docker部署

# docker环境安装

## 自动安装

### 官方：

|  |
| --- |
| curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun |

### 国内：

|  |
| --- |
| curl -sSL https://get.daocloud.io/docker | sh |

## 手动安装

### 卸载旧版本：

|  |
| --- |
| sudo yum remove docker \  docker-client \  docker-client-latest \  docker-common \  docker-latest \  docker-latest-logrotate \  docker-logrotate \  docker-engine |

### 安装 Docker Engine-Community

#### 安装最新版

|  |
| --- |
| sudo yum install docker-ce docker-ce-cli containerd.io |

#### 安装指定版本

查看版本号：

|  |
| --- |
| yum list docker-ce --showduplicates | sort -r |
| docker-ce.x86\_64 3:18.09.1-3.el7 docker-ce-stable  docker-ce.x86\_64 3:18.09.0-3.el7 docker-ce-stable  docker-ce.x86\_64 18.06.1.ce-3.el7 docker-ce-stable  docker-ce.x86\_64 18.06.0.ce-3.el7 docker-ce-stable |

选择需要的版本号进行安装：

通过其完整的软件包名称安装特定版本，该软件包名称是软件包名称（docker-ce）加上版本字符串（第二列），从第一个冒号（:）一直到第一个连字符，并用连字符（-）分隔。例如：docker-ce-18.09.1。

|  |
| --- |
| sudo yum install docker-ce-<VERSION\_STRING> docker-ce-cli-<VERSION\_STRING> containerd.io |

到这里，两种安装方式就完成了，任选一种即可。

## docker启动

docker启动命令：

sudo systemctl start docker

查看版本号：

docker --version

显示版本号，及代表成功。



# docker基础命令

## 启动命令

systemctl start docker

## 查看本地镜像

docker images

## 检索仓库镜像

docker search image\_name

例如：docker search jdk

或在仓库里手动检索

仓库网站：<https://hub.docker.com/>

## 拉取仓库镜像到本地

docker pull image\_name:<version>

docker pull jdk:8

## 删除本地镜像

docker rmi image\_id

## 查看所有容器

docker ps -a

## 查看正在运行的容器

docker ps -s

## 运行镜像

docker run image\_id

## 运行容器

docker start CONTAINER\_ID

## 停止容器

docker stop CONTAINER\_ID

## 删除容器

docker rm CONTAINER\_ID

# 部署springboot项目

## 环境准备：

java镜像下载:

