ECS\EC2\Lambda都是计算服务

Lambda 是事件触发-> 解压-> 运行

ECR

镜像的仓库Amazon Elastic Container Registry

EC2实例

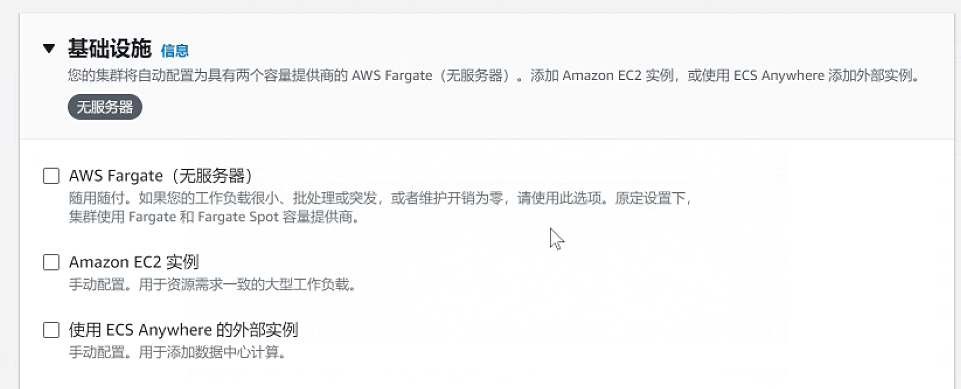
可以通过docker pull 拉取镜像

Docker run 启动容器

ECS

Amazon Elastic Container Service —— 弹性容器服务，运行、停止和管理集群上的容器

ECS可以自动创建EC2实例并对多个EC2实例进行docker管理



Lambda

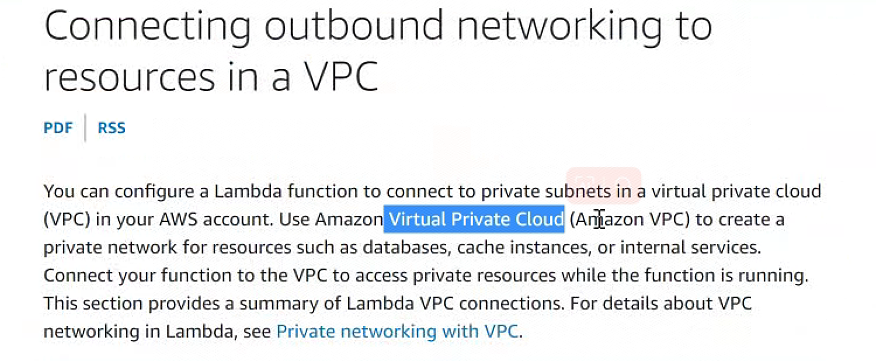
原理：事件触发 -> ECR 拉取镜像->生成容器->运行事件函数

Tips: RDS需要为public

[https://blog.csdn.net/weixin\_39967405/article/details/118813773#:~:text=Yes.%20You%20can%20access%20a%20MySql%20RDS%20database,which%20could%20be%20a%20security%20risk%20for%20you](https://blog.csdn.net/weixin_39967405/article/details/118813773).

官网介绍：

<https://repost.aws/zh-Hans/knowledge-center/connect-lambda-to-an-rds-instance>



需要使用虚拟私有云解决这个问题

系统环境配置：

1.配置EC2，ubuntu

这里如果RDS连接到EC2就不能公开数据库，只能在EC2上访问RDS数据库, 本地navicate需要使用ssh通道访问远端数据库。

<https://repost.aws/zh-Hans/knowledge-center/rds-connect-ec2-bastion-host>

以下示例使用 MySQL Workbench 客户端连接到堡垒机主机：

1.    启动新连接，然后选择 Standard TCP/IP over SSH（通过 SSH 的标准 TCP/IP）作为 Connection Method（连接方式）。

2.    对于 SSH 设置，输入有关 EC2 实例的以下详细信息： \*\*Auto-assign Public IP（自动分配公有 IP）：\*\*确保在 DNS Hostnames（DNS 主机名）选项中选择 Enable（启用）。 \*\*SSH Hostname（SSH 主机名）：\*\*输入 EC2 实例的公有 DNS 名称或其公有 IP 地址。 \*\*SSH Username（SSH 用户名）：\*\*输入 EC2 实例的用户名称。例如，ec2-user 是 EC2 Linux 计算机的用户名称。 \*\*SSH Key File（SSH 密钥文件）：\*\*选择在创建 EC2 实例时使用的私有密钥。 \*\*注意：\*\*如果创建某个 EC2 实例时所在的 VPC 已激活 DNS Hostnames（DNS 主机名），则使用公有 IP 地址启动的 EC2 实例中具有公有 DNS。

3.    输入有关 MySQL 实例设置的以下详细信息： \*\*MySQL Hostname（MySQL 主机名）：\*\*输入 RDS 数据库实例端点。 \*\*MySQL Server port（MySQL Server 端口）：\*\*输入 3306。或者，如果您使用自定义端口，则输入自定义端口号。 \*\*Username（用户名）：\*\*输入 RDS 数据库实例的用户名称。 \*\*Password（密码）：\*\*输入 RDS 数据库实例的密码。

