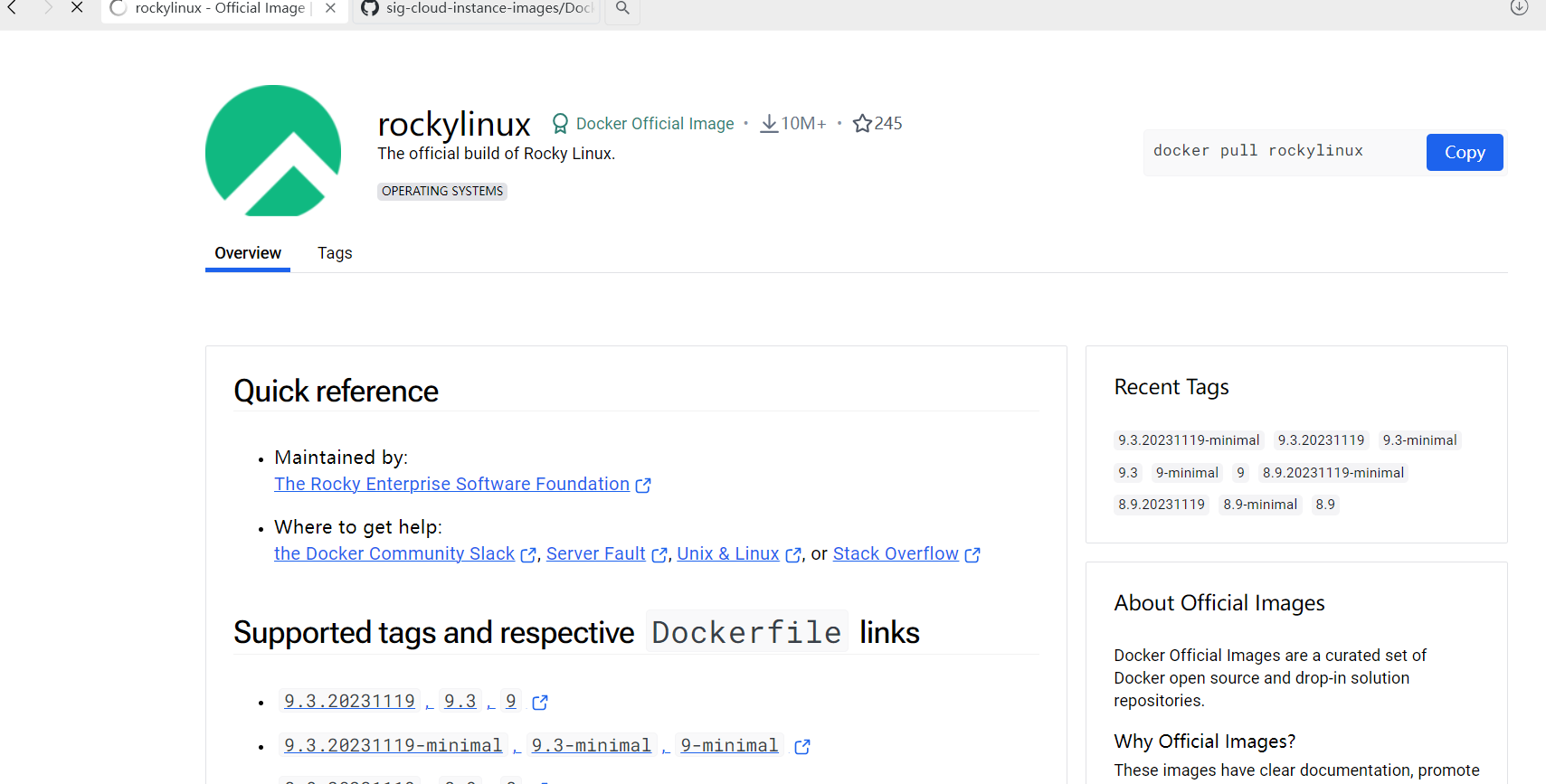
需要开两个docker机器

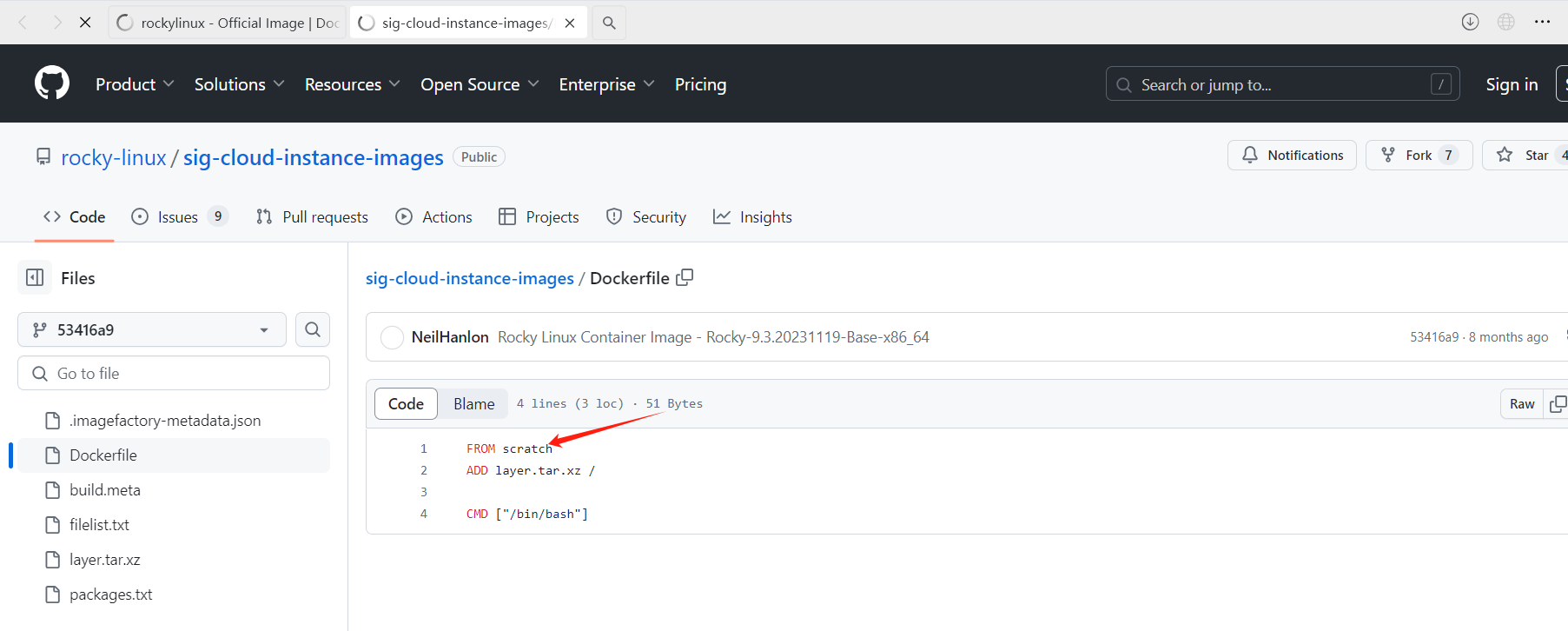
制作base镜像

不依赖以其他的镜像，从scratch构建

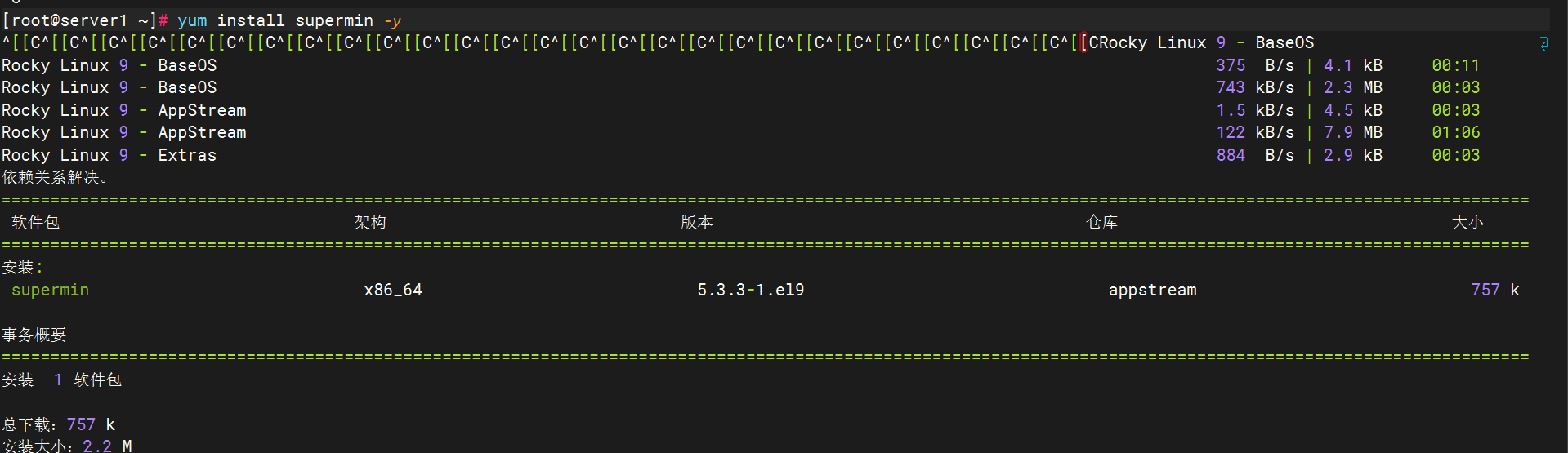
其他镜像可以为基础进行扩展，通常是系统类镜像

访问hub.docker.com网站，搜索rockylinux





安装supermin服务



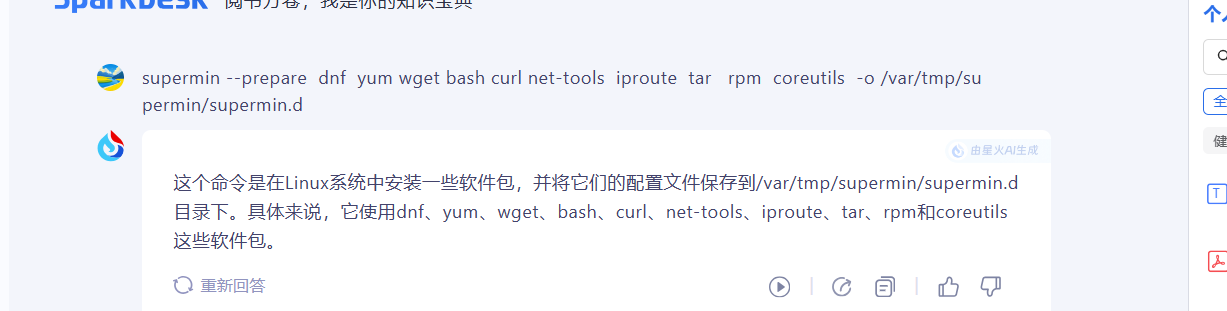
查看版本

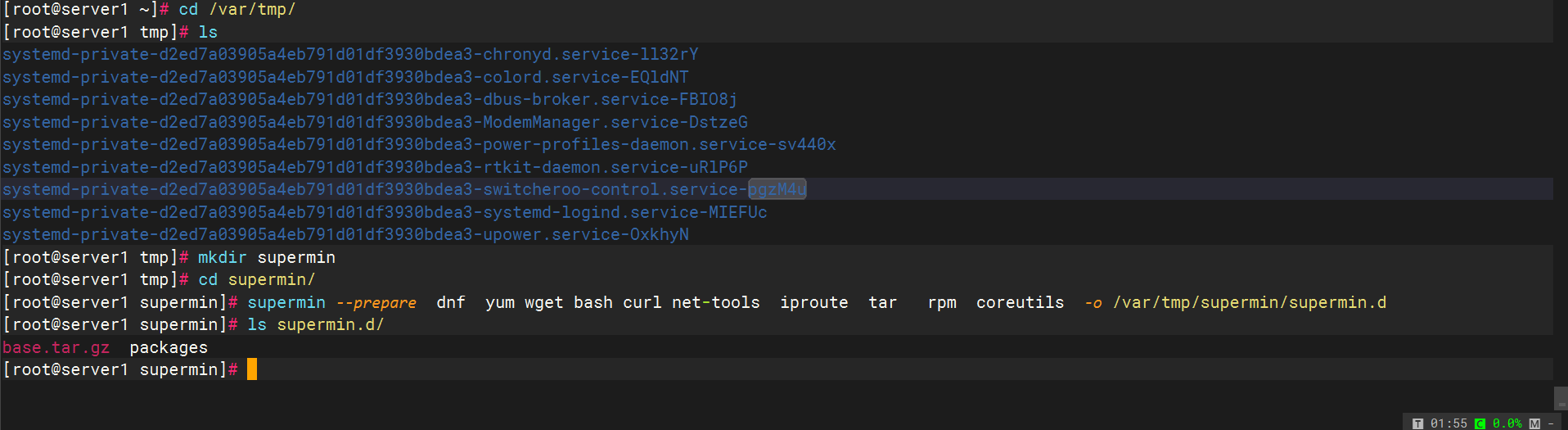


Cd /var/tmp，切换到指定目录

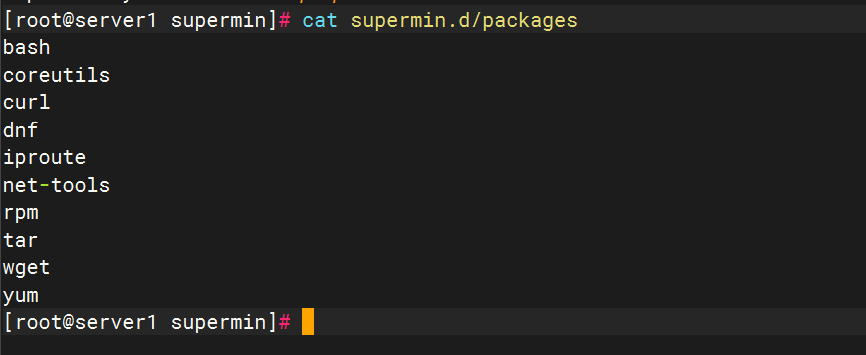
创建一个supermin的目录，

安装服务supermin --prepare dnf yum wget bash curl net-tools iproute tar rpm coreutils

