**docker**

参考视频：<https://www.bilibili.com/video/av48979566>

外部修改Tomcat配置文件：<https://blog.csdn.net/wfq784967698/article/details/103780236>

# 使用流程

1，先安装docker

2，安装所需的镜像，以MySQL为例，docker pull mysql

查看本地已经安装的镜像：docker images

搜索需要安装的镜像：docker search mysql

3,创建容器，通过我们安装好的mysql镜像进行创建容器

4，查看已经创建好的容器：docker ps -a

5，启动容器：docker start 88dab2f338c6【容器ID】

6，查看正在运行的容器：docker ps -s

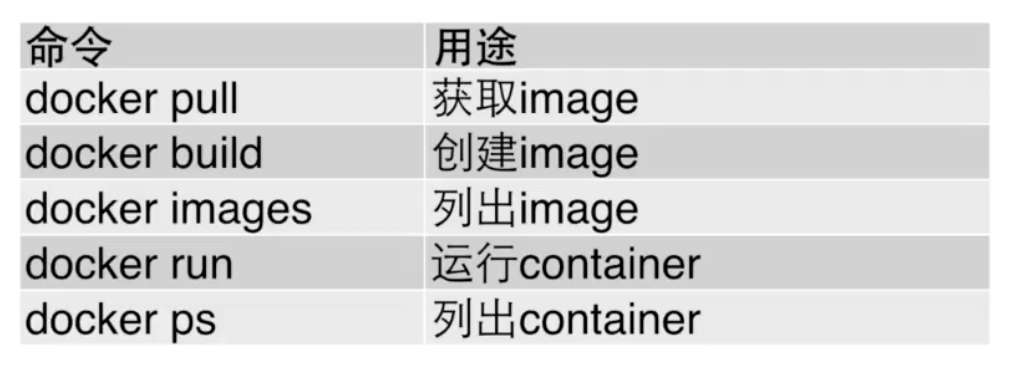
查看以前运行过的容器：docker ps -a

7,删除容器：docker rm xxx

8，删除镜像：docker rmi <image id>

概述：【先镜像------------>再容器--------------->再启动容器】

常用命令：





# docker安装

保证apt-get是最新版本：apt-get update[ubuntu版本]

进入Linux系统，任意文件夹下，使用命令：

yun install docker

查询docker版本

docker -v

# docker查看镜像

docker镜像查询地址：<https://hub.docker.com/>

docker images

报错：Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

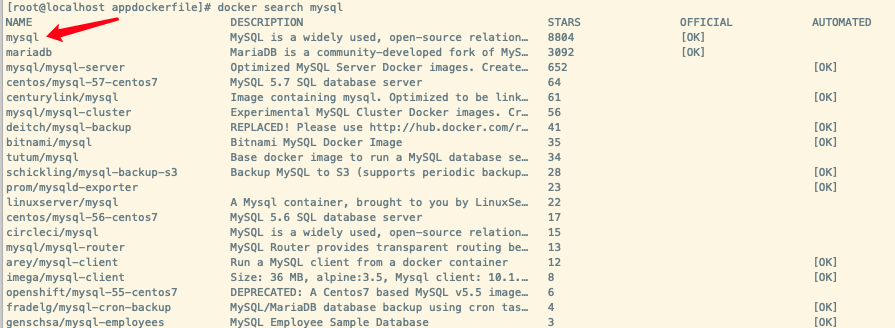
先要启动docker：systemctl start docker

docker images 为查看docker的本地自己已经安装好的镜像，可以删除

docker searche mysql 搜索所需的镜像

安装镜像：以mysql为例

docker pull mysql



# 启动docker

systemctl start docker

# 创建实例

什么是创建实例，以我们安装好的mysql为例，要创建实例后，mysql才能正常使用。

mysql创建实例的方法：<https://hub.docker.com/_/mysql>

系统默认：$ docker run --name some-mysql -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql:tag --character-set-server=utf8mb4 --collation-server=utf8mb4\_unicode\_ci

修改：$ docker run --name mysql01【实例名】 -e MYSQL\_ROOT\_PASSWORD=123456【密码】 -d mysql【镜像名】 --character-set-server=utf8mb4 --collation-server=utf8mb4\_unicode\_ci

【】为我们修改的地方



启动多个，把容器名字改一下，就可以启动多个容器，可以用于测分布式集群操作。

# 配置外部端口

容器里的端口外部不能直接使用，相当于隔离了三层，还有Linux一层。

停止容器：docker stop 【容器名】

然后删除容器

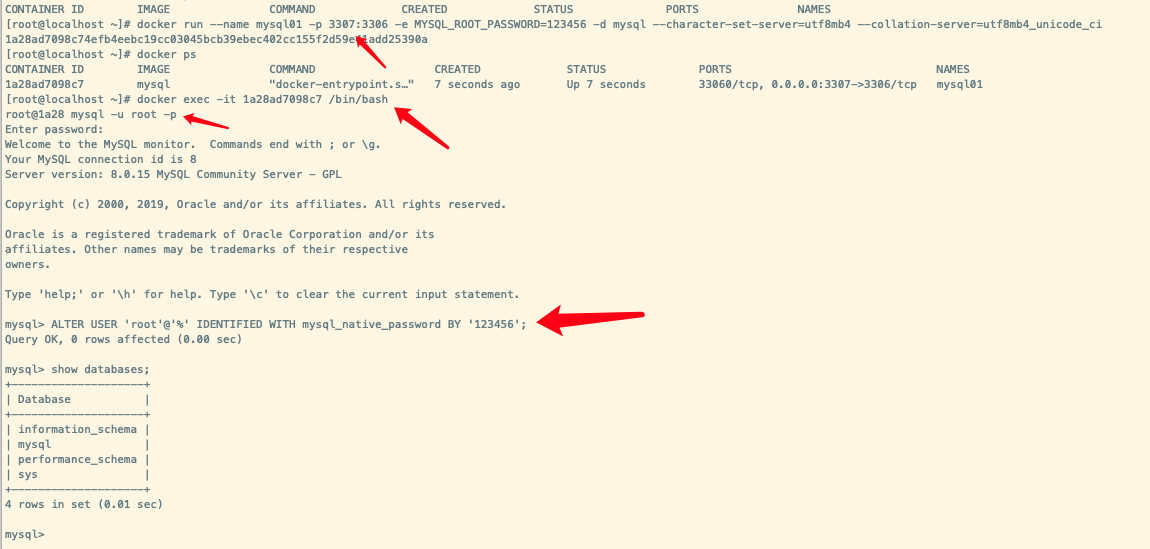
创建外部能访问的容器：

|  |
| --- |
| 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 |

再次连接MySQL会失败，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’;



