Big Data Processing

Creaing Hadoop Cluster in Google Cloud and Usage

Maryam Mohammadi

Prof. Daniele Cesini Prof. Davide Salomoni
Unibo

Introduction:

Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
- Hadoop Common – Provides common Java libraries that can be used across all modules.

The procedure of mapreducing in Hadoop:

An input to a MapReduce in Big Data job is divided into fixed-size pieces called **input splits** Input split is a chunk of the input that is consumed by a single map

Mapping

This is the very first phase in the execution of map-reduce program. In this phase data in each split is passed to a mapping function to produce output values. In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>

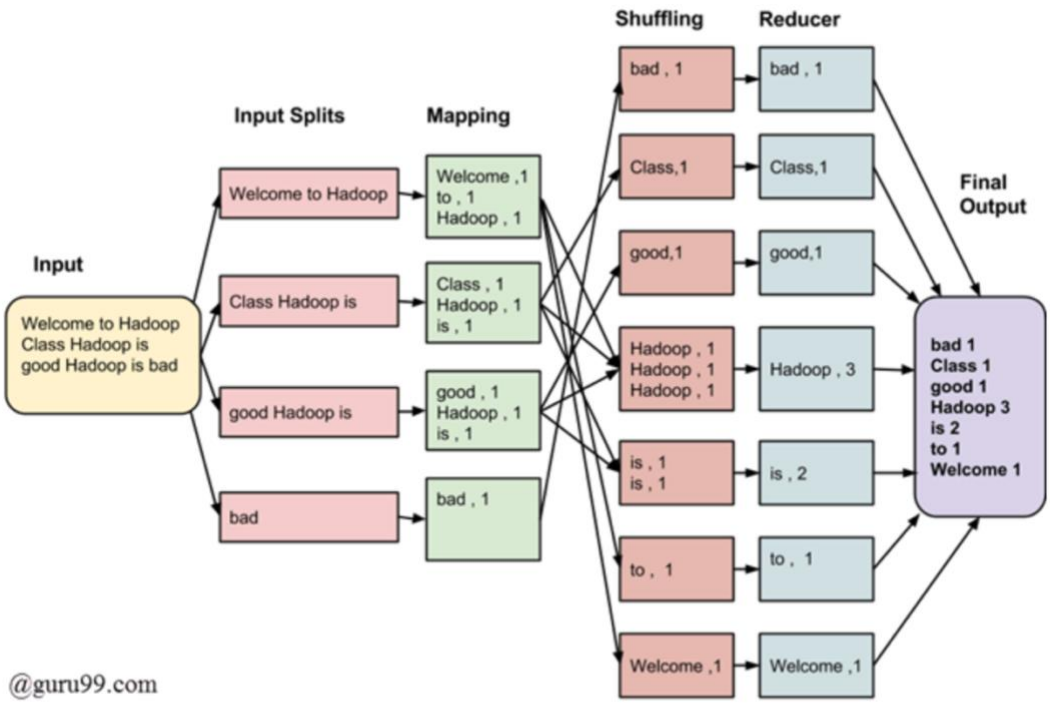
Shuffling

This phase consumes the output of Mapping phase. Its task is to consolidate the relevant records from Mapping phase output. In our example, the same words are clubed together along with their respective frequency.

Reducing

In this phase, output values from the Shuffling phase are aggregated. This phase combines values from Shuffling phase and returns a single output value. In short, this phase summarizes the complete dataset.

In our example, this phase aggregates the values from Shuffling phase i.e., calculates total occurrences of each word.



MapReduce Architecture

1)create small hadoop cluster in google cloud

Update the package list for upgrades for packages that need upgrading , as well as new packages that have just come to repsitories

sudo apt-get update
installing java
sudo apt-get install default-jdk
java -version

finding the path that java has been installed

sudo update-alternatives --config java

add a dedicated Hadoop user

sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
sudo adduser hduser sudo

checking if we create the Hadoop group and hduser
groups hduser

the Hadoop control scripts rely on ssh to perform cluster-wide operations. For example there is a script for stopping and starting all daemons in the clsuters. To work seamlessly, ssh needs to be setup to allow password-less login for the Hadoop user from machine in the cluster. The simplest way to achive this is to generate a public/private key pair and it will be shared across the cluster.
Hadoop requires SSH to manage its node. Remote machines plus your local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost for the hduser we created in the earlier.
We have to generate an SSH key for hduser user.
sudo apt-get install ssh
which ssh
which sshd

Hadoop uses SSH to access its nodes which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands.
su hduser
ssh-keygen -t rsa -P ""
the following command adds the newly createad key to the list of authorized keys so that Hadoop can use ssh without prompting for a password
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys

checking the ssh

ssh localhost

move the Hadoop installation to the /usr/local/Hadoop directory. So we should create the directory first :

sudo mkdir -p /usr/local/Hadoop

installing hadoop
wget https://archive.apache.org/dist/hadoop/core/hadoop-2.6.5/hadoop-2.6.5.tar.gz

tar xvzf hadoop-2.6.5.tar.gz
ls
cd hadoop-2.6.5/

now move to the folder, where your Hadoop downloaded is available and execute the following

sudo mv * /usr/local/hadoop/
cd ..

set read/write permission

sudo chown -R hduser:hadoop /usr/local/hadoop

==========
Now in this step we wanna set up configuration fiels: before editing the .bashrc file in hdusers home directory, we need to find the path where java has been installed to set the JAVA_HOME environemtn variable
sudo nano .bashrc

 # -- HADOOP ENVIRONMENT VARIABLES START -- #
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export
HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
export HADOOP_HOME_WARN_SUPRESS=1
export HADOOP_ROOT_LOGGER="WARN,DRFA"

 # -- HADOOP ENVIRONMENT VARIABLES END -- #

==========

source ~/.bashrc

==========
sudo nano /usr/local/hadoop/etc/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
==========
sudo mkdir -p /app/hadoop/tmp
sudo chown hduser:hadoop /app/hadoop/tmp

==========

sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:54310</value>
<description>the name of the default file system. a URI whose scheme and authority determine file system implementation</description>
</property>
</configuration>
==========

stat -c "%a %n" /usr/local/hadoop
sudo chmod 777 /usr/local/hadoop

cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml

==========
sudo nano /usr/local/hadoop/etc/hadoop/mapred-site.xml

<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:54311</value>
</property>
</configuration>
==========

sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
sudo chown -R hduser:hadoop /usr/local/hadoop_store

==========
sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml

<configuration>
<property>

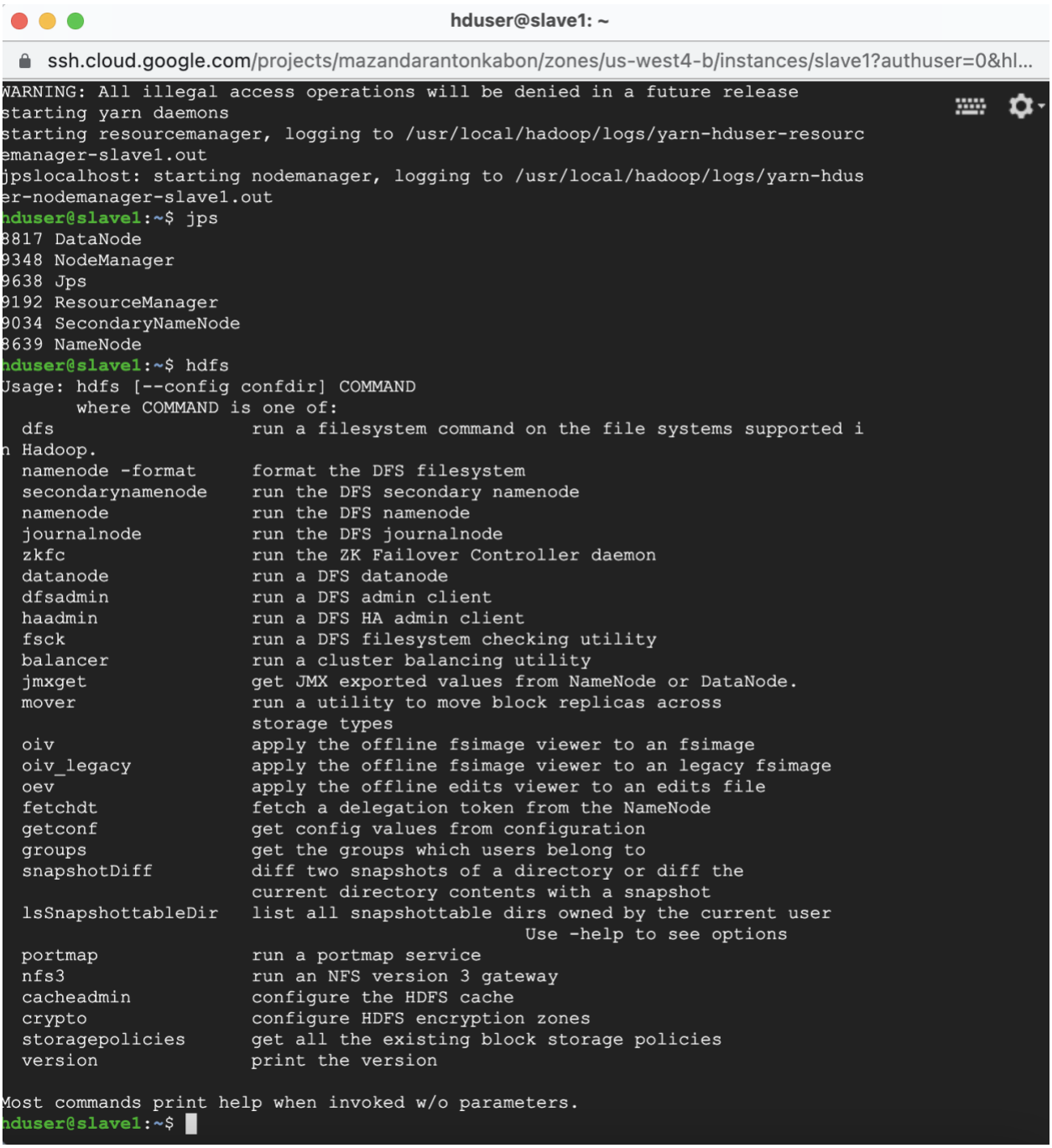
<name>dfs.replication</name>
<value>1</value>
</property>

