
      labels:
        app: eureka-app
    spec:
      containers:
        - name: eureka-app
          # 指定Docker Hub中的镜像地址
          image: zhuge666/microservice-eureka-server:0.0.1
          ports:
            - containerPort: 8761
```

通过应用配置文件来创建Deployment

```
kubectl apply -f eureka-app-deployment.yaml
```

查看下deploy和pod信息

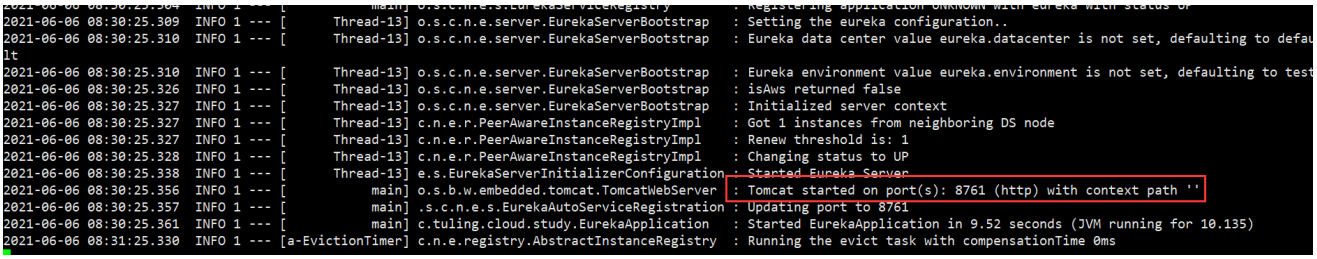
```
kubectl get deploy,pod
```

```
[root@k8s-master k8s]# kubectl get deploy,pod
NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eureka-app-deployment   1/1     1            1           2m17s
deployment.apps/my-tomcat               3/3     3            3           10d
deployment.apps/my-tomcat-yaml          2/2     2            2           8d
deployment.apps/nginx                   1/1     1            1           11d
deployment.apps/nginx-deployment        1/1     1            1           2d17h

NAME                                        READY   STATUS    RESTARTS   AGE
pod/eureka-app-deployment-7cd4b6f4d4-lkzs4  1/1     Running   0          2m17s
pod/my-tomcat-685b8fd9c9-4ngsb              1/1     Running   0          10d
pod/my-tomcat-685b8fd9c9-lrwst              1/1     Running   0          10d
pod/my-tomcat-685b8fd9c9-q6xzh              1/1     Running   0          10d
pod/my-tomcat-yaml-685b8fd9c9-8glxt         1/1     Running   0          8d
pod/my-tomcat-yaml-685b8fd9c9-ltbbf         1/1     Running   0          8d
pod/nginx-deployment-7cf97748c4-7gg28       1/1     Running   0          2d17h
pod/nginx-f89759699-ngqjl                   1/1     Running   0          11d
```

我们可以通过kubectl logs命令来查看应用的启动日志

```
kubectl logs -f pod/eureka-app-deployment-7cd4b6f4d4-lkzs4
```



如果想要从外部访问应用，需要**创建Service**，添加配置文件eureka-app-service.yaml用于创建Service；

```yaml
apiVersion: v1
kind: Service
metadata:
 name: eureka-app-service
spec:
 type: NodePort
 selector:
 app: eureka-app
 ports:
 - name: http
 protocol: TCP
 port: 8761 #service的端口
 targetPort: 8761 #pod的端口，一般与pod内部容器的服务端口一致
```

通过应用配置文件来创建Service

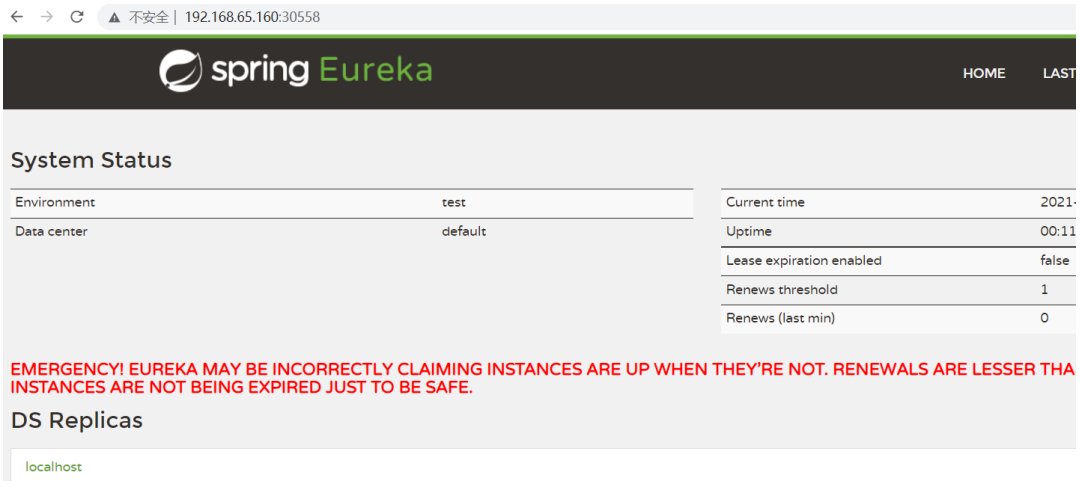
```
kubectl apply -f eureka-app-service.yaml
```

查看服务Service信息