好了，这样就可以正常连接了。

# docker部署war包

<https://blog.csdn.net/xiaoye319/article/details/90515996>

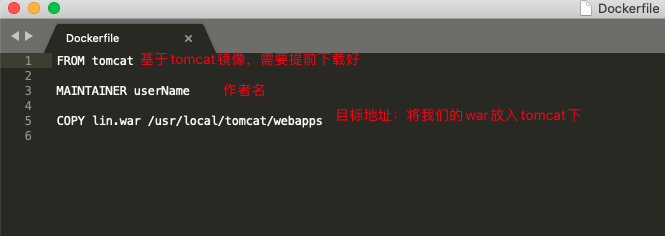
# docker运行Nginx

docker run -d -p 8080:80【外网访问端口：Nginx默认端口】 【容器名】

# 创建war镜像

1,创建一个文件夹，存放自己要发布的war包

2,创建Dockerfile文件



3，编译dockerfile文件，就会生成镜像

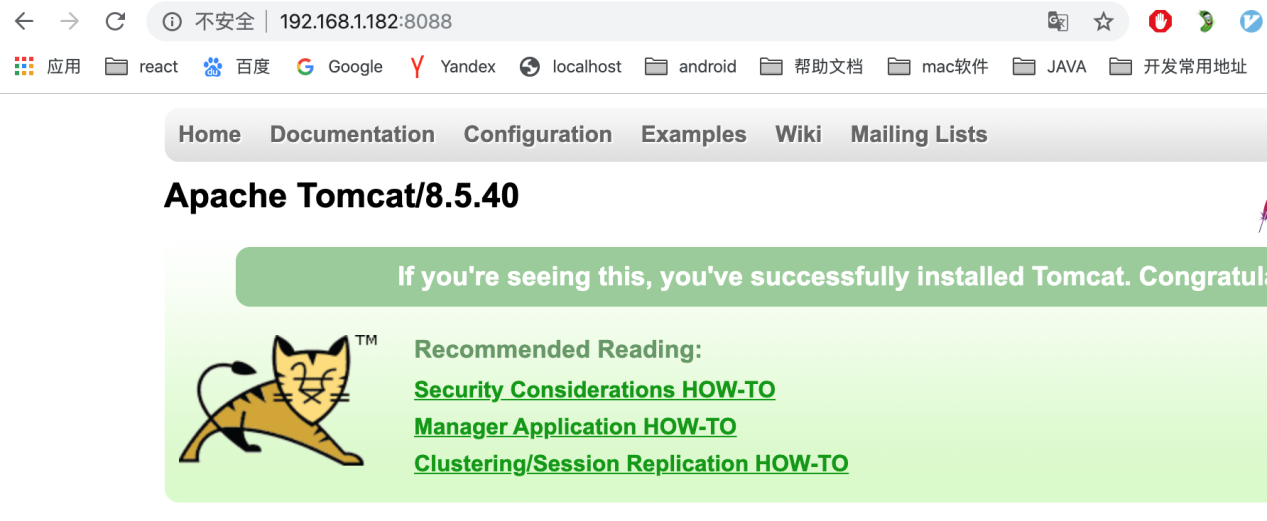
docker build -t lin:latest【镜像名：标签或版本号】



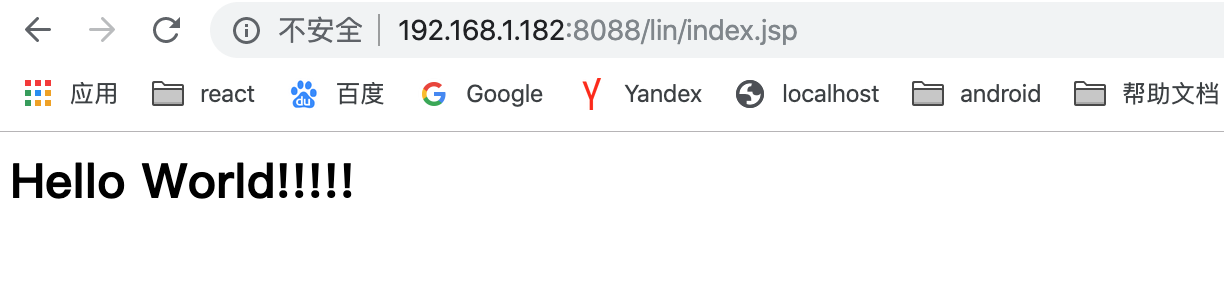
4，运行镜像

docker run -d -p 8088:8080 lin 【外网访问端口：tomcat默认端口】

5，外网访问tomcat,如下表示成功



访问我们自己的项目，如下表示成功



检查端口是否可用：

netstat -na|grep 8088

处于listen表示正常使用

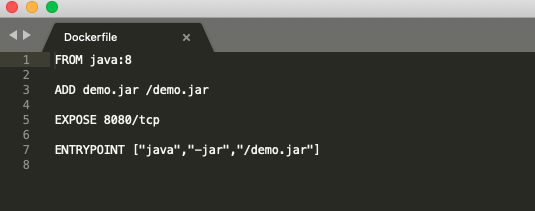
# 创建jar镜像

<https://www.cnblogs.com/shamo89/p/9201513.html>

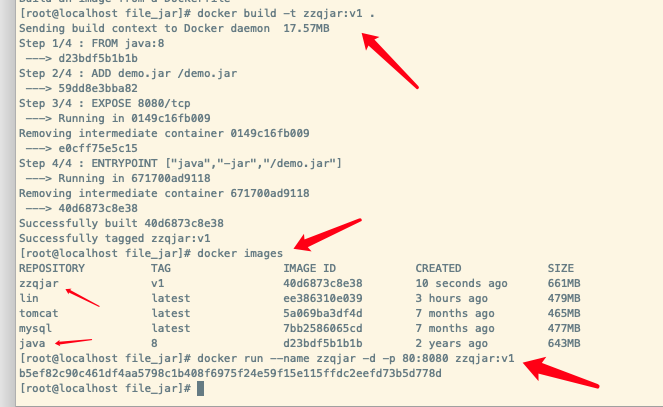
<https://www.jianshu.com/p/397929dbc27d>