<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>
</configuration>

We execute this command just once before we start using Hadoop and if we execute again after haoop that has been used it will destroy all the data on the Hadoop file system

hadoop namenode -format
start-all.sh
for cheking running process in our Hadoop cluster we use jps command . jps stands for java virtural machine process status tool
jps

after running jps and hdfs for all of the nodes as following :


hduser@slave1: ~
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/slave1?authuser=0&hl...
WARNING: All illegal access operations will be denied in a future release
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourc
emanager-slave1.out
jpslocalhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hdus
er-nodemanager-slave1.out
hduser@slave1:~$ jps
8817 DataNode
9348 NodeManager
9638 Jps
9192 ResourceManager
9034 SecondaryNameNode
8639 NameNode
hduser@slave1:~$ hdfs
Usage: hdfs [--config confdir] COMMAND
where COMMAND is one of:
dfs run a filesystem command on the file systems supported i
n Hadoop.
namenode -format format the DFS filesystem
secondarynamenode run the DFS secondary namenode
namenode run the DFS namenode
journalnode run the DFS journalnode
zkfc run the ZK Failover Controller daemon
datanode run a DFS datanode
dfsadmin run a DFS admin client
haadmin run a DFS HA admin client
fsck run a DFS filesystem checking utility
balancer run a cluster balancing utility
jmxget get JMX exported values from NameNode or DataNode.
mover run a utility to move block replicas across
storage types
oiv apply the offline fsimage viewer to an fsimage
oiv_legacy apply the offline fsimage viewer to an legacy fsimage
oev apply the offline edits viewer to an edits file
fetchdt fetch a delegation token from the NameNode
getconf get config values from configuration
groups get the groups which users belong to
snapshotDiff diff two snapshots of a directory or diff the
current directory contents with a snapshot
lsSnapshottableDir list all snapshottable dirs owned by the current user
Use -help to see options
portmap run a portmap service
nfs3 run an NFS version 3 gateway
cacheadmin configure the HDFS cache
crypto configure HDFS encryption zones
storagepolicies get all the existing block storage policies
version print the version
Most commands print help when invoked w/o parameters.
hduser@slave1:~$

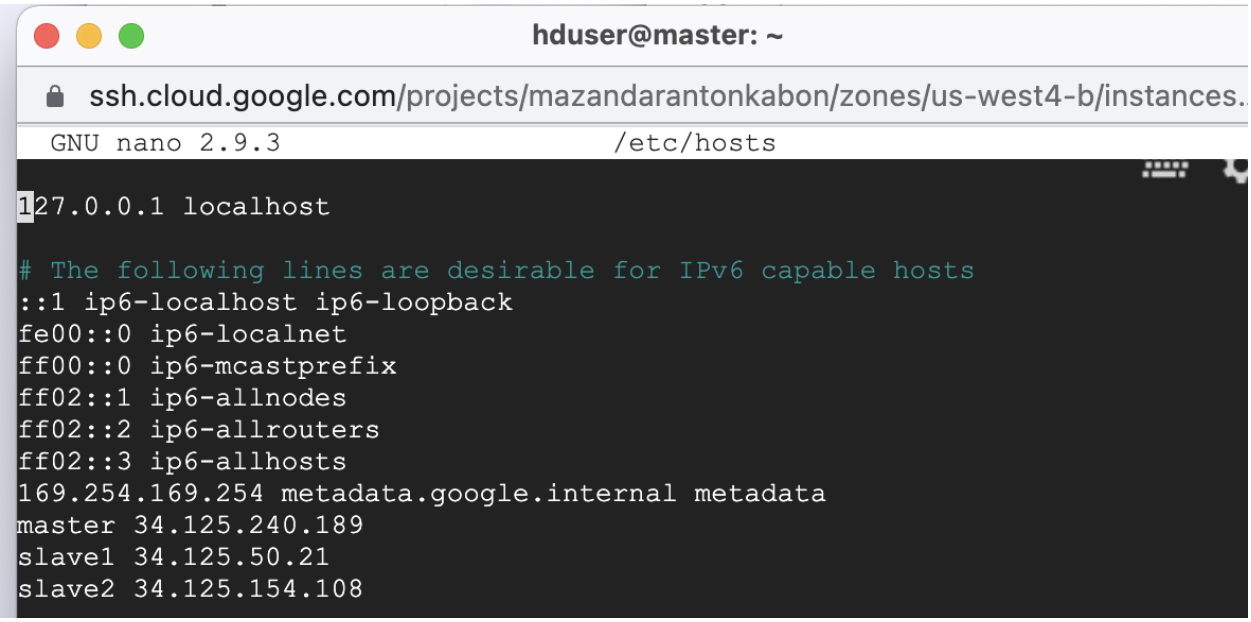

hduser@slave2: ~
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/sla...
WARNING: All illegal access operations will be denied in a future release
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resou
rcemanager-slave2.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduse
r-nodemanager-slave2.out
hduser@slave2:~$ jps
9344 ResourceManager
9186 SecondaryNameNode
8791 NameNode
8969 DataNode
9501 NodeManager
9790 Jps
hduser@slave2:~$ hdfs
Usage: hdfs [--config confdir] COMMAND
where COMMAND is one of:
dfs run a filesystem command on the file systems supported
in Hadoop.
namenode -format format the DFS filesystem
secondarynamenode run the DFS secondary namenode
namenode run the DFS namenode
journalnode run the DFS journalnode
zkfc run the ZK Failover Controller daemon
datanode run a DFS datanode
dfsadmin run a DFS admin client
haadmin run a DFS HA admin client
fsck run a DFS filesystem checking utility
balancer run a cluster balancing utility
jmxget get JMX exported values from NameNode or DataNode.
mover run a utility to move block replicas across
storage types
oiv apply the offline fsimage viewer to an fsimage
oiv_legacy apply the offline fsimage viewer to an legacy fsimage
oev apply the offline edits viewer to an edits file
fetchdt fetch a delegation token from the NameNode
getconf get config values from configuration
groups get the groups which users belong to
snapshotDiff diff two snapshots of a directory or diff the
current directory contents with a snapshot
lsSnapshottableDir list all snapshottable dirs owned by the current user
Use -help to see options
portmap run a portmap service
nfs3 run an NFS version 3 gateway
cacheadmin configure the HDFS cache
crypto configure HDFS encryption zones
storagepolicies get all the existing block storage policies
version print the version
Most commands print help when invoked w/o parameters.
hduser@slave2:~$


hduser@master: ~
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/ma...
WARNING: All illegal access operations will be denied in a future release
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-r
esourcemanager-master.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-h
duser-nodemanager-master.out
hduser@master:~$ jps
8580 NameNode
9300 NodeManager
8758 DataNode
9579 Jps
9133 ResourceManager
8975 SecondaryNameNode
hduser@master:~$ hdfs
Usage: hdfs [--config confdir] COMMAND
where COMMAND is one of:
dfs run a filesystem command on the file systems supported in
Hadoop.
namenode -format format the DFS filesystem
secondarynamenode run the DFS secondary namenode
namenode run the DFS namenode
journalnode run the DFS journalnode
zkfc run the ZK Failover Controller daemon
datanode run a DFS datanode
dfsadmin run a DFS admin client
haadmin run a DFS HA admin client
fsck run a DFS filesystem checking utility
balancer run a cluster balancing utility
jmxget get JMX exported values from NameNode or DataNode.
mover run a utility to move block replicas across
storage types
oiv apply the offline fsimage viewer to an fsimage
oiv_legacy apply the offline fsimage viewer to an legacy fsimage
oev apply the offline edits viewer to an edits file
fetchdt fetch a delegation token from the NameNode
getconf get config values from configuration
groups get the groups which users belong to
snapshotDiff diff two snapshots of a directory or diff the
current directory contents with a snapshot
lsSnapshottableDir list all snapshottable dirs owned by the current user
Use -help to see options
portmap run a portmap service
nfs3 run an NFS version 3 gateway
cacheadmin configure the HDFS cache
crypto configure HDFS encryption zones
storagepolicies get all the existing block storage policies
version print the version
Most commands print help when invoked w/o parameters.
hduser@master:~$

