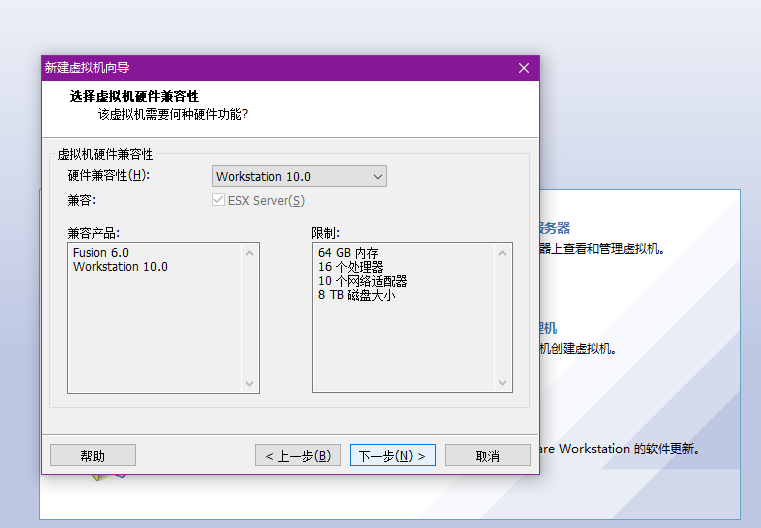
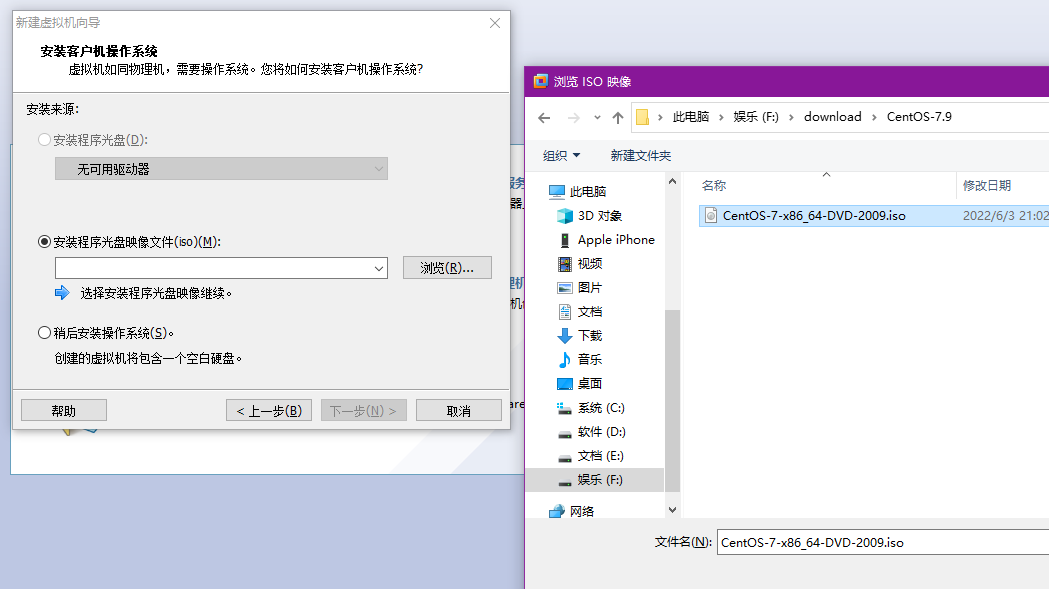
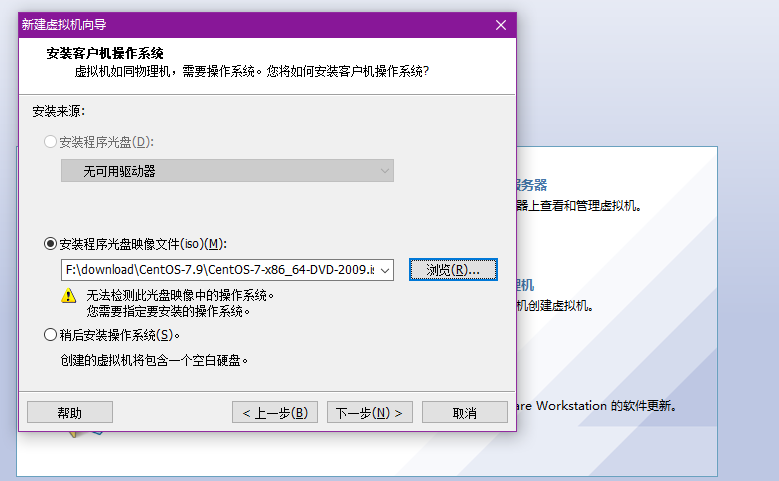
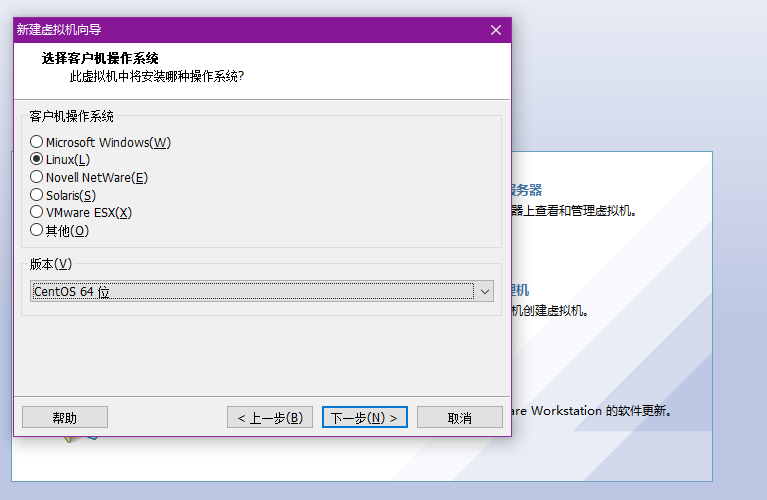
虚拟机创建步骤如下：

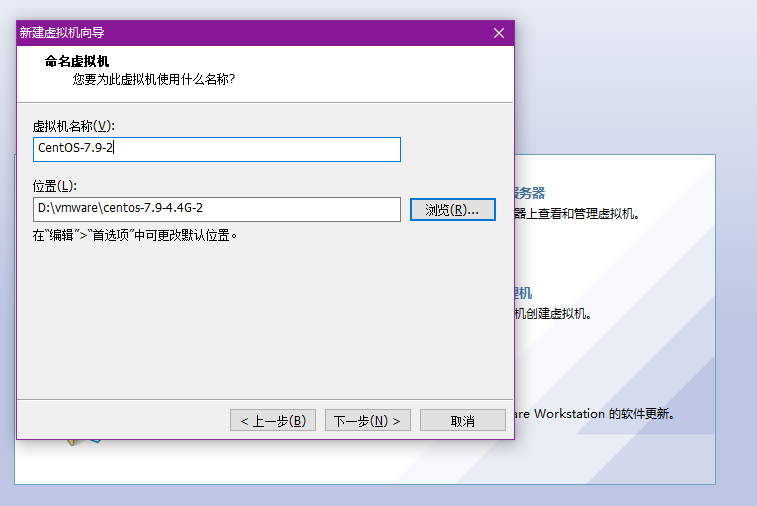


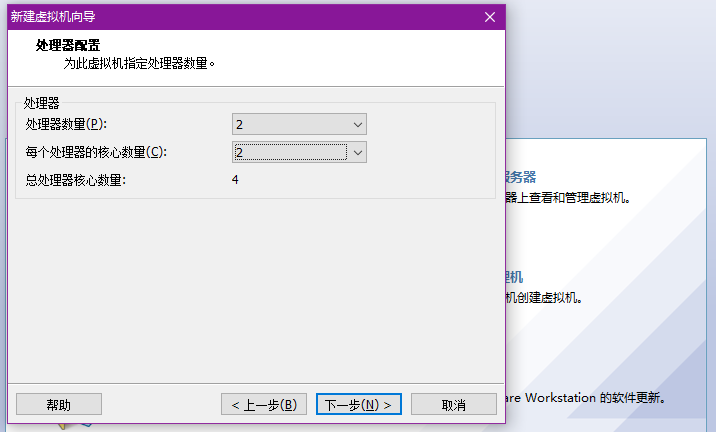


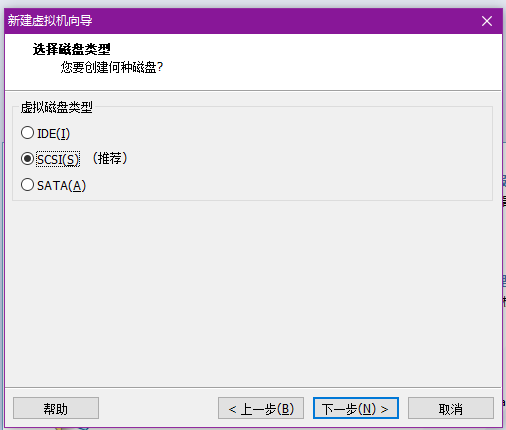
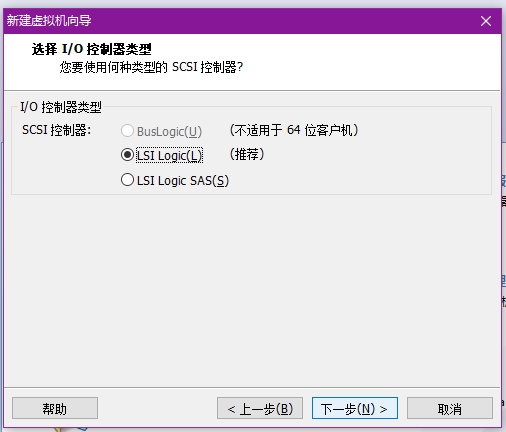
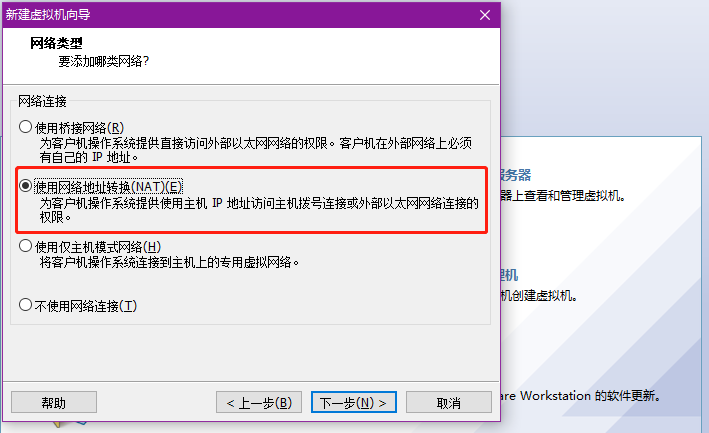
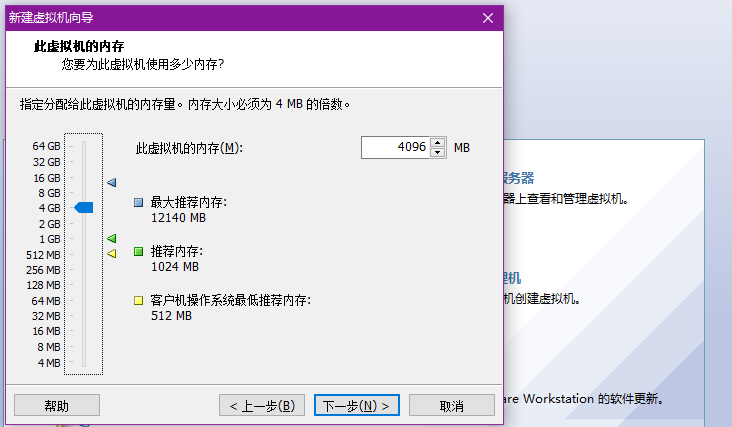


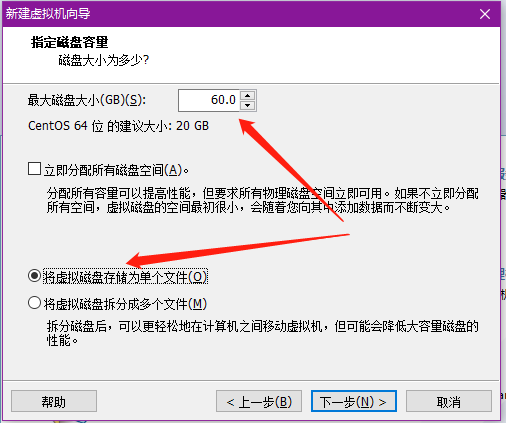
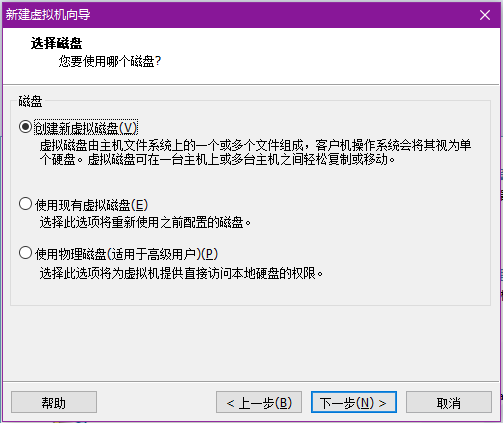


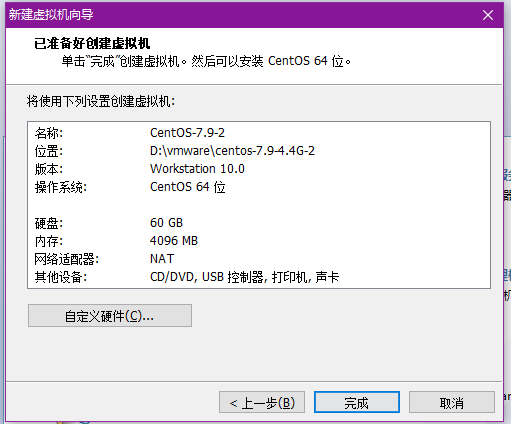
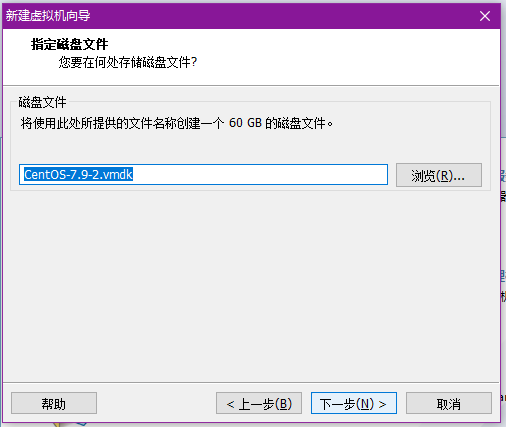