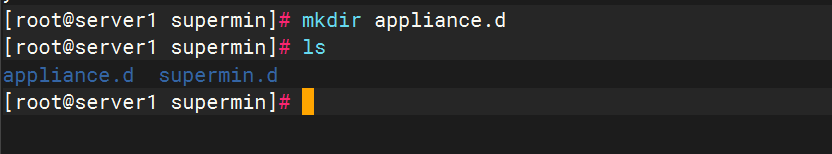




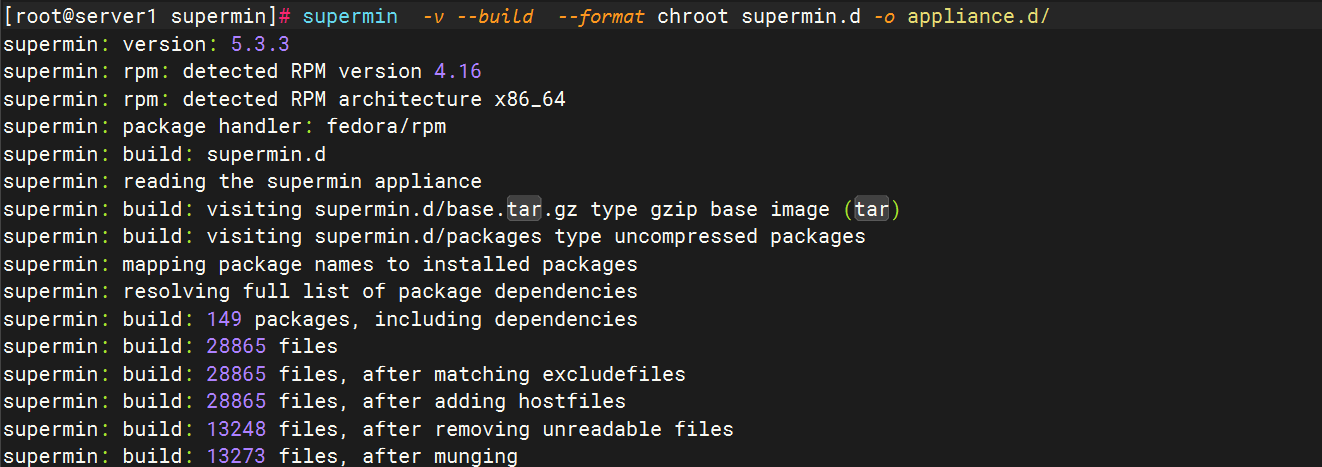
查看包文件



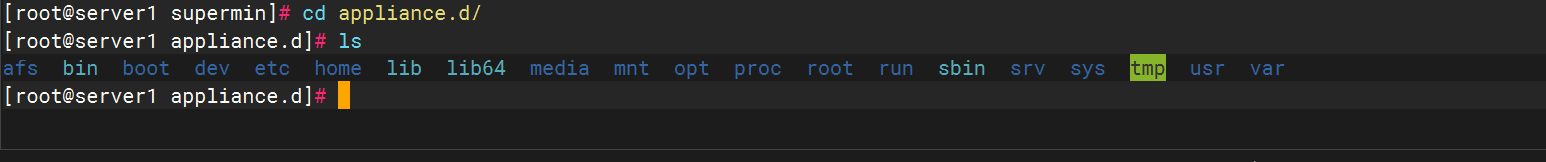
创建一个目录

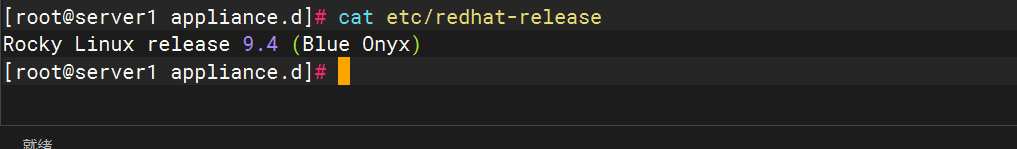


-v是显示过程，进行构建，以chroot的格式将supermin下面的东西输出到app目录下

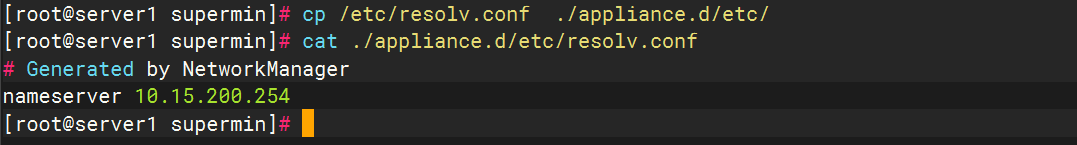


形成了一个操作系统



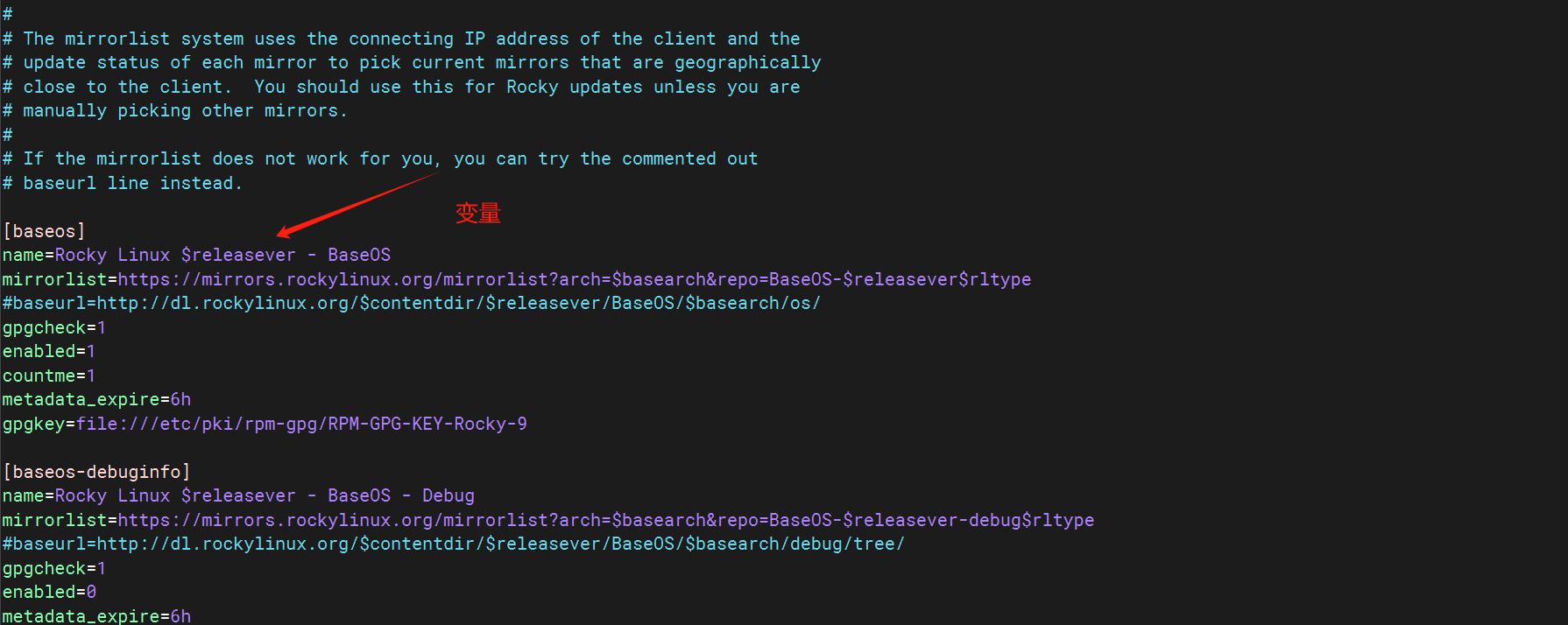


复制主机的dns到新装的系统当中

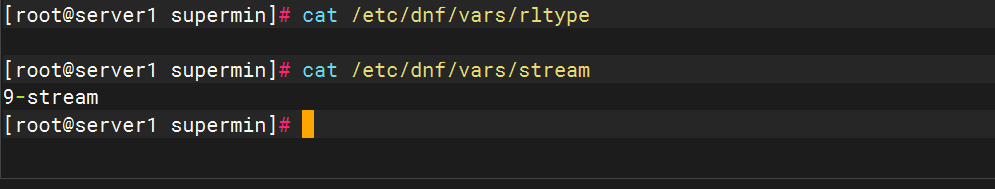


Yum仓库

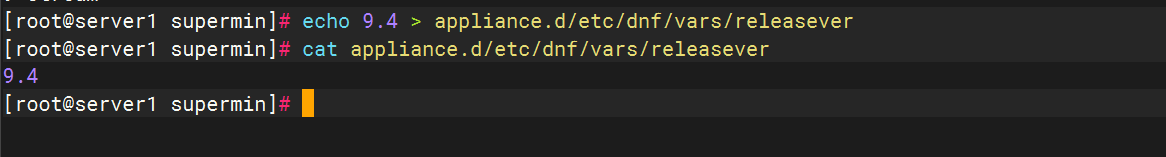




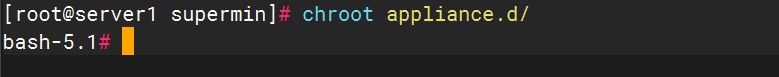
查看物理机的变量



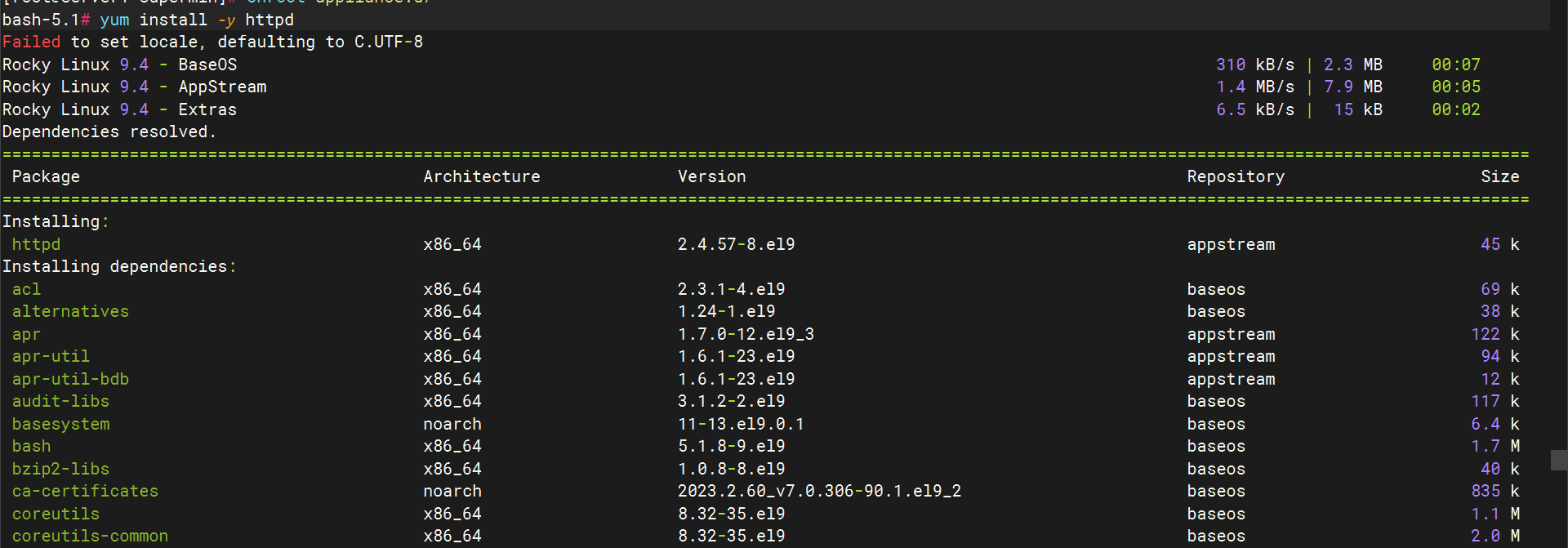
输出9.4的版本放到刚才chroot的新系统当中



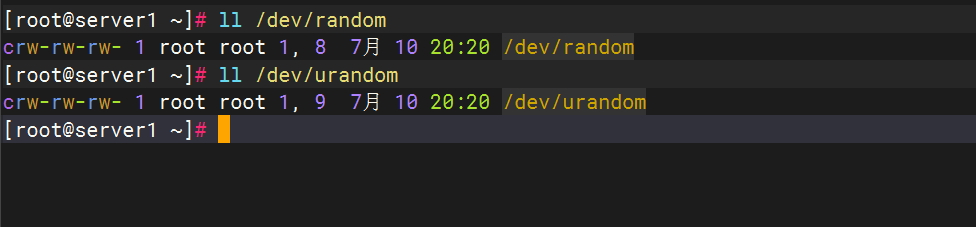
进入新装的系统



看看yum能否正常安装（如果error报错了，安装libstdc++ 软件包）



