K8s案例汇总

# nginx+tomcat+NFS实现动静分离

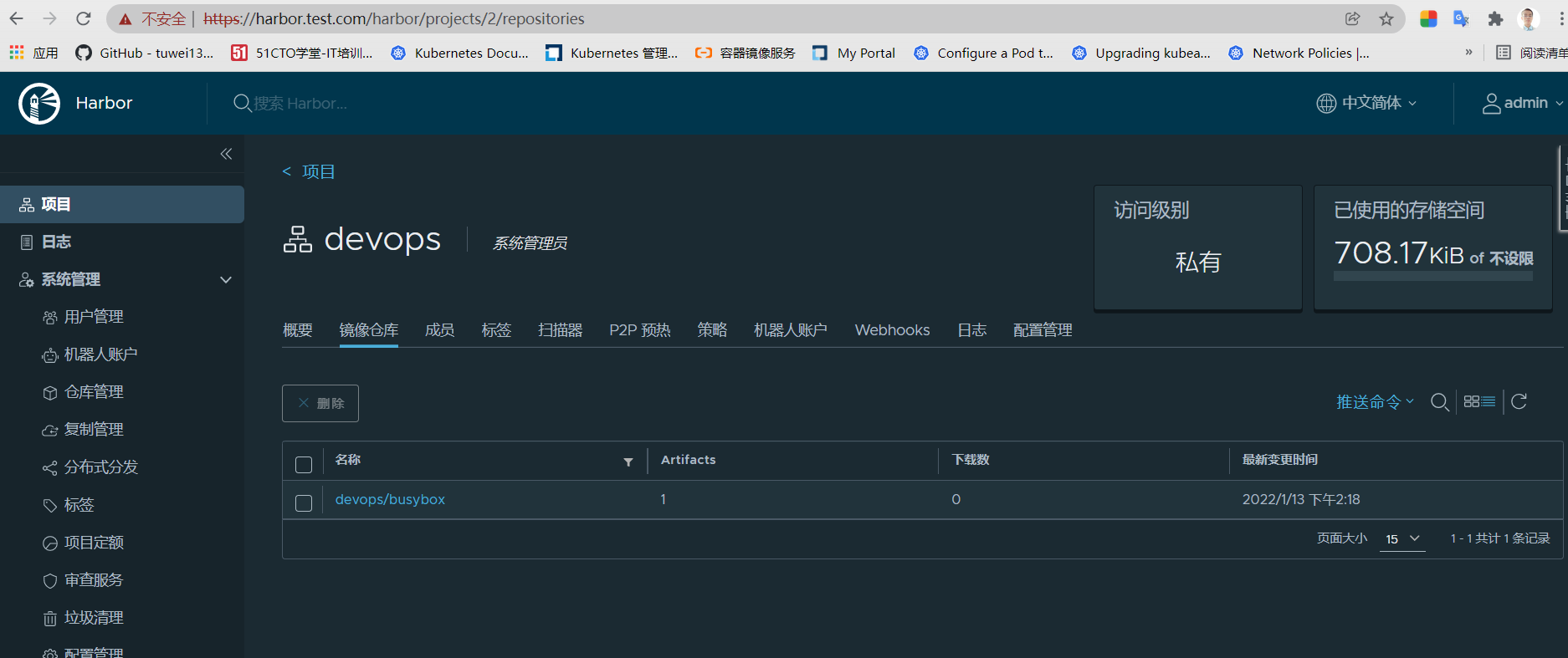
## 案例说明

在k8s部署nginx+tomcat容器，静态的数据通过nginx处理，动态的数据通过nginx代理给tomcat处理（nginx配置中tomcat地址采用service方式访问集群内部资源），其中一些静态数据如图片等存储在NFS共享存储中，实现数据持久化。

