

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>

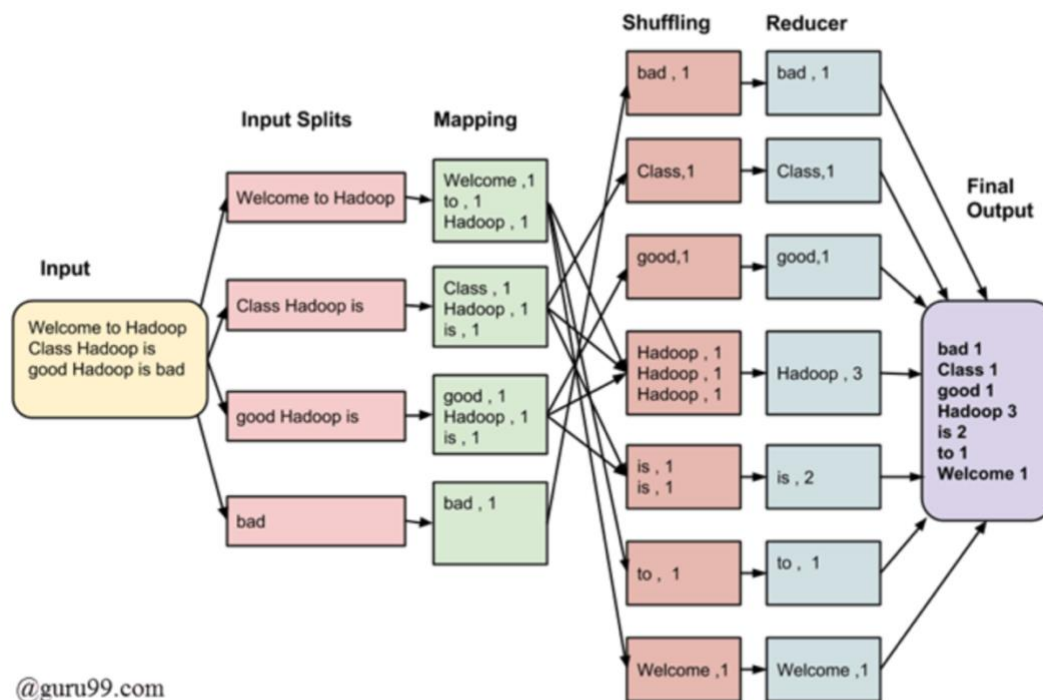
Shuffling

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 clubbed 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 repositories

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 clusters. 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 achieve 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 created 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 hadoop that has been used it will destroy all the data on the Hadoop file system

hadoop namenode -format
start-all.sh

for checking running process in our Hadoop cluster we use jps command . jps stands for java virtual 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
anager-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
oiv	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
lsSnapshottableDir	current directory contents with a snapshot
	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

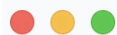
where COMMAND is one of:

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
oiv	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...

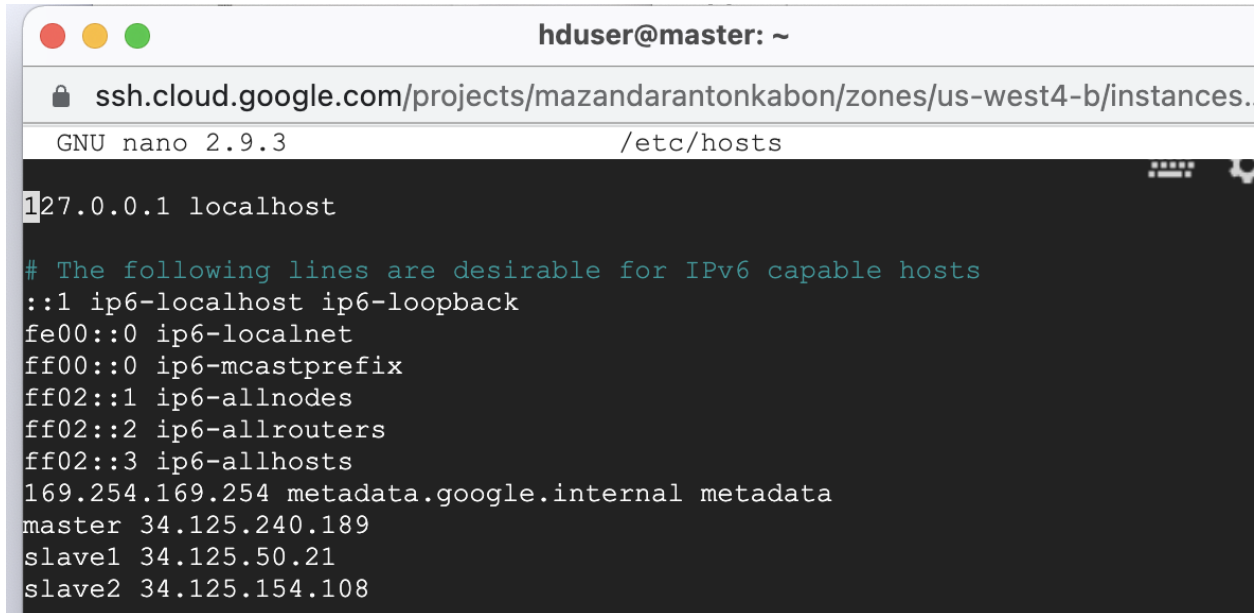
```
WARNING: All illegal access operations will be denied in a future release
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-r
esourcemanager-master.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-h
duser-nodemanager-master.out
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
```