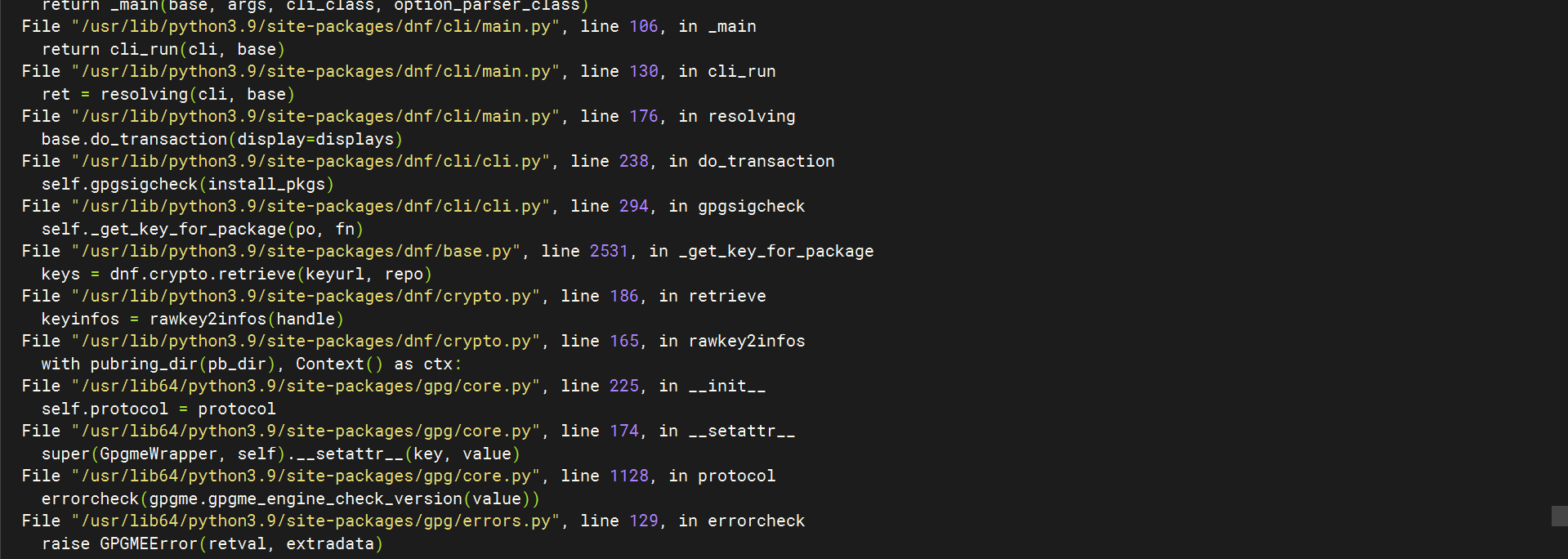
1是主设备号，8是次设备号，c是代表字符设备



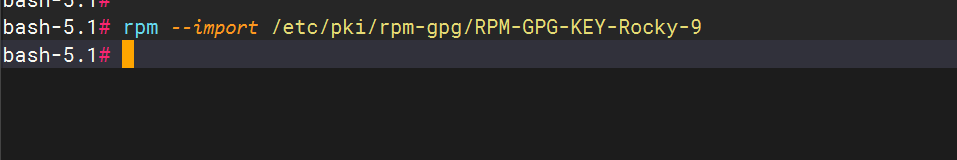
然后进入系统，创建主次设备，-m是给权限，c是代表字符设备



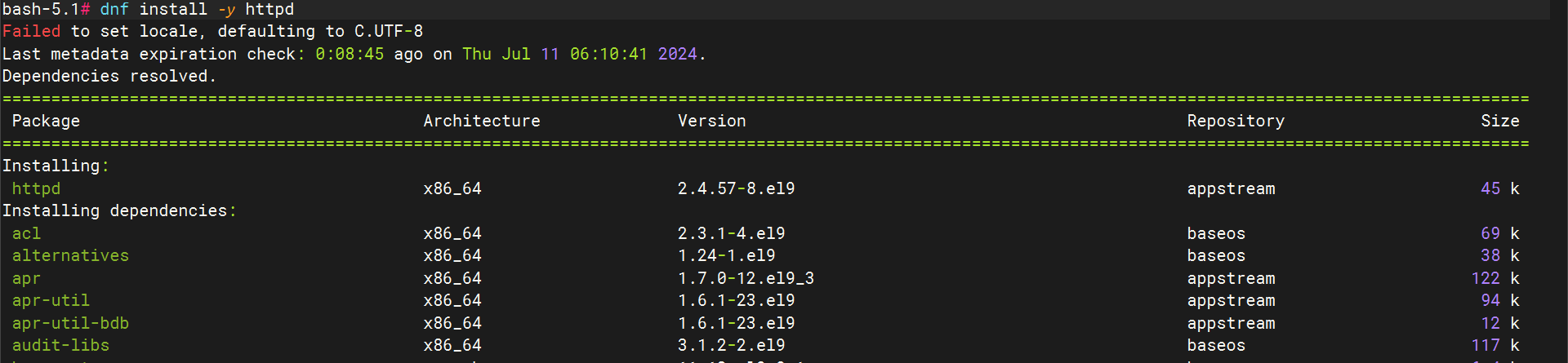
安装服务时会报错

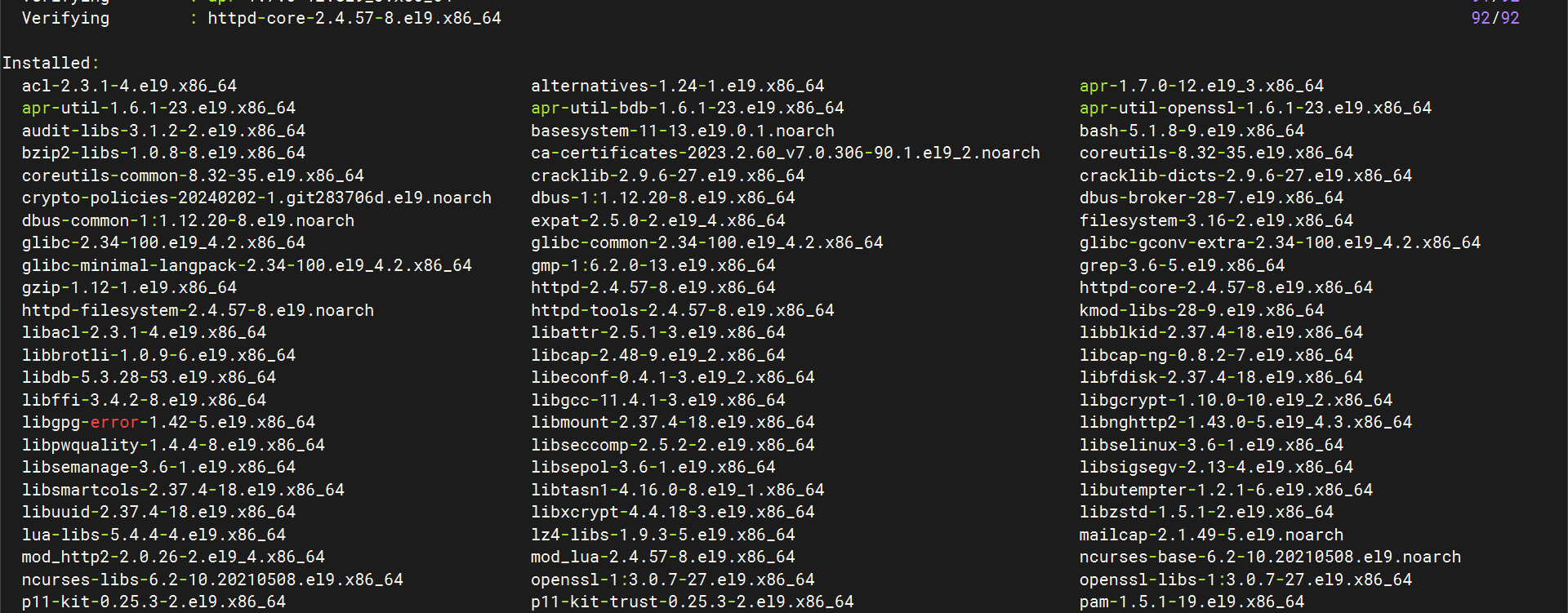


导入密钥

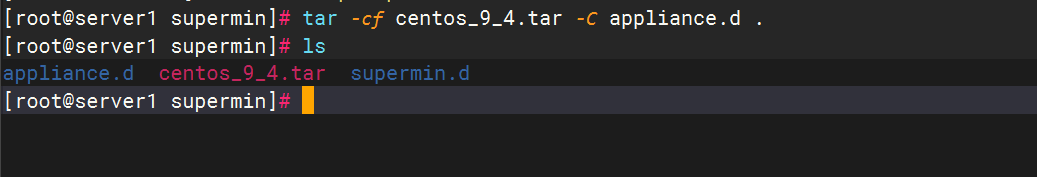


再次安装试试（成功）

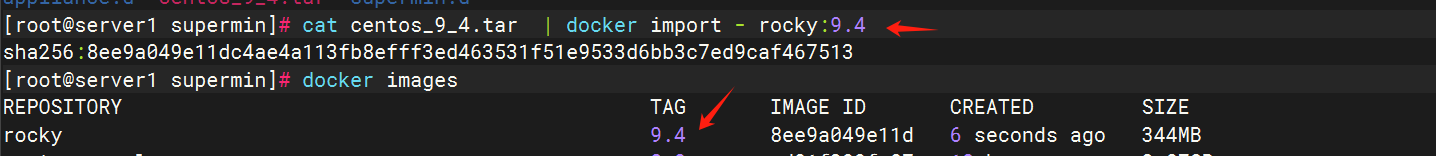




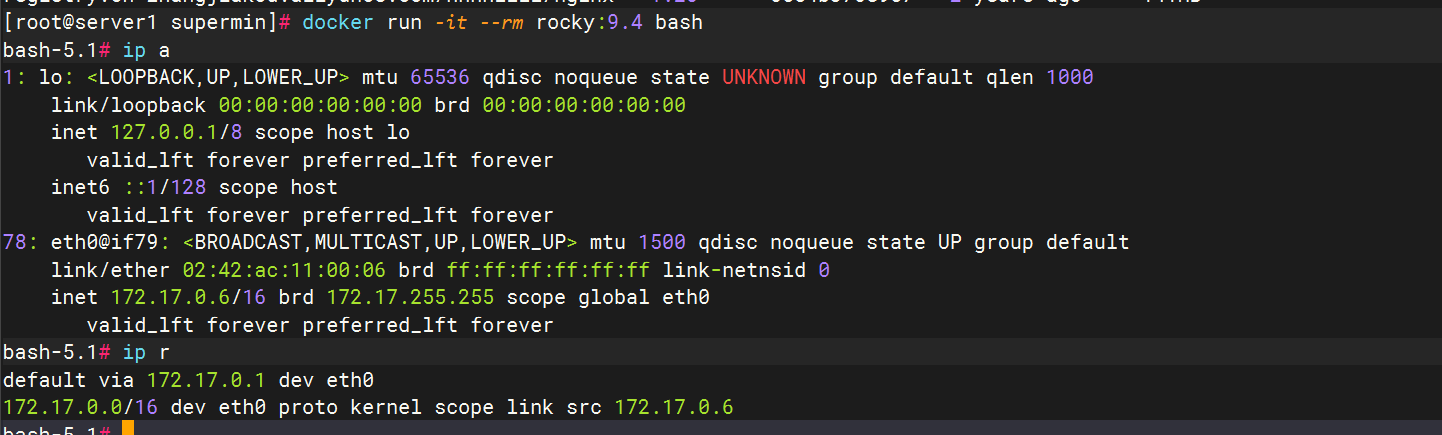
将系统做成一个tar包



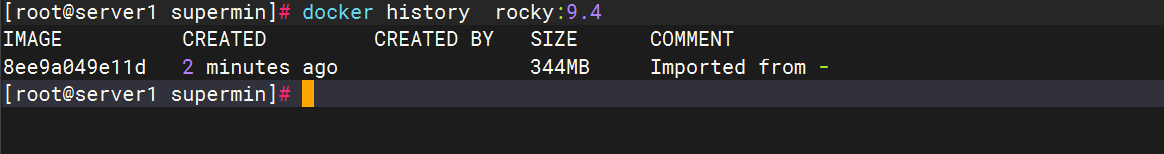
导二进制码为镜像



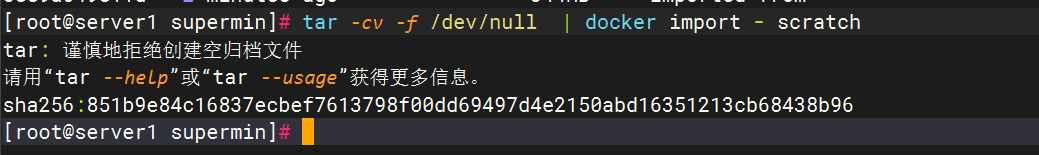
验证容器是否可以正常使用

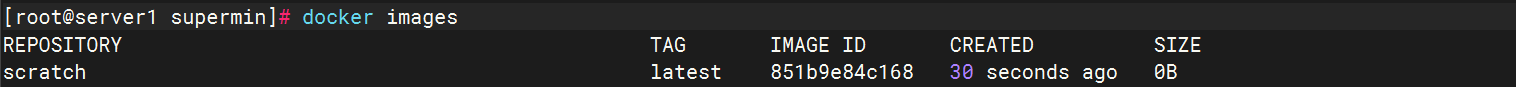


但是现在的镜像还是不合法的

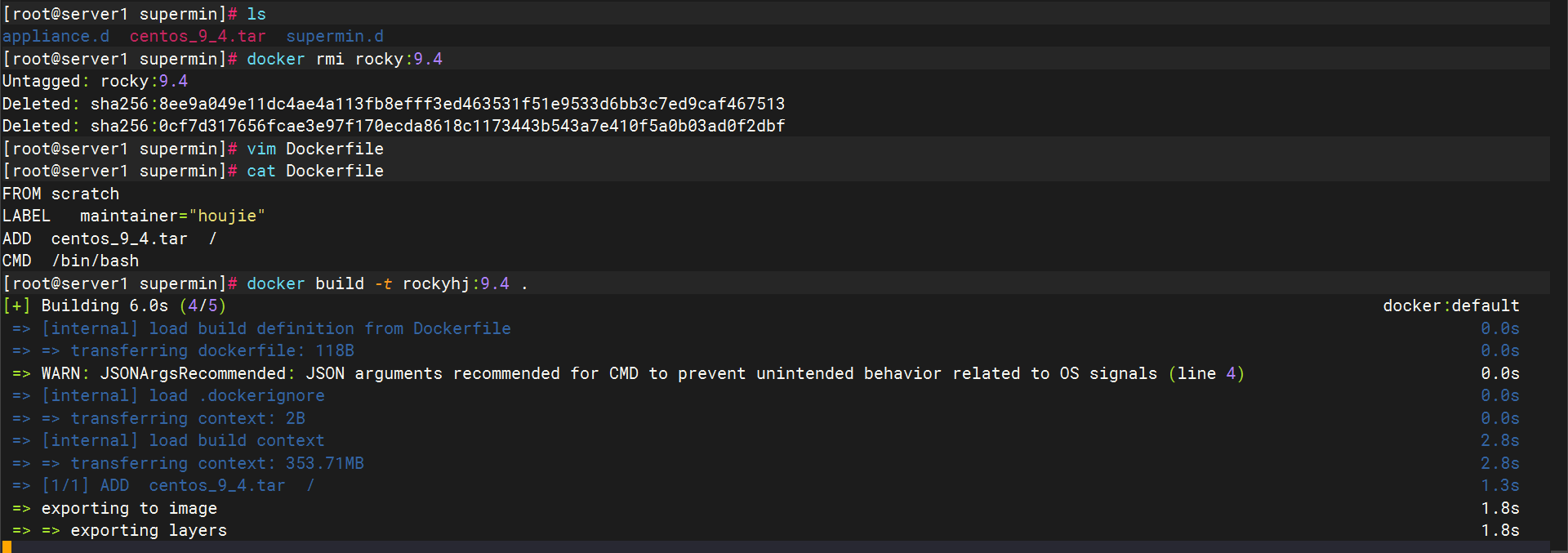


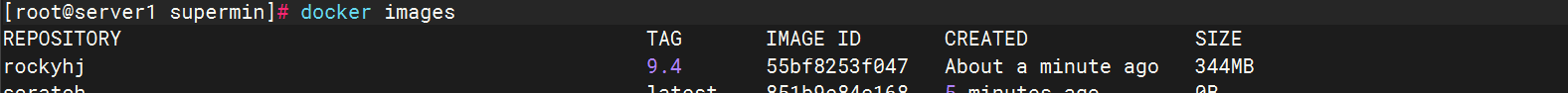