1,上传jar包

2，编写Dockerfile文件



3，创建镜像

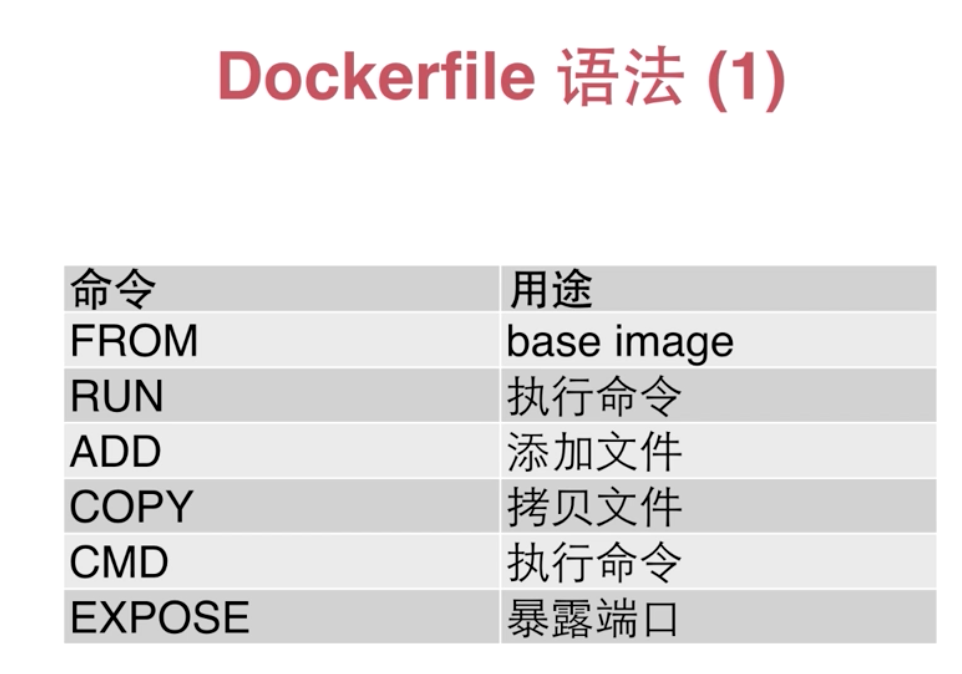


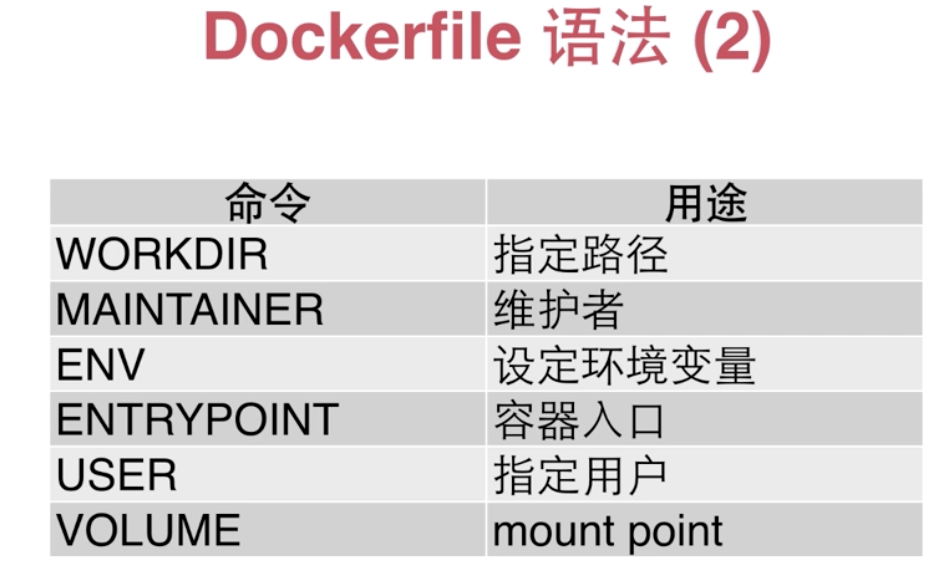
4，运行容器

5，查看项目运行日志

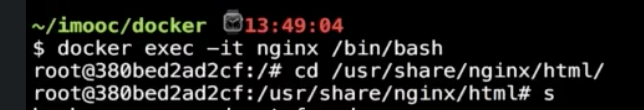


# dockerfile文件常用命令

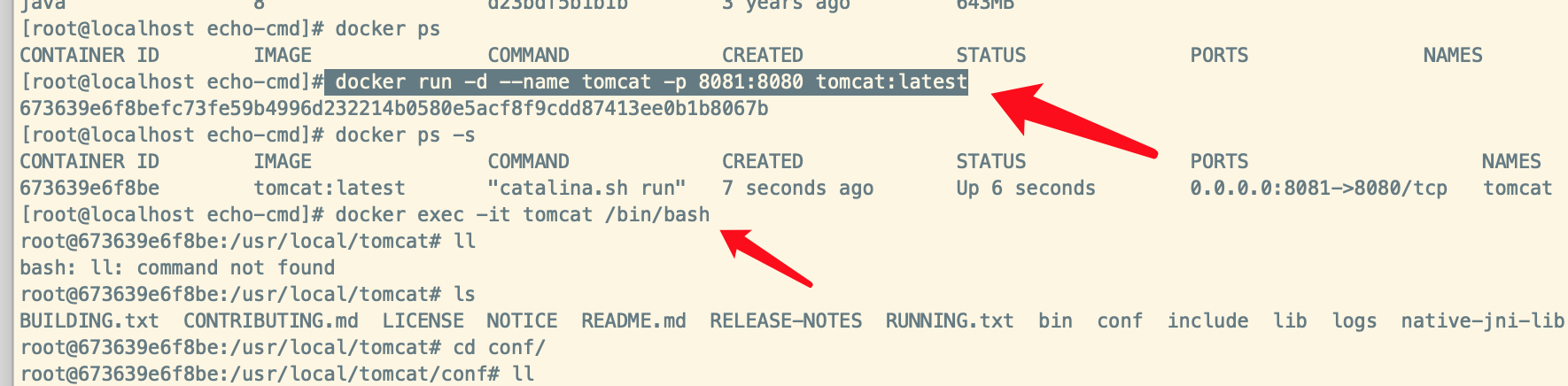




# 进入容器内部文件



# 运行Tomcat



# 上传自己的镜像

<https://www.hangge.com/blog/cache/detail_2409.html>

