**使用docker快速部署**

**若依前后端分离项目**

# 项目规划与前置工作

## 1.1、docker所需镜像

**java:8（jdk1.8）**

**mysql:8.0.19**

**redis:6.0.8**

**nginx:1.18.0**

## 1.2、使用工具

**IDEA**

**Navicat**

**WinSCP**

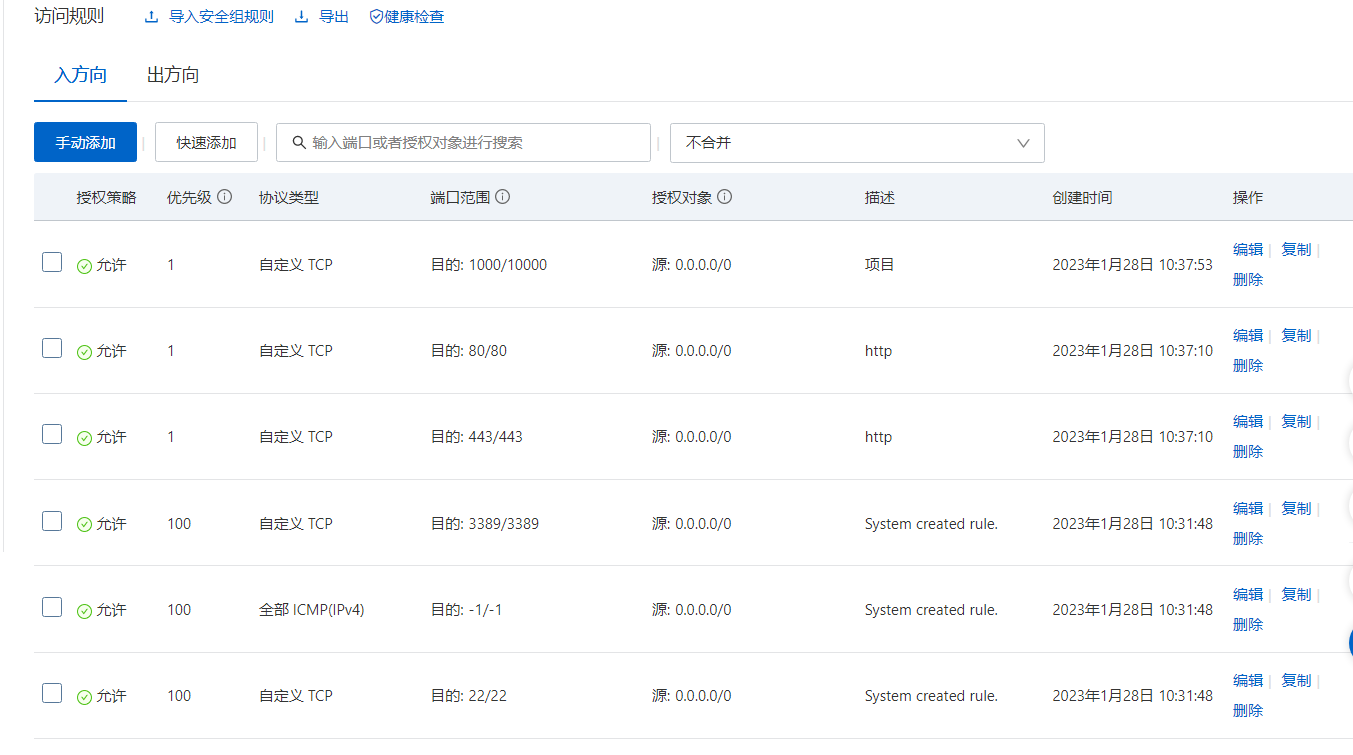
**Putty**

## 1.3、防火墙与安全组的配置

1、一键关闭防火墙（方法1） 或者 手动开放我们所需要的端口（方法2）

为了方便，我们直接关闭防火墙：systemctl disable firewalld

2、安全组配置（截图如下，可根据这个进行配置）



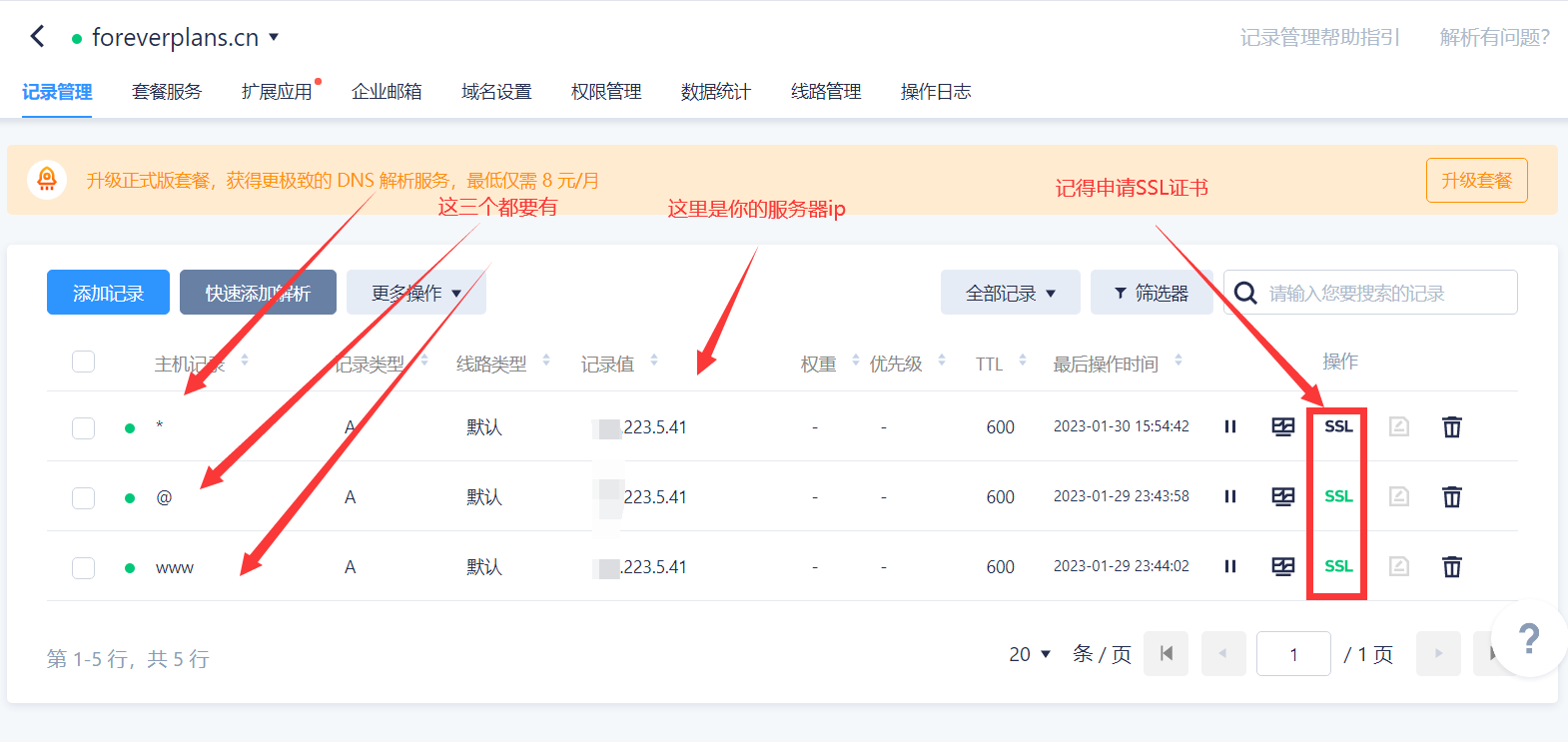
## 1.4、域名解析、备案、SSL证书的准备（可选）

**1、如果没有域名，也可以用ip访问（后面有介绍）**

**2、域名解析与SSL证书申请**

**根域名的SSL证书申请**

**子域名的SSL证书申请**



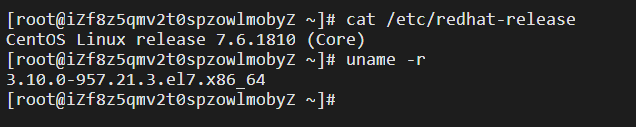
# Docker和镜像安装

## 2.1、Docker安装

### 2.1.1、检查环境

**Docker运行环境要求系统为64位、Linux系统内核版本为 3.8以上**

**查看自己虚拟机的内核：cat /etc/redhat-release**



### 2.1.2、搭建gcc环境（gcc是编程语言译器）

yum -y install gcc

yum -y install gcc-c++

### 2.1.3、安装需要的软件包

yum install -y yum-utils

### 2.1.4、装镜像仓库

**因为docker的服务器是在国外，所以有时候从仓库中下载镜像的时候会连接被拒绝或者连接超时的情况！**

**因此可以使用阿里云镜像仓库**

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

### 2.1.5、新yum软件包索引

yum makecache fast

### 2.1.6、安装docker引擎

yum install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

### 2.1.7、启动docker

systemctl start docker

### 2.1.8、查看docker版本

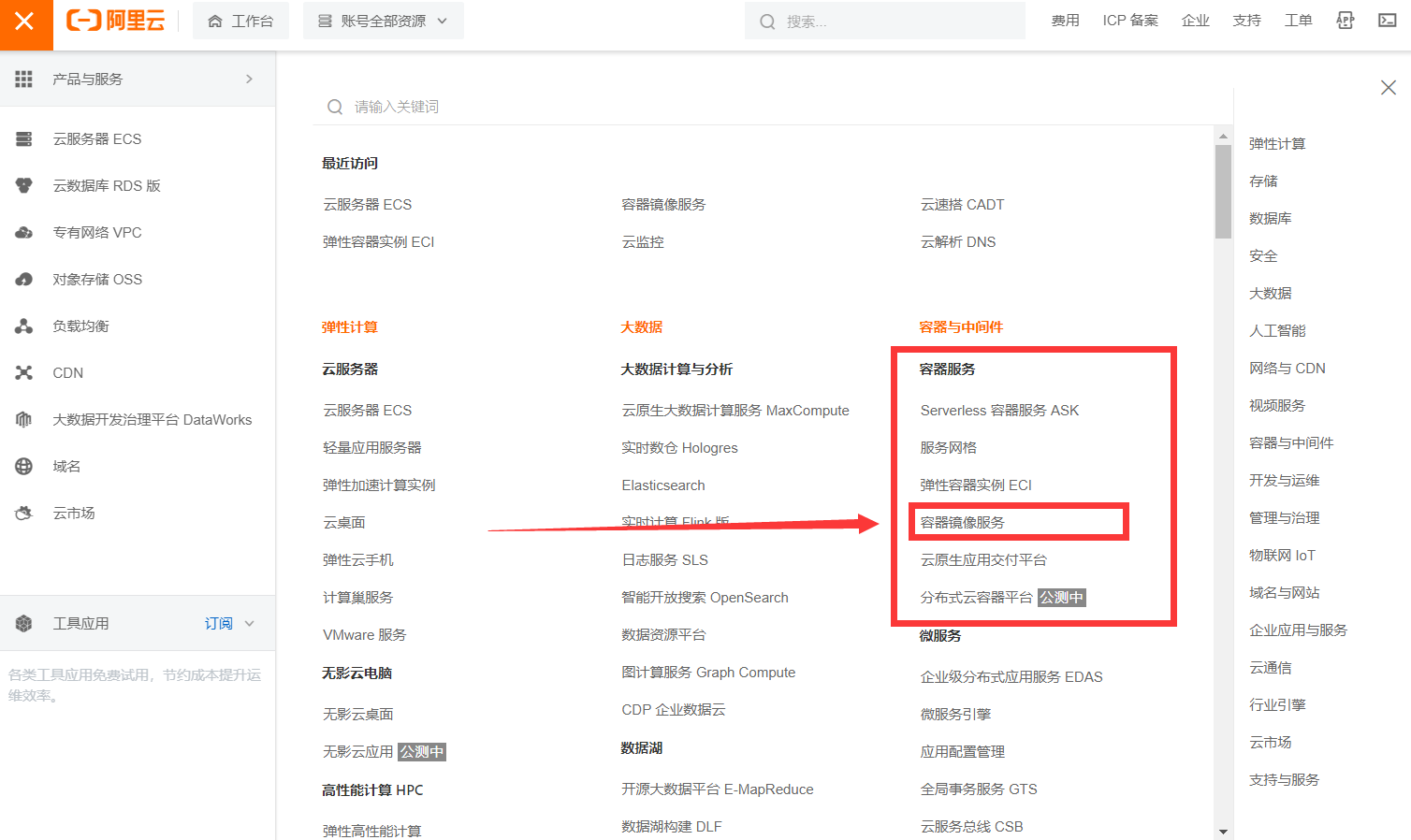
docker version



### 2.1.9、配置阿里云镜像加速

为了提高镜像的拉取、发布的速度，可以配置阿里云镜像加速（推荐）

1、找到容器镜像服务ARC



2、复制加速器地址



3、在CentOS下配置镜像加速器

mkdir -p /etc/docker

tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["你个人的阿里云镜像加速器地址"]