Second step : Connecting nodes with together:

First we should introduce all of nodes with together, so we should add these ip addresses in this pathway:

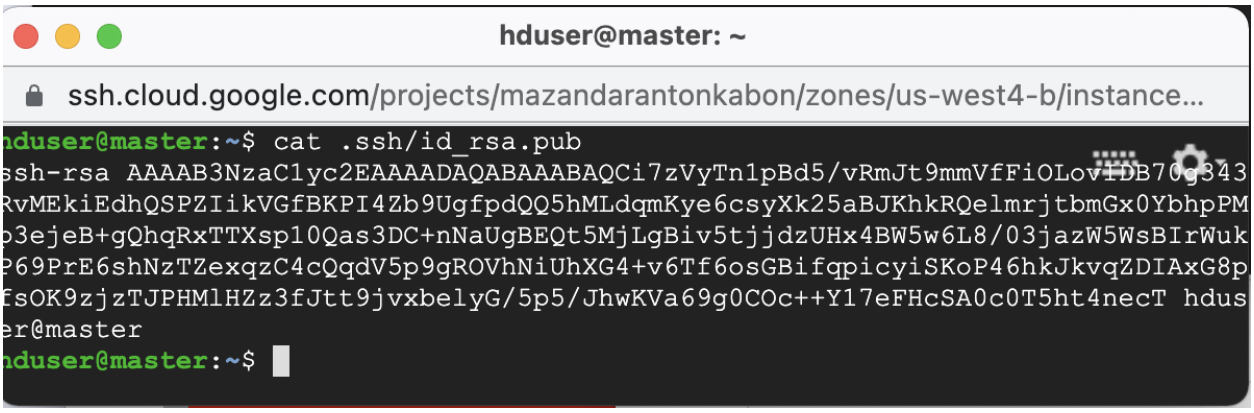
sudo nano /etc/hosts
master 34.125.240.189
slave1 34.125.50.21
slave2 34.125.154.108



Then if we want that slave1 and slave2 copy the master data , so we should give an access the slave1 and slave2 to master machine

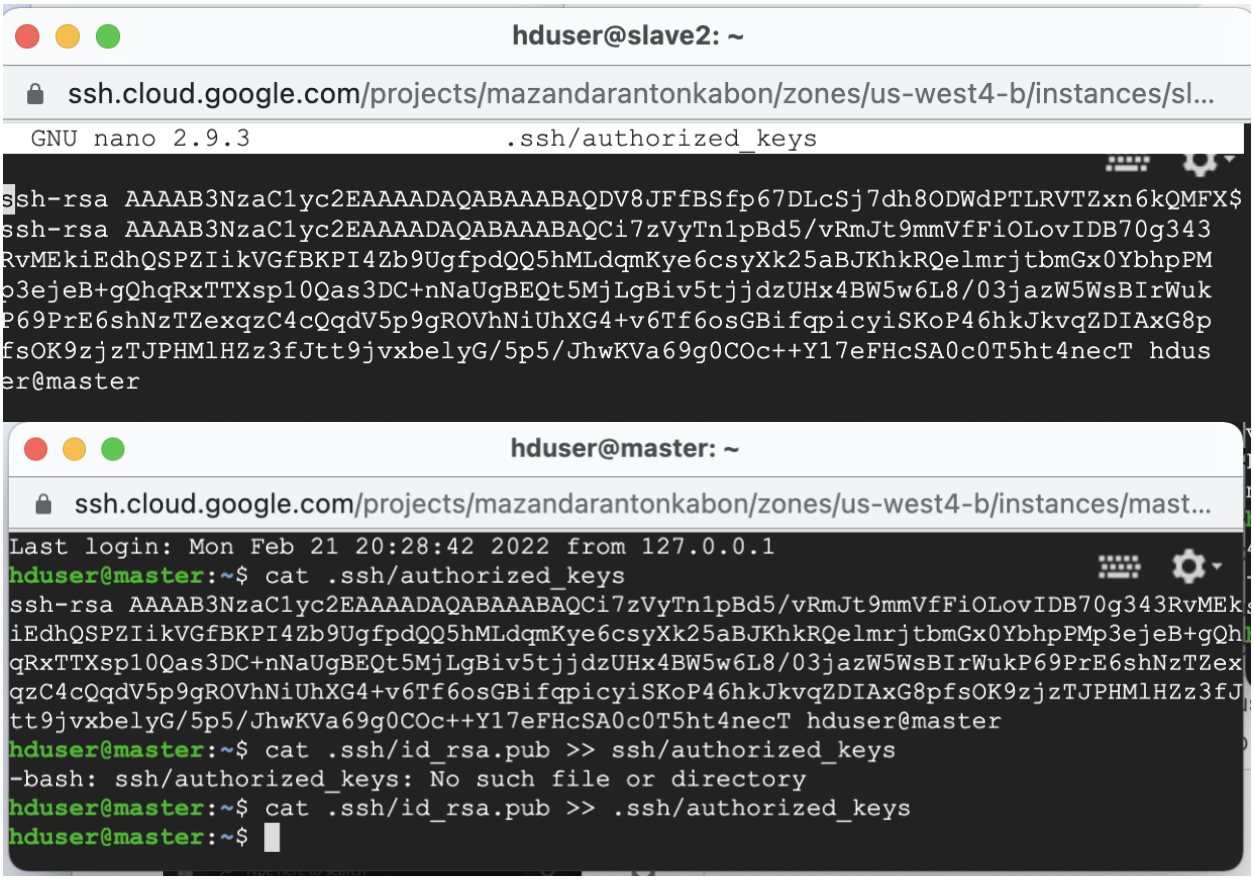
In the master machine , we get the public key to copy paste in authorized key of slave 1 and slave 2, so we should go to master console and run this command to get the public key

cat .ssh/id_rsa.pub



then in slave1 and slave2 machine we should excute this command to add the master node public key that we get from the previous step:

sudo nano .ssh/authorized_keys
and then we copy paste the puclic key of master machine

