# DockerCompose

## 前言：

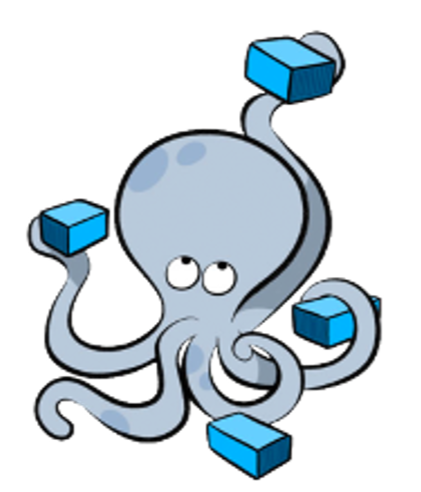
前面我们已经实现了用Docker来部署像Redis、Mysql这样的中间件，还有利用Dockerfile来实现微服务的自定义镜像构建还有docker部署，但是，所有的这些部署都是我们手动一个个完成的，你想象一下，在实际生产环境下，微服务的数量可是非常多的，这么多的微服务我们都一个个去构建镜像、部署容器吗？！这个工作量几乎是不可能的，所以我们一定需要一种集群部署的手段，这就是我们接下来学习的DockerCompose。

DockerCompose：众多Docker镜像集群部署的工具！

## 初始DockerCompose：

### 什么是DockerCompose?

请看这个章鱼：



见图知意，章鱼手上拿的是容器，章鱼在帮我们部署容器呢！**DockerCompose可以基于Compose文件帮我们快速部署分布式应用，**而无需手动一个个创建镜像和一个个运行容器！即DockerCompose是部署分布式应用的帮手，但是DockerCompose做集群部署，是要基于Compose文件才行。

注意不要把Dockerfile和DockerCompose弄混哦！

1：Docker基于Dockerfile文件构建自定义docker镜像

2：DockerCompose基于Compose文件快速部署分布式应用，即快速构建并运行多个docker镜像

### Compose文件

#### 什么是Compose文件？

**Compose文件是一个文本文件！通过指令来定义集群中每个容器如何运行。可以回顾一下，我们以前是通过docker run命令来运行每个容器，而现在是用Compose文件来定义集群中的N个容器如何运行，于是可以认为Compose文件就是N个docker run命令的集合，事实上就是如此！**既然是多个docker run命令的集合，那么不就可以快速部署分布式应用了吗！只不过，Compose文件不是直接用docker run，而是把run里面的东西用另外一种语法：即指令来代替。这个语法长什么样呢？见下：

version: "3.8"

services:

  mysql:

    image: mysql:5.7.25

environment:

MYSQL\_ROOT\_PASSWORD: 123

    volumes:

     - "/tmp/mysql/data:/var/lib/mysql"

     -  "/tmp/mysql/conf/hmy.cnf:/etc/mysql/conf.d/hmy.cnf"

  web:

    build: .

    ports:

     - "8090:8090"

**可以看到，这个语法整体是一个YAML格式！这点要时刻记住，这样就比较容易记忆了！记住了：Compse文件就是定义了docker镜像构建并docker容器运行命令的文件！**

#### Compose文件格式

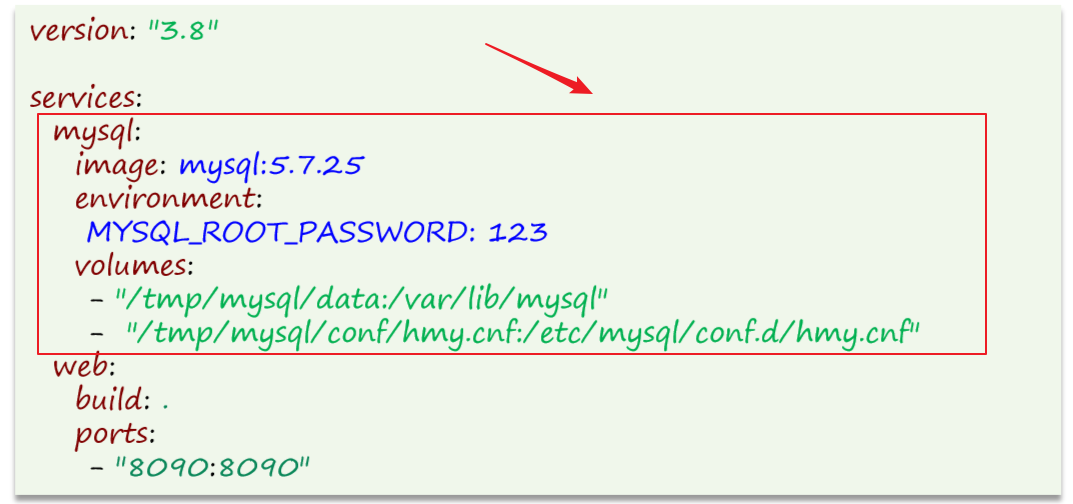
##### version：是compose语法的版本

Compose目前有三个大的版本，分别是1.x，2.x，3.x，这里用3.8的版本！

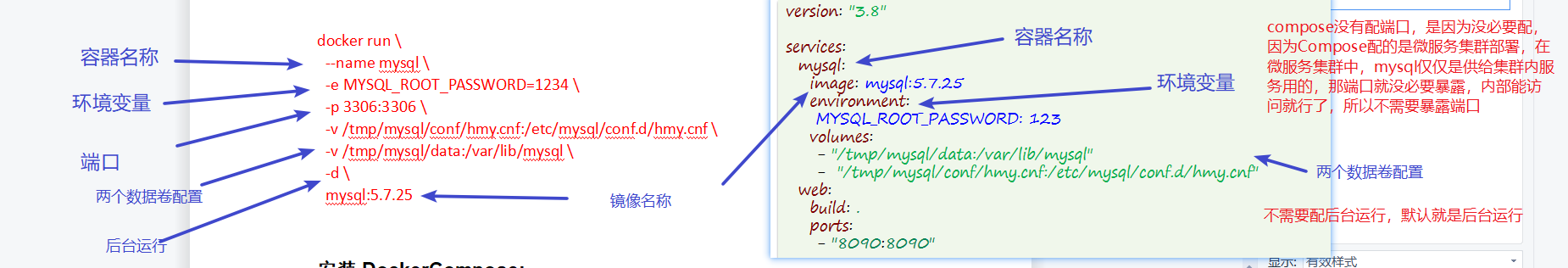
##### services：version往下就是具体的微服务定义，可以看到上面例子的services有两个二级配置：

例子中展示了mysql、web，那么就是两个微服务配置，我们可以看到上面的是mysql的配置；下面的web是web工程的配置。这是Compose文件中比较典型的两种配置方式。