## 平台及资源预览

集群k8s版本：1.20.4，双master，三个计算节点，已安装nfs存储类，master通过nginx+keepalived做负载均衡及高可用。

另外，搭建了一个harbor镜像仓库，<https://harbor.test.com>，本机hosts访问，用于存放镜像。



## 镜像资源

### nginx镜像dockerfile配置

FROM centos:7.6.1810

MAINTAINER https://blog.51cto.com/tuwei

RUN yum install -y gcc gcc-c++ make \

openssl-devel pcre-devel gd-devel \

iproute net-tools telnet wget curl && \

yum clean all && \

rm -rf /var/cache/yum/\*

RUN wget http://nginx.org/download/nginx-1.20.2.tar.gz && \

tar zxf nginx-1.20.2.tar.gz && \

cd nginx-1.20.2 && \

./configure --prefix=/usr/local/nginx \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module && \

make -j 4 && make install && \

mkdir -p /usr/local/nginx/html/webapp/static /usr/local/nginx/html/webapp/images && \

cd / && rm -rf nginx-1.20.2\* && \

ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

ENV PATH $PATH:/usr/local/nginx/sbin

COPY nginx.conf /usr/local/nginx/conf/nginx.conf

ADD index.html /usr/local/nginx/html/webapp/

WORKDIR /usr/local/nginx

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

该dockerfile采用centos7.6基础镜像，源码编译安装nginx1.20.2版本并且删除相关文件，减小镜像大小，同时创建了两个目录用于存放nginx的静态数据。镜像构建：docker build -t harbor.test.com/devops/nginx:1.20.2 -f Dockerfile-nginx . 推送到镜像仓库docker push harbor.test.com/devops/nginx:1.20.2

### Tomcat镜像构建

之前已经构建了tomcat镜像（tuwei1314/tomcat:8.5.75），构建用的dockerfile为

FROM centos:7.6.1810

MAINTAINER <https://blog.51cto.com/tuwei>

ENV VERSION=8.5.75

RUN yum install java-1.8.0-openjdk wget curl unzip iproute net-tools -y &&  
yum clean all &&  
rm -rf /var/cache/yum/\*

RUN wget [https://dlcdn.apache.org/tomcat/tomcat-8/v${VERSION}/bin/apache-tomcat-${VERSION}.tar.gz](https://dlcdn.apache.org/tomcat/tomcat-8/v$%7BVERSION%7D/bin/apache-tomcat-$%7BVERSION%7D.tar.gz) --no-check-certificate &&  
tar zxf apache-tomcat-${VERSION}.tar.gz &&  
mv apache-tomcat-${VERSION} /usr/local/tomcat &&  
rm -rf apache-tomcat-${VERSION}.tar.gz /usr/local/tomcat/webapps/\* &&  
mkdir /usr/local/tomcat/webapps/myapp &&  
echo "welcome to tomcat,tomcat server is ok" > /usr/local/tomcat/webapps/myapp/status.html &&  
sed -i '1a JAVA\_OPTS="-Djava.security.egd=file:/dev/./urandom"' /usr/local/tomcat/bin/catalina.sh &&  
ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

ENV PATH $PATH:/usr/local/tomcat/bin

WORKDIR /usr/local/tomcat

EXPOSE 8080 CMD ["catalina.sh", "run"]

本次以该镜像为基础镜像进行构建

FROM tuwei1314/tomcat:8.5.75

ADD ly-simple-tomcat-0.0.1-SNAPSHOT.war /usr/local/tomcat/webapps/myapp/ROOT.war

该war包为java示例。

## yaml文件

### nginx服务的 yaml配置

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nginx-data

annotations:

volume.beta.kubernetes.io/storage-class: "managed-nfs-storage"

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

---

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: nginx-deployment

name: nginx-deployment

spec:

replicas: 2

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- {key: app, operator: In, values: ["nginx"]}

topologyKey: kubernetes.io/hostname

containers:

- name: nginx-container

image: harbor.test.com/devops/nginx:1.20.2

imagePullPolicy: IfNotPresent

ports:

- containerPort: 80

protocol: TCP

name: http

- containerPort: 443

protocol: TCP

name: https

resources:

limits:

cpu: 2

memory: 2Gi

requests:

cpu: 200m

memory: 256Mi

volumeMounts:

- name: nginx-data

mountPath: /usr/local/nginx/html/webapp/static

readOnly: false

volumes:

- name: nginx-data

persistentVolumeClaim:

claimName: nginx-data

---

kind: Service

apiVersion: v1

metadata:

labels:

app: nginx-service

name: nginx-service

spec:

type: NodePort

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

nodePort: 30090

- name: https

port: 443

protocol: TCP

targetPort: 443

nodePort: 30091

selector:

app: nginx

### nginx配置文件

#user nobody;

worker\_processes auto;

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

upstream tomcat\_webserver {

server tomcat-app1-service.default.svc.cluster.local:80;

}

server {

listen 80;

server\_name localhost;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root html;

index index.html index.htm;

