开发环境从IDEA到Kubernetes流程

目录

[开发环境从IDEA到Kubernetes流程 1](#_Toc37868737)

[准备工作 1](#_Toc37868738)

[Docker方面 1](#_Toc37868739)

[开通2375 1](#_Toc37868740)

[重启docker 1](#_Toc37868741)

[下载基本镜像 1](#_Toc37868742)

[搭建镜像私服 1](#_Toc37868743)

[Kubernetes方面 2](#_Toc37868744)

[集群的搭建 2](#_Toc37868745)

[注意事项 2](#_Toc37868746)

[开发环境设置 2](#_Toc37868747)

[IDEA打镜像 2](#_Toc37868748)

[准备Dockerfile 2](#_Toc37868749)

[配置docker远程连接 2](#_Toc37868750)

[配置镜像相关 3](#_Toc37868751)

[Docker 处理 6](#_Toc37868752)

[查看镜像 6](#_Toc37868753)

[推送镜像 6](#_Toc37868754)

[Kubernetes运行 7](#_Toc37868755)

[终结点的准备 7](#_Toc37868756)

[环境变量设置 8](#_Toc37868757)

[应用的yaml样例 9](#_Toc37868758)

[访问应用 12](#_Toc37868759)

[主页 12](#_Toc37868760)

[测试mysql页 12](#_Toc37868761)

[测试redis页 13](#_Toc37868762)

# 准备工作

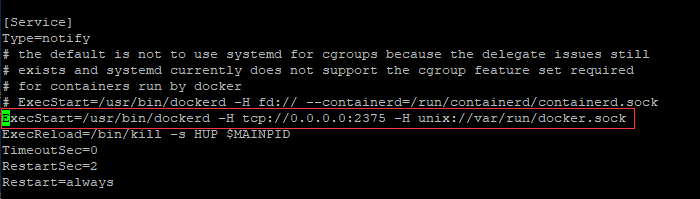
注：开发机-Docker-Kubernetes等网络都打通，可以互相访问；

## Docker方面

### 开通2375

远程Docker服务器需要打开2375端口，供idea连接使用

#vi /usr/lib/systemd/system/docker.service



加入 -H tcp://0.0.0.0:2375 -H unix://var/run/docker.sock

### 重启docker

#systemctl daemon-reload

#systemctl restart docker

### 下载基本镜像

openjdk:8

### 搭建镜像私服

参照《docker私服搭建.txt》

## Kubernetes方面

### 集群的搭建

参照《kubernetes-v1.17.x配置文档.doc》

### 注意事项

创建访问私服的密钥

创建终结点访问集群外的服务(mysql/redis)