###### 第一种方式：基于镜像直接部署，这里的mysql是第一种部署服务的方式，就是基于镜像直接部署！



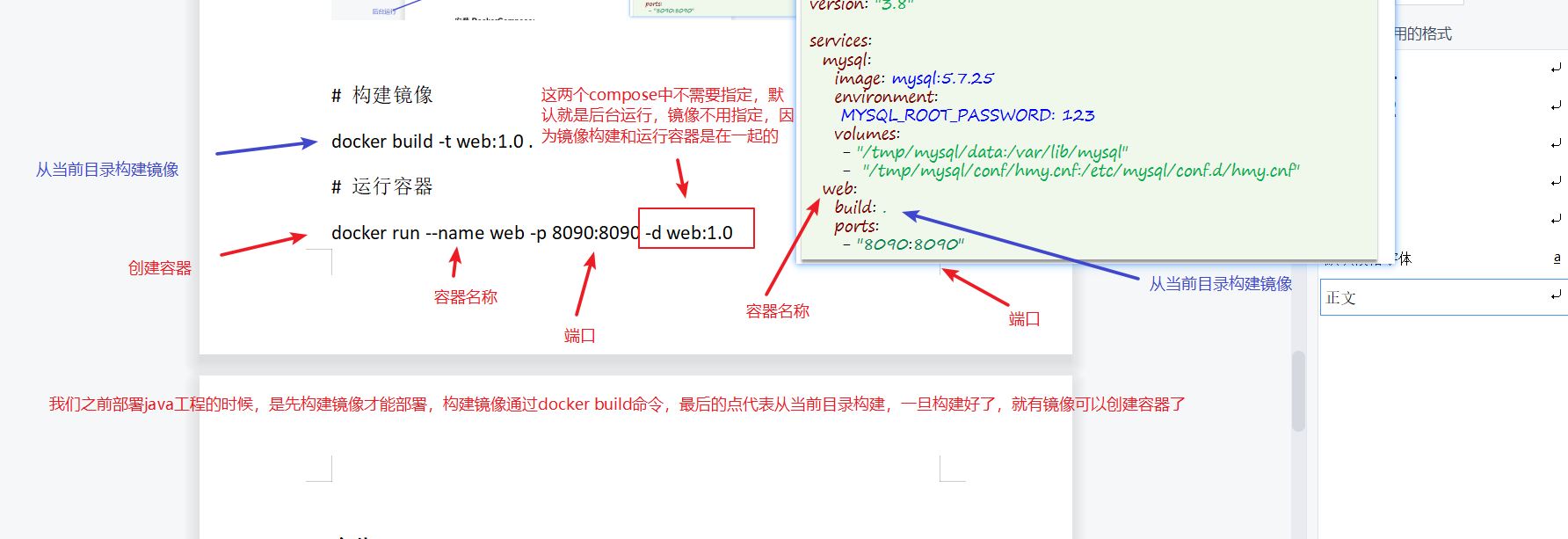
即然services是在定义容器的部署，我们有说了Compose文件是吧docker run集合进来了，然后把语法转换了一下：我们可以比较一下：见下图，可以发现，确实把docker run中的所有参数都转换成了Compose中的指令了！



###### 第二种方式：临时构建镜像并运行，下面的第二种web没有指定镜像，容器部署没有镜像怎么部署呢？其实下面是把docker build和docker run都包含进来，即临时构建镜像并运行。

其实这种方式就是没有镜像，要先基于Dockerfile文件构建出镜像，然后再运行镜像；而第一种是已经有了镜像（mysql）!





###### 总结：

从上面可以看出，Compose文件就是把Docker run的各种参数转化成指令（如mysql方式），就是把镜像构建docker build并运行镜像docker run转化成指令（如web方式）！没什么难度！

Docker Compose的详细语法请参考官网：

<https://docs.docker.com/compose/compose-file/>

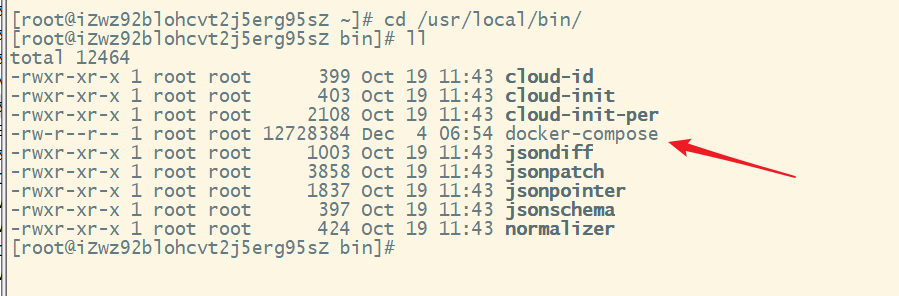
### 安装DockerCompose:

#### 1：下载DockerCompose

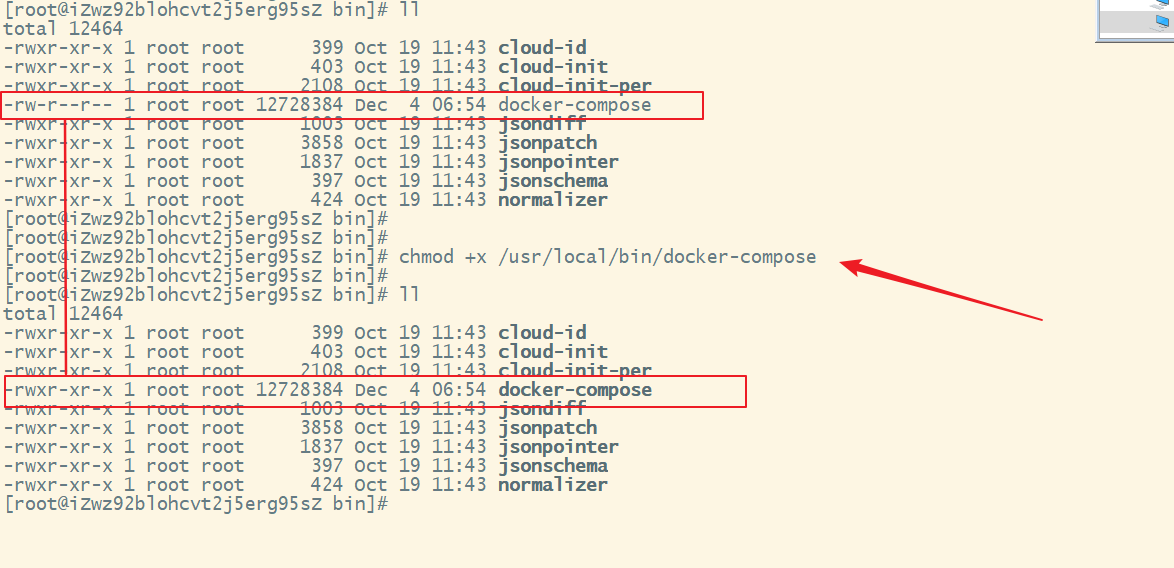
Linux下需要通过命令下载：不推荐使用

curl -L https://github.com/docker/compose/releases/download/1.23.1/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