```
[macro@linux-local k8s]$ kubectl get services
```

```
[root@k8s-master k8s]# kubectl get services
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
eureka-app-service  NodePort    10.106.87.91     <none>        8761:30558/TCP               58s
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP                      17d
nginx               NodePort    10.109.128.56    <none>        80:30433/TCP                 11d
test-service        NodePort    10.104.189.121   <none>        80:32080/TCP,8080:32088/TCP  8d
tomcat              NodePort    10.101.176.202   <none>        8080:32224/TCP               10d
tomcat-service-yaml NodePort    10.104.55.220    <none>        8080:31524/TCP               8d
```

此时就可以通过外网来访问了：http://192.168.65.160:30558/



用k8s部署电商项目微服务

以**product服务**为例，我们来创建对应的**deployment和service**，做之前需要把商品服务做成镜像推到docker镜像仓库里去，参考docker的第一节课。

```
docker login
docker tag mall/tulingmall-product:0.0.1 zhuge666/tulingmall-product:0.0.1
docker push zhuge666/tulingmall-product:0.0.1
```

新增文件tulingmall-product-deployment.yaml，内容如下：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tulingmall-product-deployment
  labels:
    app: tulingmall-product
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tulingmall-product
 template:
    metadata:
      labels:
        app: tulingmall-product
    spec:
      containers:
        - name: tulingmall-product
          image: zhuge666/tulingmall-product:0.0.1
  #imagePullPolicy: Always # 1)Always 总是拉取镜像， 2)IfNotPresent(默认该值) 本地有则使用本地镜像，3)Never 只使用本地镜像，从不拉取，即使本地没有镜像
          ports:
            - containerPort: 8866
          env:
  - name: TZ
  value: Asia/Shanghai
  - name: spring.cloud.nacos.config.server-addr
  value: 192.168.65.42:8848
            - name: LOG_FILE
              value: /var/logs
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-product/logs
            type: DirectoryOrCreate
  dnsPolicy: Default #继承Pod所在宿主机的DNS设置，使pod能访问外网
```

执行如下命令创建商品服务的deployment：

```
kubectl apply -f tulingmall-product-deployment.yaml
```

新增文件tulingmall-product-service.yaml，内容如下：

```
apiVersion: v1
kind: Service
metadata:
 name: tulingmall-product-service
spec:
 type: NodePort
 selector:
 app: tulingmall-product
 ports:
 - name: http
 protocol: TCP
 port: 8866
 targetPort: 8866
```

执行如下命令创建商品服务的service：

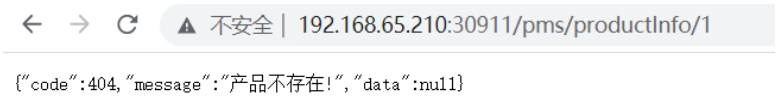
```
kubectl apply -f tulingmall-product-service.yaml
```

查看商品服务的对外暴露端口：

```
[root@k8s-master tuling-mall]# kubectl get svc|grep product
tulingmall-product-service   NodePort    10.106.36.91     <none>        8866:30911/TCP              28m
```

访问下查询商品的接口，如果有json数据返回，代表服务正常：

```
http://192.168.65.210:30911/pms/productInfo/1
```