docker search java

这里选择JDK8版本

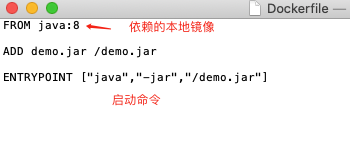
docker pull java:8

## springboot项目制作镜像

在服务器上建一个文件夹，用来放jar文件和Dockerfile文件，如下：



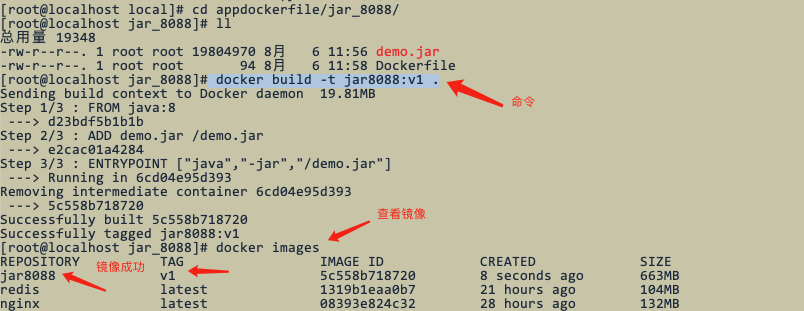
Dockerfile文件编写：



制作镜像命令：【 docker build -t jar8088:v1 .】

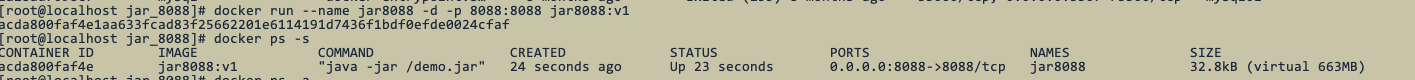
jar8088为镜像名

v1为tag



出现我们的镜像名则表示成功。

## 运行springboot镜像



命令：

|  |
| --- |
| docker run --name jar8088 -d -p 8088:8088 jar8088:v1 |

--name jar8088为容器名

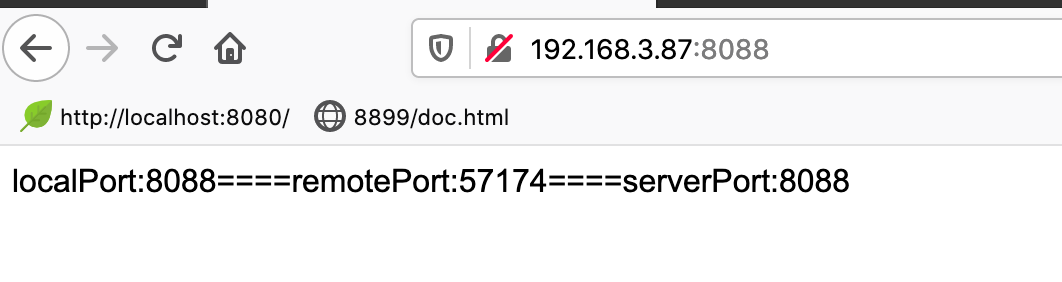
-p 8088:8088

前一个8088为外部访问的端口，服务器需要开放端口

后一个8088为项目内部的端口，及yml里的端口

通过docker ps -s查看镜像是否运行成功。

运行成功后访问方式：IP+映射端口



测试项目内的接口，访问成功即可。

# 部署Nginx

## nginx镜像准备

docker pull nginx

这里采用外部文件挂载方式进行部署。

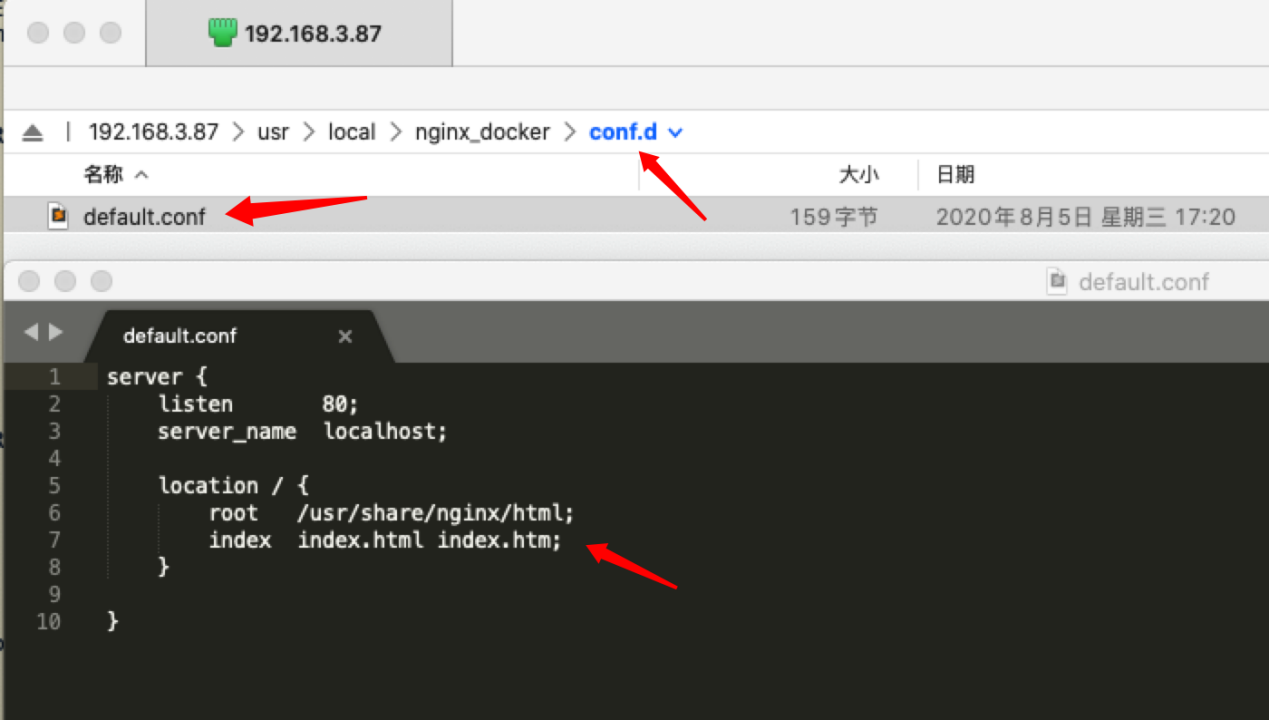
## 新建文件夹

在服务器上创建文件夹，如下：



## 编写default.conf文件

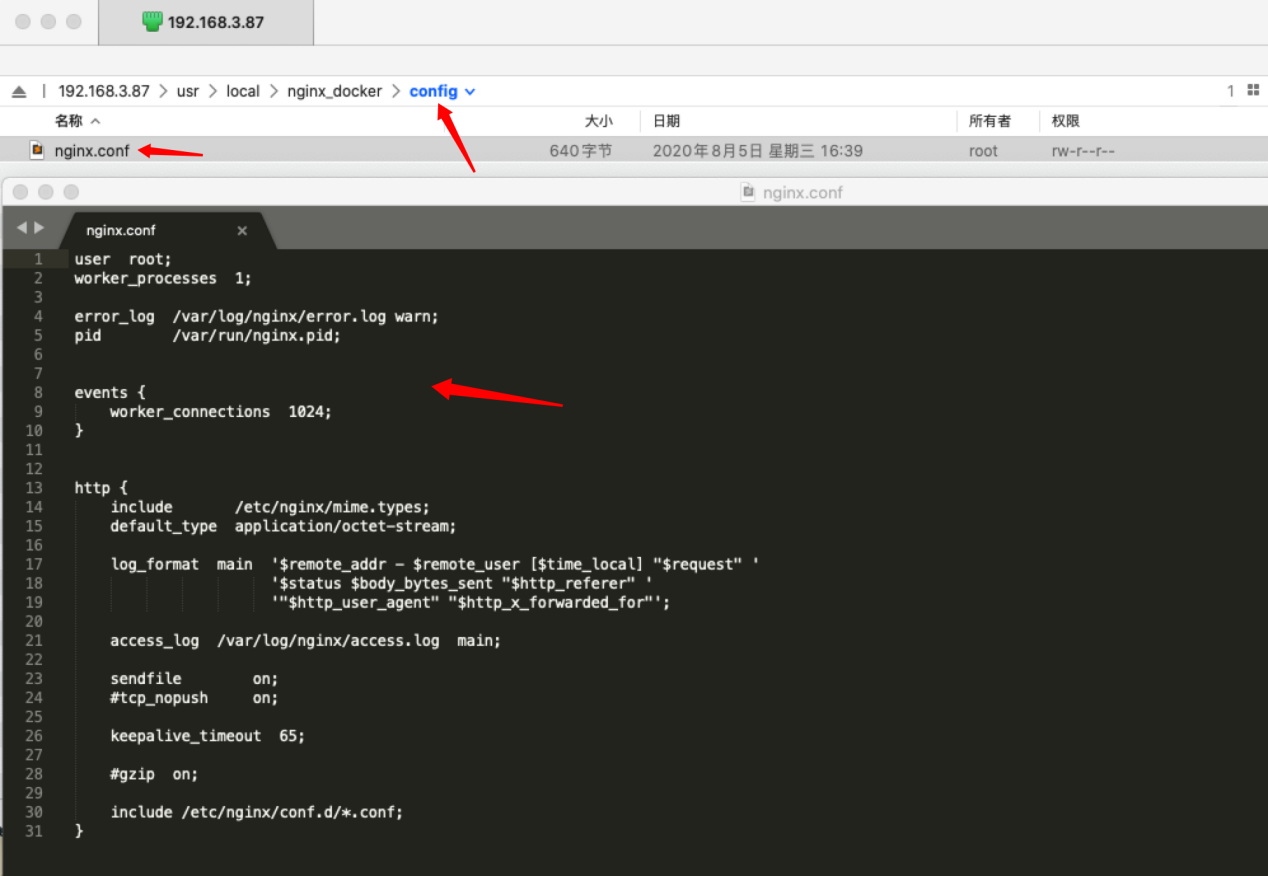
在conf.d文件夹里编写default.conf文件



|  |
| --- |
| server {  listen 80;  server\_name localhost;  location / {  root /usr/share/nginx/html;  index index.html index.htm;  }  } |

## 编写nginx.conf文件

在config文件夹里编写nginx.conf文件

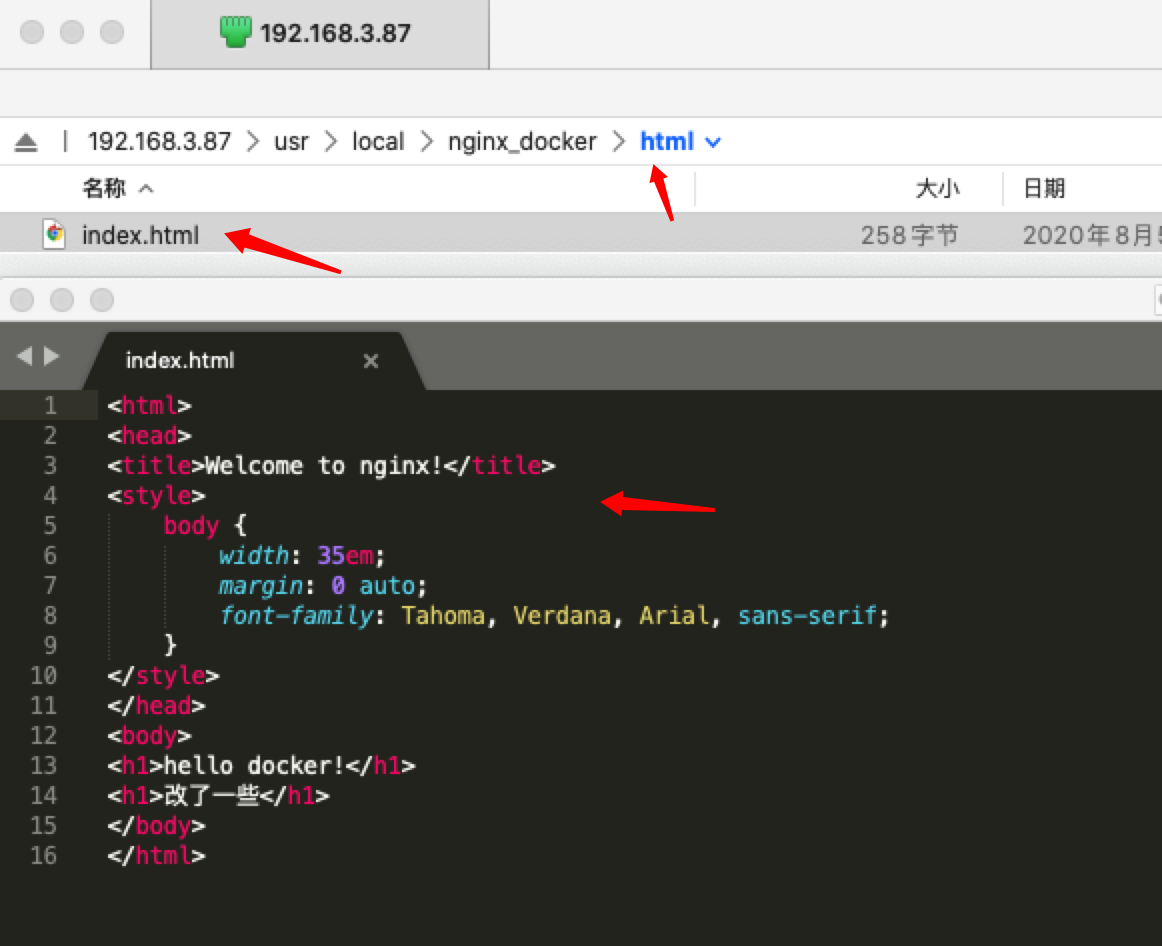


|  |
| --- |
| user root;  worker\_processes 1;  error\_log /var/log/nginx/error.log warn;  pid /var/run/nginx.pid;  events {  worker\_connections 1024;  }  http {  include /etc/nginx/mime.types;  default\_type application/octet-stream;  log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  '$status $body\_bytes\_sent "$http\_referer" '  '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  access\_log /var/log/nginx/access.log main;  sendfile on;  #tcp\_nopush on;  keepalive\_timeout 65;  #gzip on;  include /etc/nginx/conf.d/\*.conf;  } |