A screenshot of a terminal window titled 'hduser@master: ~'. The terminal shows the 'nano' text editor editing the '/etc/hosts' file. The file content includes the standard localhost entry, IPv6 loopback and network addresses, and three entries for 'master', 'slave1', and 'slave2' with their respective IP addresses. The cursor is positioned at the end of the first line.

```
GNU nano 2.9.3 /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
169.254.169.254 metadata.google.internal metadata
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
```

```
hduser@master: ~  
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instance...  
hduser@master:~$ cat .ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACi7zVyTn1pBd5/vRmJt9mmVfFiOLovIDB70g343RvMEkiEdhQSPZiikVGfBKPI4Zb9UgfpdQQ5hMLdqmKye6csyXk25aBJKhkRQelmrjtbmGx0YbhpPMp3ejeB+gQhqRxTTXsp10Qas3DC+nNaUgBEQt5MjLgBiv5tjjdzUHx4BW5w6L8/03jazW5WsBIrWukP69PrE6shNzTZexqzC4cQqdV5p9gROVhNiUhXG4+v6Tf6osGBifqpicyiSKoP46hkJkvqZDIAxG8pfsOK9zjzTJPHM1HZz3fJtt9jvxbelyG/5p5/JhwKVa69g0COc++Y17eFHcSA0c0T5ht4necT hduser@master  
hduser@master:~$
```

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 pulic key of master machine

```
hduser@slave2: ~  
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/sl...  
GNU nano 2.9.3 .ssh/authorized keys  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACi7zVyTn1pBd5/vRmJt9mmVfFiOLovIDB70g343RvMEkiEdhQSPZiikVGfBKPI4Zb9UgfpdQQ5hMLdqmKye6csyXk25aBJKhkRQelmrjtbmGx0YbhpPMp3ejeB+gQhqRxTTXsp10Qas3DC+nNaUgBEQt5MjLgBiv5tjjdzUHx4BW5w6L8/03jazW5WsBIrWukP69PrE6shNzTZexqzC4cQqdV5p9gROVhNiUhXG4+v6Tf6osGBifqpicyiSKoP46hkJkvqZDIAxG8pfsOK9zjzTJPHM1HZz3fJtt9jvxbelyG/5p5/JhwKVa69g0COc++Y17eFHcSA0c0T5ht4necT hduser@master  
hduser@master: ~  
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/mast...  
Last login: Mon Feb 21 20:28:42 2022 from 127.0.0.1  
hduser@master:~$ cat .ssh/authorized_keys  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQACi7zVyTn1pBd5/vRmJt9mmVfFiOLovIDB70g343RvMEkiEdhQSPZiikVGfBKPI4Zb9UgfpdQQ5hMLdqmKye6csyXk25aBJKhkRQelmrjtbmGx0YbhpPMp3ejeB+gQhqRxTTXsp10Qas3DC+nNaUgBEQt5MjLgBiv5tjjdzUHx4BW5w6L8/03jazW5WsBIrWukP69PrE6shNzTZexqzC4cQqdV5p9gROVhNiUhXG4+v6Tf6osGBifqpicyiSKoP46hkJkvqZDIAxG8pfsOK9zjzTJPHM1HZz3fJtt9jvxbelyG/5p5/JhwKVa69g0COc++Y17eFHcSA0c0T5ht4necT hduser@master  
hduser@master:~$ cat .ssh/id_rsa.pub >> ssh/authorized_keys  
-bash: ssh/authorized_keys: No such file or directory  
hduser@master:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys  
hduser@master:~$
```