}

EOF

systemctl daemon-reload

systemctl restart docker

## 2.2、拉取镜像

### 2.2.1、拉取镜像java:8(jdk1.8)

docker pull java:8

### 2.2.2、拉取镜像mysql:8.0.19

docker pull mysql:8.0.19

### 2.2.3、拉取镜像redis:6.0.8

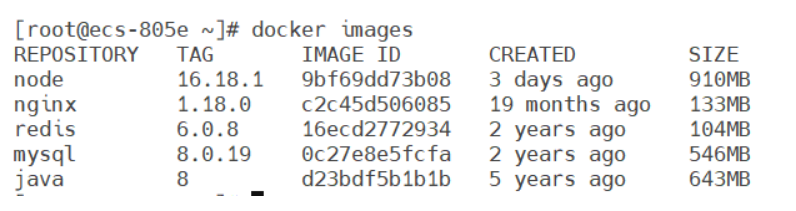
docker pull redis:6.0.8

### 2.2.4、拉取镜像nginx:1.18.0

docker pull nginx:1.18.0

### 2.2.5、查看镜像

docker images



# 部署后端

**命令参数说明：**

**-d：后台运行容器**

**-p：端口映射，格式为主机端口:容器端口**

**-e：设置环境变量，这里设置的是root密码**

**--name：设置容器别名**

**-v 挂载文件，格式为：宿主机绝对路径目录:容器内目录，**

**比如我们使用：-v /usr/local/mysql/logs:/var/log/mysql**

**来将mysql容器存放日志文件的目录：/var/log/mysql挂载在宿主机的/usr/local/mysql/logs下**

## 3.1、启动数据库容器

docker run --name mysql -v /myapp/mysql:/var/lib/mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 -d mysql:8.0.19

【注意：上面的123456是密码，可以自行修改】

## 3.2、启动Redis容器

docker run -d --name redis -p 6379:6379 redis:6.0.8 --requirepass "123456"

【注意：上面的123456是密码，可以自行修改】

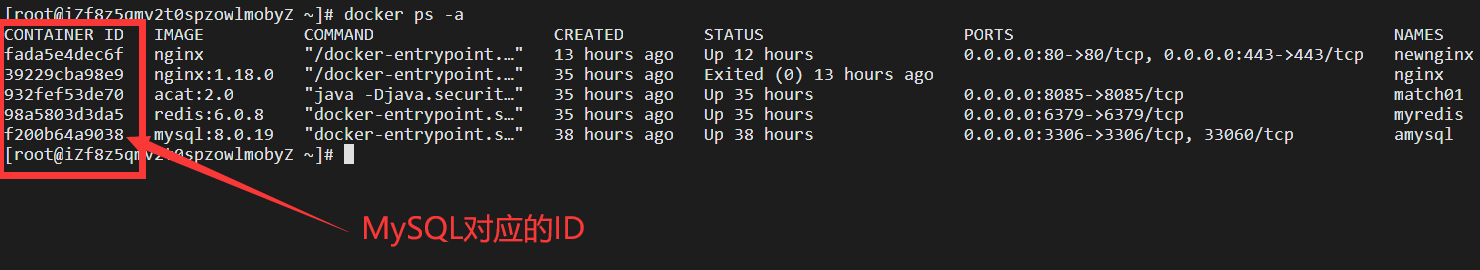
## 3.3、导入SQL数据（方法1）【两个方法自行选择】

### 3.3.1、使用winSCP将SQL文件导入服务器的/myapp/mysql目录中



### 3.3.2、进入MySQL容器实例，导入数据

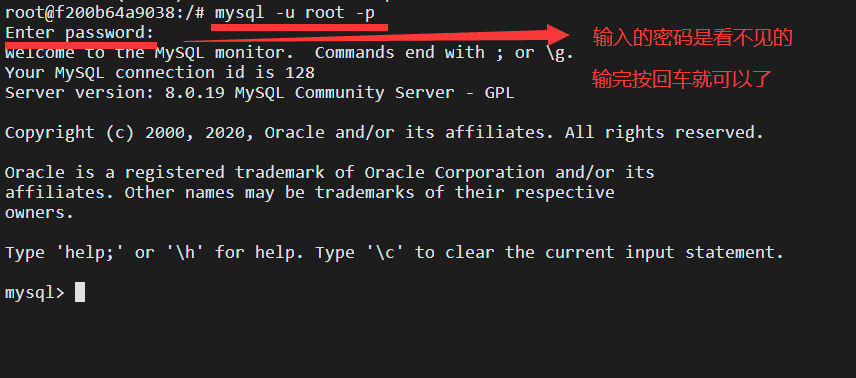
1、查看容器ID：docker ps -a



2、进入容器： docker exec -it [容器ID] /bin/bash

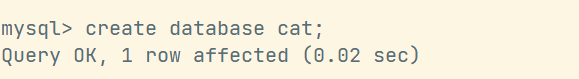


3、登录MySQL：mysql -u root -p



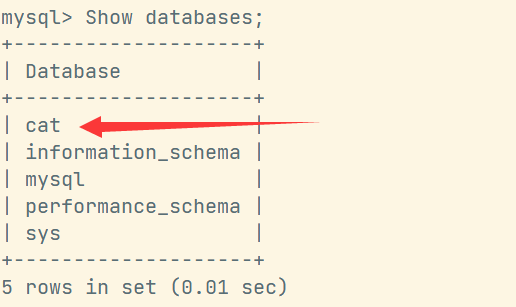
4、创建数据库

create database cat;

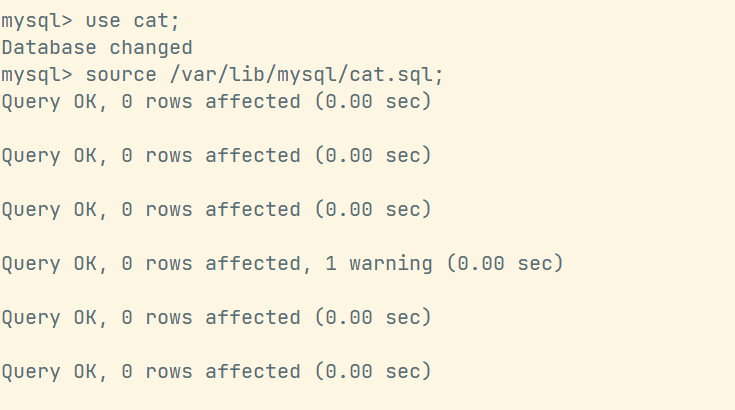


5、查看数据库

Show databases;

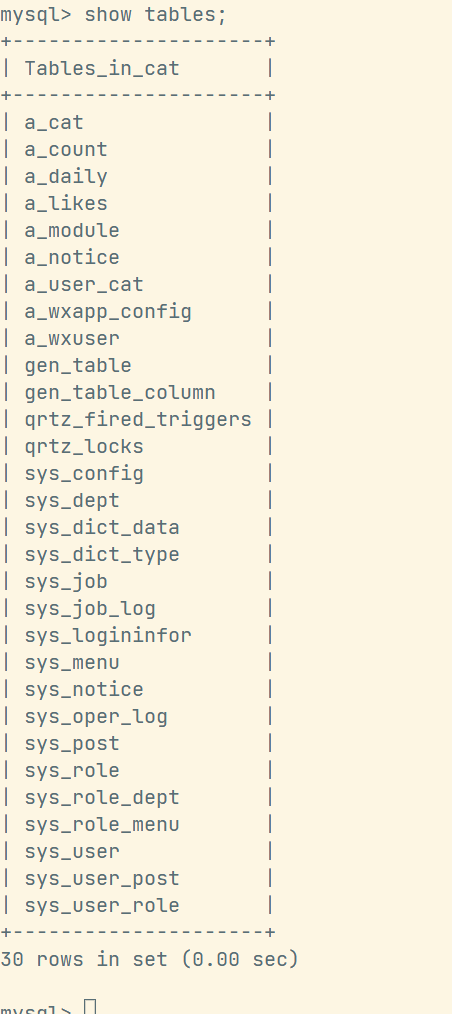


6、进入指定数据库，并导入数据库文件



7、查看是否导入完成

show tables;