如果下载速度较慢，或者下载失败，可以使用课前资料提供的docker-compose文件，上传到/usr/local/bin目录也可以。



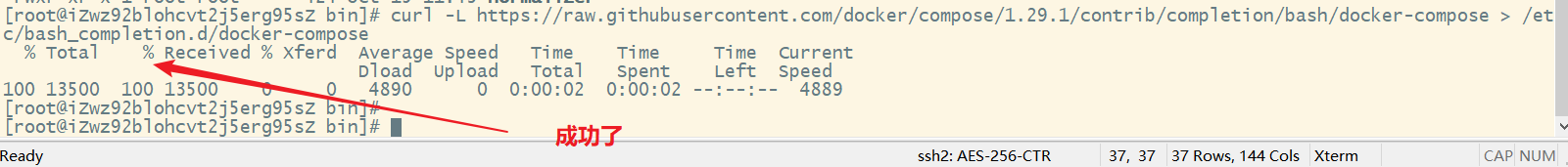
#### 2：修改文件权限：chmod +x /usr/local/bin/docker-compose



以上，DockerCompose就装好了，就是上传上去配置一下就行！

#### 3：base自动补全命令（以以后便用DockerCompose时有提示）：

curl -L https://raw.githubusercontent.com/docker/compose/1.29.1/contrib/completion/bash/docker-compose > /etc/bash\_completion.d/docker-compose



#### 4：3如果出现错误，需要修改自己的hosts文件后再次执行3：当然第3步如果成功这里就不用做了

echo "199.232.68.133 raw.githubusercontent.com" >> /etc/hosts

#### 总结：DockerCompose作用：

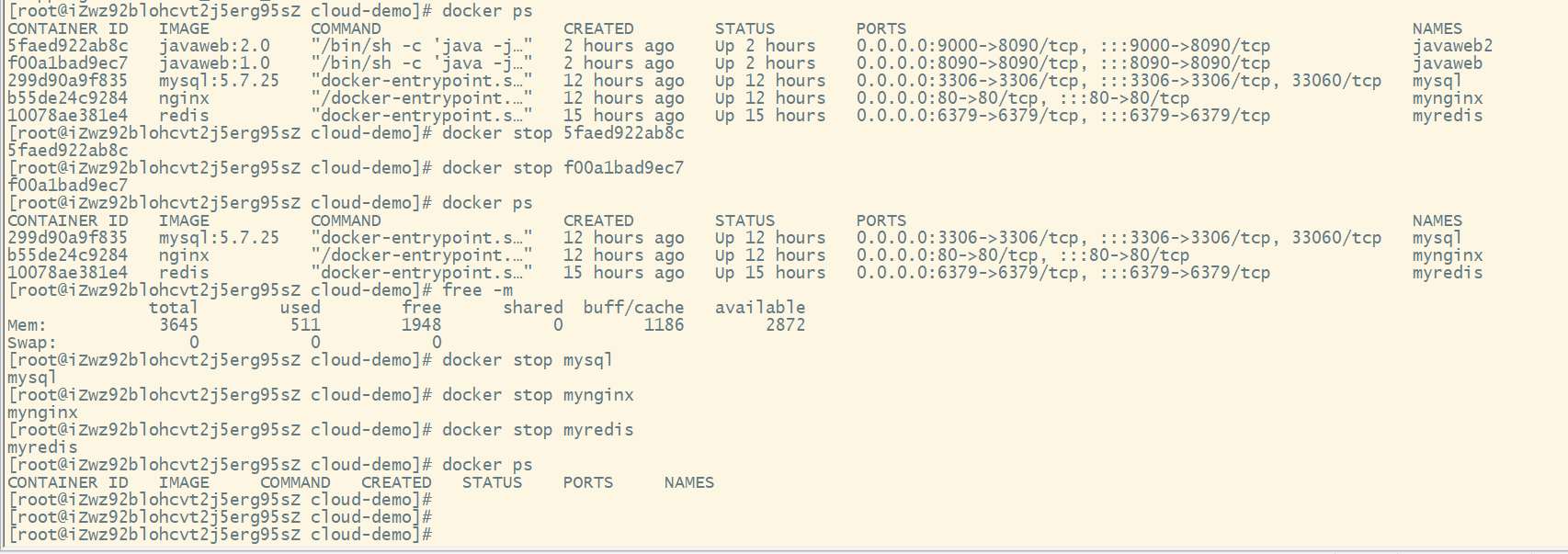
**基于Compose文件实现集群的快速构建和部署**，帮助我们快速部署分布式应用，无需一个个微服务去构建镜像和部署。

## 利用DockerCompose去部署微服务集群：

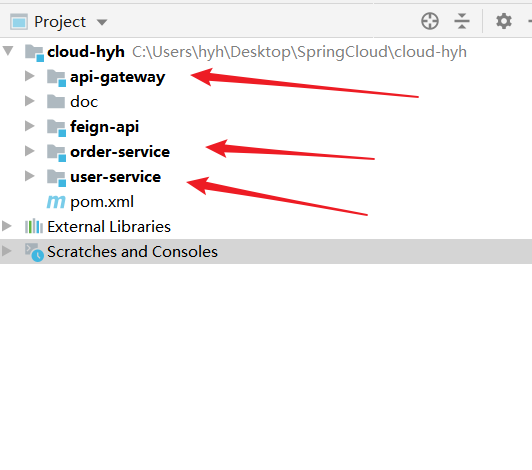
上面学习了什么是DockerCompose，什么是Compose文件以及编写Compse文件的基本语法！先择利用DockerCompse完成微服务集群的

