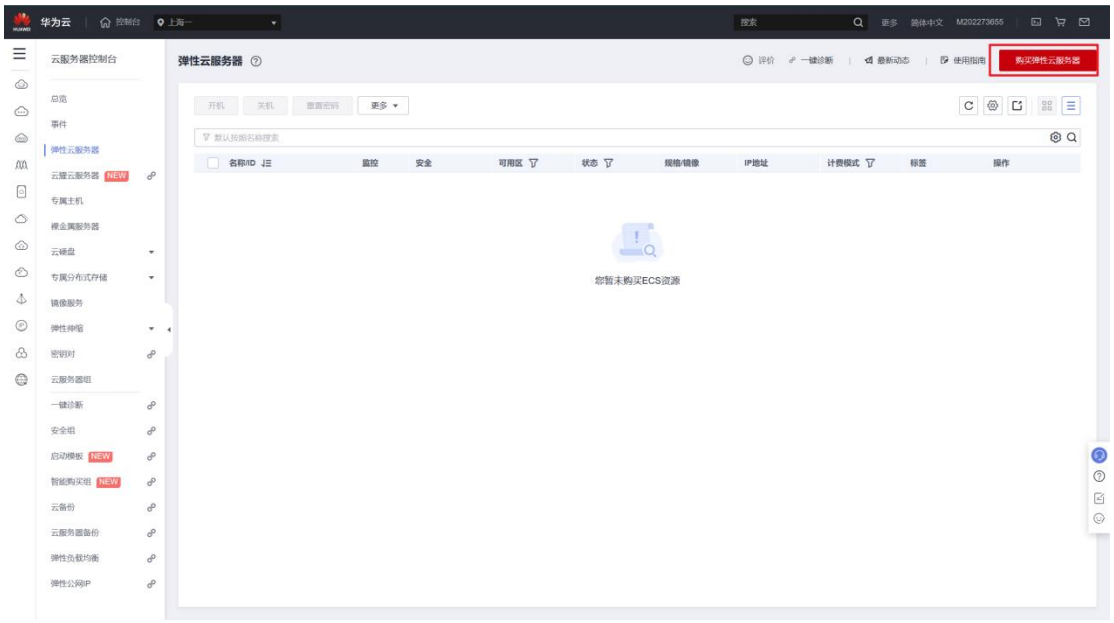
实验环境部署手册

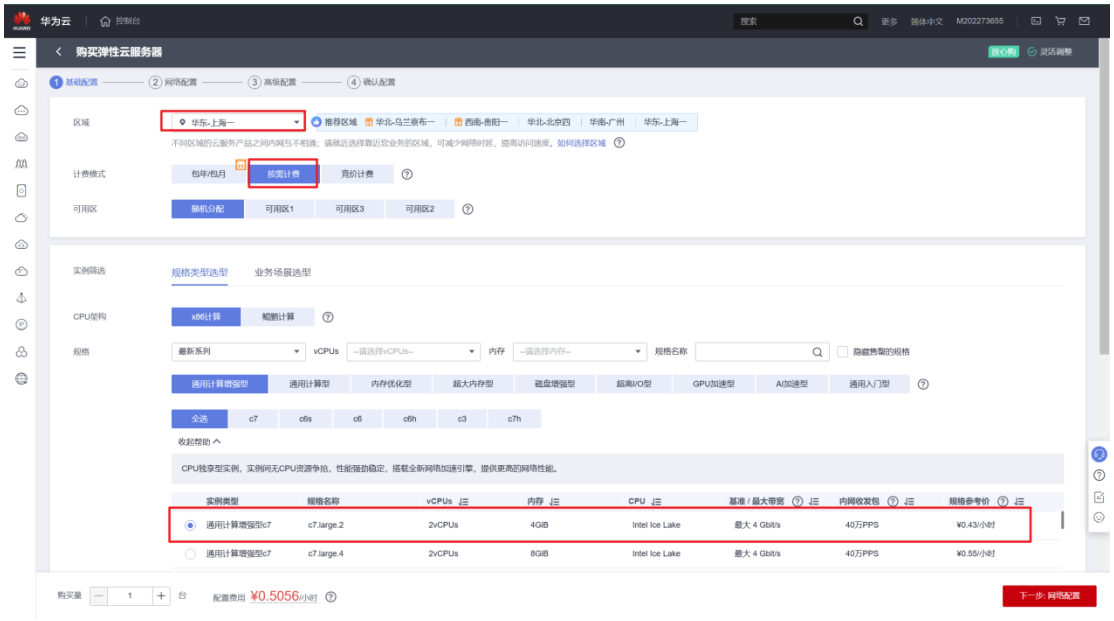
一、华为云 Linux 服务器环境搭建

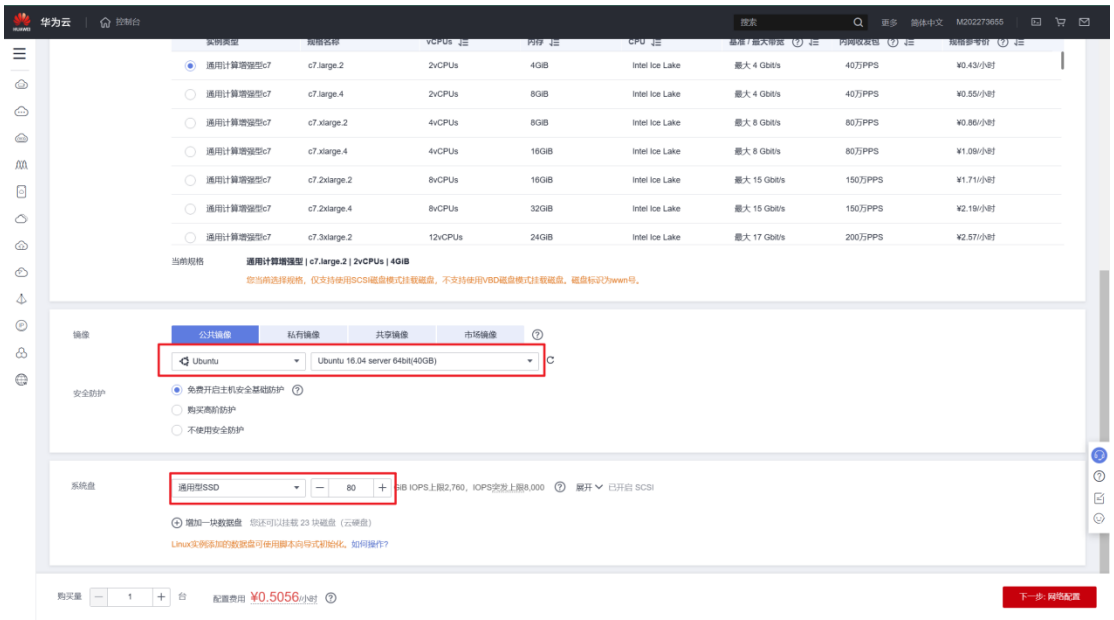
1. 购买云服务器

(1) 登录华为云官网购买弹性云服务器 ECS。

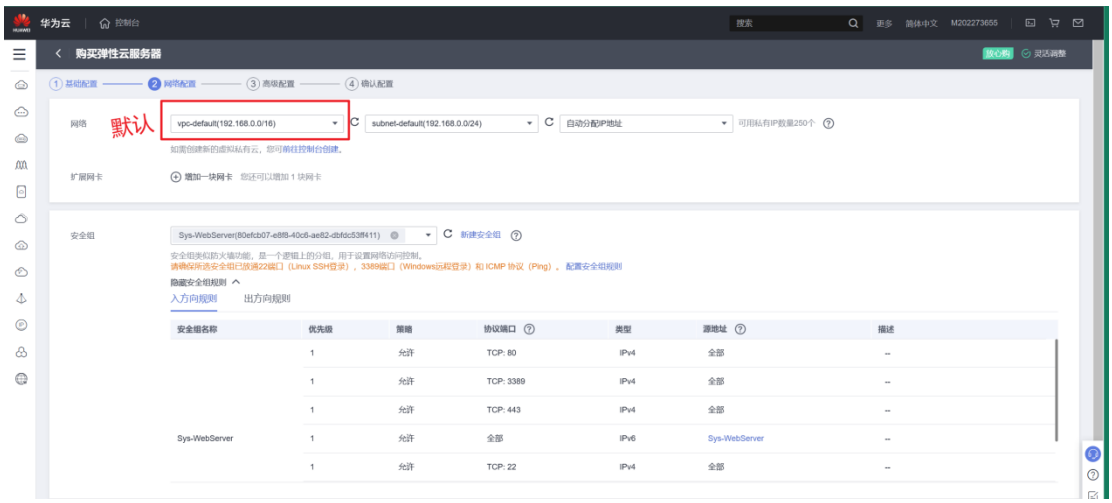


(2) 计费模式选择【按需计费】，区域选择【华东-上海一】(可选其它区域，选择其它区域可能会在下数据集的时候慢，建议选相同区域)，选择 Linux 系统(Ubuntu16.04)镜像（配置最低为 2 核，内存 4GB，硬盘 80GB），其它为默认配置，最后点击立即购买。

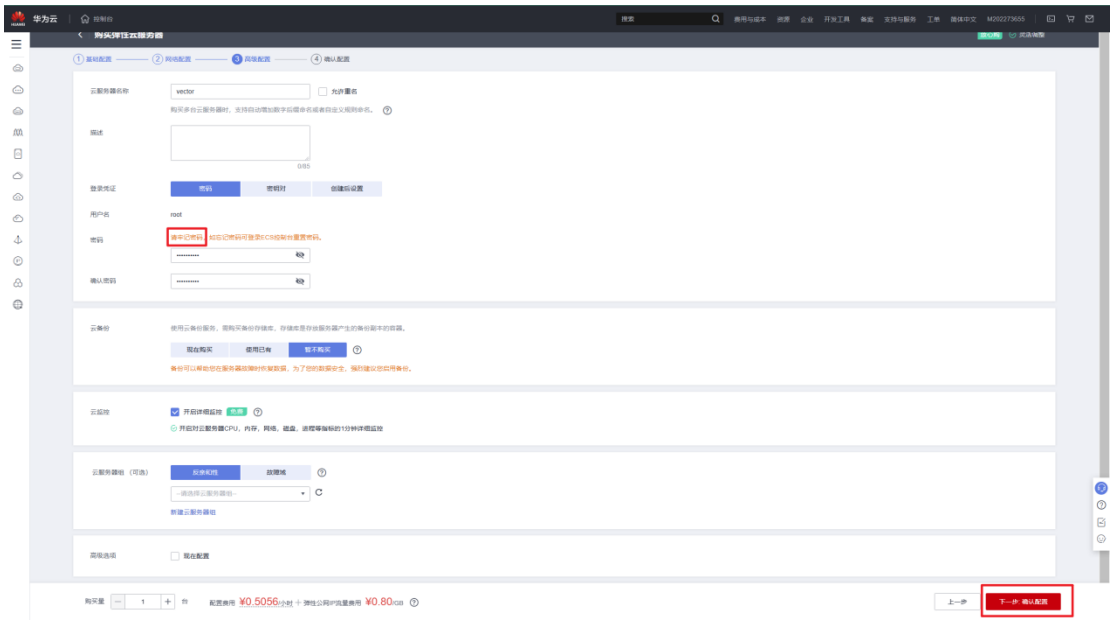

华为云
控制台
购买弹性云服务器
区域
华东-上海一
计费模式
按需计费
可用区
实例筛选
规格类型选型
业务场景选型
CPU架构
规格
通用计算增强型
通用计算型
内存优化型
超大内存型
磁盘增强型
超高I/O型
GPU加速型
AI加速型
通用入门型
实例类型
规格名称
vCPUs
内存
CPU
基准 / 最大带宽
内网收发包
规格参考价
通用计算增强型c7
c7.large.2
2vCPUs
4GiB
Intel Ice Lake
最大 4 Gbit/s
40万PPS
¥0.43/小时
c7.large.4
8GiB
¥0.55/小时
购买量
配置费用 ¥0.5056/小时
下一步: 网络配置