4.    选择 Test Connection（测试连接）。

5.    连接成功后，输入连接名称，然后保存连接。

要使用 SSH 隧道从本地 MySQL 客户端连接到私有 RDS 实例

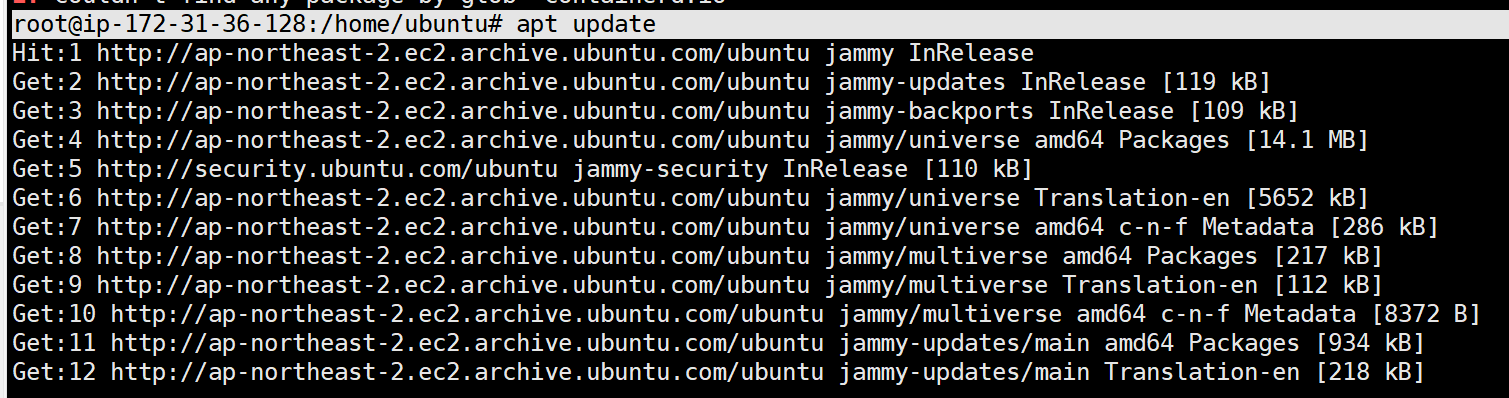
# 配置运行环境：

## 1. 安装docker

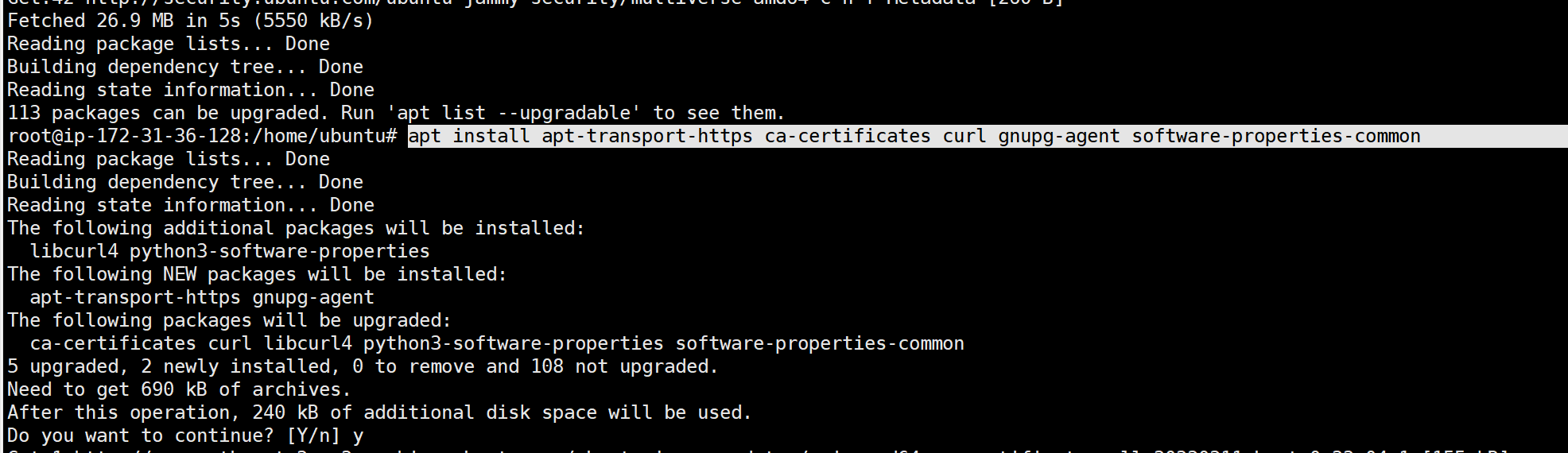
sudo -s

第一步获取权限，减少sudo麻烦

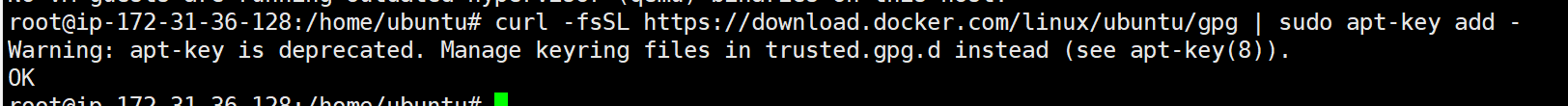
apt update



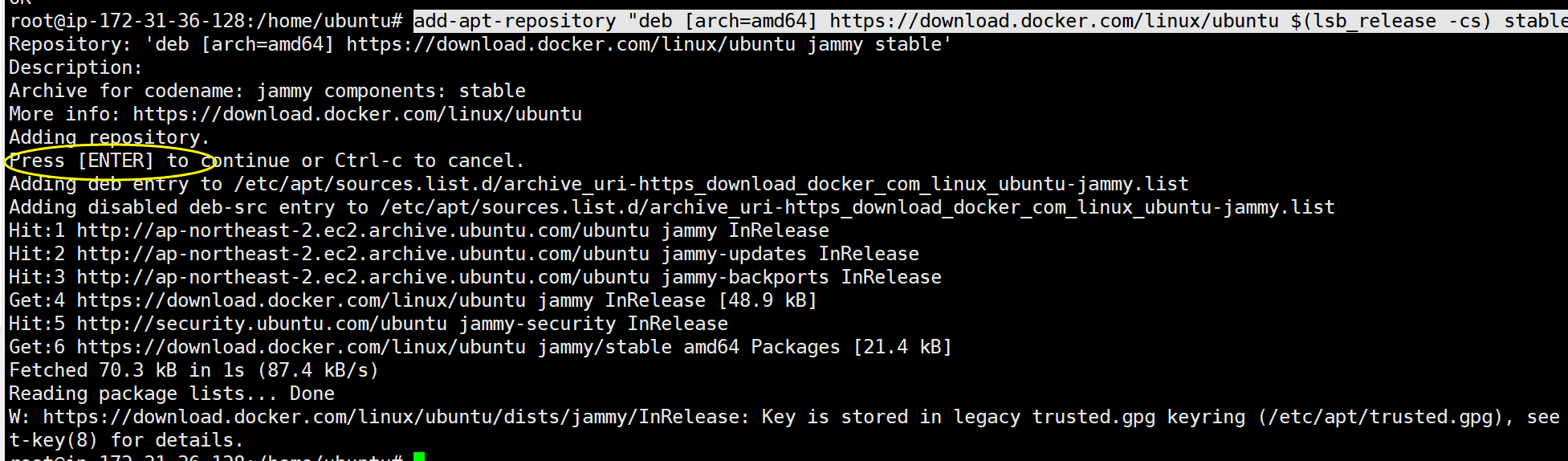
apt install apt-transport-https ca-certificates curl gnupg-agent software-properties-common



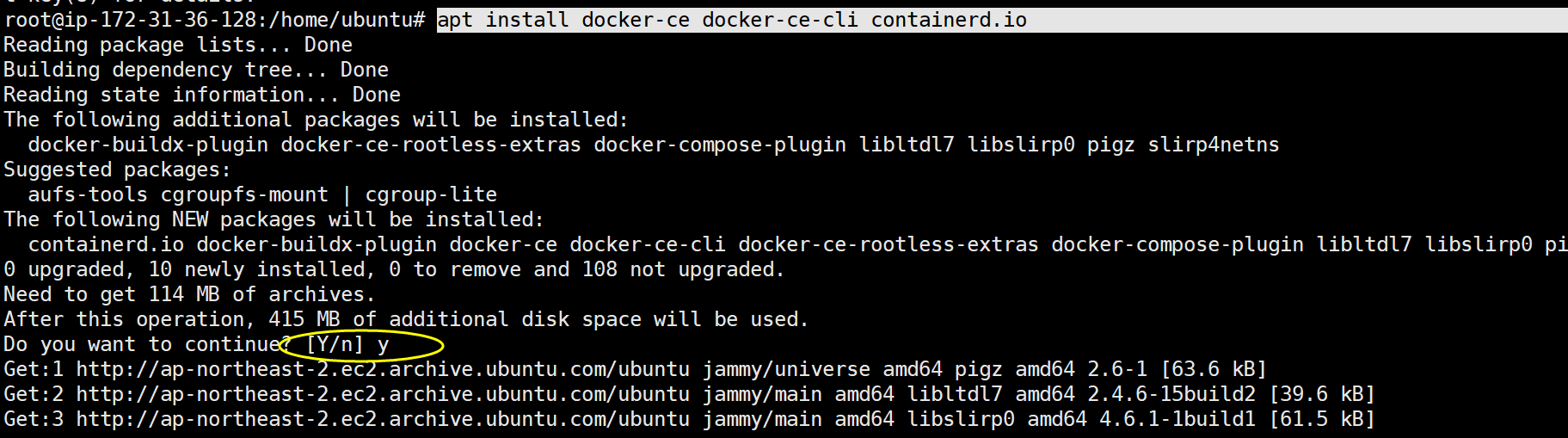
curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | sudo apt-key add -



add-apt-repository "deb [arch=amd64] <https://download.docker.com/linux/ubuntu> $(lsb\_release -cs) stable"