## 编写一个网页用于测试

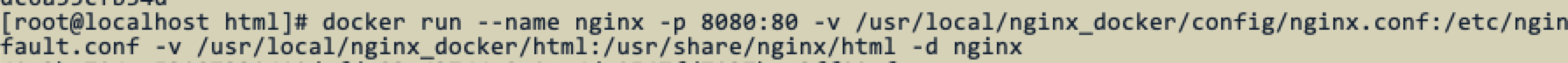
在html文件夹里编写一个网页用于测试

这个文件夹可以放Vue项目dist文件夹



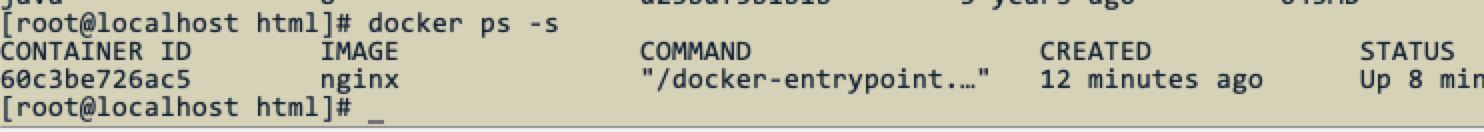
## log文件夹为日志文件夹

## 运行nginx镜像

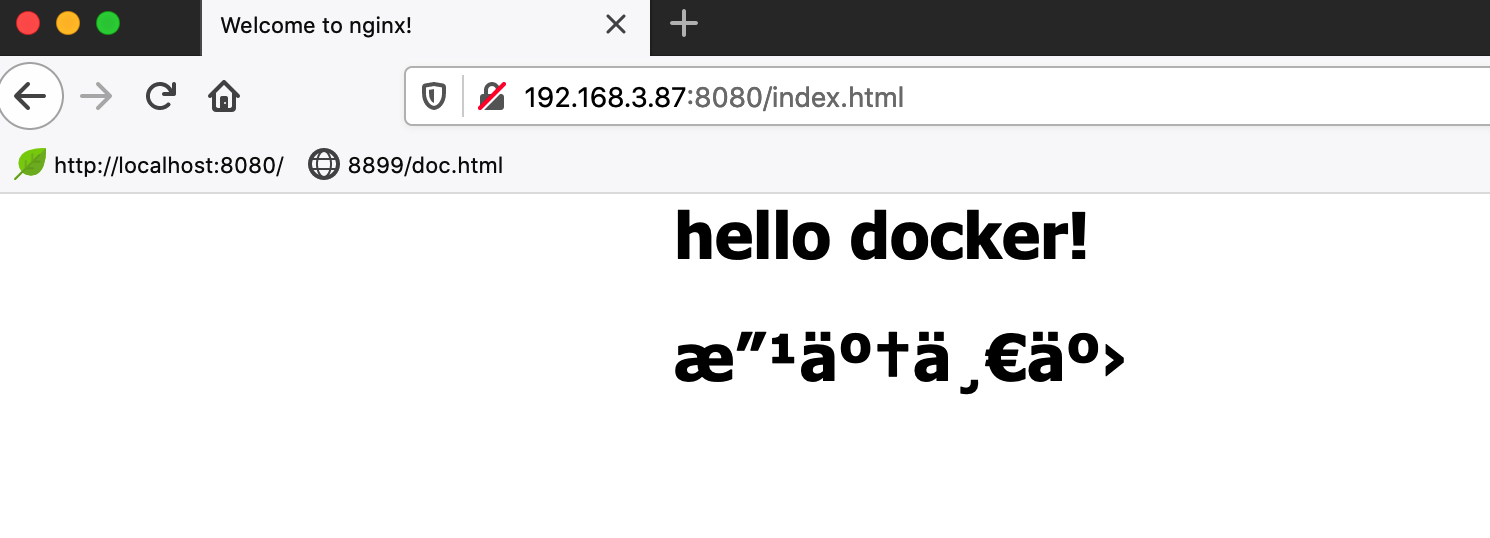


|  |
| --- |
| docker run --name nginx -p 8080:80 -v /usr/local/nginx\_docker/config/nginx.conf:/etc/nginx/nginx.conf -v /usr/local/nginx\_docker/log:/var/log/nginx -v /usr/local/nginx\_docker/conf.d/default.conf:/etc/nginx/conf.d/default.conf -v /usr/local/nginx\_docker/html:/usr/share/nginx/html -d nginx |