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`

```
hduser@master: ~  
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instance...  
hduser@master:~$ ssh slave1  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1064-gcp x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Mon Feb 21 22:07:28 UTC 2022  
  
System load:  0.13           Processes:            123  
Usage of /:   33.3% of 9.52GB Users logged in:        2  
Memory usage: 15%           IP address for ens4: 10.182.0.18  
Swap usage:   0%  
  
10 updates can be applied immediately.  
2 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
New release '20.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Feb 21 20:28:44 2022 from 127.0.0.1  
hduser@slave1:~$ ssh slave2  
The authenticity of host 'slave2 (10.182.0.19)' can't be established.  
ECDSA key fingerprint is SHA256:fZGjZ4KGEWoops7CrpNAwmskHkxmsLUJ7AE62/iA25k.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'slave2,10.182.0.19' (ECDSA) to the list of known  
hduser@slave2: Permission denied (publickey).
```

ssh slave2

```
hduser@master: ~  
ssh.cloud.google.com/projects/mazandarantonkabon/zones/us-west4-b/instances/ma...  
hduser@master:~$ ssh slave2  
The authenticity of host 'slave2 (10.182.0.19)' can't be established.    
ECDSA key fingerprint is SHA256:fZGjZ4KGEWoops7CrpNAwmskHkxmsLUJ7AE62/iA25k.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'slave2,10.182.0.19' (ECDSA) to the list of known ho  
sts.  
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1064-gcp x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Mon Feb 21 22:17:01 UTC 2022  
  
System load:  0.0                Processes:            128  
Usage of /:   33.3% of 9.52GB     Users logged in:     2  
Memory usage: 15%                IP address for ens4: 10.182.0.19  
Swap usage:   0%  
  
10 updates can be applied immediately.  
2 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
New release '20.04.3 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Last login: Mon Feb 21 20:28:47 2022 from 127.0.0.1  
hduser@slave2:~$ exit  
logout  
Connection to slave2 closed.  
hduser@master:~$
```

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:  
  libaiol 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> █
```

Google Cloud Platform Big data processing Search Products, resources, docs (/)

Connections

All instances > masterdb

✓ masterdb

MySQL 5.7

NETWORKING SECURITY **CONNECTIVITY TESTS**

This test analyzes your project to detect configuration issues. When applicable, it also sends packets through the data plane to verify connectivity. [Learn more](#)

CREATE RERUN DELETE

Filter Filter by test name or protocol

<input type="checkbox"/>	Name ↑	Protocol	Source	Destination	Destination port	Last test time	Last packet transmission result	Last configuration analysis result	Result details
<input type="checkbox"/>	master	tcp	masterdb	34.125.67.133	80	2022-02-19 (23:29:04)	—	✓ Reachable	VIEW

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.

<input type="checkbox"/>	client	34.125.31.27	us-west4	Static	IPv4	VM instance client (Zone us-west4-b)	Premium	CHANGE
--------------------------	--------	--------------	----------	--------	------	--------------------------------------	---------	------------------------

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.

← Create a MySQL instance

Instance info

Instance ID *

clientsql

Use lowercase letters, numbers, and hyphens. Start with a letter.

Password *

unix



GENERATE

Set a password for the root user. [Learn more](#)

☐ No password

Database version *

MySQL 8.0

Choose region and zonal availability

For better performance, keep your data close to the services that need it. Region is permanent, while zone can be changed any time.

Region

us-central1 (Iowa)

Zonal availability

☐ Single zone

In case of outage, no failover. Not recommended for production.

☒ Multiple zones (Highly available)

Automatic failover to another zone within your selected region. Recommended for production instances. [Increases cost.](#)

Uploads and Big data processing operations



Created [clientsql](#)

10:43:12 AM GMT+1

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.

Authorized networks

You can specify CIDR ranges to allow IP addresses in those ranges to access your instance. [Learn more](#)

clientnetwork (34.125.31.27)	(Not saved) ▼
ADD NETWORK	

App Engine authorization

All apps in this project are authorized by default. You can use [Cloud IAM](#) to authorize apps in other projects. [Learn more](#)

SAVE

DISCARD CHANGES



Connect to this instance

Public IP address

104.197.225.3



Connection name

mazandarantonkaron:us-central1:clientsql



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

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	client	us-west4-b			10.182.0.31 (nic0)	34.125.31.27	SSH ▾
<input type="checkbox"/>	✓	master	us-west4-b			10.182.0.28 (nic0)	34.125.231.254	SSH
<input type="checkbox"/>	✓	slave1	us-west4-b			10.182.0.29 (nic0)	34.125.51.22	SSH
<input type="checkbox"/>	✓	slave2	us-west4-b			10.182.0.30 (nic0)	34.125.92.172	SSH

⋮

Open in browser window

Open in browser window on custom port

Open in browser window using provided private SSH key

View gcloud command

Use another SSH client

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

gcloud command line

The following gcloud command line can be used to SSH into this instance. [gcloud reference](#)