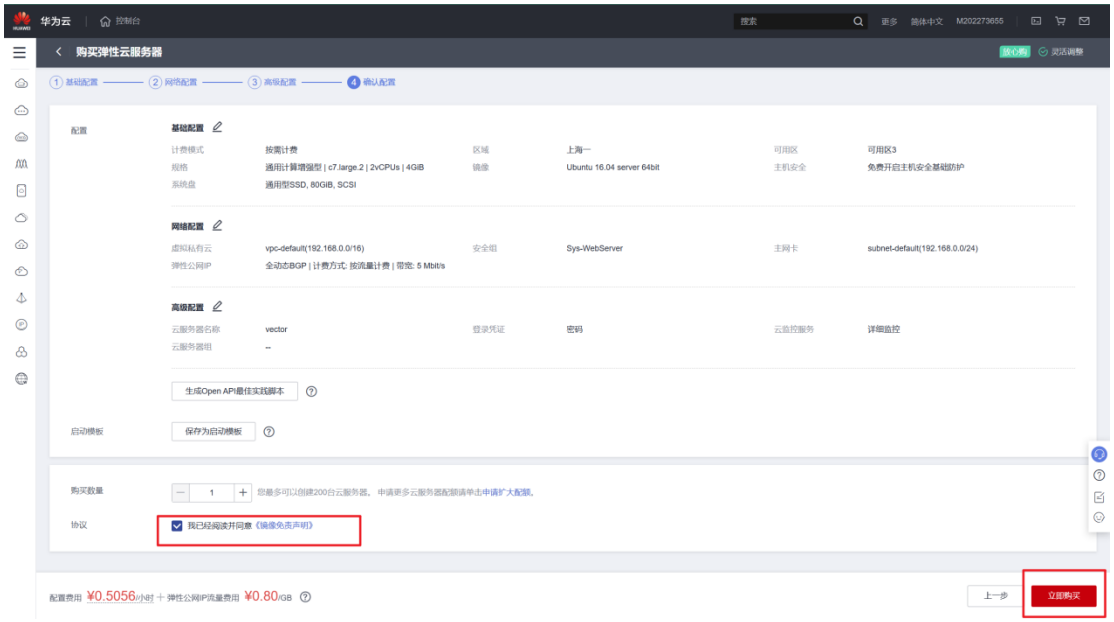

华为云
控制台
通用计算增强型c7
c7.large.2
2vCPUs
4GiB
c7.large.4
8GiB
c7.xlarge.2
4vCPUs
8GiB
c7.xlarge.4
16GiB
c7.2xlarge.2
8vCPUs
16GiB
c7.2xlarge.4
32GiB
c7.3xlarge.2
12vCPUs
24GiB
Intel Ice Lake
当前规格
通用计算增强型 | c7.large.2 | 2vCPUs | 4GiB
镜像
公共镜像
私有镜像
共享镜像
市场镜像
Ubuntu
Ubuntu 16.04 server 64bit(40GB)
安全防护
免费开启主机安全基础防护
购买高阶防护
不使用安全防护
系统盘
通用型SSD
80
增加一块数据盘
购买量
配置费用 ¥0.5056/小时
下一步: 网络配置

进入网络配置

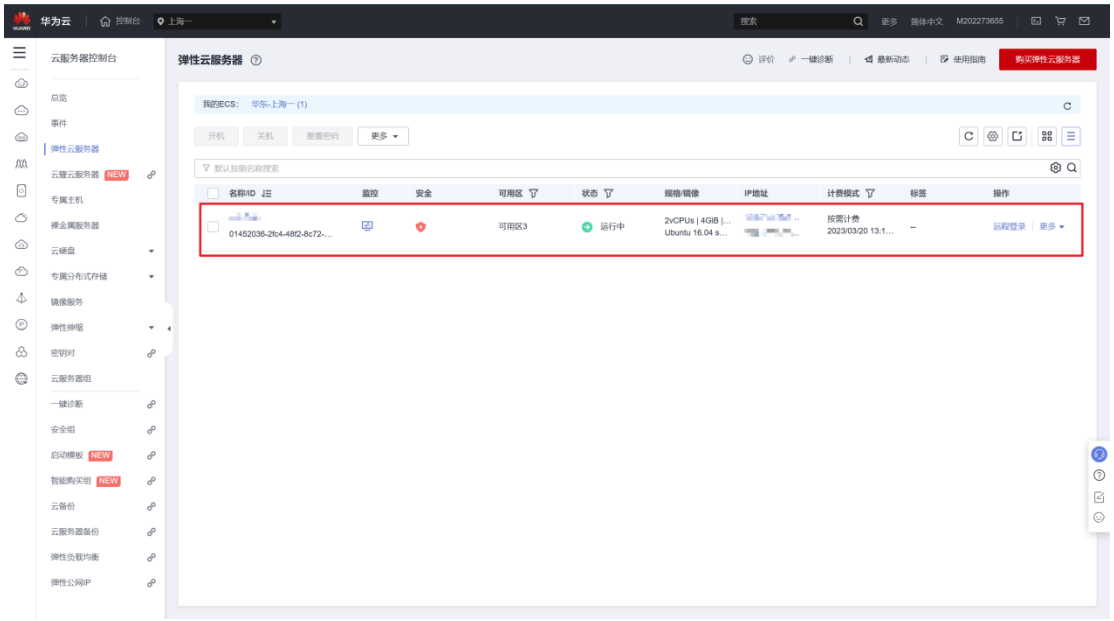

华为云
控制台
购买弹性云服务器
网络
默认
vpc-default(192.168.0.0/16)
subnet-default(192.168.0.0/24)
自动分配IP地址
扩展网卡
增加一块网卡
安全组
Sys-WebServer(80efcb07-e8f8-40c6-ae82-dbfdc53ff411)
新建安全组
入方向规则
出方向规则
安全组名称
优先级
策略
协议端口
类型
源地址
描述
Sys-WebServer
允许
全部
IPv4
IPv6
TCP: 80
TCP: 3389
TCP: 443
TCP: 22

可以自己设定服务器名称，设置密码，密码要记住。

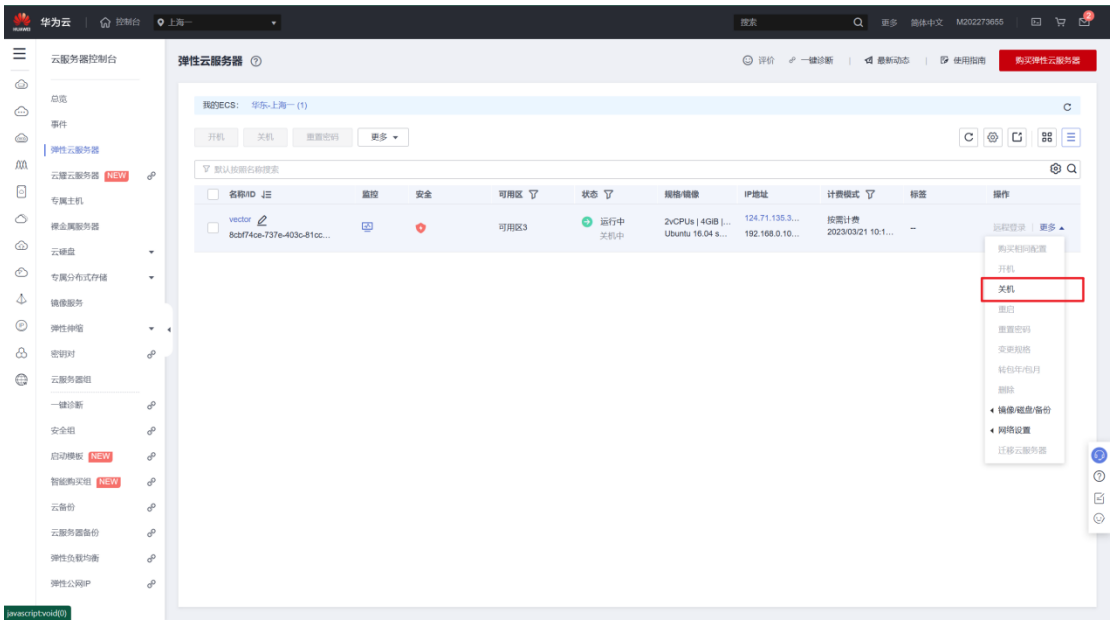




(3) 支付完成后，进入华为云控制台，找到自己的服务器，其中弹性公网 IP 地址需要记下来，之后连接会使用。点击“名称”可进入更详细配置。



所选为按需计费，华为云支持即开即停，秒级计费。如上图是启动状态，不使用的服务器，一定关机，关机能减少大量计费。普通实例关机后，基础资源(包括 vCPU、内存、镜像)不计费。其他绑定资源（云硬盘、弹性公网 IP）正常计费，详细费用说明见官网。



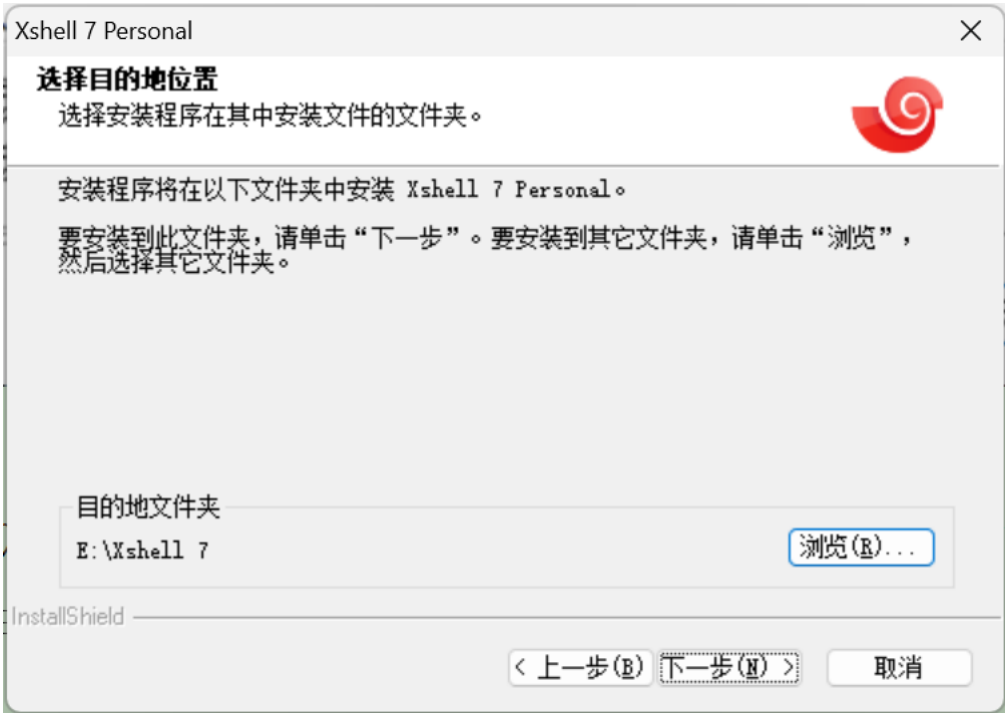
2. 安装 Xshell

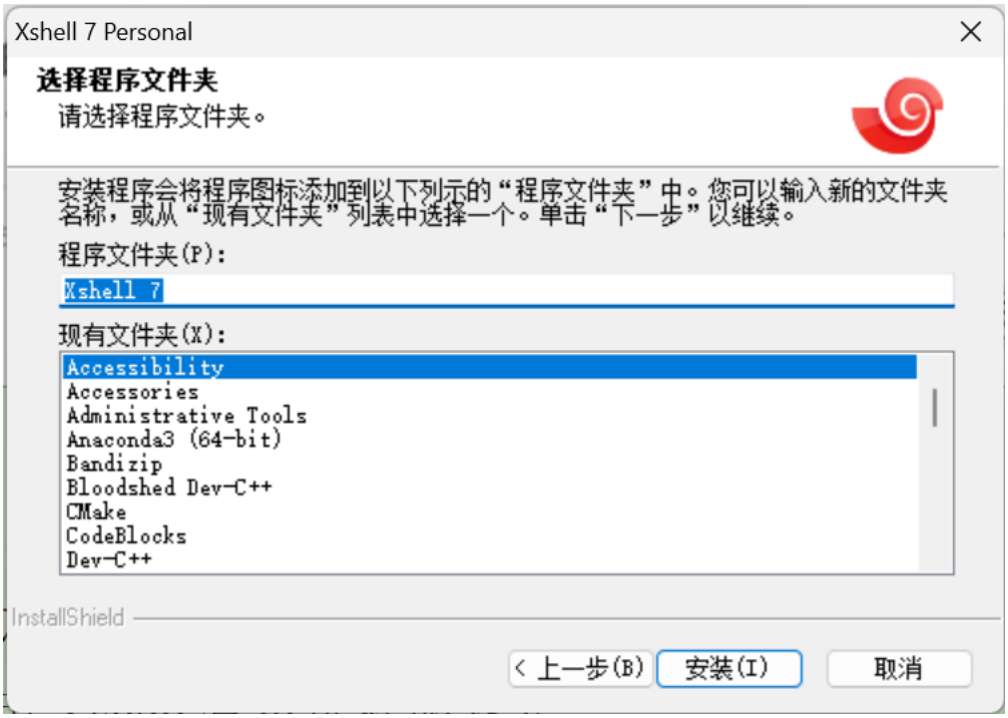
在本地机器上下载安装 Xshell，实现远程登录服务器。


Xshell 7 Personal
欢迎使用 Xshell 7 Personal InstallShield Wizard
InstallShield(R) Wizard 将在计算机中安装 Xshell 7 Personal。若要继续，请单击“下一步”。
< 上一步(B)
下一步(N) >
取消