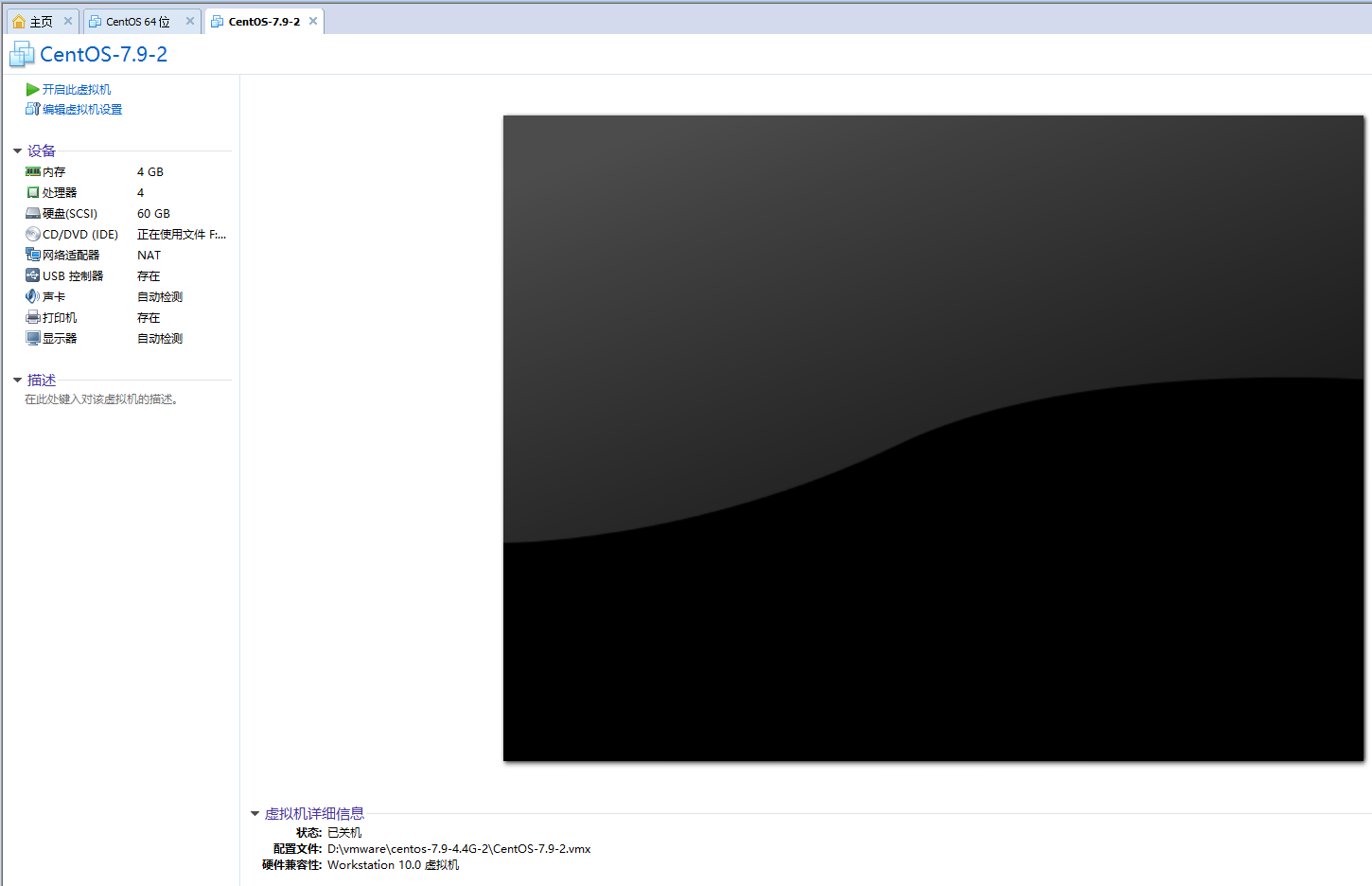




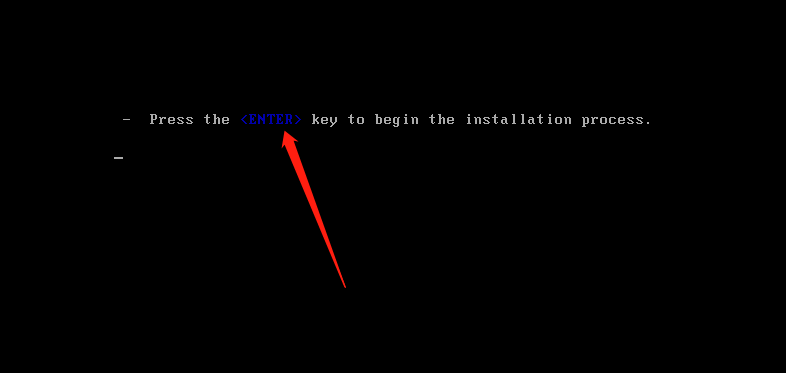




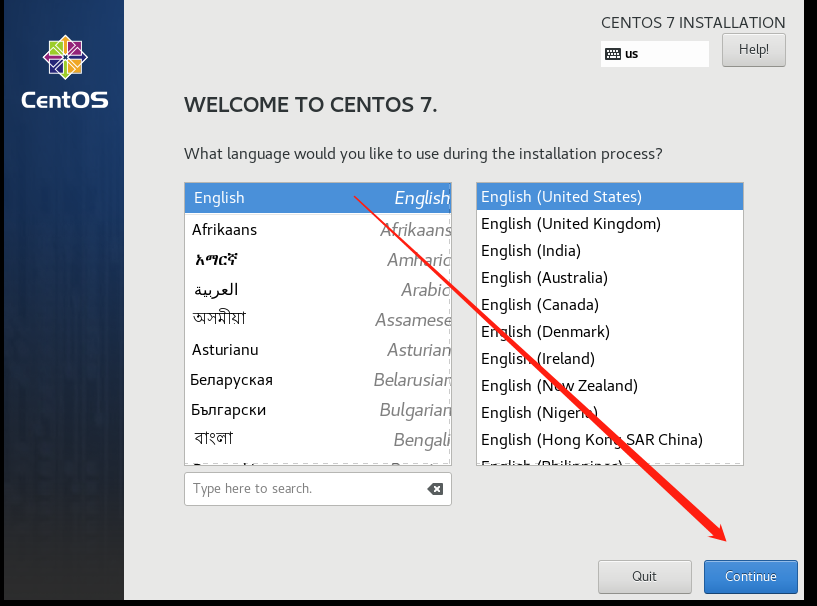




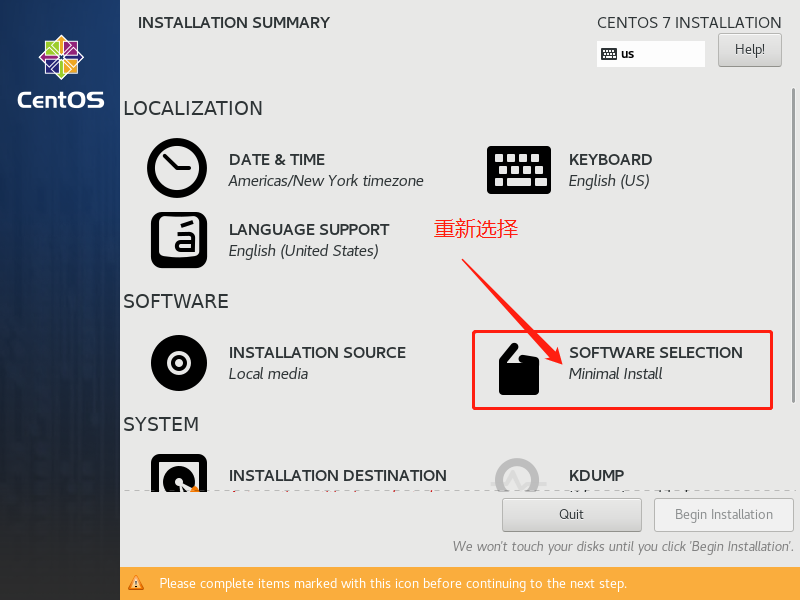


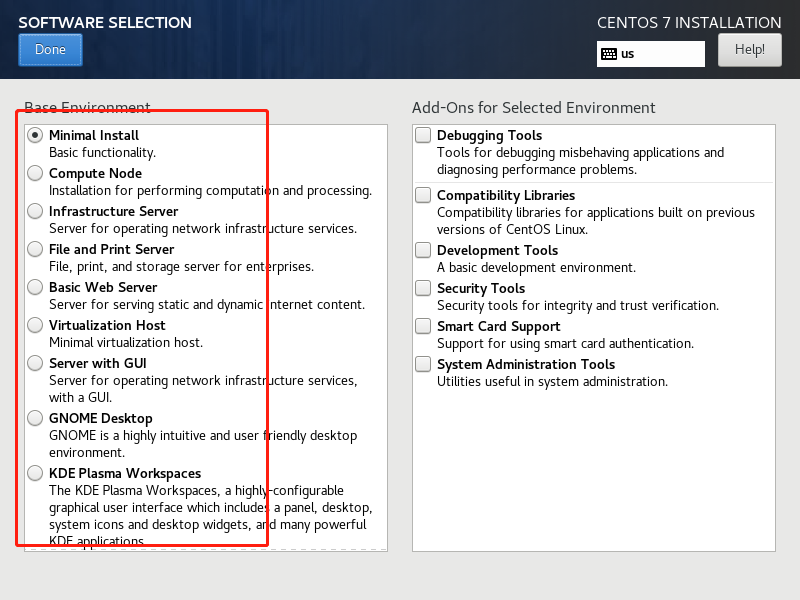


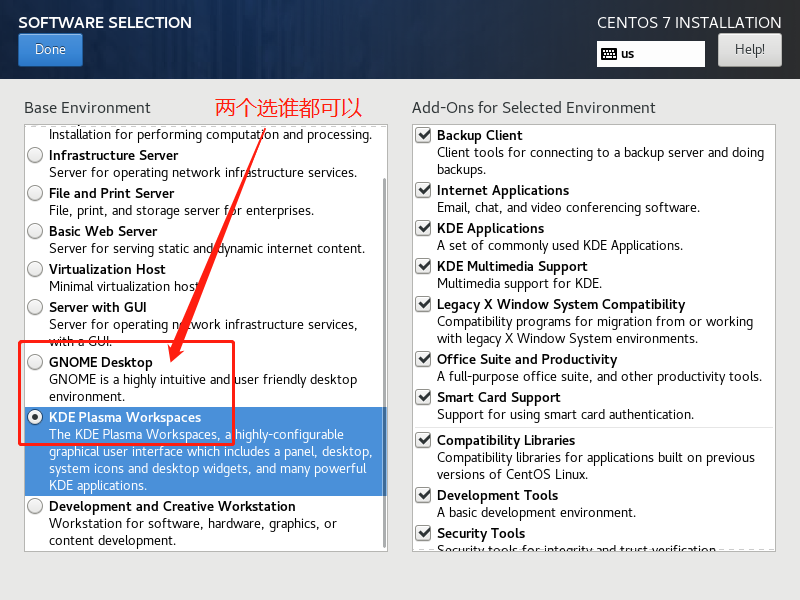
等待后续自动安装完成。

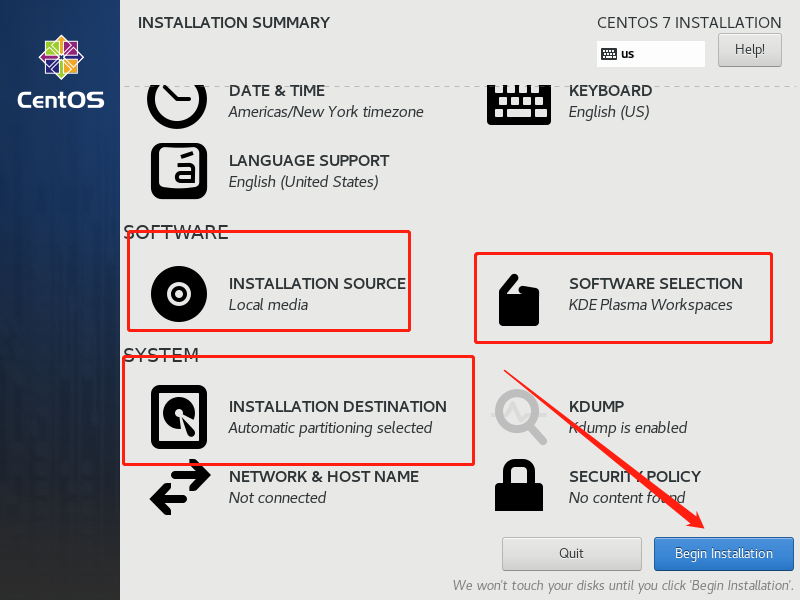


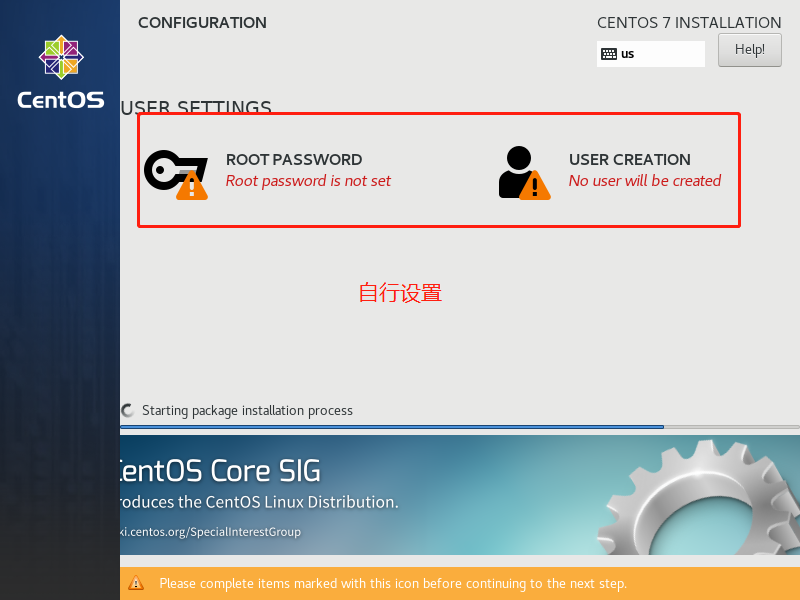




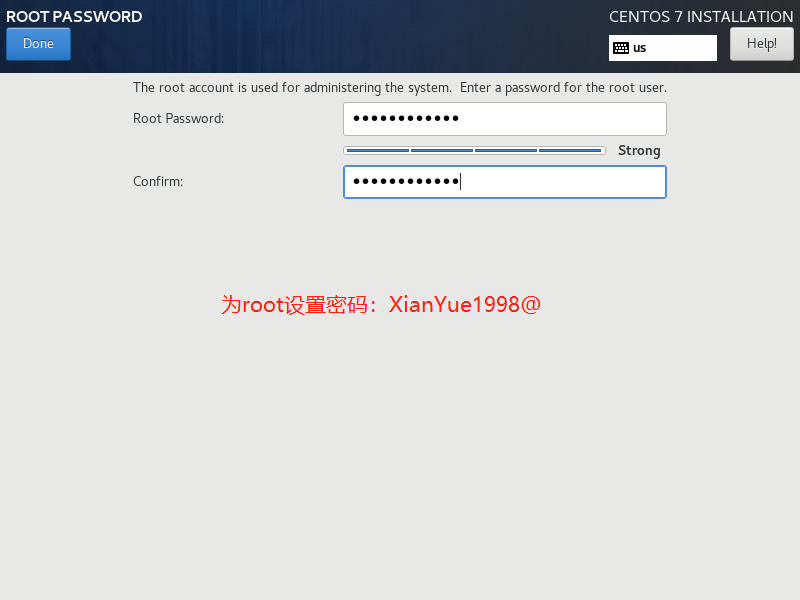


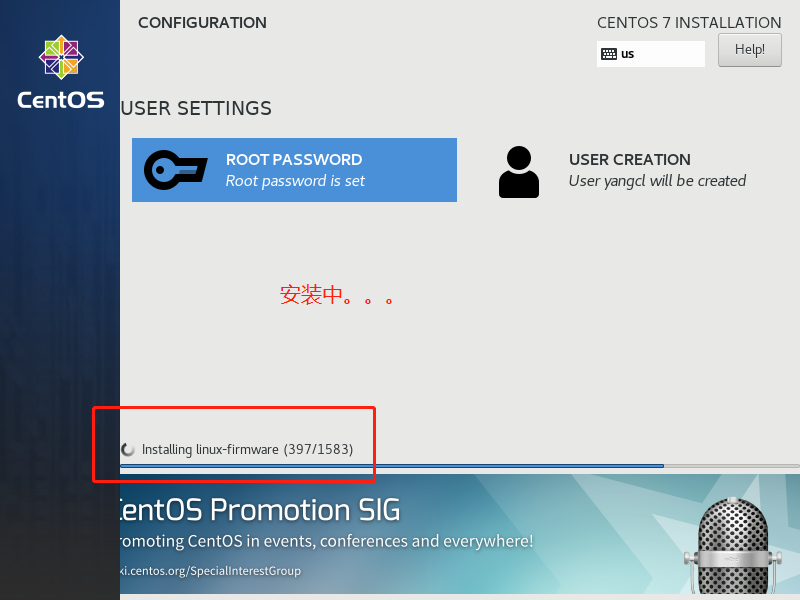


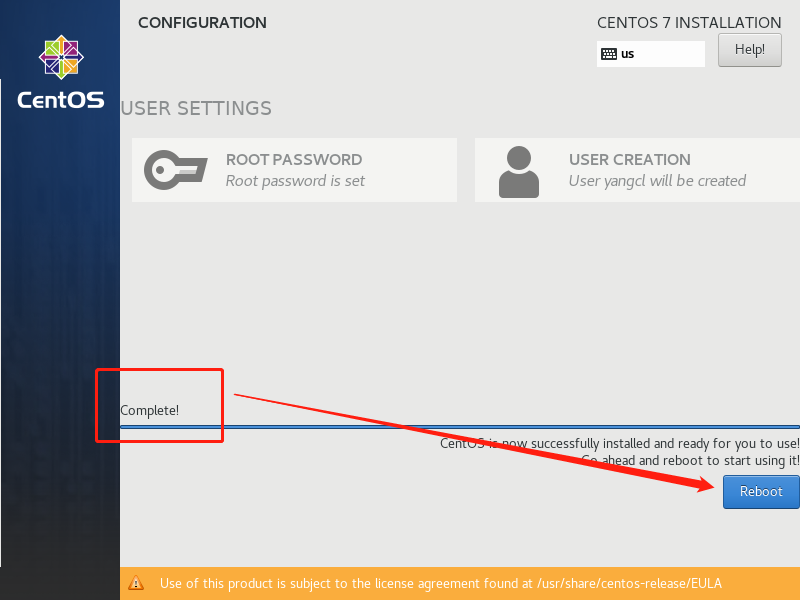




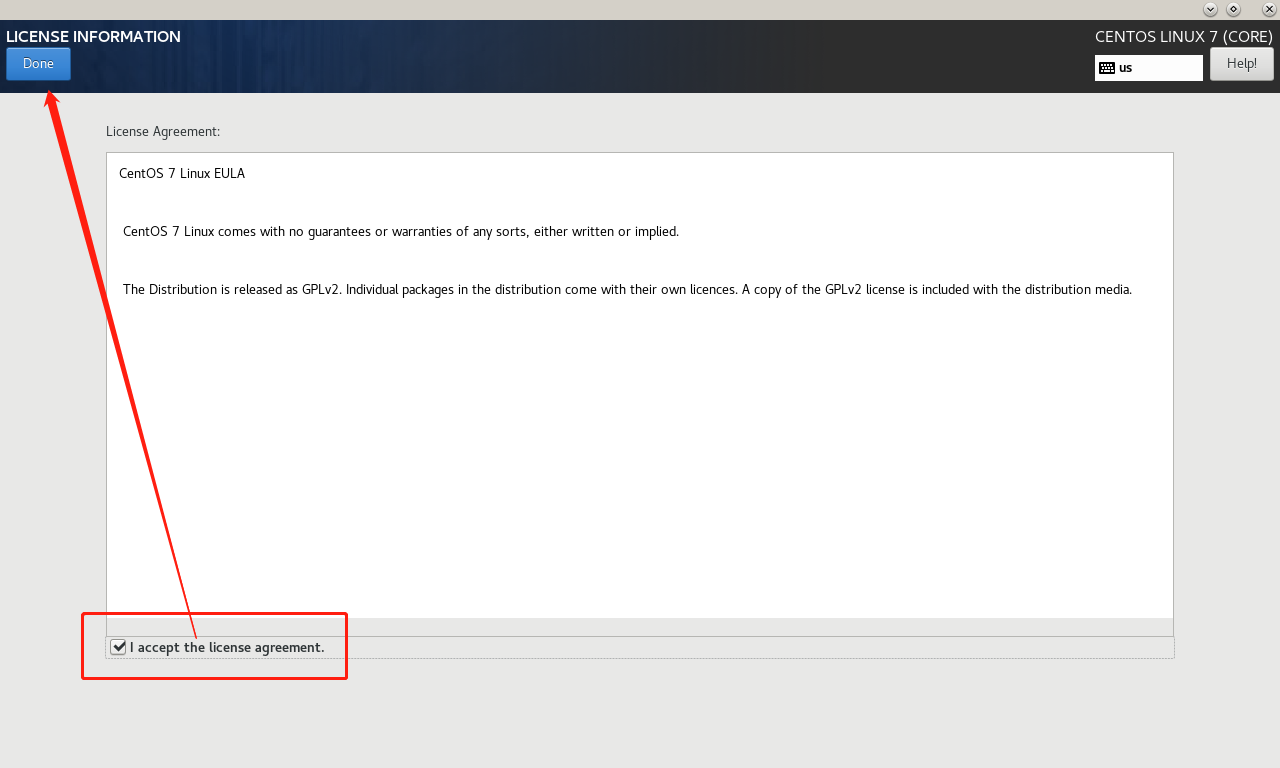


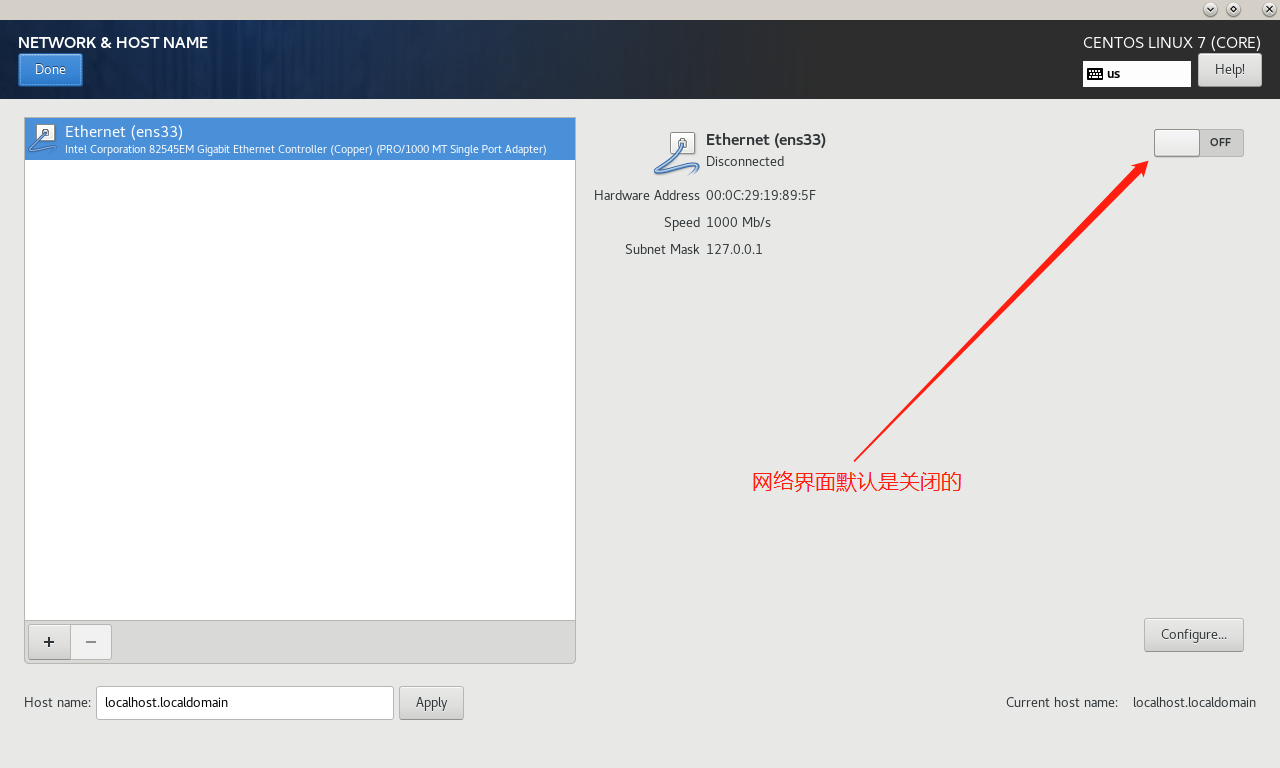


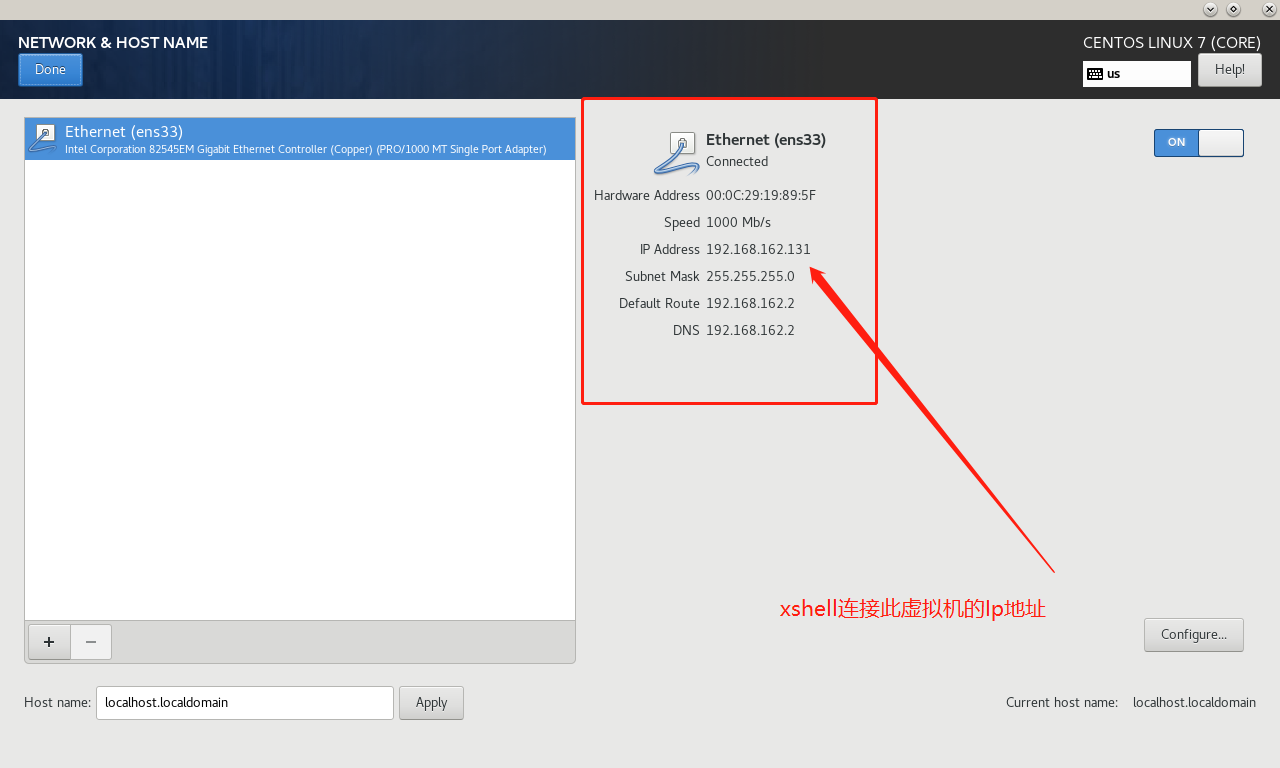


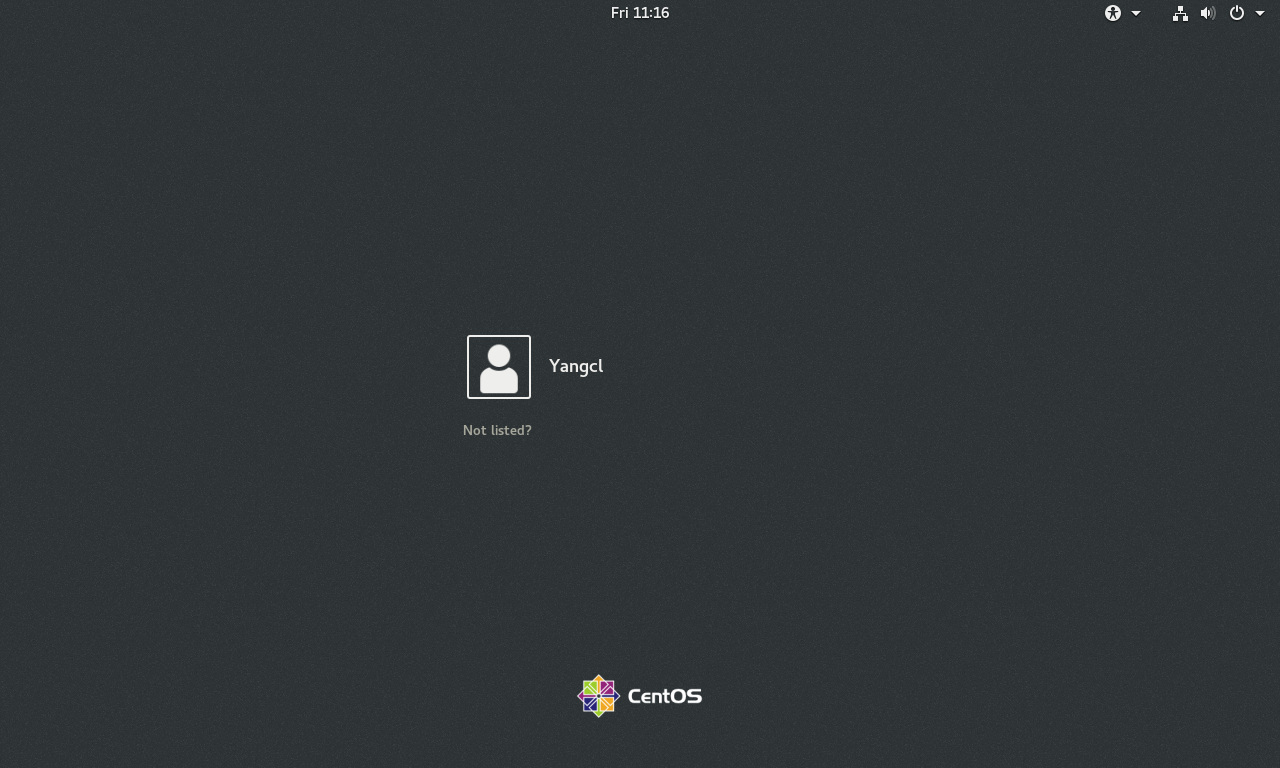