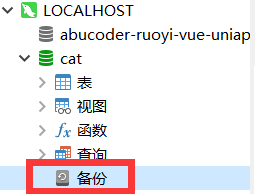


## 3.4、导入SQL数据（方法2）【如果已经导入可忽略】

### 3.4.1、使用Navicat远程连接MySQL

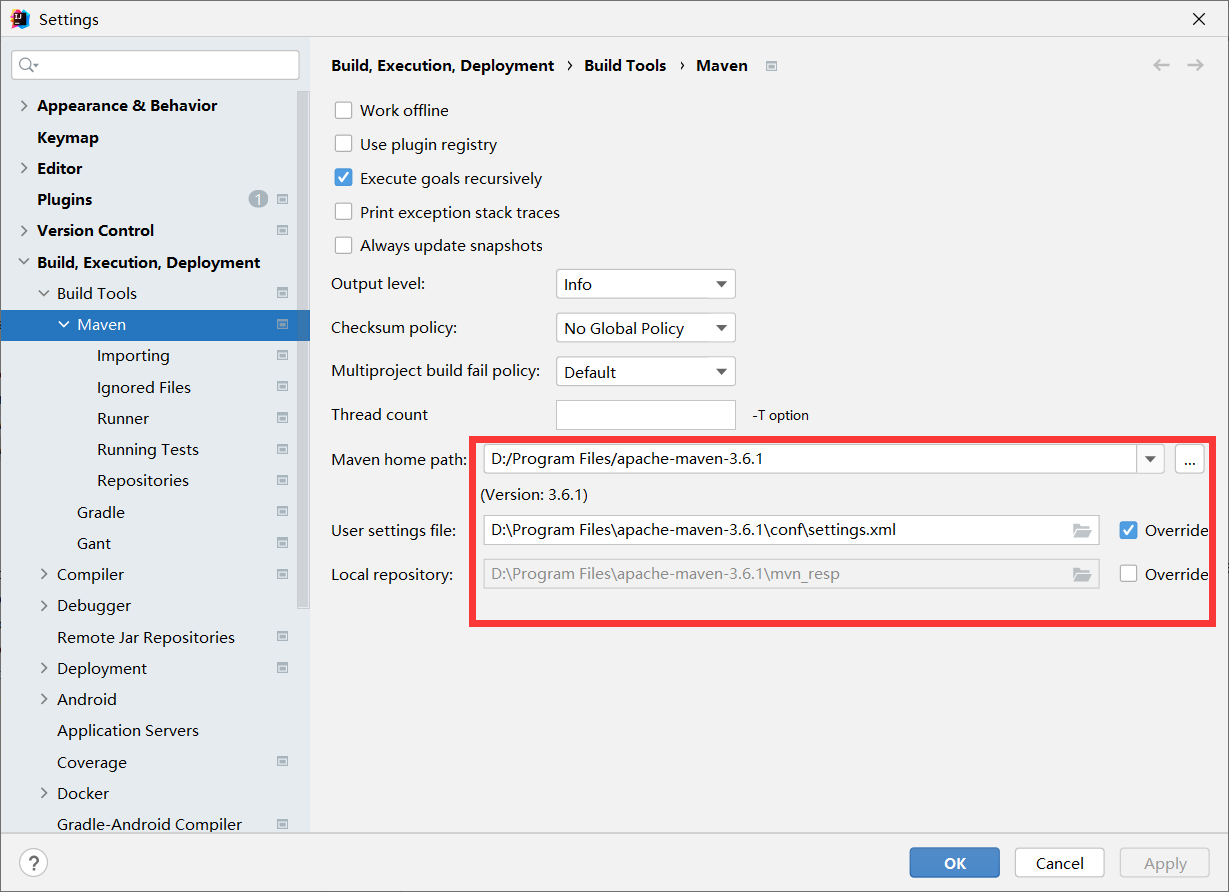


### 3.4.2、使用Navicat备份本地数据库后，导入服务器的数据库

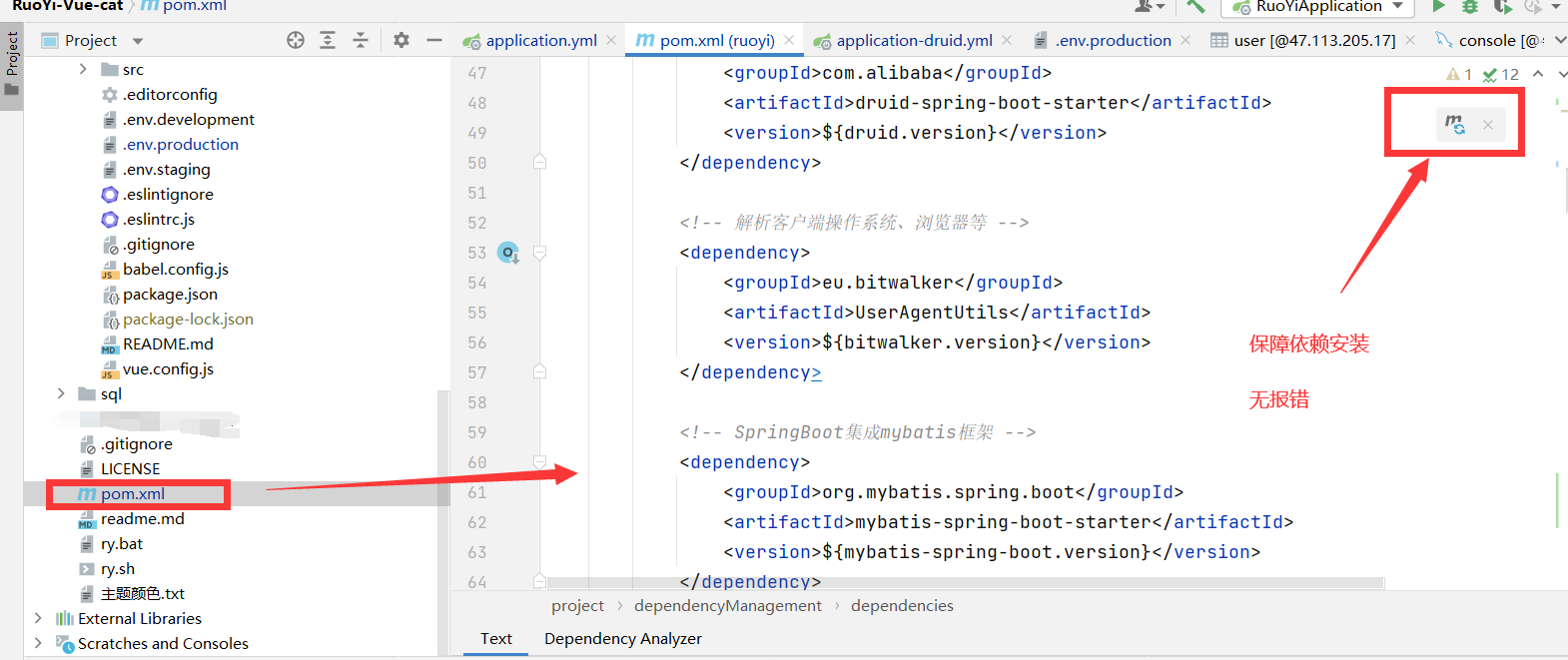


## 3.5、后端项目配置修改与打包

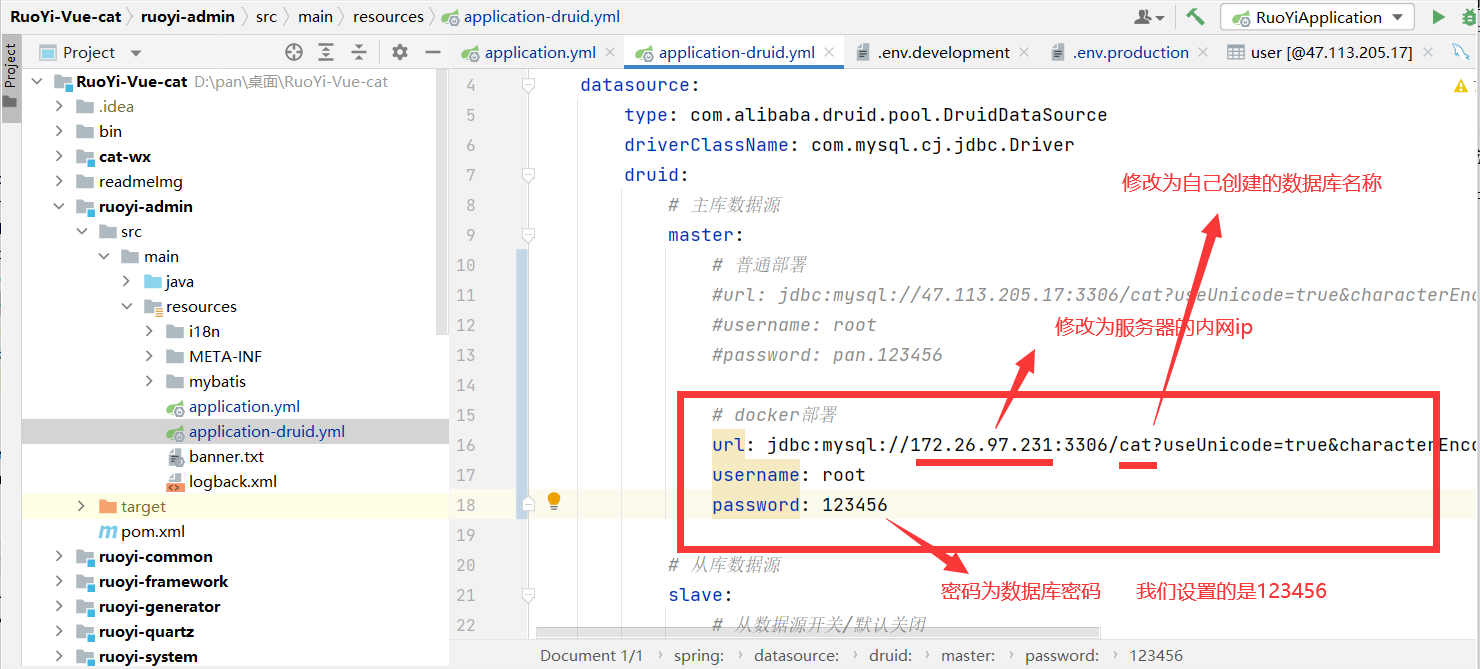
### 3.5.1、配置maven



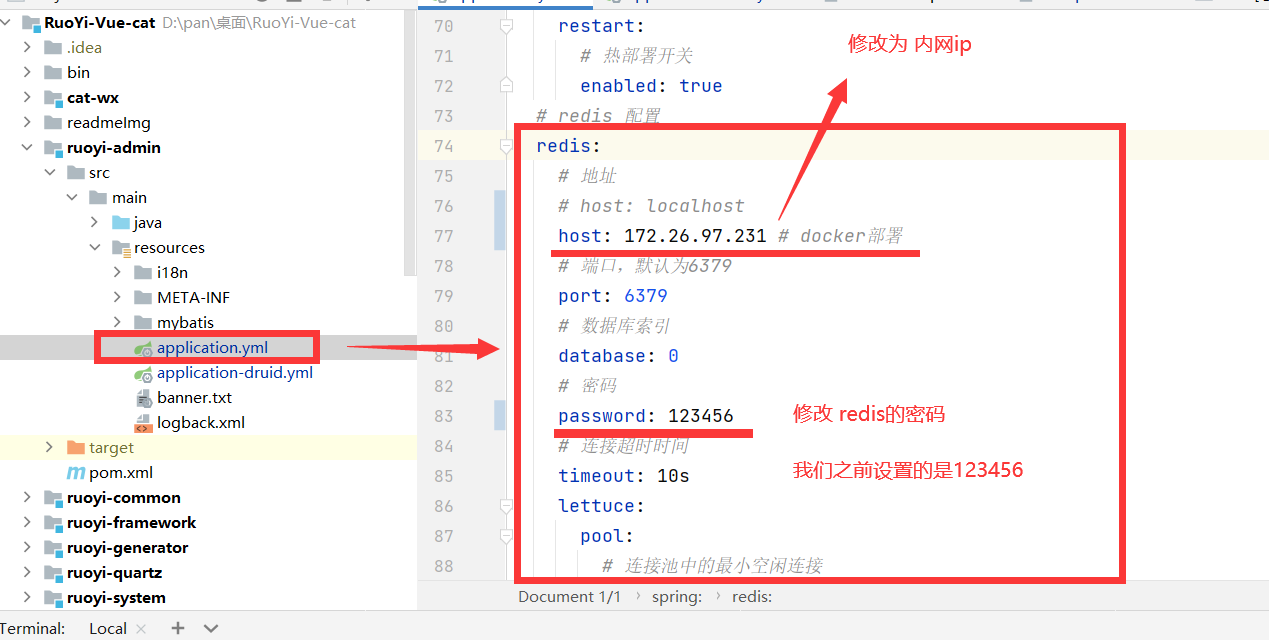
### 3.5.2、刷新依赖（保证无报错）



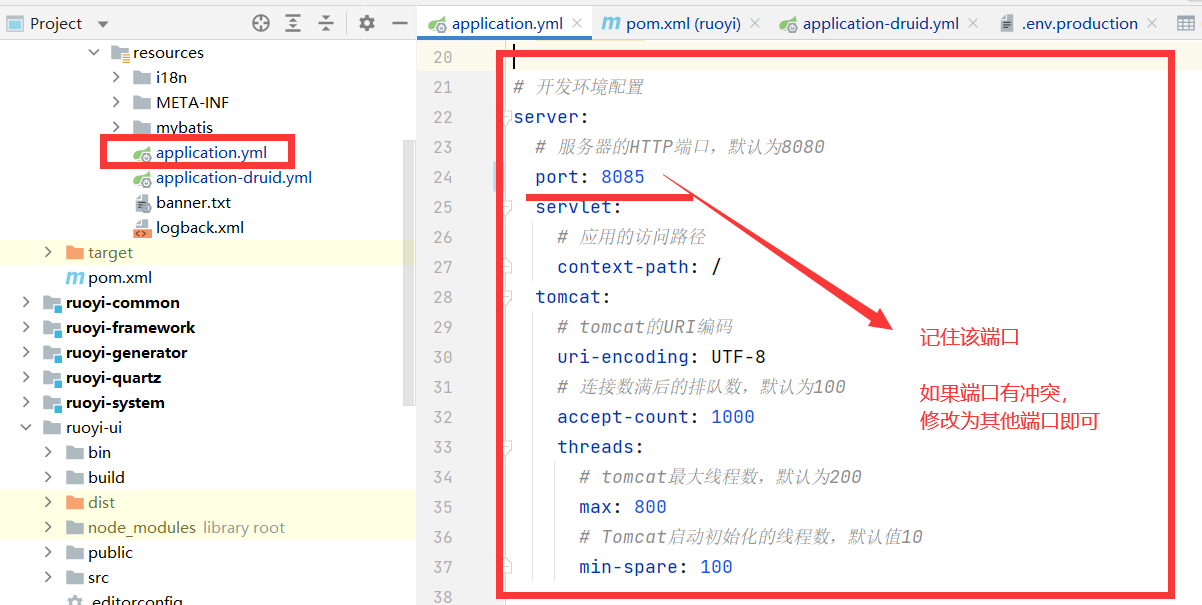
### 3.5.3、修改数据库配置



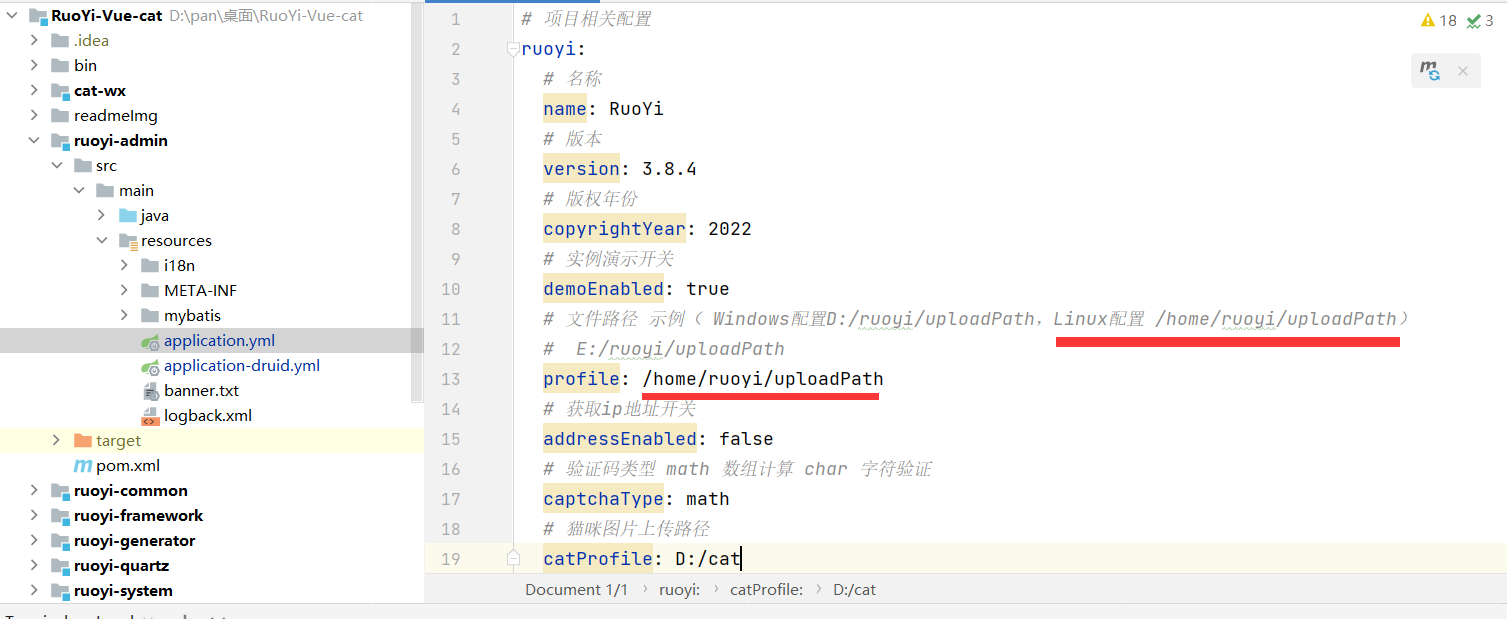
### 3.5.4、修改Redis配置



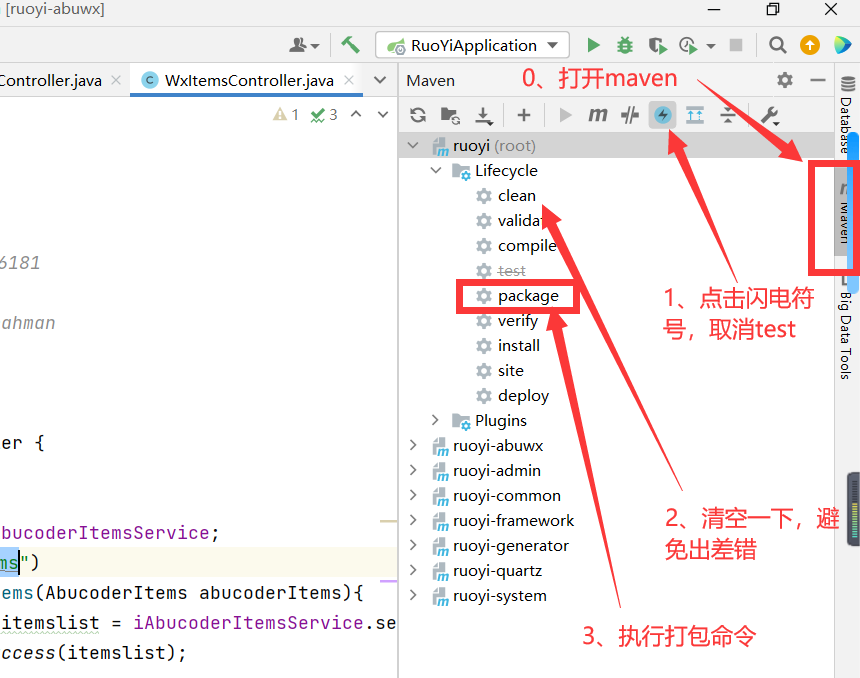
### 3.5.5、查看开放端口



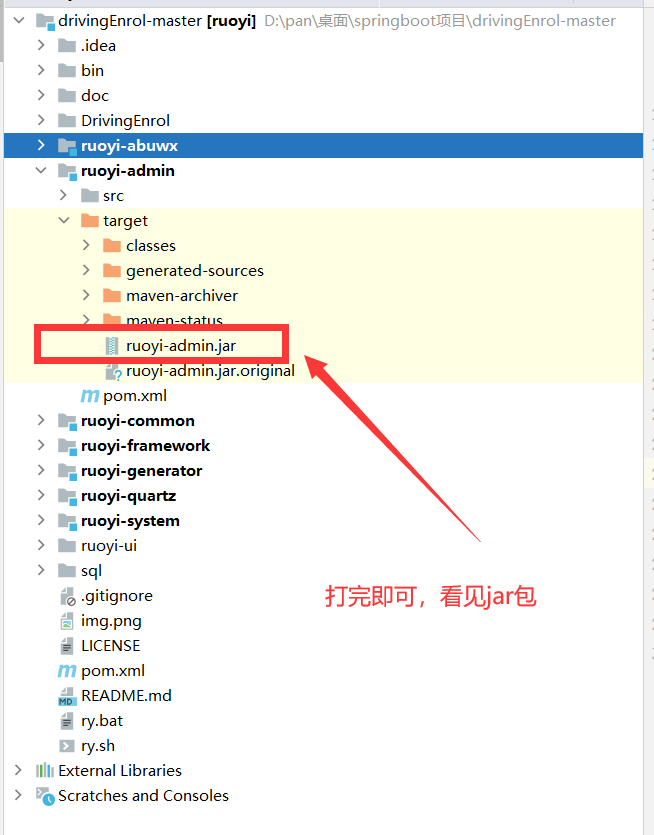
### 3.5.6、查看文件路径（修改为Linux配置）



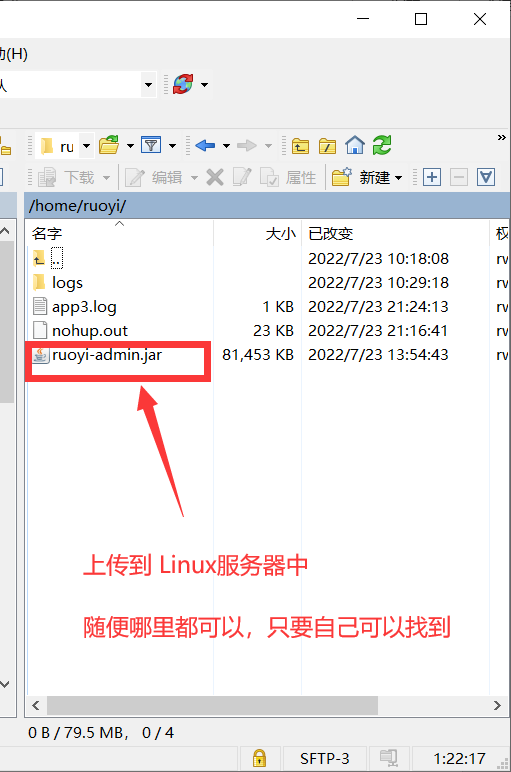
### 3.5.7、后端项目打包



### 3.5.8、查看是否打包成功

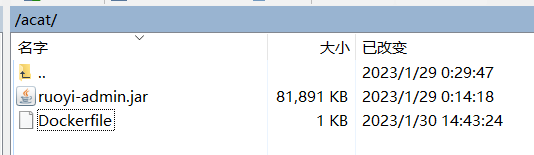


## 3.6、后端项目上传



## 3.7、后端项目镜像构建

### 3.7.1、进入jar包同级目录，创建一个Dockerfile文件



### 3.7.2、打开文件，将下面的数据复制到Dockerfile文件中

#基础镜像使用jdk1.8