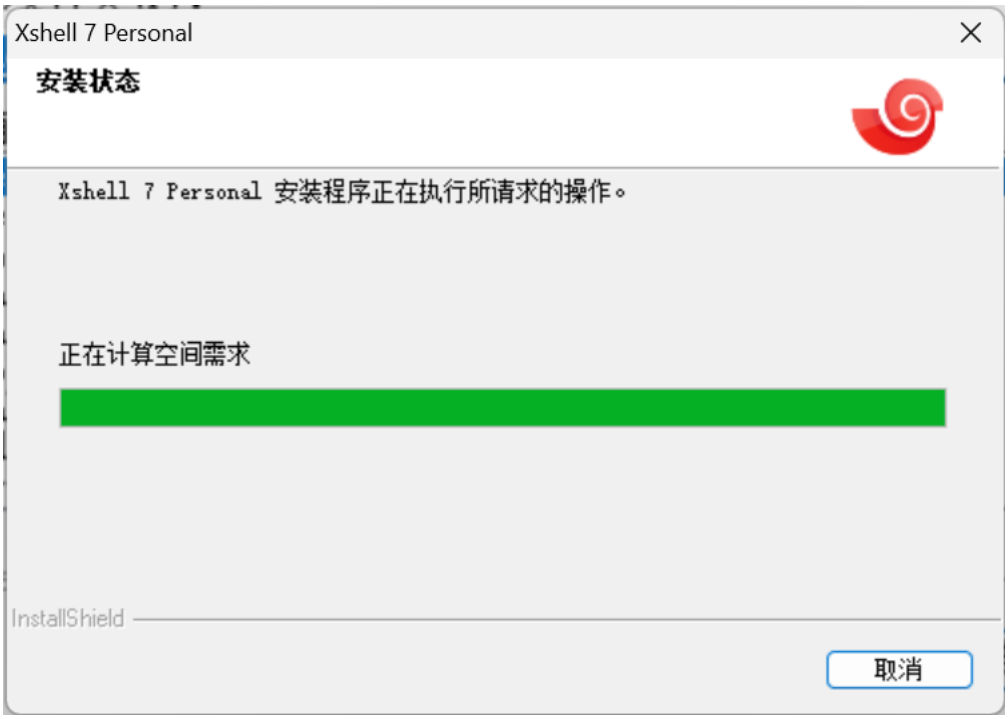

Xshell 7 Personal
许可证协议
请仔细阅读下面的许可证协议。
NetSarang Computer Software License Agreement & Privacy Policy
This license agreement and privacy policy applies to the free version of NetSarang Computer Software and the tool used to install NetSarang Computer Software.
License Agreement
我接受许可证协议中的条款(A)
我不接受许可证协议中的条款(D)
打印(P)
InstallShield
< 上一步(B)
下一步(N) >
取消

选择安装路径


Xshell 7 Personal
选择目的地位置
选择安装程序在其中安装文件的文件夹。
安装程序将在以下文件夹中安装 Xshell 7 Personal。
要安装到此文件夹，请单击“下一步”。要安装到其它文件夹，请单击“浏览”，然后选择其它文件夹。
目的地文件夹
E:\Xshell 7
浏览(R)...
InstallShield
< 上一步(B)
下一步(N) >
取消

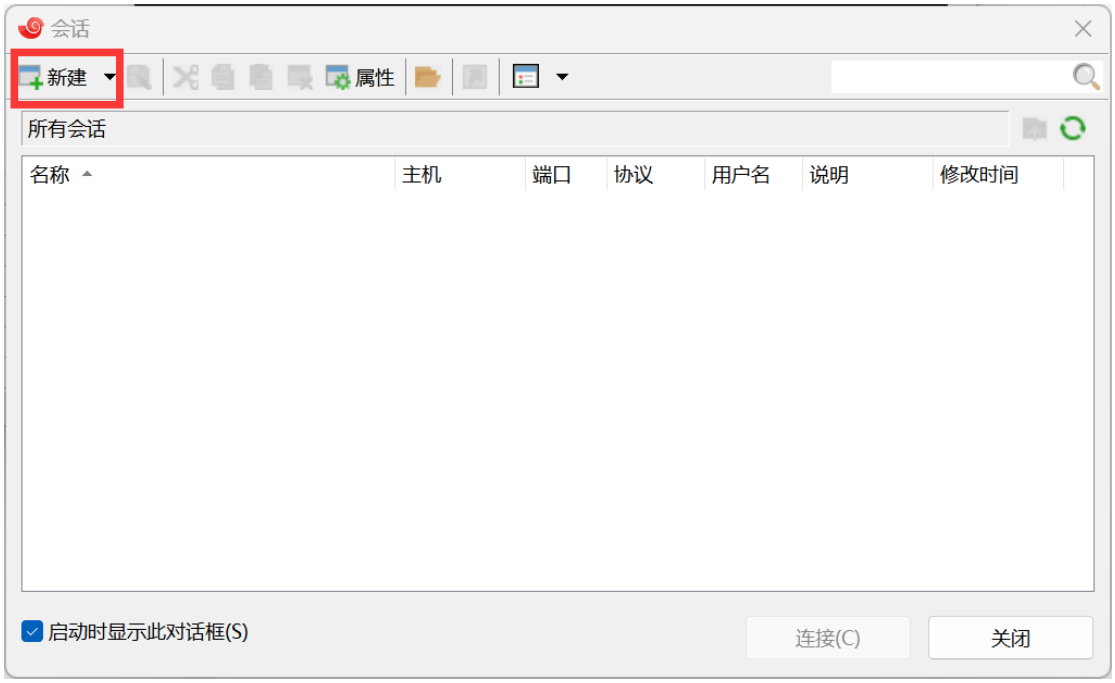

Xshell 7 Personal
选择程序文件夹
请选择程序文件夹。
安装程序会将程序图标添加到以下列示的“程序文件夹”中。您可以输入新的文件夹名称，或从“现有文件夹”列表中选择一个。单击“下一步”以继续。
程序文件夹(P):
Xshell 7
现有文件夹(X):
Accessibility
Accessories
Administrative Tools
Anaconda3 (64-bit)
Bandizip
Bloodshed Dev-C++
CMake
CodeBlocks
Dev-C++
InstallShield
< 上一步(B)
安装(I)
取消