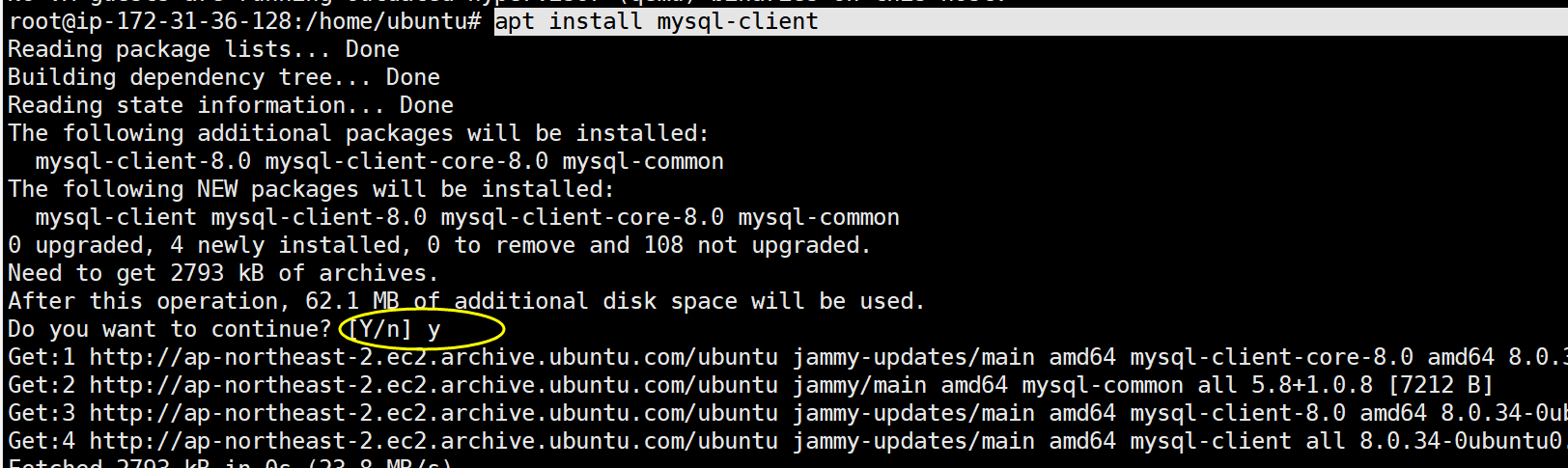
apt install docker-ce docker-ce-cli containerd.io