# 开发环境设置

## IDEA打镜像

### 准备Dockerfile

准备Dockerfile(放在工程根目录)文件，内容参照如下

FROM openjdk:8

VOLUME /tmp

ADD ./target/spring-boot-demo-0.0.1-SNAPSHOT.jar app.jar

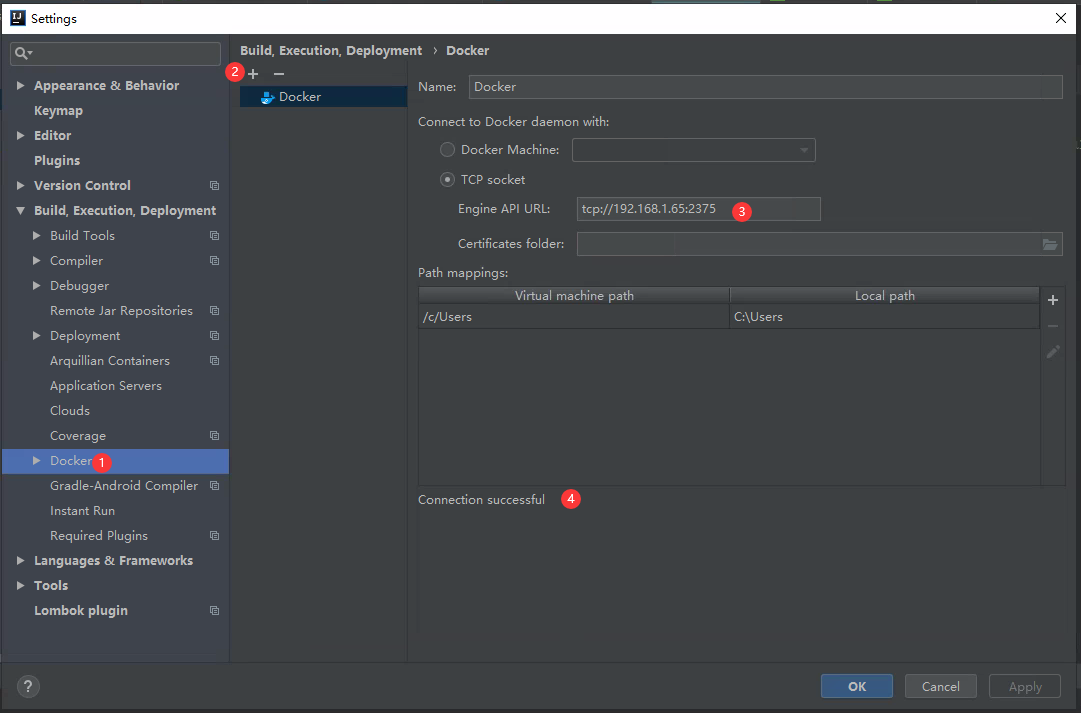
RUN bash -c 'touch /app.jar'

ENTRYPOINT ["java","-jar","-server","-Djava.security.egd=file:/dev/./urandom","/app.jar"]

### 配置docker远程连接

如果是第一次做远程连接，需要配置docker远程连接

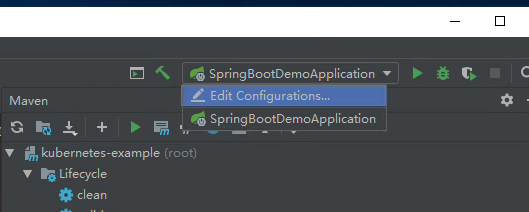
File-Settings-Build,Execution,Deployment-Docker