点击安装，等待安装完成

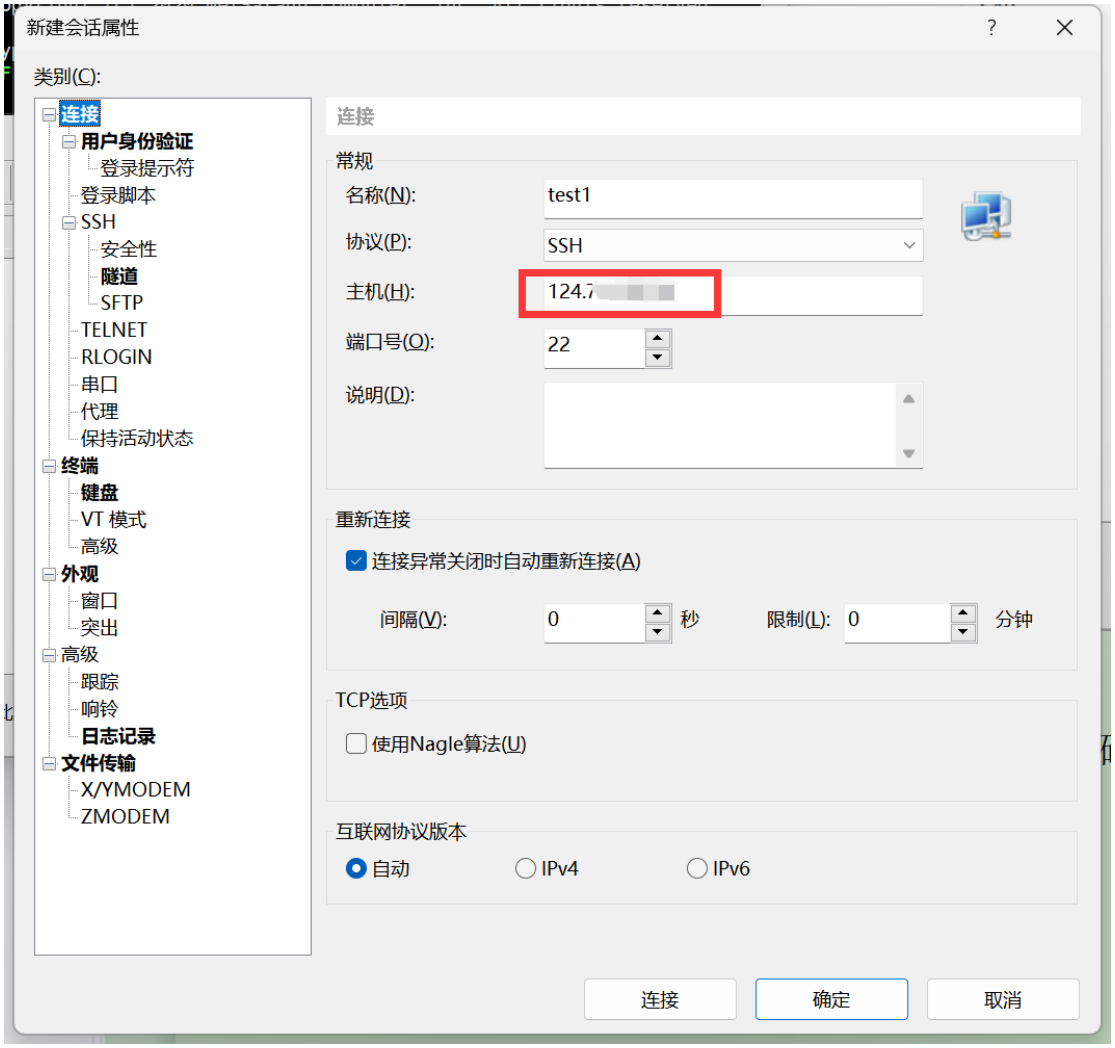


3. 使用 Xshell 建立连接

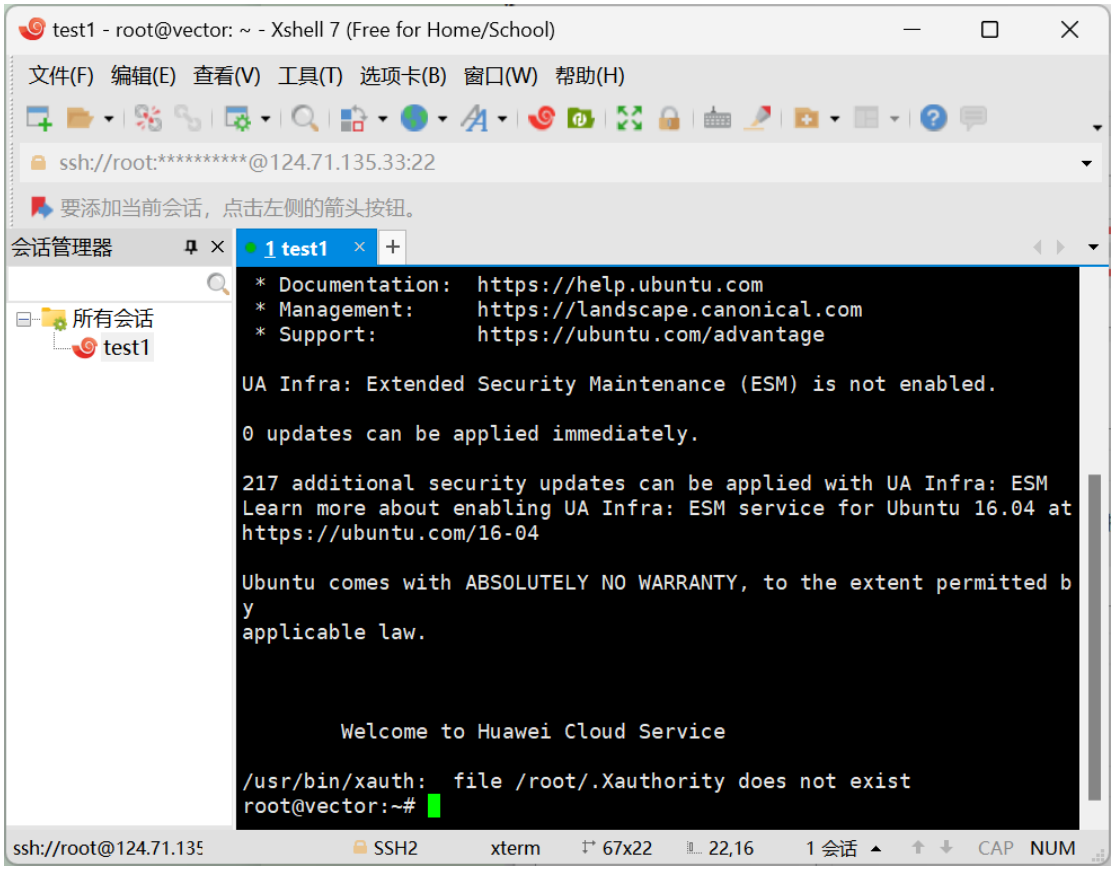
新建连接，指定服务器 IP 地址和端口号：



IP 地址为 1.3 步中的 IP 地址（弹性公网 IP 地址）。

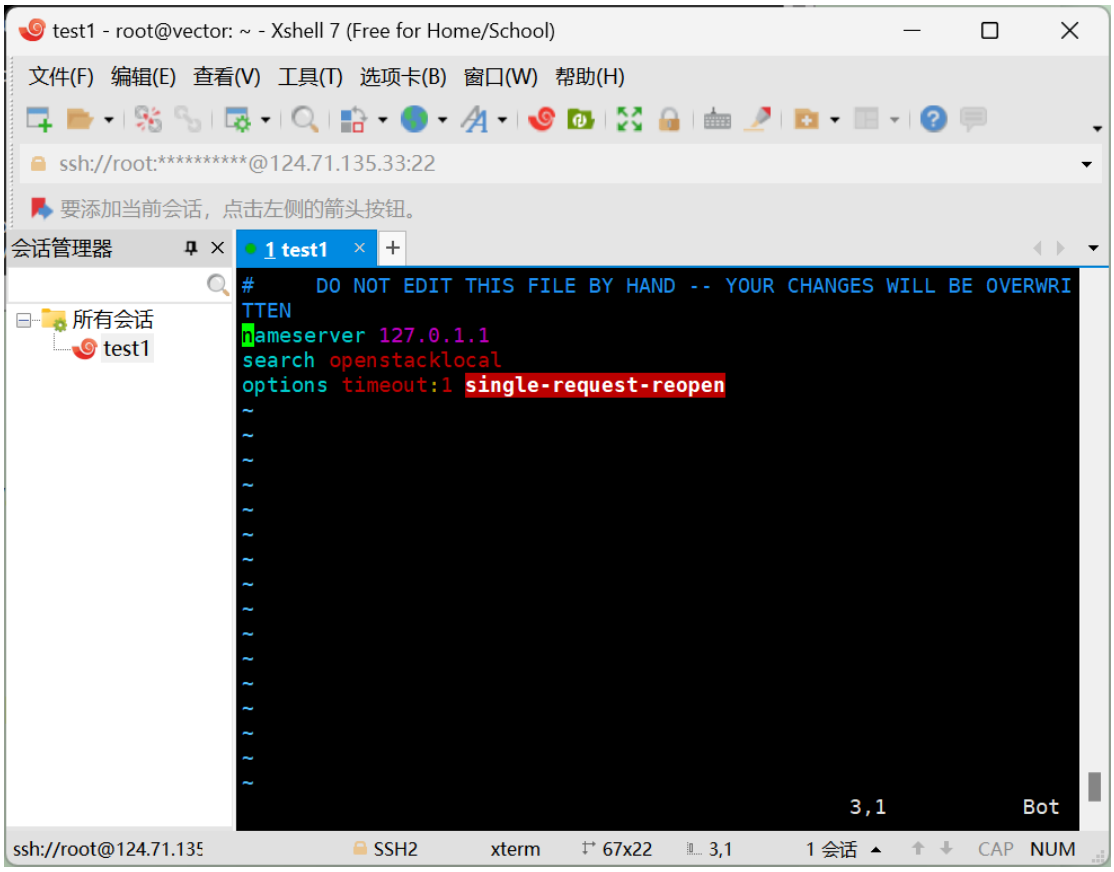


点击用户身份验证，输入用户名（默认为 root）和设置的密码（服务器端可重置），最后点连接，使用 Xshell 登录远程服务器：

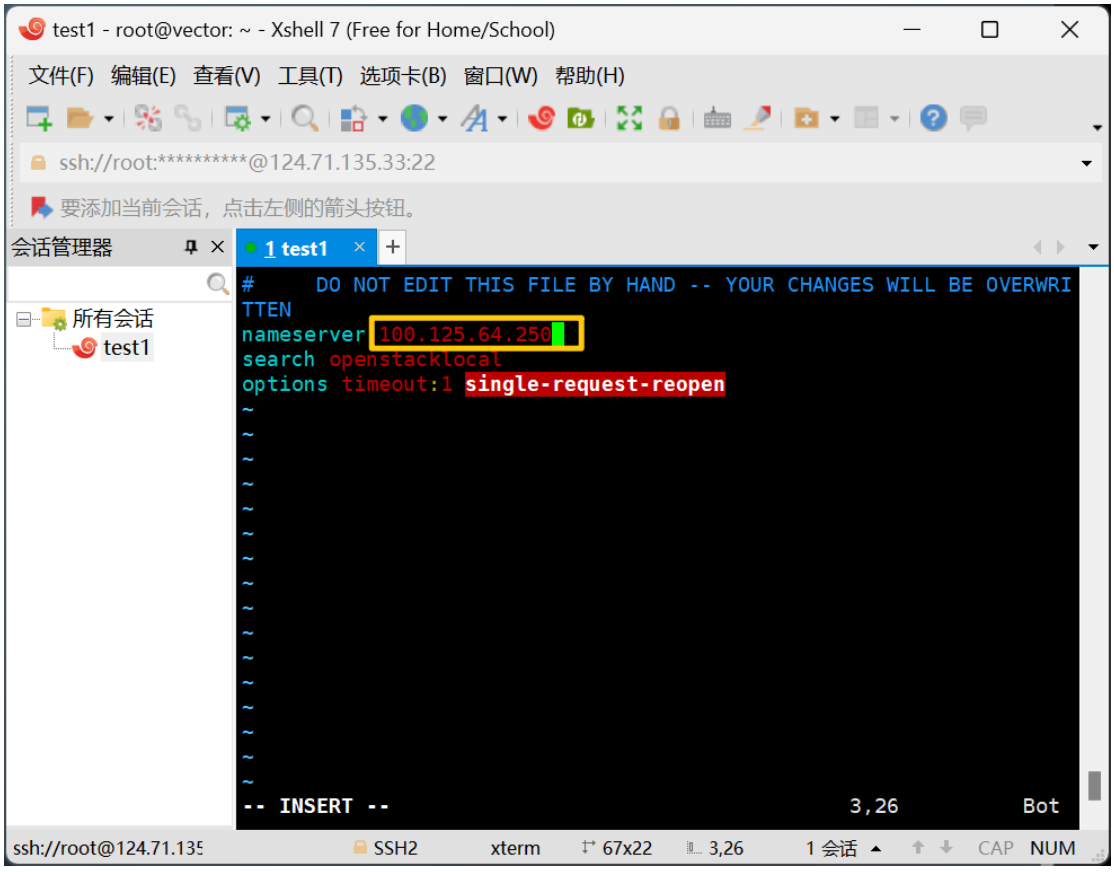


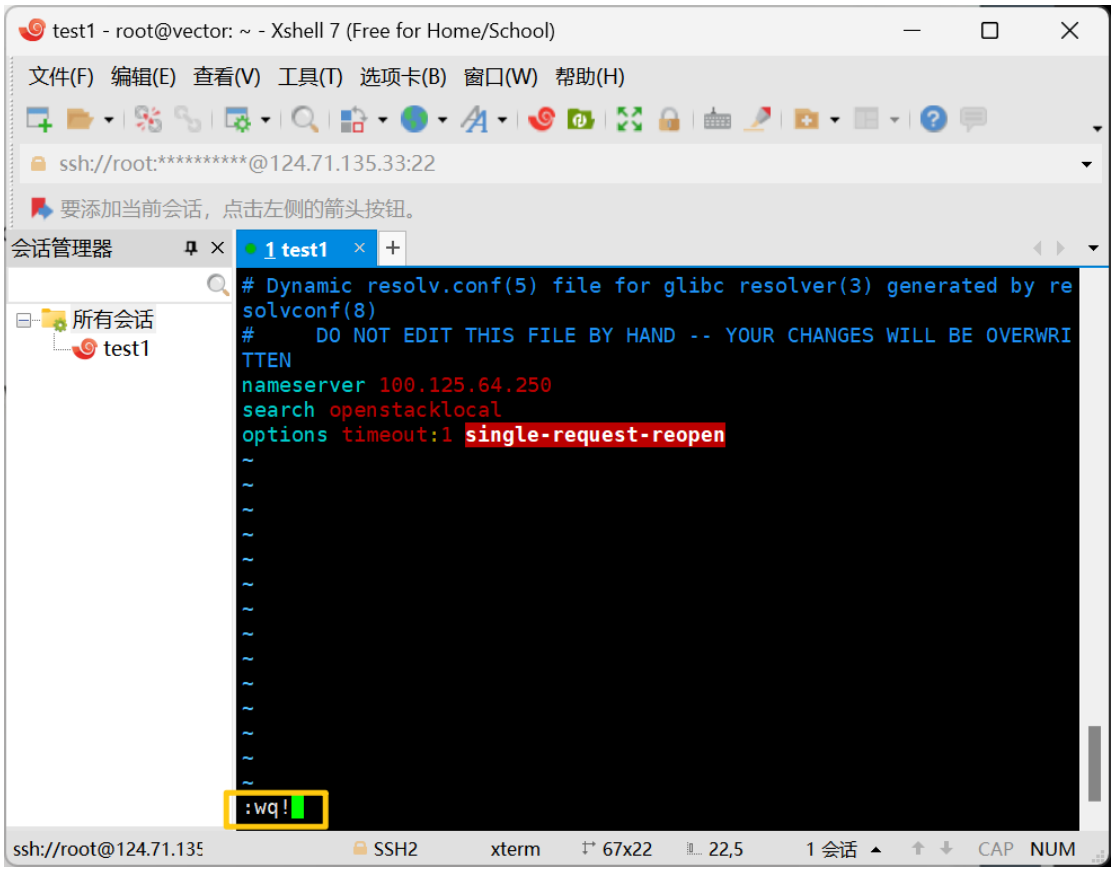
4. 增加内网配置

命令行输入 vim /etc/resolv.conf，服务购买的是上海 1 的增加 ip 地址为 100.125.64.250，选择其它区域服务器的参看官网。



先按 i 键，转换为插入模式，然后增加内网地址，按 esc 后输入**:wq!**回车保存并退出。

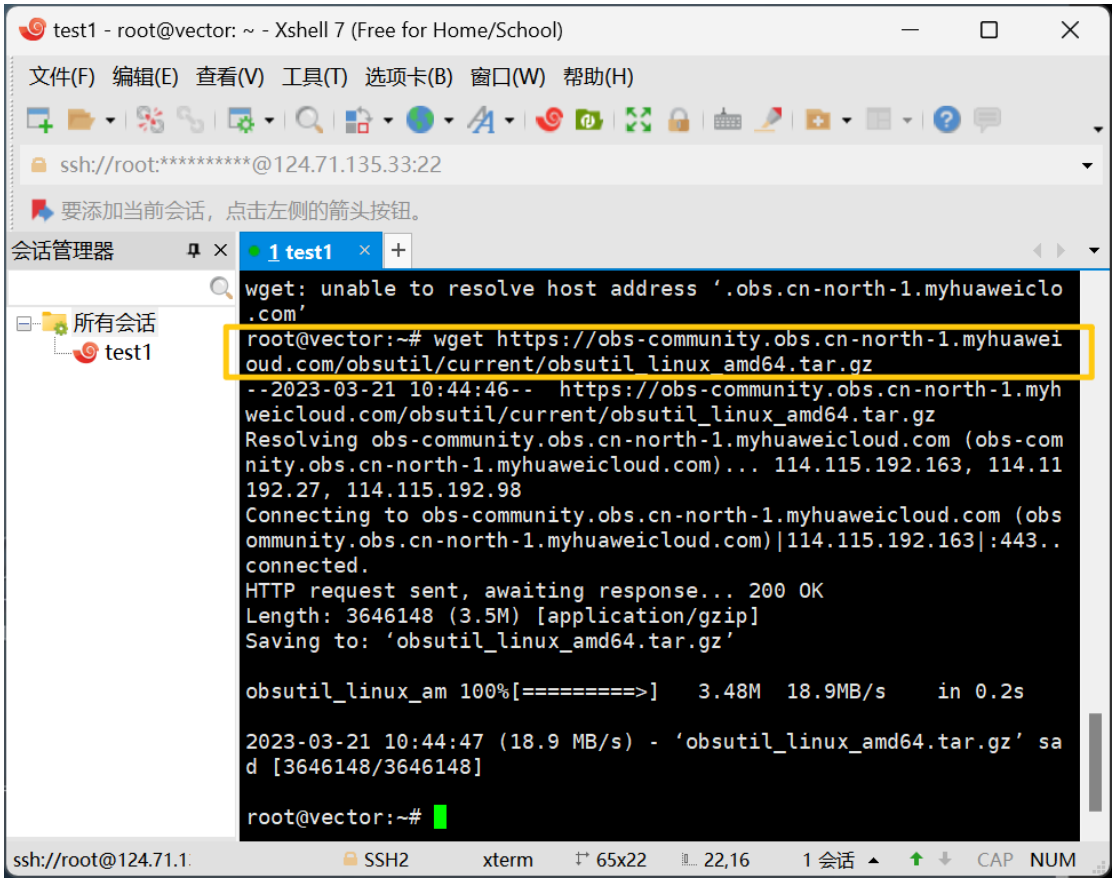




5. 安装 obsutil 以快速下载 yelp 数据集

(1) 输入下载命令

wget https://obs-community.obs.cn-north-1.myhuaweicloud.com/obsutil/current/obsutil_linux_amd64.tar.gz