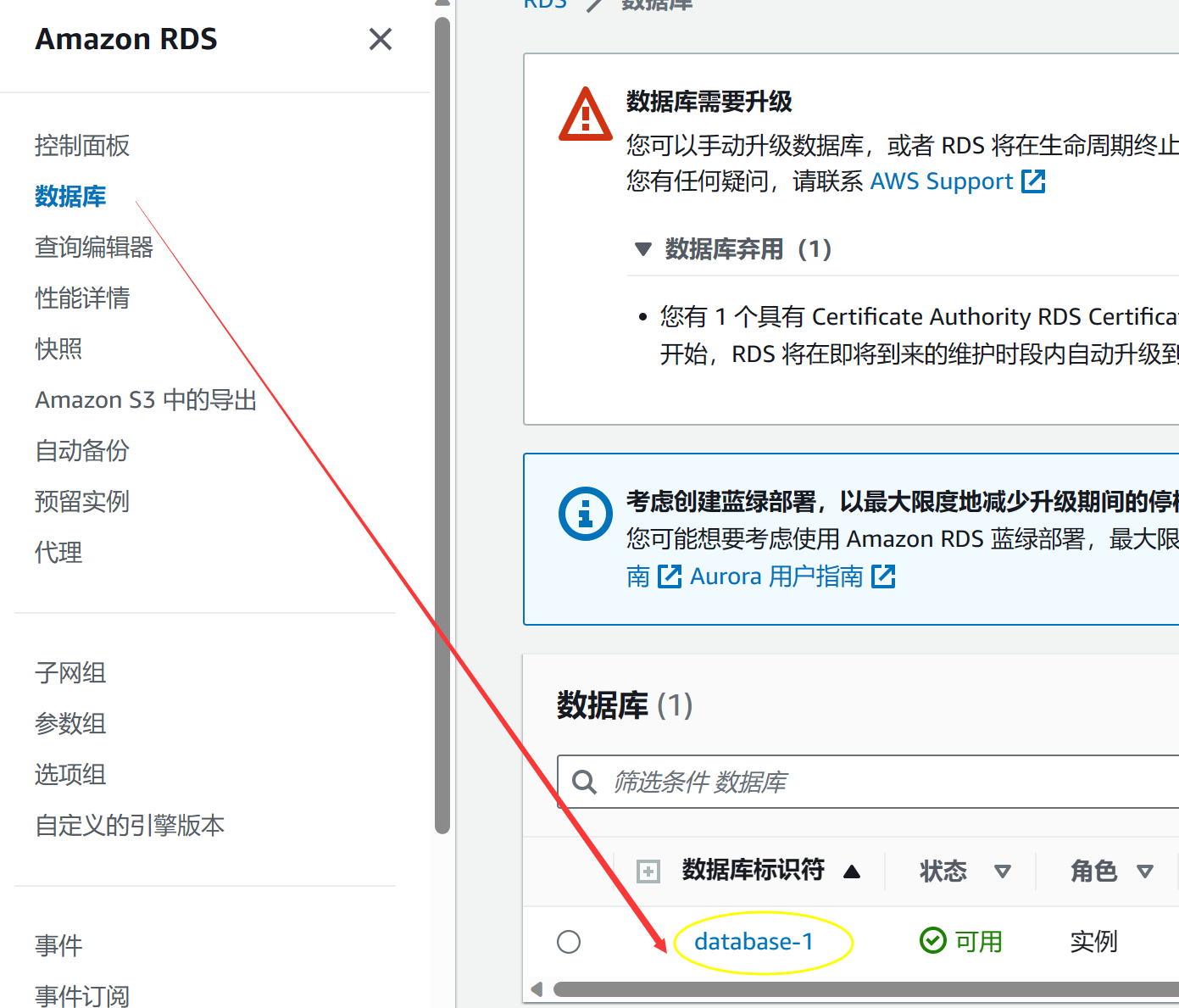


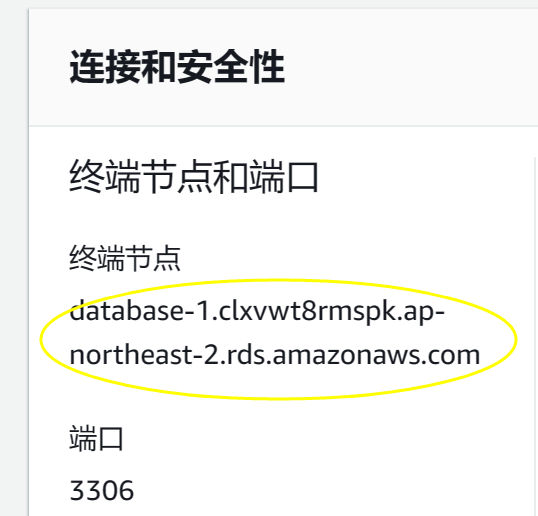
## 安装mysql

### 2.1安装mysql-client

apt install mysql-client

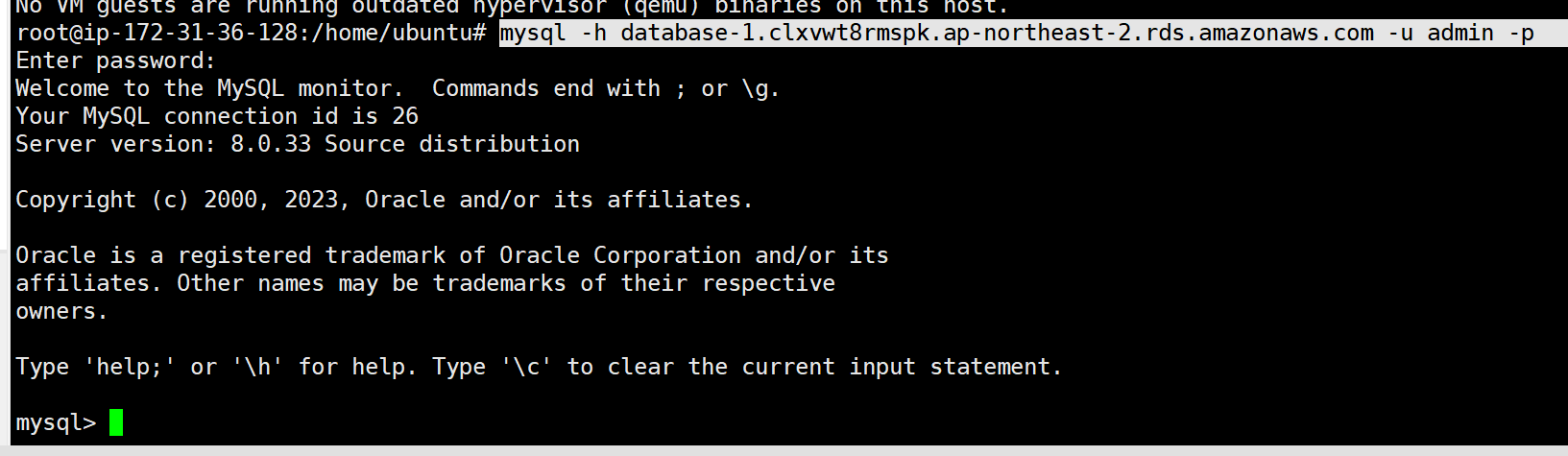






mysql -h 终端节点 -u admin -p

例：mysql -h database-1.clxvwt8rmspk.ap-northeast-2.rds.amazonaws.com -u admin -p



### 2.2数据库添加

create database digital;

use digital;

CREATE TABLE IF NOT EXISTS `user`(

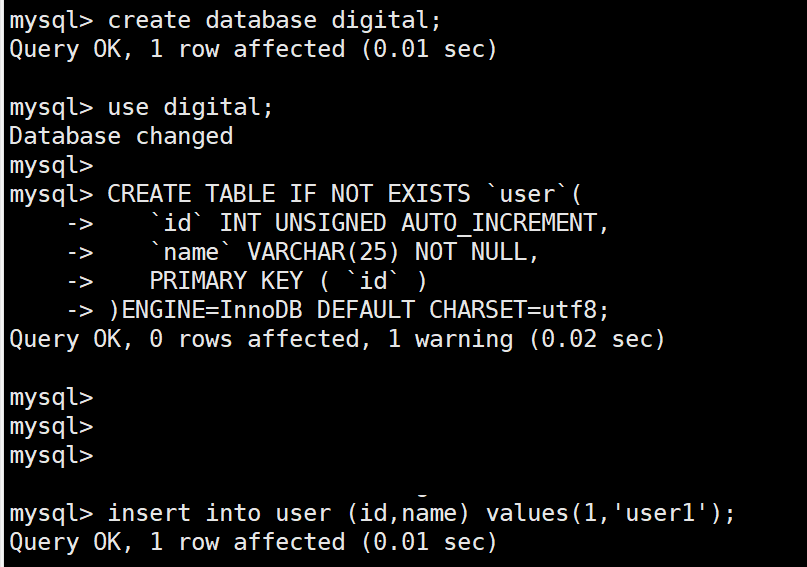
   `id` INT UNSIGNED AUTO\_INCREMENT,

   `name` VARCHAR(25) NOT NULL,

   PRIMARY KEY ( `id` )

)ENGINE=InnoDB DEFAULT CHARSET=utf8;

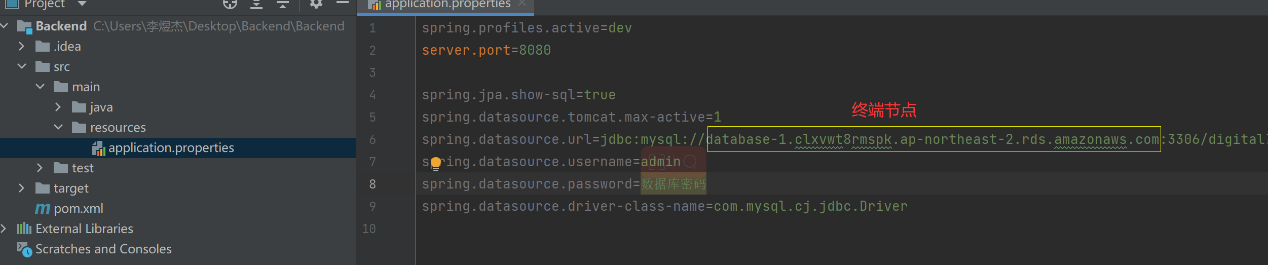
insert into user (id,name) values(1,'user1');