创建合法设备（null是零字节的设备）



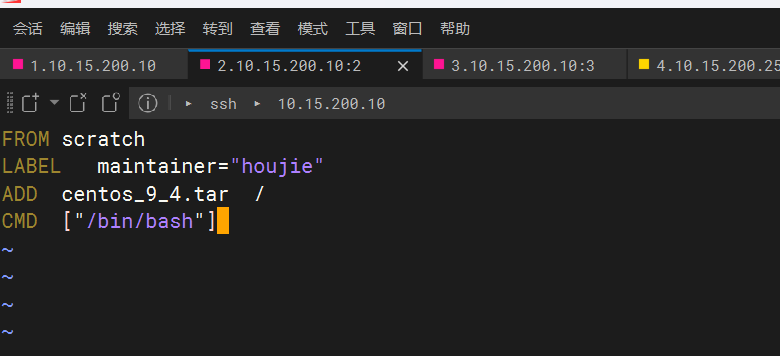


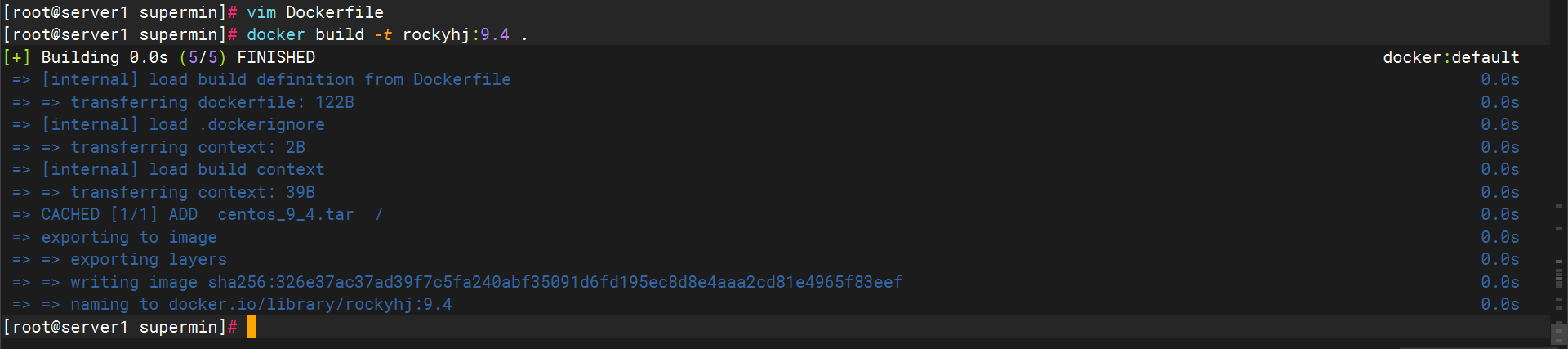
删除之前做的测试镜像，然后自己做一个镜像（label是写一个标签的意思，add将之前的tar包进行解压）



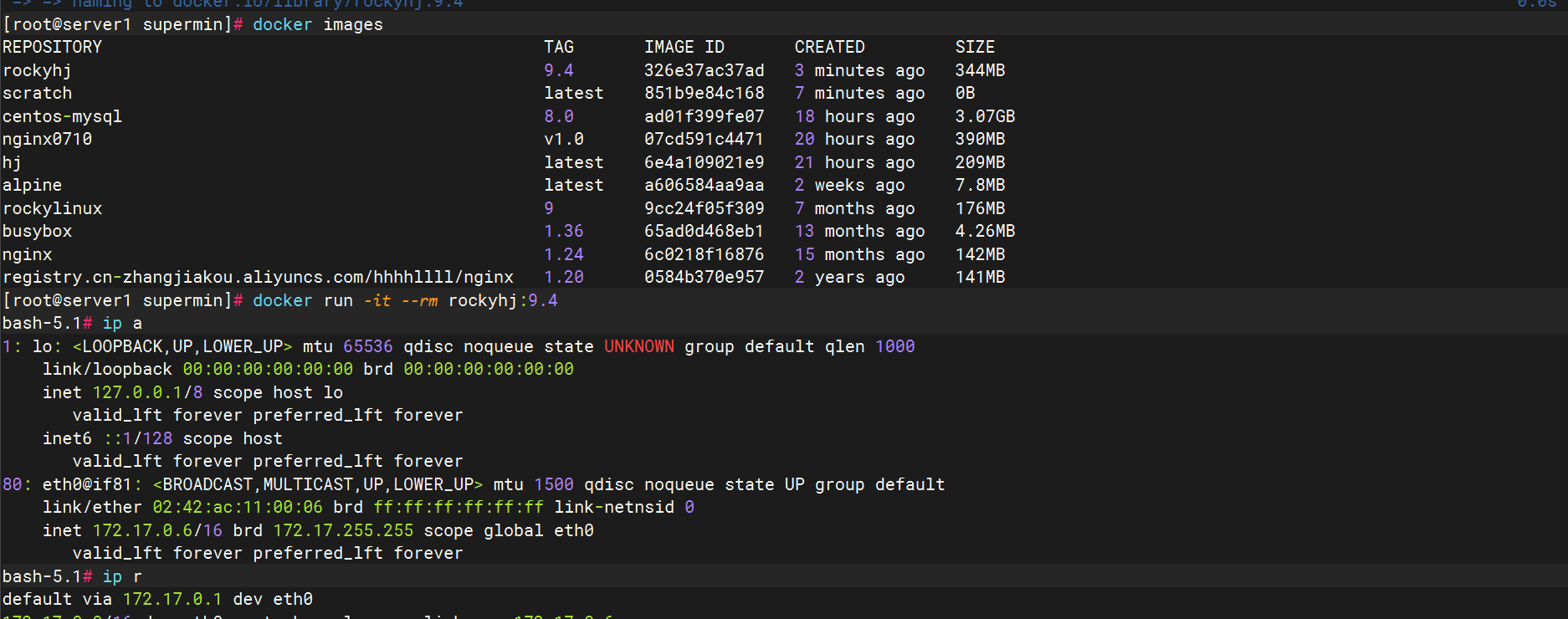


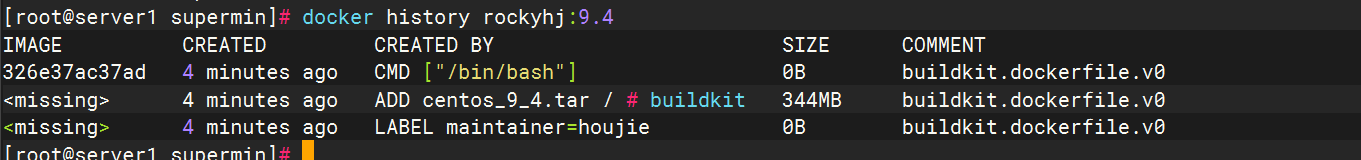
取消那个报错（这是因为josn格式，是希望执行exec格式）



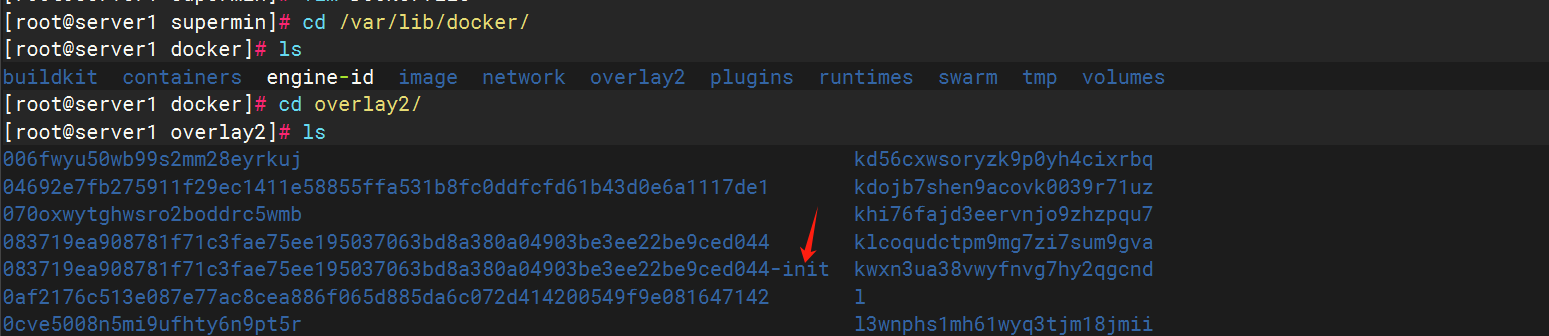


这时在创建容器后面就不需要写bash了



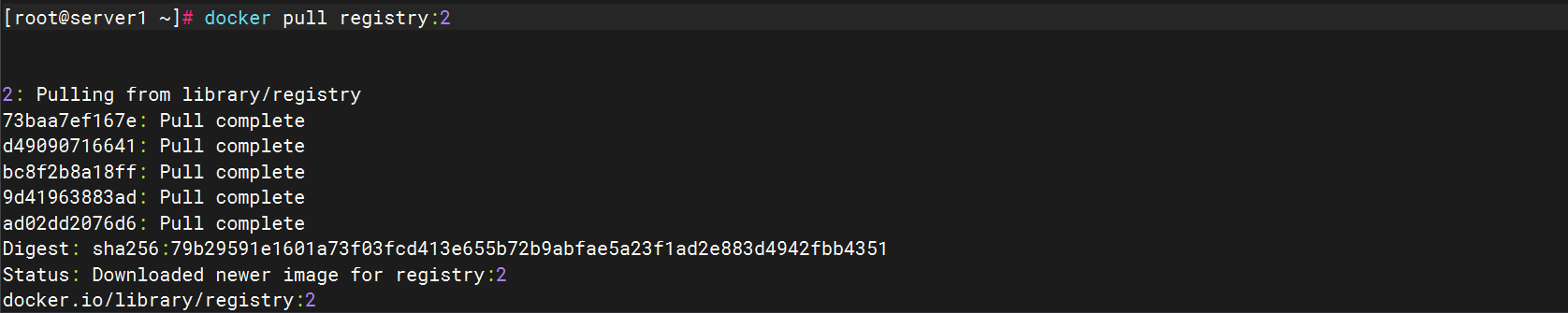


运行容器时会生成新的一个init层（引导）

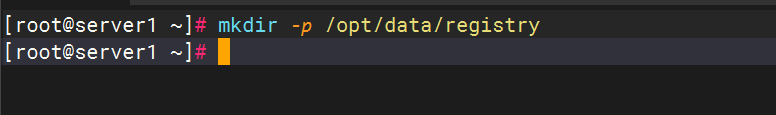


制作私有仓库（一台服务端，一台客户端）

下载go语言的镜像



创建目录

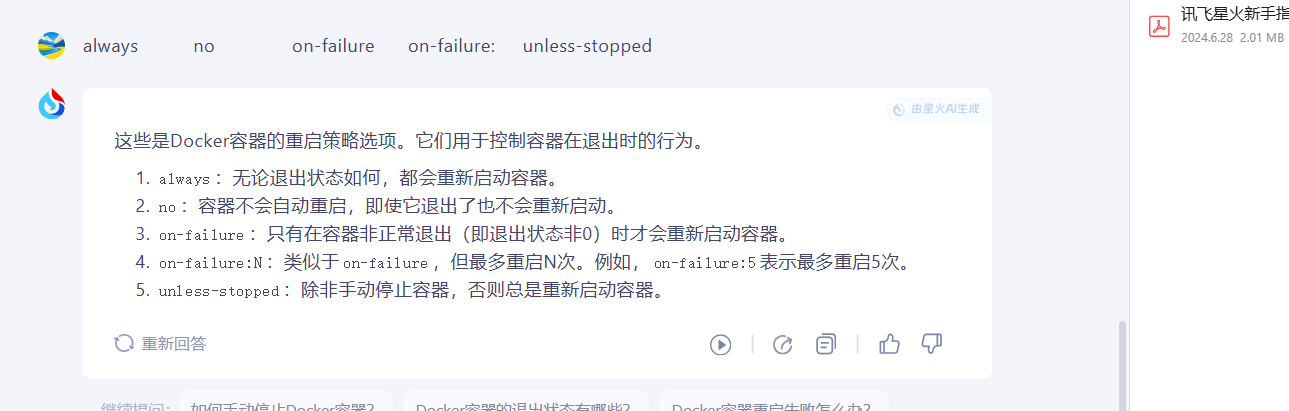


Docker run -itd -p 5000（物理机）：5000（容器） --restart（跟容器启动有关系，启动容器的进程就启动docker容器），always表示总是（非人为）