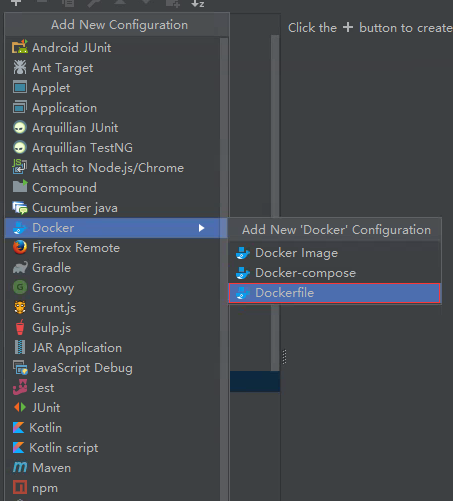


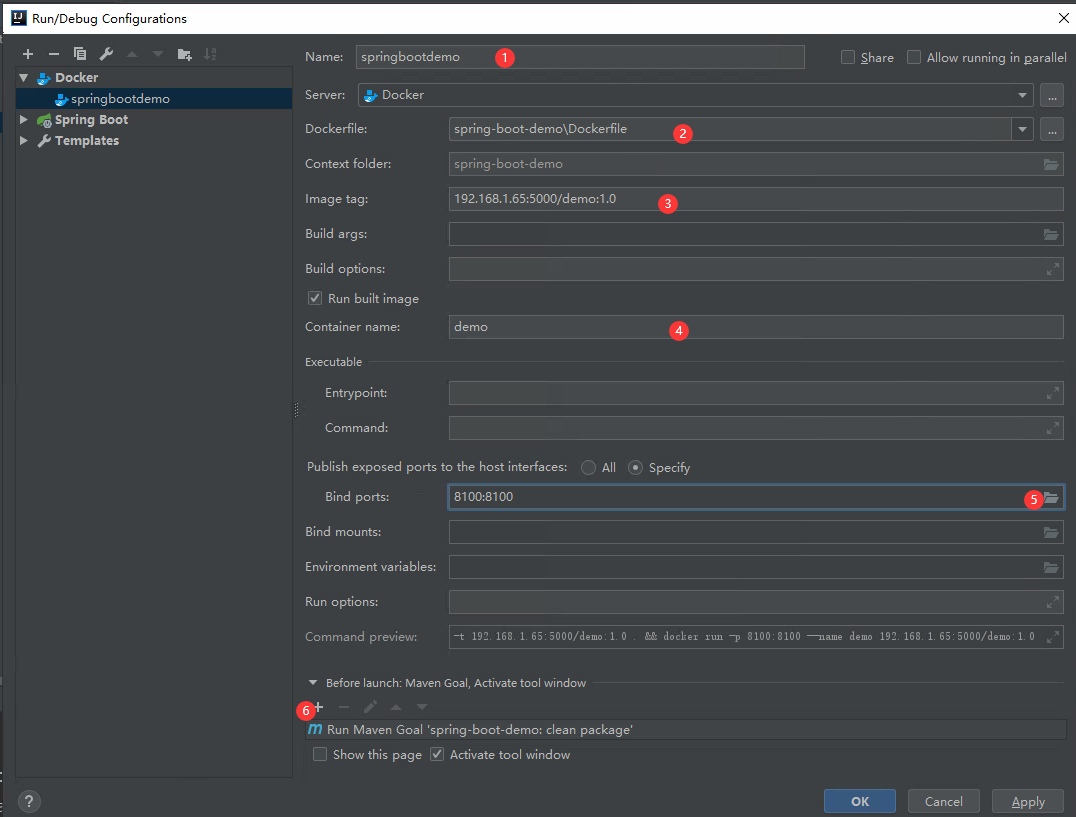
1. 点击Docker
2. 添加Docker
3. 设置名称和TCP socket
4. 设置完成后需要看连通性，返回Connection successful为成功