FROM java:8

#作者

MAINTAINER pan

# VOLUME 指定临时文件目录为/tmp，在主机/var/lib/docker目录下创建了一个临时文件并链接到容器的/tmp

VOLUME /tmp

# 将jar包添加到容器中并更名

ADD ruoyi-admin.jar app.jar

# 运行jar包

RUN bash -c 'touch /app.jar'

# 为了缩短 Tomcat 启动时间，添加一个系统属性指向 “/dev/./urandom” 作为 Entropy Source

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

#ENTRYPOINT ["nohup","java","-jar","/data/cat/ruoyi-admin.jar","&"]

#暴露8085端口

EXPOSE 8085



### 3.7.3、构建镜像（需要在Dockerfile同级目录下构建）

docker build -t cat:1.0 .

### 3.7.4、查看当前的镜像

docker images

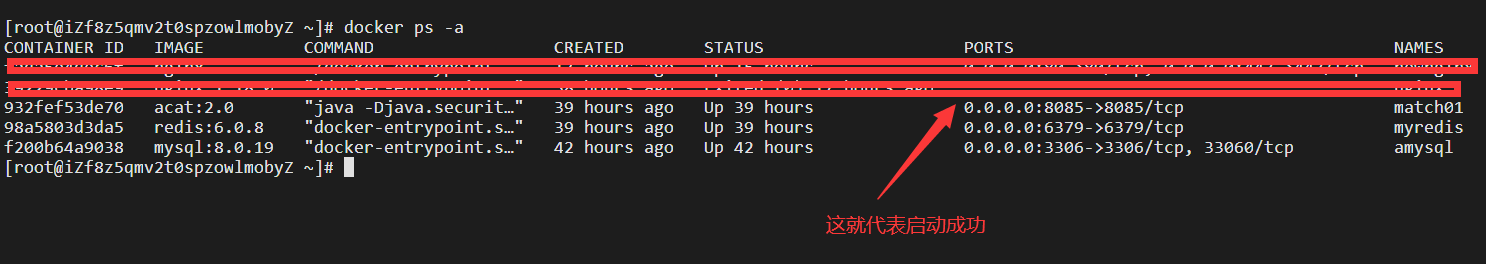
### 3.7.5、启动后端项目镜像

docker run -d --name cat -p 8085:8085 cat:1.0

### 3.7.6、查看容器是否启动成功

docker ps -a

如果成功则代表后端部署完成！



可以访问进一步验证



**后端部署完成！**

### 3.7.7、如果容器没有启动成功，请看这一步（成功的则跳过，去看前端部署）

1、查看容器日志

docker logs -f -t --tail 1000 [容器ID]