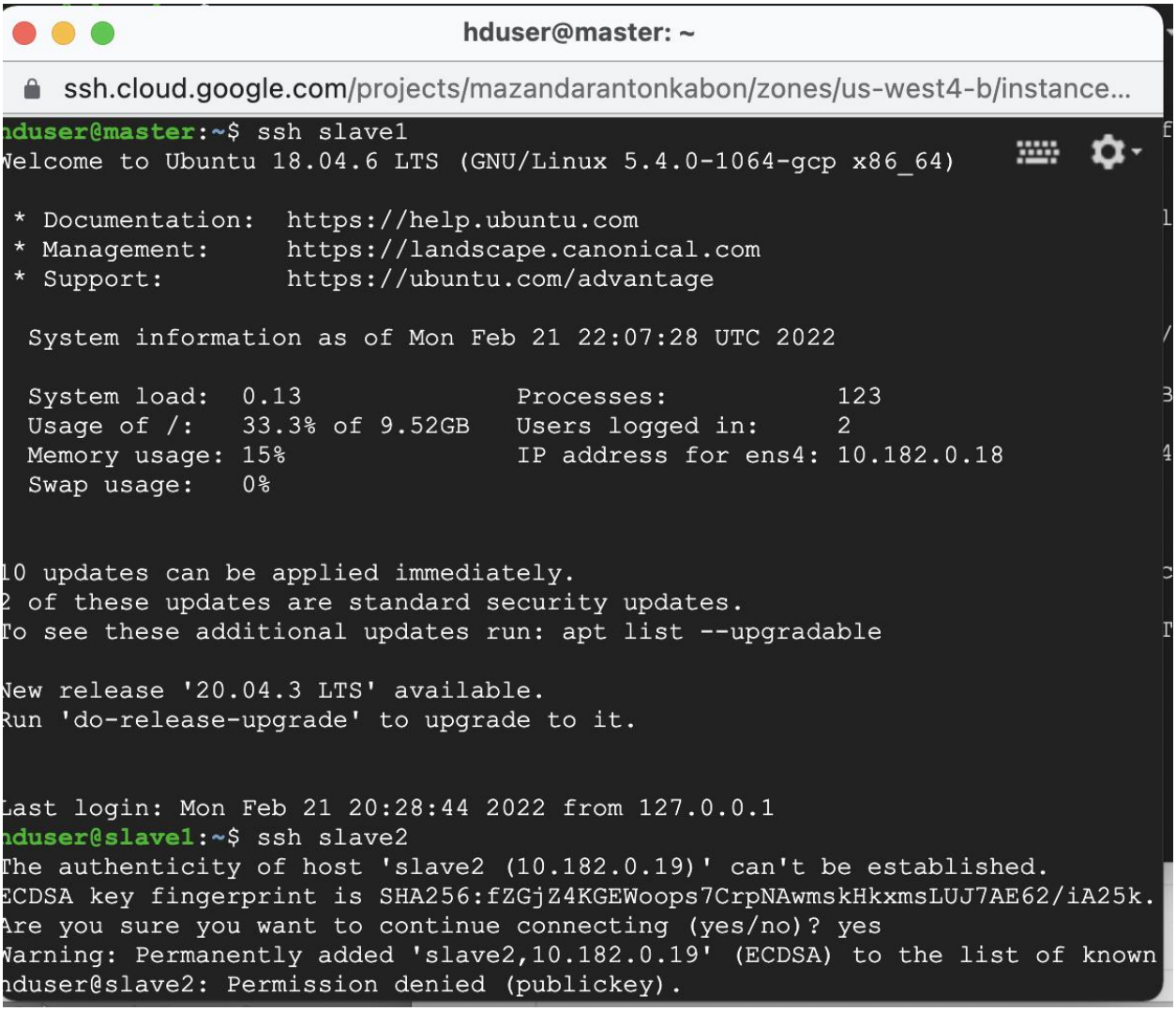


Running this command in master
ssh master
cat .ssh/authorized_keys

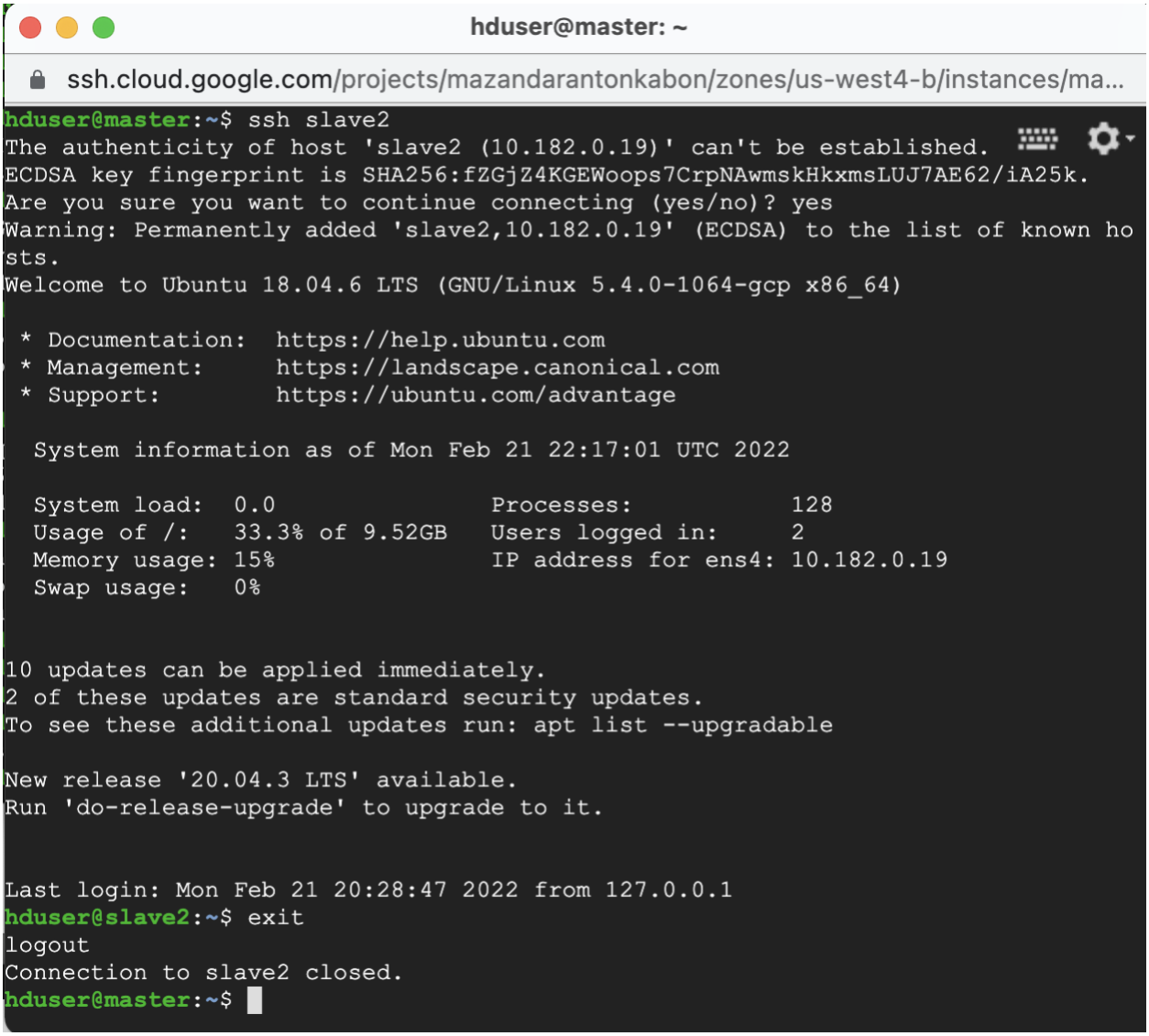
cat .ssh/id_rsa.pub >> ssh/authorized_keys