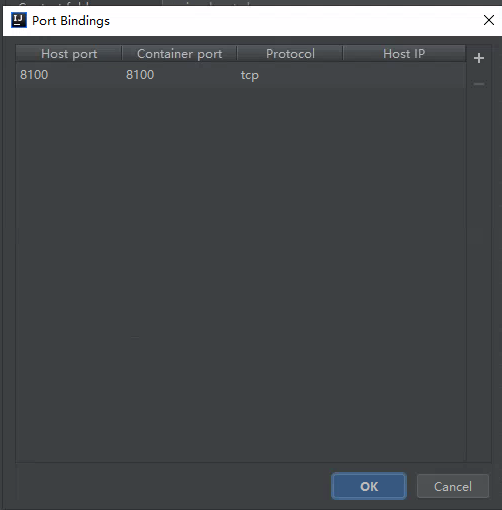
### 配置镜像相关

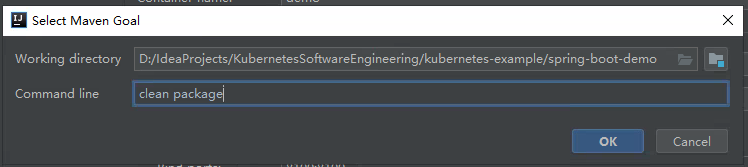
假设程序已经在本地测试完毕，需要打成镜像











1. 配置名称
2. Dockerfile文件地址
3. 镜像名称
4. 运行容器的名称
5. 对外的端口号
6. 附带的maven命令，注意选择路径

完成后运行镜像配置，看到类似如下信息就是成功了

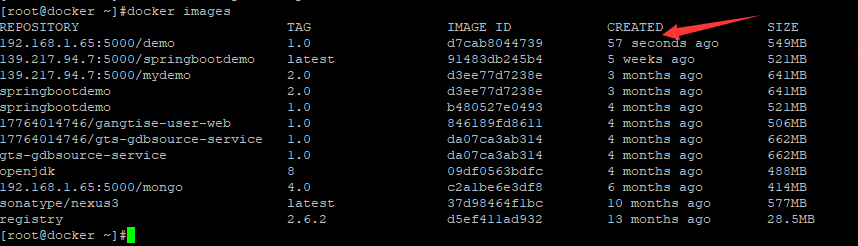


## Docker 处理

### 查看镜像

登录到远程Docker 的服务器上

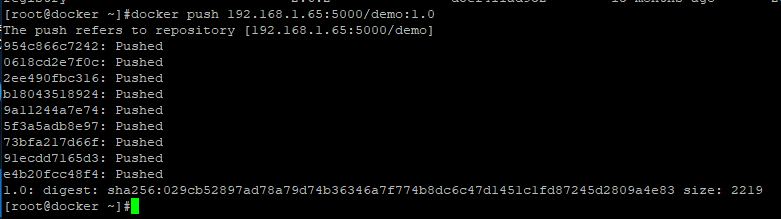
#docker images



可以看到刚刚生成的镜像已经在docker服务器中了

### 推送镜像

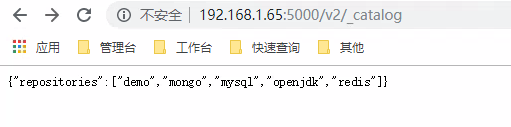
#docker push 192.168.1.65:5000/demo:1.0



打开浏览器检查

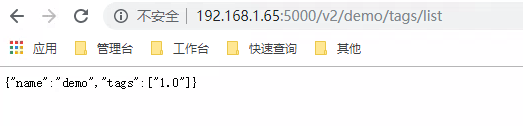
检查镜像

http://192.168.1.65:5000/v2/\_catalog



检查镜像版本

http://192.168.1.65:5000/v2/demo/tags/list



# Kubernetes运行

## 终结点的准备

由于需要访问集群外部资源，需要先做一些终结点。示例：

apiVersion: v1

kind: Service

metadata:

name: redis

spec:

ports:

- name: p0

port: 13000

protocol: TCP

- name: p1

port: 13001

protocol: TCP

- name: p2

port: 13002

protocol: TCP

- name: p3

port: 13003

protocol: TCP

- name: p4

port: 13004

protocol: TCP

- name: p5

port: 13005

protocol: TCP

---

kind: Endpoints

apiVersion: v1

metadata:

name: redis

subsets:

- addresses:

- ip: 192.168.1.93

ports:

- name: p0

port: 13000

protocol: TCP

- name: p1

port: 13001

protocol: TCP

- name: p2

port: 13002

protocol: TCP

- name: p3

port: 13003

protocol: TCP

- name: p4

port: 13004

protocol: TCP

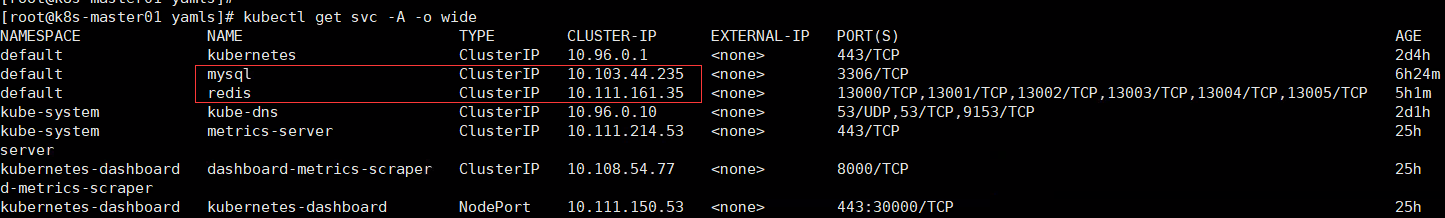
- name: p5

port: 13005

protocol: TCP

终结点yaml文件编写好以后去kubernetes运行即可，终结点完成后kubernetes内的pod可以通过ClusterIP进行访问

#kubectl get svc -A -o wide



## 环境变量设置

编写环境变量yaml文件

apiVersion: v1

kind: ConfigMap

metadata:

name: env-config

data:

MYSQL\_SERVER\_URL: "10.103.44.235"

MYSQL\_USER: "root"

MYSQL\_PWD: "123456"

REDIS\_IP1: "10.111.161.35"

REDIS\_IP2: "10.111.161.35"

REDIS\_IP3: "10.111.161.35"

REDIS\_IP4: "10.111.161.35"

REDIS\_IP5: "10.111.161.35"

REDIS\_IP6: "10.111.161.35"