}

location /webapp {

root html;

index index.html index.htm;

}

location /myapp {

proxy\_pass http://tomcat\_webserver;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Real-IP $remote\_addr;

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

# another virtual host using mix of IP-, name-, and port-based configuration

#

#server {

# listen 8000;

# listen somename:8080;

# server\_name somename alias another.alias;

# location / {

# root html;

# index index.html index.htm;

# }

#}

# HTTPS server

#

#server {

# listen 443 ssl;

# server\_name localhost;

# ssl\_certificate cert.pem;

# ssl\_certificate\_key cert.key;

# ssl\_session\_cache shared:SSL:1m;

# ssl\_session\_timeout 5m;

# ssl\_ciphers HIGH:!aNULL:!MD5;

# ssl\_prefer\_server\_ciphers on;

# location / {

# root html;

# index index.html index.htm;

# }

#}

}

### tomcat配置的yaml文件

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: tomcat-app1-deployment-label

name: tomcat-app1-deployment

spec:

replicas: 1

selector:

matchLabels:

app: tomcat-app1-selector

template:

metadata:

labels:

app: tomcat-app1-selector

spec:

containers:

- name: tomcat-app1-container

image: harbor.test.com/devops/tomcat:8.5.75

imagePullPolicy: IfNotPresent

ports:

- containerPort: 8080

protocol: TCP

name: http

resources:

limits:

cpu: 1

memory: "512Mi"

requests:

cpu: 500m

memory: "512Mi"

volumeMounts:

- name: nginx-data

mountPath: /usr/local/nginx/html/webapp/static

readOnly: false

volumes:

- name: nginx-data

persistentVolumeClaim:

claimName: nginx-data

---

kind: Service

apiVersion: v1

metadata:

labels:

app: tomcat-app1-service

name: tomcat-app1-service

spec:

type: NodePort

ports:

- name: http

port: 80

protocol: TCP

targetPort: 8080

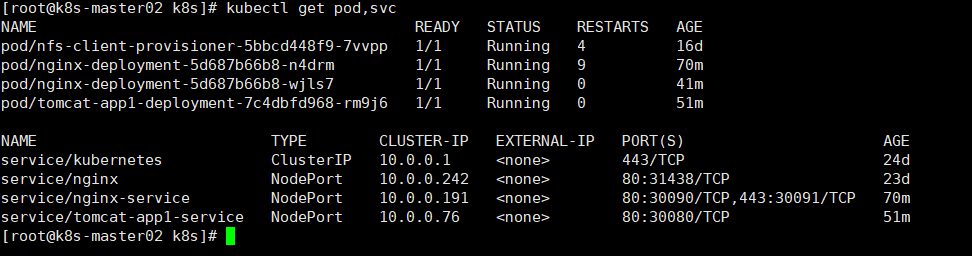
nodePort: 30080

selector:

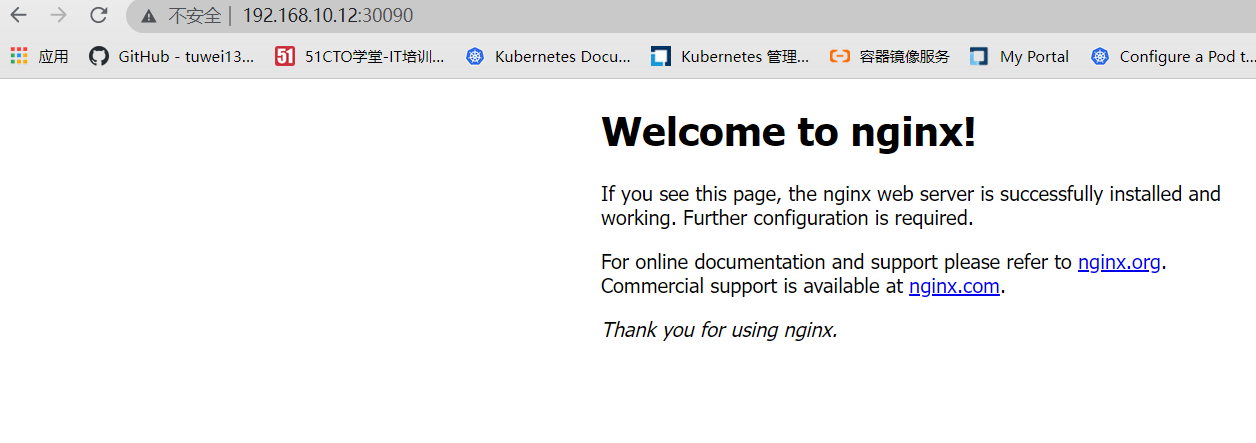
app: tomcat-app1-selector

## 服务运行情况

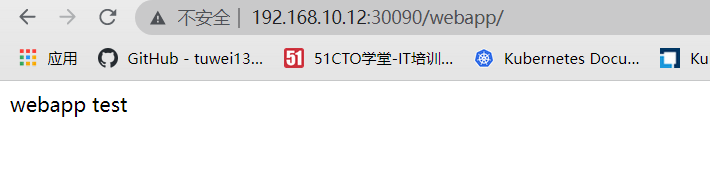
之前由于nginx.conf中配置了tomcat服务，tomcat没起来前，nginx服务是无法正常运行的，两个pod创建后如下：



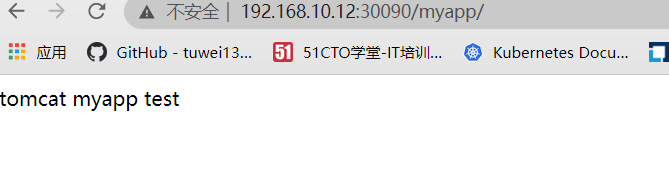
### 通过nodeport方式访问nginx首页如下：



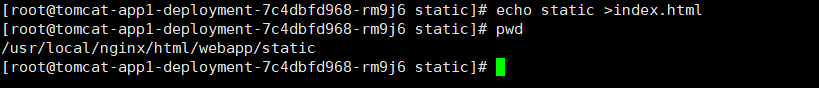
### 访问/webapp



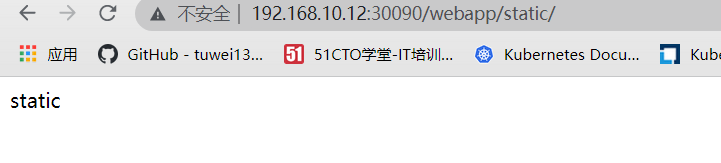
### 访问/myapp,然后nginx将请求转给tomcat处理



进入tomcat容器的持久化目录，模拟创建一个html文件



访问如下：



### 单独访问tomcat实例



# 部署zookeeper集群

分布式系统中一般采用zookeeper作为注册中心，三个节点组成集群。

# Mysql主从架构

Mysql服务书序有状态服务，而且主从架构中mysql实例有主从之分，需要有唯一的网络标识，需要用到statefulset控制器部署。本次实验部署一主两从，共3个pod

## 存储准备

创建三个pv（如果没有存储类动态创建pv）

[root@k8s-master01 pv]# cat mysql-persistentvolume.yaml

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-datadir-1

spec:

capacity:

storage: 10Gi

accessModes:

- ReadWriteOnce

nfs:

path: /nfs/kubernetes/mysql-datadir-1

server: 192.168.10.12

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-datadir-2

spec:

capacity:

storage: 10Gi

accessModes:

- ReadWriteOnce

nfs:

path: /nfs/kubernetes/mysql-datadir-2

server: 192.168.10.12

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-datadir-3

spec:

capacity:

storage: 10Gi

accessModes:

- ReadWriteOnce

nfs:

path: /nfs/kubernetes/mysql-datadir-3

server: 192.168.10.12

我这边集群安装了nfs作为默认存储类，可以利用pvc模板动态创建。

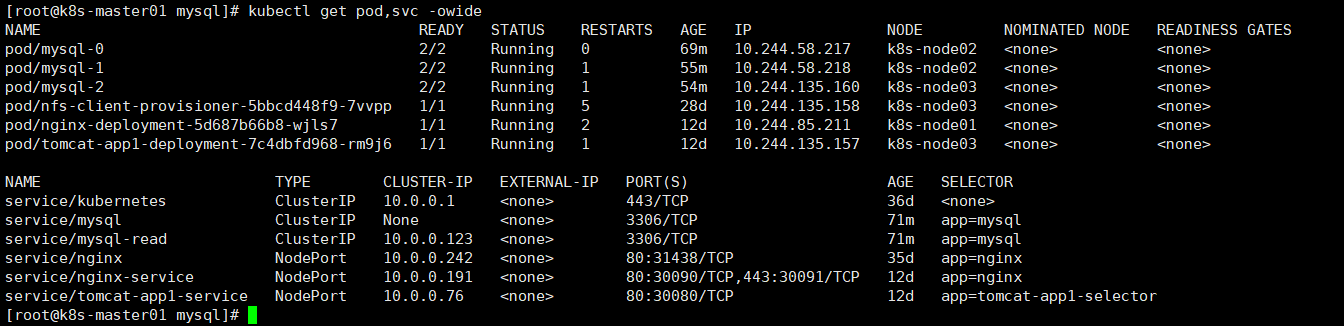
## Yaml准备

部署mysql，设计到配置文件，服务访问和statefulset，共三个文件。

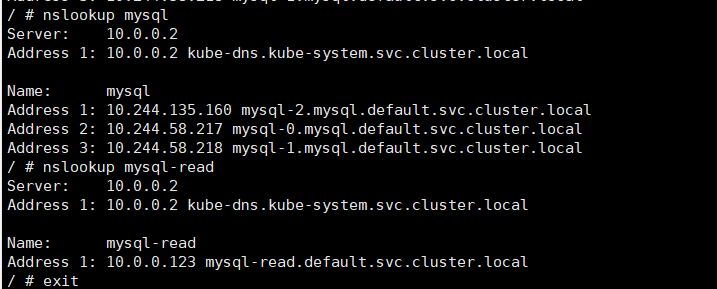
其中名称为mysql-0作为主库，程序写的话连接该库其他两个作为从库，负责读的操作，通过service名称访问。



创建后如下图：



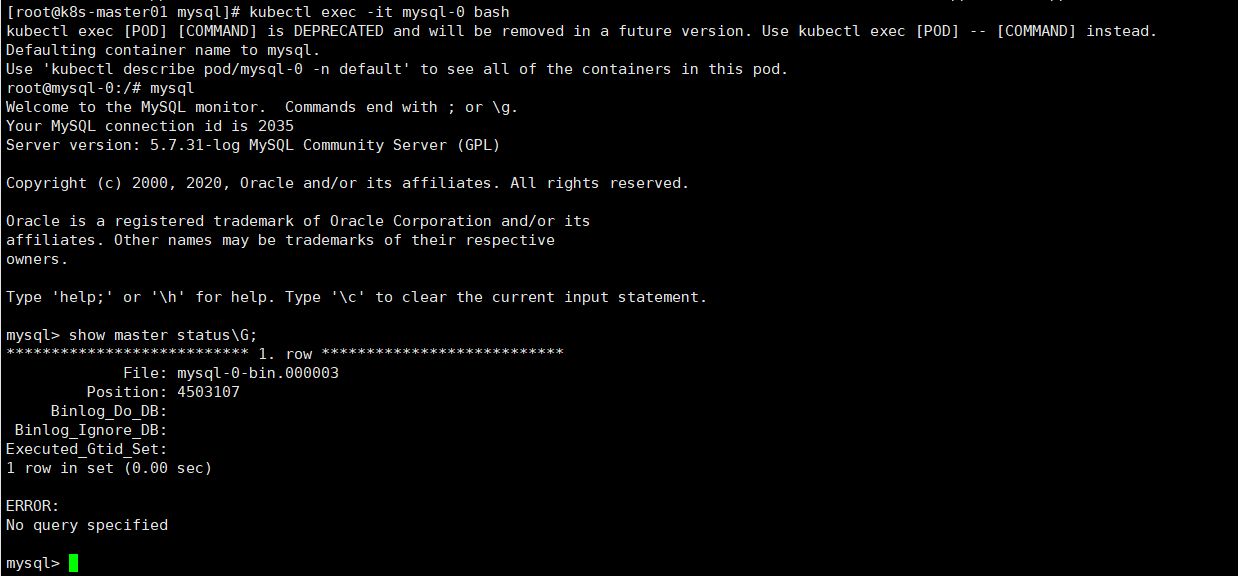
这里创建了两个service，一个是headless service，一个是常用的service。