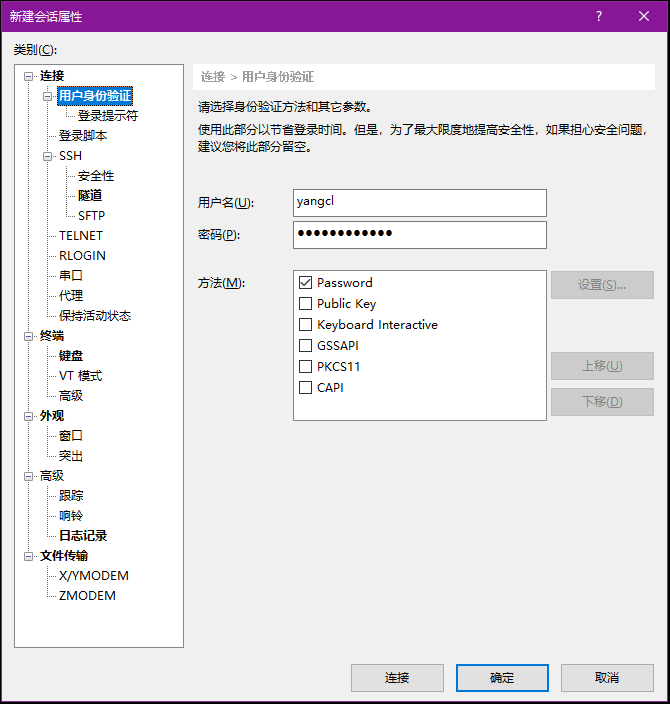
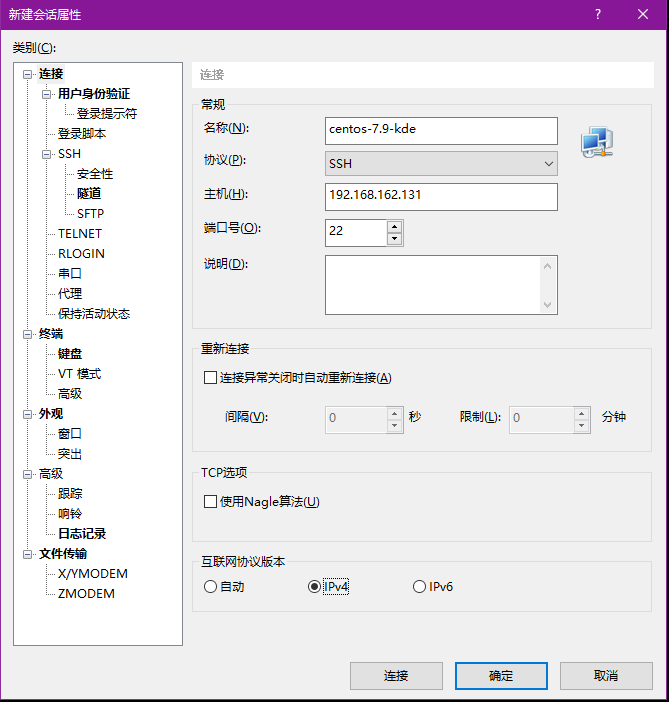


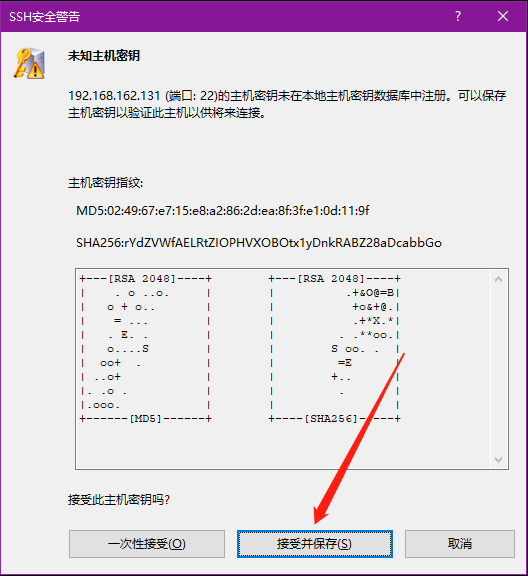










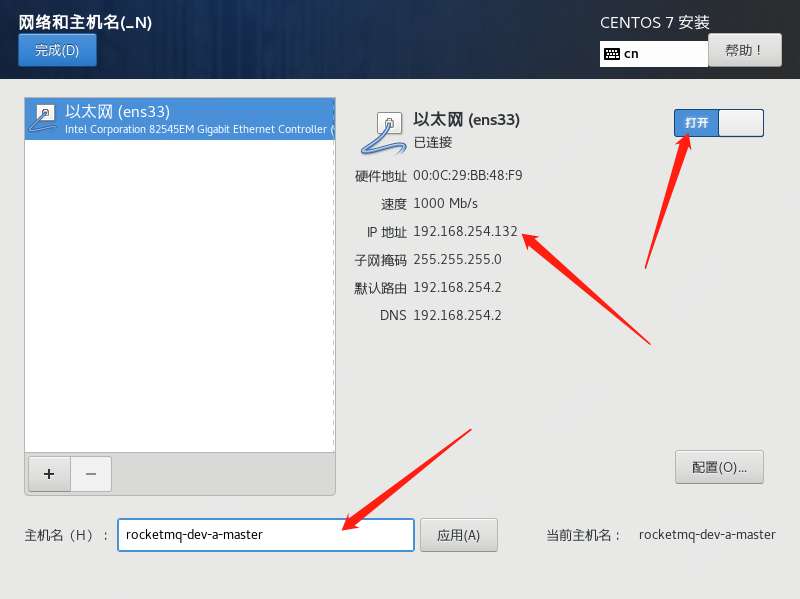


虚拟机最小化安装问题汇总

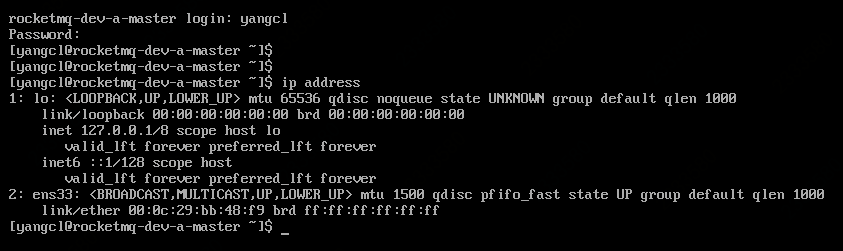
1、安装完成无内网IP问题



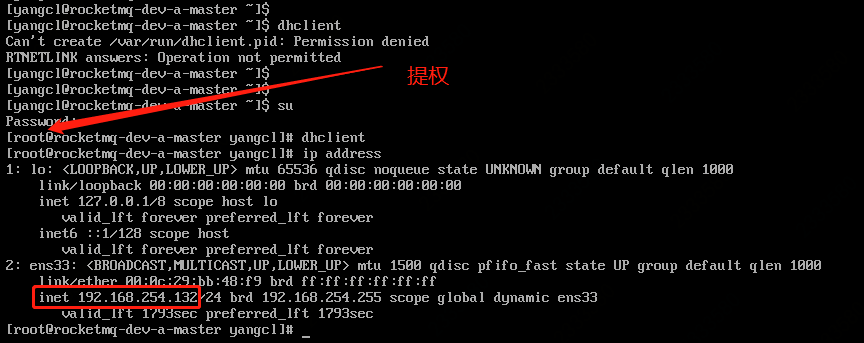




这是一种方法，还可以通过dhclient命令来实现。



可以看到，没有内网IP地址。



dhclient命令提权以后即可自动分配一个内网IP地址：192.168.254.132。但这种方式有一个比较大的缺陷：重启后IP地址会丢失。

2、linux修改主机名



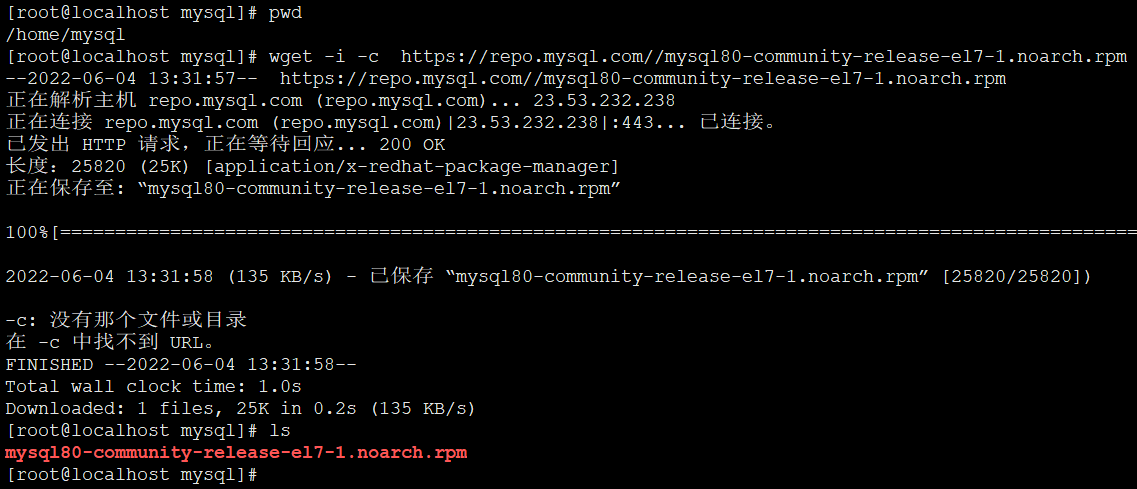
Mysql8.0安装过程

1、准备工作

创建一个文件夹，用于存放MySQL安装文件：/home/mysql。

下载并安装 yum repostory：

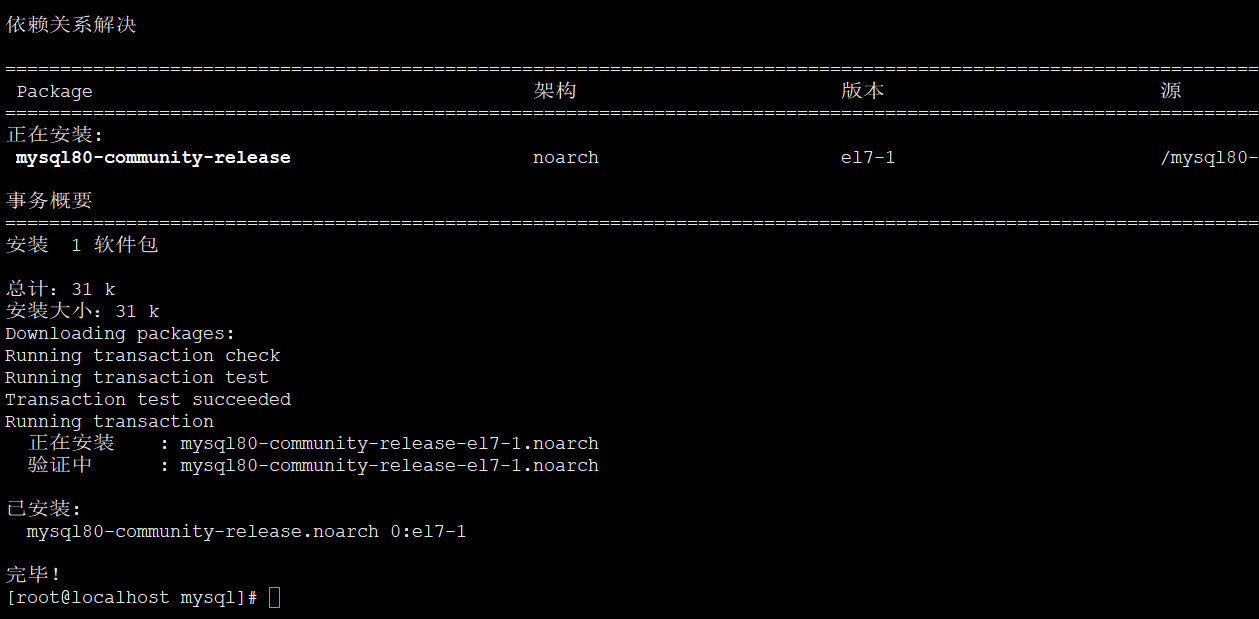
wget -i -c https://repo.mysql.com//mysql80-community-release-el7-1.noarch.rpm