(2) 解压缩

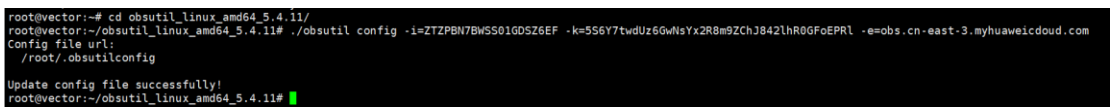
tar -zvxf obsutil_linux_amd64.tar.gz

```
root@vector:~# ls
'   obsutil_linux_amd64.tar.gz
root@vector:~# tar -zvxf obsutil_linux_amd64.tar.gz
obsutil_linux_amd64_5.4.11/
obsutil_linux_amd64_5.4.11/setup.sh
obsutil_linux_amd64_5.4.11/obsutil
root@vector:~#
```

(3) 进入该目录配置 obsutil

需要 OBS 终端节点地址(Endpoint)和访问密钥（AK 和 SK），选择上海 1 区域服务器的输入命令：

./obsutil config -i=ZTZPBN7BWSS01GDSZ6EF -k=5S6Y7twdUz6GwNsYx2R8m9ZChJ842lhR0GFoEPRl -e=obs.cn-east-3.myhuaweicloud.com



(4) 如何下载数据集

下面是如何下载 obs 中的数据集，在 obsutil 的同级目录下创建 data 文件夹用于存放下载的 yelp 数据集。

```
root@vector:~# ls
'   obsutil_linux_amd64_5.4.11   obsutil_linux_amd64.tar.gz
root@vector:~# mkdir data
root@vector:~# ls
'   data   obsutil_linux_amd64_5.4.11   obsutil_linux_amd64.tar.gz
root@vector:~#
```

下载数据集命令为：

./obsutil cp 源地址 目标地址

mongo 和 neo4j 的 yelp 数据集的源地址分别为：

obs://big-data-management/dataset/data_for_mysql.zip

obs://big-data-management/dataset/data_for_mongo.zip

obs://big-data-management/dataset/data_for_neo4j.zip

下面是下载 data_for_neo4j.zip 数据集的示例：

```
root@cs-tan:~/obsutil_linux_amd64_5.4.11# ./obsutil cp obs://big-data-management/dataset/data_for_neo4j.zip ../data/
Start at 2023-10-02 01:55:49.377220186 +0000 UTC


Parallel:      5                 Jobs:         5
Threshold:     50.00MB           PartSize:     auto
VerifyLength:  false             VerifyMd5:    false
CheckpointDir: /root/.obsutil_checkpoint
TempFileDir: /root/obsutil_linux_amd64_5.4.11

Waiting to prepare the temp file [2018713676].

[-----------------------------------] 100.00% 149.46MB/s 1.88GB/1.88GB 13.083s
Waiting to rename temporary file...

Download successfully, 1.88GB, n/a, obs://big-data-management/dataset/data_for_neo4j.zip --> /root/data/data_for_neo4j.zip, cost [14083], status [
206], request id [0000018AEE194E09B164A73AC407A34D]
```

下载完后 cd 到 data 文件夹查看目标地址：

```
root@tan-shard1:~/data# ls
data_for_mongo  data_for_mongo.zip  data_for_mysql  data_for_mysql.zip  data_for_neo4j  data_for_neo4j.zip
```

二、安装 MongoDB

1. 准备,更新软件源并安装 mongodb

准备安装 MongoDB4.4 版本，官网查找软件源公钥并导入：

curl -fsSL https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

为 mongodb 创建软件源 list 文件：

echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list

sudo apt update

sudo apt install mongodb-org

2. 使用 Mongo

命令行启动 mongod 服务：

mongod --dbpath /var/lib/mongodb/ --logpath /var/log/mongodb/mongodb.log --logappend &

命令说明：

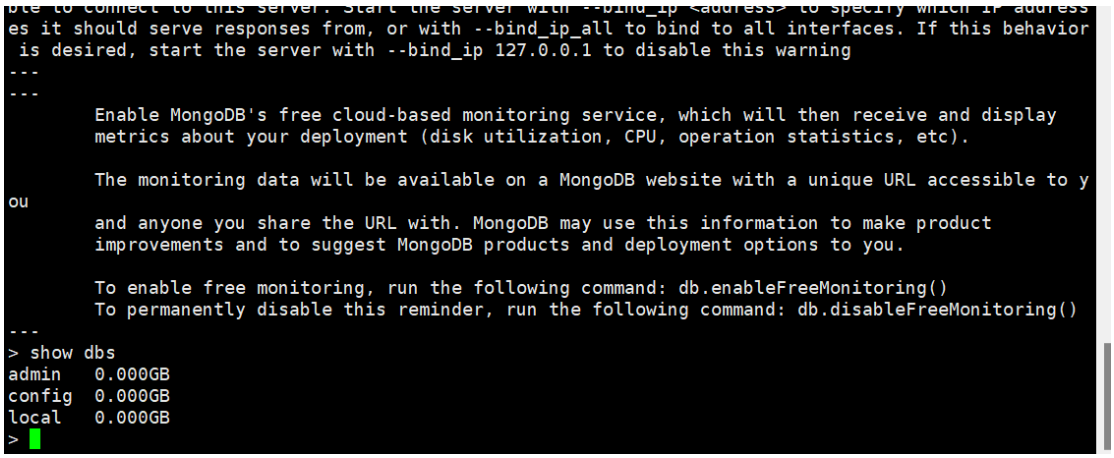
--dbpath：指定 mongo 的数据库文件在哪个文件夹

--logpath：指定 mongo 的 log 日志是哪个，这里 log 一定要指定到具体的文件名

--logappend：表示 log 的写入是采用附加的方式，默认的是覆盖之前的文件

&：后台运行

输入 mongo 即可使用 MongoDB：



3. 启动、重启和关闭服务命令

service mongod start

service mongod restart

service mongod stop

5. 导入数据

请将下载的数据集压缩文件都解压后使用，下面是 neo4j 数据集的解压示例：

```
root@vector:~/data/data_for_neo4j# ls
data_for_neo4j.zip
root@vector:~/data/data_for_neo4j# unzip data_for_neo4j.zip
Archive:  data_for_neo4j.zip
  inflating: business_header.csv
  inflating: business_IN_CATEGORY_category.csv
  inflating: business_IN_CATEGORY_category_header.csv
  inflating: business_IN_CITY_city.csv
  inflating: business_IN_CITY_city_header.csv
  inflating: BusinessNode.csv
  inflating: category_header.csv
  inflating: CategoryNode.csv
  inflating: city_header.csv
  inflating: CityNode.csv
  inflating: review.csv
  inflating: review_header.csv
  inflating: review_REVIEWS_business.csv
  inflating: review_REVIEWS_business_header.csv
  inflating: user.csv
  inflating: user_FRIENDS_user.csv
  inflating: user_FRIENDS_user_header.csv
  inflating: user_header.csv
  inflating: user_WROTE_review.csv
  inflating: user_WROTE_review_header.csv
root@vector:~/data/data_for_neo4j# ls
business_header.csv                       review.csv
business_IN_CATEGORY_category.csv         review_header.csv
business_IN_CATEGORY_category_header.csv  review_REVIEWS_business.csv
business_IN_CITY_city.csv                 review_REVIEWS_business_header.csv
business_IN_CITY_city_header.csv          user.csv
BusinessNode.csv                          user_FRIENDS_user.csv
category_header.csv                       user_FRIENDS_user_header.csv
CategoryNode.csv                          user_header.csv
city_header.csv                           user_WROTE_review.csv
CityNode.csv                              user_WROTE_review_header.csv
data_for_neo4j.zip
root@vector:~/data/data_for_neo4j# rm data_for_neo4j.zip
root@vector:~/data/data_for_neo4j# ls
business_header.csv                       review.csv
business_IN_CATEGORY_category.csv         review_header.csv
business_IN_CATEGORY_category_header.csv  review_REVIEWS_business.csv
business_IN_CITY_city.csv                 review_REVIEWS_business_header.csv
business_IN_CITY_city_header.csv          user.csv
BusinessNode.csv                          user_FRIENDS_user.csv
category_header.csv                       user_FRIENDS_user_header.csv
CategoryNode.csv                          user_header.csv
city_header.csv                           user_WROTE_review.csv
CityNode.csv                              user_WROTE_review_header.csv
root@vector:~/data/data_for_neo4j#
```

直接执行：

mongorestore -h localhost -d yelp --dir <mongo 数据集所在位置>

如图：

```
root@vector:~/data# cd data_for_mongo/
root@vector:~/data/data_for_mongo# ls
business.bson           review.bson           test_map_reduce.metadata.json
business.metadata.json  review.metadata.json  user.bson
data_for_mongo.zip      test_map_reduce.bson  user.metadata.json
root@vector:~/data/data_for_mongo# rm  data_for_mongo.zip
root@vector:~/data/data_for_mongo# ls
business.bson           review.bson           test_map_reduce.bson           user.bson
business.metadata.json  review.metadata.json  test_map_reduce.metadata.json  user.metadata.json
root@vector:~/data/data_for_mongo# cd ..
root@vector:~/data# mongorestore -h localhost -d yelp --dir data_for_mongo
2023-03-21T17:07:02.942+0800    The --db and --collection flags are deprecated for this use-case; pl
ease use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLLECTION}
2023-03-21T17:07:02.942+0800    building a list of collections to restore from data_for_mongo dir
2023-03-21T17:07:02.943+0800    reading metadata for yelp.business from data_for_mongo/business.meta
data.json
2023-03-21T17:07:02.943+0800    reading metadata for yelp.review from data_for_mongo/review.metadata
.json
2023-03-21T17:07:02.943+0800    reading metadata for yelp.test_map_reduce from data_for_mongo/test_m
ap_reduce.metadata.json
2023-03-21T17:07:02.943+0800    reading metadata for yelp.user from data_for_mongo/user.metadata.jso
n
2023-03-21T17:07:02.964+0800    restoring yelp.review from data_for_mongo/review.bson
2023-03-21T17:07:02.969+0800    restoring yelp.user from data_for_mongo/user.bson
2023-03-21T17:07:02.974+0800    restoring yelp.business from data_for_mongo/business.bson
2023-03-21T17:07:02.977+0800    restoring yelp.test_map_reduce from data_for_mongo/test_map_reduce.b
son
2023-03-21T17:07:05.951+0800    [........................]           yelp.review  64.2MB/5.16GB   (1
.2%)
2023-03-21T17:07:05.951+0800    [........................]             yelp.user   111MB/2.90GB   (3
.7%)
2023-03-21T17:07:05.951+0800    [##########..............]         yelp.business   66.8MB/155MB  (43
.1%)
2023-03-21T17:07:05.951+0800    [########................]  yelp.test_map_reduce  15.1MB/44.3MB  (34
.1%)
2023-03-21T17:07:05.951+0800
2023-03-21T17:07:08.942+0800    [........................]           yelp.review   124MB/5.16GB   (2
.4%)
2023-03-21T17:07:08.942+0800    [#.......................]             yelp.user   226MB/2.90GB   (7
.6%)
2023-03-21T17:07:08.942+0800    [####################....]         yelp.business    131MB/155MB  (84
.6%)
2023-03-21T17:07:08.942+0800    [################........]  yelp.test_map_reduce  29.2MB/44.3MB  (66
```

查看可知导入成功：

```
root@vector:~/data# mongo
MongoDB shell version v4.4.19
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c7085de6-8c37-4df9-bd09-e791f6e2b42b") }
MongoDB server version: 4.4.19
---
The server generated these startup warnings when booting:
        2023-03-21T17:02:13.138+08:00: Using the XFS filesystem is strongly recommended with the Wir
edTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2023-03-21T17:02:13.778+08:00: Access control is not enabled for the database. Read and writ
e access to data and configuration is unrestricted
        2023-03-21T17:02:13.778+08:00: You are running this process as the root user, which is not r
ecommended
        2023-03-21T17:02:13.778+08:00: This server is bound to localhost. Remote systems will be una
ble to connect to this server. Start the server with --bind_ip <address> to specify which IP address
es it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior
 is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to y
ou
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
yelp    5.913GB
>
```

三、安装 Neo4J

1. 安装配置 java11 环境

(1) 打开终端，添加 PPA 存储库

sudo add-apt-repository ppa:openjdk-r/ppa

(2) 更新 apt-get 包管理器的软件包索引

sudo apt-get update

(3) 安装 OpenJDK 11

sudo apt-get install openjdk-11-jdk

(4) 检查 Java 版本是否正确安装

java -version

(5) 设置 JAVA_HOME 环境变量

编辑/etc/environment 文件，添加以下内容：

JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64/"

(6) 更新系统环境变量

source /etc/environment

(7) 检查 JAVA_HOME 环境变量是否设置正确

echo $JAVA_HOME

```
root@vector:~# java -version
openjdk version "11.0.14" 2022-01-18
OpenJDK Runtime Environment (build 11.0.14+9-Ubuntu-0ubuntu2.16.04)
OpenJDK 64-Bit Server VM (build 11.0.14+9-Ubuntu-0ubuntu2.16.04, mixed mode, sharing)
root@vector:~# vi /etc/environment
root@vector:~# source /etc/environment
root@vector:~# echo $JAVA_HOME
```