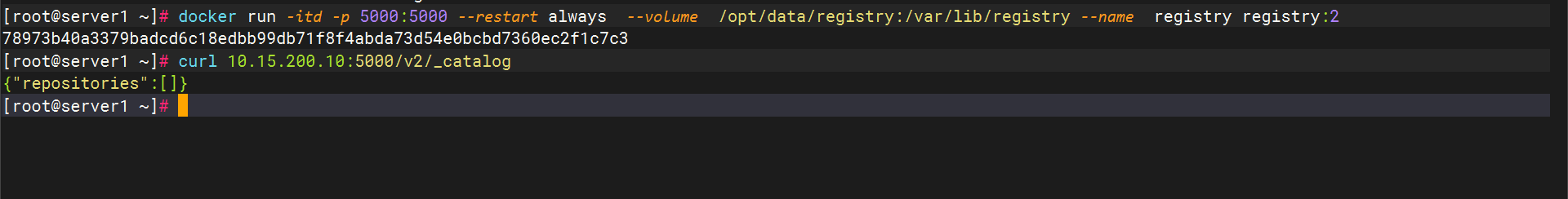




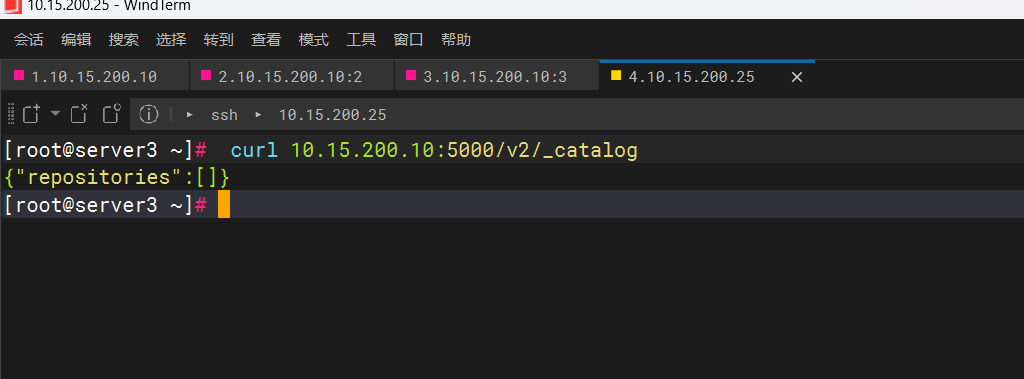




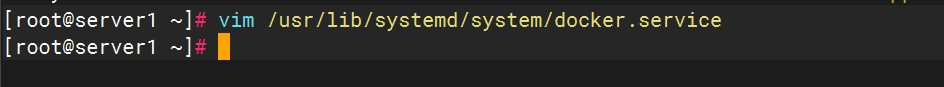


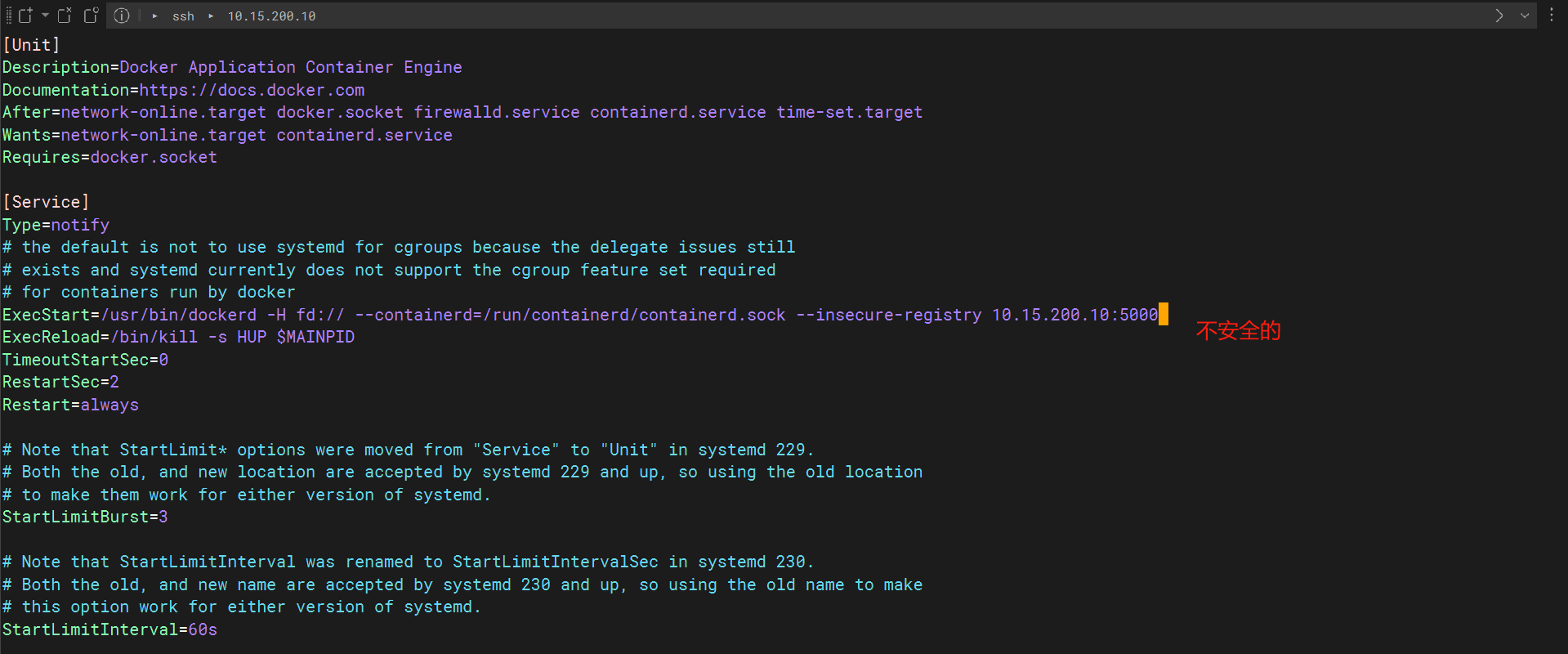


第二台也可以

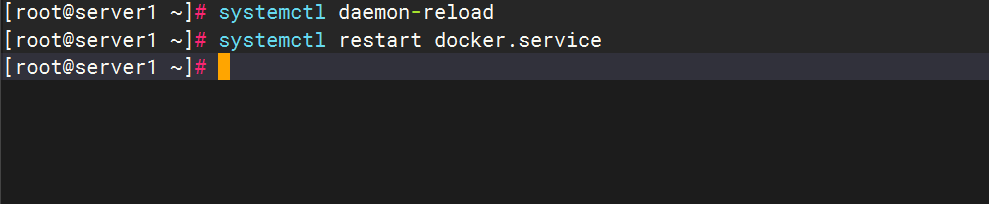


修改docker的启动文件（不安全的仓库）



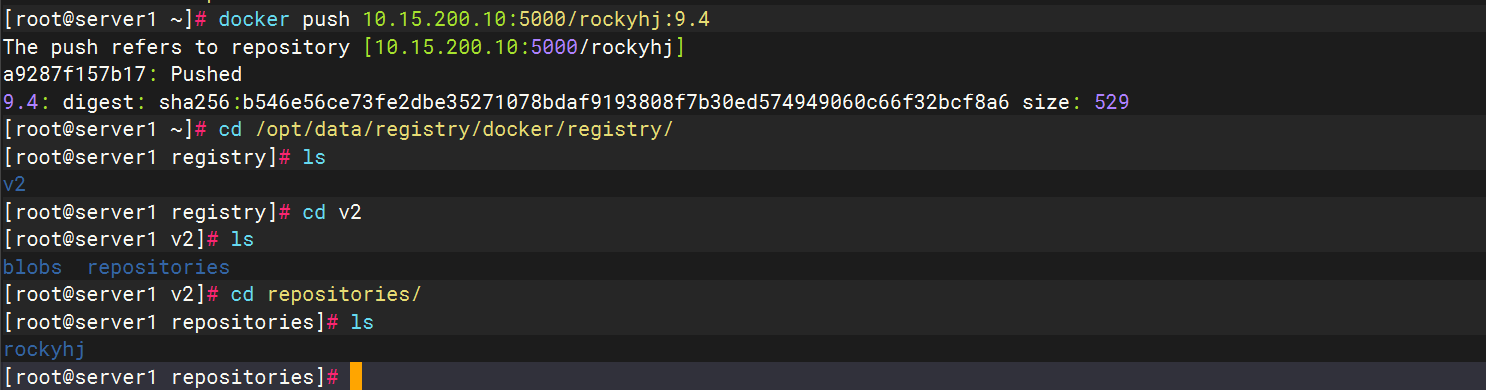


重启服务

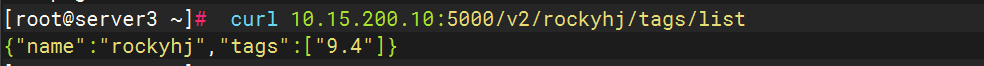


修改标签

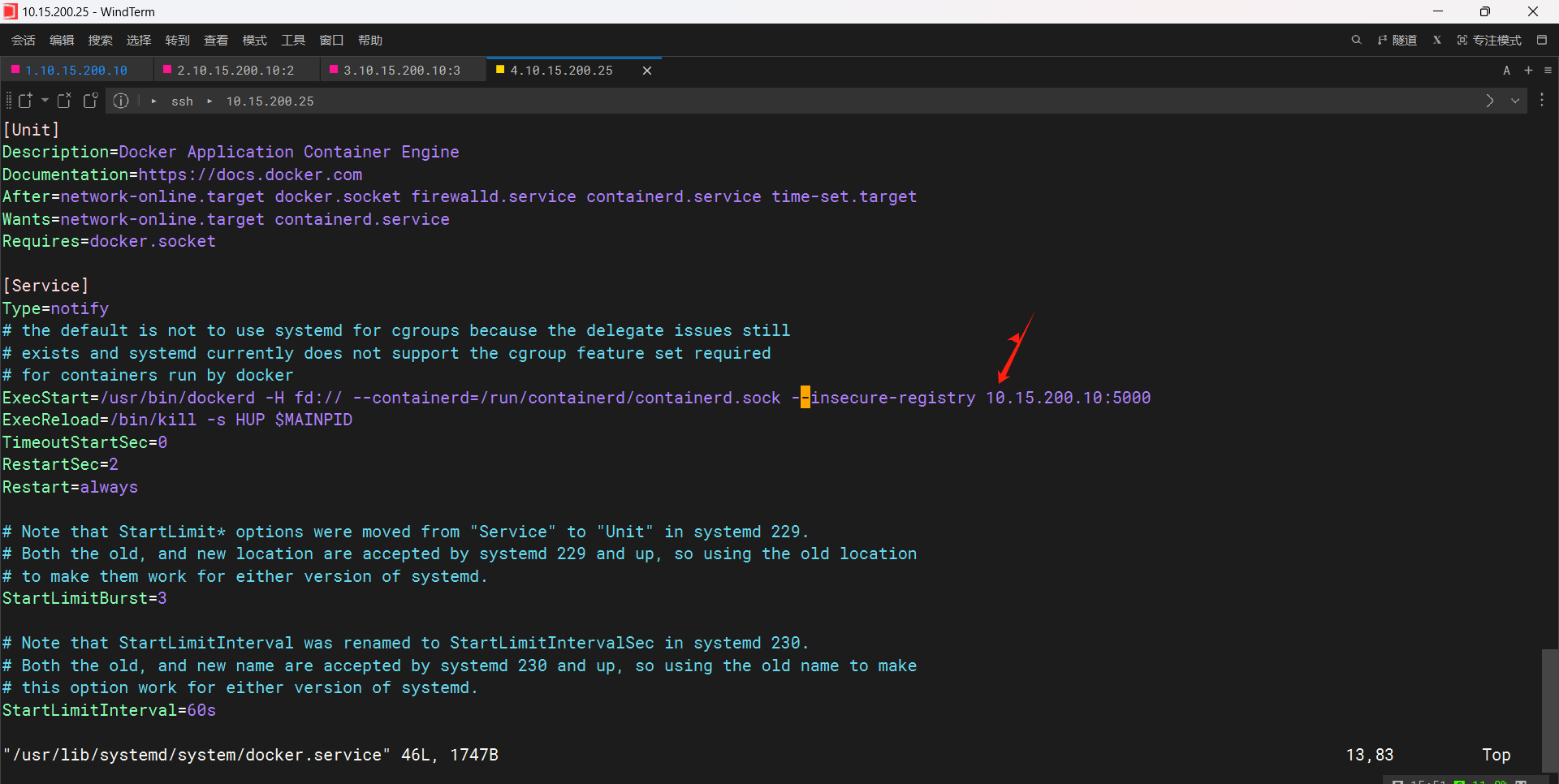


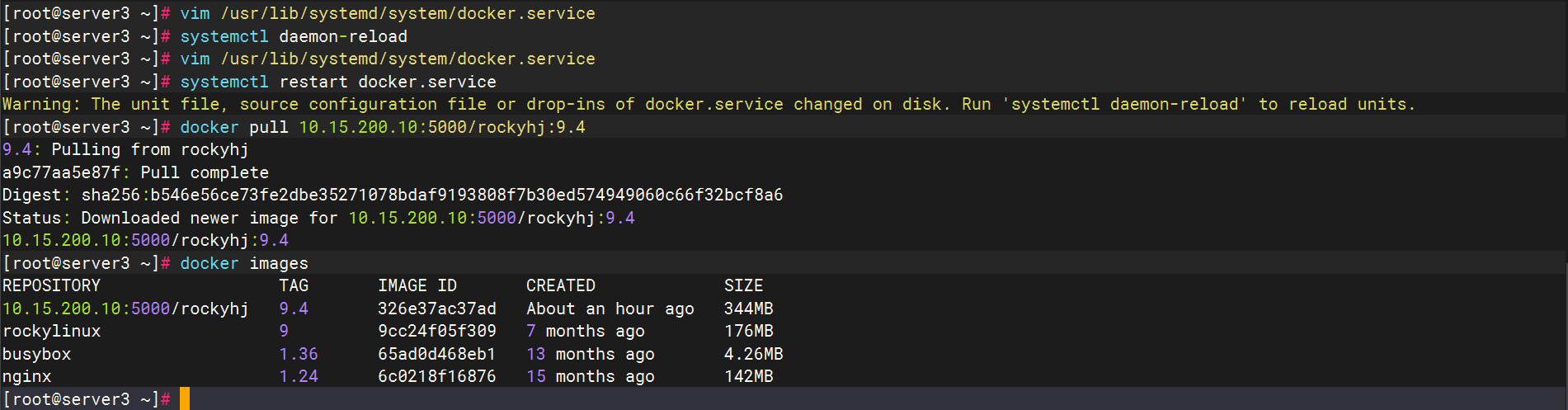


另一台先访问



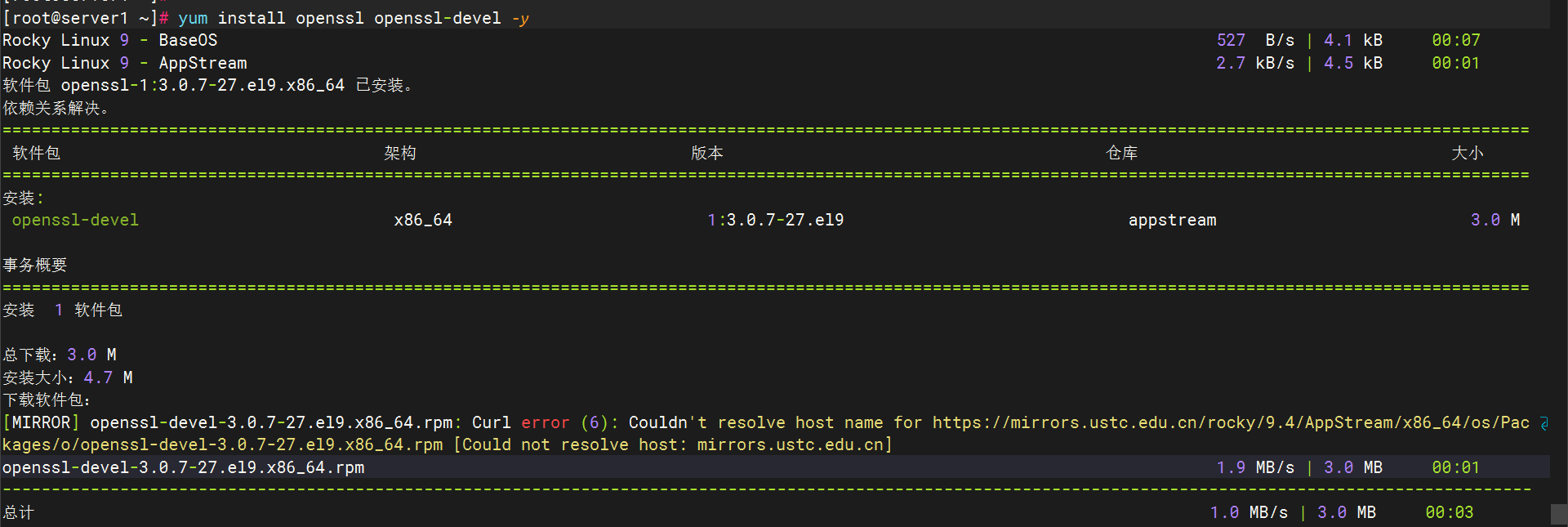
然后修改配置文件（加的和server1的数据一样），重启服务，然后下载镜像



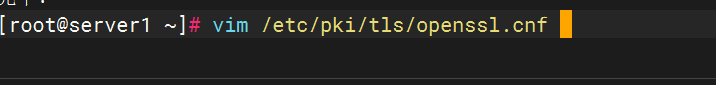