部署！

部署前先将docker容器停掉，放出一点空间！

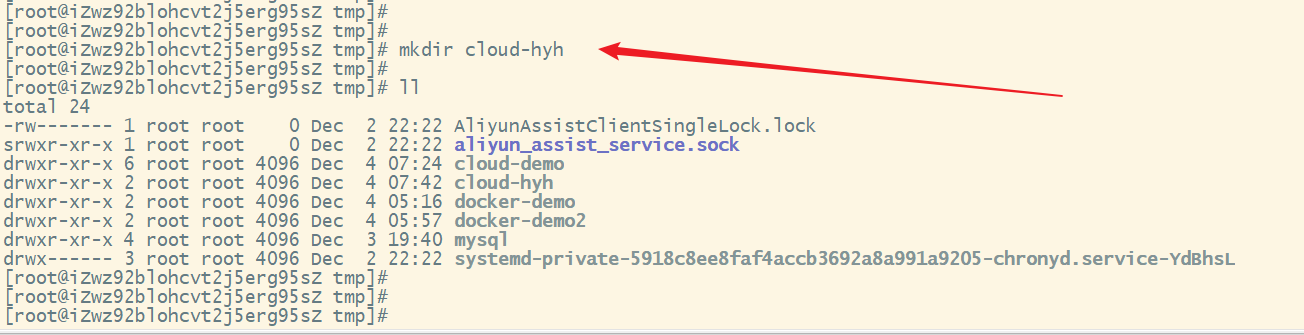


eg：将之前学习的alibaba-micro微服务集群利用DcokerCompose部署：

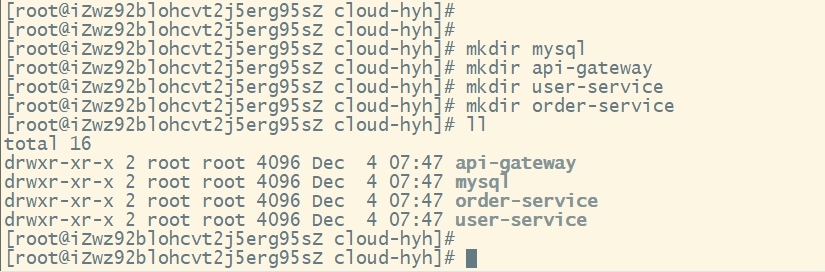


实现思路如下：

### 1：在/tmp目录下创建cloud-hyh文件夹：mkdir cloud-hyh

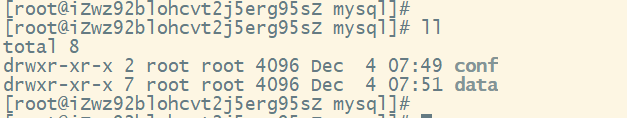


### 2：在cloud-hyh目录下，分别创建mysql、user-service、order-service、api-gateway目录



### 3：将准备好的mysql数据：conf、data拷贝到/tem/cloud-hyh/mysql目录下

Data不是空的，我们已经把项目数据库放进去了，将来只需要把Mysql容器挂载到这两个目录上，那么数据、配置就都有了！



### 4：在本地创建docker-compose.yml文件，并编写指令

可以看到：

第一个是Nacos服务，因为所有的服务都需要注册到Nacos注册中心，镜像就是nacos/nacos-server，没有会去docker-hub下载，环境变量配置成单机，端口暴露成8848；

第二个配的是mysql，配了密码1234，然后是数据卷挂载，mysql容器的conf、data目录挂载到宿主机文件系统的/tmp/cloud-hyh/conf以及/tmp/cloud-hyh/data，$PWD就是获取当前路径，因为后续都会在/tem/cloud-hyh操作，所以$PWD = /tem/cloud-hyh，mysql端口并没有对外部暴露，因为mysql仅仅对内访问，所以不需要暴露

接下来是构建并运行userservice orderservice gateway这三个镜像的命令，都是基于各自目录下的Dockerfile去构建，所以是build；这里只有网关暴露了端口，userservice容器、orderservice容器都没有暴露端口，一样的道理，微服务端口不应该暴露在外界，网关才是整个微服务的入口！

将来这个Docker-compose.yml文件一旦执行，就会其启动nacos容器、mysql容器，构建userservice、orderservice、gateway这三个镜像，运行userservice、orderservice、gateway这三个容器！这样就可以实现一次性把整个微服务集群部署好了，如果将来有更多微服务，那就接着往下写呗！