查看是否运行成功



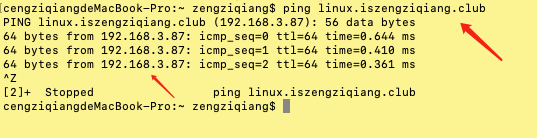
访问出现我们自己编写的网页及表示外部文件挂载成功，乱码是网页没有做处理。



# Nginx配置域名

## 前提准备

准备一个可以用的域名



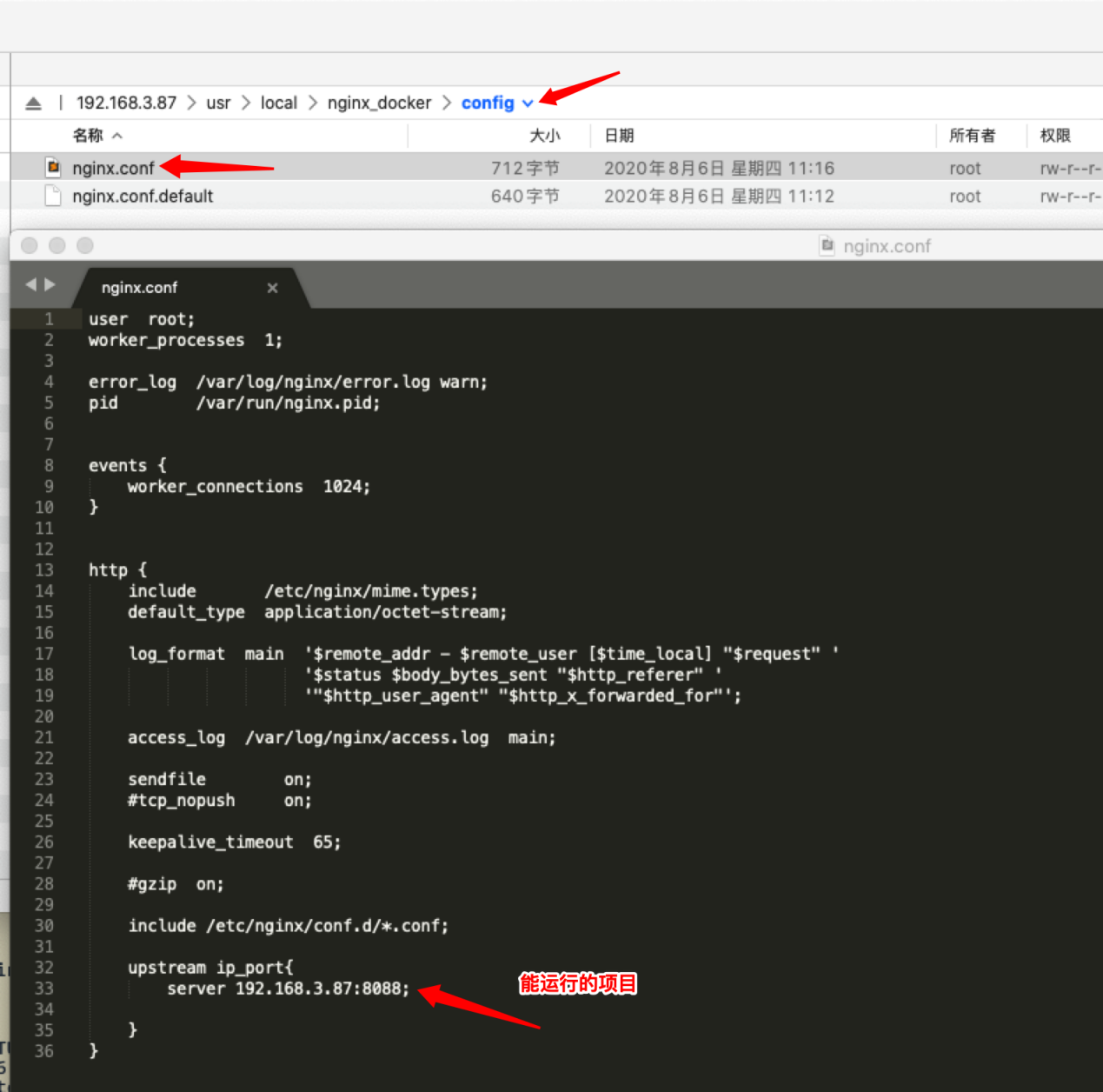
域名：linux.iszengziqiang.club

IP：192.168.3.87

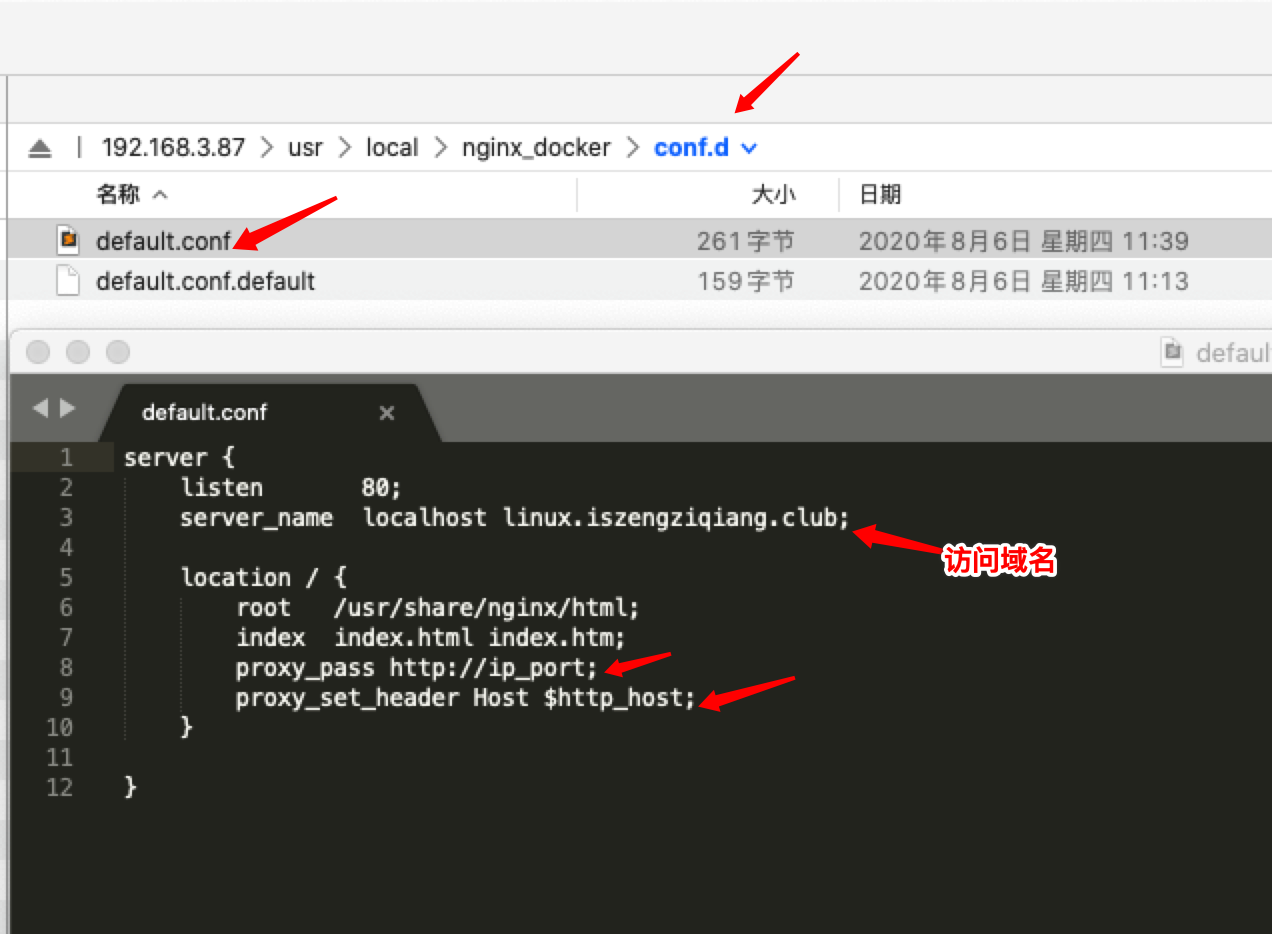
使用上面的springboot镜像进行操作。

## 修改Nginx配置文件

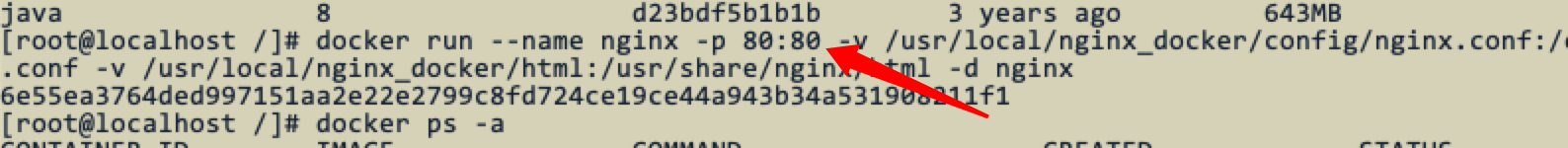
### nginx.conf修改



### default.conf修改

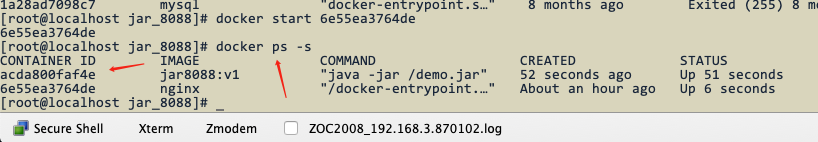


## 运行Nginx镜像



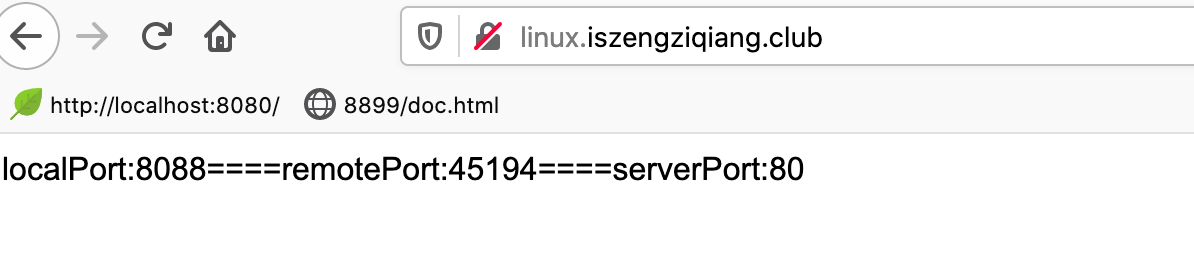
|  |
| --- |
| docker run --name nginx -p 80:80 -v /usr/local/nginx\_docker/config/nginx.conf:/etc/nginx/nginx.conf -v /usr/local/nginx\_docker/log:/var/log/nginx -v /usr/local/nginx\_docker/conf.d/default.conf:/etc/nginx/conf.d/default.conf -v /usr/local/nginx\_docker/html:/usr/share/nginx/html -d nginx |

查看两个镜像是否运行成功，如下表示成功：



## 使用域名访问

域名访问成功，与springboot镜像的访问一样。



# 部署MySQL

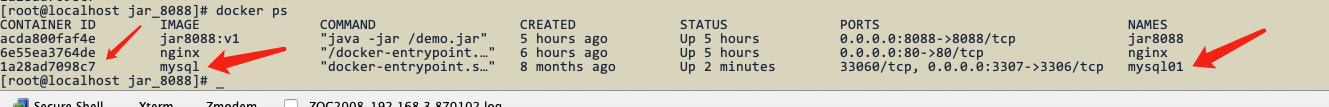
## MySQL镜像下载

search pull mysql

## 运行mysql镜像

|  |
| --- |
| docker run --name mysql01【实例名】-p 3307:3306【Linux操作系统的端口映射 : docker的端口】 -e MYSQL\_ROOT\_PASSWORD=123456【密码】 -d mysql【镜像名】 --character-set-server=utf8mb4 --collation-server=utf8mb4\_unicode\_ci |
| 因为mysql默认端口为3306，现在修改的3307为我们等会客户端连接的端口 |