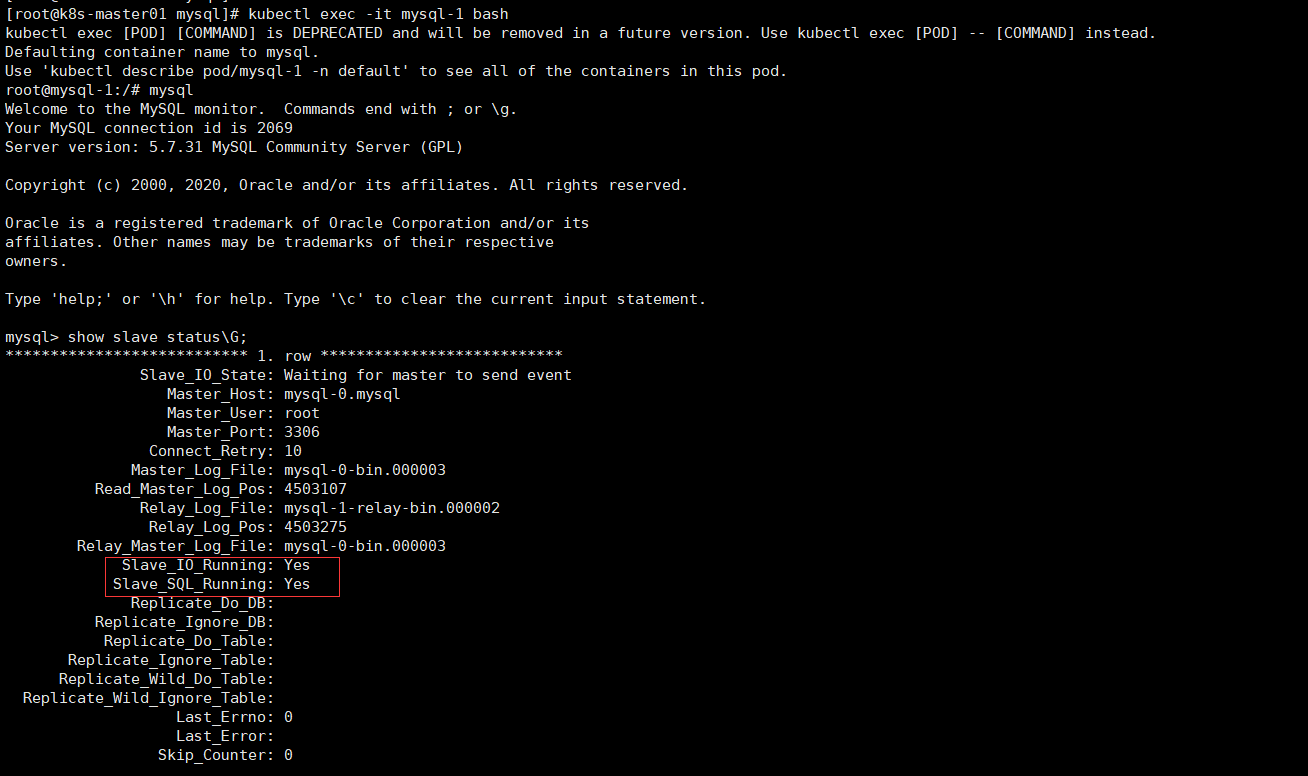
普通的service通过解析得到clusterip地址，statefulset中用到的headless service俗称无头服务，由于没有clusterip，解析后得到后端endpoint列表，即对应pod ip信息，每个pod都有一个对应的domain name