# nginx外部挂载

Nginx挂载外部文件： <https://blog.csdn.net/hnw13938056090/article/details/105782931>

外部挂载，将Nginx里的文件进行替换，方便修改

## 在服务器上创建文件夹

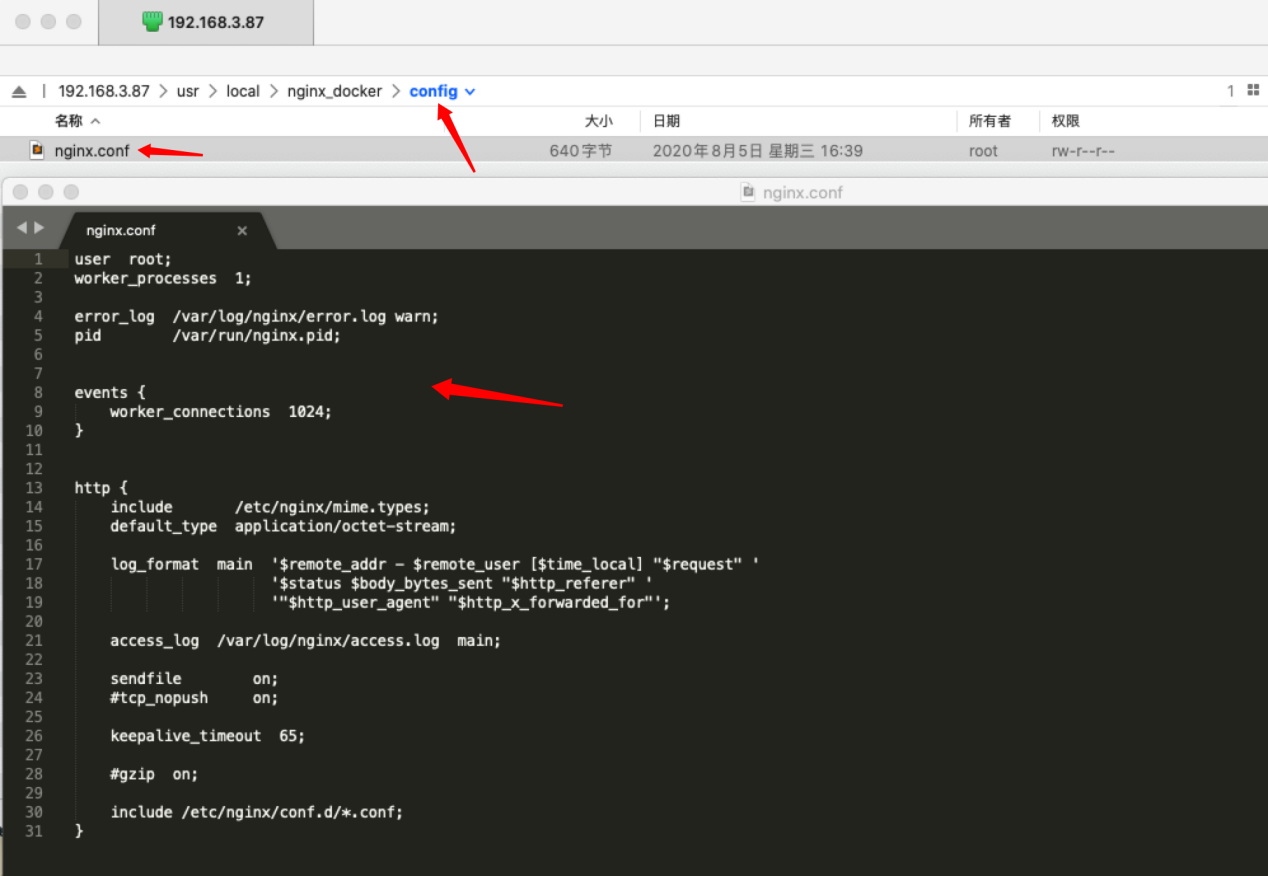
在服务器上创建用于替换的文件夹，如下：



### 1

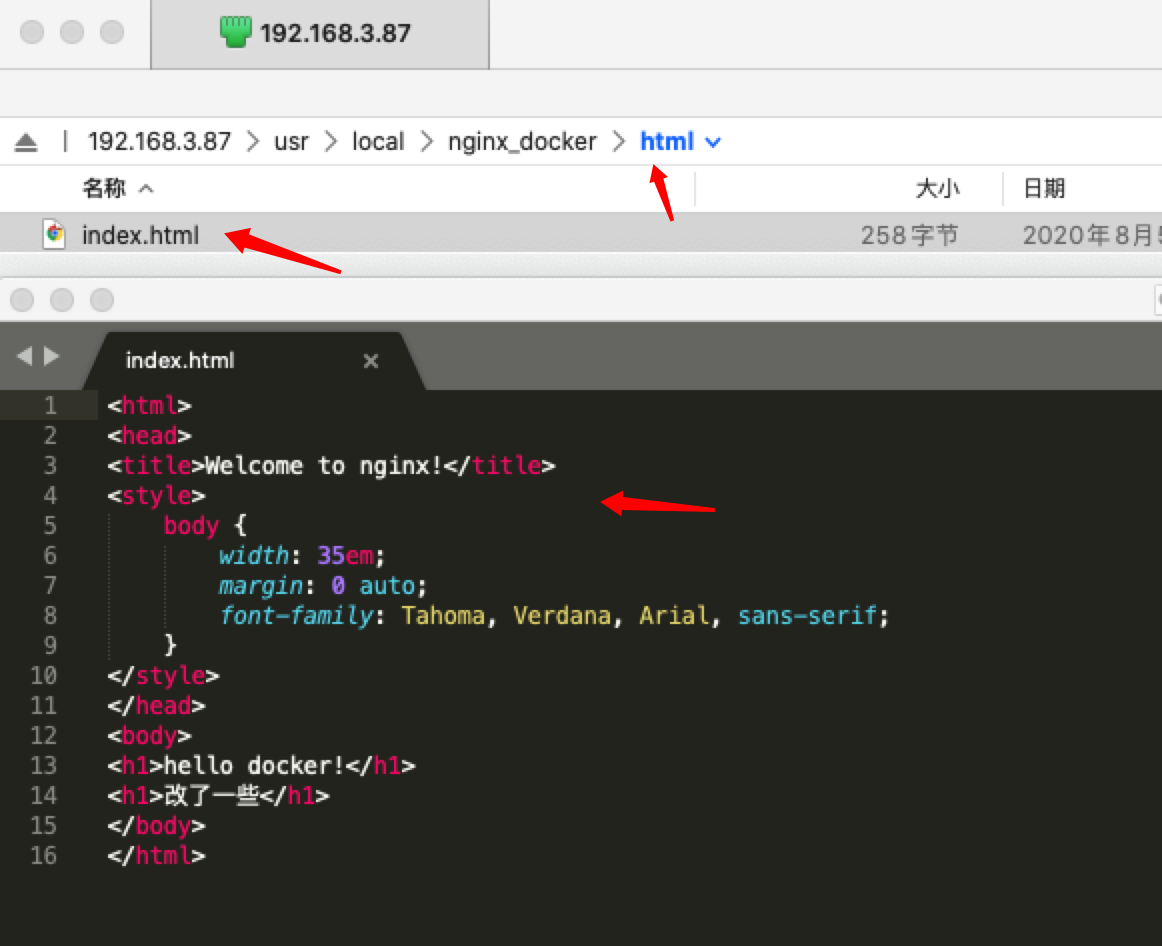
|  |
| --- |
| server {  listen 80;  server\_name localhost;  location / {  root /usr/share/nginx/html;  index index.html index.htm;  }  } |