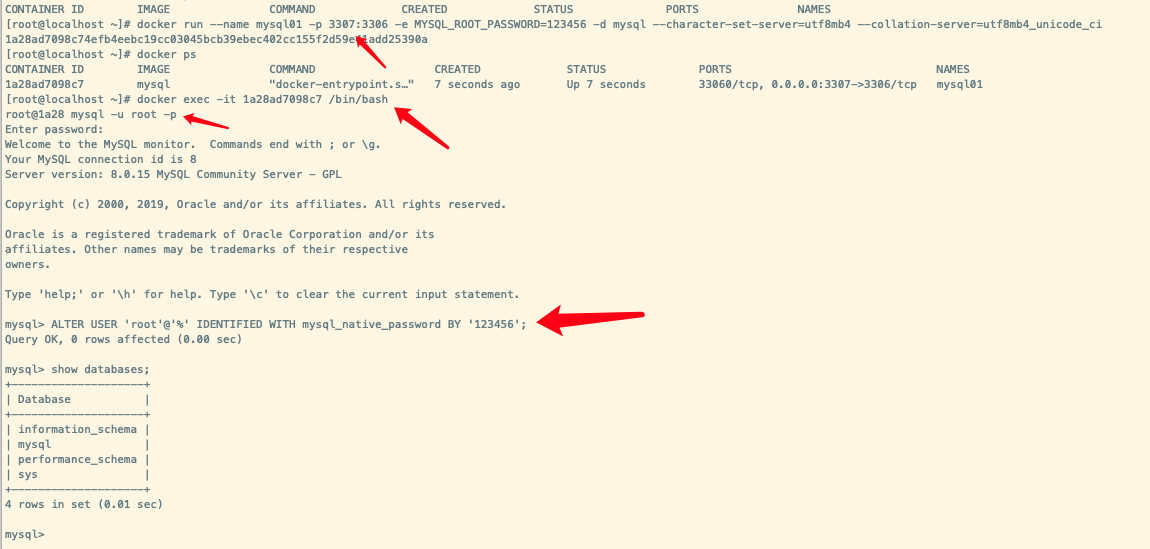
镜像运行成功：



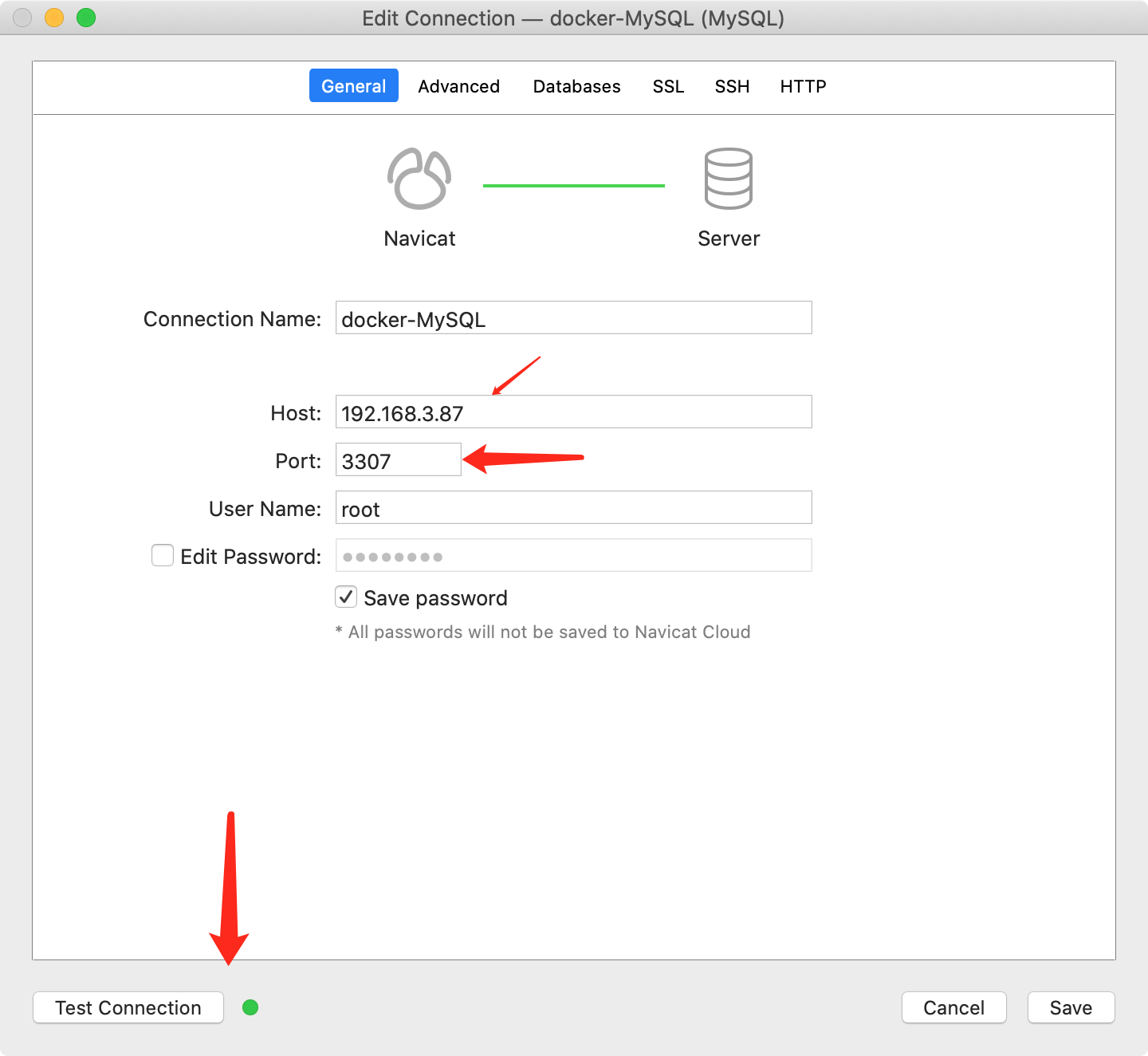
## 客户端连接失败，解决远程连接报错

1，docker exec -it [容器ID] /bin/bash

2, ALTER USER ‘root’@’%’ IDENTIFIED WITH mysql\_native\_password BY ‘123456’;