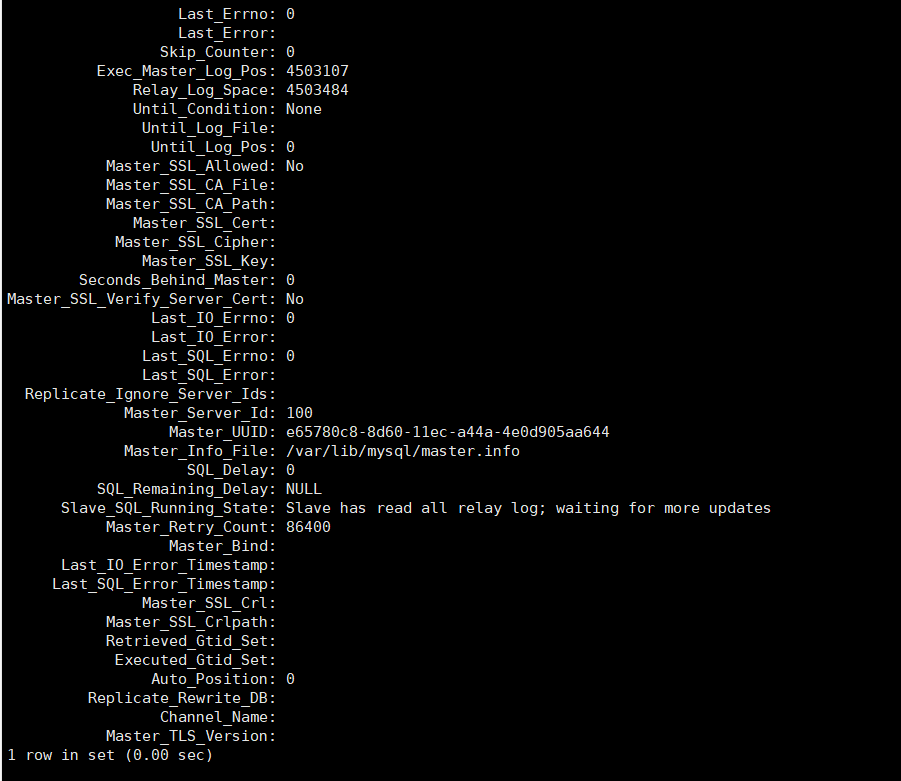
## 主从库查看

进入主库



进入从库





可以看到两个线程状态为yes。

# Nginx+php搭建wordpress博客系统

## 镜像准备

这里采用nginx+php一体的镜像tuwei1314/nginx-php-fp:latest

其中nginx版本为1.18，php版本为7.4.9,root目录为/var/www/html

## Yaml准备



将wordpress相关文件放在nfs路径下并给该目录普通用户可写权限

kind: Deployment

apiVersion: apps/v1

metadata:

labels:

app: wordpress-app

name: wordpress-app-deployment

spec:

replicas: 1

selector:

matchLabels:

app: wordpress-app

template:

metadata:

labels:

app: wordpress-app

spec:

containers:

- name: wordpress-app-nginx

image: tuwei1314/nginx-php-fpm:latest

imagePullPolicy: Always

ports:

- containerPort: 80

protocol: TCP

name: http

- containerPort: 443

protocol: TCP

name: https

volumeMounts:

- name: wordpress

mountPath: /var/www/html/

readOnly: false

volumes:

- name: wordpress

nfs:

server: 192.168.10.12

path: /nfs/kubernetes/wordpress

---

kind: Service

apiVersion: v1

metadata:

labels:

app: wordpress-app

name: wordpress-app

spec:

type: NodePort

ports:

- name: http

port: 80

protocol: TCP

targetPort: 80

nodePort: 30031

- name: https

port: 443

protocol: TCP

targetPort: 443

nodePort: 30033

selector:

app: wordpress-app

访问wordpress <http://192.168.10.12:30031/>



在之前创建好的主库中创建wordpress数据库，并给用户wordpress授予权限。

create database wordpress;

grant all on wordpress.\* to wordpress@‘%’ identified by ‘123456’;