安装MySQL：

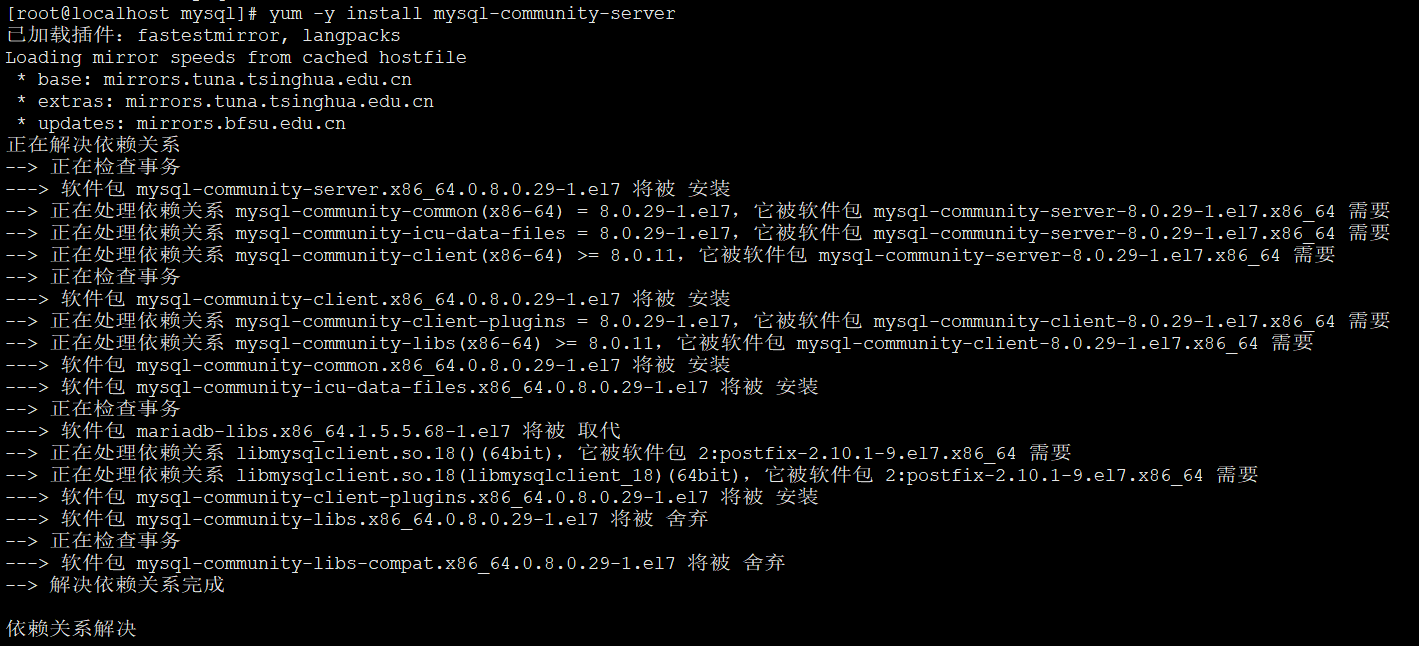
yum -y install mysql80-community-release-el7-1.noarch.rpm

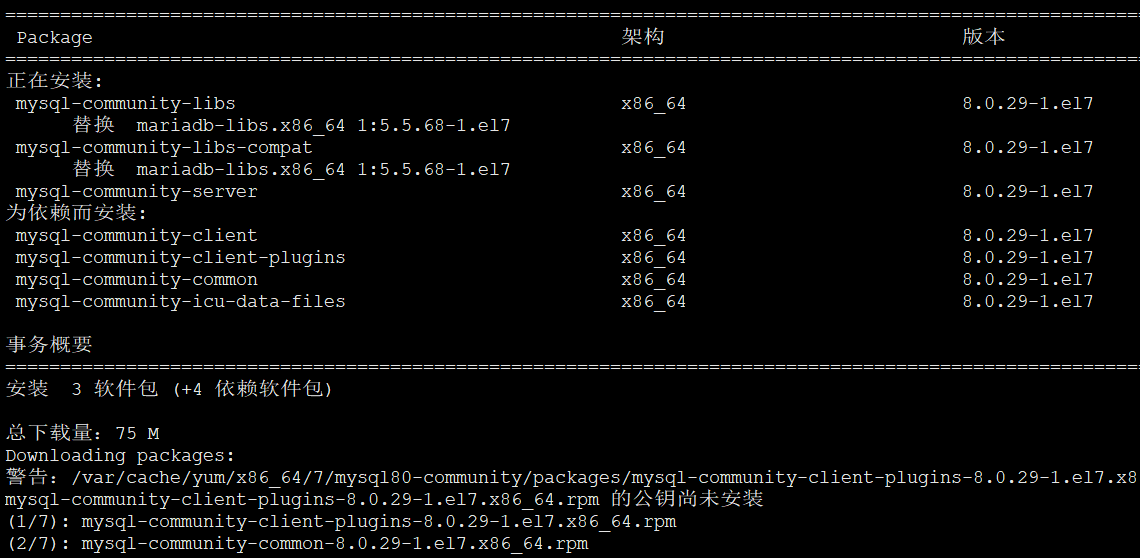


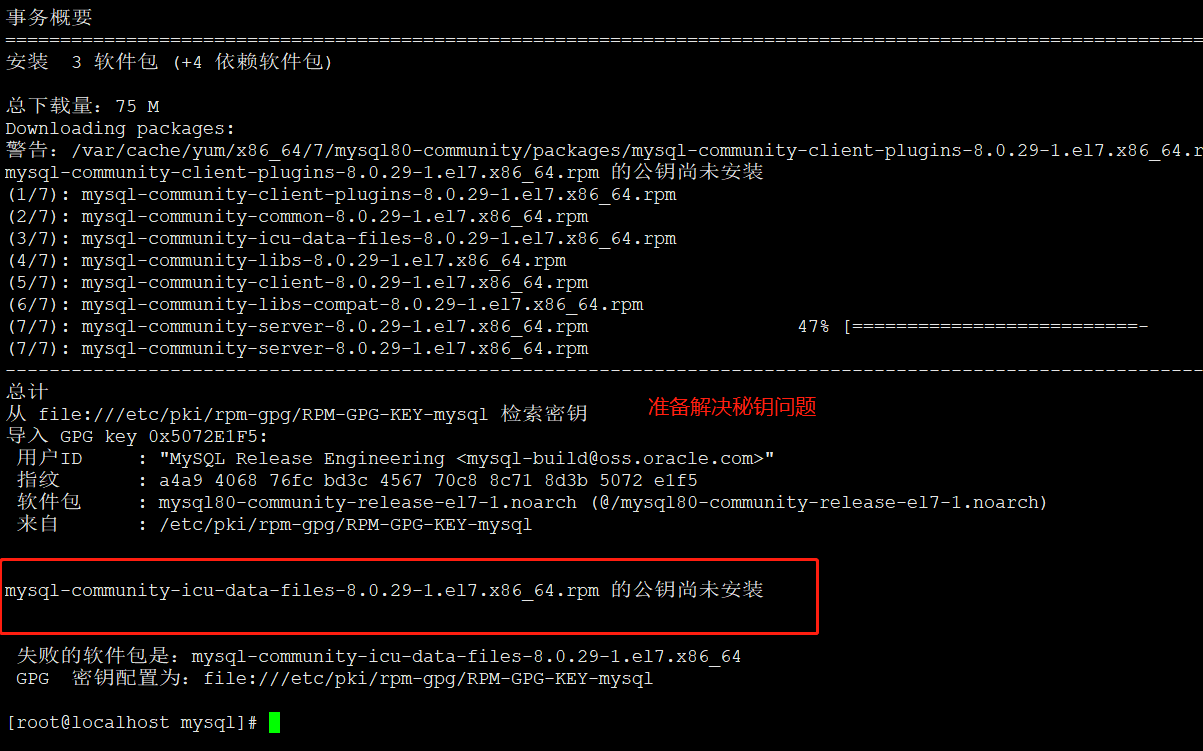


安装mysql服务：

yum -y install mysql-community-server

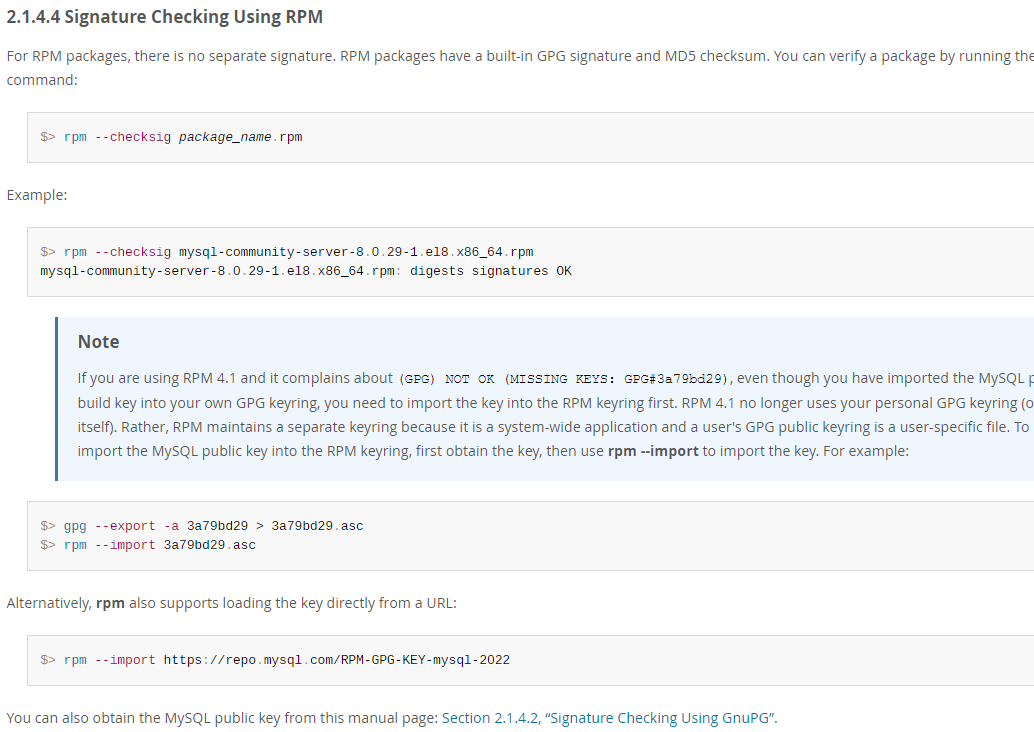


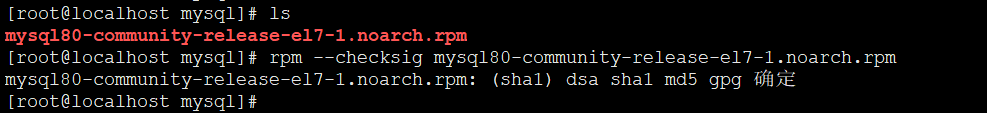


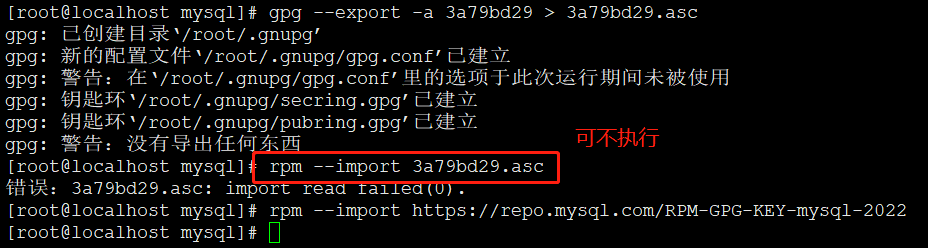


2、yum安装Mysql8提示“公钥尚未安装”

参考官网：https://dev.mysql.com/doc/refman/8.0/en/checking-rpm-signature.html







命令如下：

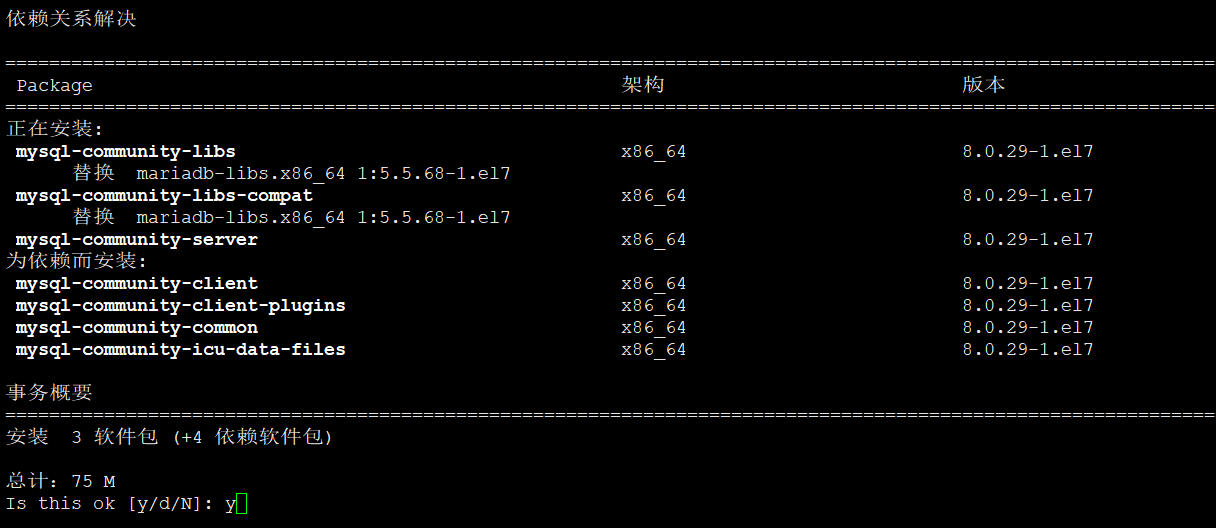
gpg --export -a 3a79bd29 > 3a79bd29.asc

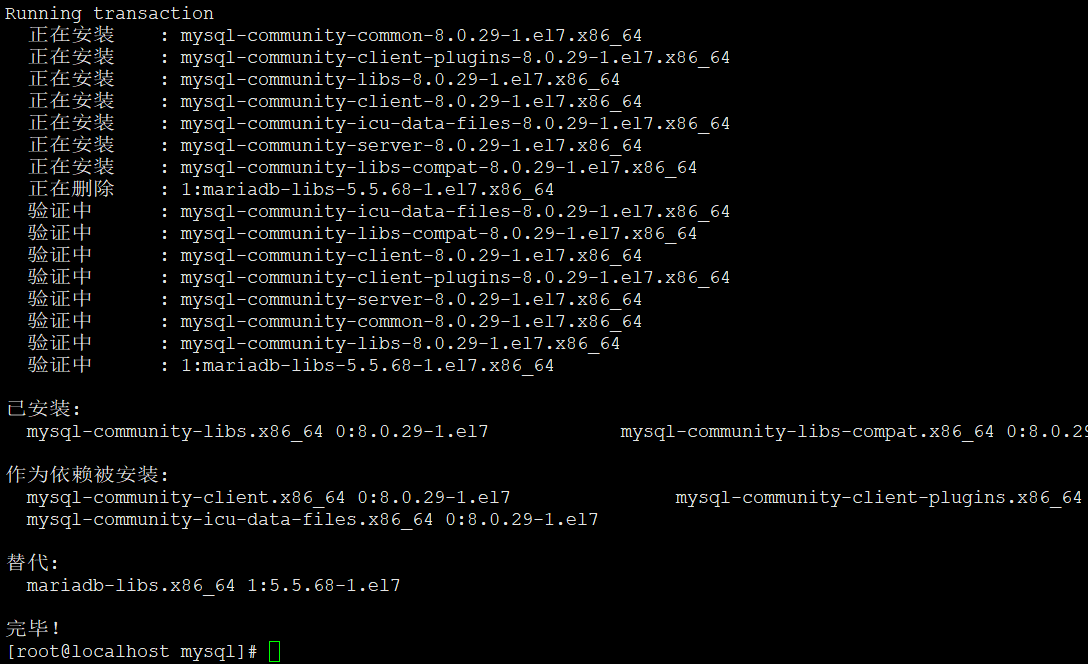
rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022