### 2



|  |
| --- |
| 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;  } |

### 3



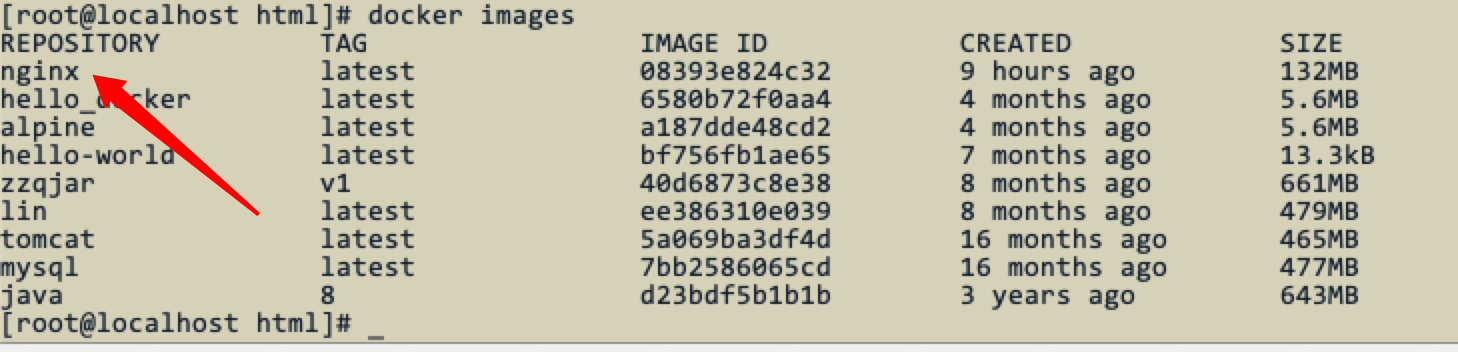
|  |
| --- |
| <html>  <head>  <title>Welcome to nginx!</title>  <style>  body {  width: 35em;  margin: 0 auto;  font-family: Tahoma, Verdana, Arial, sans-serif;  }  </style>  </head>  <body>  <h1>hello docker!</h1>  <h1>改了一些</h1>  </body>  </html> |