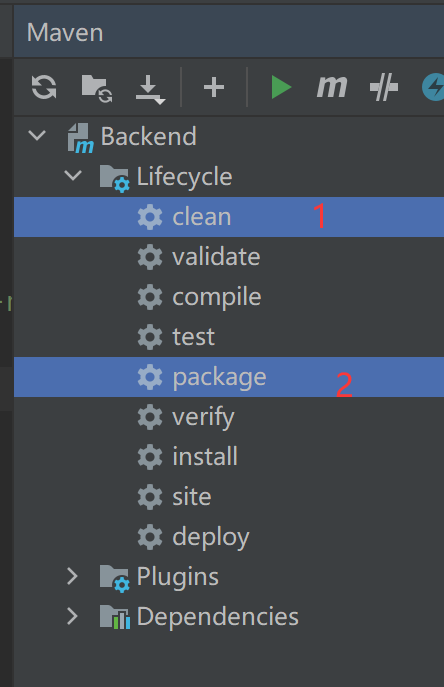


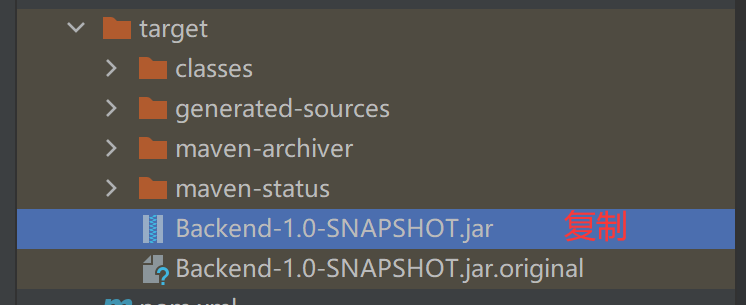
## 3.制备部署包

### 3.1后端部署包

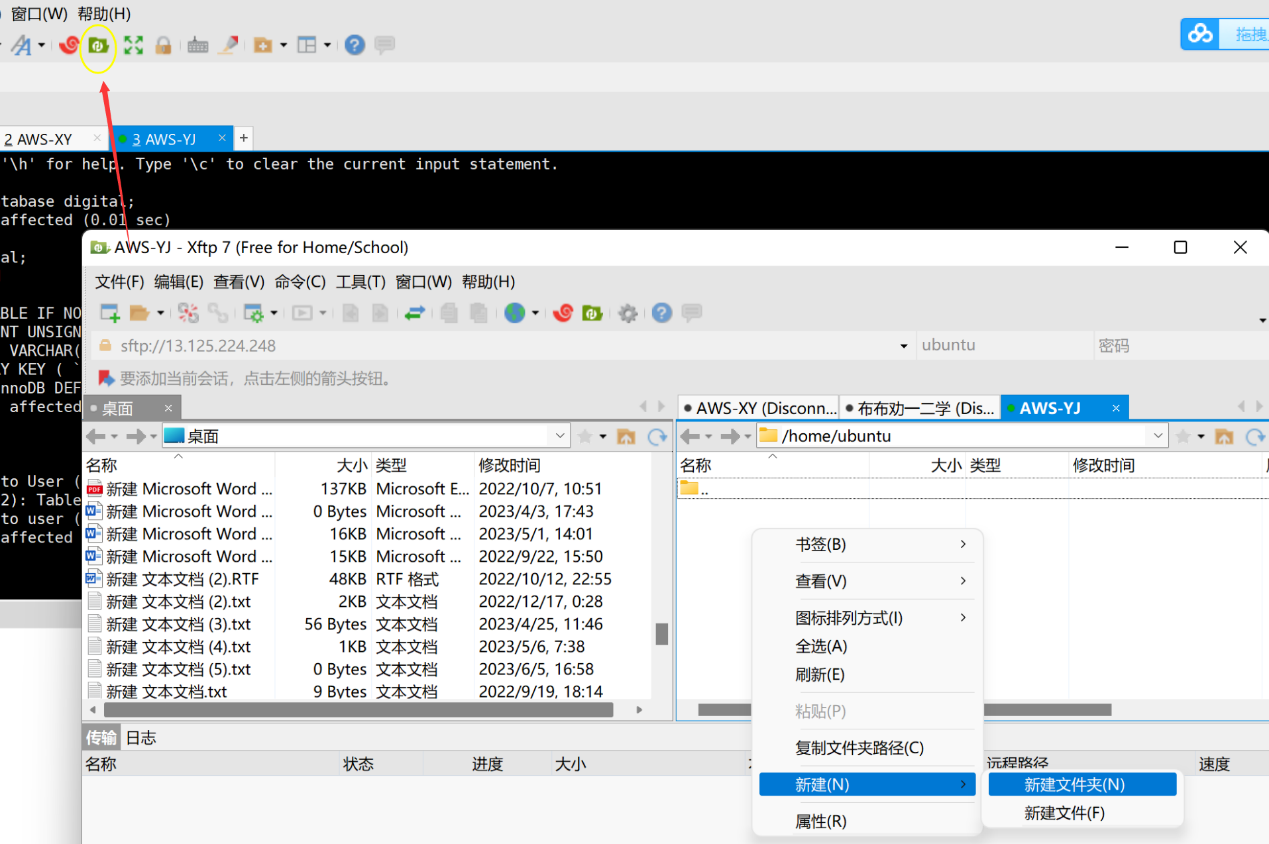
后端



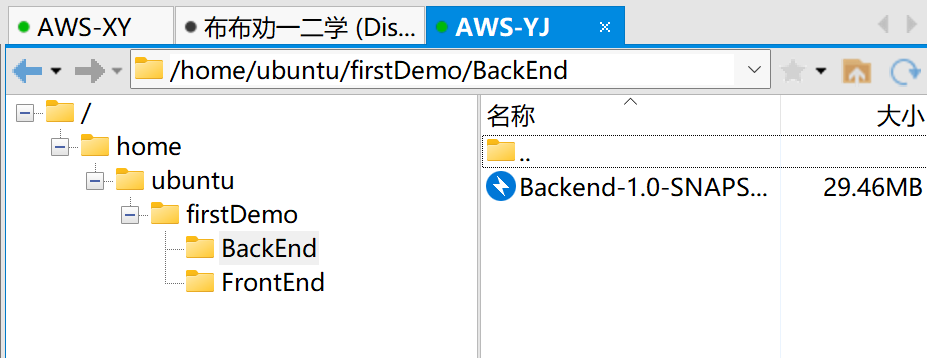




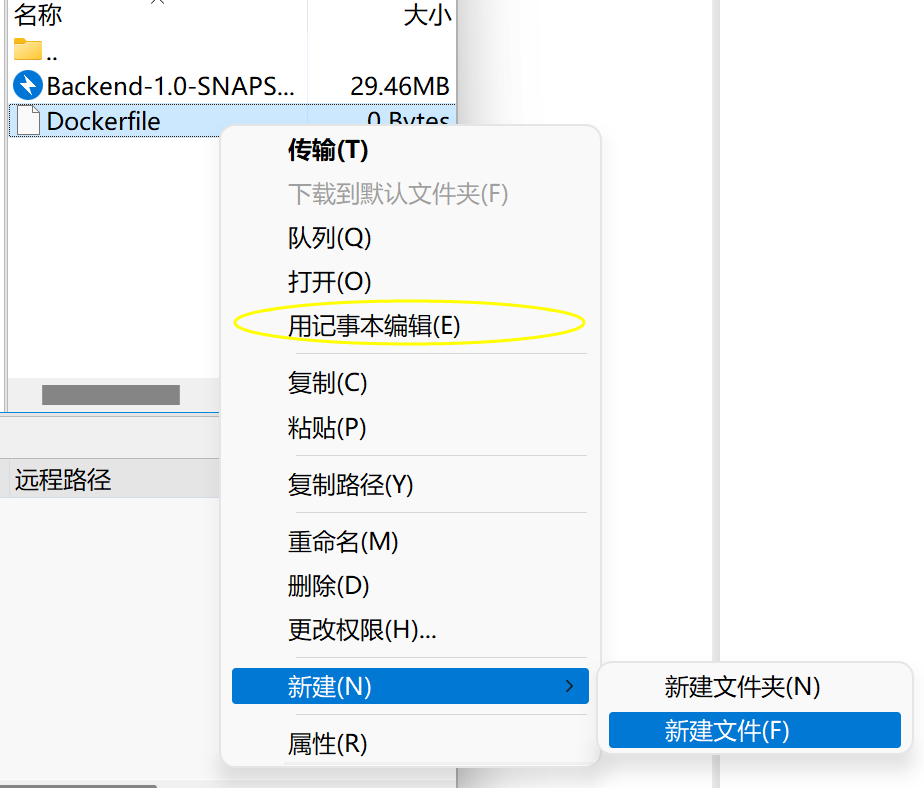
打开xftp



粘贴



新建Dockerfile文件，没有后缀，用记事本编辑



粘贴，保存

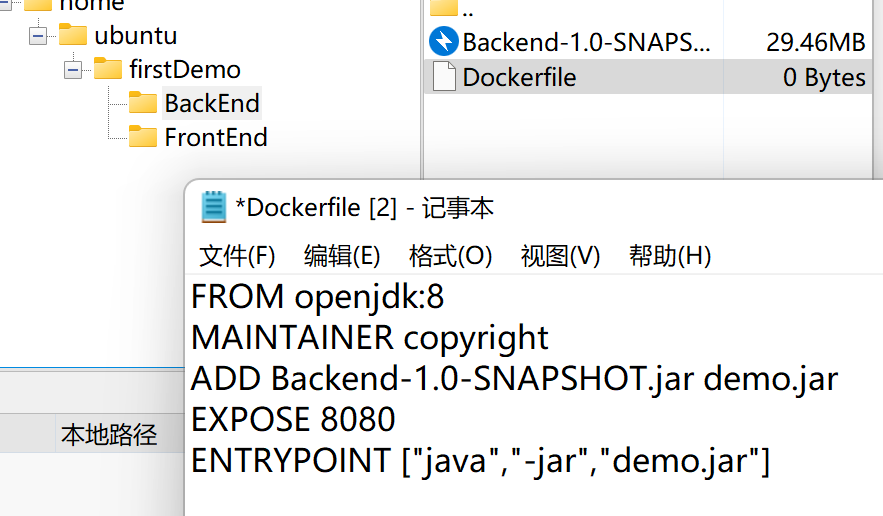
FROM openjdk:8

MAINTAINER copyright

ADD Backend-1.0-SNAPSHOT.jar demo.jar

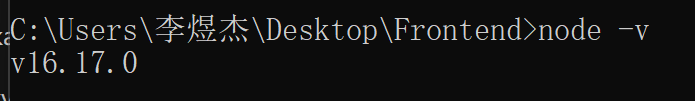
EXPOSE 8080

ENTRYPOINT ["java","-jar","demo.jar"]