2、通过日志确定问题，将报错进行复制，进行百度搜索，自行解决

# 部署前端

## 4.1、Nginx镜像部署

**因为需要nginx的初始化配置文件，为了保证不出错，**

**所以我们直接启动一个nginx容器，把配置文件拉取下来，然后删除容器！**

### 4.1.1、创建nginx工作目录

#需要一个conf文件存放目录，和html文件目录,及日志存放目录

mkdir -p /data/applications/nginx/conf

mkdir -p /data/applications/nginx/html

mkdir -p /data/logs

### 4.1.2、启动nginx容器，如果这个镜像本机中没有，会自动下载镜，我这下载是的1.18版本

docker run --name nginx -p 80:80 -d nginx:1.18.0

### 4.1.3、复制容器中配置文件目录到本机中，然后删除该容器

docker cp nginx:/etc/nginx/nginx.conf /data/applications/nginx/conf/nginx.conf

docker cp nginx:/etc/nginx/conf.d /data/applications/nginx/conf/conf.d

docker cp nginx:/usr/share/nginx/html /data/applications/nginx/

#先用 [docker ps – a] 命令 找到对应容器ID

#才可以删除容器

docker rm -f [容器ID]

### 4.1.4、启动新的nginx容器

docker run -d \

-p 80:80 \

-p 443:443 \

--name nginx \

--restart=always \

--network-alias nginx \

-e TZ="Asia/Shanghai" \

-v /data/applications/nginx/conf/nginx.conf:/etc/nginx/nginx.conf \

-v /data/applications/nginx/conf/conf.d:/etc/nginx/conf.d \

-v /data/logs/nginx:/var/log/nginx \

-v /data/applications/nginx/html:/usr/share/nginx/html \

-v /data/applications/nginx/cert:/etc/nginx/cert \

nginx:1.18.0

**再一次提醒firewalld防火墙是否关闭，或者端口是否开启，以及服务器的安全组是否配置！**

**命令说明：**

**–privileged=true # 容器内被赋予root权限**

**-v /data/applications/nginx/html:/usr/share/nginx/html # 挂载的静态文件目录**

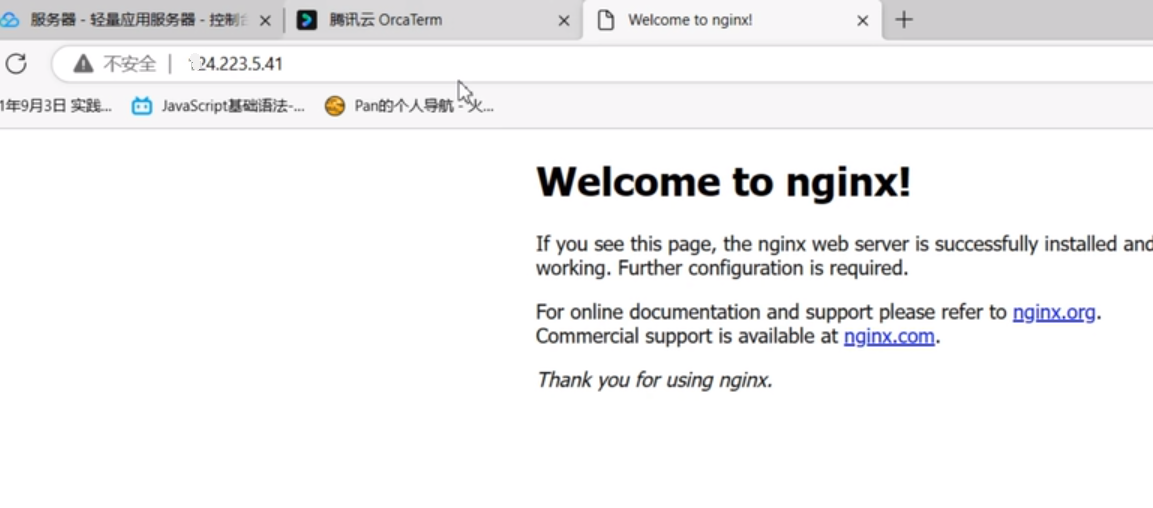
**-v /data/applications/nginx/nginx.conf:/etc/nginx/nginx.conf # 挂载的nginx配置文件目录**