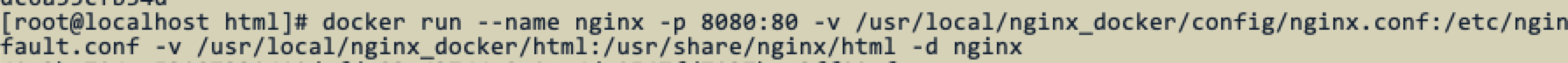
### 4

log文件夹

## 运行镜像

命令

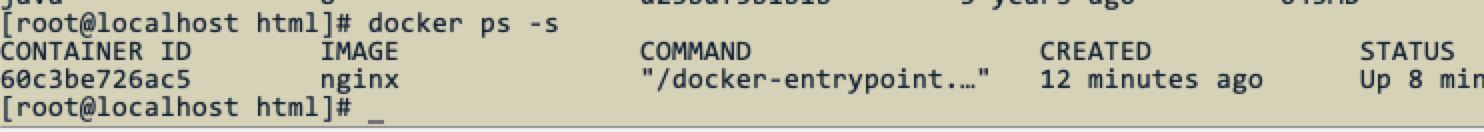




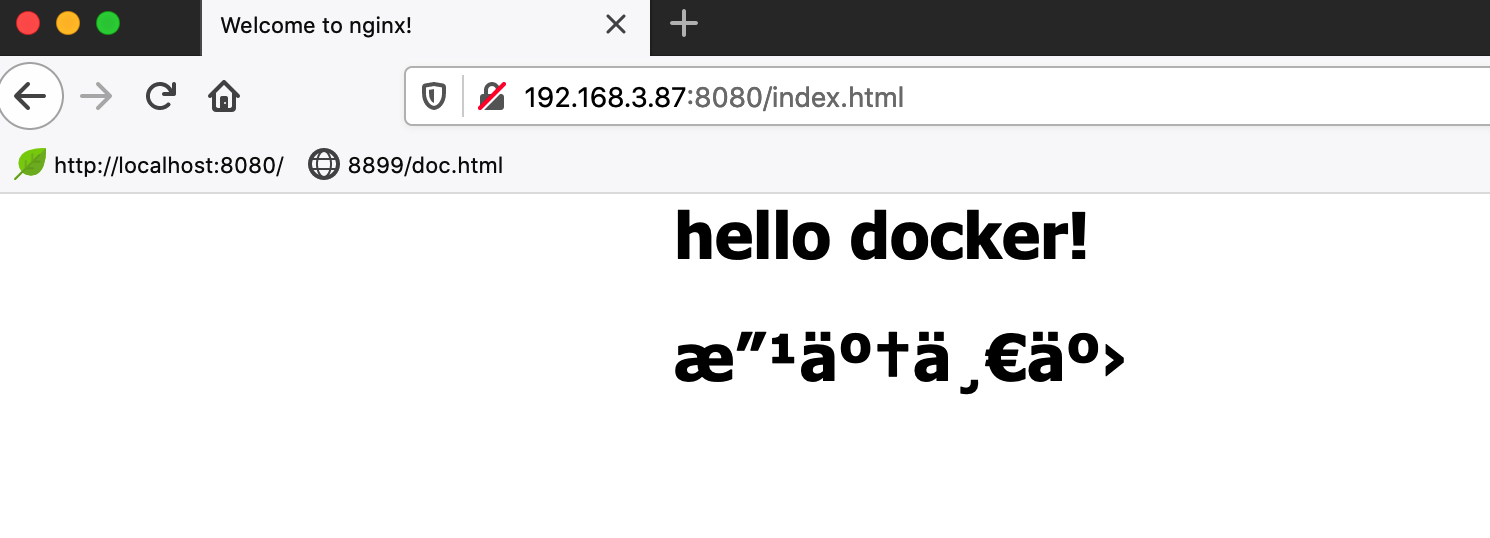
|  |
| --- |
| docker run --name nginx -p 8080:80 -v /usr/local/nginx\_docker/config/nginx.conf:/etc/nginx/nginx.conf -v /usr/local/nginx\_docker/log:/var/log/nginx -v /usr/local/nginx\_docker/conf.d/default.conf:/etc/nginx/conf.d/default.conf -v /usr/local/nginx\_docker/html:/usr/share/nginx/html -d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