正常运行连接成功



客户端正常连接，项目里正常配置即可使用。

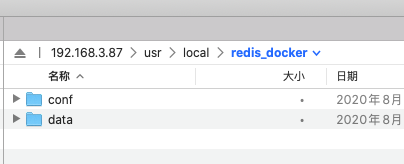
# 部署Redis

## Redis镜像下载

docker pull redis

采用外部挂载方式部署

## 创建文件夹



## 编写redis.conf文件

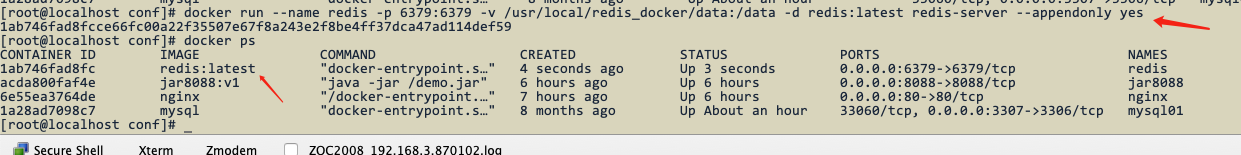
复制一份redis.conf文件即可

修改部分：

1. 注释掉：#daemonize yes 否则无法启动容
2. 设置requirepass 123456

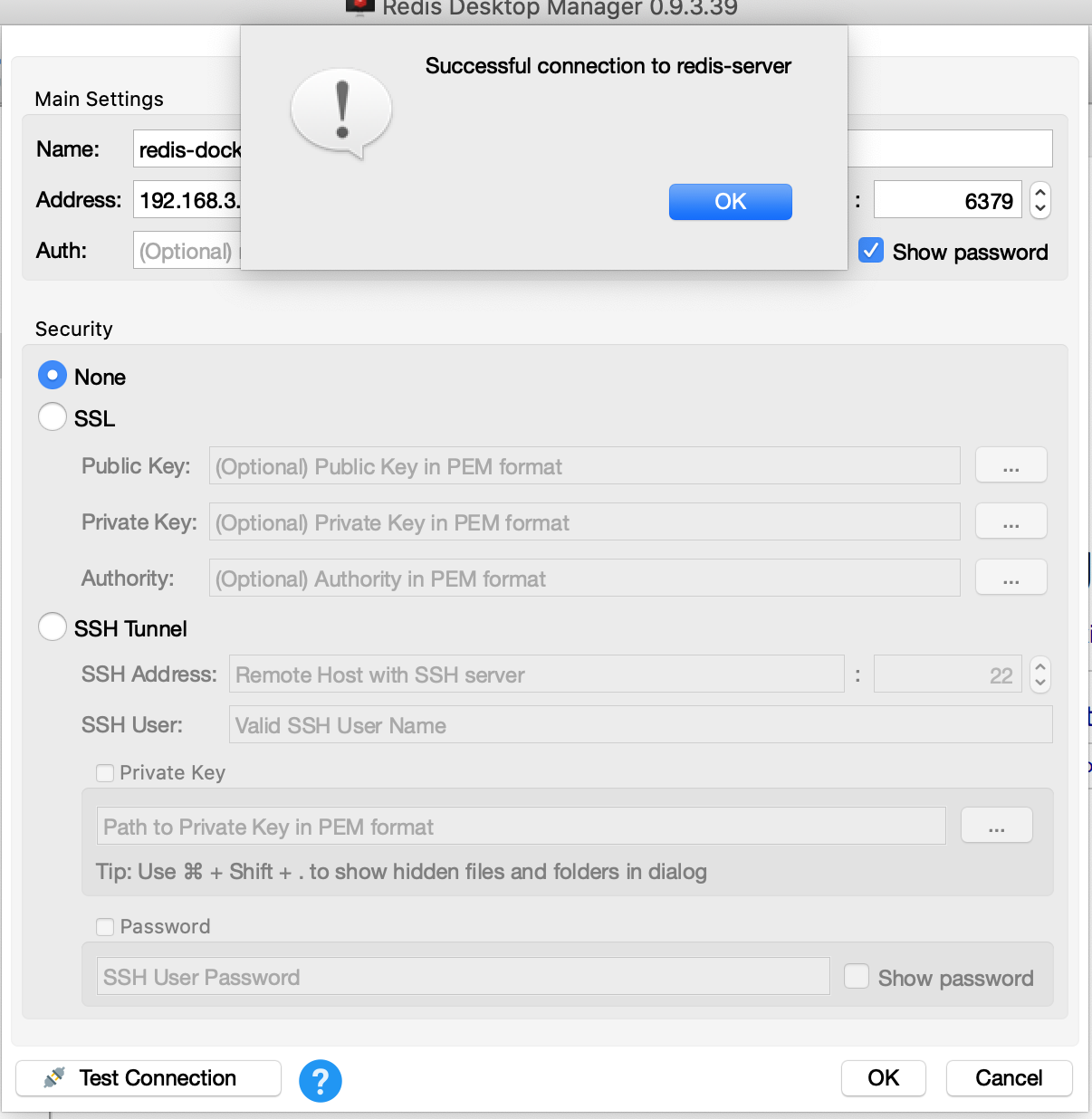
## 运行Redis镜像

|  |
| --- |
| docker run --name redis -p 6379:6379 -v /usr/local/redis\_docker/data:/data -d redis:latest redis-server --appendonly yes |
|  |



客户端访问如下，连接成功。IP+6379

访问密码可以自行设置。



配置文件和日志文件都外部挂载方式：

|  |
| --- |
| docker run --name redis -p 6379:6379 -v /usr/local/redis\_docker/conf/redis.conf:/etc/redis/redis.conf -v /usr/local/redis\_docker/data:/data -d redis:latest redis-server /etc/redis/redis.conf --requirepass "123456" --appendonly yes |

## 部署在209上

修改配置文件：

bind 0.0.0.0

protected-mode no【设置为No】

port 6379

databases 16

#daemonize yes【注释掉】

requirepass 【设置密码】

appendonly yes【aof持久化】

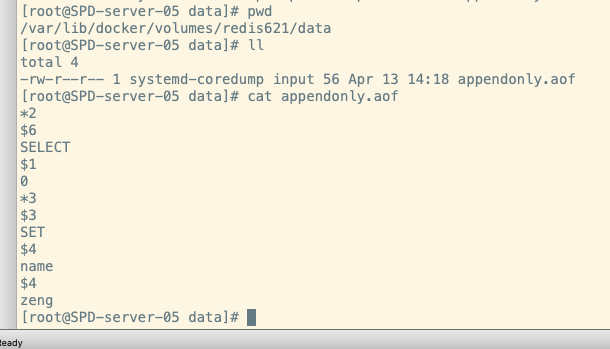
创建文件夹：

创建/var/lib/docker/volumes/redis621/config文件夹路径

和/var/lib/docker/volumes/redis621/data

运行命令：

|  |
| --- |
| docker run -d --privileged=true -p 6379:6379 -v /var/lib/docker/volumes/redis621/config/redis.conf:/etc/redis/redis.conf -v /var/lib/docker/volumes/redis621/data:/data --name redis621dev redis:6.2.1 redis-server /etc/redis/redis.conf --appendonly yes |



持久化成功。

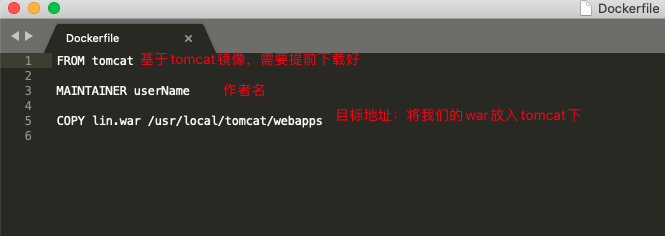
# 部署war包项目

## 前提准备

下载Tomcat镜像

docker pull tomcat

## 编写war包的Dockerfile文件



## 生成镜像

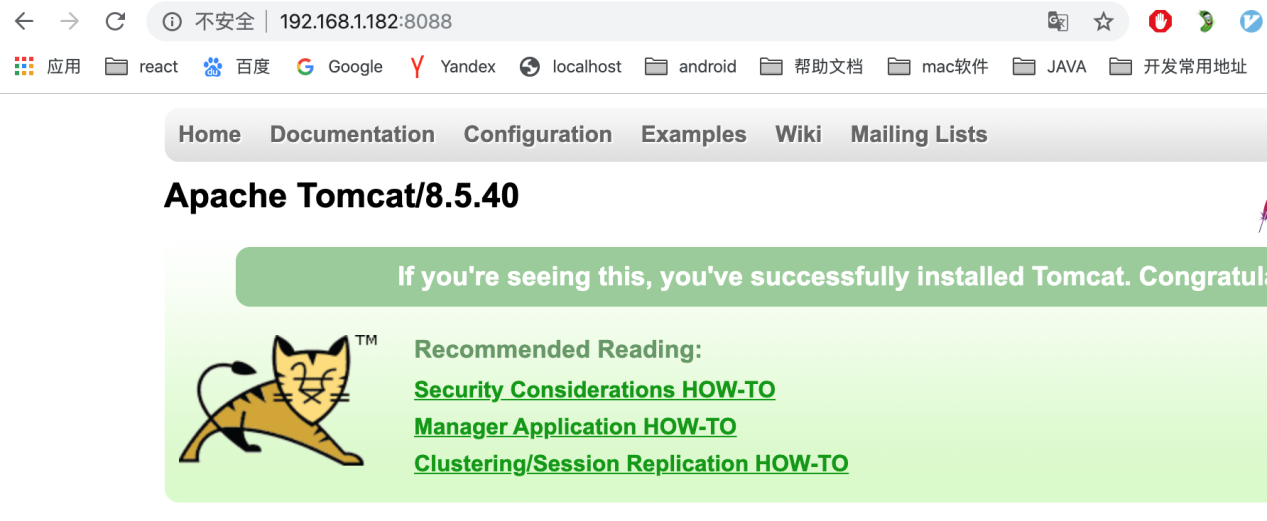
docker build -t lin:latest



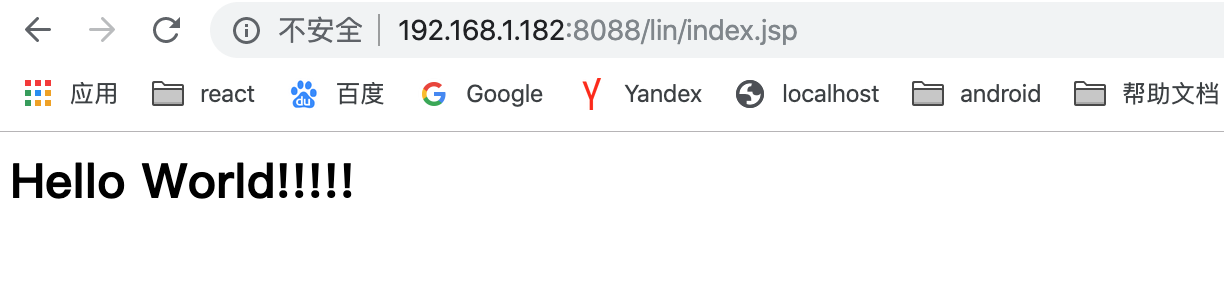
## 运行war包镜像

docker run -d -p 8088:8080 lin

外网访问如下，表示成功



访问war包项目内的接口或界面



war包镜像运行成功。

# 部署elasticsearch

这里以6.8.0版本为例

## 镜像下载

docker pull elasticsearch:6.8.0

## 运行镜像

ElasticSearch的默认端口是9200，我们把宿主环境9200端口映射到Docker容器中的9200端口，就可以访问到Docker容器中的ElasticSearch服务了，同时我们把这个容器命名为es。

|  |
| --- |
| docker run -d --name es -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" elasticsearch:6.8.0 |