**-v /data/applications/nginx/logs:/var/log/nginx #挂载nginx日志文件**

**-v /data/applications/nginx/cert:** **/etc/nginx/cert # ssl证书挂载目录**

### 4.1.5、查看容器 或 访问服务器ip地址（检验是否启动成功）

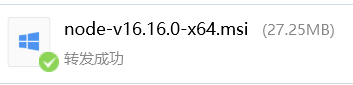
docker ps -a



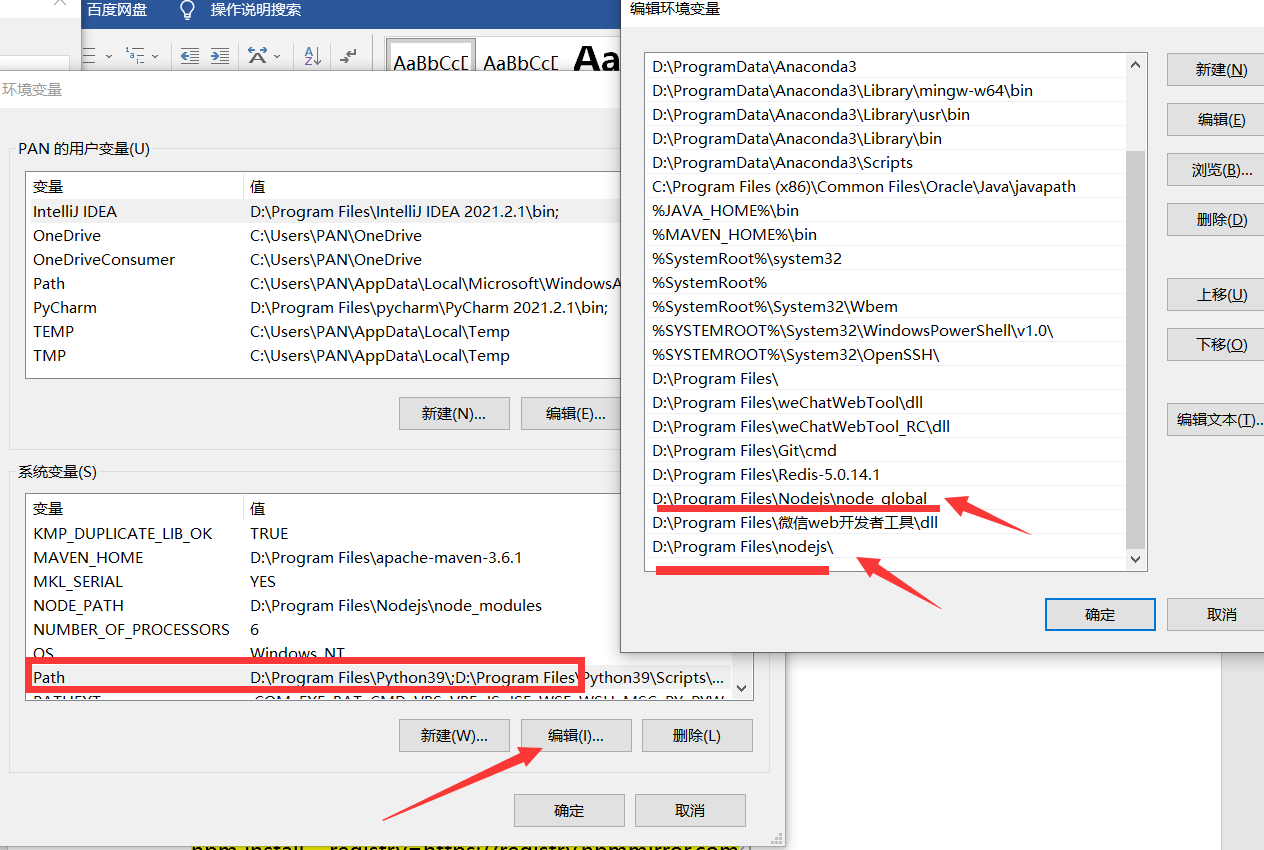
**看见Nginx首页，代表我们部署成功**

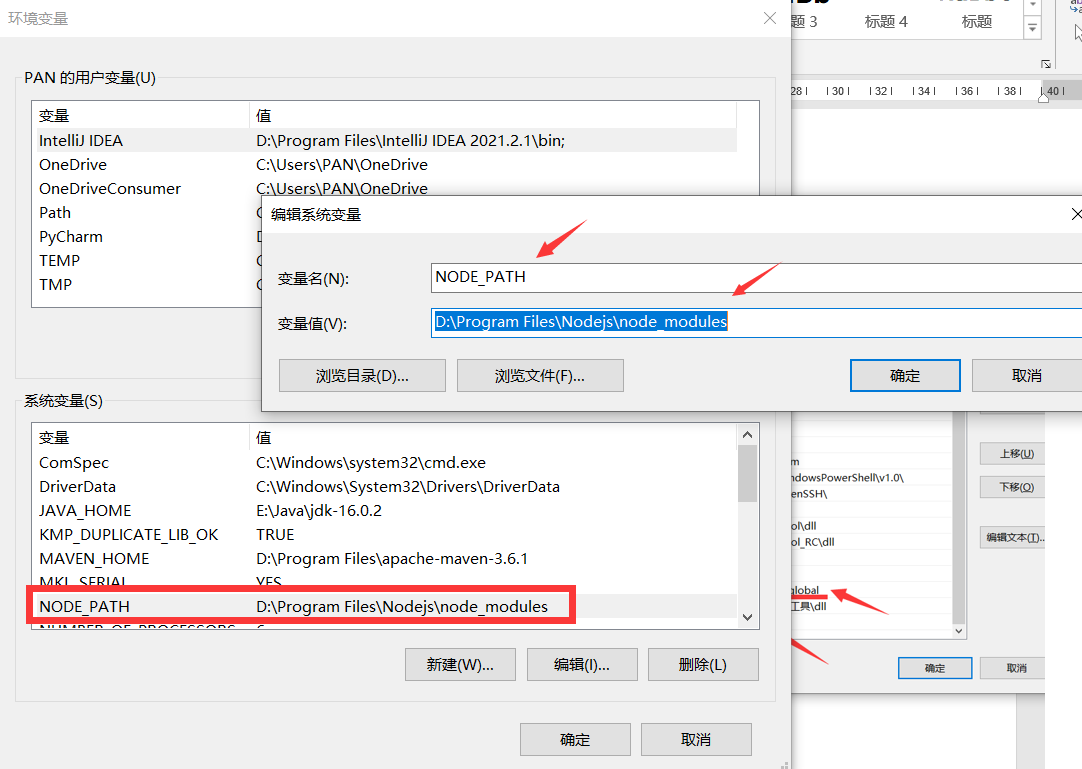
## 4.2、本地node安装

### 4.2.1 安装node



### 4.2.2 配置环境变量（使终端可以使用npm命令）





## 4.3、方案分流（有域名则继续往下）

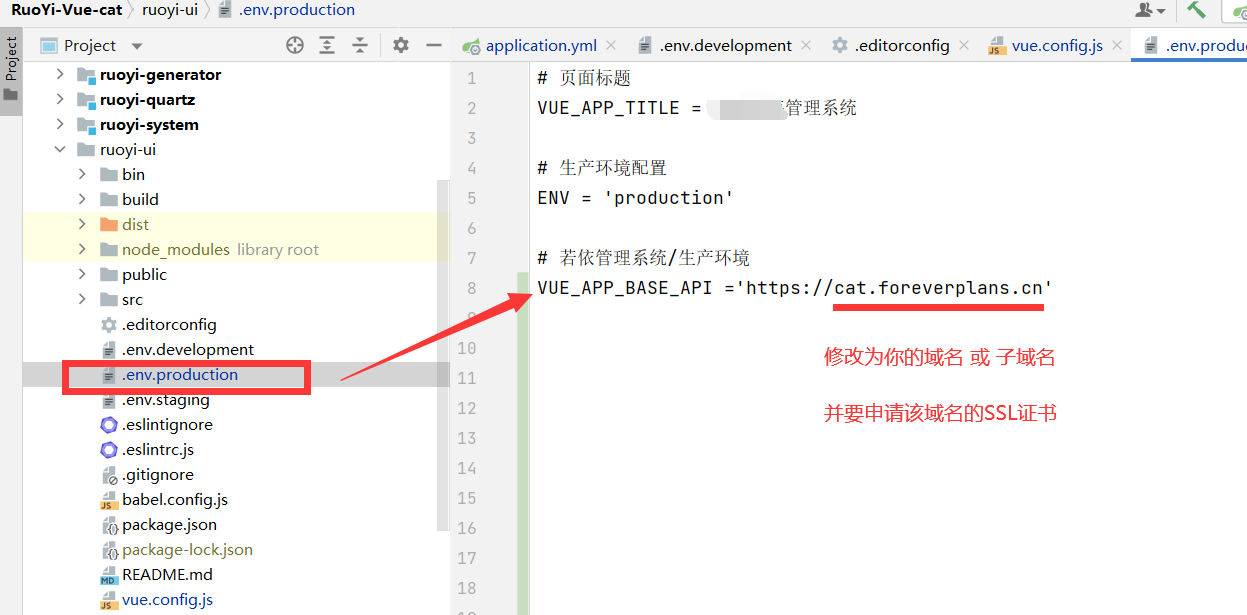
（如果这一步成功，并且你没有域名，则请跳到6.1观看无域名版本）

## 4.4、前端项目配置、打包、上传

### 4.4.1、前端项目配置 代理域名

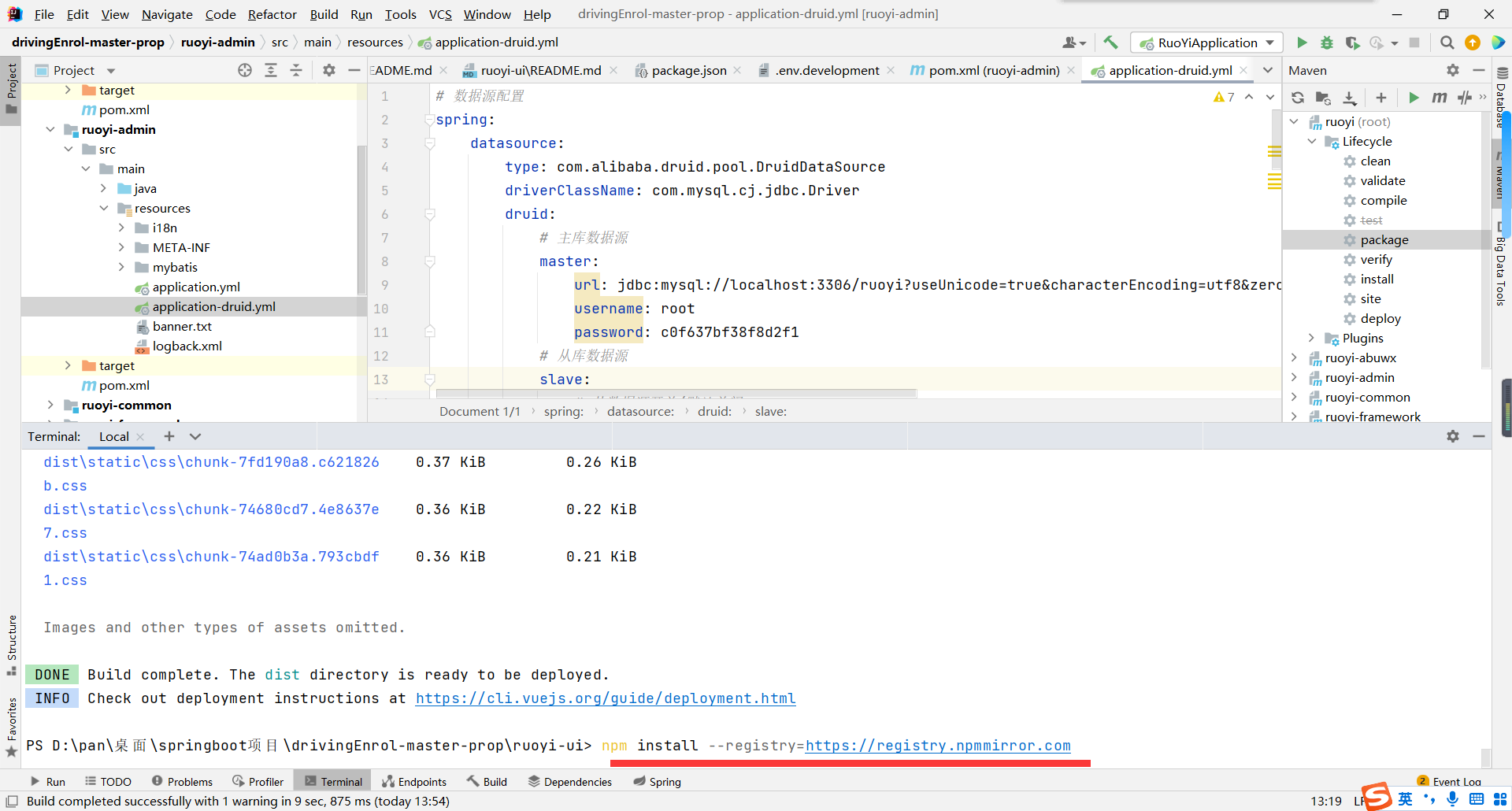
记住代理的域名，我们一会将下面的域名设置反向代理，代理端口为：【公网ip:端口号】

也就是我们的后端端口



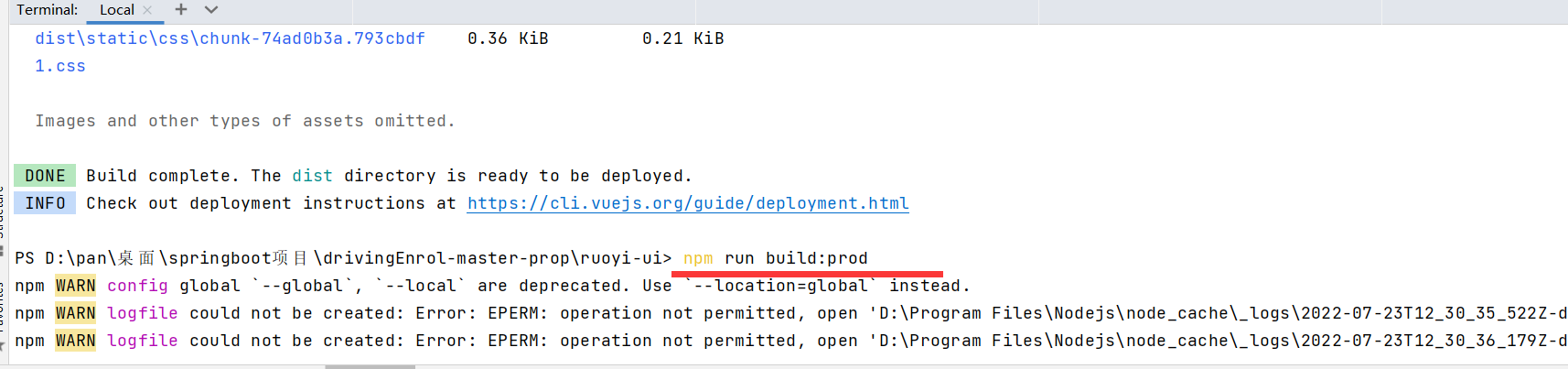
### 4.4.2、在前端项目中打开终端【执行下面命令】

npm install --registry=https://registry.npmmirror.com

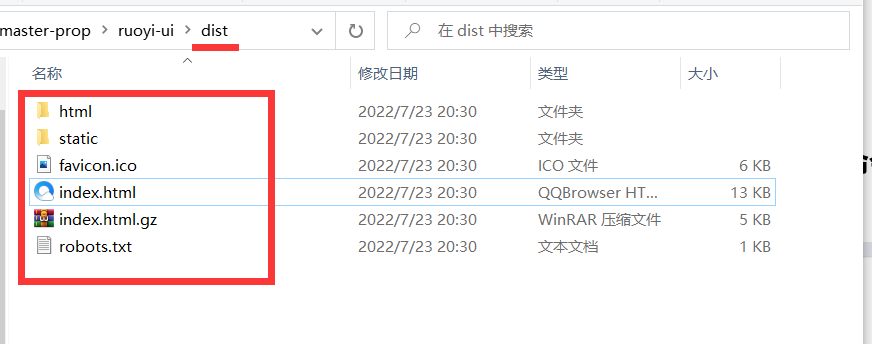


### 4.4.3、继续在终端执行打包命令

npm run build:prod

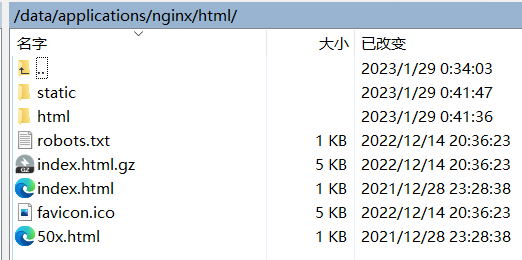


### 4.4.4、查看dist目录【如下情况，即是 打包成功】



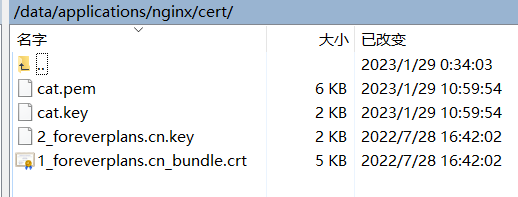
### 4.4.5、将 前端项目 拷贝到Linux服务器中nginx容器对应的html目录下





## 4.5、配置域名的SSL证书

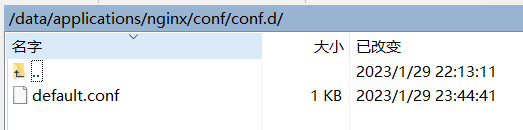




## 4.6、nginx配置

### 4.6.1、进入nginx配置目录下，删除default.conf文件中的所有内容





### 4.6.2、将下面的配置复制到default.conf文件中

# 第一个server放置前端项目-------------------------------------------

server {

listen 80; #侦听80端口

listen 443 ssl; #侦听443端口，用于SSL

server\_name foreverplans.cn www.foreverplans.cn; # 修改为你自己的域名 域名添加www的前缀

return 301 https://www.foreverplans.cn$request\_uri; # 修改为你自己的域名

# 注意证书文件位置，是从/etc/nginx/下开始算起的

ssl\_certificate cert/1\_foreverplans.cn\_bundle.crt; #修改证书公钥文件路径

ssl\_certificate\_key cert/2\_foreverplans.cn.key; #修改证书私钥文件路径

ssl\_session\_timeout 5m; #5分钟session会话保持

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE; #加密套件

ssl\_prefer\_server\_ciphers on;

root /usr/share/nginx/html; #指向前端项目

index index index.html index.php index.htm;

location / {

try\_files $uri $uri/ /index.html;

#try\_files $uri $uri/ =404;

}

client\_max\_body\_size 1024m;

}

# 第二个server代理8085端口------------------------------

server {

listen 80; #侦听80端口

listen 443 ssl; #侦听443端口，用于SSL

server\_name cat.foreverplans.cn www.cat.foreverplans.cn; # 修改为你自己的域名 域名添加www的前缀

# 注意证书文件位置，是从/etc/nginx/下开始算起的

ssl\_certificate cert/cat.pem; #修改证书公钥文件路径

ssl\_certificate\_key cert/cat.key; #修改证书私钥文件路径

ssl\_session\_timeout 5m;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers ECDHE-RSA-AES128-GCM-SHA256:HIGH:!aNULL:!MD5:!RC4:!DHE; #加密套件

ssl\_prefer\_server\_ciphers on;

client\_max\_body\_size 1024m;

location / {

proxy\_set\_header HOST $host;

proxy\_set\_header X-Forwarded-Proto $scheme;