运行命令注意空格

运行成功：



访问成功的样子

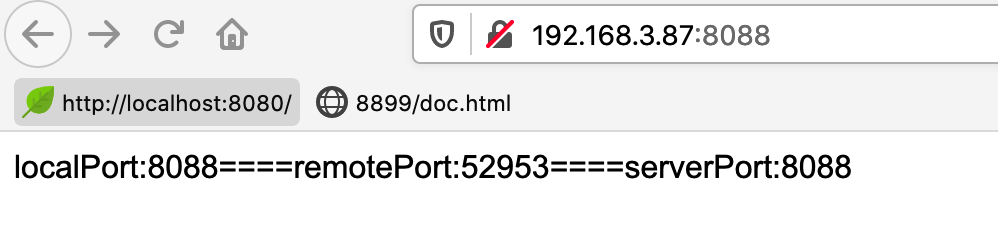


修改配置文件后，直接restart一下就好了

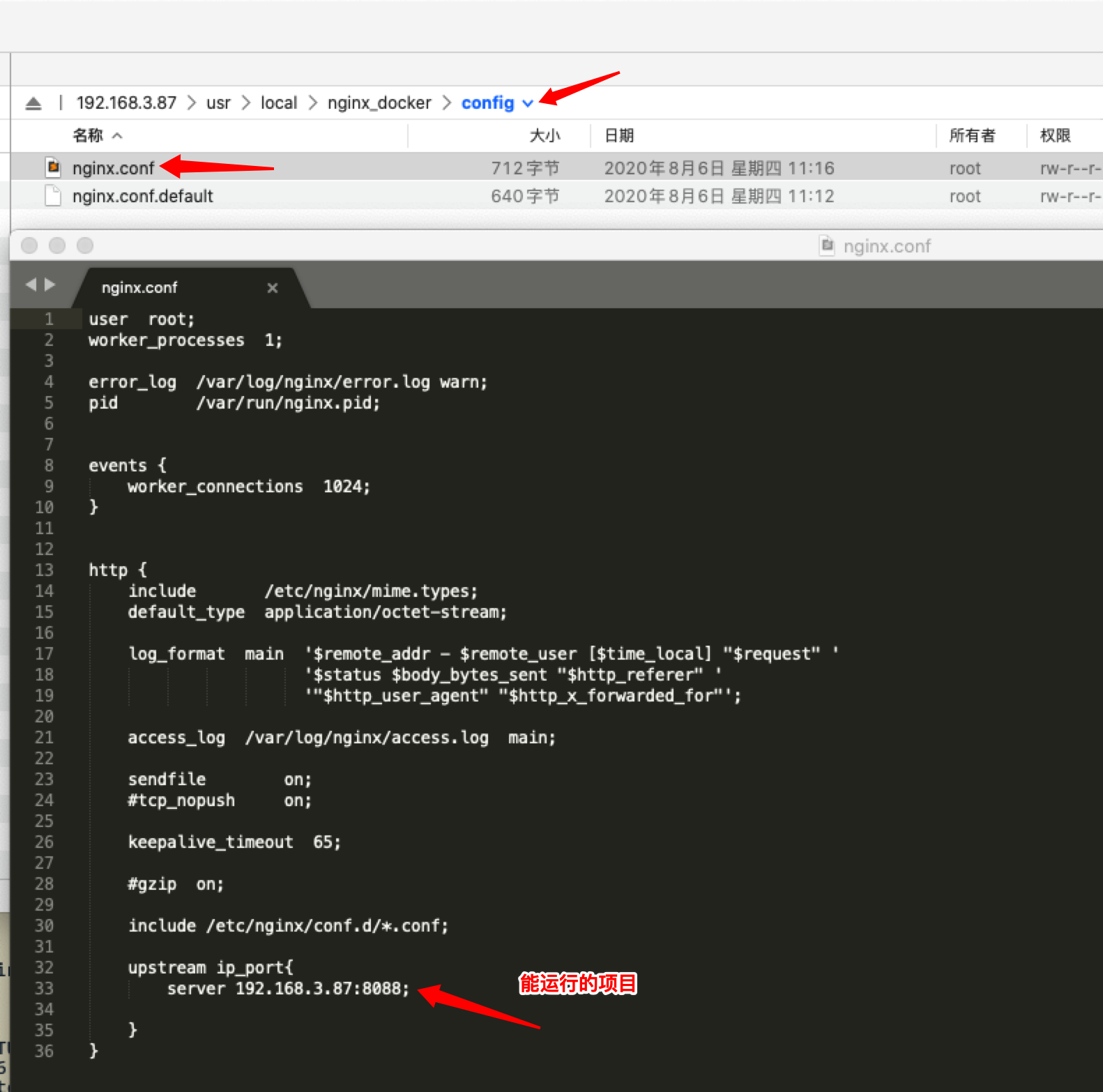
docker restart 60c3be726ac5

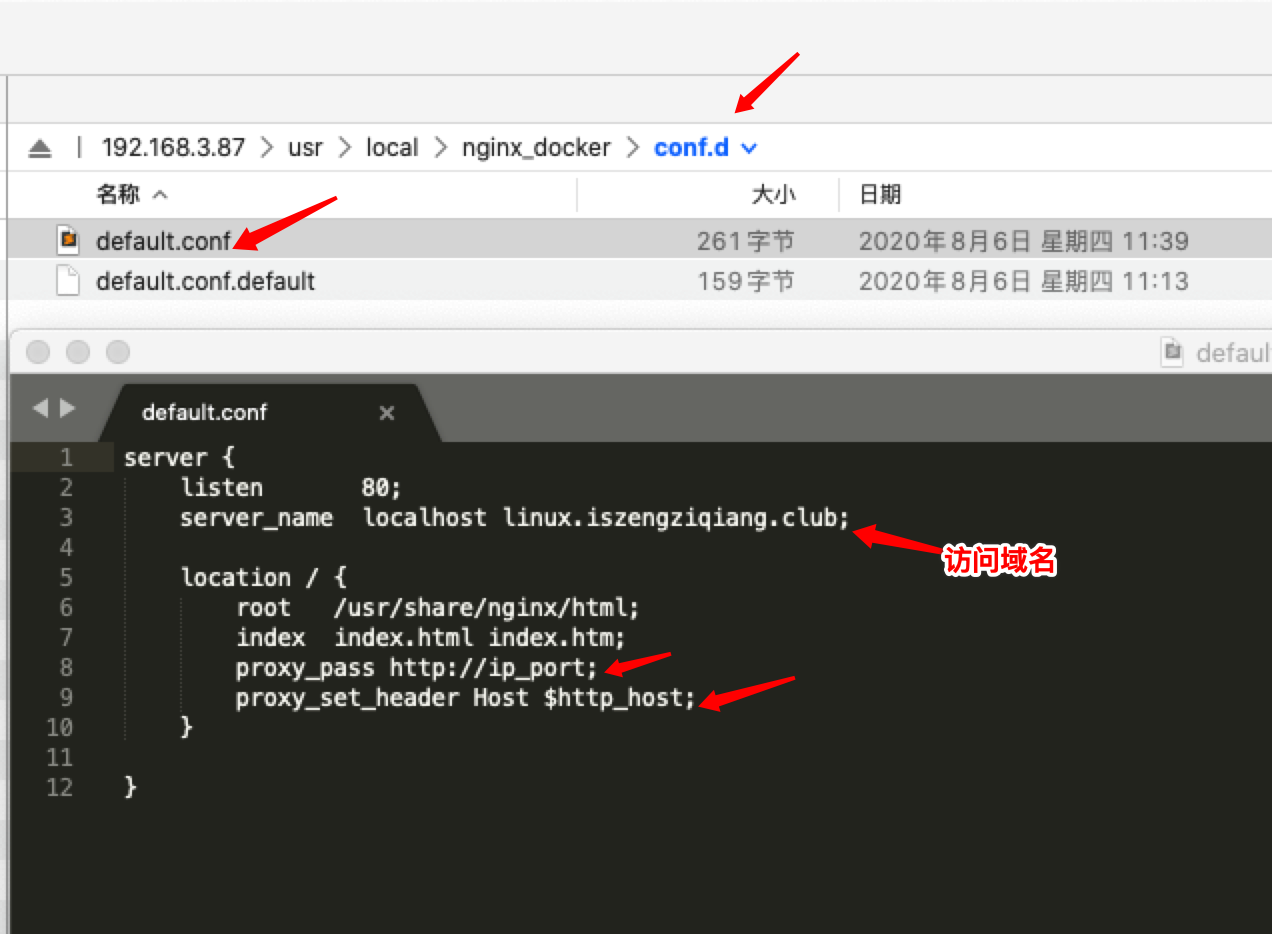
## 如何配置域名

### 1在机器上准备一个能运行的项目：

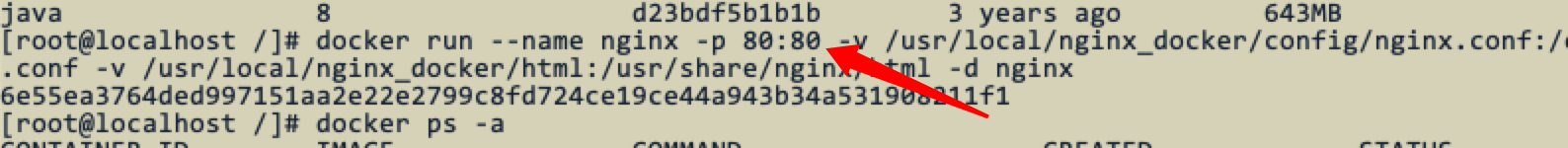


### 2修改Nginx的配置文件





运行命令，从镜像开始启动：



|  |
| --- |
| 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 |

映射端口换成80:80

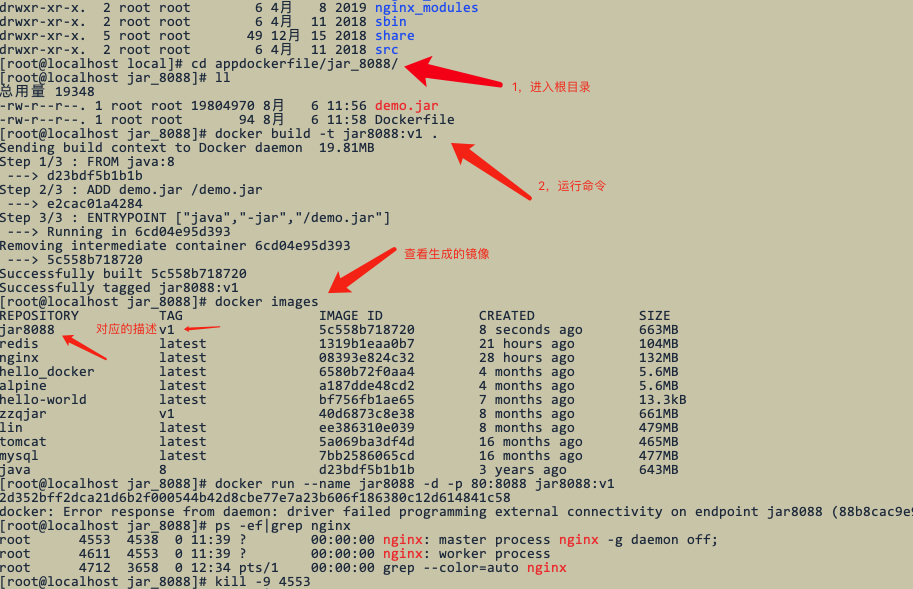
### 3Nginx访问jar镜像

制作一个jar镜像：

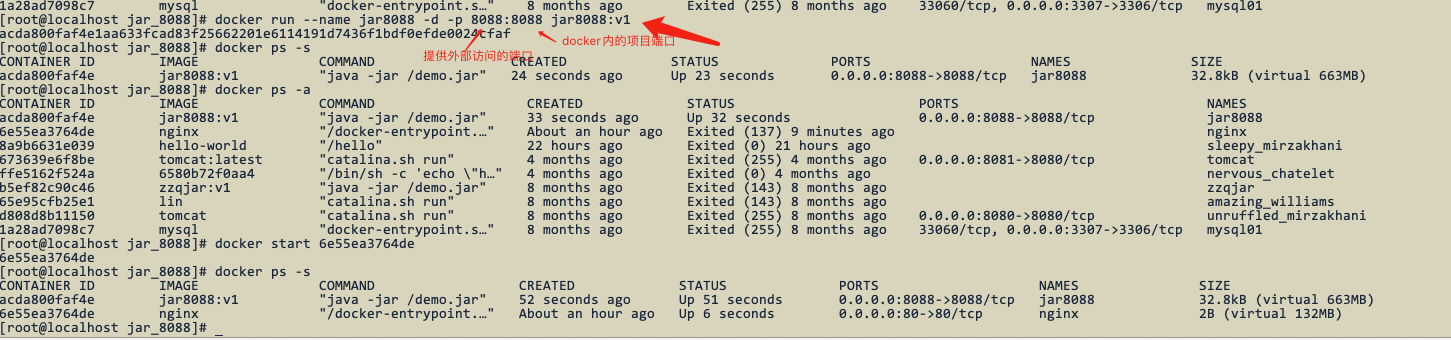