**version: "3.2"**

**services:**

**nacos:**

**image: nacos/nacos-server**

**environment:**

**MODE: standalone**

**ports:**

**- "8848:8848"**

**mysql:**

**image: mysql:5.7.25**

**environment:**

**MYSQL\_ROOT\_PASSWORD: 1234**

**volumes:**

**- "$PWD/mysql/data:/var/lib/mysql"**

**- "$PWD/mysql/conf:/etc/mysql/conf.d/"**

**userservice:**

**build: ./user-service**

**orderservice:**

**build: ./order-service**

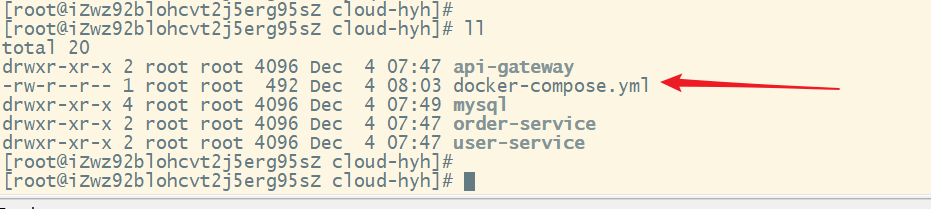
**gateway:**

**build: ./api-gateway**

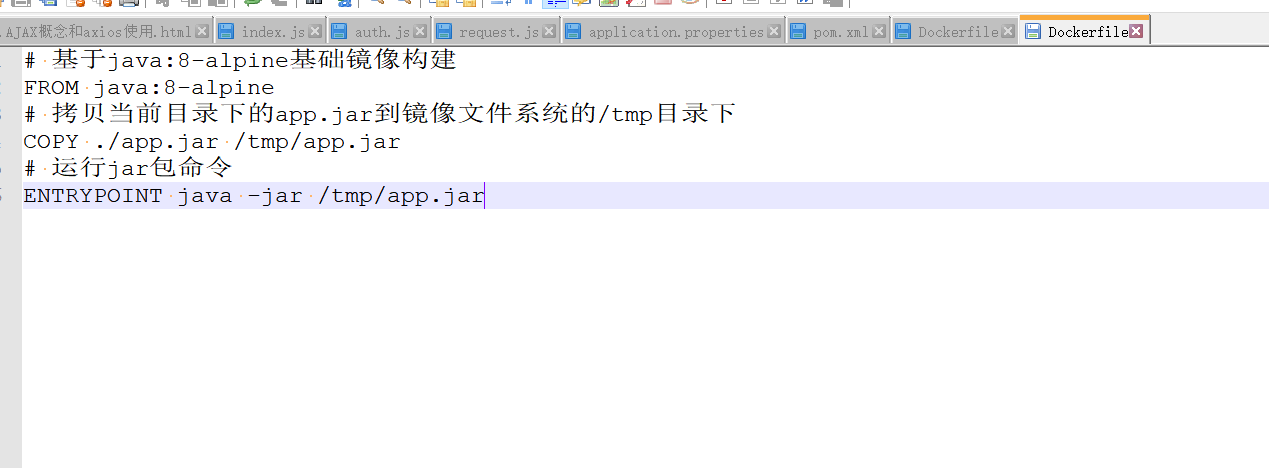
**ports:**

**- "8001:8001"**

### 5：将编写好的docker-compose.Yml文件上传到在cloud-hyh目录下：

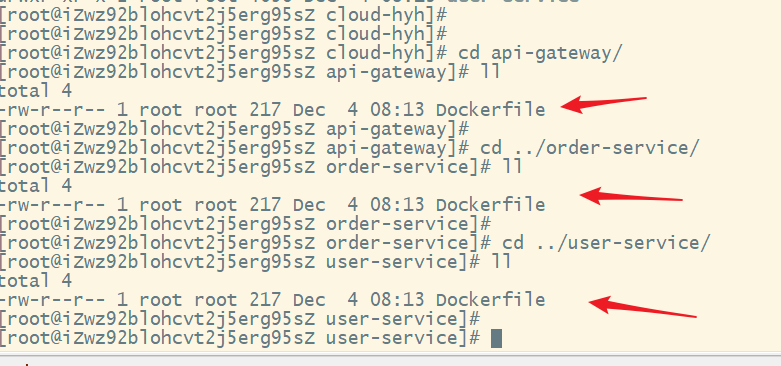


### 6：在本地编写构建api-gateway镜像、user-service镜像、order-service镜像的Dockerfile文件



Dockerfile文件是一样的，只需要将jar包上传到对应目录，就可以通用！

### 7：将dockerfile文件分贝上传到api-gateway、user-service、order-service目录下

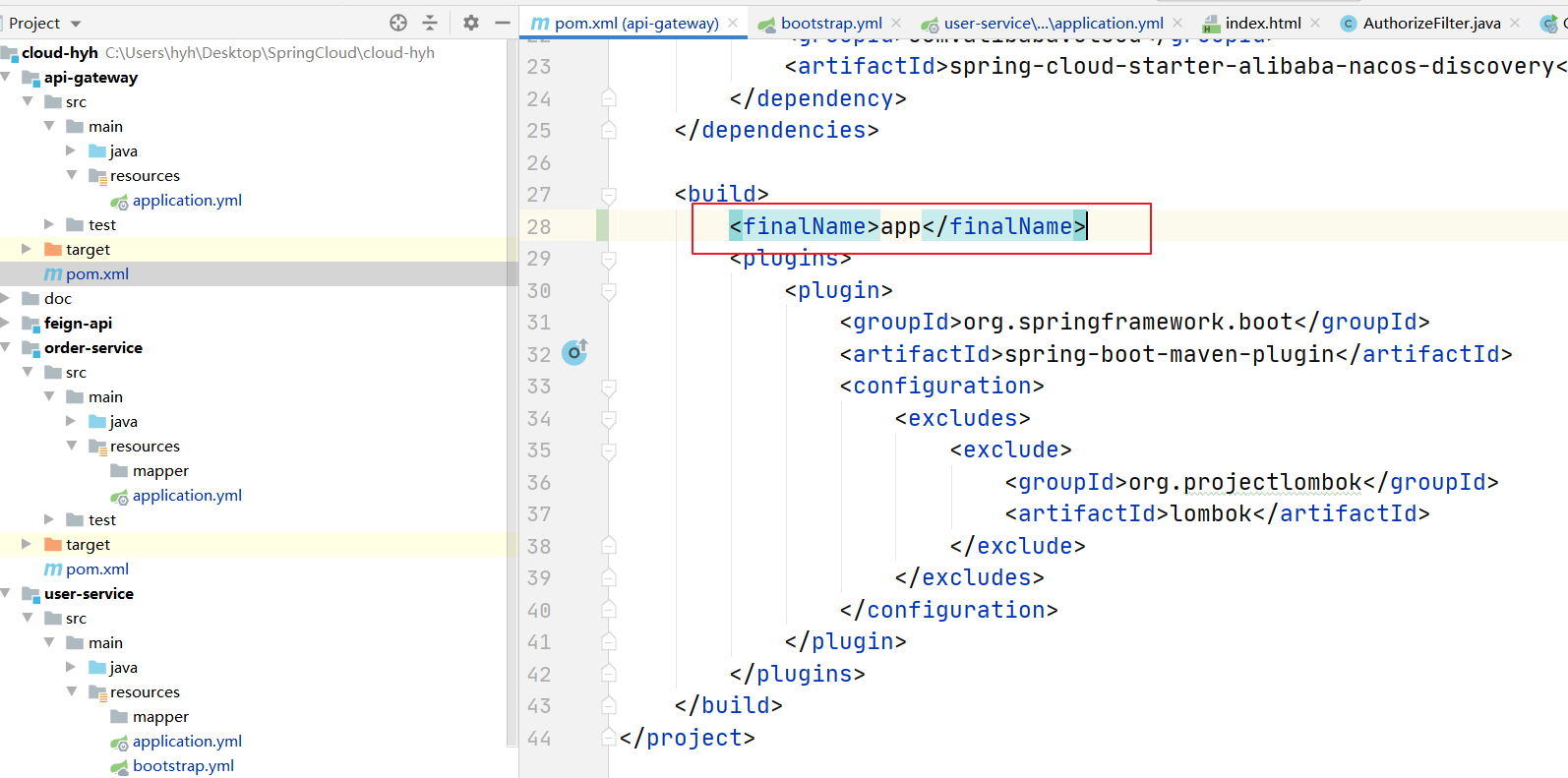


### 8：在idea打包cloud-hyh项目：

要先修改cloud-hyh项目，将数据库、nacos地址都命名为docker-compose中的服务名！为什么呢？

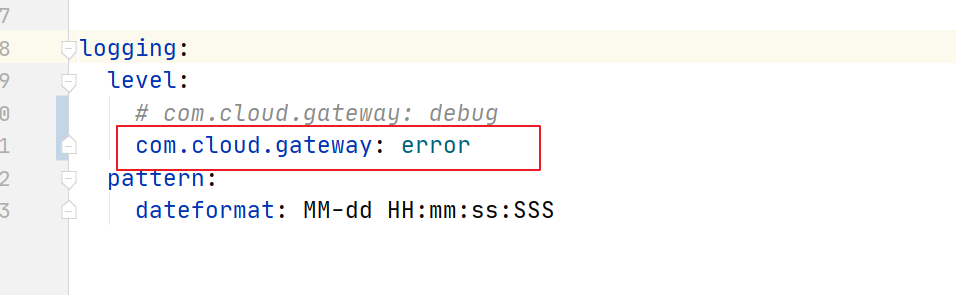
以前的地址写的都是ip地址，现在是集群部署，但现在用DockerCompose部署时，所有的服务之间都可以用服务名互相访问，也就是说比如userservice要想访问mysql，就用mysql的服务名mysql访问就行，要想访问nacos，就用nacos的服务名nacos就行。这是Compose底层做了是实现！