### 3.2前端部署包

百度搜索下载安装Node.js，确保14.20版本以上



tips：想要删除node\_modules，用rimraf工具

npm install rimraf -g

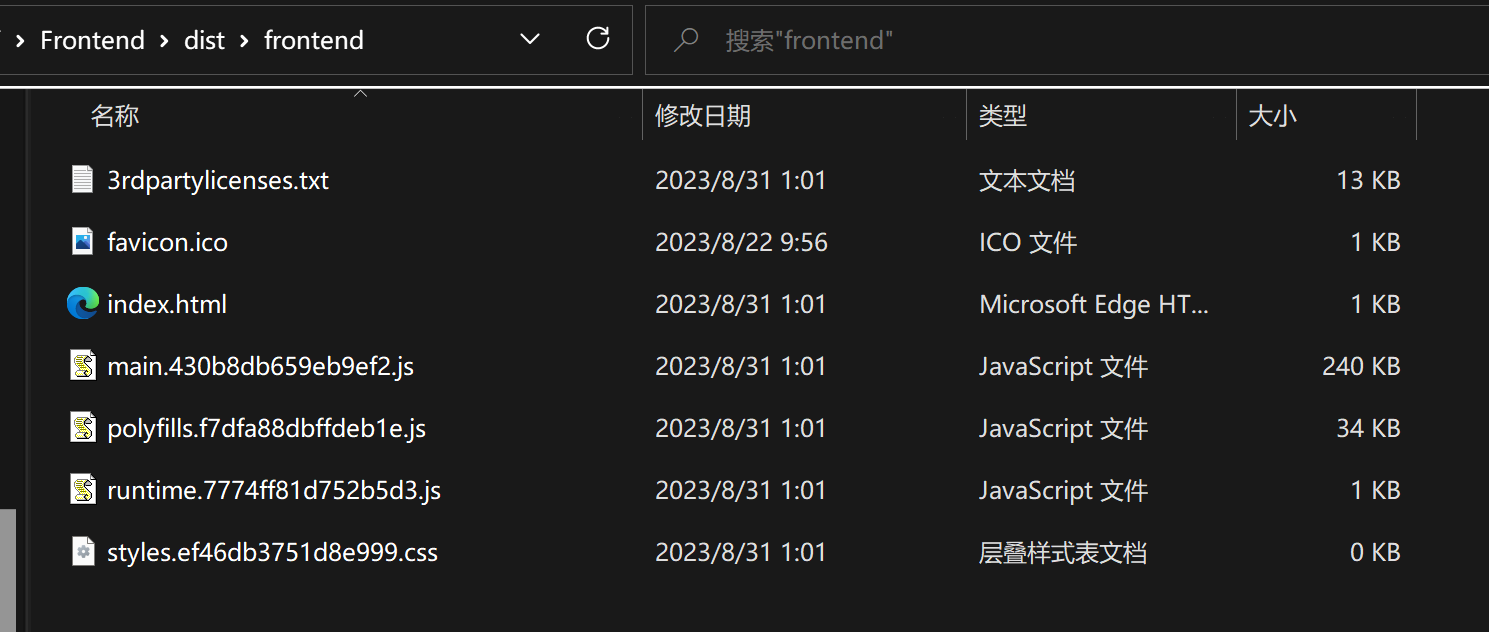
rimraf node\_modules

npm install

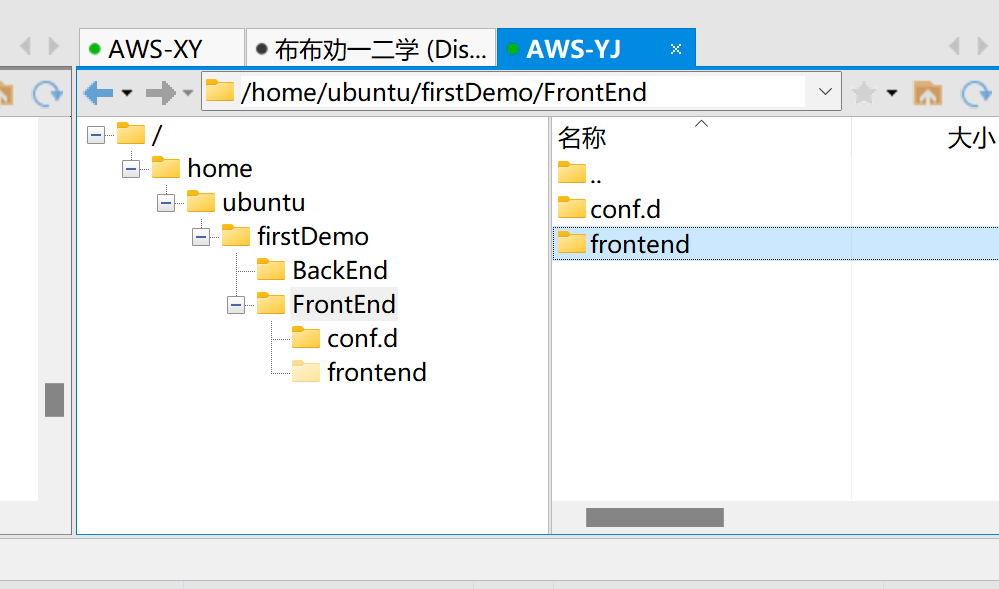
npm build -prod



找到这个文件夹，复制整个frontend文件夹



 粘贴，并建一个conf.d文件夹



在conf.d里面建一个default.conf，右键用记事本编辑，粘贴，修改，保存

 server {

    listen       4200; # nginx监听端口，angular默认端口，这个是容器内端口，可以在docker启动时映射

    location /api/user/ {

        proxy\_pass <http://13.125.224.248:8080/user/>; #改这里，ip地址换成EC2的ip

    }

    location / {

        root   /usr/share/nginx/html; #test项目路径

        index index.html index.html; #默认起始页

        try\_files $uri $uri/ /index.html; #前端项目路由配置

    }