再次执行安装命令：

yum install mysql-community-server





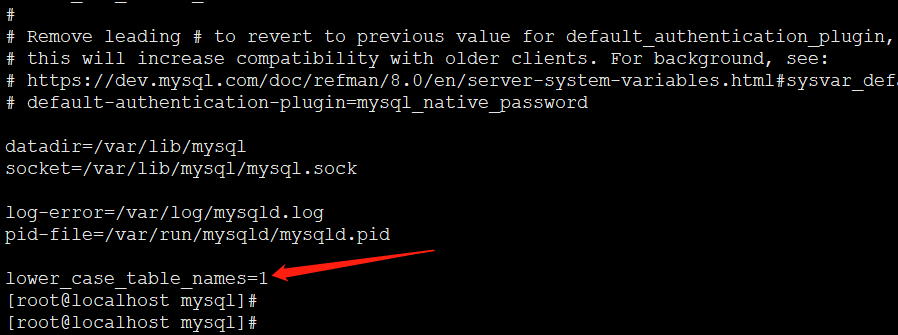


至此，数据库安装完成。

3、配置数据库

3.1 敏感度设置

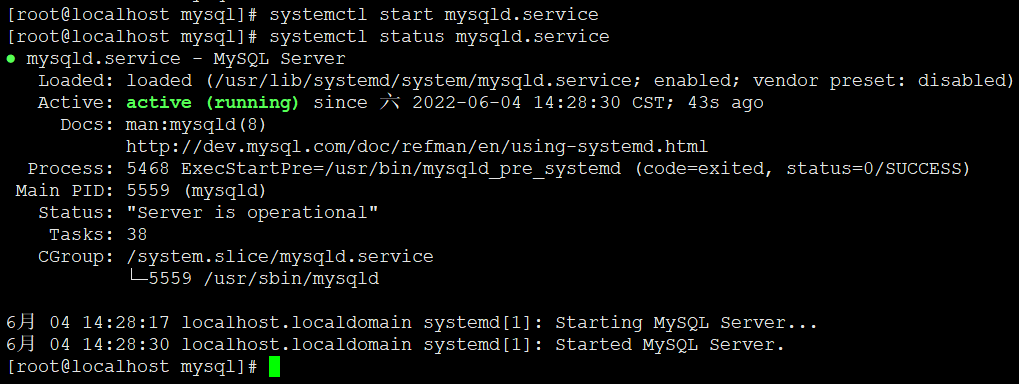
如果有需要可以设置u不区分大小写，vim /etc/my.cnf，lower\_case\_table\_names=1 // 1：不敏感，0：敏感



3.2 启动mysql8 |查看mysql8运行状态

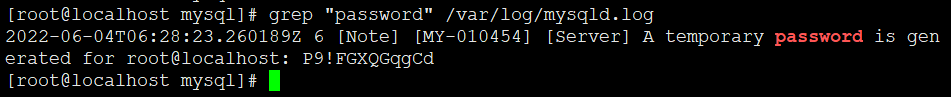
systemctl start mysqld.service

systemctl status mysqld.service

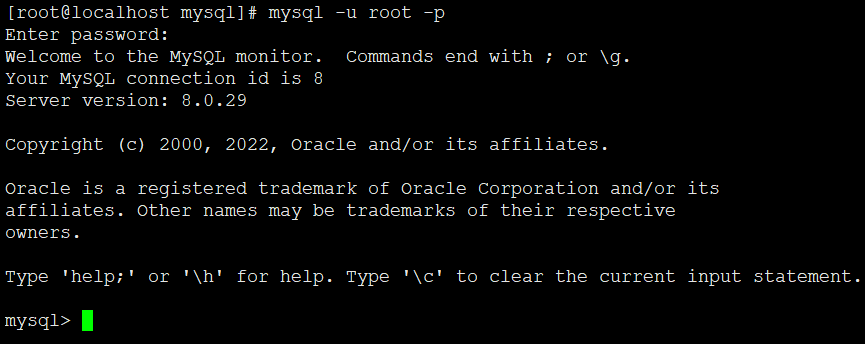


3.3 设置密码

设置密码首先我们要知道root账户的初始密码使用以下命令查看：grep "password" /var/log/mysqld.log



默认密码是：P9!FGXQGqgCd复制该密码，使用以下命令登录数据库：mysql –u root -p

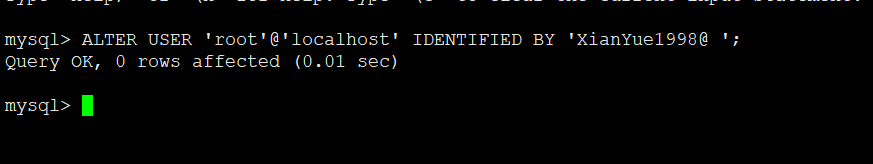


注意：这时候用初始密码登录是无法做任何事的，需要先修改初始密码，修改初始密码如果密码不符合规则过于简单是会报错的，

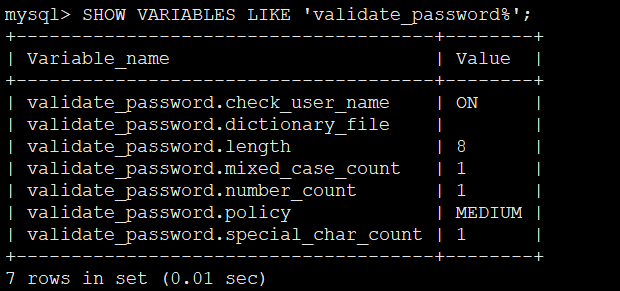
如果想设置个简单的密码，那第一次你还是先设置个比较复杂的，设置完以后再修改密码策略，这时候再次修改为简单的密码即可，

具体操作如下：

修改一个复杂的密码：ALTER USER 'root'@'localhost' IDENTIFIED BY 'XianYue1998@ ';

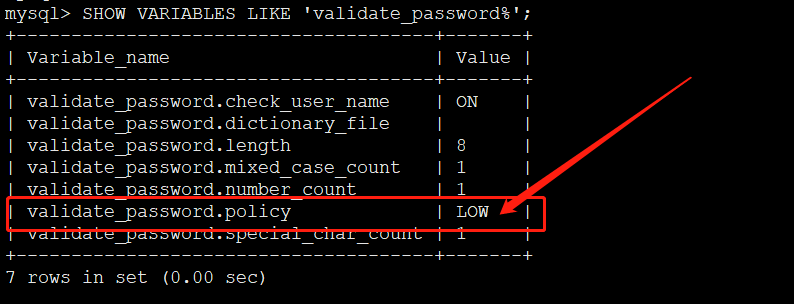


查看密码策略：SHOW VARIABLES LIKE 'validate\_password%';



注意这里mysql8的密码策略是validate\_password.policy，而mysql5的时候密码策略是validate\_password\_policy

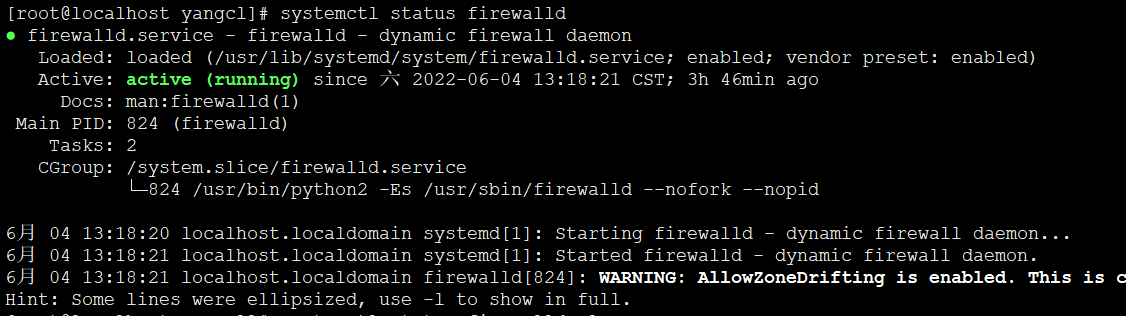
修改策略为低级：set global validate\_password.policy=LOW;



然后就可以用ALTER USER 'root'@'localhost' IDENTIFIED BY '2222 ';再次设置一个简单命令，但不推荐。

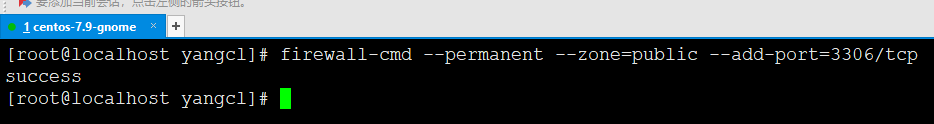
3.4 设置远程访问：开放防火墙3306端口

查看防火墙状态： systemctl status firewalld；开启防火墙：systemctl start firewalld；刷新防火墙：firewall-cmd --reload



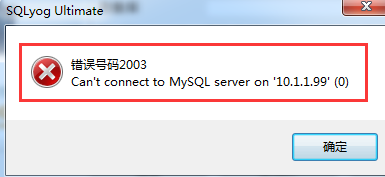
开放防火墙3306端口：

firewall-cmd --permanent --zone=public --add-port=3306/tcp

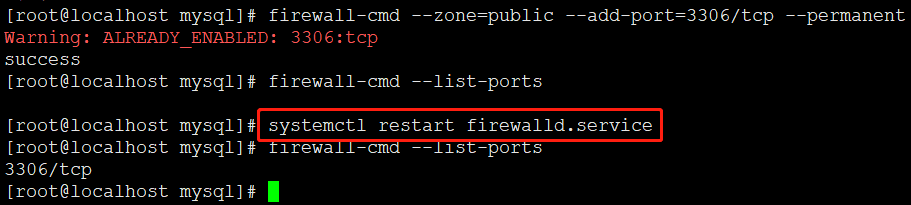


重启防火墙服务：systemctl restart firewalld.service

3.5 异常原因的防火墙3306端口开启失败



如上图所示，mysql服务已经启动，ip地址也是通的，但连不上服务；这说明端口没打开，虽然返回了success。有可能是没有重启防火墙服务导致的。



查看防火墙已绑定端口：firewall-cmd --list-ports；如果为空则需要重启防火墙服务来查看。

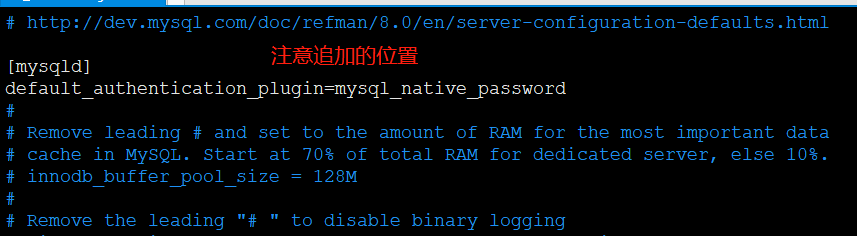
到目前为止，安装全部完成。但如果你用navicat或SQLyog客户端登录可能会报错，因为密码认证方式不支持，

默认是caching\_sha2\_password，可以修改 /etc/my.cnf文件改变密码认证方式为：mysql\_native\_password。

4、配置客户端工具可用

4.1 修改默认配置 如果是docker环境，此步骤省略。

在my.cnf文件的[mysqld]下面加上如下配置：default\_authentication\_plugin=mysql\_native\_password。



4.2 登录mysql修改mysql.user表信息

查询用户信息：select host, user, authentication\_string, plugin from user;

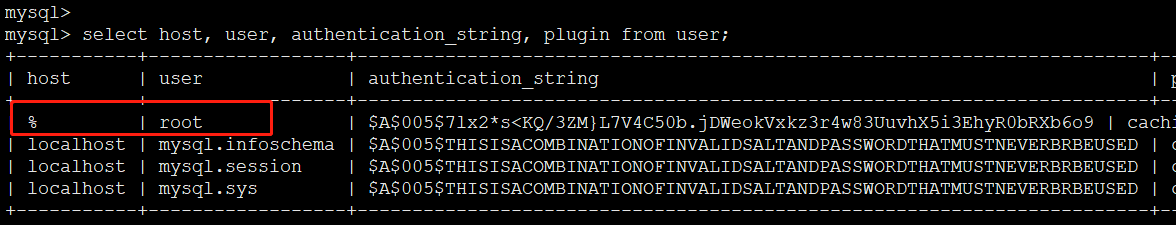


但docker环境中，此处是% | root

上图中我们能看到相关信息，默认localhost，不支持所有ip访问，我们修改所有ip都允许，则把localhost改为【%】。

update user set host='%' where user = 'root';

再次查看：

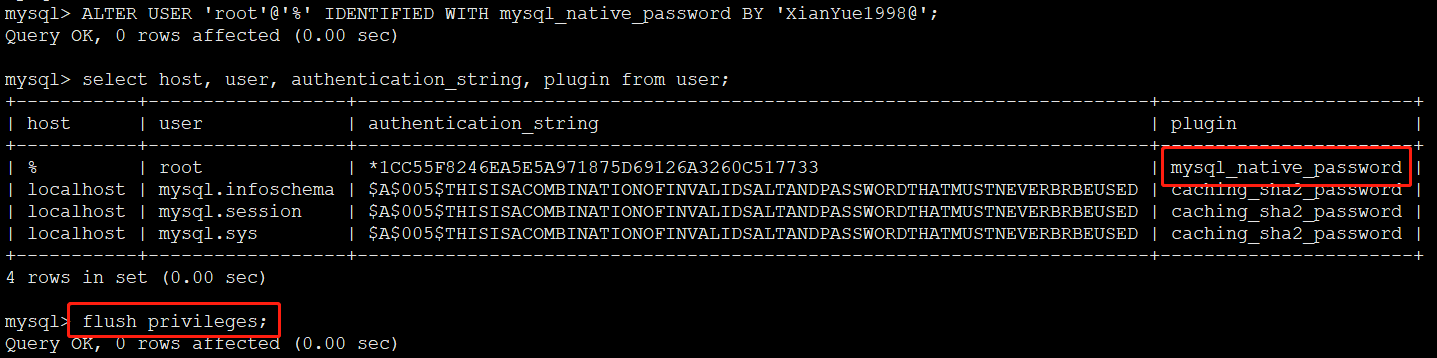


修改加密规则：

ALTER USER 'root'@'%' IDENTIFIED BY 'XianYue1998@' PASSWORD EXPIRE NEVER; 设置为永不过期

修改密码认证方式：ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'XianYue1998@';

刷新权限：flush privileges;



至此全部完成。

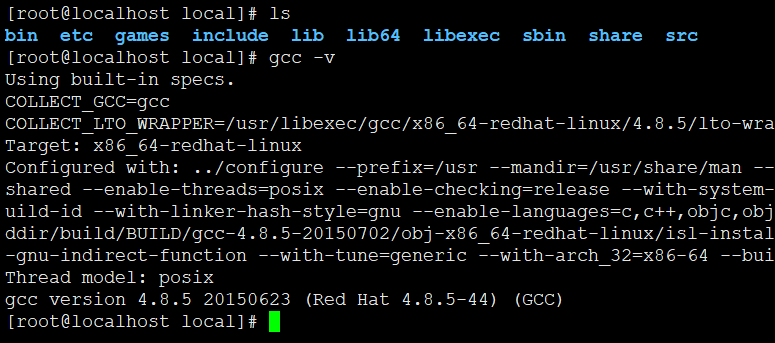
Redis安装和步骤

1、基础安装过程

1.1、安装gcc依赖

由于 redis 是用 C 语言开发，安装之前必先确认是否安装 gcc 环境（gcc -v），如果没有安装，执行以下命令进行安装

[root@localhost local]# yum install -y gcc

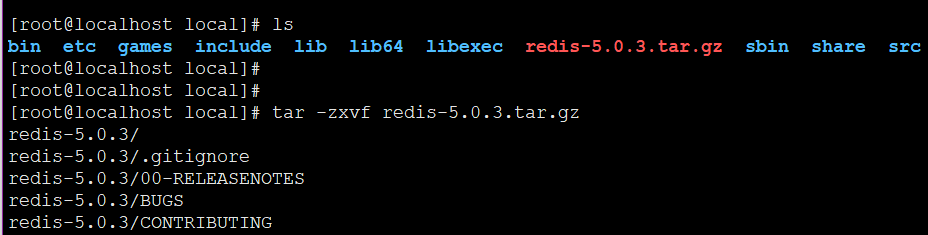


1.2、下载并解压安装包

[root@localhost local]# wget http://download.redis.io/releases/redis-5.0.3.tar.gz



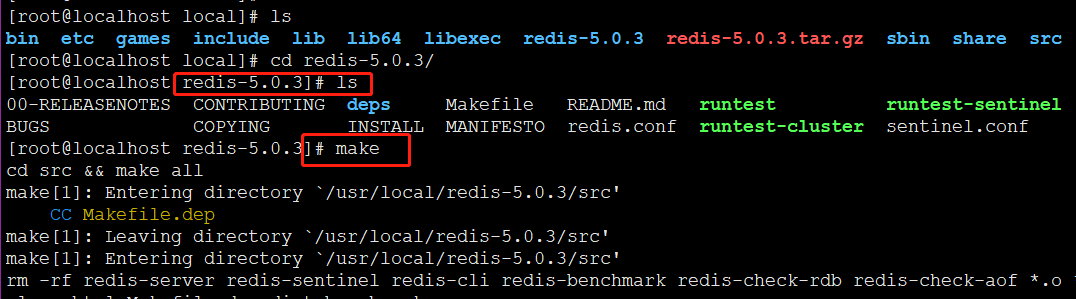
[root@localhost local]# tar -zxvf redis-5.0.3.tar.gz

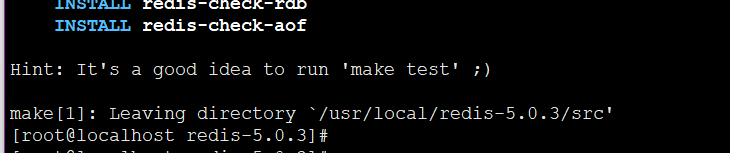


1.3、cd切换到redis解压目录下，执行编译

[root@localhost local]# cd redis-5.0.3

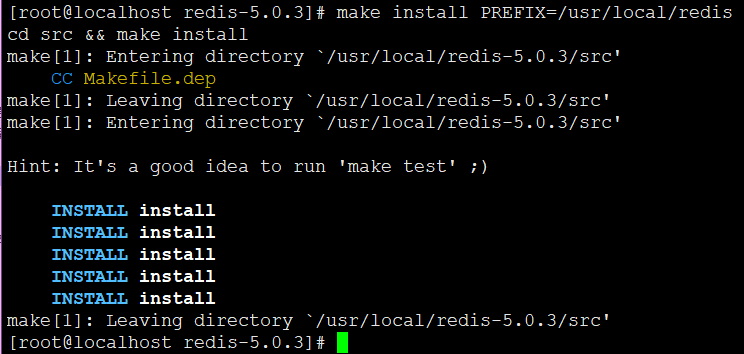
[root@localhost redis-5.0.3]# make





1.4、安装并指定安装目录

[root@localhost redis-5.0.3]# make install PREFIX=/usr/local/redis



1.5.1前台启动 – 一般不用

[root@localhost redis-5.0.3]# cd /usr/local/redis/bin/

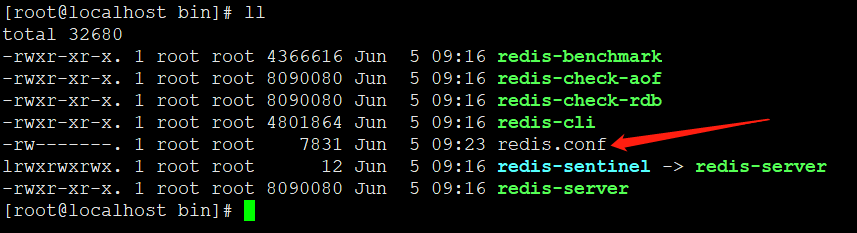
[root@localhost bin]# ./redis-server

1.5.2后台启动

从redis的源码目录中复制redis.conf到redis的安装目录，修改redis.conf文件，把daemonize no改为daemonize yes。

redis.conf是一个核心配置文件，一般都是在Windows上改好了再拷贝到Linux中。

[root@localhost bin]# cp /home/yangcl/redis.conf /usr/local/redis/bin/



[root@localhost bin]# vim redis.conf

修改redis.conf文件, 解除绑定本地ip, 同时关闭保护模式

注释掉 bind 127.0.0.1bash

protected-mode yes 改成 protected-mode no

1.6 防火墙放行redis默认的6379端口

1）防火墙开启端口

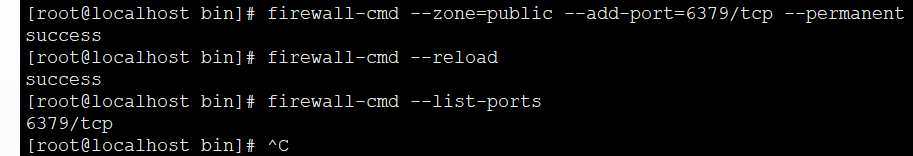
firewall-cmd --zone=public --add-port=6379/tcp --permanent