#### 8.1：修改api-gateway的pom.xml

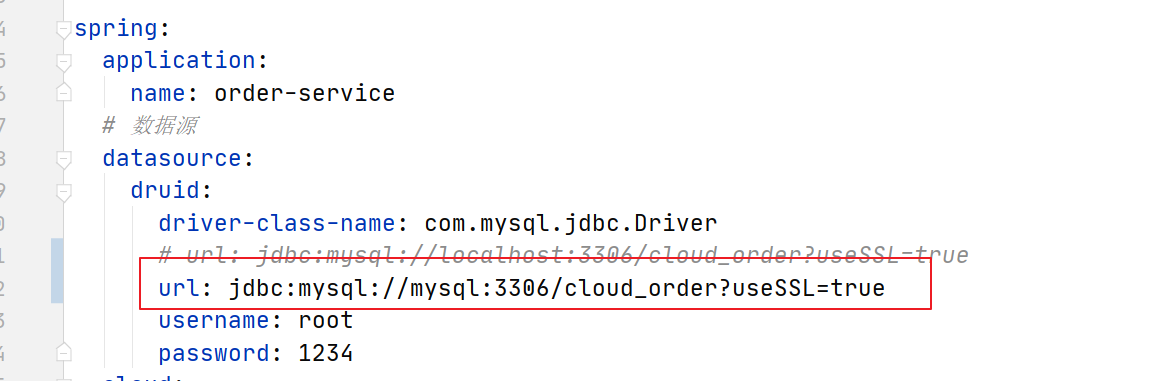


#### 8.2：修改api-gateway的application.yml配置文件，修改nacos地址和mysql地址和日志级别

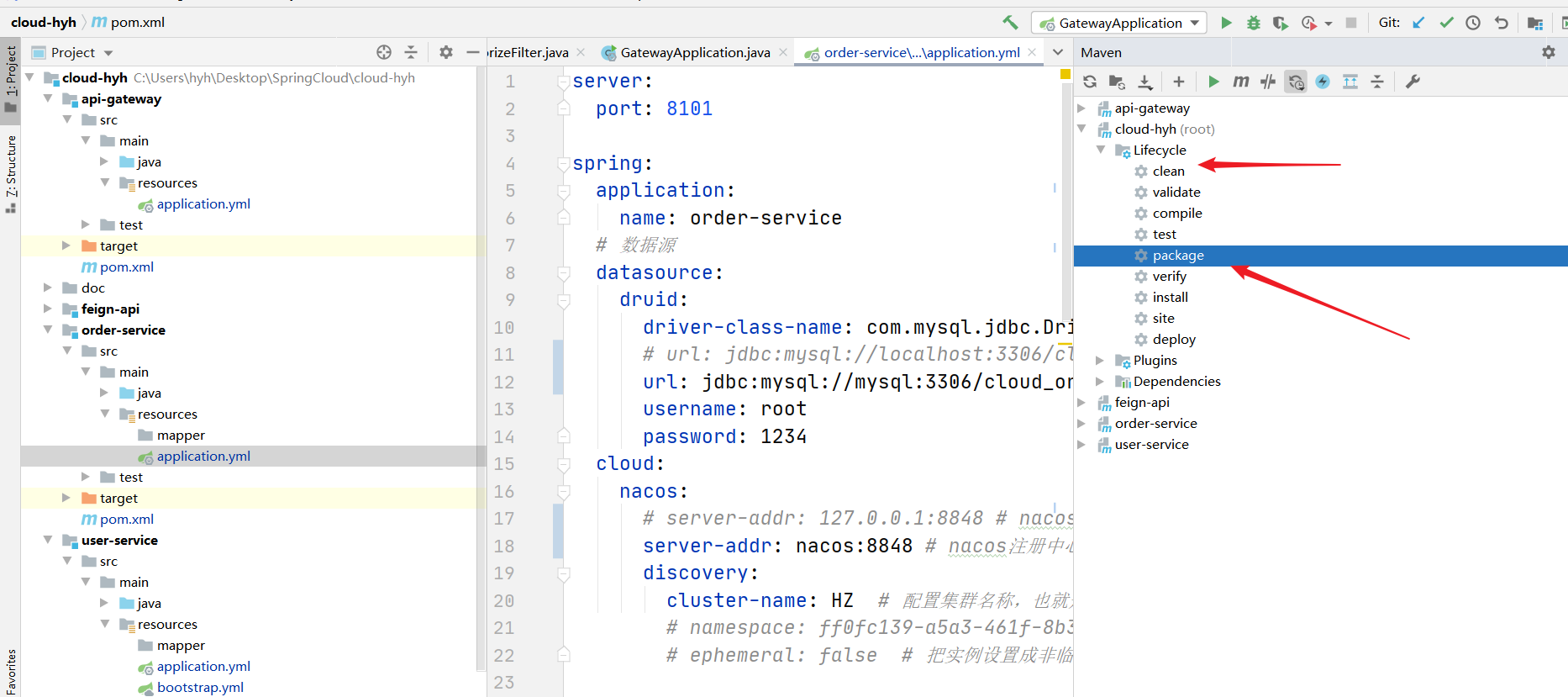




#### 8.3：相同地去修改user-service模块、order-service模块



#### 8.4：使用maven打包工具打包：cloud-hyh先clear，再package，打包



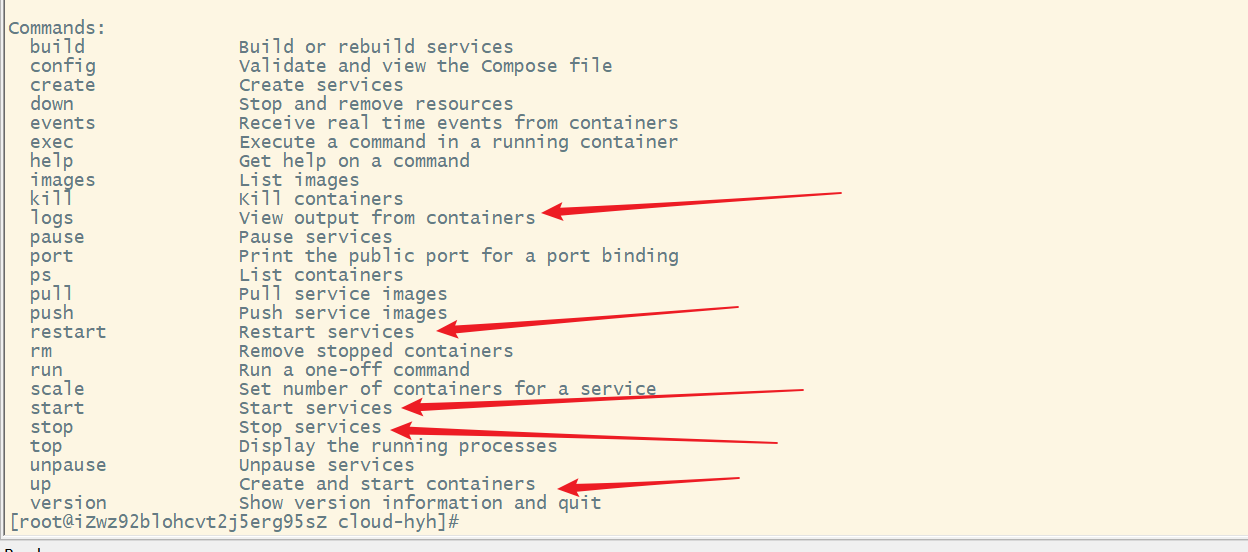
#### 8.5：将app..jar包上传到对应目录



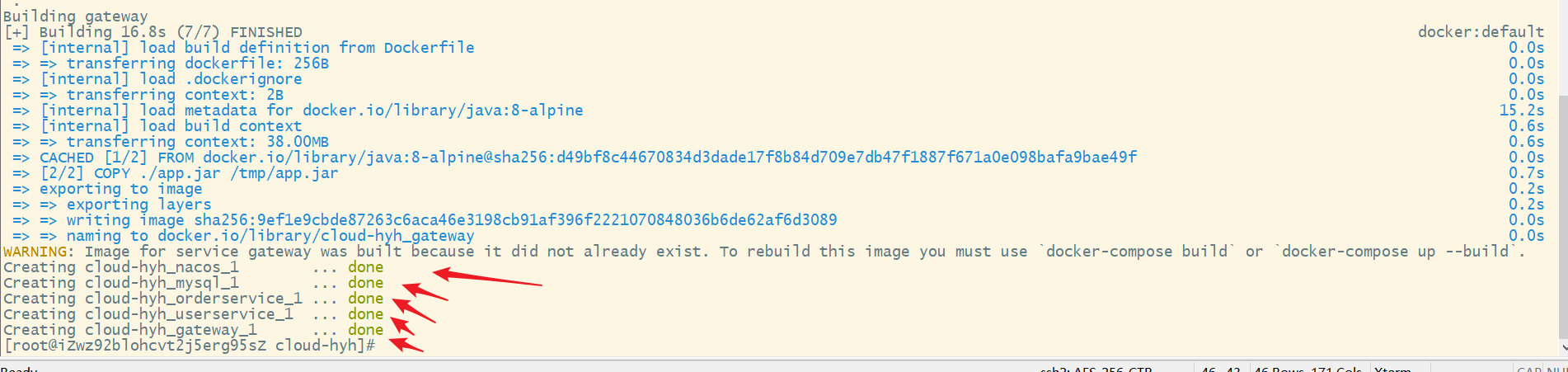
**api-gateway、user-service、order-service各自的Dockerfile准备好了，api-gateway、user-service、order-servicejar各自的jar包准备好了，那么将来DockerCompose就会帮助我们自动构建三个微服务的镜像并运行镜像！**