{"code":404,"message":"产品不存在!","data":null}

用相同的方法部署下order，member，gateway，authcenter等服务，这里不一一详述了，附上每个服务k8s部署的yaml文件供大家参考。

authcenter服务

tulingmall-authcenter-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tulingmall-authcenter-deployment
  labels:
    app: tulingmall-authcenter
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tulingmall-authcenter
  template:
    metadata:
      labels:
```

```
        app: tulingmall-authcenter
    spec:
      containers:
        - name: tulingmall-authcenter
          image: zhuge666/tulingmall-authcenter:0.0.1
imagePullPolicy: Always
          ports:
            - containerPort: 9999
          env:
- name: TZ
value: Asia/Shanghai
- name: spring.cloud.nacos.config.server-addr
value: 192.168.65.42:8848
            - name: LOG_FILE
              value: /var/logs
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-authcenter/logs
            type: DirectoryOrCreate
dnsPolicy: Default #继承Pod所在宿主机的DNS设置，使pod能访问外网
```

tulingmall-authcenter-service.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: tulingmall-authcenter-service
spec:
 type: NodePort
 selector:
 app: tulingmall-authcenter
 ports:
 - name: http
 protocol: TCP
 port: 9999
 targetPort: 9999
```

gateway服务

tulingmall-gateway-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: tulingmall-gateway-deployment
  labels:
    app: tulingmall-gateway
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tulingmall-gateway
   template:
    metadata:
      labels:
        app: tulingmall-gateway
    spec:
      containers:
        - name: tulingmall-gateway
          image: zhuge666/tulingmall-gateway:0.0.1
  imagePullPolicy: Always
          ports:
            - containerPort: 8888
          env:
  - name: TZ
  value: Asia/Shanghai
  - name: spring.cloud.nacos.config.server-addr
  value: 192.168.65.42:8848
            - name: LOG_FILE
              value: /var/logs
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-gateway/logs
            type: DirectoryOrCreate
  dnsPolicy: Default #继承Pod所在宿主机的DNS设置，使pod能访问外网
```

tulingmall-gateway-service.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: tulingmall-gateway-service
spec:
 type: NodePort
 selector:
 app: tulingmall-gateway
 ports:
 - name: http
```

```
  protocol: TCP
  port: 8888
  targetPort: 8888
```

order服务

tulingmall-order-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tulingmall-order-deployment
  labels:
    app: tulingmall-order
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tulingmall-order
 template:
    metadata:
      labels:
        app: tulingmall-order
    spec:
      containers:
        - name: tulingmall-order
          image: zhuge666/tulingmall-order:0.0.1
  imagePullPolicy: Always
          ports:
            - containerPort: 8844
          env:
  - name: TZ
  value: Asia/Shanghai
  - name: spring.cloud.nacos.config.server-addr
  value: 192.168.65.42:8848
            - name: LOG_FILE
              value: /var/logs
  - name: JAVA_TOOL_OPTIONS
  value: -Xms1G -Xmx1G -Xmn512M -Xss512K -XX:MetaspaceSize=256M -XX:MaxMetaspaceSize=25
6M -javaagent:/agent/skywalking-agent.jar -DSW_AGENT_NAME=tulingmall-order -DSW_AGENT_COL
LECTOR_BACKEND_SERVICES=192.168.65.204:11800
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-order/logs
```

```
            type: DirectoryOrCreate
  dnsPolicy: Default #继承Pod所在宿主机的DNS设置，使pod能访问外网
```

tulingmall-order-service.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: tulingmall-order-service
spec:
 type: NodePort
 selector:
 app: tulingmall-order
 ports:
  - name: http
  protocol: TCP
  port: 8844
  targetPort: 8844
```

member服务

tulingmall-member-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tulingmall-member-deployment
  labels:
    app: tulingmall-member
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tulingmall-member
   template:
    metadata:
      labels:
        app: tulingmall-member
    spec:
      containers:
        - name: tulingmall-member
          image: zhuge666/tulingmall-member:0.0.1
  imagePullPolicy: Always
          ports:
            - containerPort: 8877
          env:
  - name: TZ
  value: Asia/Shanghai
  - name: spring.cloud.nacos.config.server-addr
```