2. 安装 Neo4J

下载 neo4j 压缩包 neo4j-community-4.0.9-unix.tar.gz，使用命令解压：

tar -zxvf neo4j-community-4.0.9-unix.tar.gz

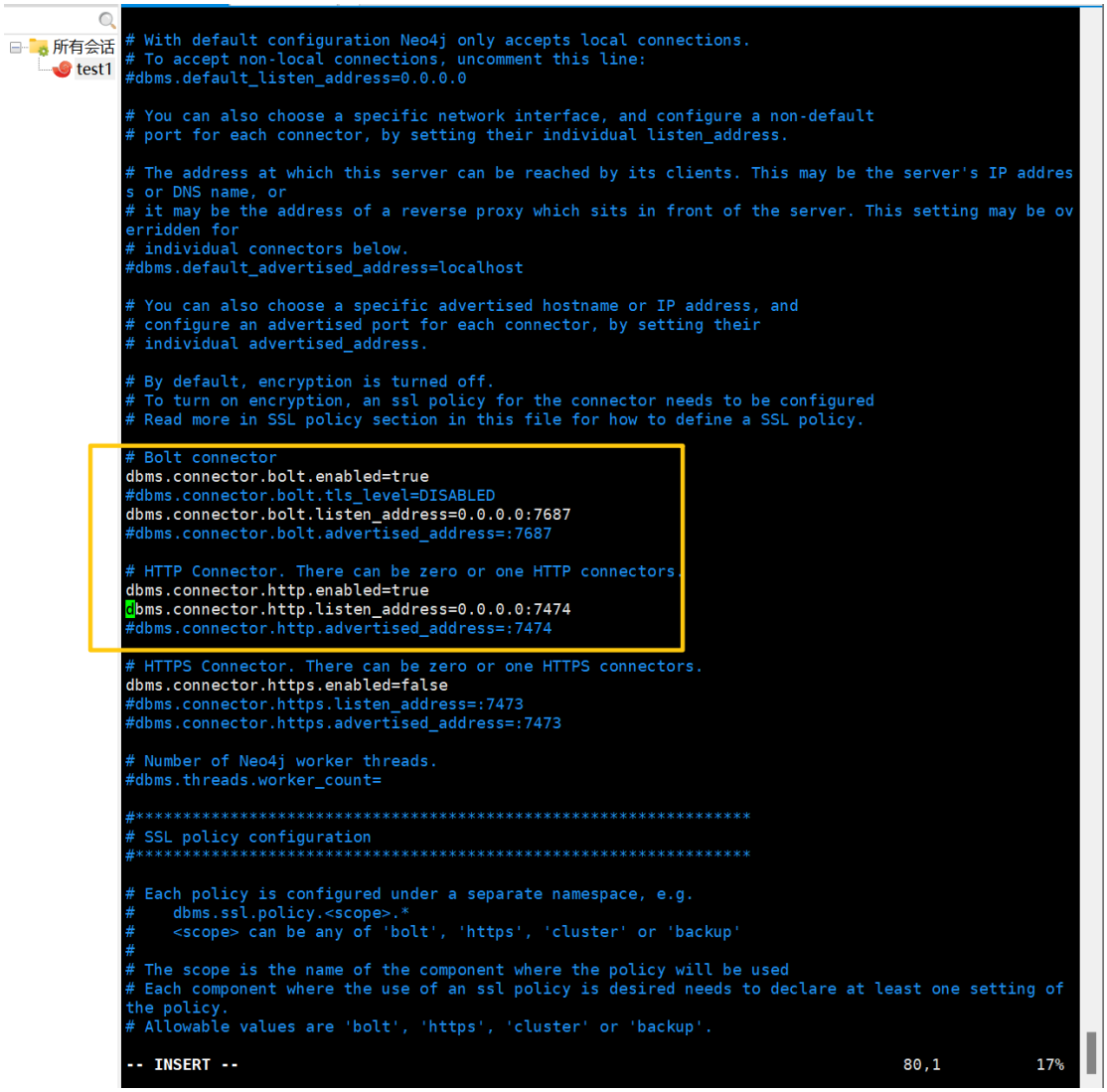
修改 conf 文件，实现本地浏览器访问,修改

#dbms.connector.http.listen_address=:7474

为 dbms.connector.http.listen_address=0.0.0.0:7474

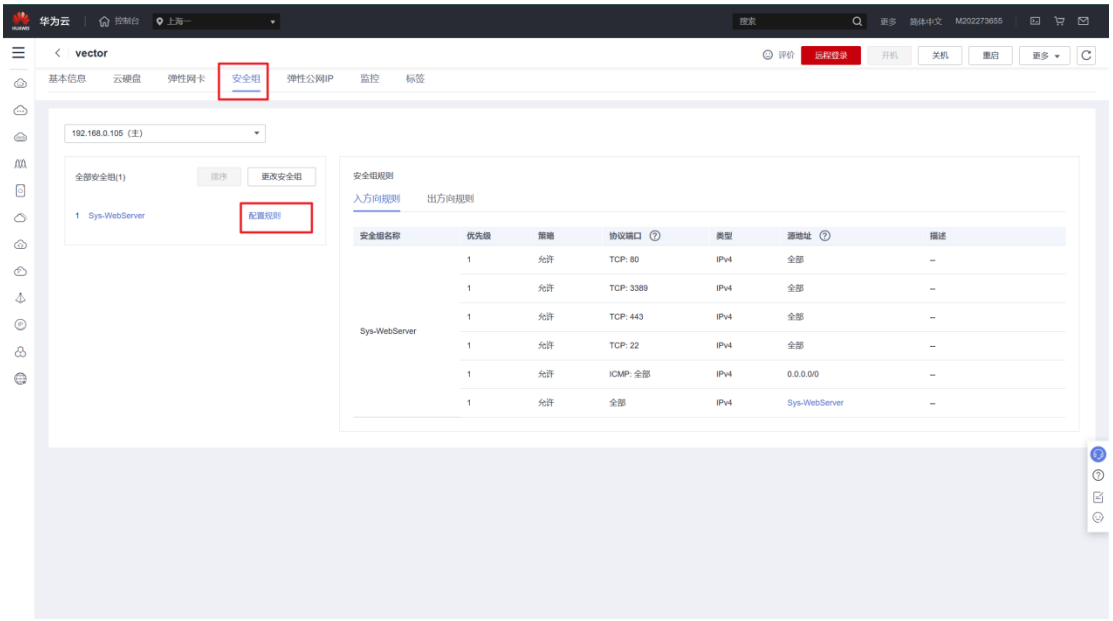
修改远程 bolt 连接#dbms.connector.bolt.listen_address=:7687

为 dbms.connector.bolt.listen_address=0.0.0.0:7687

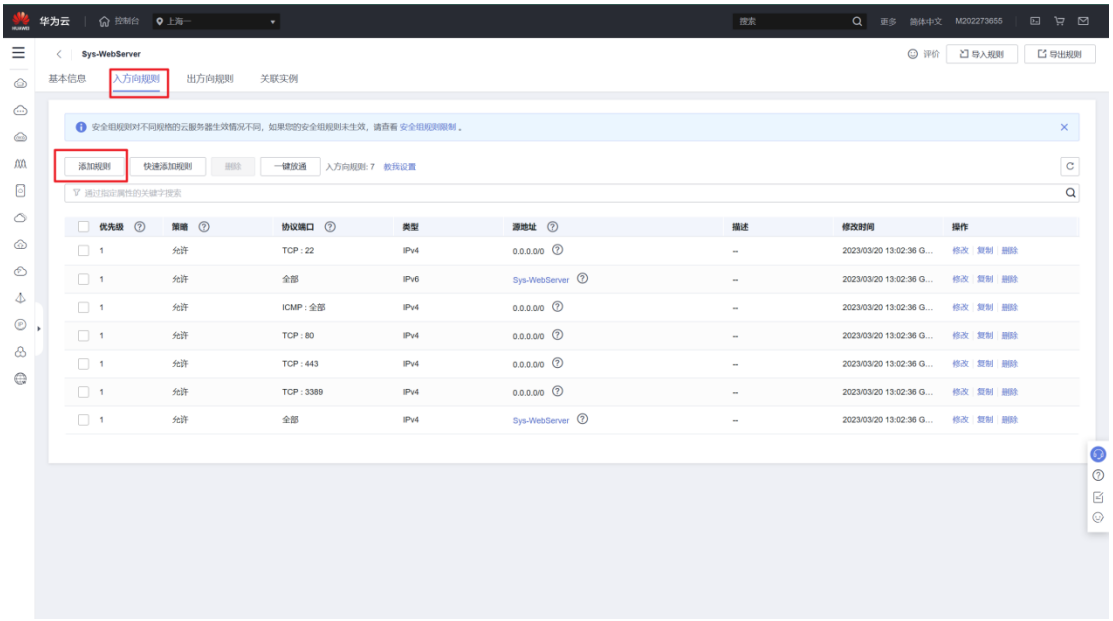


3. 华为云配置安全组规则

点击华为云控制台的服务器，选择安全组，配置规则；

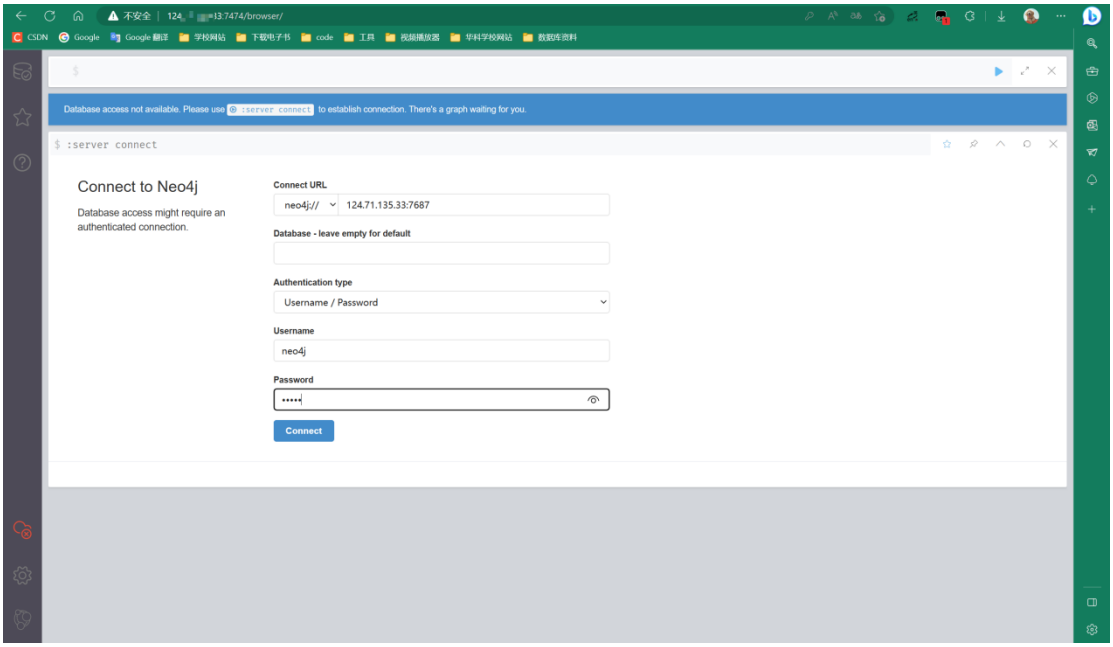


点击添加入方向规则，开放 7474 端口和 7687 端口:



4. 启动 Neo4J

进入解压后的文件夹的 bin 目录下，使用./neo4j console 或./neo4j start 命令即可启动数据库，然后在浏览器中输入 http://<服务器 IP 地址>:7474/ 即可访问，初始时用户名和密码均为 neo4j。



5. 数据导入

在使用之前，我们需要导入实验需要用到的数据。

(1) 将 data_for_neo4j 文件夹下的数据全部拷贝到 neo4j 安装目录下的 import 文

件夹下

```
root@vector:~/neo4j-community-4.0.9/import# ls
root@vector:~/neo4j-community-4.0.9/import# cp ../../data/data_for_neo4j/* ./
root@vector:~/neo4j-community-4.0.9/import# ls
business_header.csv                       review.csv
business_IN_CATEGORY_category.csv         review_header.csv
business_IN_CATEGORY_category_header.csv  review_REVIEWS_business.csv
business_IN_CITY_city.csv                 review_REVIEWS_business_header.csv
business_IN_CITY_city_header.csv          user.csv
BusinessNode.csv                          user_FRIENDS_user.csv
category_header.csv                       user_FRIENDS_user_header.csv
CategoryNode.csv                          user_header.csv
city_header.csv                           user_WROTE_review.csv
CityNode.csv                              user_WROTE_review_header.csv
```

(2) 进入 neo4j 的 bin 目录，执行以下命令：

./neo4j-admin import --id-type=STRING --database=yelp --nodes=UserNode="../import/user_header.csv,../import/user.csv" --nodes=ReviewNode="../import/review_header.csv,../import/review.csv" --nodes=BusinessNode="../import/business_header.csv,../import/BusinessNode.csv" --nodes=CityNode="../import/city_header.csv,../import/CityNode.csv" --nodes=CategoryNode="../import/category_header.csv,../import/CategoryNode.csv" --relationships=HasFriend="../import/user_FRIENDS_user_header.csv,../import/user_FRIENDS_user.csv" --relationships=Review="../import/user_WROTE_review_header.csv,../import/user_WROTE_review.csv" --relationships=Reviewed="../import/review_REVIEWS_business_header.csv,../import/review_REVIEWS_business.csv" --relationships=IN_CITY="../import/business_IN_CITY_city_header.csv,../import/business_IN_CITY_city.csv" --relationships=IN_CATEGORY="../import/business_IN_CATEGORY_category_header.csv,../import/business_IN_CATEGORY_category.csv" --multiline-fields=true --skip-bad-relationships