2）重启防火墙服务

firewall-cmd –reload

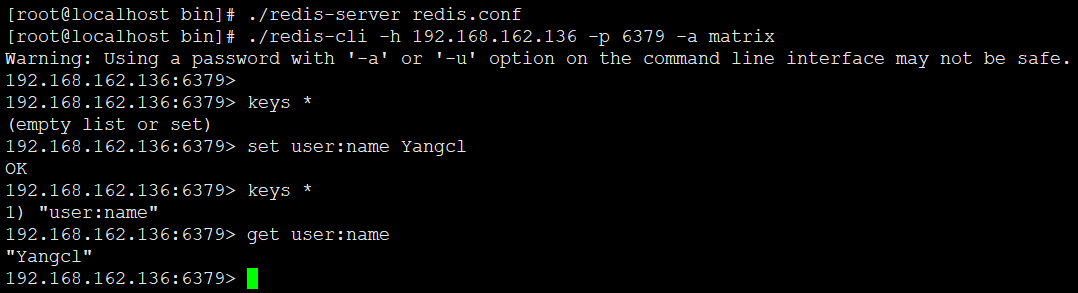
3）检查端口是否开启

firewall-cmd --reload



4）后台启动

[root@localhost bin]# ./redis-server redis.conf



至此redis单机模式基础安装完成，已经可以使用。

1.7 设置开机启动

添加开机启动服务，新增redis.service文件，注意：ExecStart配置成自己的路径

[root@localhost bin]# vim /etc/systemd/system/redis.service

文件中写入如下代码：

[Unit]

Description=redis-server

After=network.target

[Service]

Type=forking

ExecStart=/usr/local/redis/bin/redis-server /usr/local/redis/bin/redis.conf

PrivateTmp=true

[Install]

WantedBy=multi-user.target

创建 redis 命令软链接

[root@localhost ~]# ln -s /usr/local/redis/bin/redis-cli /usr/bin/redis

3、为什么linux安装程序 都要放到/usr/local目录下

Linux 的软件安装目录是也是有讲究的，理解这一点，在对系统管理是有益的

/usr：系统级的目录，可以理解为C:/Windows/，/usr/lib理解为C:/Windows/System32。

/usr/local：用户级的程序目录，可以理解为C:/Progrem Files/。用户自己编译的软件默认会安装到这个目录下。

/opt：用户级的程序目录，可以理解为D:/Software，opt有可选的意思，这里可以用于放置第三方大型软件（或游戏），当你不需要时，

直接rm -rf掉即可。在硬盘容量不够时，也可将/opt单独挂载到其他磁盘上使用。

源码放哪里？

/usr/src：系统级的源码目录。

/usr/local/src：用户级的源码目录.

4、关于redis.conf配置文件

https://blog.csdn.net/weixin\_44906977/article/details/118994047

https://blog.csdn.net/sinat\_25207295/article/details/117925174 ./redis-cli -h host -p port -a password

http://www.javashuo.com/article/p-ozxwfyff-kx.html

开启端口，防火墙刷新。。。

RocketMq 双M-S配置过程

1、移除centos自带的openJdk

1.1查看JDK安装版本

[yangcl@b-master home]$ java -version

openjdk version "1.8.0\_262"

OpenJDK Runtime Environment (build 1.8.0\_262-b10)

OpenJDK 64-Bit Server VM (build 25.262-b10, mixed mode)

1.2查找OpenJDK安装包

[yangcl@b-master home]$ rpm -qa | grep openjdk

java-1.8.0-openjdk-headless-1.8.0.262.b10-1.el7.x86\_64

java-1.7.0-openjdk-headless-1.7.0.261-2.6.22.2.el7\_8.x86\_64

java-1.8.0-openjdk-1.8.0.262.b10-1.el7.x86\_64

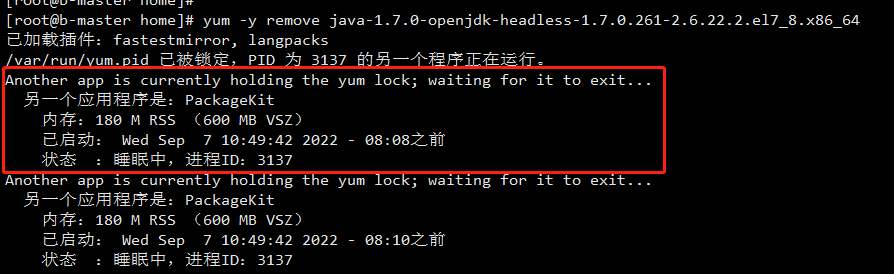
java-1.7.0-openjdk-1.7.0.261-2.6.22.2.el7\_8.x86\_64

[yangcl@b-master home]$

1.3卸载OpenJDK安装包

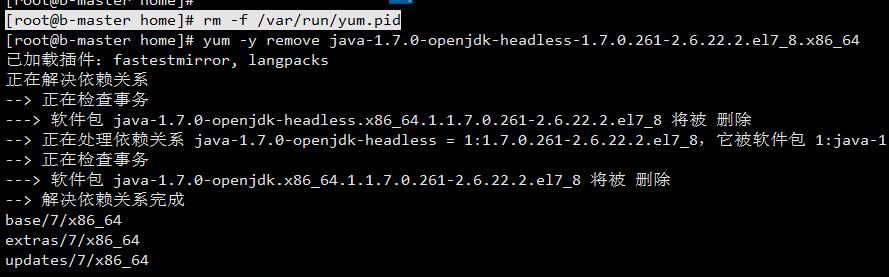
切换到root权限后：

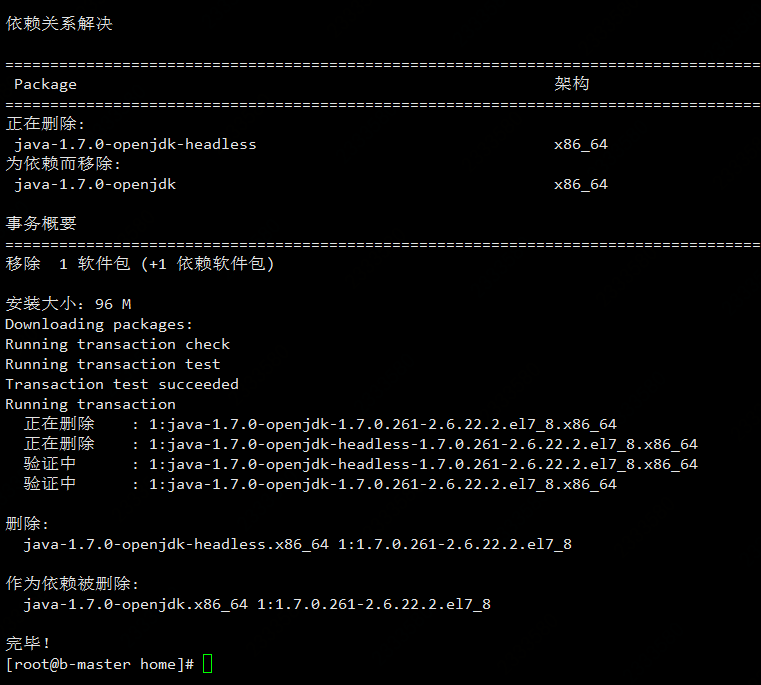
[**root**@b-master home]# yum -y remove java-1.7.0-openjdk-headless-1.7.0.261-2.6.22.2.el7\_8.x86\_64



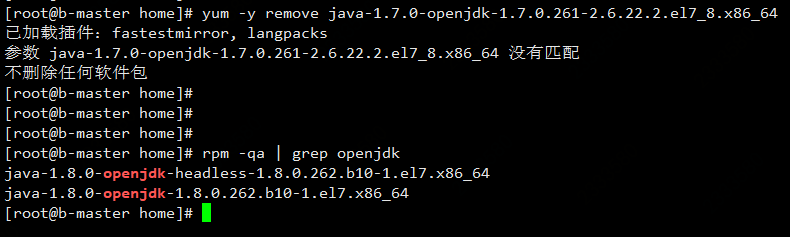
原因：yum在锁定状态中；可以通过强制关掉yum进程解决此问题：

[root@b-master home]# rm -f /var/run/yum.pid 重新执行yum –y remove命令：

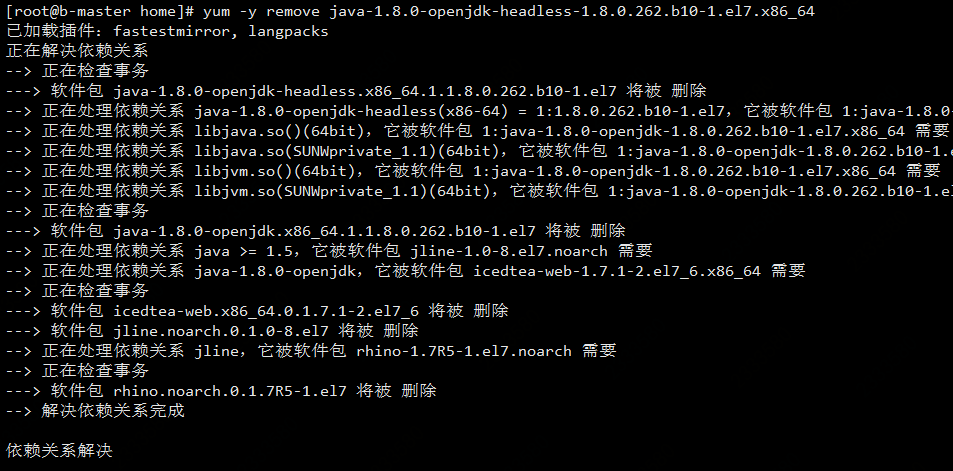


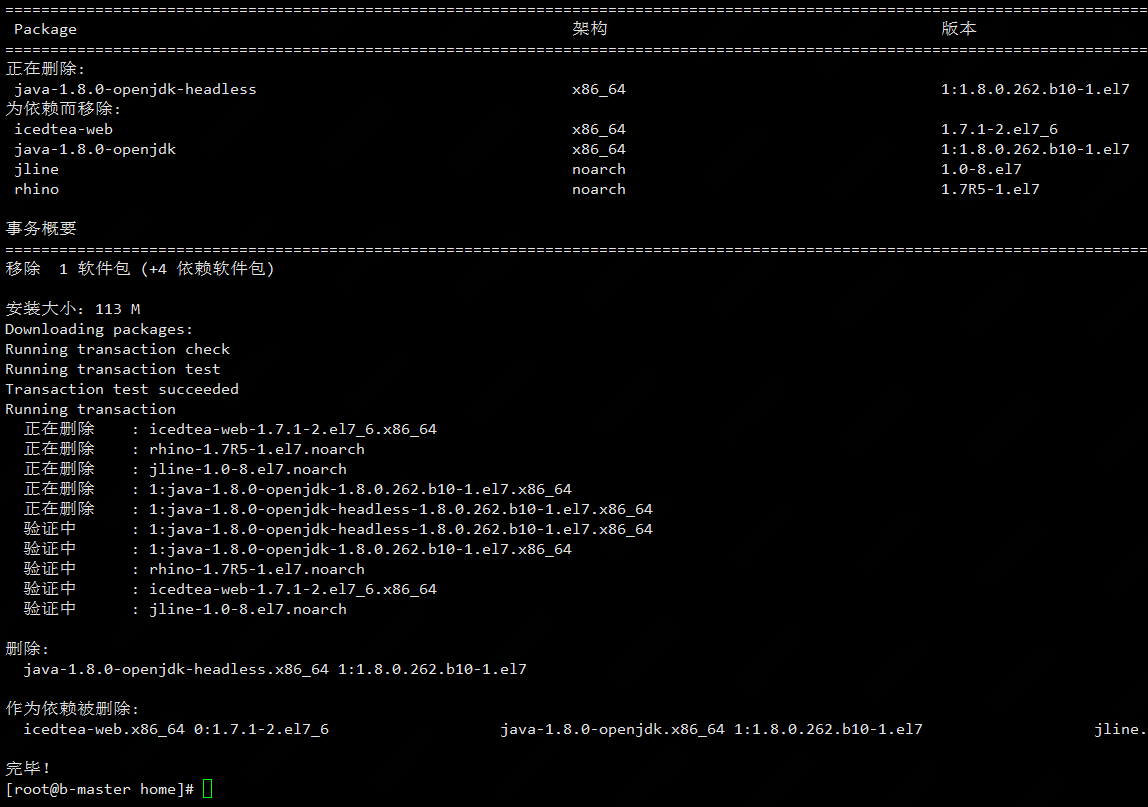


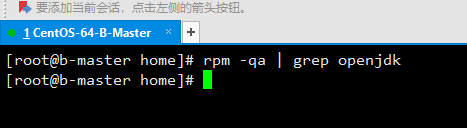
移除headless.x86\_64后，继续删除：java-1.7.0-openjdk-1.7.0.261-2.6.22.2.el7\_8.x86\_64



继续删除openJDK1.8







1.4准备安装JDK1.8

访问[https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html 此处有jdk1.8](https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html%20此处有jdk1.8)所有版本的信息

下载：jdk-8u202-linux-x64.rpm到本地。

使用rz命令上传，虚拟机传不了。

1.5 解决虚拟机传不了文件的问题

即：解决open-vm-tools无法复制粘贴文件问题

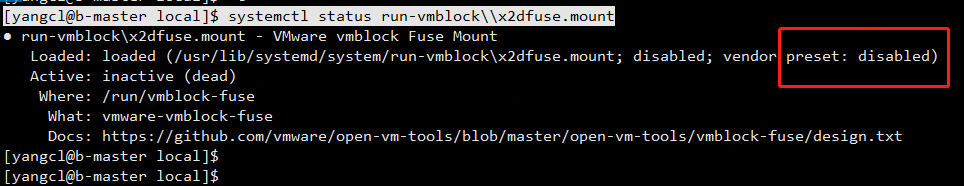


vmware的增强工具是：open-vm-tools和open-vm-tools-desktop。复制字符没问题，就是复制文件和拖拽文件有问题。

拖拽和复制、粘贴文件，需要挂载一个特殊的文件系统：vmblock-fuse，先查看这个文件系统挂载服务的情况：

切换到 usr/local文件夹下：

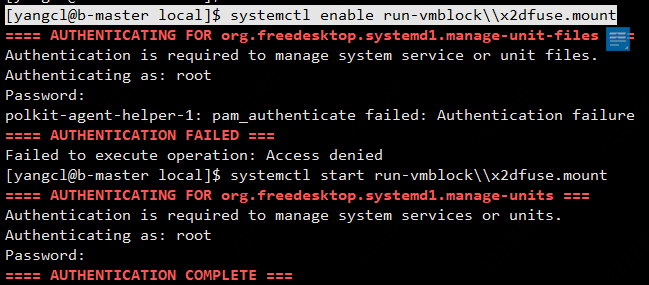
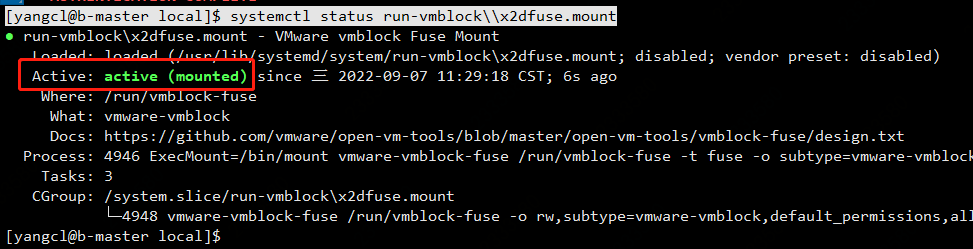
[yangcl@b-master local]$ systemctl status run-vmblock\\x2dfuse.mount



发现这个服务居然是inactive状态并且是disabled状态，意味着：没有启动，并且重启后也不会自启。

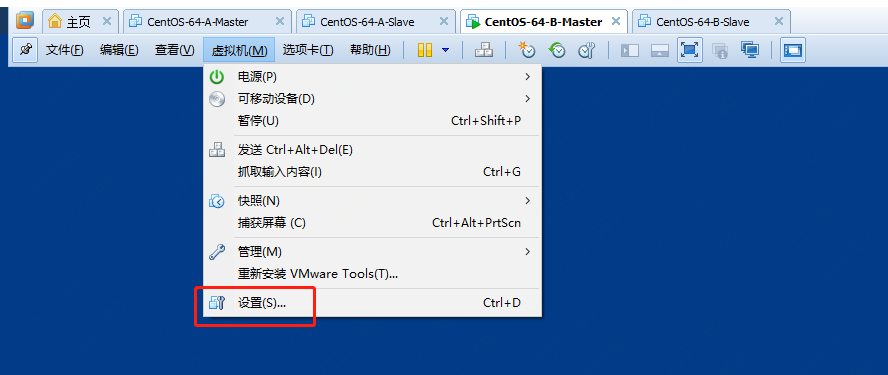
1. 开机自启：systemctl enable run-vmblock\\x2dfuse.mount

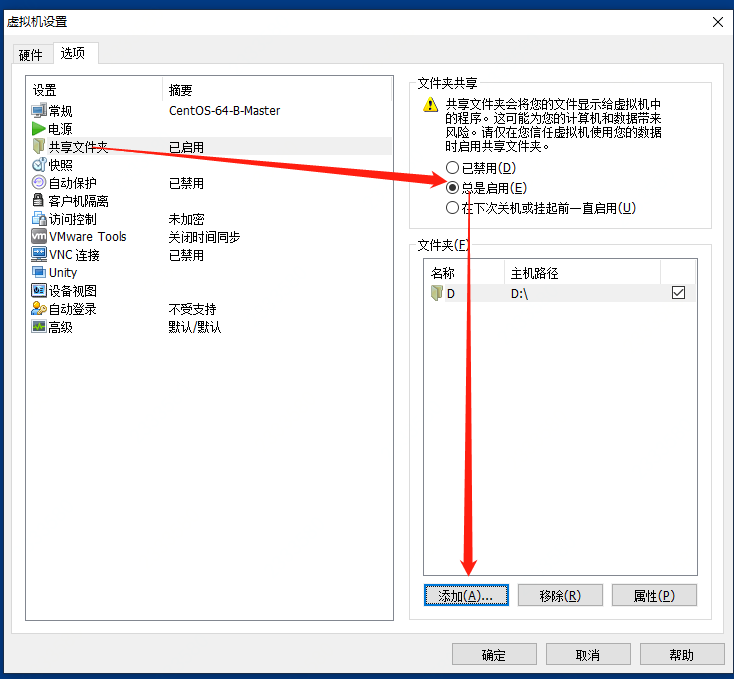
2. 启动服务：systemctl start run-vmblock\\x2dfuse.mount