```
  value: 192.168.65.42:8848
            - name: LOG_FILE
              value: /var/logs
  - name: JAVA_TOOL_OPTIONS
  value: -Xmx1g -Xms1g -XX:MaxMetaspaceSize=256m -javaagent:/agent/skywalking-agent.jar
-DSW_AGENT_NAME=tulingmall-order -DSW_AGENT_COLLECTOR_BACKEND_SERVICES=192.168.65.204:118
00
          volumeMounts:
            - mountPath: /var/logs
              name: log-volume
      volumes:
        - name: log-volume
          hostPath:
            path: /mydata/k8s-app/tulingmall-member/logs
            type: DirectoryOrCreate
  dnsPolicy: Default #继承Pod所在宿主机的DNS设置，使pod能访问外网
```

tulingmall-member-service.yaml

```
apiVersion: v1
kind: Service
metadata:
 name: tulingmall-member-service
spec:
 type: NodePort
 selector:
 app: tulingmall-member
 ports:
  - name: http
  protocol: TCP
  port: 8877
  targetPort: 8877
```

创建网关的Ingress(相当于Nginx)

最后，我们来创建网关服务的Ingress，创建文件tulingmall-gateway-ingress.yaml，内容如下：

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: tulingmall-gateway-ingress
spec:
 rules:
 - host: gateway.tuling.com #转发域名
 http:
 paths:
  - path: /
  backend:
  serviceName: tulingmall-gateway-service
  servicePort: 8888 #service的端口
```

执行如下命令生效规则：

```
kubectl apply -f tulingmall-gateway-ingress.yaml
```

查看生效的ingress规则：

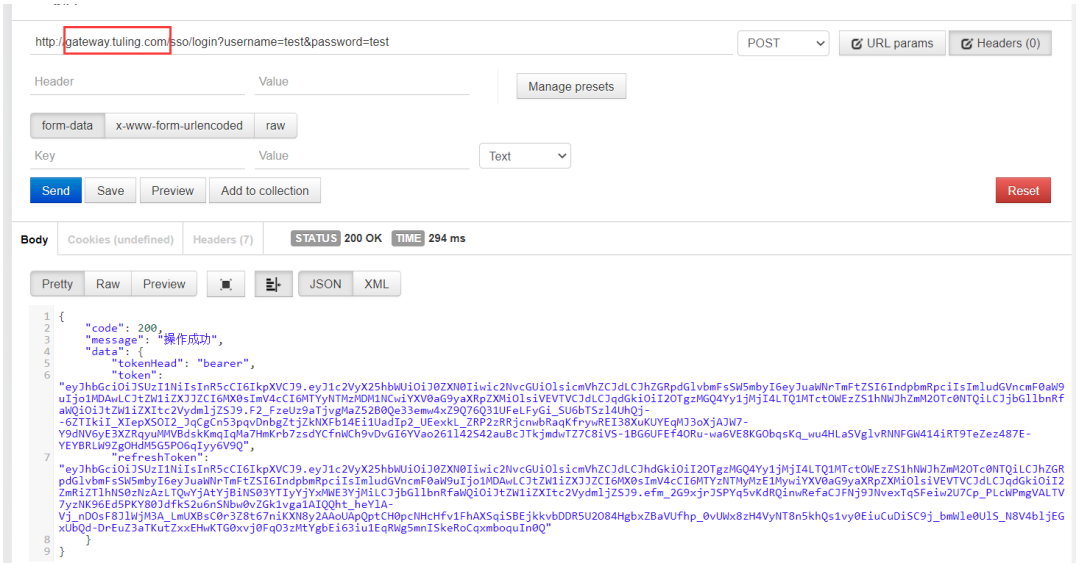
```
kubectl get ing
```

```
[root@k8s-master tulingmall-gateway]# kubectl get ing
NAME                        CLASS    HOSTS                ADDRESS   PORTS   AGE
tulingmall-gateway-ingress  <none>   gateway.tuling.com             80      33s
web-ingress                 <none>   tomcat.tuling.com              80      4d6h
```

在访问机器配置host，win10客户机在目录：C:\Windows\System32\drivers\etc，在host里增加如下host(ingress部署的机器ip对应访问的域名)

```
192.168.65.203 gateway.tuling.com
或者
192.168.65.210 gateway.tuling.com
```

配置完后直接在客户机用域名请求下网关：




```
文档：Kubernetes电商微服务部署实战
链接：http://note.youdao.com/noteshare?id=dbaa1b80aafc1769a5866e693b44d08e&sub=DD4D0195FDE04F91B926AC9E064D9394
```