then we connect slave 1 and slave 2 to our master with this commands >

ssh slave1

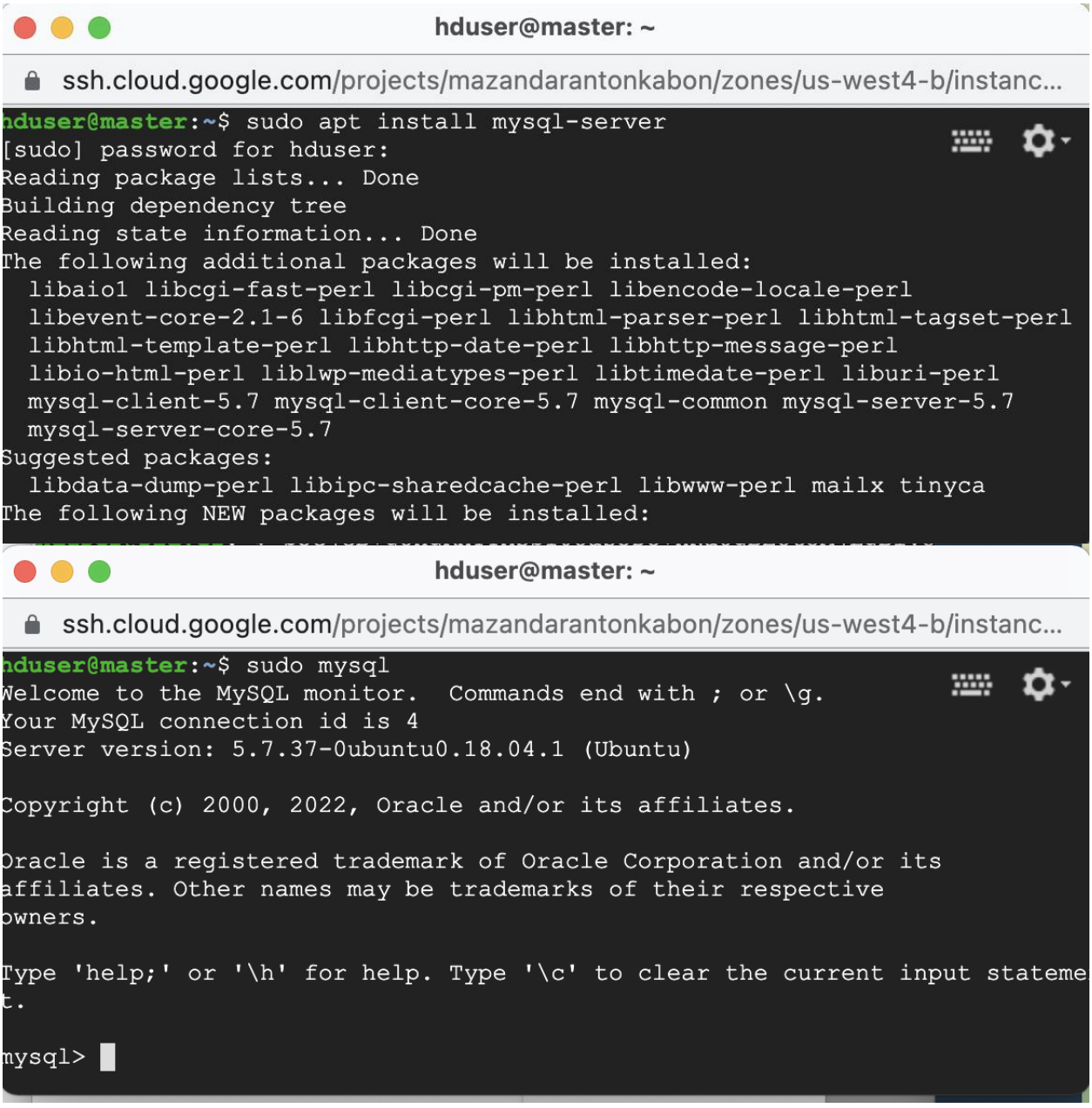


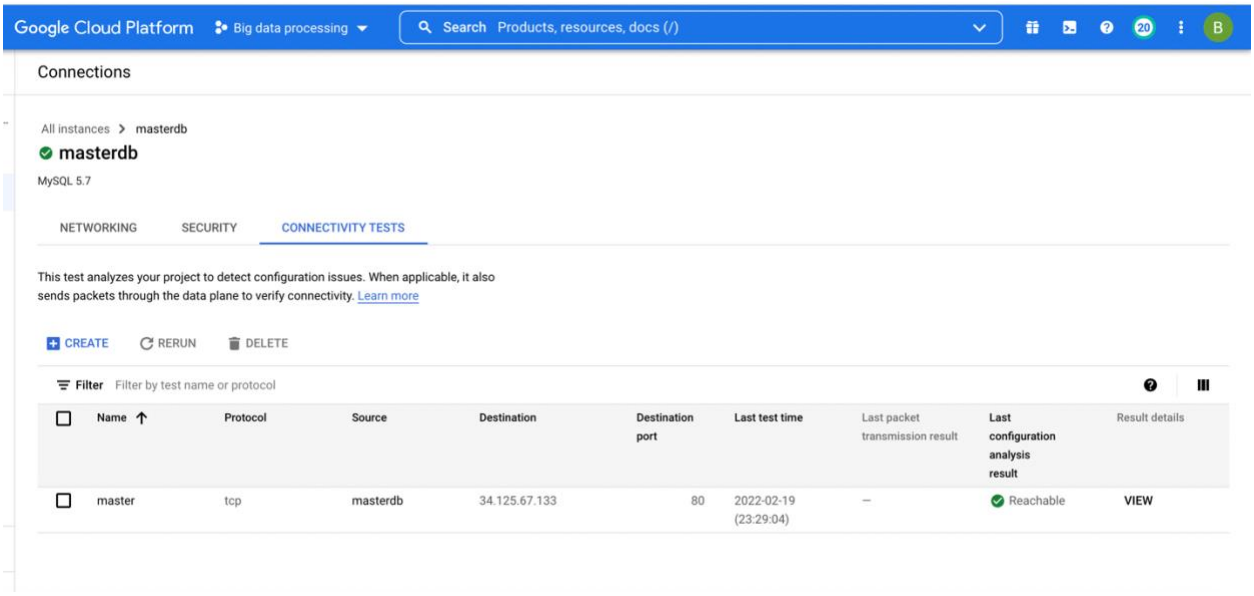
ssh slave2



the third step is creating bucket in google cloud and builing the sql database

---


hduser@master: ~
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instanc...
hduser@master:~$ sudo apt install mysql-server
[sudo] password for hduser:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
libaio1 libcgi-fast-perl libcgi-pm-perl libencode-locale-perl
libevent-core-2.1-6 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
libhtml-template-perl libhttp-date-perl libhttp-message-perl
libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl
mysql-client-5.7 mysql-client-core-5.7 mysql-common mysql-server-5.7
mysql-server-core-5.7
Suggested packages:
libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca
The following NEW packages will be installed:
hduser@master: ~
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instanc...
hduser@master:~$ sudo mysql
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.37-0ubuntu0.18.04.1 (Ubuntu)
Copyright (c) 2000, 2022, 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 stateme
t.
mysql>



Creating database:

Step1: Enable Compute Engine and Cloud SQL

1. Select a project, or create a new one.

   Big data processing

2. The necessary APIs are enabled:
   - Compute Engine API
   - Cloud SQL Admin API

Step2: Create a VM instance

To create a ubuntu VM, complete the following steps:

1. In the Cloud Console navigation menu, click **Compute Engine > VM instances**.

   You can see where it is by clicking the following button: Compute Engine chevron_right VM instances

2. Click **Create instance**.
3. On the following screen, give the VM a **Name**.
4. Click **Create** to generate your VM.

The **VM instances** page loads with your new VM added. After the VM has successfully started and is available, a green checkmark shows in the **Status** column next to the VM name.

Step3: Reserve a static IP address for your VM

Your VM needs a static IP address to connect to your MySQL instance. To set one up, complete the following instructions:

1. In the Cloud Console navigation menu, click **VPC network > External IP addresses**.

You can see where it is by clicking the following button: VPC network chevron_right External IP addresses

2. In the **In use by** column, find your VM's name.
3. In the **Type** column, check that your VM's external IP address is **Static**. If it's **Ephemeral**, change it:
    a. In the same row as your VM, click **Reserve**.
    b. Enter a **Name** and **Description** of your choice for the static address, then click **Reserve**.
4. Find your VM's external IP address in the **External Address** column, then copy it somewhere — you'll need it later.

Step4: Create a MySQL instance

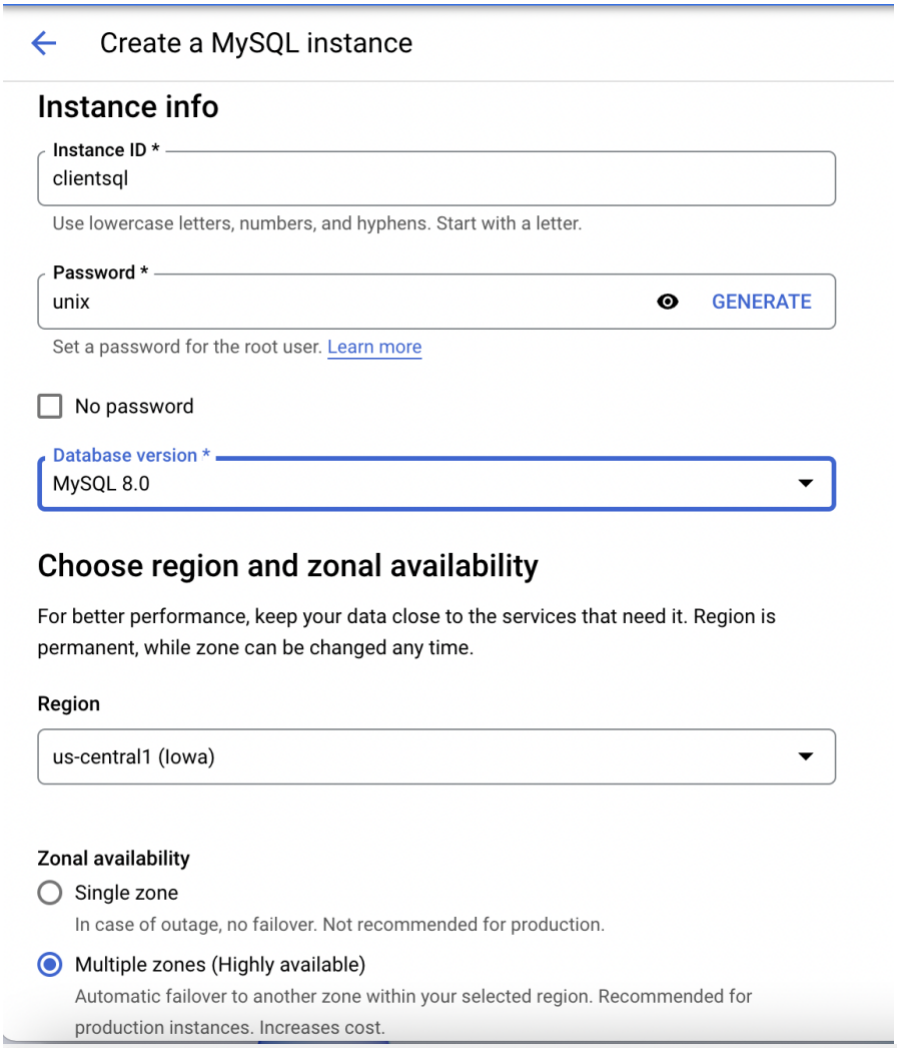
To create a MySQL instance, complete the following steps:

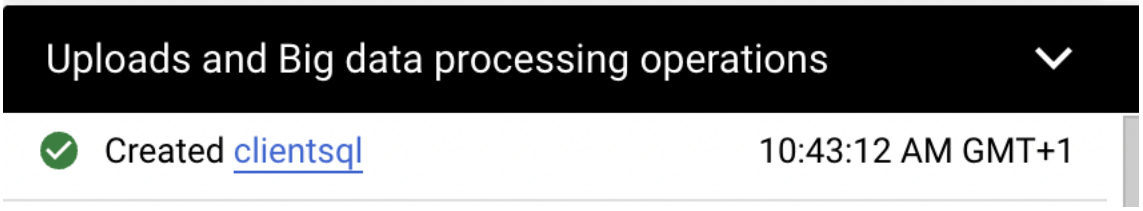
1. In the Cloud Console navigation menu, click **SQL**.

    You can see where it is by clicking the following button:

    SQL
2. Click **Create instance**.
3. Click **Choose MySQL**.
4. Enter an **Instance ID** and **Password** of your choice.
5. For the **Database version**, select **MySQL 8.0**.
6. Click **Create instance**.

After a short time the new instance's page shows, but the creation process might take a few minutes to complete. When it's finished, you can proceed to the next step.