自签名证书（验证），网站（ssl2）

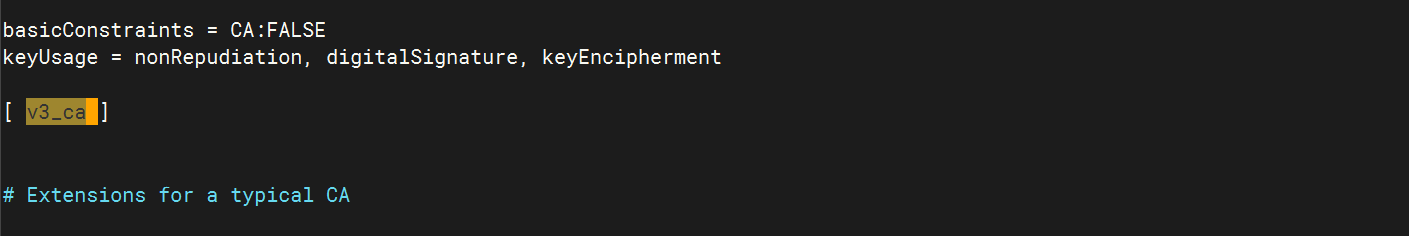
先安装服务



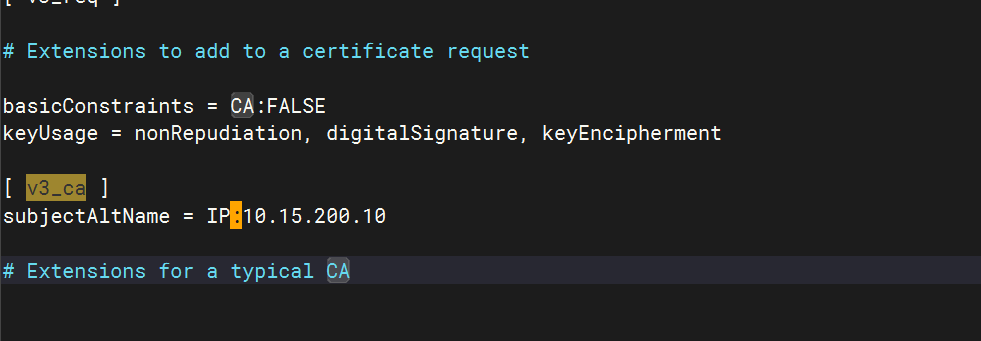
修改配置文件



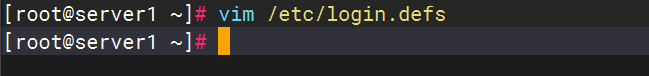
V3的颁发机构

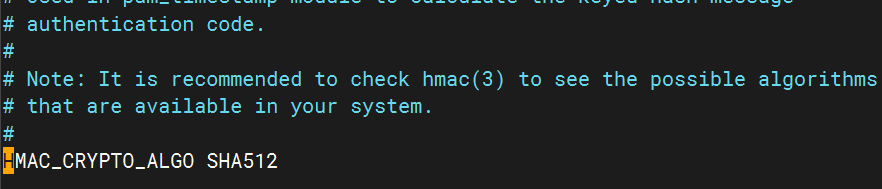


进行修改



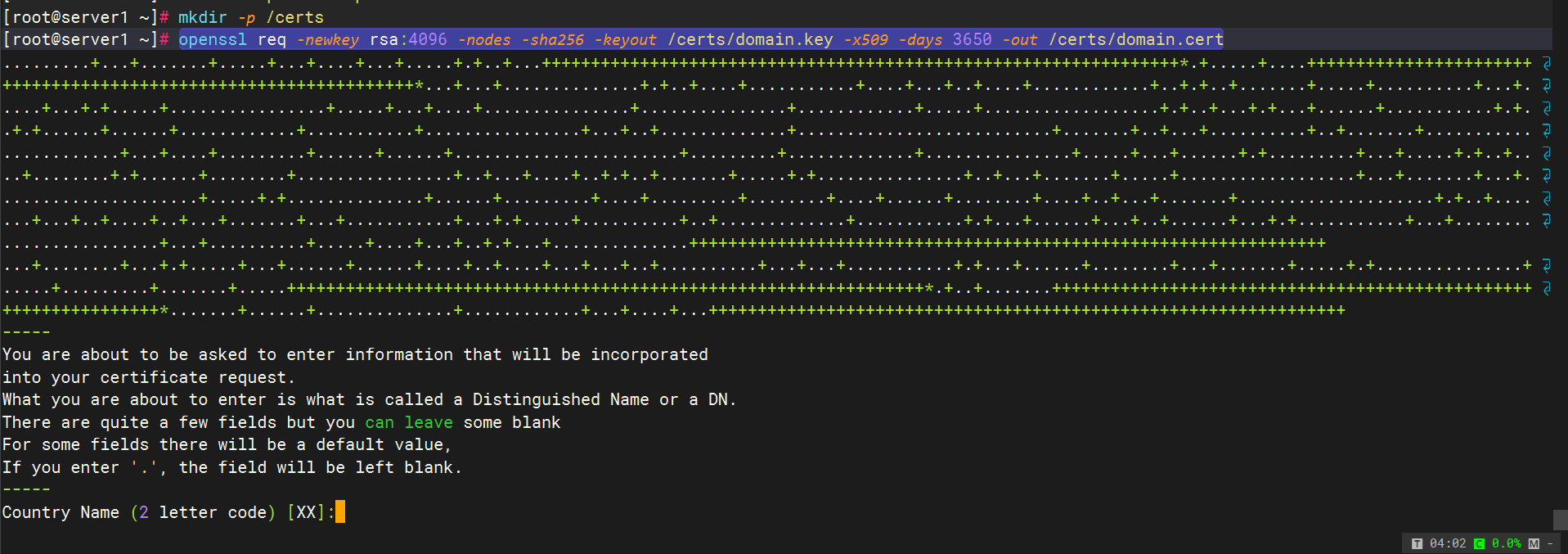
查看主机支持的加密方式



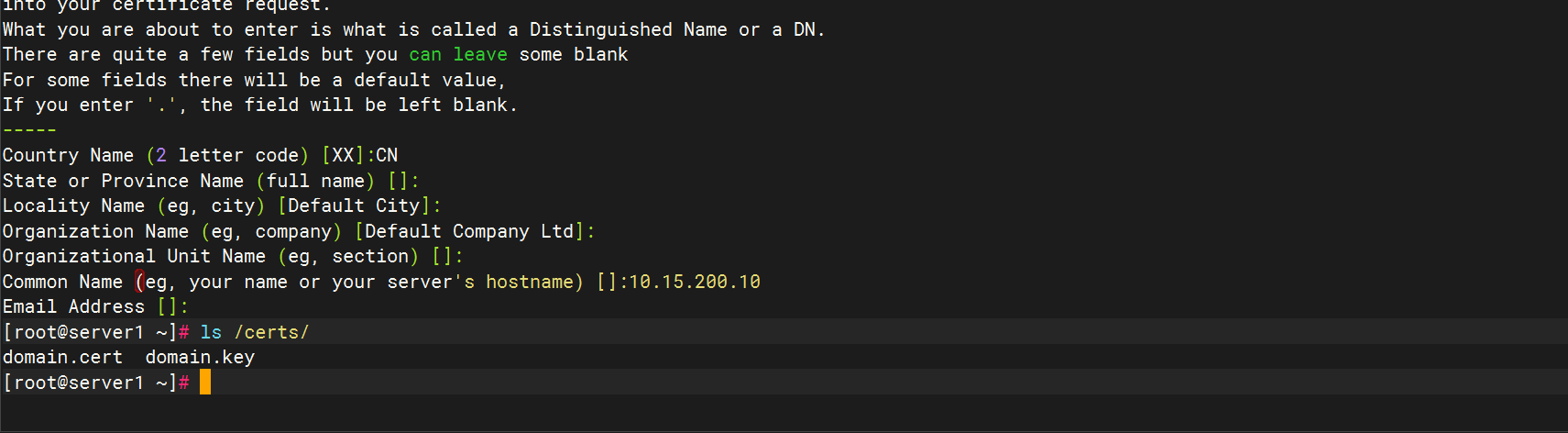


生成密钥和证书

openssl req（证书的签署请求CSR和自签名） -newkey rsa:4096（创建长度为4096的新密钥） -nodes（不加密输出私钥） -sha256（指定加密算法为sha512） -keyout /certs/domain.key（输出密钥的路径） -x509（证书的标准） -days 3650（证书有效期） -out /certs/domain.cert （证书输出的路径）