```
$ gcloud compute ssh --zone "us-west4-b" "client" --project "mazandarantonkabon"
```

☒ Line wrapping

 COPY TO CLIPBOARD [RUN IN CLOUD SHELL](#) [CLOSE](#)

- 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.
- 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`

The screenshot shows the Google Cloud Storage interface for a bucket named 'irisdataset1'. The bucket is located in 'us (multiple regions in the United States)', has a 'Standard' storage class, 'Not public' access, and 'None' protection. The 'OBJECTS' tab is selected, showing a list of objects. One object, 'iris.data', is listed with a size of 4.4 kB, created on 21 Feb 2022 at 20:18:40, and stored in the 'Standard' class. The interface includes navigation links like 'UPLOAD FILES', 'CREATE FOLDER', and 'MANAGE BLOCKS', as well as a filter bar and a table of object details.

First name	Dimensions	Guy	Creation date / time	Storage class	Last edit	Public access	Version history	Encryption
<input type="checkbox"/> iris.data	4.4 kB	application / octet-stream	21 Feb 2022, 20:18:40	Standard	21 Feb...	Non publica	—	Google-managed k

Part 4: Mapreducing with python programming

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

```
1 import sys
2 #input comes from standard input
3 for line in sys.stdin:
4     #remove line leading and trailing whitespace
5     line=line.strip()
6     #split the line into words
7     words=line.split()
8     #increase counters
9     for word in words:
10         #write the result to stdout or standard output
11         #what we output here will be the input for the reduce step (reduce.py)
12         #tab-delimited; the trivial word count is 1
13         print('%s\t%s' % (word, 1))
14
```

The result of the above code

```
✓ TERMINAL
(base) -> ~ desktop
(base) -> desktop cat iris.txt | python mapwordcount.py
Virginica      1
Virginica      1
Virginica      1
Virginica      1
Virginica      1
Virginica      1
Virginica      1
Virginica      1
Setosa         1
Setosa         1
Setosa         1
Setosa         1
Versicolor     1
Versicolor     1
Versicolor     1
Versicolor     1
Versicolor     1
(base) -> desktop
```

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

```
reducerwordcount.py
Maryam_Mohammadi_iris.ipynb reducerwordcount.py x mapwordcount.py tssst.py
Users > maryammohammadi > Desktop > reducerwordcount.py > ...
1  from operator import itemgetter
2  import sys
3  current_word=None
4  current_count=0
5  word=None
6  #input comes from stdin
7  for line in sys.stdin:
8      #remoev leading and trailing whitespace
9      line=line.strip()
10     #parse the input we got from mapper.py
11     word,count=line.split('\t',1)
12     #convert count (currently a string) to int
13     try:
14         count =int(count)
15     except ValueError:
16         #coutn was not a number , so silengltly
17         #ingonre or discard this line
18         continue
19     #works becasue hadoop sort map out
20     if current_word==word:
21         current_count += count
22     else:
23         if current_word:
24             print ('%s\t%s') % (current_word,current_count)
25             current_count=count
26             current_word=word
27 if current_word==word:
28     print ('%s\t%s') %( current_word, current_count)
```

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
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 reducerwordcount.py
Virginica      1
Virgini 6
Setosa 4
Versicolor    5
(base) → desktop
```