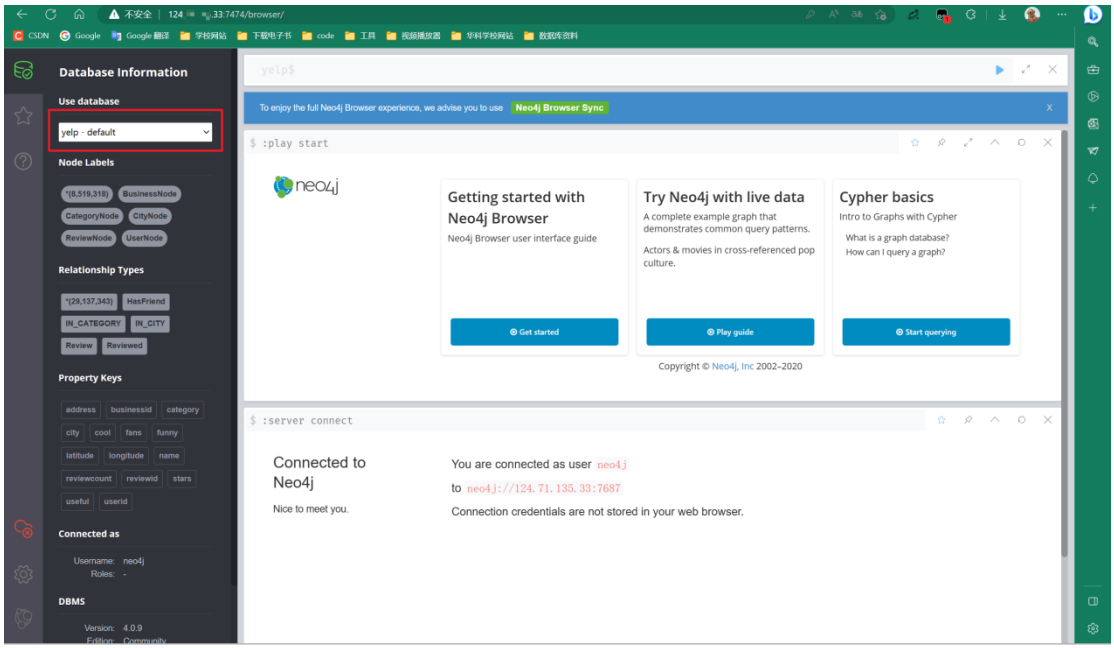
```
root@vector:~/neo4j-community-4.0.9/bin# ./neo4j-admin import --id-type=STRING --database=yelp --nod
es=UserNode="../import/user_header.csv,../import/user.csv" --nodes=ReviewNode="../import/review_head
er.csv,../import/review.csv" --nodes=BusinessNode="../import/business_header.csv,../import/BusinessN
ode.csv" --nodes=CityNode="../import/city_header.csv,../import/CityNode.csv" --nodes=CategoryNode=".
./import/category_header.csv,../import/CategoryNode.csv" --relationships=HasFriend="../import/user_F
RIENDS_user_header.csv,../import/user_FRIENDS_user.csv" --relationships=Review="../import/user_WROTE
_review_header.csv,../import/user_WROTE_review.csv" --relationships=Reviewed="../import/review_REVIE
WS_business_header.csv,../import/review_REVIEWS_business.csv" --relationships=IN_CITY="../import/bus
iness_IN_CITY_city_header.csv,../import/business_IN_CITY_city.csv" --relationships=IN_CATEGORY="../i
mport/business_IN_CATEGORY_category_header.csv,../import/business_IN_CATEGORY_category.csv" --multil
ine-fields=true --skip-bad-relationships
Neo4j version: 4.0.9
Importing the contents of these files into /root/neo4j-community-4.0.9/data/databases/yelp:
Nodes:
  [CategoryNode]:
  /root/neo4j-community-4.0.9/bin/../import/category_header.csv
  /root/neo4j-community-4.0.9/bin/../import/CategoryNode.csv

  [CityNode]:
  /root/neo4j-community-4.0.9/bin/../import/city_header.csv
  /root/neo4j-community-4.0.9/bin/../import/CityNode.csv

  [ReviewNode]:
  /root/neo4j-community-4.0.9/bin/../import/review_header.csv
  /root/neo4j-community-4.0.9/bin/../import/review.csv

  [UserNode]:
  /root/neo4j-community-4.0.9/bin/../import/user_header.csv
  /root/neo4j-community-4.0.9/bin/../import/user.csv

  [BusinessNode]:
```

```
..... ..... ..... ..... .....
IMPORT DONE in 22m 6s 936ms.
Imported:
  8519318 nodes
  29137343 relationships
  24739171 properties
Peak memory usage: 264.4MiB
There were bad entries which were skipped and logged into /root/neo4j-community-4.0.9/bin/import.report
```

数据导入成功后，修改 neo4j.conf 的默认 database 为 yelp：

```
#*****************************************************************
# Neo4j configuration
#
# For more details and a complete list of settings, please see
# https://neo4j.com/docs/operations-manual/4.0/reference/configuration-settings/
#*****************************************************************

# The name of the default database
dbms.default_database=yelp

# Paths of directories in the installation.
#dbms.directories.data=data
#dbms.directories.plugins=plugins
#dbms.directories.logs=logs
#dbms.directories.lib=lib
#dbms.directories.run=run
#dbms.directories.transaction.logs.root=data/transactions

# This setting constrains all `LOAD CSV` import files to be under the `import` directory. Remove or comment it out to
# allow files to be loaded from anywhere in the filesystem; this introduces possible security problems. See the
# `LOAD CSV` section of the manual for details.
dbms.directories.import=import

# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
```

执行./neo4j restart 命令重启数据库，然后./neo4j console 启动服务，在浏览器中重新登录数据库：



四、安装 MySQL

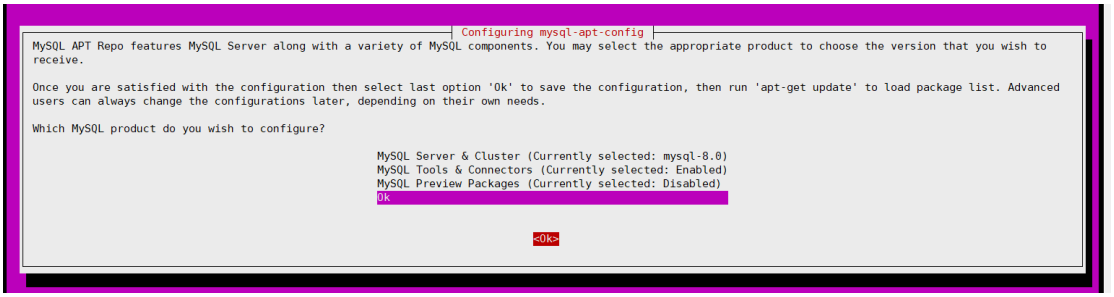
使用以下命令即可进行 mysql8.0 版本安装：

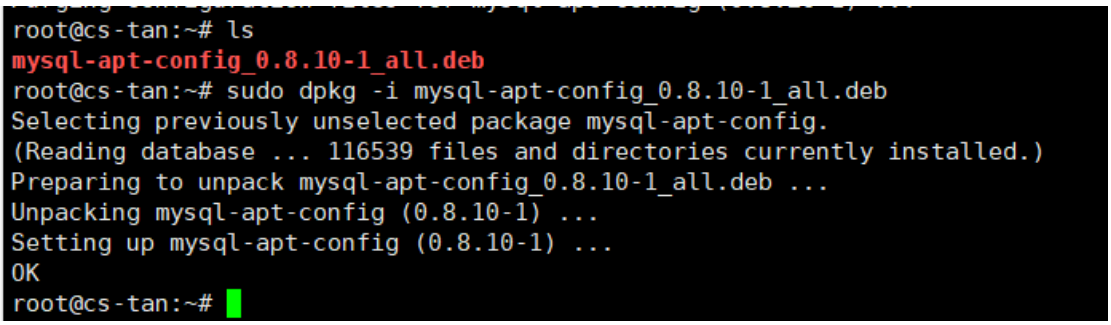
先下载.deb 文件

wget -c https://dev.mysql.com/get/mysql-apt-config_0.8.10-1_all.deb

找到下载好的 mysql-apt-config_0.8.10-1_all.deb 文件

使用 sudo dpkg -i mysql-apt-config_0.8.10-1_all.deb 之后出现设置界面



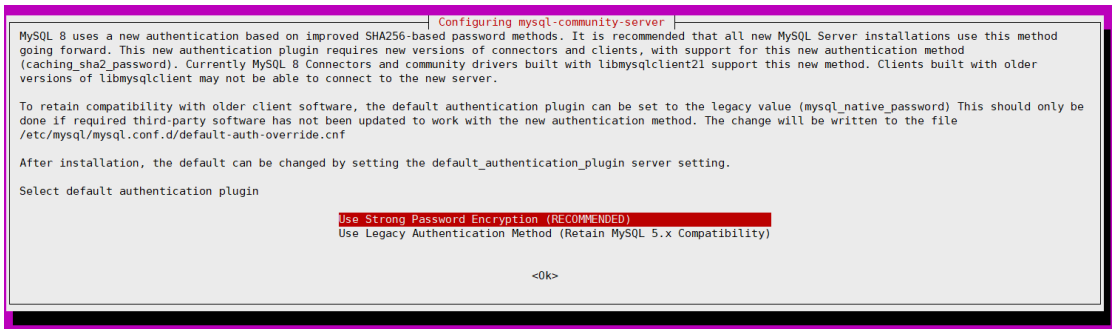


选择 ok 之后回车退出 ,再使用 sudo apt-get update #更新软件源

sudo apt-get install mysql-server #安装 mysql8.0

会出现两次选项都选择 y 然后进入安装过程

安装过程中会出现设置 MySQL 密码界面和确认密码界面,输入自己的密码即可
然后出现以下界面,选择默认的推荐选项 ok 即可完成安装



使用 mysql -u root -p 再键入密码即可进入 mysql,安装完成.

```
done!
update-alternatives: using /var/lib/mecab/dic/ipadic-utf8 to provide /var/lib/mecab/dic/debian (mecab-dictionary) in auto mode
Setting up mysql-server (8.0.25-1ubuntu16.04) ...
Processing triggers for libc-bin (2.23-0ubuntu11.3) ...
root@cs-tan:~#  mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.25 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

如下命令是启动和关闭 mysql 服务器：

sudo service mysql start

sudo service mysql stop

启动服务器后使用 mysql -u root -p 命令，后跟密码即可使用 mysql 数据库。

MySQL 数据导入:

将 data_for_mysql.zip 解压到 data_for_mysql 文件夹中

```
root@cs-tan:~/data# ls
data_for_mongo  data_for_mysql  data_for_mysql.zip  data_for_neo4j
root@cs-tan:~/data# cd data_for_mysql
root@cs-tan:~/data/data_for_mysql# unzip ../data_for_mysql.zip
Archive:  ../data_for_mysql.zip
  inflating: test_business.sql
  inflating: test_tip.sql
  inflating: test_user.sql
root@cs-tan:~/data/data_for_mysql# ls
test_business.sql  test_tip.sql  test_user.sql
```

输入 mysql -u root -p 然后键入你的密码进入 mysql

使用 create database test; 和 use test; 切换到 test 数据库

依次输入: source /root/data/data_for_mysql/test_business.sql

source /root/data/data_for_mysql/test_user.sql

source /root/data/data_for_mysql/test_tip.sql

完成三张表数据的导入,最后三张表中数据行数如下

```
mysql> select count(*) from tip;
+----------+
| count(*) |
+----------+
|   908915 |
+----------+
1 row in set (0.22 sec)

mysql> select count(*) from user;
+----------+
| count(*) |
+----------+
|  1987897 |
+----------+
1 row in set (0.26 sec)

mysql> select count(*) from business;
+----------+
| count(*) |
+----------+
|   150346 |
+----------+
1 row in set (0.06 sec)
```

五、MongoDB 创建分片集群

需要在华为云购置三台服务器，可在不同区域。以我的三台服务器为例，三台服务器的弹性公网 IP 地址如下：

server1：124.71.135.33

server2：123.249.45.70

server3：123.249.39.176

1. 准备

在华为云为三台服务器都添加入方向规则和出方向规则，否则后面会出现解析 IP 错误，添加方法同本手册第三部分第 3 部分。之后可在本机 ping 服务器判断是否添加规则成功。

如图所示，则添加规则成功。

```
C:\Users\Vector>ping 124.71.135.33

正在 Ping 124.71.135.33 具有 32 字节的数据:
来自 124.71.135.33 的回复: 字节=32 时间=27ms TTL=49
来自 124.71.135.33 的回复: 字节=32 时间=26ms TTL=49
来自 124.71.135.33 的回复: 字节=32 时间=27ms TTL=49
来自 124.71.135.33 的回复: 字节=32 时间=30ms TTL=49