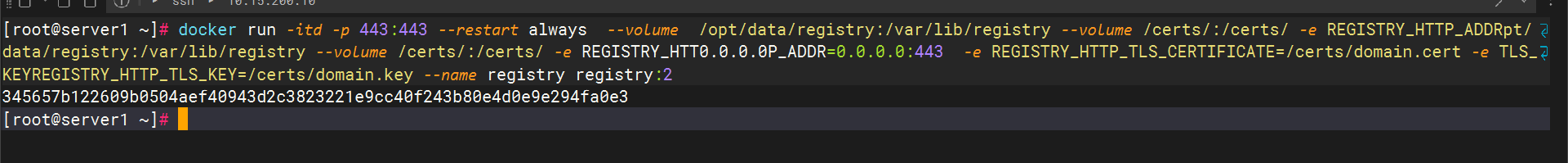


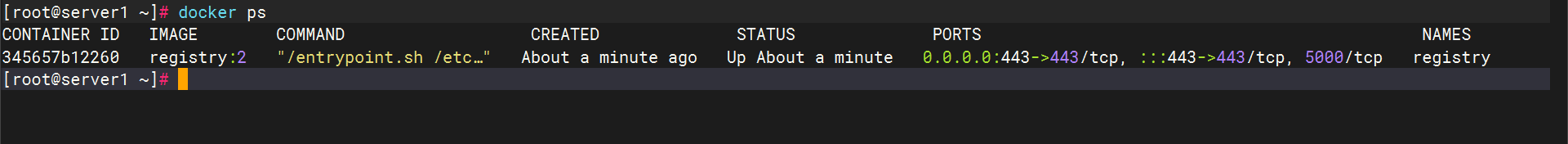
写一个国家名称和下面的IP



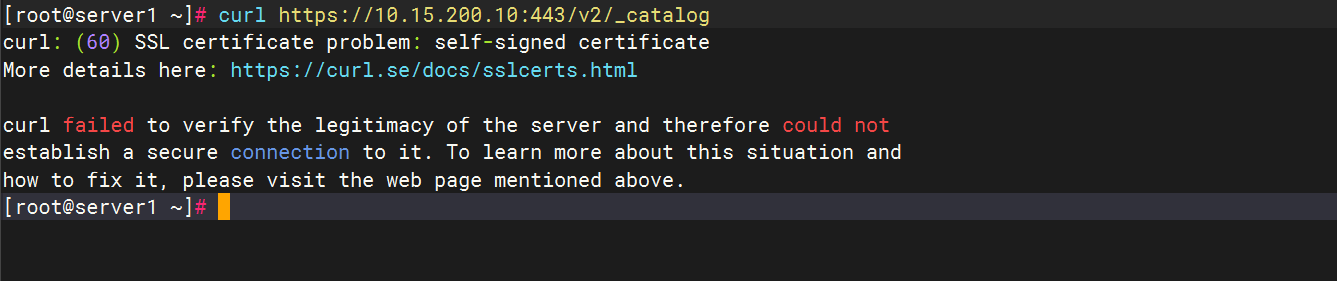
再次创建容器（添加证书路径，以及允许通过443端口进行访问）

docker run -itd -p 443:443 --restart always --volume /opt/data/registry:/var/lib/registry --volume /certs/:/certs/ -e REGISTRY\_HTTP\_ADDR=0.0.0.0:443 -e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.cert -e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key --name registry registry:2