### 9：在/tmp/cloud-hyh目录下，利用docker-compose up -d来部署

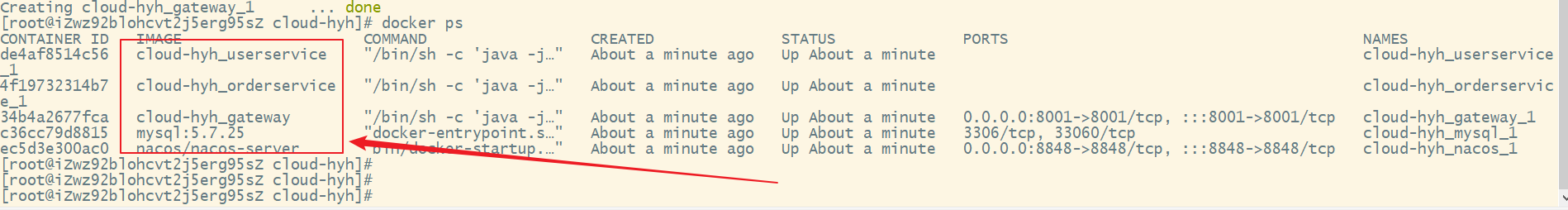
常用的docker-compose命令：docker-compose xxx



#### docker-compose up -d，全部构建完成，



#### docker ps瞧一瞧，看一看：

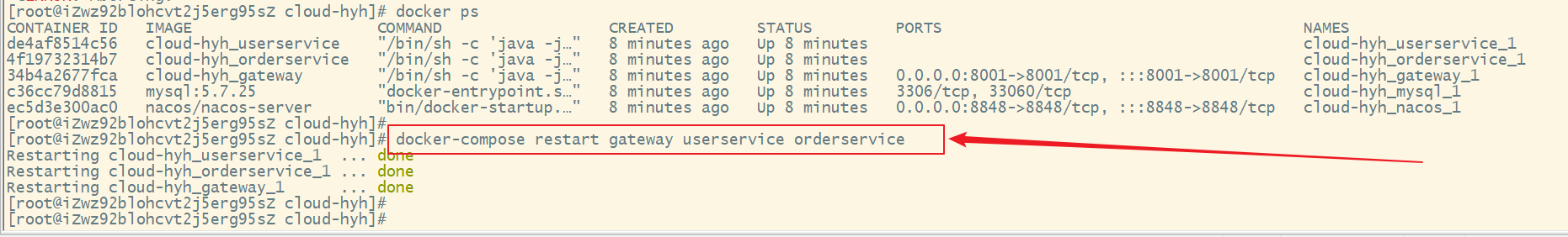


#### docker-compose logs -f，查看日志，居然有报错

因为naco启动太慢了，userservice orderservice gateway服务启动完了，nacos还没有启动成功，那不可就会报错吗！