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

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

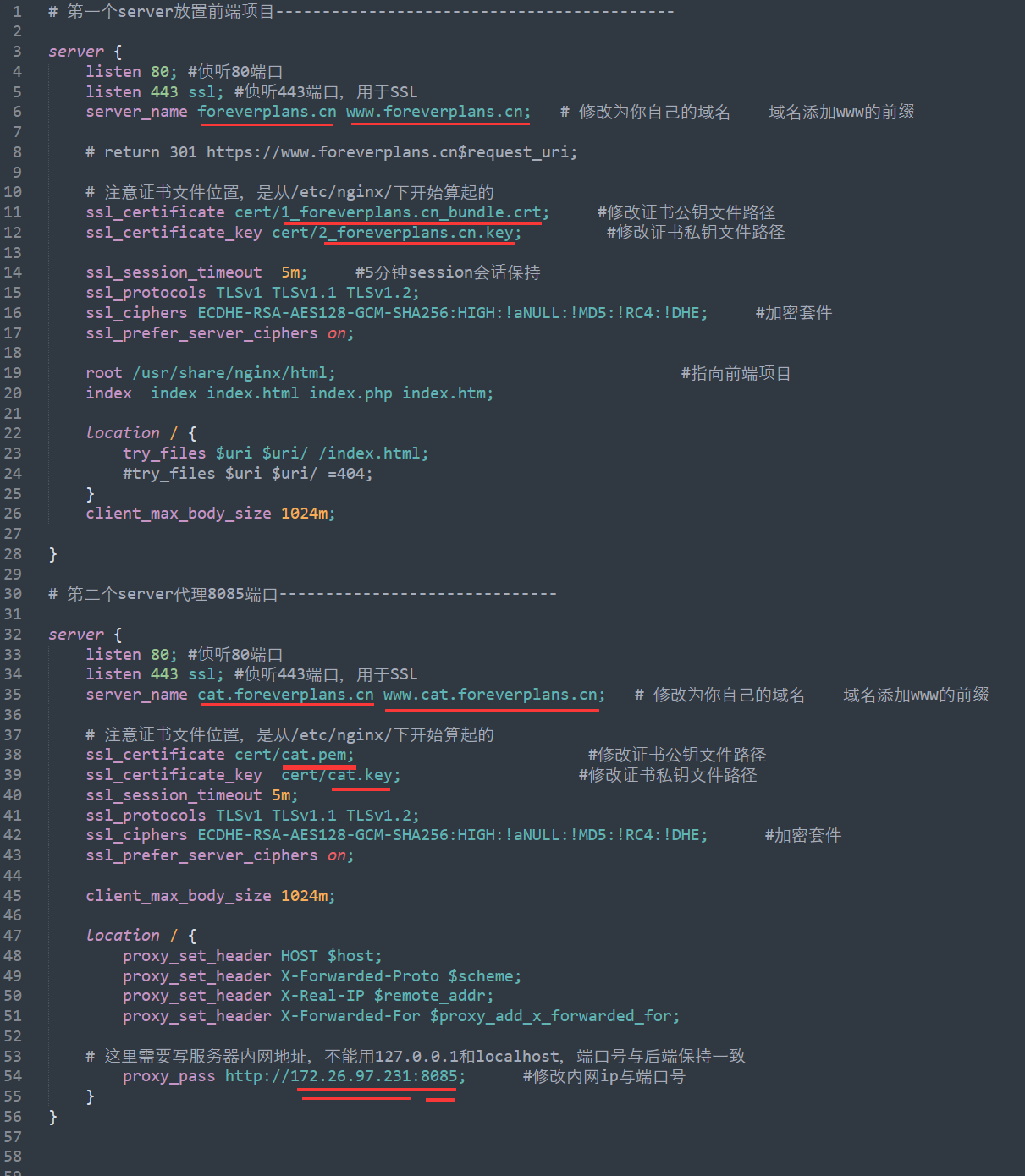
# 这里需要写服务器内网地址，不能用127.0.0.1和localhost，端口号与后端保持一致

proxy\_pass http://172.26.97.231:8085; #修改内网ip与端口号

}

}

### 4.6.2、修改配置（域名，SSL证书名称，代理地址）



### 4.6.3、修改后保存并退出，重新启动nginx容器

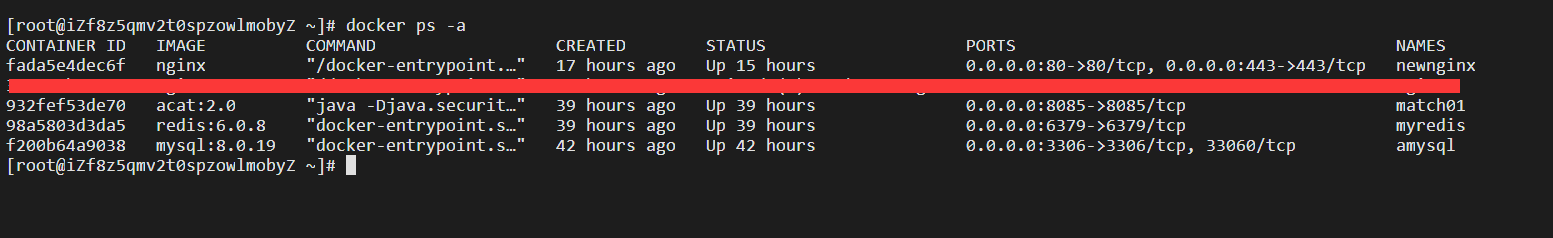
#先用 [docker ps – a] 命令 找到对应容器ID

#才可以删除容器

docker restart [容器ID]