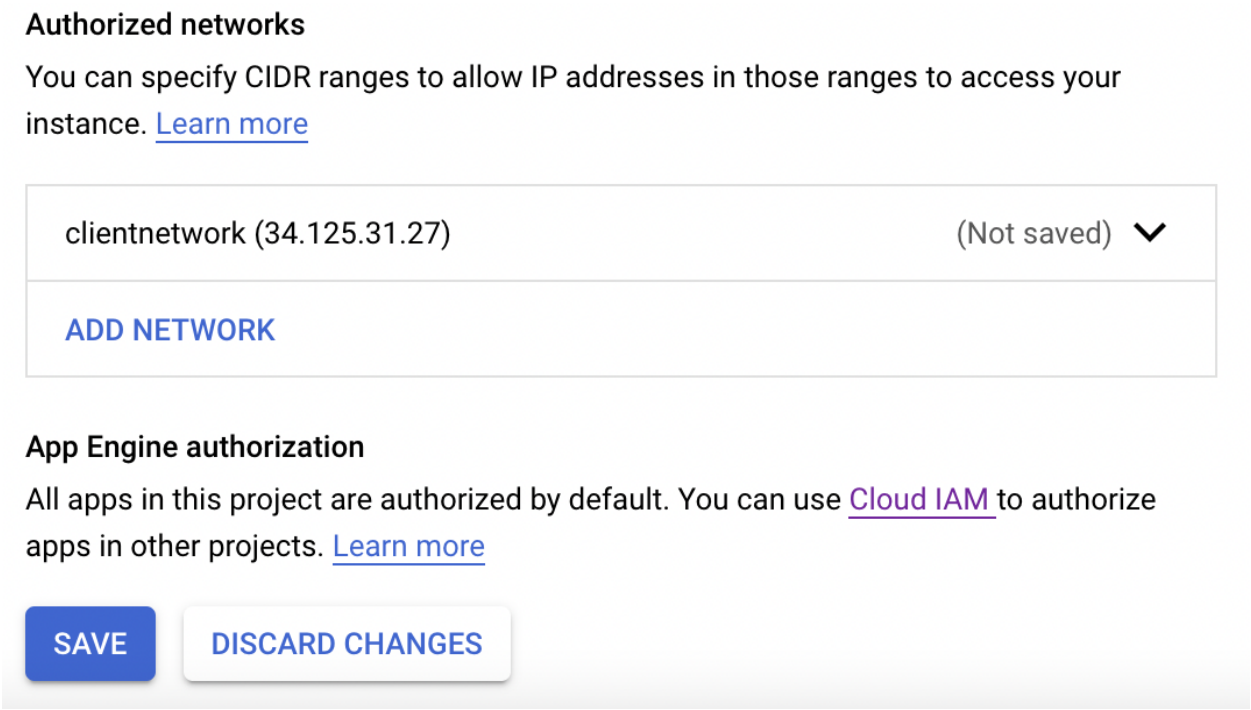


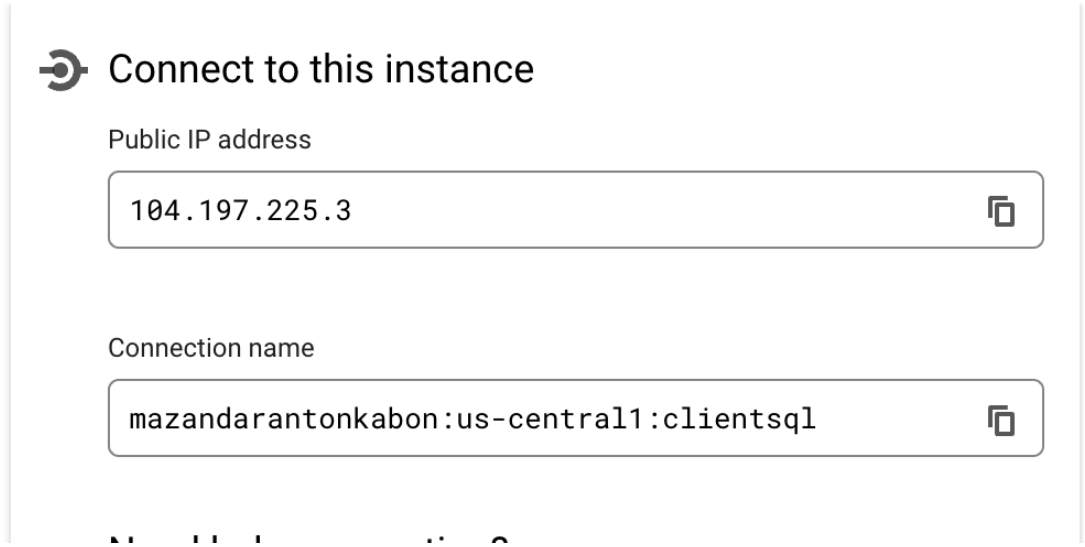
Step5: Authorize your VM to connect to your MySQL instance

After your MySQL instance has finished creating, add your VM's external IP as an authorized network:

1. Click **Connections**.
2. Click **Add network**.
3. Enter a **Name** of your choice, then in the **Network** box enter the static IP address of your VM.
4. Click **Done**, then click **Save**.
5. Click **Overview**.

6. In the **Connect to this instance** card, copy the **Public IP address** somewhere, as you'll need it later.
7.



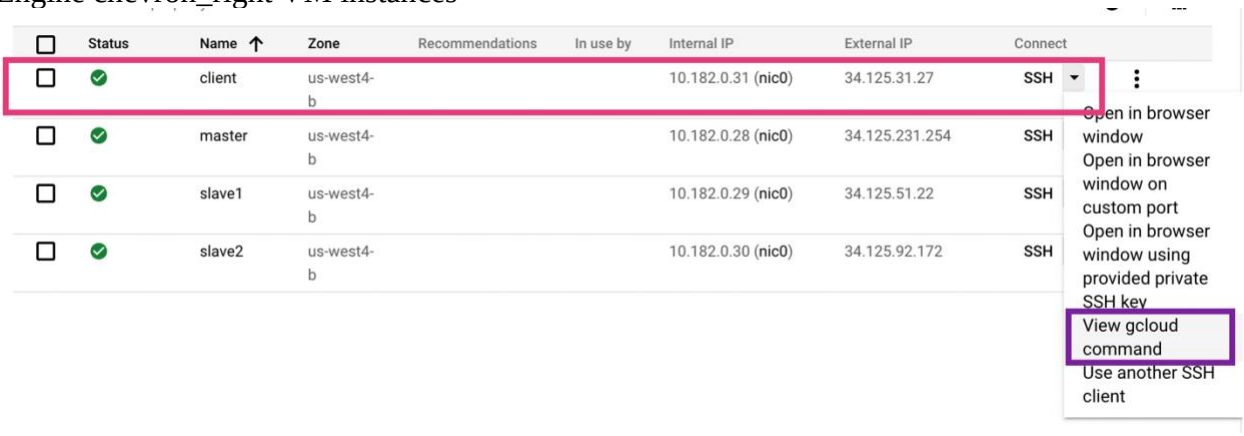


Step6: Install the MySQL client on your VM

Connect to your VM over SSH using the `gcloud` CLI:

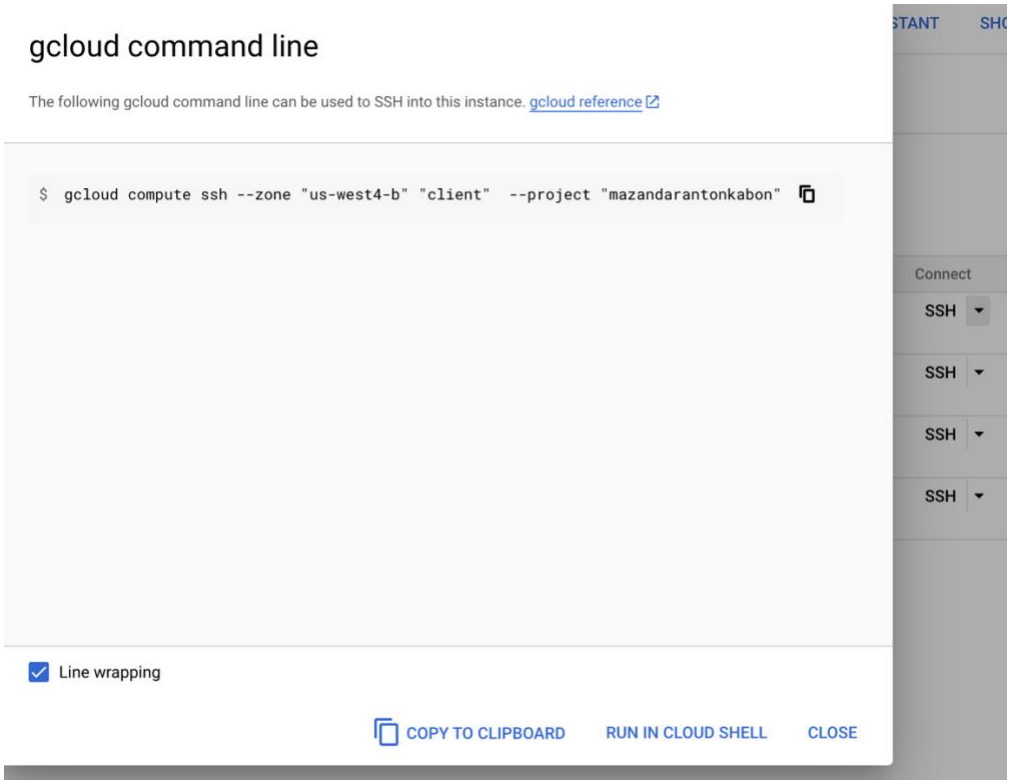
1. In the Cloud Console navigation menu, click **Compute Engine > VM instances**.