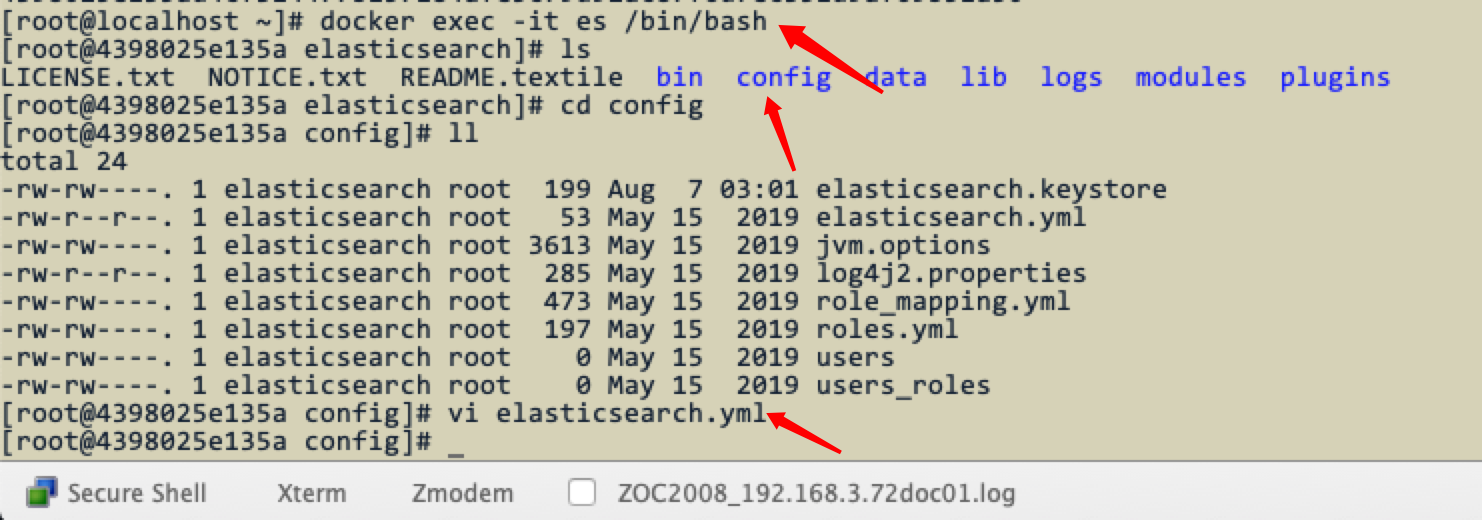
## 配置跨域

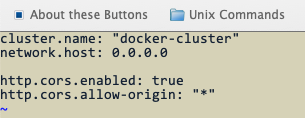
进入容器：

|  |
| --- |
| docker exec -it es /bin/bash |

编辑elasticsearch.yml，添加如下：

|  |
| --- |
| # 加入跨域配置  http.cors.enabled: true  http.cors.allow-origin: "\*" |

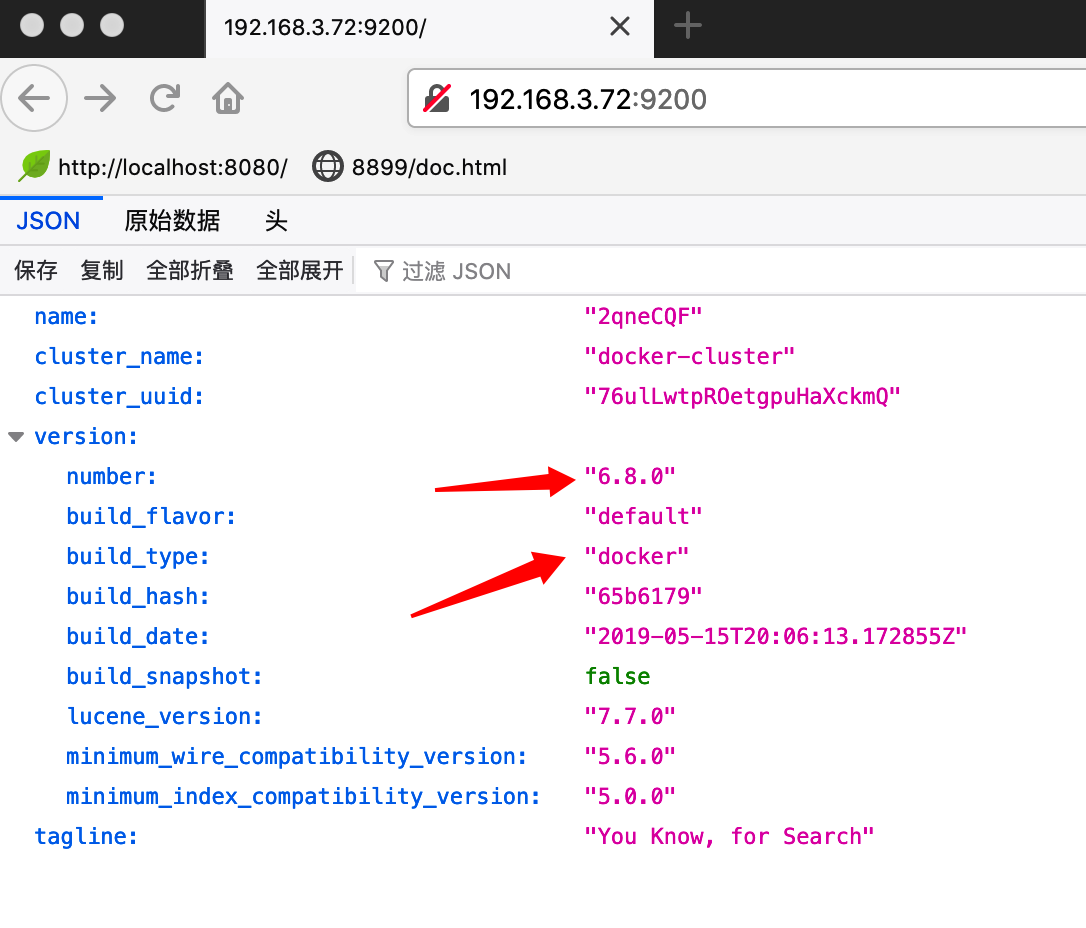




## 重启容器

|  |
| --- |
| docker restart es |

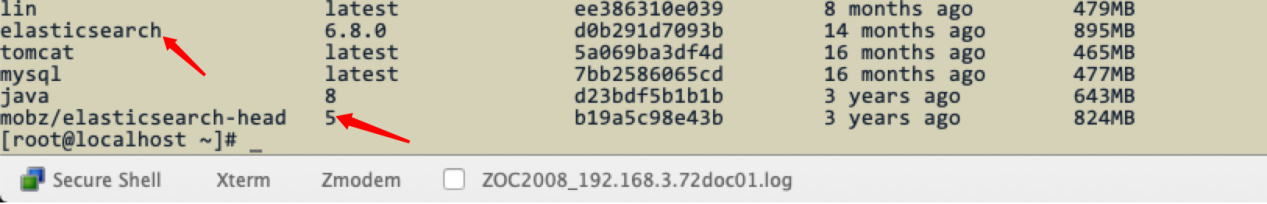
外网访问如下表示成功



## 部署elasticsearch-head

### 拉取镜像

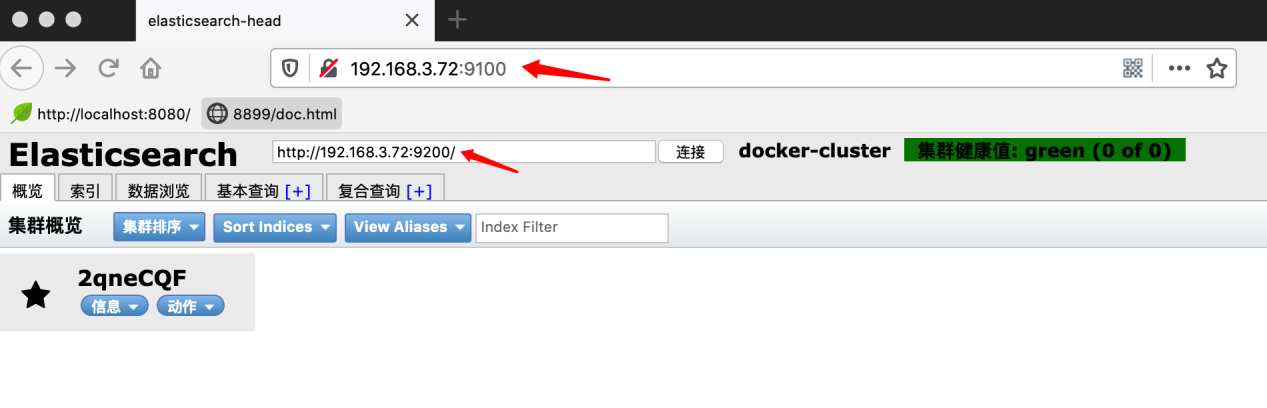
|  |
| --- |
| docker pull mobz/elasticsearch-head:5 |