数据库这里需要配置主库的地址，写完整的地址mysql-0.mysql.default.svc.cluster.local



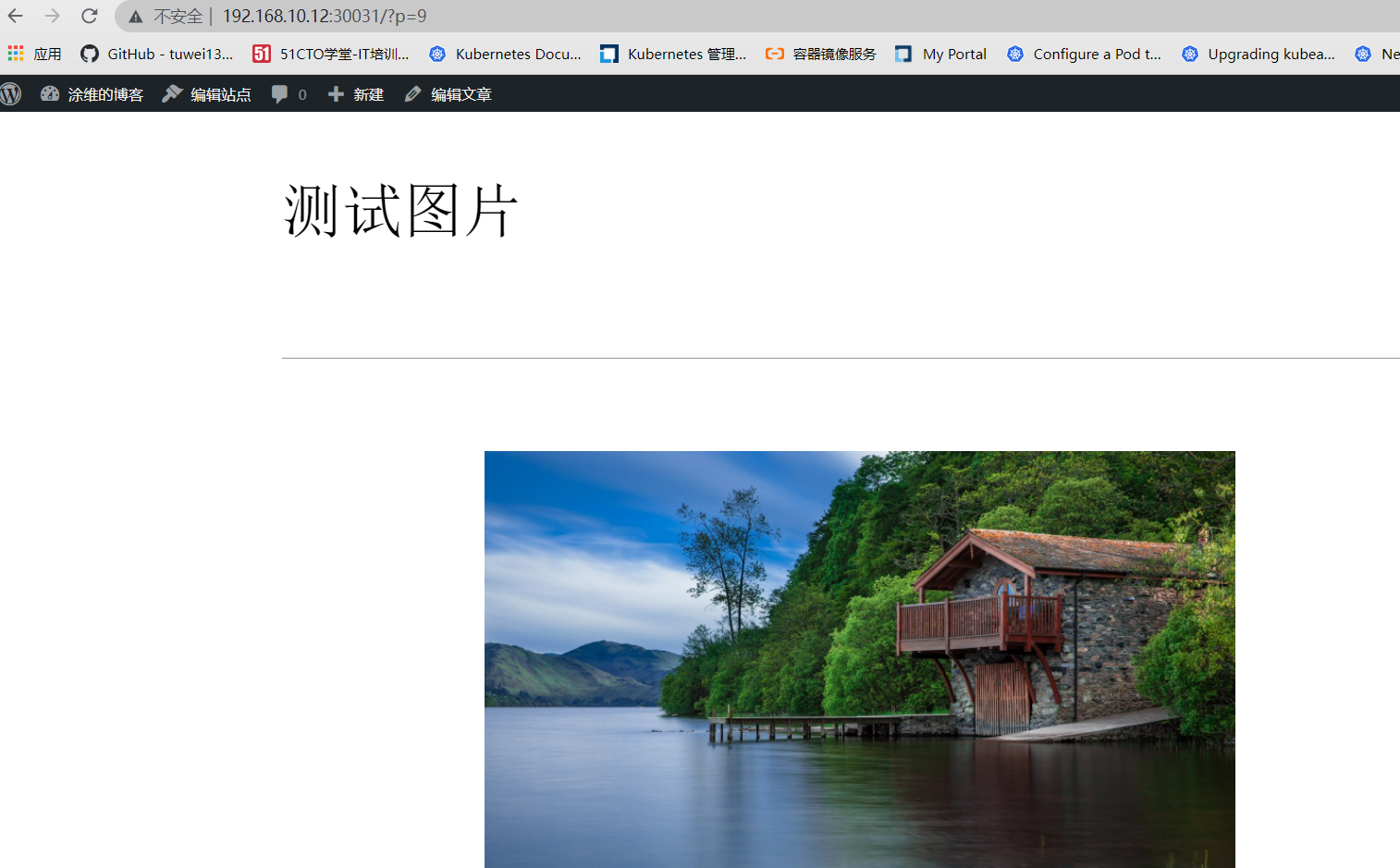




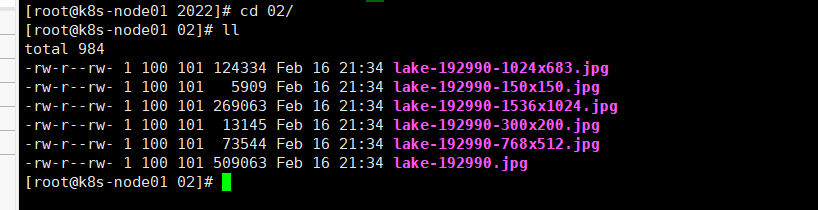


写博客进行发布如下图：





Nfs存储上可以查看上传的图片



# 利用jdk环境部署jenkins

## Jdk镜像



本次镜像中还添加了maven环境，其中maven配置文件中添加了阿里云的maven源，编译java环境时会快些。

FROM centos:7.6.1810

MAINTAINER https://blog.51cto.com/tuwei

ADD jdk-8u301-linux-x64.tar.gz /usr/local/src/

ADD apache-maven-3.6.0.tar.gz /usr/local/src/

RUN ln -sv /usr/local/src/jdk1.8.0\_301 /usr/local/jdk1.8.0 && \

ln -sv /usr/local/src/apache-maven-3.6.0 /usr/local/maven3.6

ADD profile /etc/profile

ENV JAVA\_HOME /usr/local/jdk1.8.0

ENV MAVEN\_HOME /usr/local/maven3.6

ENV JRE\_HOME $JAVA\_HOME/jre

ENV CLASSPATH $JAVA\_HOME/lib/:$JRE\_HOME/lib/

ENV PATH $PATH:$JAVA\_HOME/bin:${MAVEN\_HOME}/bin

构建好的镜像是tuwei1314/jdk1.8-maven3.6:v1

# Redis集群