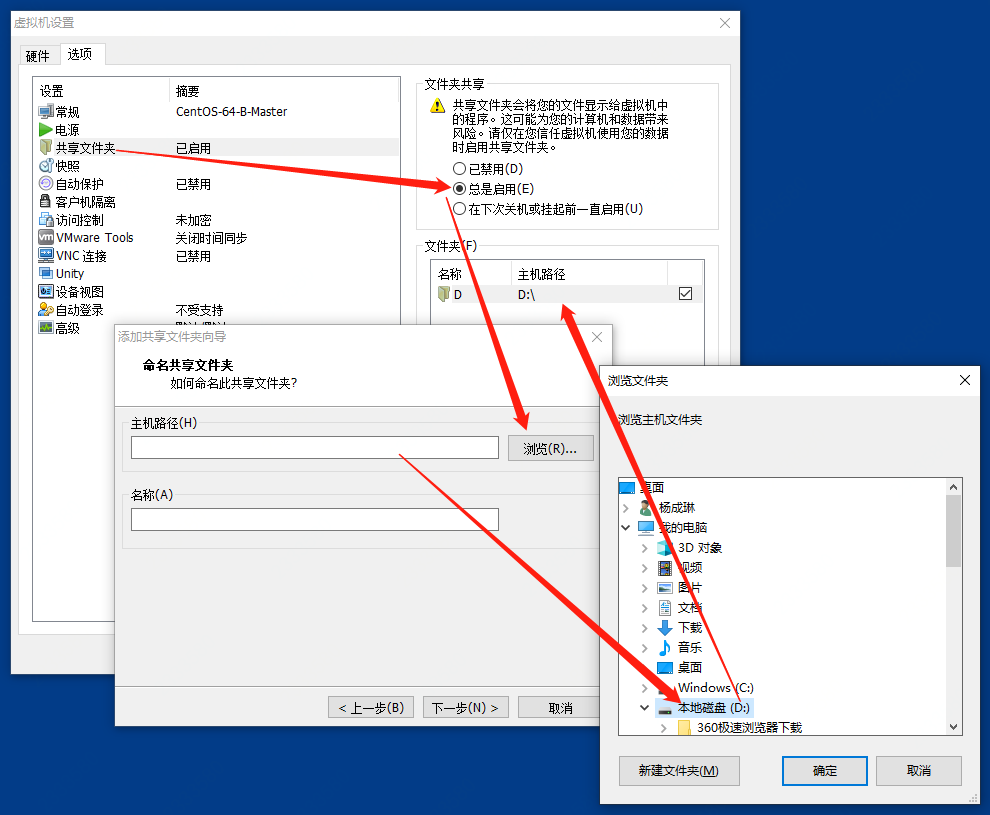
 

上面的步骤感觉在Linux最小化安装的时候，没有效果。

但是此时仍然无法复制粘贴



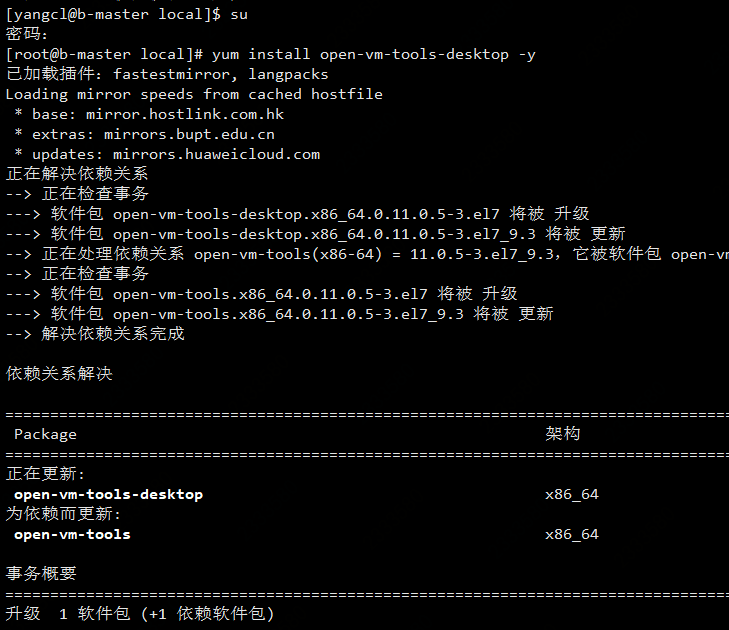




设置完成共享文件夹后，执行下面的命令：

yum install open-vm-tools-desktop -y

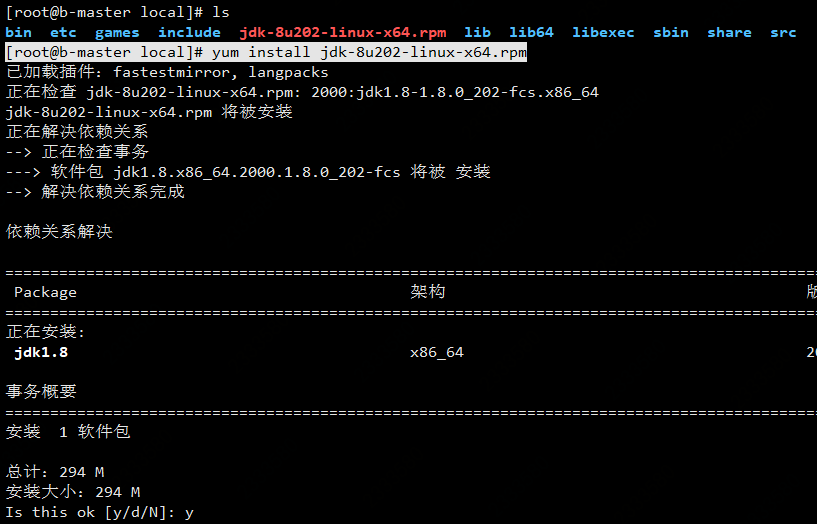
yum reinstall open-vm-tools -y

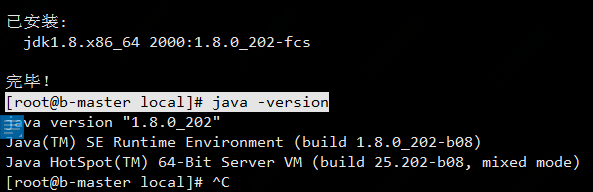


再次使用rz指令上传文件；如果还失败切换成root用户来传即可。

1.6 安装JDK1.8

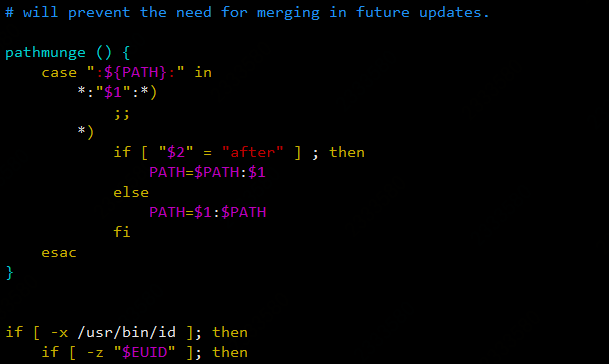
[root@b-master local]# yum install jdk-8u202-linux-x64.rpm





1.7 配置环境变量

[root@b-master local]# vim /etc/profile



文件末尾追加如下内容：

JAVA\_HOME=/usr/java/jdk1.8.0\_202-amd64

PATH=$JAVA\_HOME/bin:$PATH

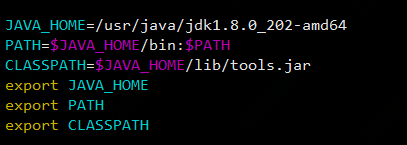
CLASSPATH=$JAVA\_HOME/lib/tools.jar

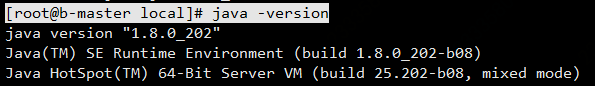
export JAVA\_HOME

export PATH

export CLASSPATH

查看结果：





切记=号后面不能有空格，否则会出现如下情况：

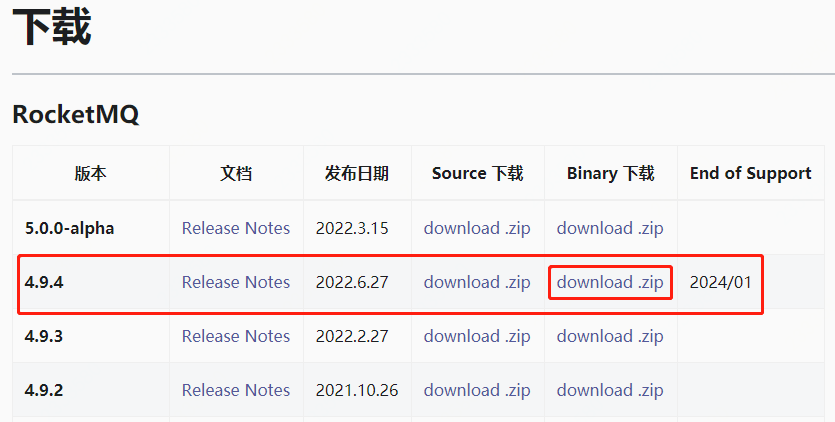
[root@b-master yangcl]# source /etc/profile

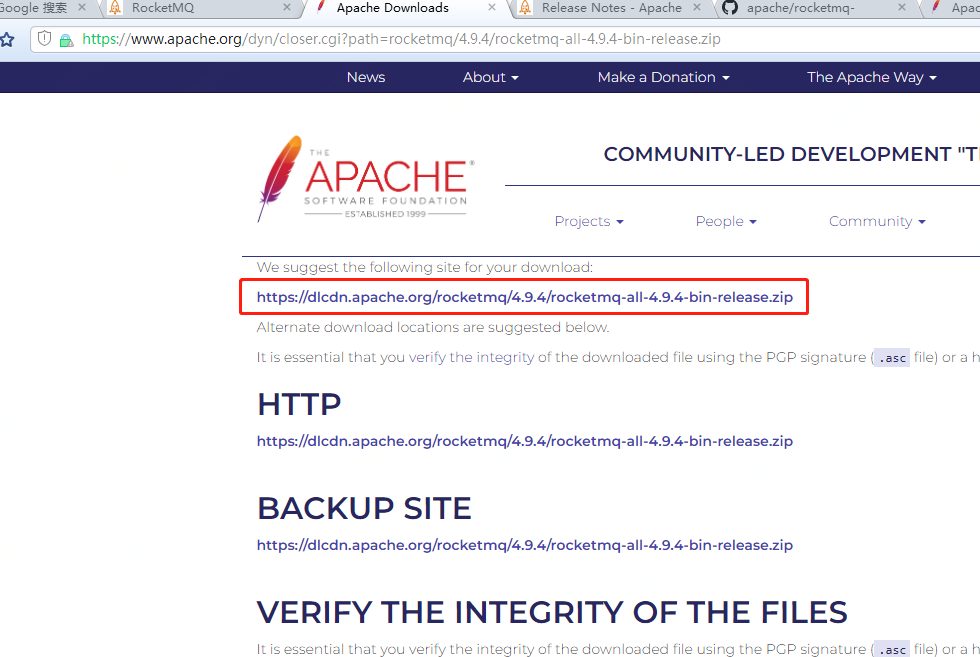
bash: /usr/java/jdk1.8.0\_202-amd64: 是一个目录

2、安装RocketMq

2.1 上传RocketMq同时解压

访问：<https://rocketmq.apache.org/download/>

 点击download.zip进入阿帕奇官网



[root@b-master soft]# unzip rocketmq-all-4.9.4-bin-release.zip 先解压。

[root@b-master local]# mv /usr/soft/rocketmq-all-4.9.4-bin-release /usr/local/rocketmq-4.9.4 移动到新文件夹

2.2 简单测试可执行性 – NameServer

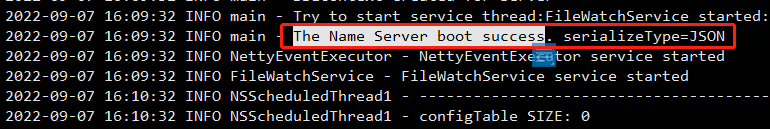
[root@b-master rocketmq-4.9.4]# cd bin/ 进入到bin目录下



[root@b-master bin]# nohup sh mqnamesrv & “&”是必须写的



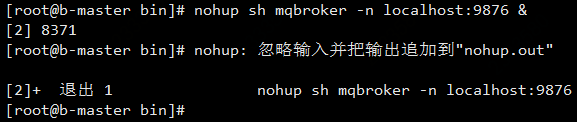
[root@b-master bin]# tail -200f ~/logs/rocketmqlogs/namesrv.log 这里是默认启动的日志目录，默认在home下。



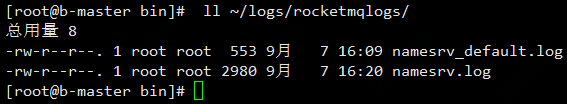
**NameServer启动成功**。

2.3 简单测试可执行性 – Broker

[root@b-master bin]# nohup sh mqbroker -n localhost:9876 & 本机端口默认9876



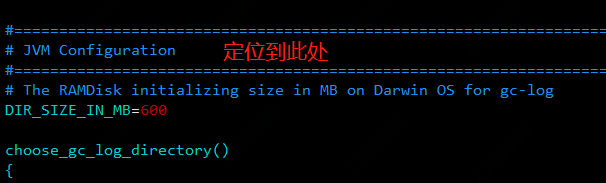
ll ~/logs/rocketmqlogs/



启动失败！因为默认rocket分配的内存较大，所以需要修改默认分配的内存值。

修改：runbroker.sh

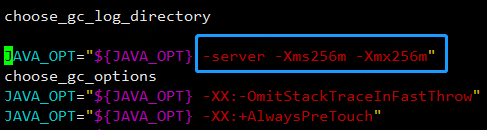
[root@b-master bin]# vim runbroker.sh





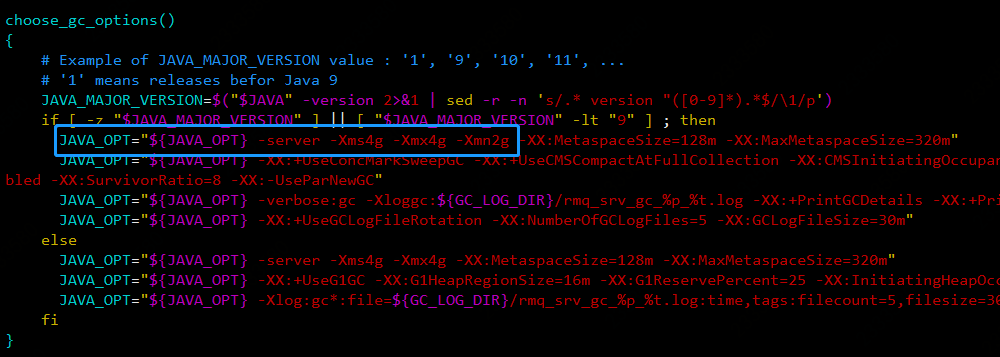
8G是官方建议的配置，也就是说生产环境一台RocketMq消息队列服务器的虚机至少应该维持在8G以上，12G为合适的区间。2M-2S的配置，共需要内存48G。

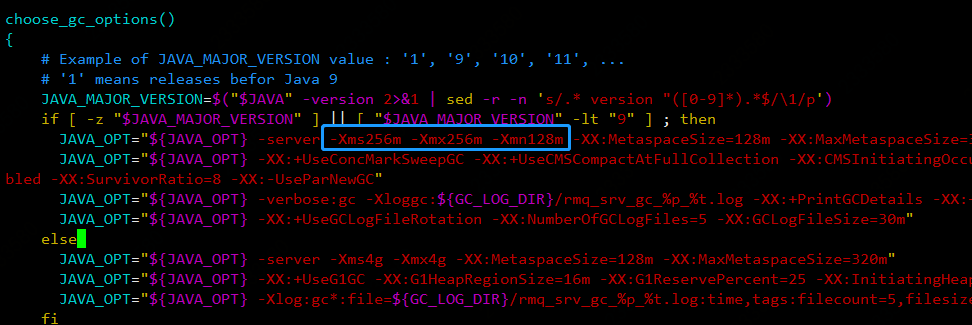
这里改成256M，仅用于测试。



修改：runserver.sh

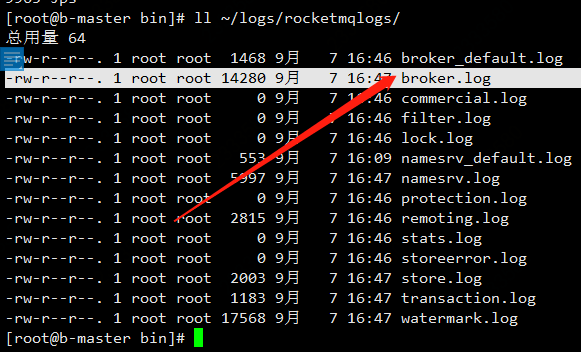
[root@b-master bin]# vim runserver.sh

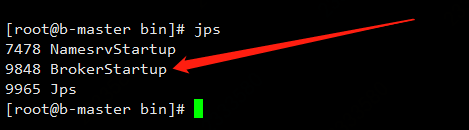




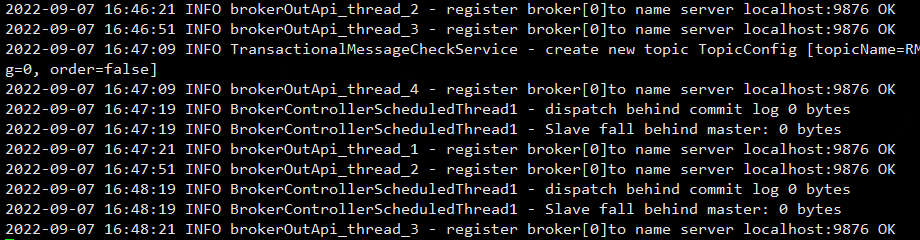
再次启动broker

[root@b-master bin]# nohup sh mqbroker -n localhost:9876 &





[root@b-master bin]# tail -200f ~/logs/rocketmqlogs/broker.log



broker启动成功。

2.4 关闭命令

sh bin/mqshutdown namesrv

sh bin/mqshutdown broker

2.5 RocketMq自身消息收发测试

进入：/usr/local/rocketmq-4.9.4/bin此目录。

启动生产者

设置临时环境变量：

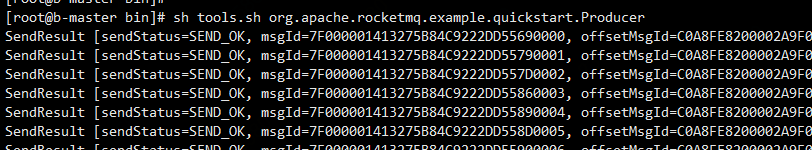
[root@b-master bin]# export NAMESRV\_ADDR=localhost:9876