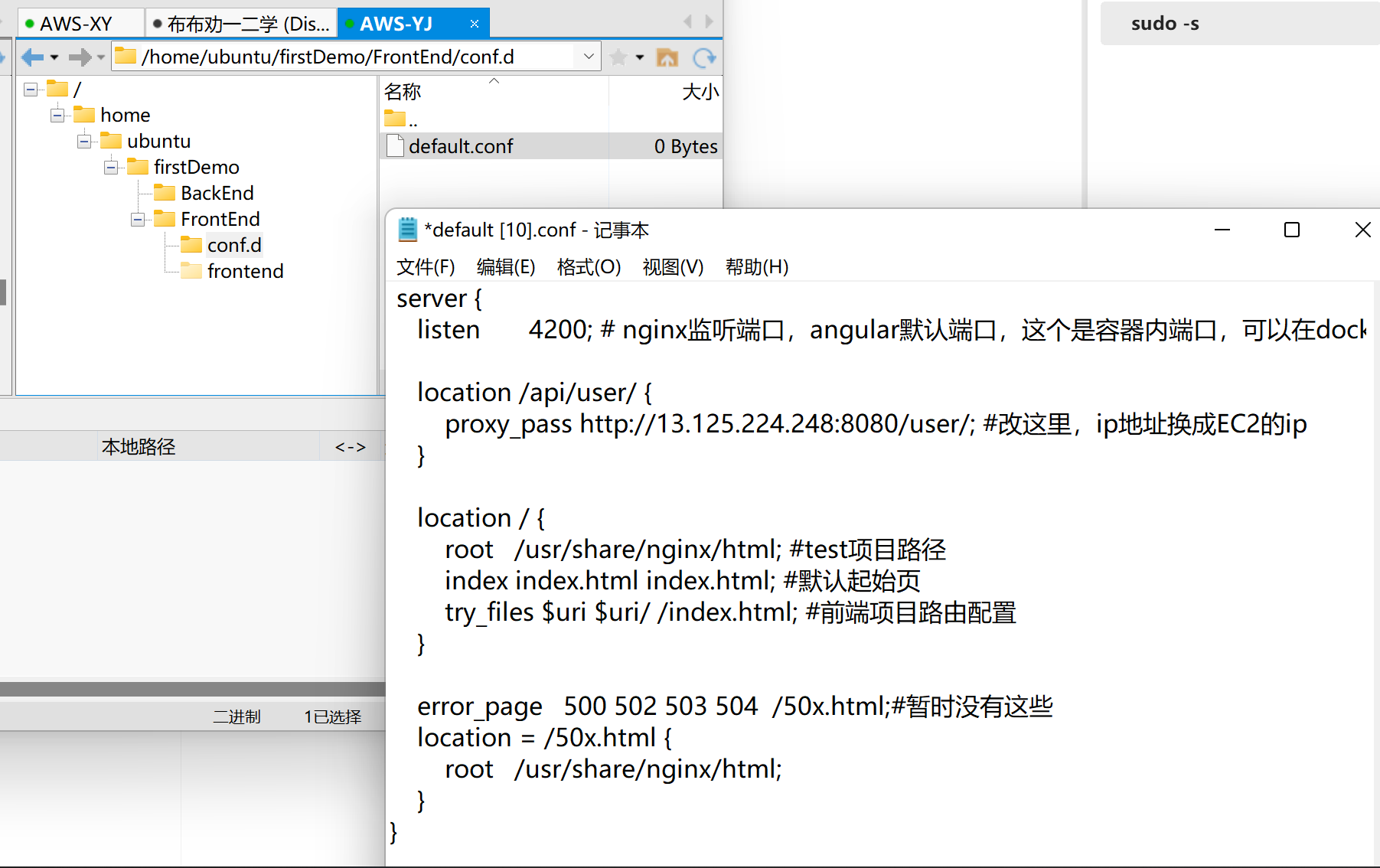
    error\_page   500 502 503 504  /50x.html;#暂时没有这些

    location = /50x.html {

        root   /usr/share/nginx/html;

    }

}



前后端部署包制备完成

## 4.docker拉取镜像

拉俩镜像

docker pull openjdk:8

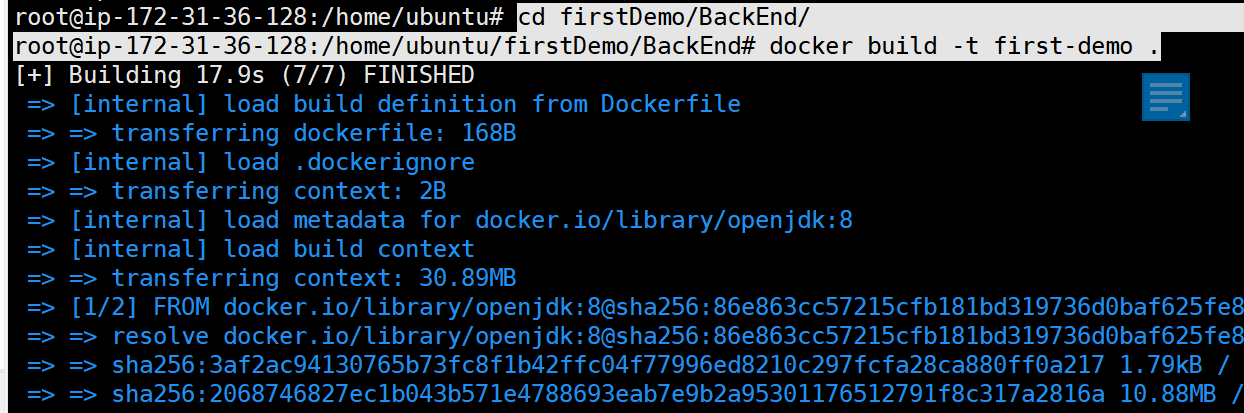
docker pull nginx

进后端文件夹

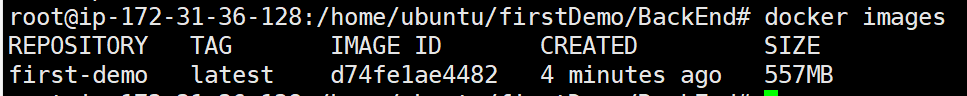
cd firstDemo/BackEnd/

这里有Dockerfile文件，build他（注意最后空格和点）

docker build -t first-demo .

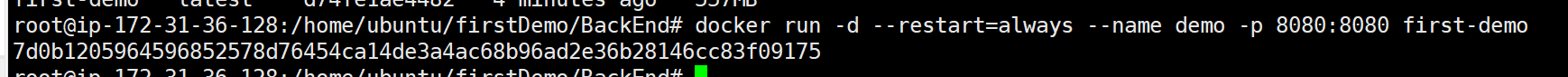


docker images查看构建的镜像



docker run -d --restart=always --name demo -p 8080:8080 first-demo

（-restart=always可以去掉，该命令意味着如果出错的话会自动重启）

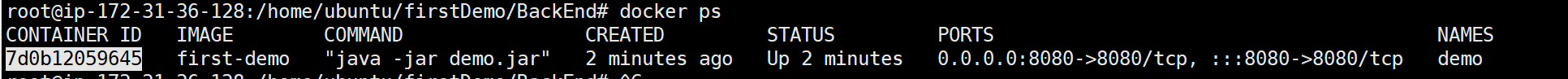


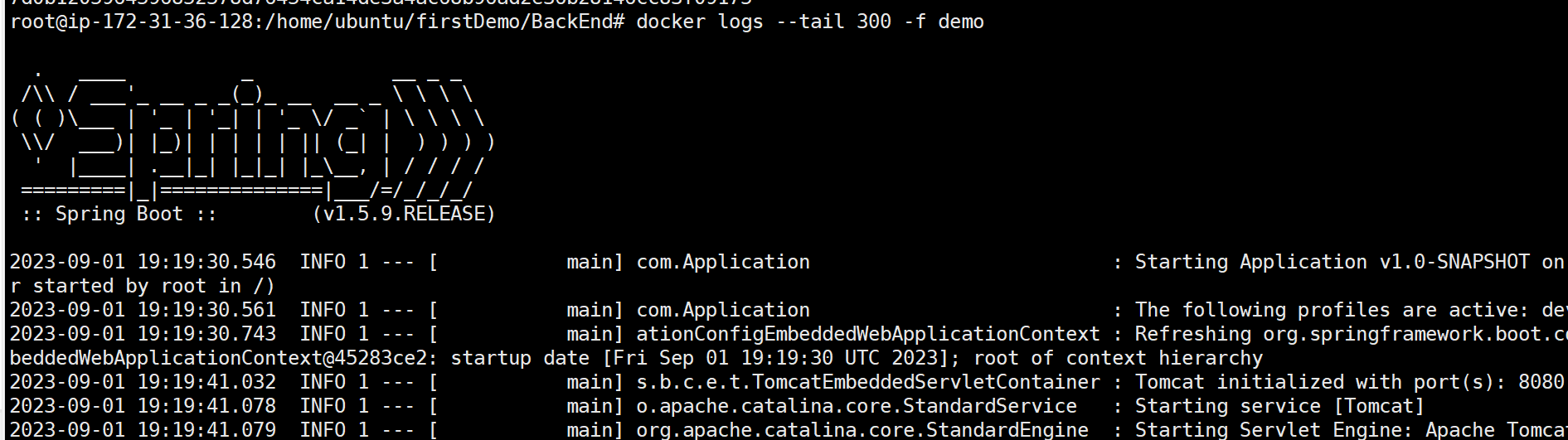
docker logs --tail 300 -f 容器id ,可以查看容器的日志。