### 4.6.4、查看容器（是否启动成功）

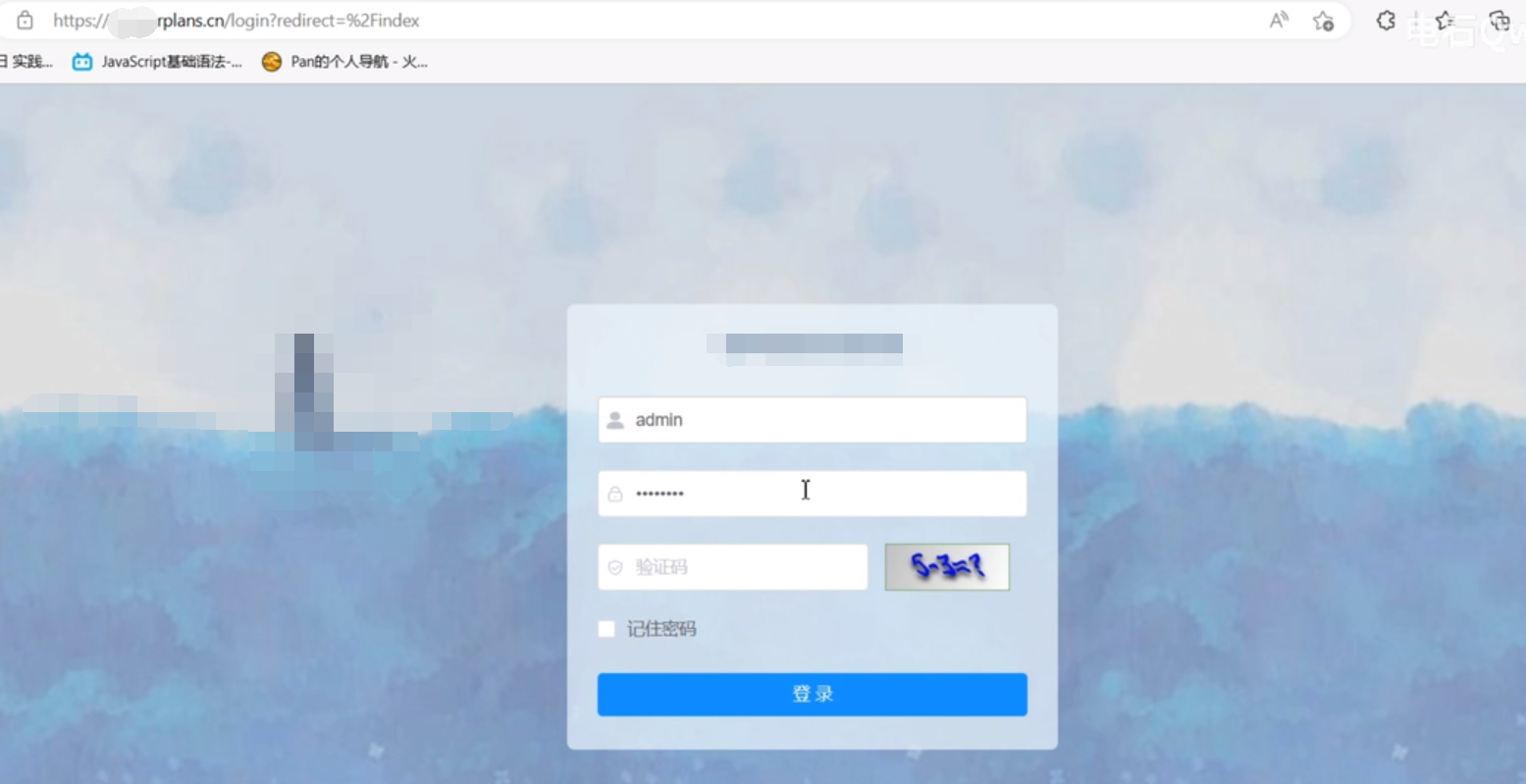
docker ps -a



**好啦！我们已经全部部署完毕！**

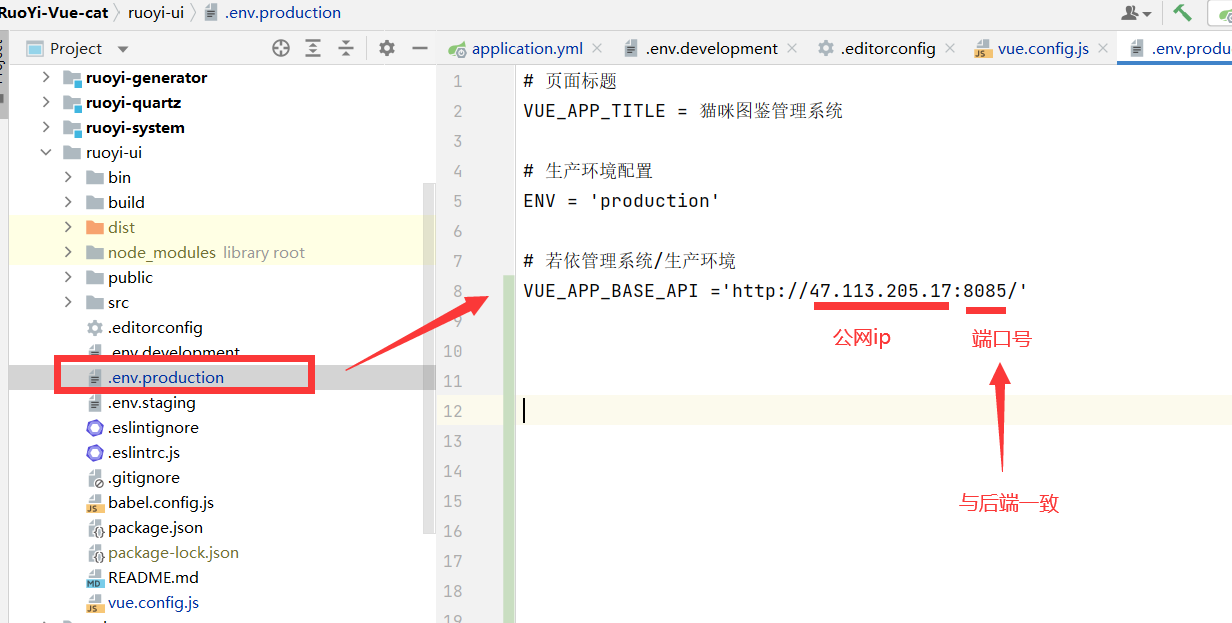
# 访问域名【看看效果】

**成功啦！**



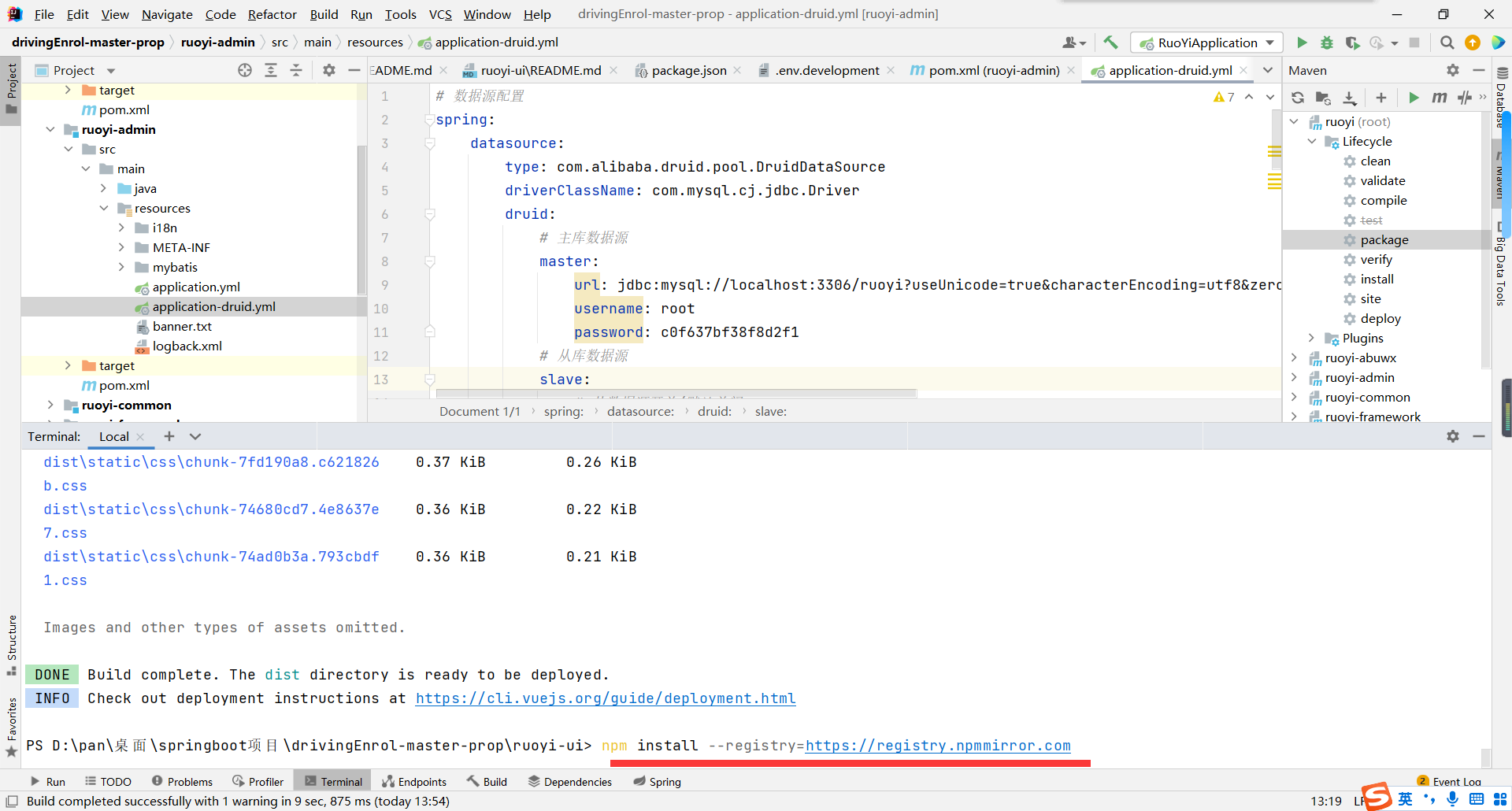
# 无域名版本

## 6.1、前端项目配置 代理端口



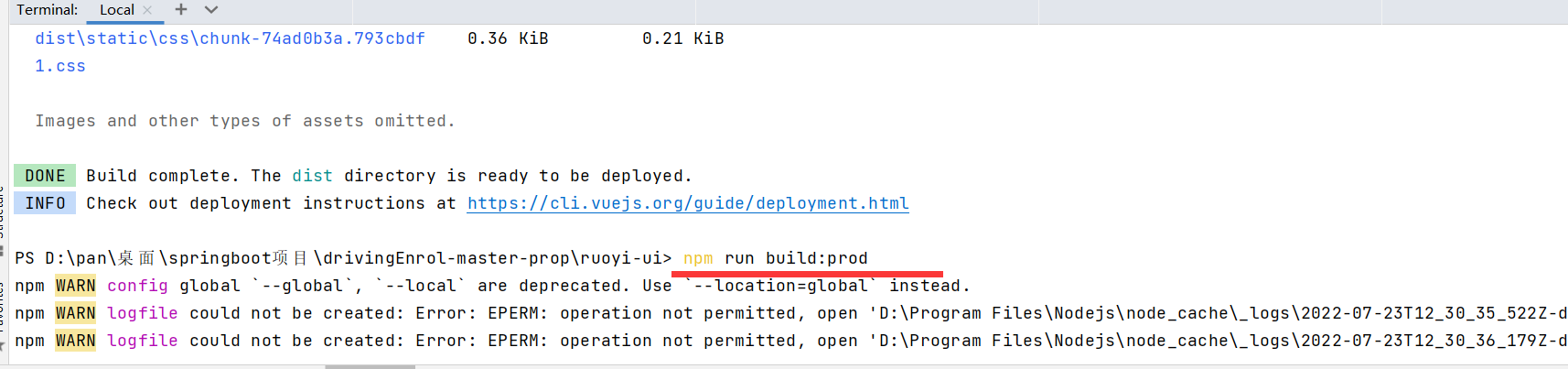
## 6.2、在前端项目中打开终端【执行下面命令】

npm install --registry=https://registry.npmmirror.com

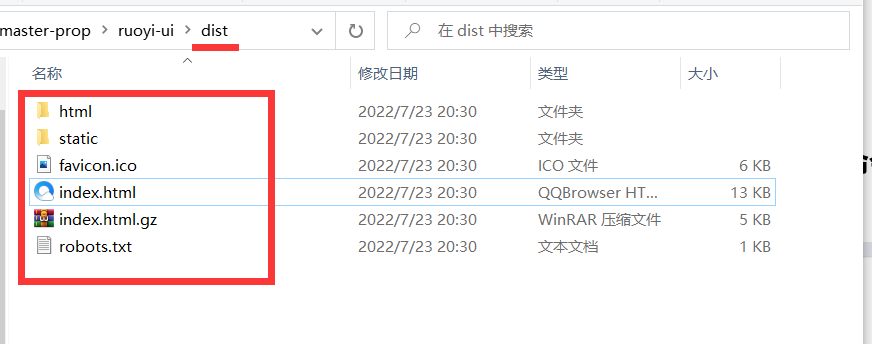


## 6.3、继续在终端执行打包命令

npm run build:prod



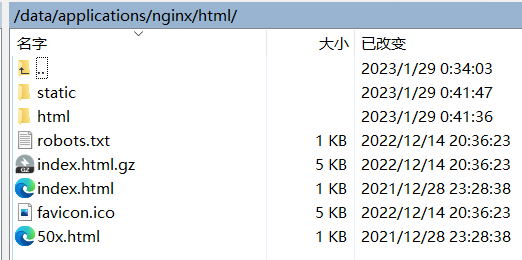
## 6.4、查看dist目录【如下情况，即是 打包成功】



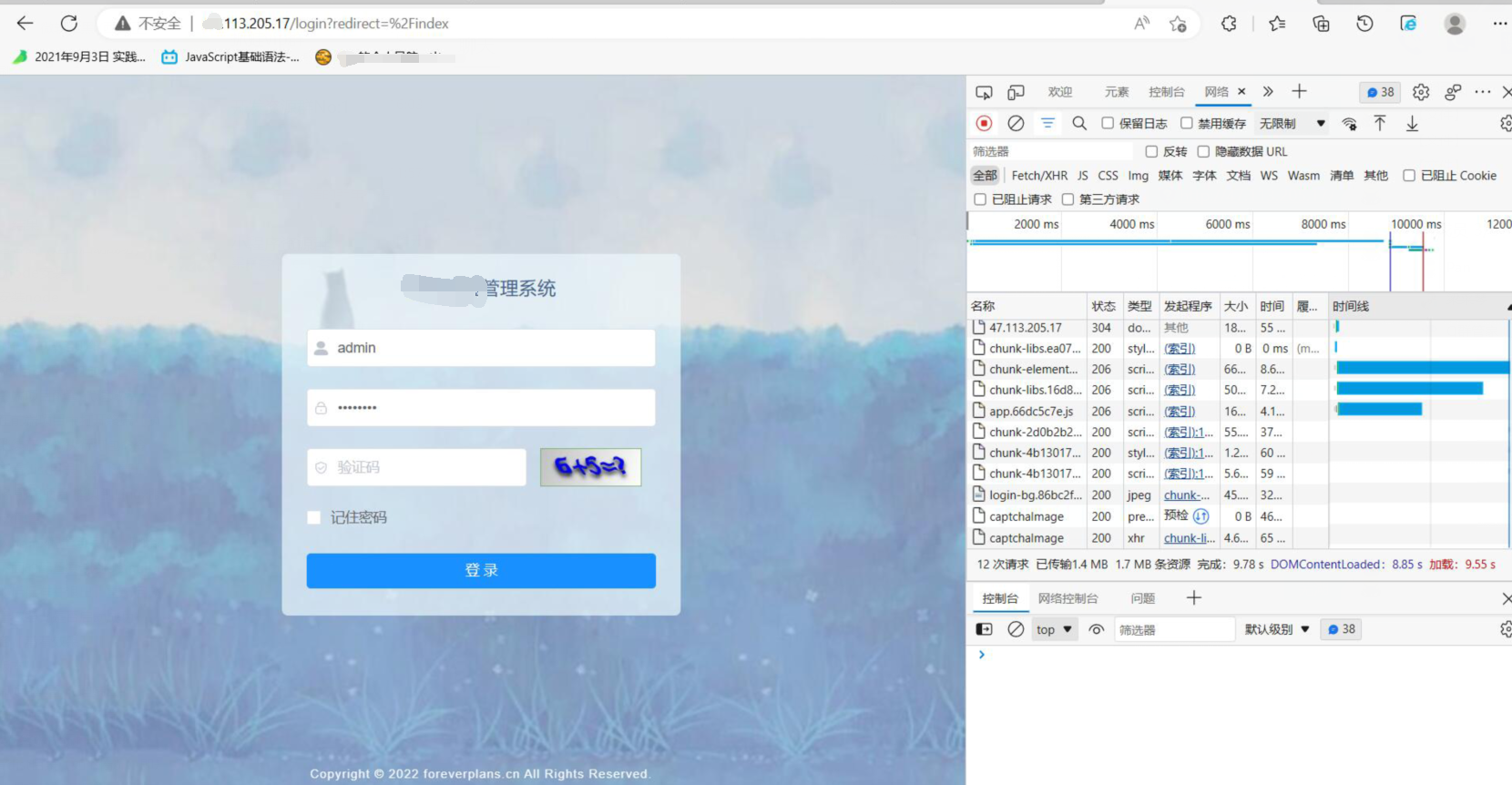
## 6.5、将 前端项目 拷贝到Linux服务器中nginx容器对应的html目录下

**目录下的原文件全部删除！**





## 6.6、访问站点（ 直接使用公网ip即可 ）



# 总结

**当你在有问题时：**

* **确保上述步骤是否操作完成，是否有操作错误**
* **确定自己每次输入的命令行是否在对应的目录下**
* **容器启动失败，可以查看容器日志，进行诊断**
* **无法解决报错时，可将报错复制，进行百度**
* **域名备案的地址一定要和该服务器一致！**