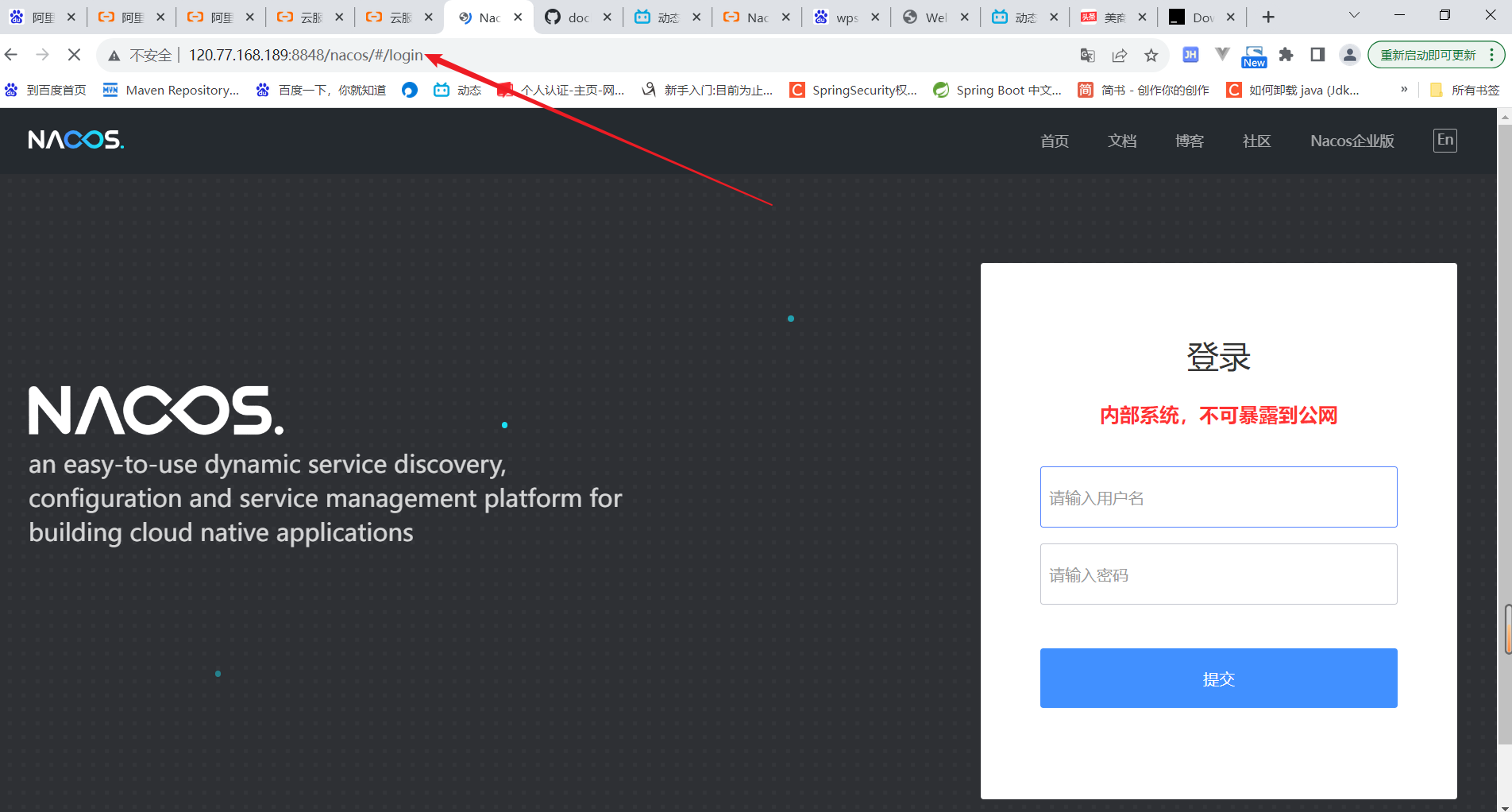


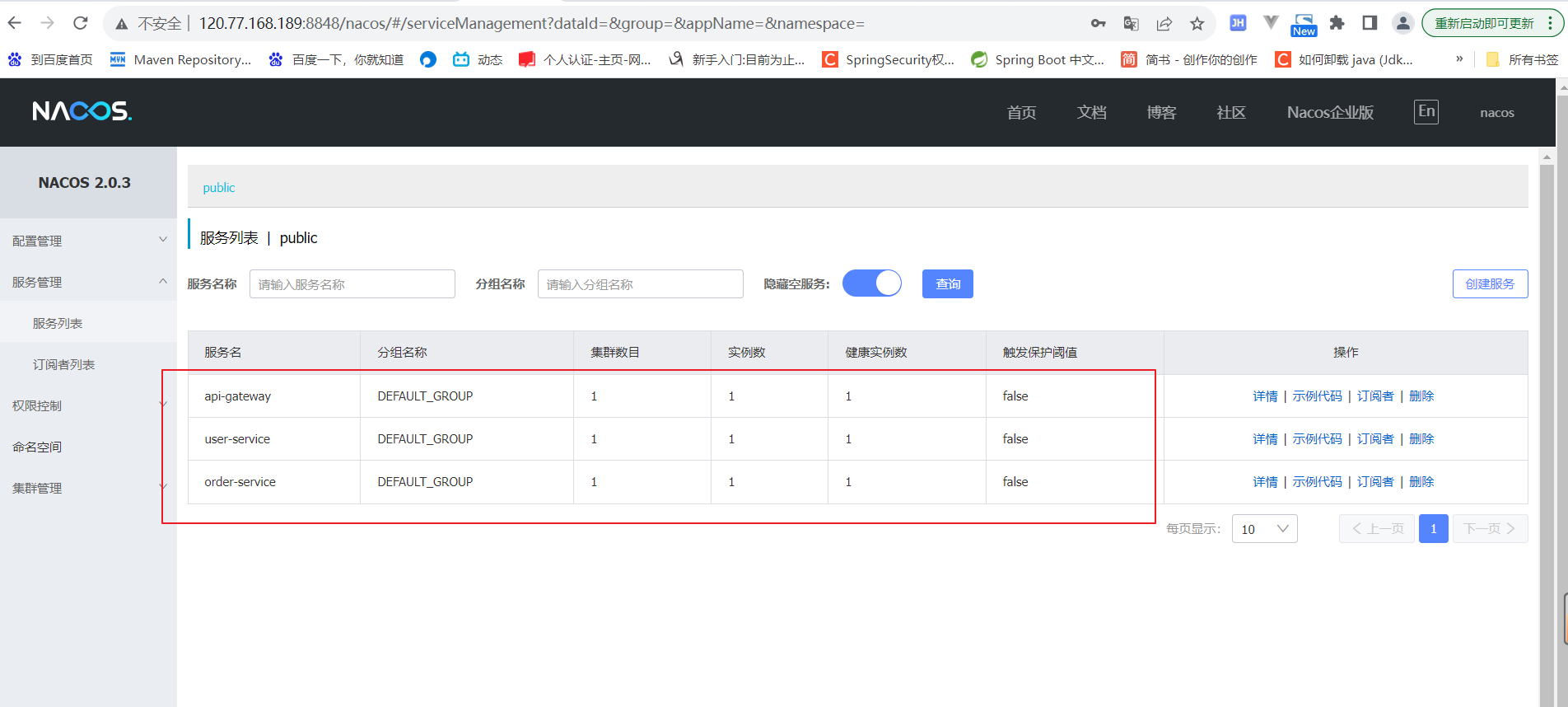
#### 其实在生产部署的时候，nacos通常要提前部署，而后再去部署微服务，这里简单一点，这里直接把userservice、orderservice、gateway这三个微服务重启一下：docker-compose restart



#### docker-compose logs -f userservice 查看具体服务的日志

#### 登录nacos看一下：





#### 浏览器看一下：

可定是部署成功了，一直访问不到数据是数据库mysql没配置好