准备jar包(端口为8088)，准备Dockerfile文件



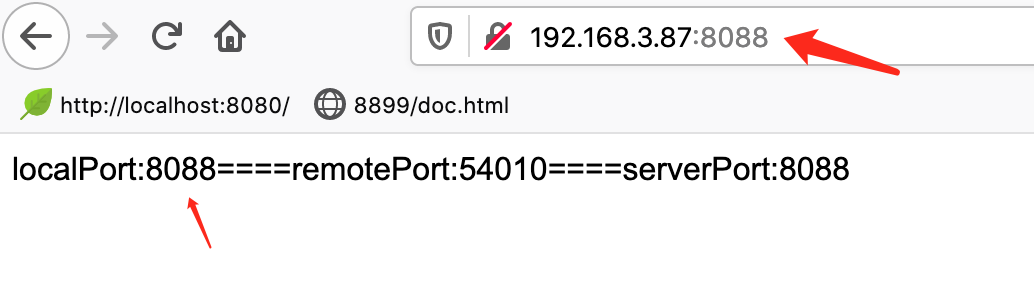
生成镜像命令：【 docker build -t jar8088:v1 .】



启动jar镜像



启动成功后，我们就可以通过主机端口+IP的方式访问了，现在用的是8088端口访问



如果你的启动命令用的是：

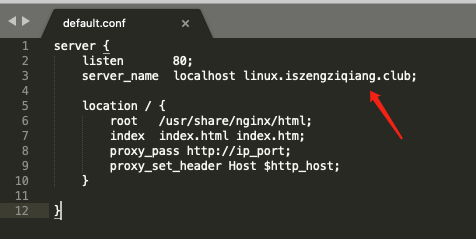
|  |
| --- |
| docker run --name jar8088 -d -p 8099:8088 jar8088:v1 |

那么你访问的方式是http://192.168.3.87:8099

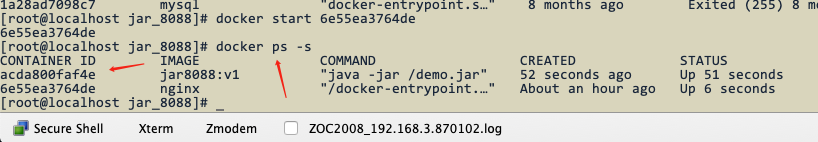
这一点非常重要，不然你后面的Nginx就不知道怎么配置了。

因为我们的Nginx里面已经配置了端口和域名

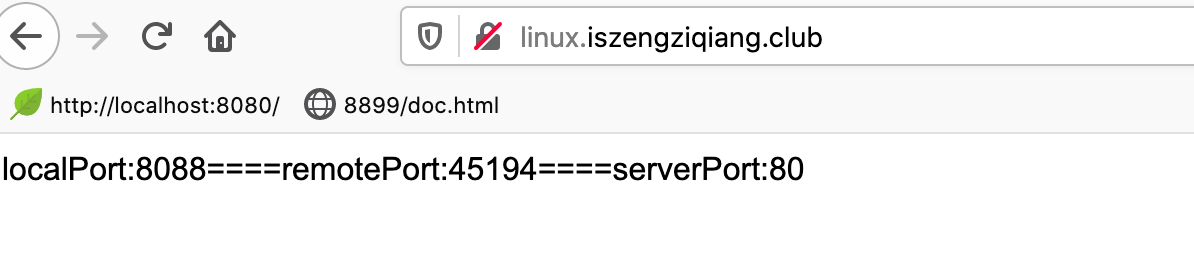




所有启动Nginx后就可以通过域名进行访问了



我们启动了项目和Nginx两个镜像



# Redis外部挂载