REDIS\_PWD: "abcd1234"

REDIS\_PORT1: "13000"

REDIS\_PORT2: "13001"

REDIS\_PORT3: "13002"

REDIS\_PORT4: "13003"

REDIS\_PORT5: "13004"

REDIS\_PORT6: "13005"

运行ConfigMap

#kubectl apply -f env-config.yaml

## 应用的yaml样例

编写程序yaml（Service、Deployment、HPA）文件

apiVersion: v1

kind: Service

metadata:

name: demo-service

labels:

app: demo-service

spec:

type: NodePort

ports:

- port: 8100

targetPort: 8100

nodePort: 31001

protocol: TCP

selector:

app: demo

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: demo

spec:

selector:

matchLabels:

app: demo

replicas: 1

template:

metadata:

labels:

app: demo

spec:

imagePullSecrets:

- name: regcred

containers:

- name: demo

image: 192.168.1.65:5000/demo:1.0

envFrom:

- configMapRef:

name: env-config

imagePullPolicy: Always

resources:

requests:

cpu: 500m

memory: 960Mi

limits:

cpu: 600m

memory: 1280Mi

---

apiVersion: autoscaling/v2beta1

kind: HorizontalPodAutoscaler

metadata:

name: demo-hpa

spec:

maxReplicas: 10

minReplicas: 1

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: demo

metrics:

- type: Resource

resource:

name: cpu

targetAverageUtilization: 70

- type: Resource

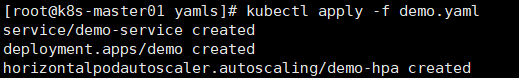
resource:

name: memory

targetAverageValue: 1024Mi

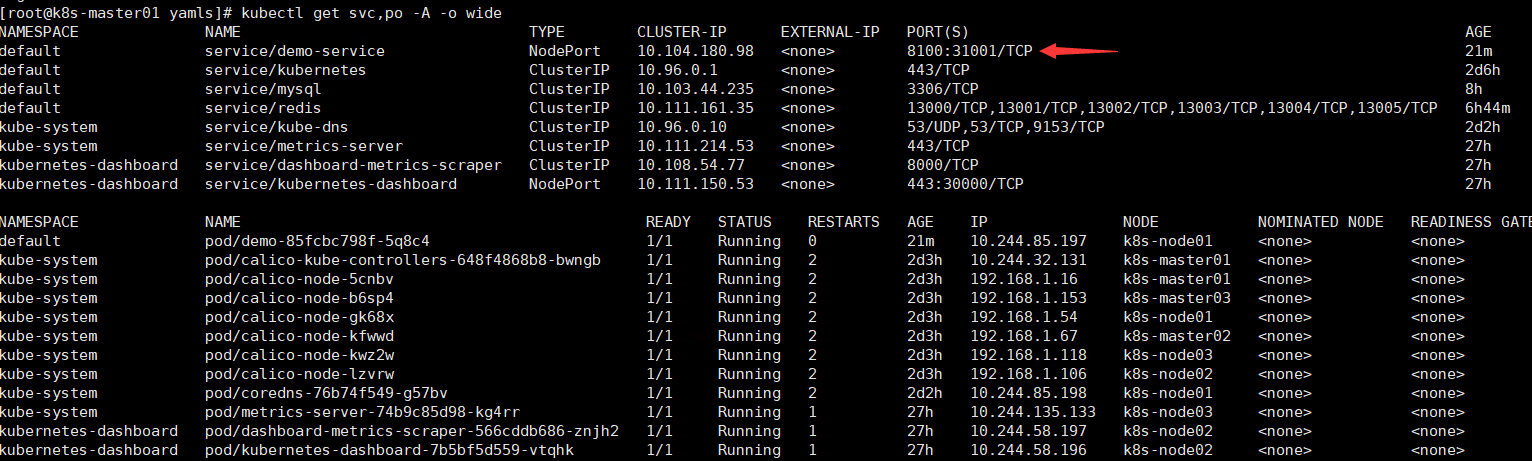
编排镜像

# kubectl apply -f demo.yaml



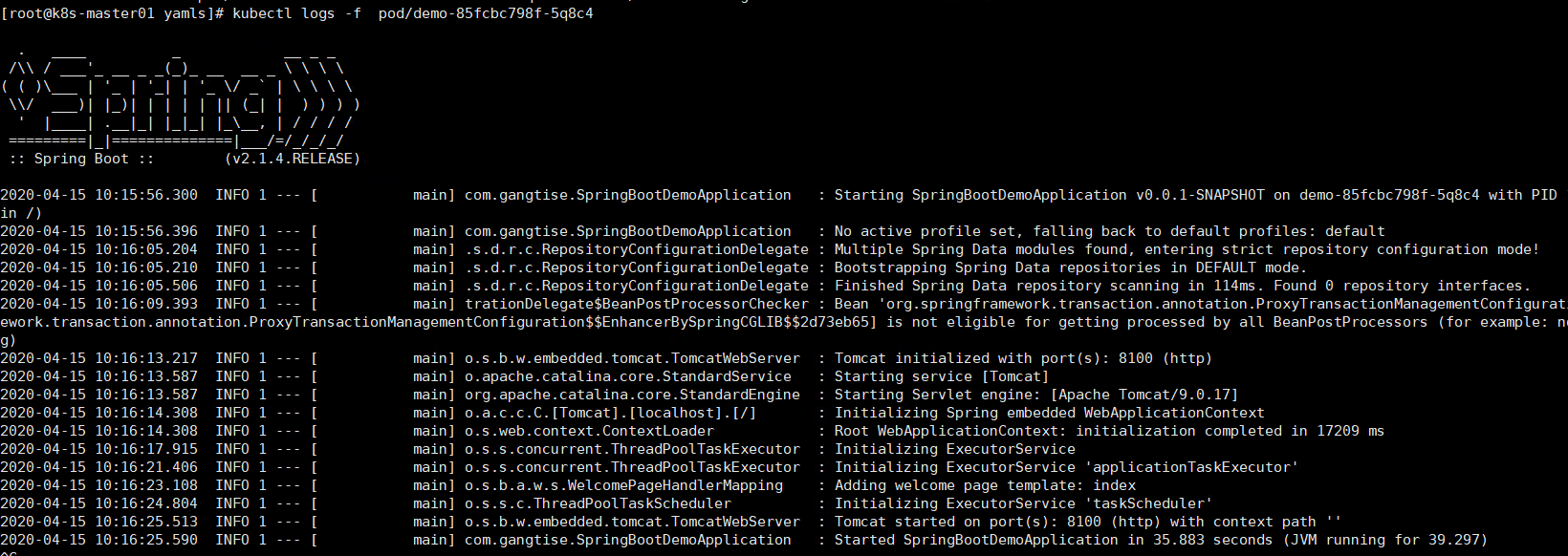
查看端口

#kubectl get svc,po -A -o wide



查看启动日志

#kubectl logs -f pod/demo-85fcbc798f-5q8c4 -n default



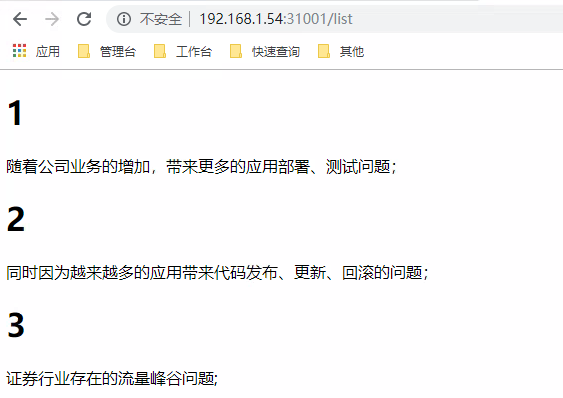
## 访问应用

访问地址http://{nodeIP}:{nodePort}

### 主页



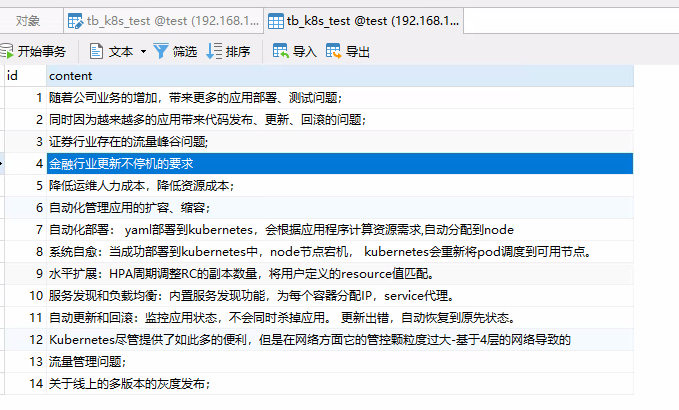
### 测试mysql页



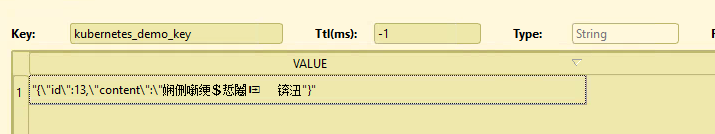
### 测试redis页



mysql中的内容

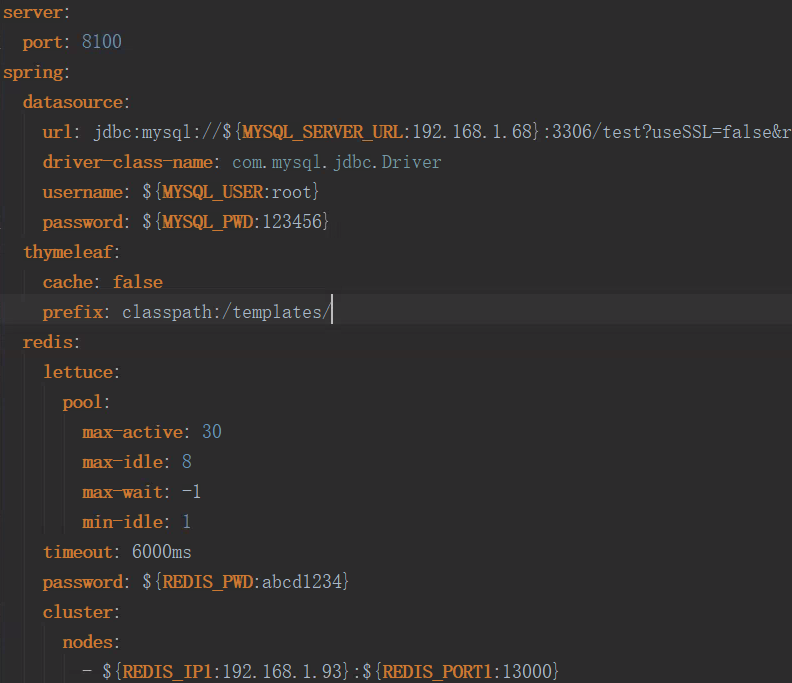


redis中的内容



# 附带文件

最后附带一份程序的配置文件，其中包含环境变量的引入



server:

port: 8100

spring:

datasource:

url: jdbc:mysql://${MYSQL\_SERVER\_URL:192.168.1.68}:3306/test?useSSL=false&requireSSL=false

driver-class-name: com.mysql.jdbc.Driver

username: ${MYSQL\_USER:root}

password: ${MYSQL\_PWD:123456}

thymeleaf:

cache: false

prefix: classpath:/templates/

redis:

lettuce:

pool:

max-active: 30

max-idle: 8

max-wait: -1

min-idle: 1

timeout: 6000ms

password: ${REDIS\_PWD:abcd1234}

cluster:

nodes:

- ${REDIS\_IP1:192.168.1.93}:${REDIS\_PORT1:13000}

- ${REDIS\_IP2:192.168.1.93}:${REDIS\_PORT2:13001}

- ${REDIS\_IP3:192.168.1.93}:${REDIS\_PORT3:13002}

- ${REDIS\_IP4:192.168.1.93}:${REDIS\_PORT4:13003}

- ${REDIS\_IP5:192.168.1.93}:${REDIS\_PORT5:13004}

- ${REDIS\_IP6:192.168.1.93}:${REDIS\_PORT6:13005}