124.71.135.33 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 26ms, 最长 = 30ms, 平均 = 27ms
```

三台服务器创建数据日志存放目录：

Server1:

su - mongodb

mkdir /usr/local/mongodb

cd /usr/local/mongodb

mkdir -p data/shard11

mkdir -p data/shard21

mkdir -p data/config

touch data/shard11.log

touch data/shard21.log

Server2:

su - mongodb

mkdir /usr/local/mongodb

cd /usr/local/mongodb

mkdir -p data/shard12

mkdir -p data/shard22

mkdir -p data/config

touch data/shard12.log

touch data/shard22.log

Server3:

su – mongodb

mkdir /usr/local/mongodb

cd /usr/local/mongodb

mkdir -p data/shard13

mkdir -p data/shard23

mkdir -p data/config

touch data/shard13.log

touch data/shard23.log

如下是正常的：

```
root@vector:~# su - mongodb
No directory, logging in with HOME=/
root@vector:~#
```

2. 配置 Shard1 的 replica sets

Sever1:

mongod --shardsvr --replSet shard1 --port 27017 --dbpath /usr/local/mongodb/data/shard11 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard11.log --logappend --bind_ip=0.0.0.0 --fork

Server2:

mongod --shardsvr --replSet shard1 --port 27017 --dbpath /usr/local/mongodb/data/shard12 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard12.log --logappend --bind_ip=0.0.0.0 --fork

Server3:

mongod --shardsvr --replSet shard1 --port 27017 --dbpath /usr/local/mongodb/data/shard13 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard13.log --logappend --bind_ip=0.0.0.0 --fork

Server1 执行示例如下：

```
root@vector:/usr/local/mongodb# mongod --shardsvr --replSet shard
1 --port 27017 --dbpath /usr/local/mongodb/data/shard11 --oplogSi
ze 2048 --logpath /usr/local/mongodb/data/shard11.log --logappend
 --bind_ip=0.0.0.0 --fork
about to fork child process, waiting until server is ready for co
nnections.
forked process: 5643
child process started successfully, parent exiting
```

之后如果遇到这种出错：

```
root@vector:/usr/local/mongodb# mongod --shardsvr --replSet shard
1 --port 27017 --dbpath /usr/local/mongodb/data/shard11 --oplogSi
ze 2048 --logpath /usr/local/mongodb/data/shard11.log --logappend
 --bind_ip=0.0.0.0 --fork
about to fork child process, waiting until server is ready for co
nnections.
forked process: 6111
ERROR: child process failed, exited with error number 48
```

需要关闭 mongod 进程：

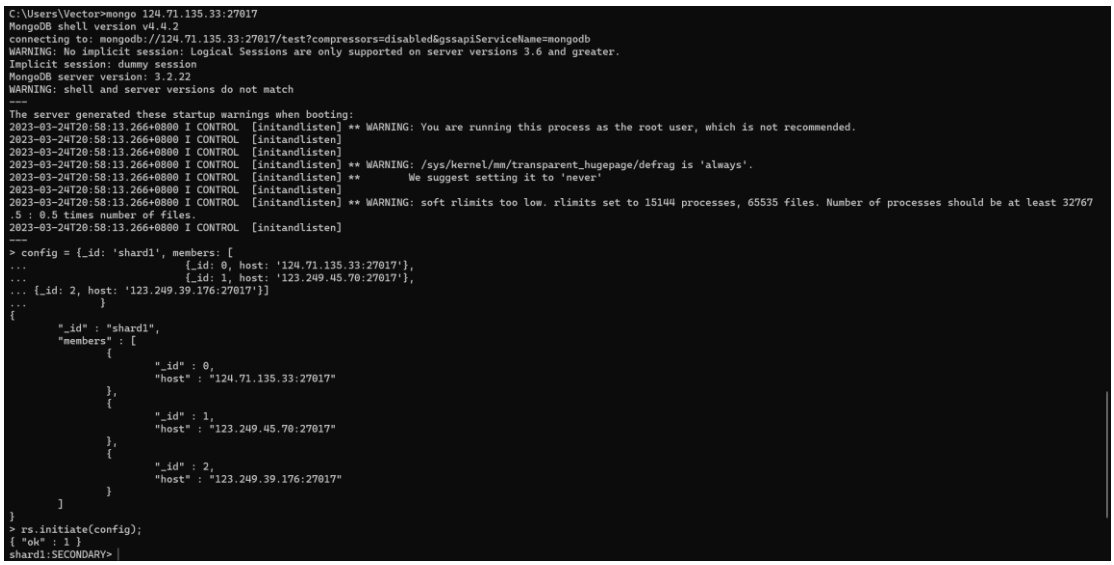
查看 mongo 相关进程：ps aux | grep mongo

关闭 mongodb 全部进程：killall mongod

3. 初始化 replica set

可在本机，也可在服务器用 mongo 连接其中一个 mongod，执行:

```
mongo 124.71.135.33:27017
config = {
   _id: 'shard1',
   members: [
      {_id: 0, host: '124.71.135.33:27017'},
      {_id: 1, host: '123.249.45.70:27017'},
      {_id: 2, host: '123.249.39.176:27017'}]
}
rs.initiate(config);
```



4. 配置 shard2 的 replica sets

Server1:

mongod --shardsvr --replSet shard2 --port 27018 --dbpath /usr/local/mongodb/data/shard21 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard21.log --logappend --bind_ip=0.0.0.0 --fork

Server2:

mongod --shardsvr --replSet shard2 --port 27018 --dbpath /usr/local/mongodb/data/shard22 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard22.log --logappend --bind_ip=0.0.0.0 --fork

Server3:

mongod --shardsvr --replSet shard2 --port 27018 --dbpath /usr/local/mongodb/data/shard23 --oplogSize 2048 --logpath /usr/local/mongodb/data/shard23.log --logappend --bind_ip=0.0.0.0 --fork

5. 初始化 replica set

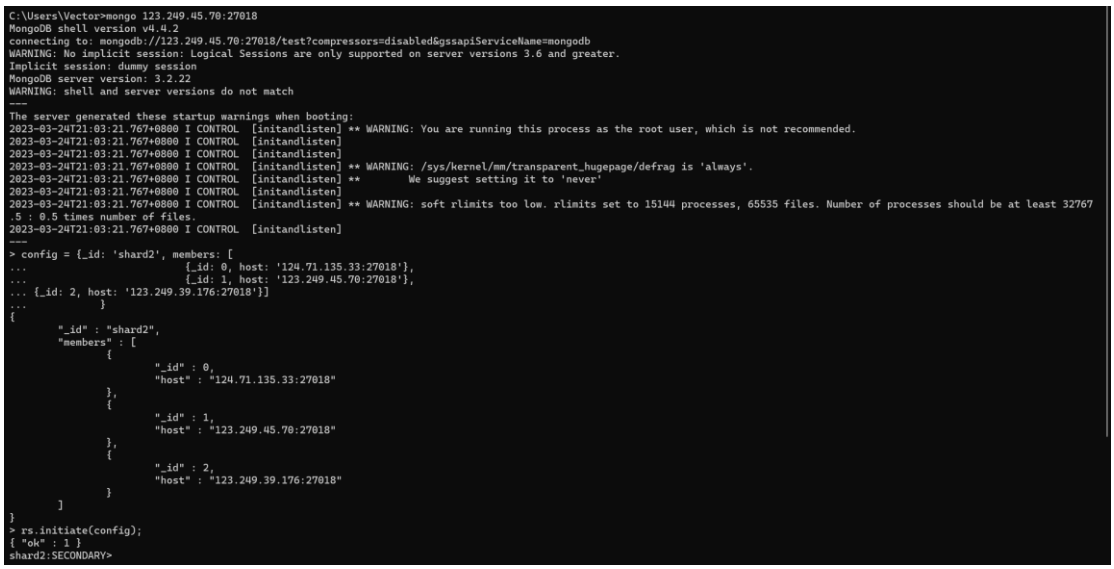
可在本机，也可在服务器用 mongo 连接其中一个 mongod，执行:

mongo 123.249.45.70:27018

config = {

_id: 'shard2',

members: [
{_id: 0, host: '124.71.135.33:27018'},
{_id: 1, host: '123.249.45.70:27018'},
{_id: 2, host: '123.249.39.176:27018'}]
}
rs.initiate(config);



6. 配置 config server

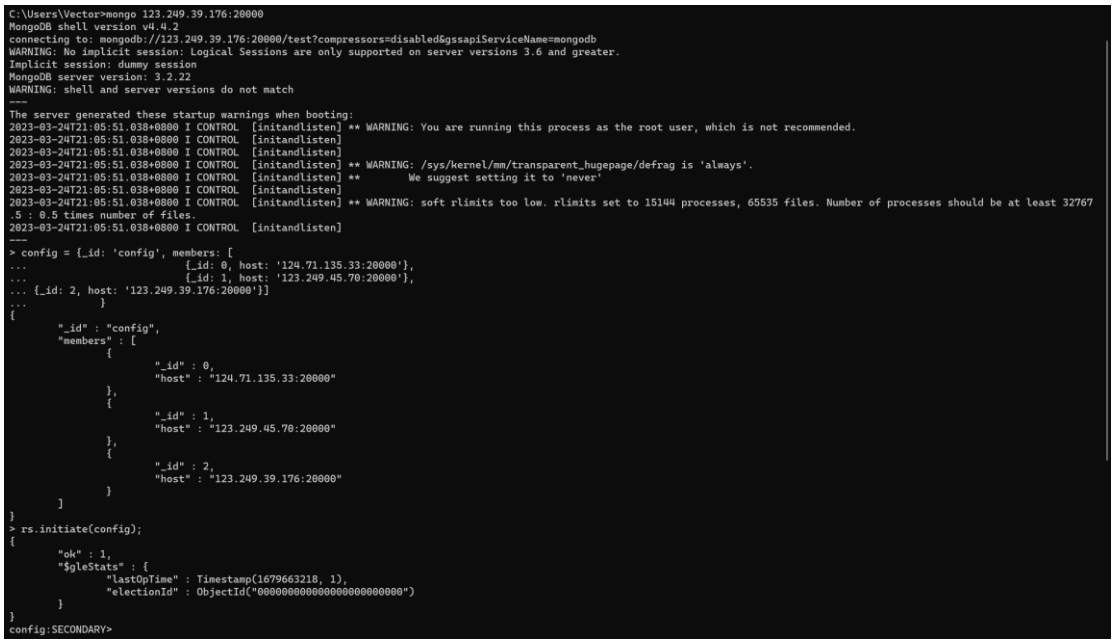
Server1、2、3 分别执行:

mongod --configsvr --replSet config --dbpath /usr/local/mongodb/data/config --port 20000 --logpath /usr/local/mongodb/data/config.log --logappend --bind_ip=0.0.0.0 --fork

可在本机，也可在服务器用 mongo 连接其中一个 mongod，执行:

mongo 123.249.39.176:20000

config = {
_id: 'config',
members: [
{_id: 0, host: '124.71.135.33:20000'},
{_id: 1, host: '123.249.45.70:20000'},
{_id: 2, host: '123.249.39.176:20000'}]
}
rs.initiate(config);



7. 配置 mongs

Server1、2、3 分别执行:

mongos --configdb config/124.71.135.33:20000,123.249.45.70:20000,123.249.39.176:20000 --port 30000 --logpath /usr/local/mongodb/data/mongos.log --logappend --bind_ip=0.0.0.0 --fork

8. 使用 mongos

可在本机，也可在服务器用 mongo 连接其中一个 mongod，执行:

mongo 124.71.135.33:20000

切换到 admin，添加分片：

use admin;

db.runCommand({addshard:"shard1/124.71.135.33:27017,121.36.200.236:27017,121.36.200.236:27017",name:"s1", maxsize:20480});

db.runCommand({addshard:"shard2/124.71.135.33:27018,121.36.200.236:27018,121.36.200.236:27018",name:"s2", maxsize:20480});

9. 激活数据库分片

创建数据库，激活数据库分片，如创建 testdb 数据库，则使用以下命令激活分片：

sh.enableSharding("testdb")

使用 sh.status()查看数据库当前情况，其中

primary：数据库主分片位置

partitioned：false 表示分片未开启/true 表示分片已开启。

```
mongos> sh.enableSharding("testdb")
{ "ok" : 1 }
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("641da0d0eee3f589725d001a")
  }
  shards:
        {  "_id" : "s1",  "host" : "shard1/123.249.39.176:27017,123.249.45.70:27017,124.71.135.33:27017" }
        {  "_id" : "s2",  "host" : "shard2/123.249.39.176:27018,123.249.45.70:27018,124.71.135.33:27018" }
  active mongoses:
        "3.2.22" : 3
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  unknown
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
        {  "_id" : "test",  "primary" : "s1",  "partitioned" : true }
        {  "_id" : "testdb",  "primary" : "s2",  "partitioned" : true }
mongos>
```

要使单个 collection 也分片存储，需要给 collection 指定一个分片 key，通过以下命令操作：

```
db.runCommand({
   shardcollection : “<namespace>”,
   key :<shardkeypatternobject>
});
```