（ctrl+c退出）

（容器id可以是run命令跟在-name后面的name，也可以是docker ps看到的CONTAINER ID，图中是7d0b12059645）

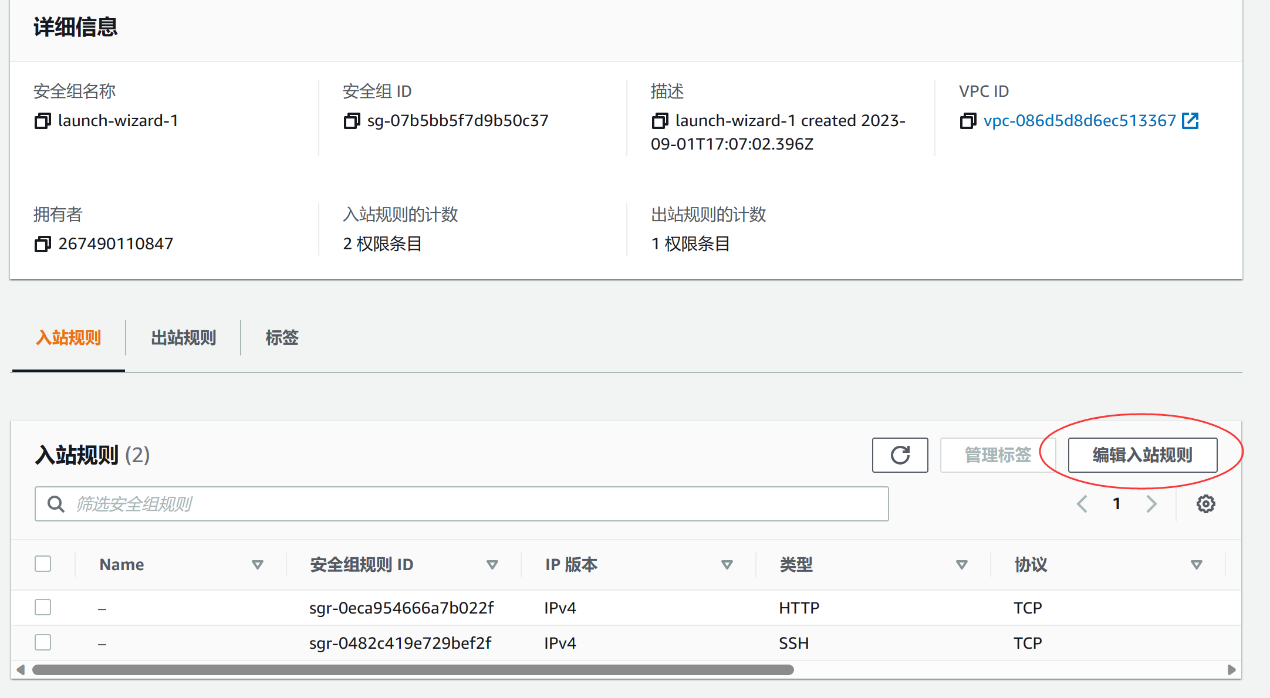
docker ps查看运行中的容器

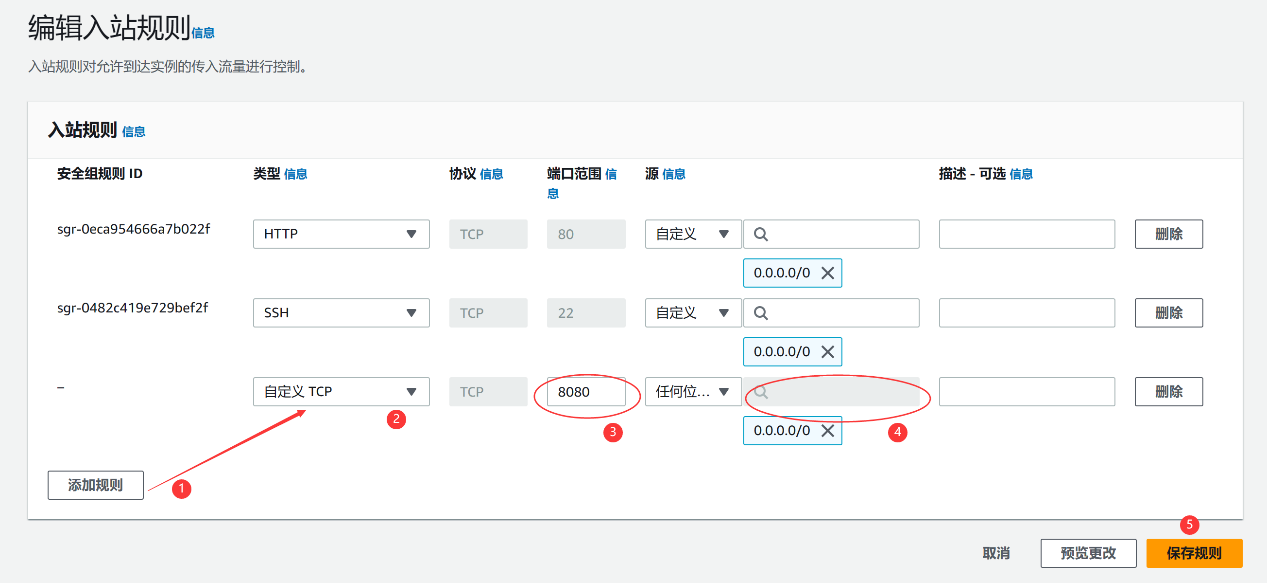




后端部署完成，想要从浏览器尝试后端api可以开放8080端口（为了安全也可以不开，容器内部可以连上l）







浏览器访问  ip:8080/user/search



tips:如果改了代码想换jar包，可以直接Backend-0.0.1-SNAPSHOT.jar是宿主机里的jar名，demo.jar是容器内的名字，在Dockerfile定义过

docker cp Backend-0.0.1-SNAPSHOT.jar 容器ID:/demo.jar

例：

docker cp Backend-0.0.1-SNAPSHOT.jar 0586c3815d74:/demo.jar

然后是前端

为了方便理解，前一段换行了，但是这是一条命令，应该用后一段没换行的

docker run -e TZ="Asia/Shanghai" -d -p 80:4200 --name first-demo

-v /home/ubuntu/firstDemo/FrontEnd/conf.d/default.conf:/etc/nginx/conf.d/default.conf

-v /home/ubuntu/firstDemo/FrontEnd/conf/nginx.conf:/etc/nginx/nginx.conf

-v /home/ubuntu/firstDemo/FrontEnd/frontend:/usr/share/nginx/html -d nginx:latest

-v是挂载，冒号前面是宿主机地址，后面是容器内地址；

-p是端口映射，冒号前是宿主机端口，后面是容器内端口

docker run -e TZ="Asia/Shanghai" -d -p 80:4200 --name first-demo -v /home/ubuntu/firstDemo/FrontEnd/conf.d/default.conf:/etc/nginx/conf.d/default.conf -v /home/ubuntu/firstDemo/FrontEnd/frontend:/usr/share/nginx/html -d nginx:latest

如果安全组添加了80端口，那么现在可以通过ip访问了