You can see where it is by clicking the following button: Compute Engine chevron_right VM instances



2. Find your VM instance in the **Instances** table. In the **Connect** column, click the **down arrow**, then click **View gcloud command**.

3. Click **Run in Cloud Shell**. After Cloud Shell has loaded, press `Enter` to run the command. If you're prompted to authorize Cloud Shell, do so.



4. If you haven't generated SSH keys before, you might be prompted to do so. If this is the case, type `y` to the prompt, then press `Enter` to continue. You can enter a passphrase if you like.

5. After you've connected to your VM, install the MySQL binaries:

```
sudo apt-get update
sudo apt-get install \
    default-mysql-server
```

Step 7: Connect to your MySQL instance

Now you're ready to connect to your MySQL instance from your VM.

Enter the following in your VM's terminal, replacing *<INSTANCE_IP>* with the public IP address of your MySQL instance and *<USERNAME>* with the appropriate username (default: root):

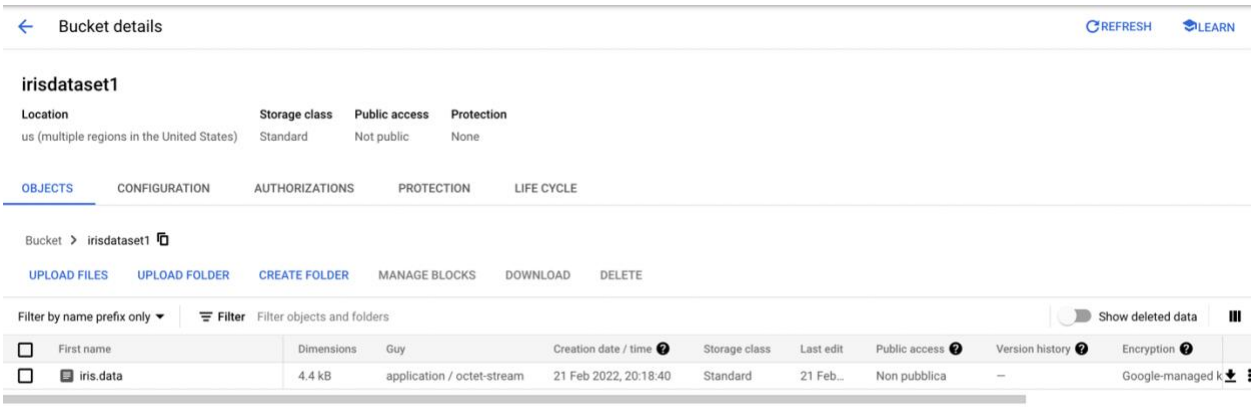
```
mysql -h 34.71.56.59 -u \
    root -p
```

After entering your password, you should see the MySQL prompt:

```
mysql>
```

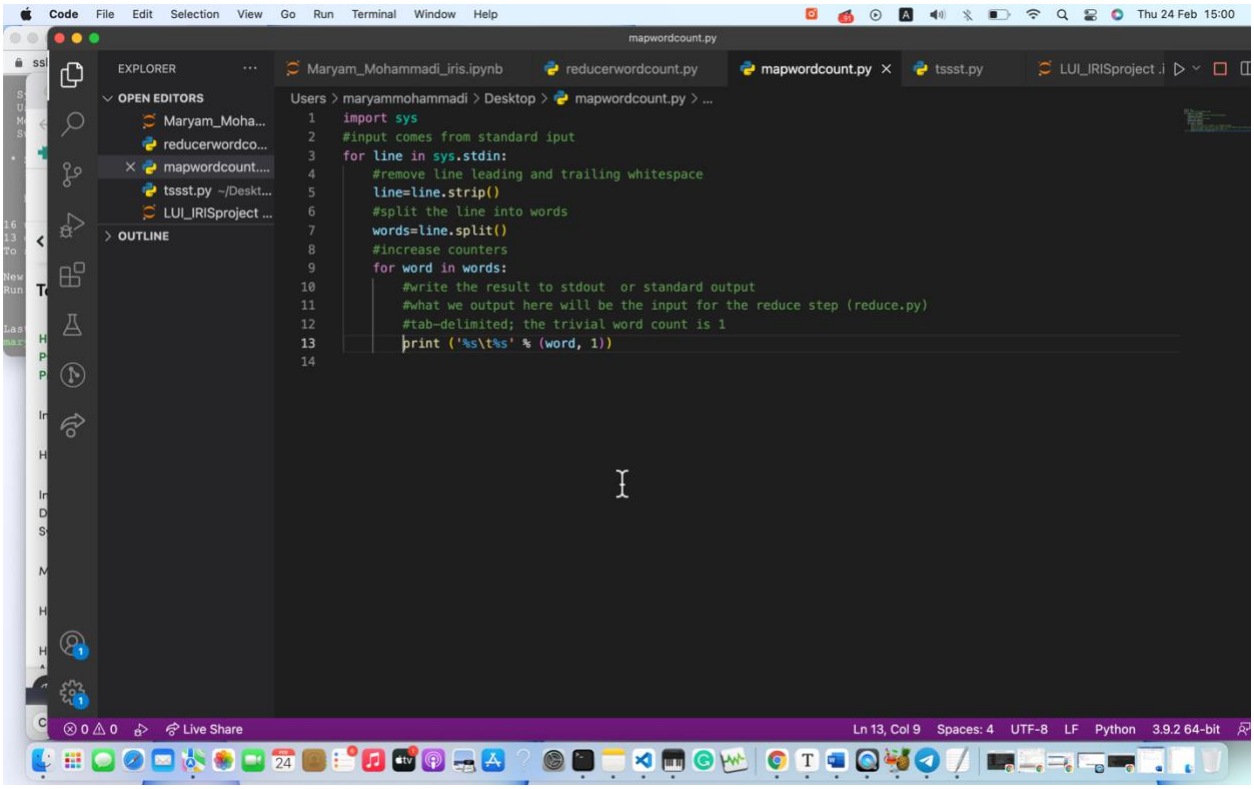
```
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
maryland_maryam22@client:~$ mysql -h 34.71.56.59 -u \
>     root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1690
Server version: 8.0.18-google (Google)

Copyright (c) 2000, 2022, 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>
```