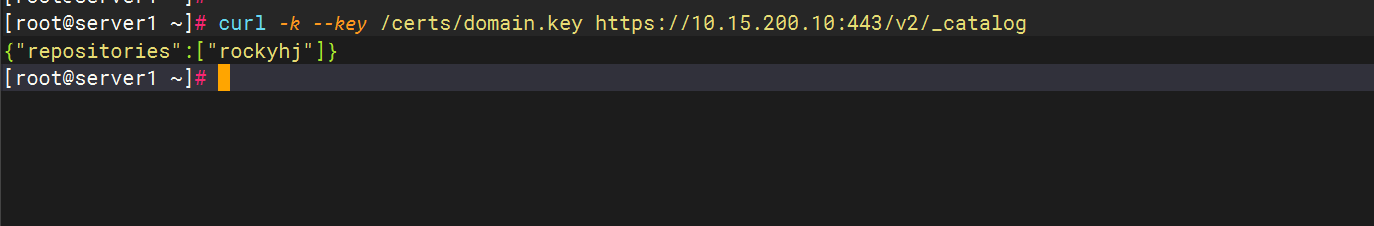




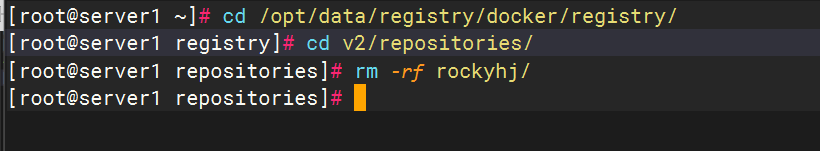
这时会报错

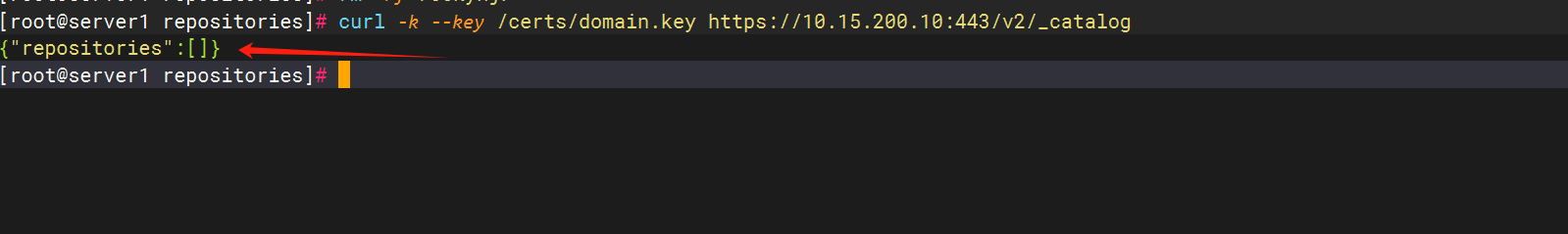


得携带自签名的证书，这时就可以访问

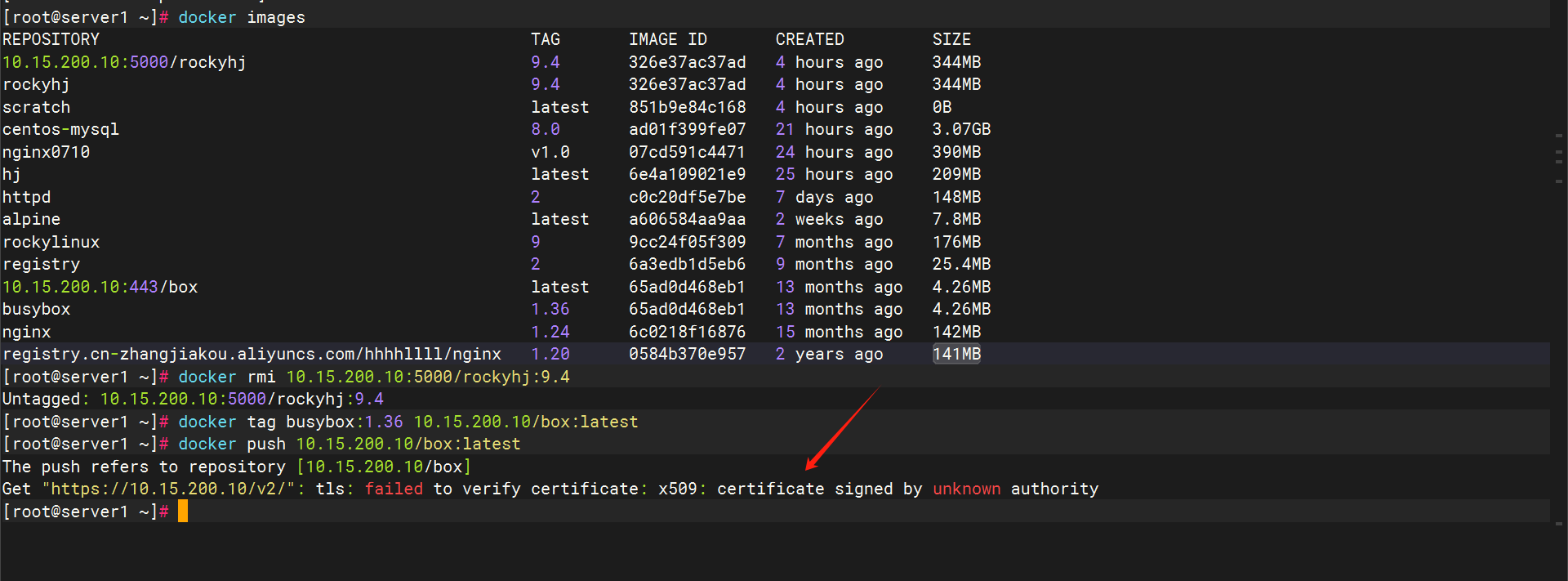


删除镜像





我们修改busybox的镜像，这个报错是因为需要证书密钥

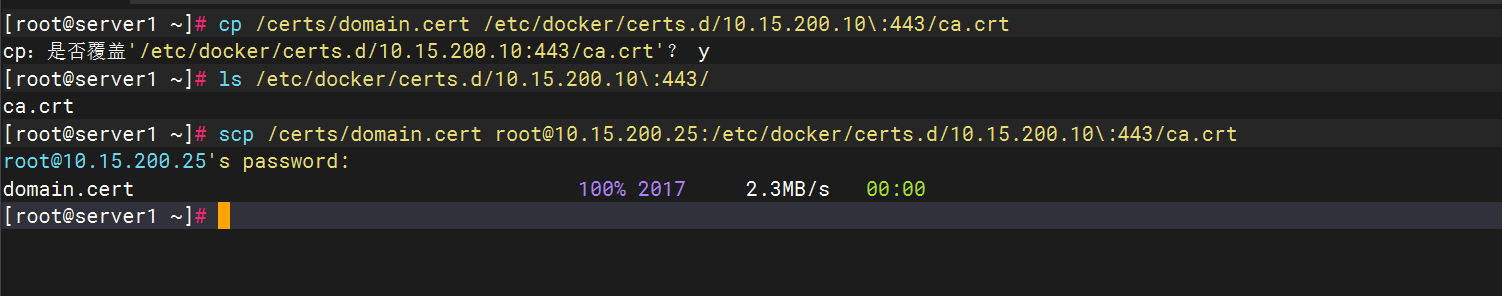


设置客户端，两台主机创建目录

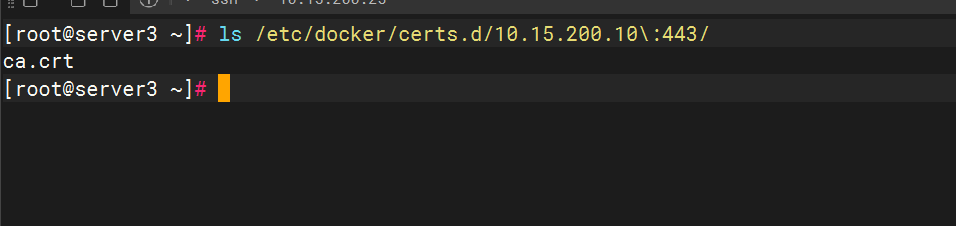




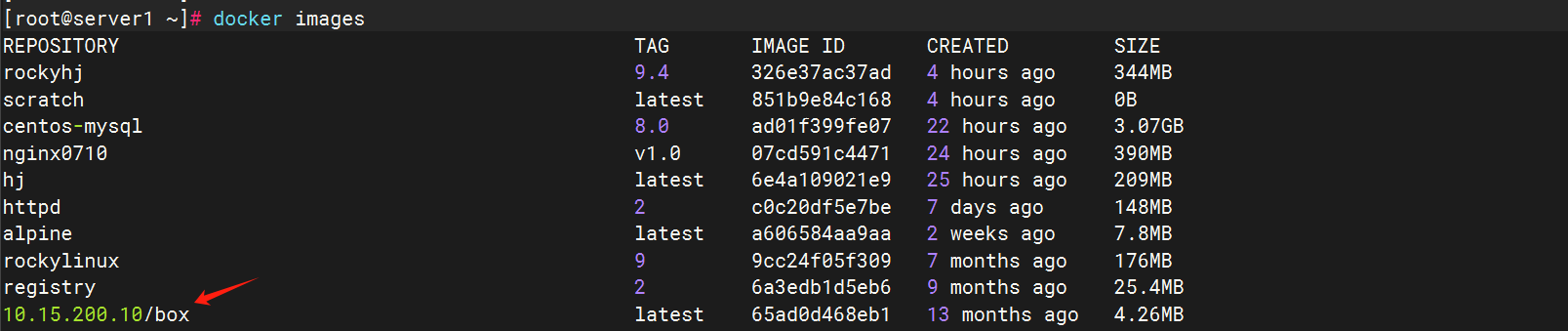
将证书进行复制



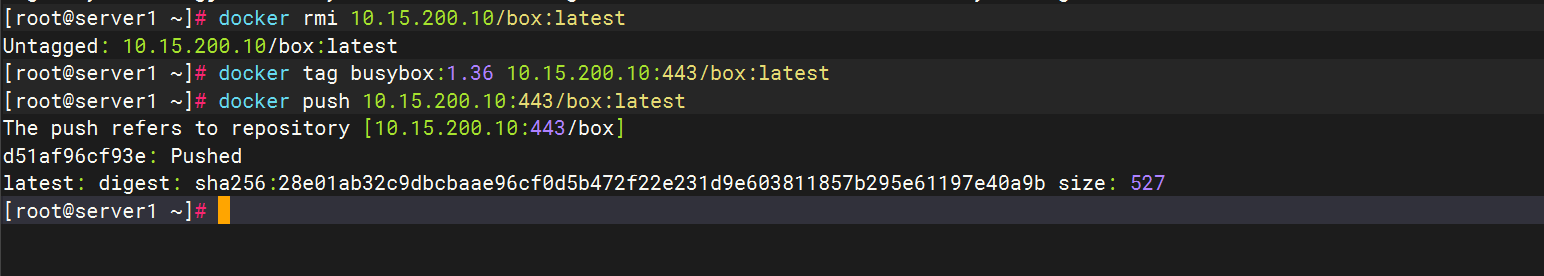
客户端查看



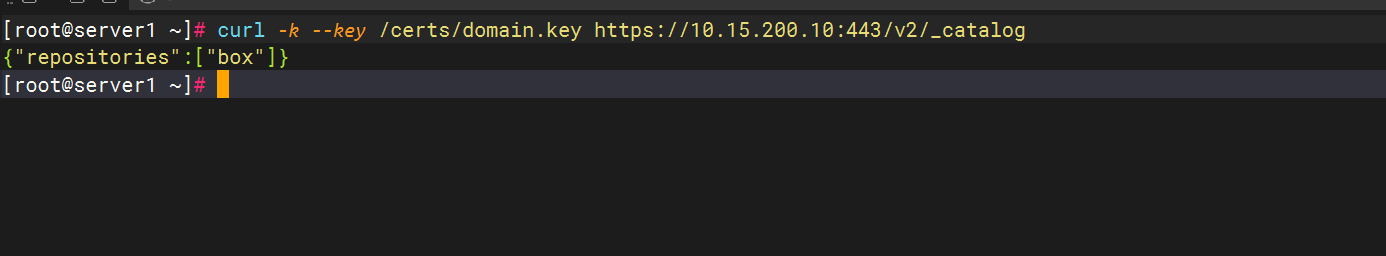
然后修改镜像（因为这个没有加443，的删除）



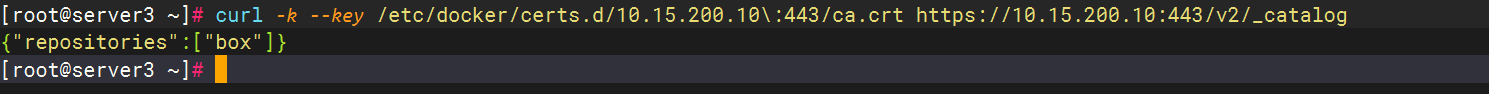
然后在上传镜像就可以了

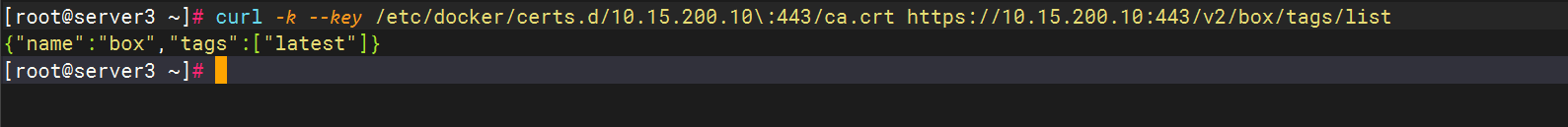


这样就能看到仓库里的box镜像

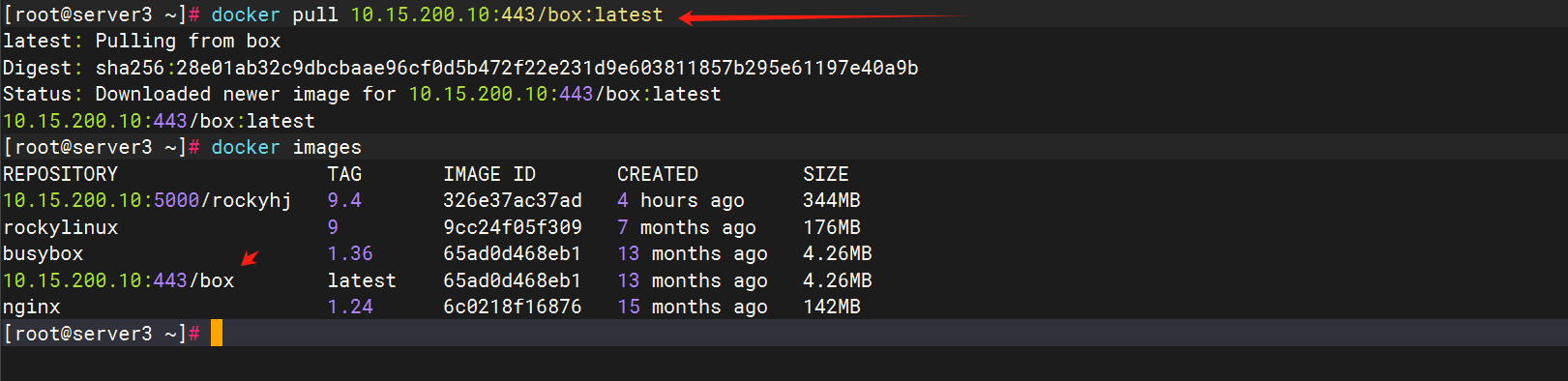


到客户端查看验证

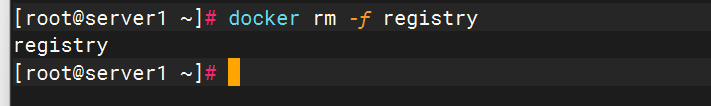




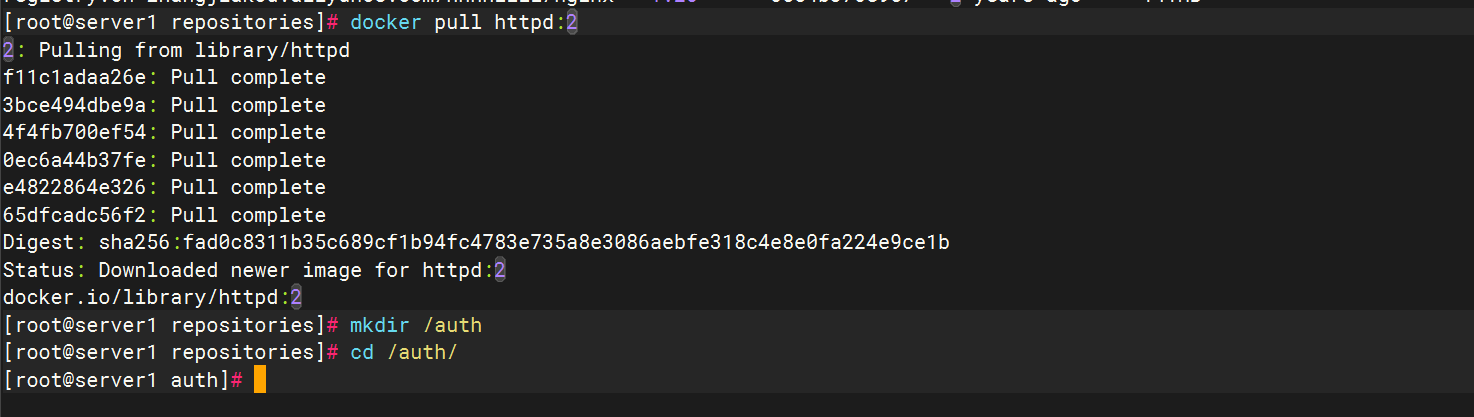
客户端验证是否能下载镜像



接下来创建账号密码（可以将之间的容器删除）



创建一个目录（把账号和密码放在这个/auth目录下）



--entrypoint htpasswd： 用于生成密码用httpd:2镜像

B:不能以明文形式验证（必须使用TLS验证），

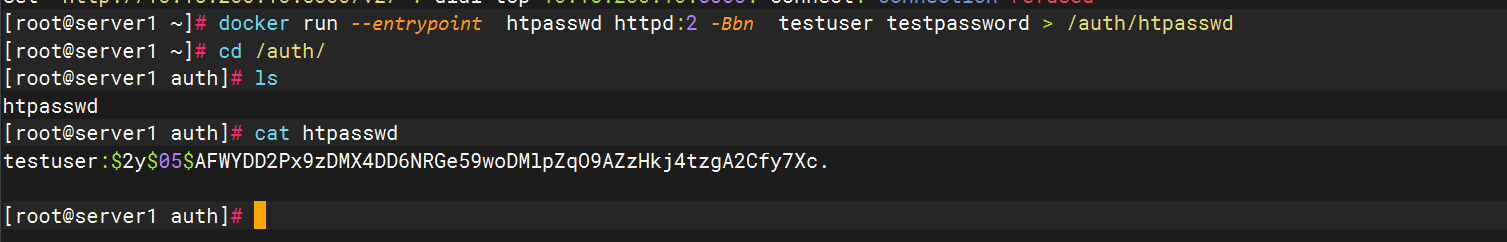
b：创建新的用户，

n:不更新/etc/passwd的文件

-m是指定使用MD5，不能使用sha256验证

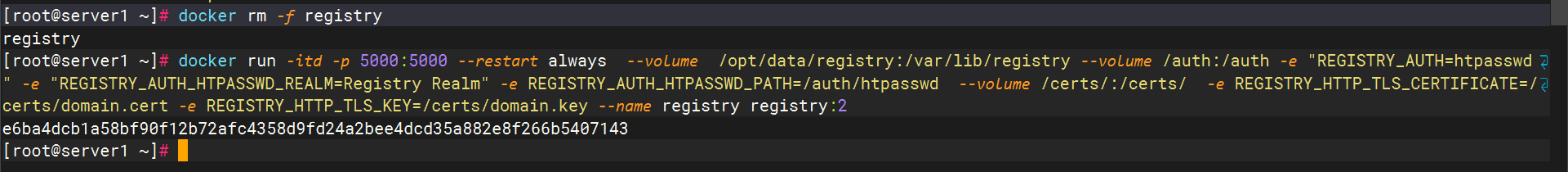
testuser 是用户

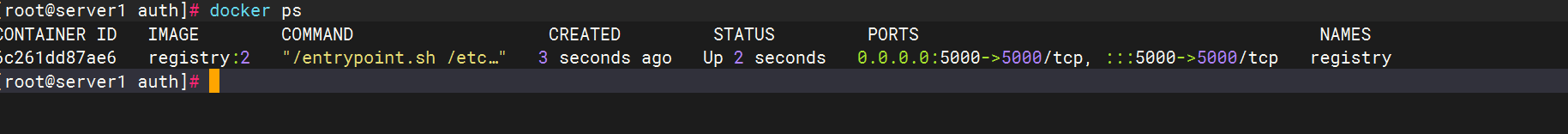
testpassword 是密码



创建容器

docker run -itd -p 5000:5000 --restart always --volume /opt/data/registry:/var/lib/registry --volume /auth:/auth -e "REGISTRY\_AUTH=htpasswd" -e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm" -e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd --volume /certs/:/certs/ -e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.cert -e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key --name registry registry:2

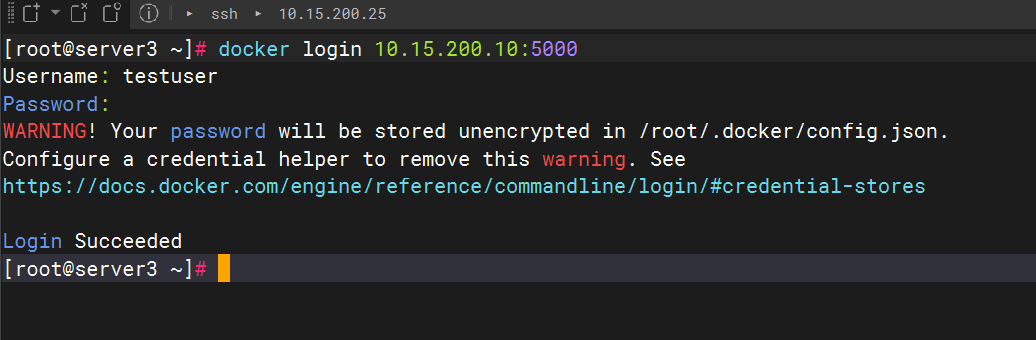


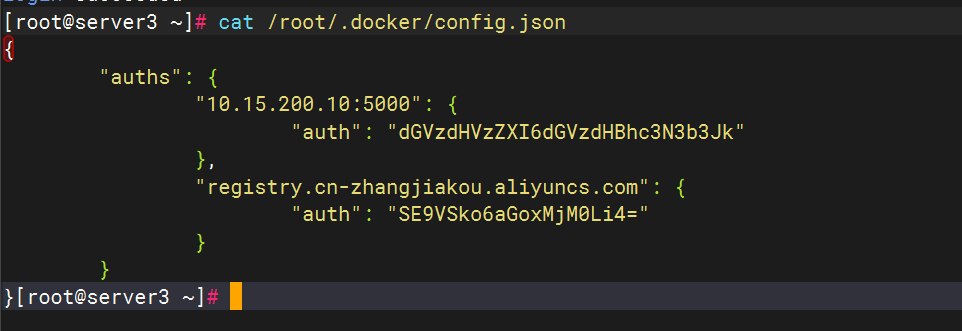


第二台登录

testuser 是用户

testpassword 是密码





可以下载镜像