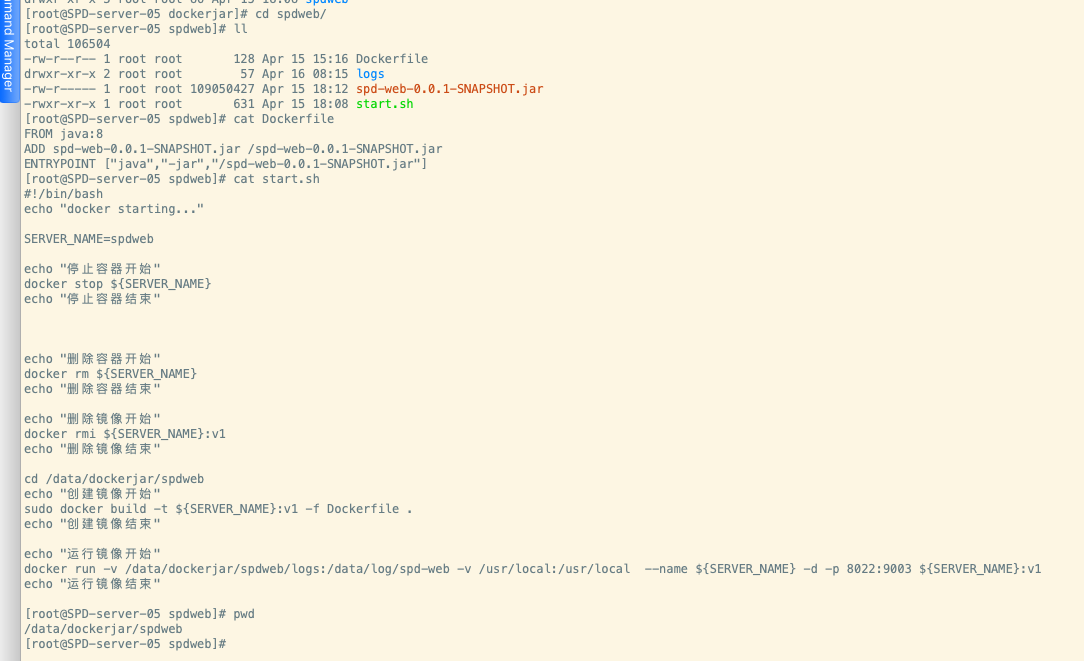
### 运行镜像

|  |
| --- |
| docker run -d --name es\_admin -p 9100:9100 mobz/elasticsearch-head:5 |

显示如下表示成功



# docker 部署springboot项目，并数据卷外部文件夹



## 进入/data/dockerjar/spdweb文件夹

### 创建Dockerfile文件

|  |
| --- |
| FROM java:8  ADD spd-web-0.0.1-SNAPSHOT.jar /spd-web-0.0.1-SNAPSHOT.jar  ENTRYPOINT ["java","-jar","/spd-web-0.0.1-SNAPSHOT.jar"] |

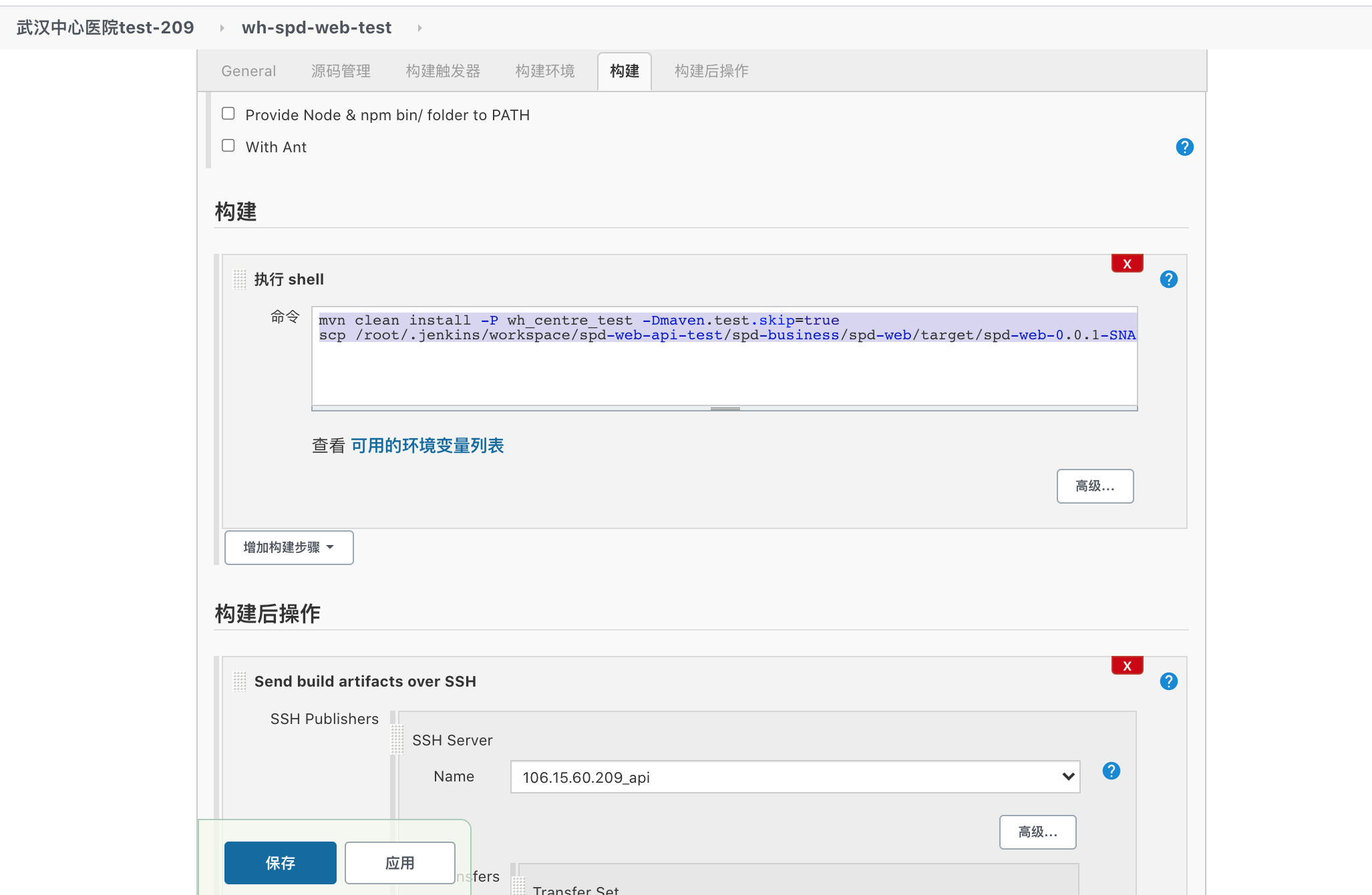
### 创建start.sh文件

|  |
| --- |
| #!/bin/bash  echo "docker starting..."  SERVER\_NAME=spdweb  echo "停止容器开始"  docker stop ${SERVER\_NAME}  echo "停止容器结束"  echo "删除容器开始"  docker rm ${SERVER\_NAME}  echo "删除容器结束"  echo "删除镜像开始"  docker rmi ${SERVER\_NAME}:v1  echo "删除镜像结束"  cd /data/dockerjar/spdweb  echo "创建镜像开始"  sudo docker build -t ${SERVER\_NAME}:v1 -f Dockerfile .  echo "创建镜像结束"  echo "运行镜像开始"  docker run -v /data/dockerjar/spdweb/logs:/data/log/spd-web -v /usr/local:/usr/local --name ${SERVER\_NAME} -d -p 8022:9003 ${SERVER\_NAME}:v1  echo "运行镜像结束" |

这里有数据卷外部地址。

给sh赋操作权限 【chmod u+x start.sh】

## Jenkins使用



|  |
| --- |
| mvn clean install -P wh\_centre\_test -Dmaven.test.skip=true  scp /root/.jenkins/workspace/spd-web-api-test/spd-business/spd-web/target/spd-web-0.0.1-SNAPSHOT.jar root@106.15.60.209:/data/dockerjar/spdweb |

将jar文件传到指定文件夹路径