Now I make a irisdataset1 bucket in google cloud and I upload the iris.data

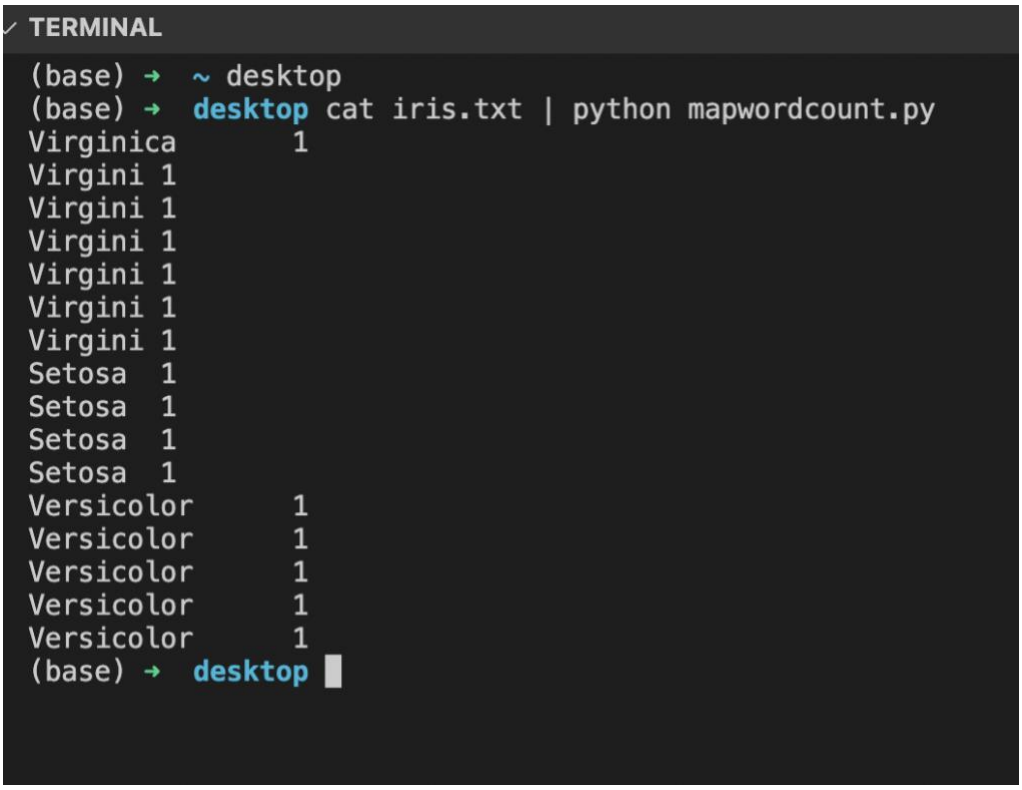


Part 4: Mapreducing with python programming

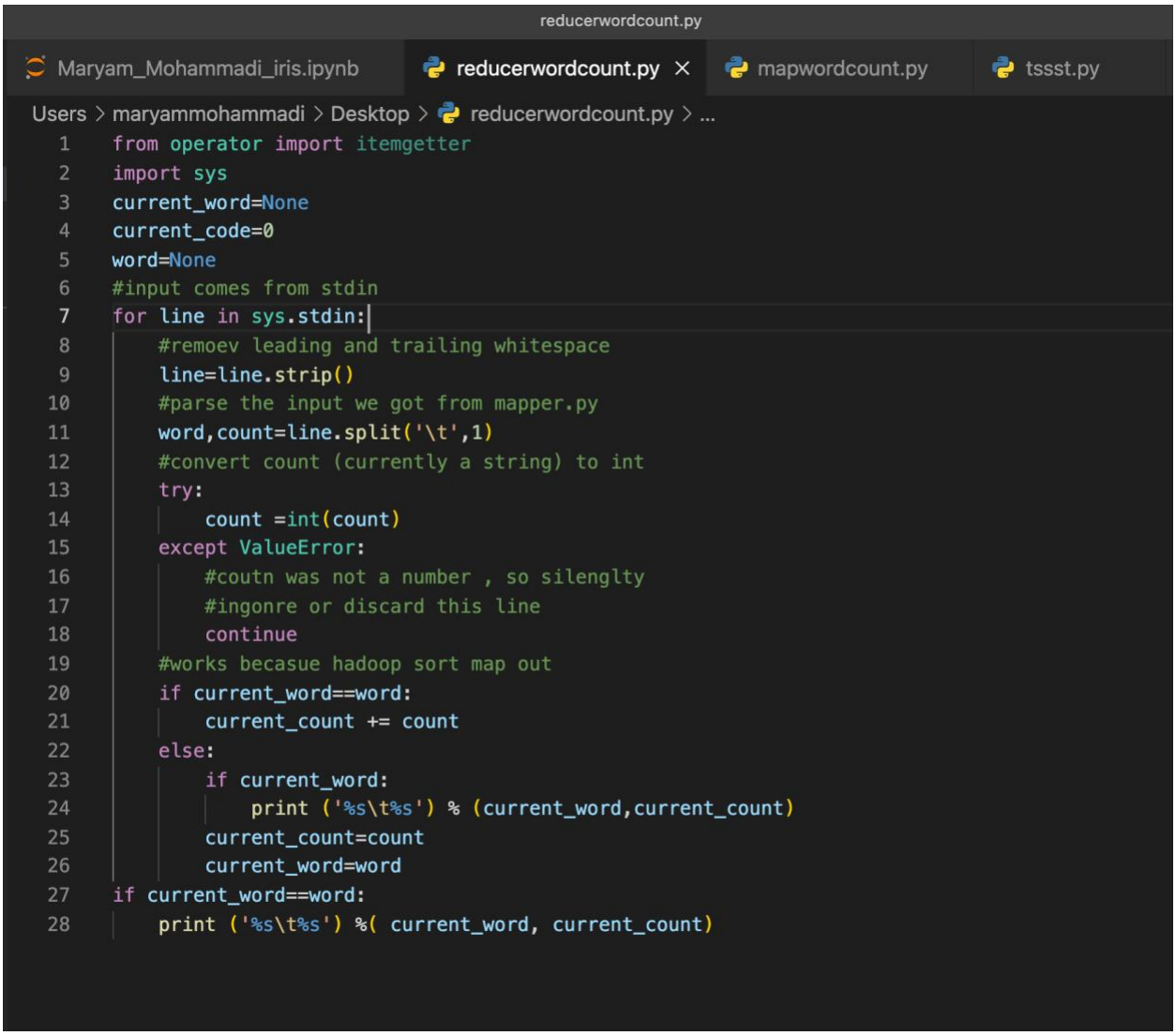
The first program map the file into word and key , the codes and its result is shown below:



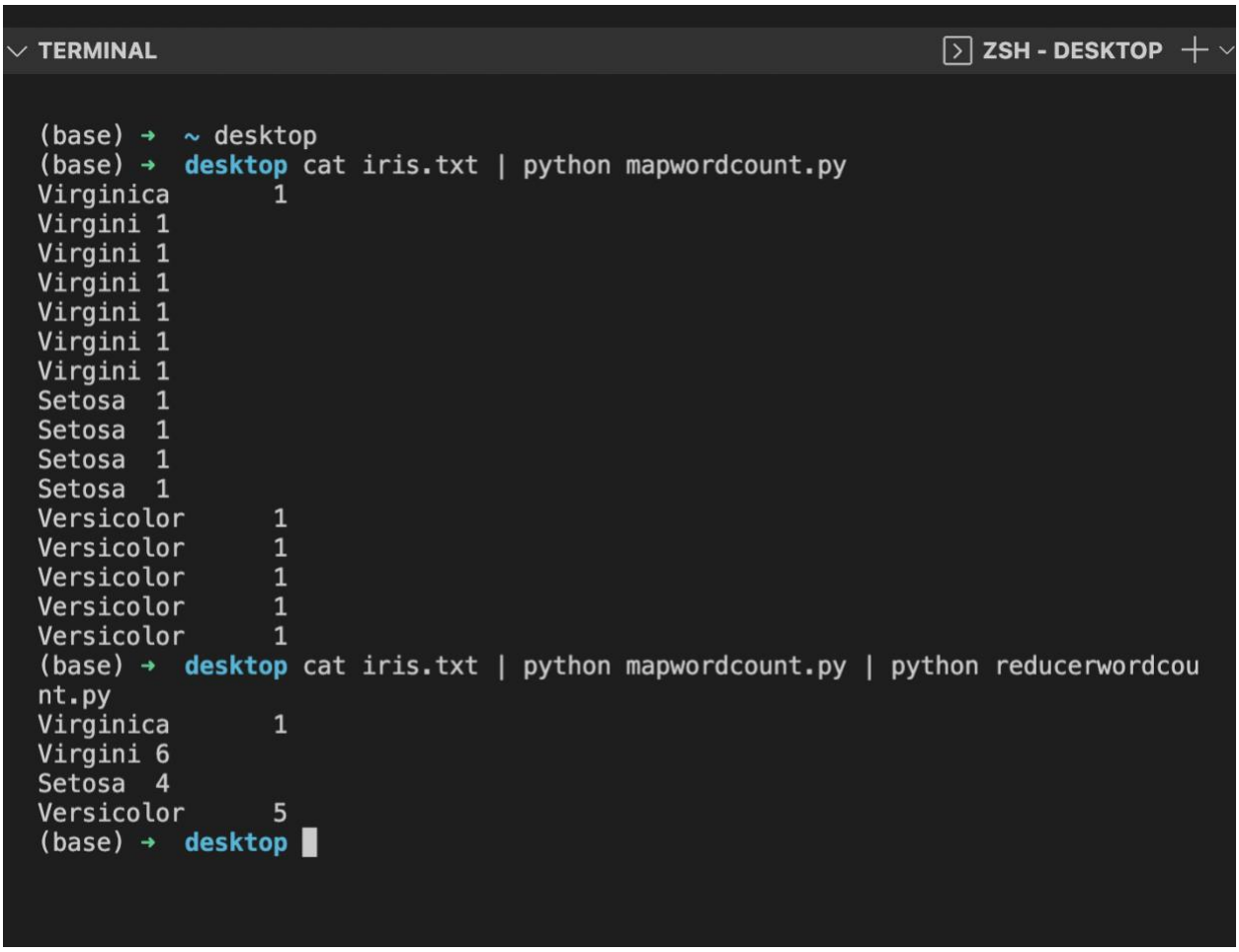
The result of the above code



The second code reduce the first file ; means count how many words there is in this file



And its result is as following:


TERMINAL
ZSH - DESKTOP
(base) ➜ ~ desktop
(base) ➜ desktop cat iris.txt | python mapwordcount.py
Virginica 1
Virgini 1
Virgini 1
Virgini 1
Virgini 1
Virgini 1
Virgini 1
Setosa 1
Setosa 1
Setosa 1
Setosa 1
Versicolor 1
Versicolor 1
Versicolor 1
Versicolor 1
Versicolor 1
(base) ➜ desktop cat iris.txt | python mapwordcount.py | python reducerwordcou
nt.py
Virginica 1
Virgini 6
Setosa 4
Versicolor 5
(base) ➜ desktop