调用自带测试类，生产者来发送消息：

[root@b-master bin]# sh tools.sh org.apache.rocketmq.example.quickstart.Producer 发送消息内容如下：

SendResult [sendStatus=SEND\_OK, msgId=7F000001413275B84C9222DD55690000,

offsetMsgId=C0A8FE8200002A9F000000000005DB24, messageQueue=MessageQueue [topic=TopicTest, brokerName=b-master, queueId=2], queueOffset=500]



启动消费者（启动后是不会停的，会一直处于监听状态，生产者发送消息就会被其接收）

[root@b-master bin]# export NAMESRV\_ADDR=localhost:9876

[root@b-master bin]# sh tools.sh org.apache.rocketmq.example.quickstart.Consumer

消费者返回消息如下：

ConsumeMessageThread\_please\_rename\_unique\_group\_name\_4\_1 Receive New Messages:

[MessageExt [brokerName=**b-master**, queueId=2, storeSize=192, queueOffset=747, sysFlag=0, bornTimestamp=1662546533635,

bornHost=/192.168.254.130:53798, storeTimestamp=1662546533635, storeHost=/192.168.254.130:10911,

msgId=C0A8FE8200002A9F000000000008BFB6, commitLogOffset=573366, bodyCRC=1209384652,

reconsumeTimes=0, preparedTransactionOffset=0,

toString()=Message{topic='TopicTest', flag=0, properties={MIN\_OFFSET=0, MAX\_OFFSET=750,

CONSUME\_START\_TIME=1662546728681, UNIQ\_KEY=7F000001413275B84C9222DD610303DC,

CLUSTER=DefaultCluster, TAGS=TagA}, body=[72, 101, 108, 108, 111, 32, 82, 111, 99, 107, 101, 116, 77, 81, 32, 57, 56, 56], transactionId='null'}]]

brokerName=**b-master** 是默认的Linux主机名

3、RocketMq集群配置与相关理论基础

3.1 支持的集群模式

1）单Master模式

这种方式风险较大，一旦Broker重启或者宕机时，会导致整个服务不可用。不建议线上环境使用，可以用于本地测试。

2）多Master模式

一个集群无Slave，全是Master，例如2个Master或者3个Master，这种模式的优缺点如下：

优点：配置简单，单个Master宕机或重启维护对应用无影响，在磁盘配置为RAID10时，即使机器宕机不可恢复情况下，

由于RAID10磁盘非常可靠，消息也不会丢（异步刷盘丢失少量消息，同步刷盘一条不丢），**性能最高**。

缺点：单台机器宕机期间，这台机器上未被消费的消息在机器恢复之前不可订阅，消息实时性会受到影响。

3）多Master多Slave模式（异步）

每个Master配置一个Slave，有多对Master-Slave，HA采用异步复制方式，主备有短暂消息延迟（毫秒级），这种模式的优缺点如下：

优点：即使磁盘损坏，消息丢失的非常少，且消息实时性不会受影响，同时Master宕机后，消费者仍然可以从Slave消费，

而且此过程对应用透明，不需要人工干预，性能同多Master模式几乎一样。

缺点：Master宕机，磁盘损坏情况下会丢失少量消息。

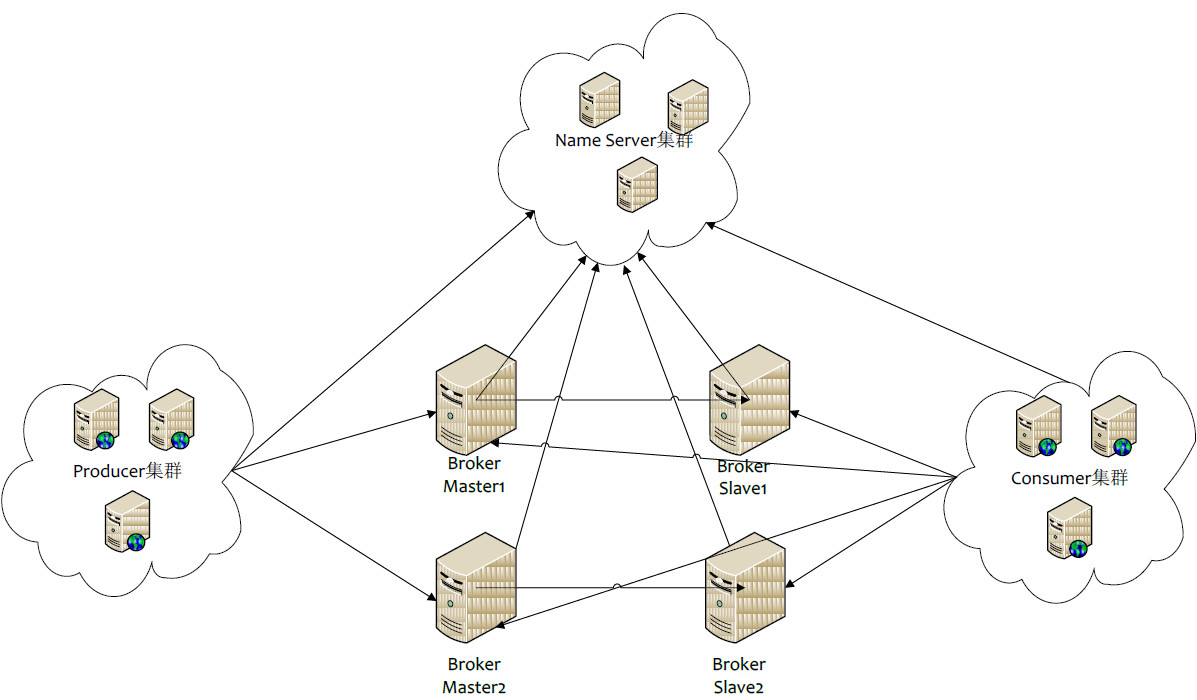
4）多Master多Slave模式（同步）

每个Master配置一个Slave，有多对Master-Slave，HA采用同步双写方式，即只有主备都写成功，才向应用返回成功，这种模式的优缺点如下：

优点：数据与服务都无单点故障，Master宕机情况下，消息无延迟，服务可用性与数据可用性都非常高。

缺点：性能比异步复制模式略低（大约低10%左右），发送单个消息的RT会略高，且目前版本在主节点宕机后，备机不能自动切换为主机。

3.2 nM-ss架构



通过配置文件中的brokerClusterName 来划分一个集群，比如：brokerClusterName=mq-cluster-order；通过brokerName划分一个主从分组，

比如：brokerName=broker-a；通过brokerId来区分主从设置brokerId=0，0表示 Master，大于0表示Slave。

在一个主从集群中，Producer发送的消息通常会由Master接收，完成写操作，然后以“同步”或“异步”的方式来给到自己的Slave节点；Consumer

作为消息的消费者，在主从集群中通常会从Slave节点中读取自己订阅的信息，完成读操作。无论是“同步”或“异步”，M-s主从集群要达成的目的都是：读写分离。

3.3 RocketMq角色

Producer：消息的发送者；举例：发信者

Consumer：消息接收者；举例：收信者

Broker：暂存和传输消息；举例：邮局

NameServer：管理Broker；举例：各个邮局的管理机构

Topic：区分消息的种类；一个发送者可以发送消息给一个或者多个Topic；一个消息的接收者可以订阅一个或者多个Topic消息

Message Queue：相当于是Topic的分区；用于并行发送和接收消息

4、双M-S配置过程

4.1 配置hosts 进入每台Linux

vim /etc/hosts

# nameserver

192.168.25.128 rocketmq.am.nameserver.128

192.168.25.129 rocketmq.as.nameserver.129

192.168.25.130 rocketmq.bm.nameserver.130

192.168.25.131 rocketmq.bs.nameserver.131

# broker

192.168.25.128 rocketmq.a.master.128

192.168.25.129 rocketmq.a.slave.129

192.168.25.130 rocketmq.b.master.130

192.168.25.131 rocketmq.b.slave.130

配置完成后, 重启网卡：

systemctl restart network

4.2 防火墙配置

宿主机需要远程访问虚拟机的rocketmq服务和web服务，需要开放相关的端口号，测试环境简单粗暴的方式是直接关闭防火墙。

# 关闭防火墙

systemctl stop firewalld.service

# 查看防火墙的状态

firewall-cmd --state

# 禁止firewall开机启动

systemctl disable firewalld.service

但正式环境为了安全只开放特定的端口号，RocketMQ默认使用3个端口：9876 、10911 、11011 。如果防火墙没有关闭的话，

那么防火墙就必须开放这些端口：

·nameserver默认使用9876端口

·master默认使用10911端口

·slave默认使用11011端口

执行以下命令，以达到开发指定端口的目的：

# 开放name server默认端口

firewall-cmd --remove-port=9876/tcp --permanent

# 开放master默认端口

firewall-cmd --remove-port=10911/tcp --permanent

# 开放slave默认端口 (当前集群模式可不开启)

firewall-cmd --remove-port=11011/tcp --permanent

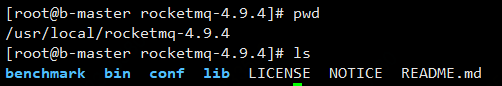
# 重启防火墙

firewall-cmd --reload

4.3 RocketMq环境变量配置

为了方便使用rocketMq命令，不用每次都进入到其bin目录之下去操作，所以此步设置环境变量。

vim /etc/profile 在profile文件的末尾加入如下命令：

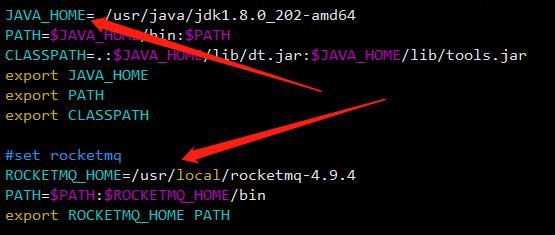


#set rocketmq

ROCKETMQ\_HOME=/usr/local/rocketmq-4.9.4

PATH=$PATH:$ROCKETMQ\_HOME/bin

export ROCKETMQ\_HOME PATH



输入:wq! 保存并退出， 并使得配置立刻生效：

source /etc/profile

4.4 创建消息队列持久化目录

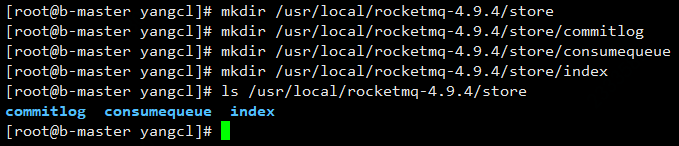
rocket mq 默认的消息存储位置在 Linux的home目录下；修改后消息将自动持久化到如下目录：

mkdir /usr/local/rocketmq-4.9.4/store

mkdir /usr/local/rocketmq-4.9.4/store/commitlog

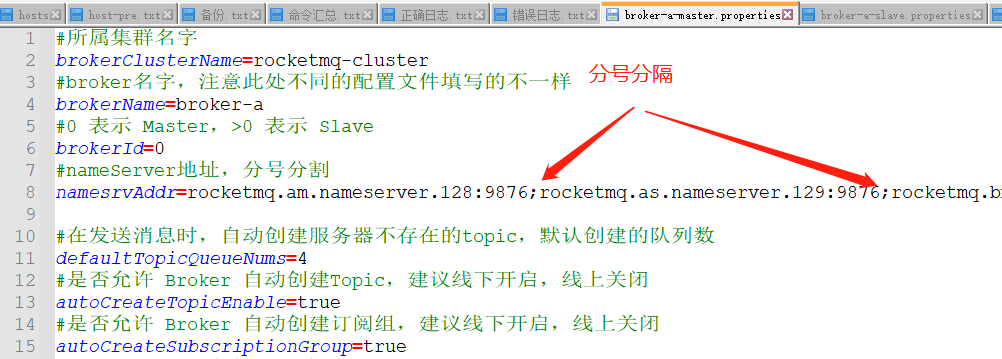
mkdir /usr/local/rocketmq-4.9.4/store/consumequeue

mkdir /usr/local/rocketmq-4.9.4/store/index



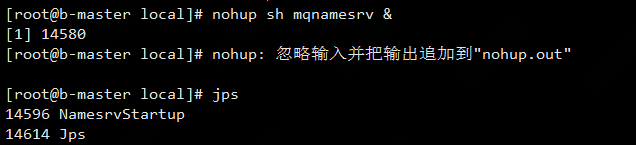
4.5 conf文件夹下的配置文件

/usr/local/rocketmq-4.9.4/conf/2m-2s-sync 下单配置文件不动，我们自己传一份（同此文档目录下【rocket-mq配置文件】）。



4.6 启动各个服务器的NameServer

[root@b-master local]# nohup sh mqnamesrv &



因为配置了环境变量，所以这里任意位置均可执行此命令。

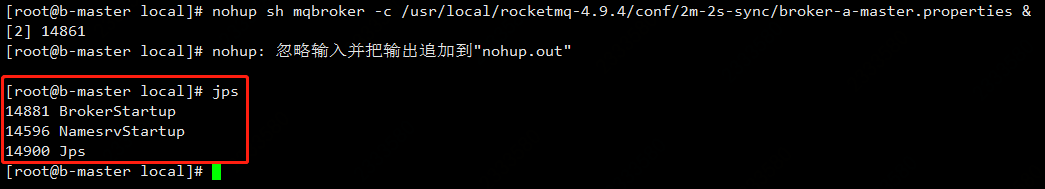
4.7 启动各个服务的Broker

nohup sh mqbroker -c /usr/local/rocketmq-4.9.4/conf/2m-2s-sync/broker-a-master.properties &

nohup sh mqbroker -c /usr/local/rocketmq-4.9.4/conf/2m-2s-sync/broker-a-slave.properties &

nohup sh mqbroker -c /usr/local/rocketmq-4.9.4/conf/2m-2s-sync/broker-b-master.properties &

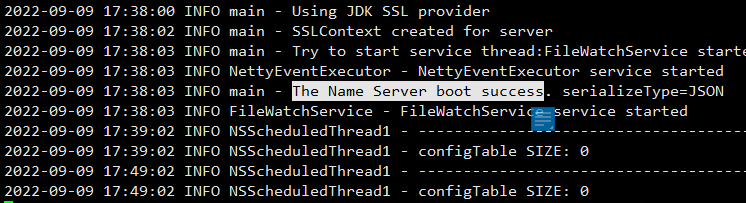
nohup sh mqbroker -c /usr/local/rocketmq-4.9.4/conf/2m-2s-sync/broker-b-slave.properties &



4.8 查看日志

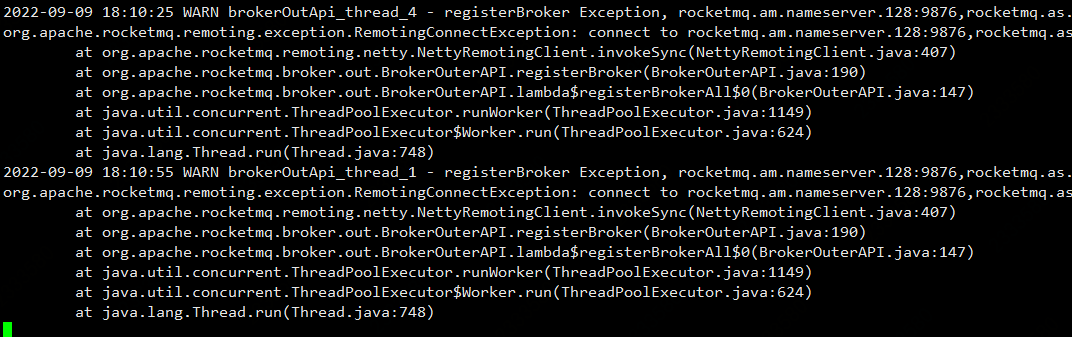
# 查看nameServer日志

tail -20f ~/logs/rocketmqlogs/namesrv.log



# 查看broker日志

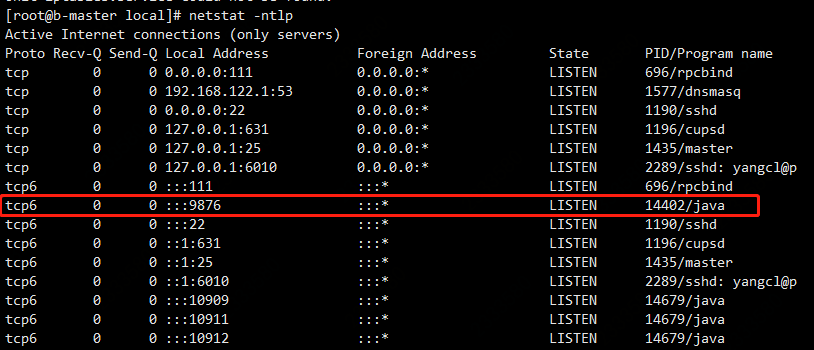
tail -50f ~/logs/rocketmqlogs/broker.log



发现broker连不上name server，看看后台进程中是否有开启相应的服务，如果有9876端口，那就说明是正常的：

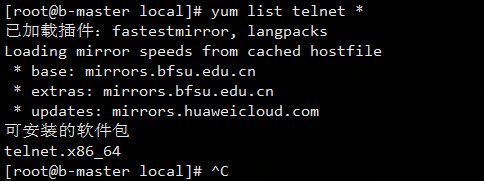
输入命令：

netstat -ntlp



安装 telnet 工具：

[root@b-master local]# yum list telnet \* 查看telnet相关的安装包



[root@b-master local]# yum install telnet-server 安装telnet服务

[root@b-master local]# yum install telnet.\* 安装telnet客户端

执行命令：

[root@b-master local]# telnet 192.168.254.129 9876

Trying 192.168.254.129...

Connected to 192.168.254.129.

Escape character is '^]'.

Connection closed by foreign host.

显示是通的。

**最终发现，原因出在name server地址上**，应该用英文分号分隔，不是用逗号。

#nameServer地址，分号分割：**;**

namesrvAddr=rocketmq.am.nameserver.128:9876**;**rocketmq.as.nameserver.129:9876**;**rocketmq.bm.nameserver.130:9876**;**rocketmq.bs.nameserver.131:9876

4.9 关闭rocketMq（已配置环境变量）

[root@b-master local]# sh mqshutdown namesrv

[root@b-master local]# sh mqshutdown broker

4.10 iptables相关

yum install iptables-services

systemctl status iptables.service

systemctl stop iptables.service

systemctl disable iptables.service

4.11 安装netstat

yum install net-tools