Documentation for the thesis:

"Comparative Performance Evaluation of Hadoop on PaaS Proposals by Leveraging HiBench"

Uluer Emre Özdil

May 1, 2021

Contents

1 Creating Dataproc Cluster on GCP

To operate on GCP, a Google account is required. Having a Google account set, go to:
https://cloud.google.com

Figure 1: Click on Sign In, giving requested account credentials will redirect to GCP main page as below. Click on the Console link on the top right of the page.

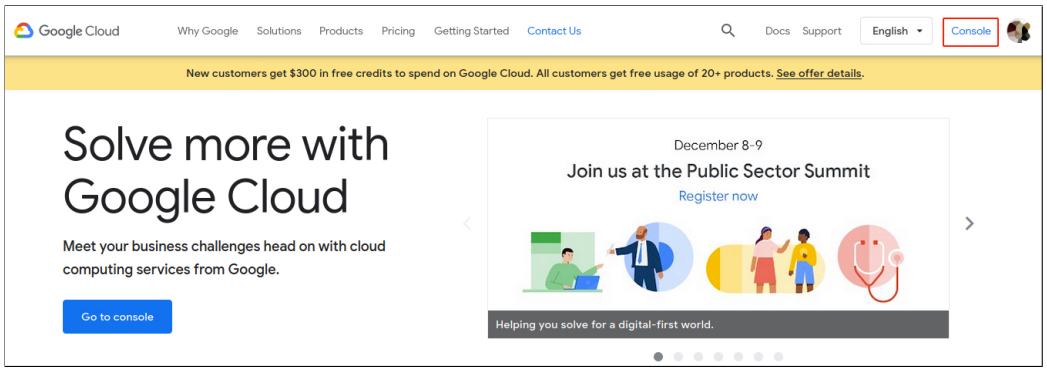

Figure 1: GCP Mainpage

Figure 2: In the dashboard the user is requested to create a project first. A project is needed to go with GCP services. Click on Create Project.

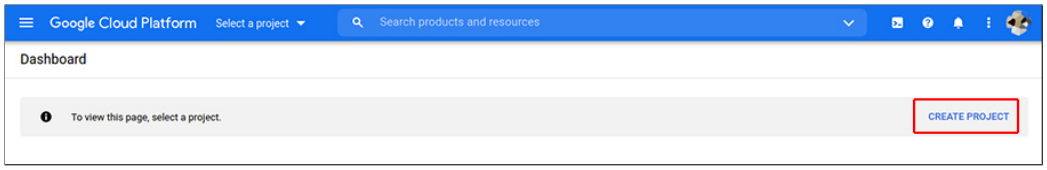

Figure 2: GCP - New Project

Figure 3: Enter project details. At this stage a billing account will be asked to be created as well.

Figure 4: Clicking on "Dataproc" in the GCP menu below the group "Big Data" will redirect to Dataproc since the API for Dataproc needs to be enabled to run. Clicking on "Enable" will set Dataproc ready to operate.

Figure 5: Before starting Dataproc cluster installation, we create a firewall rule to allow accessing Hadoop web UI over internet. From the VPC network dashboard within GCP, click on Create Firewall Rule.

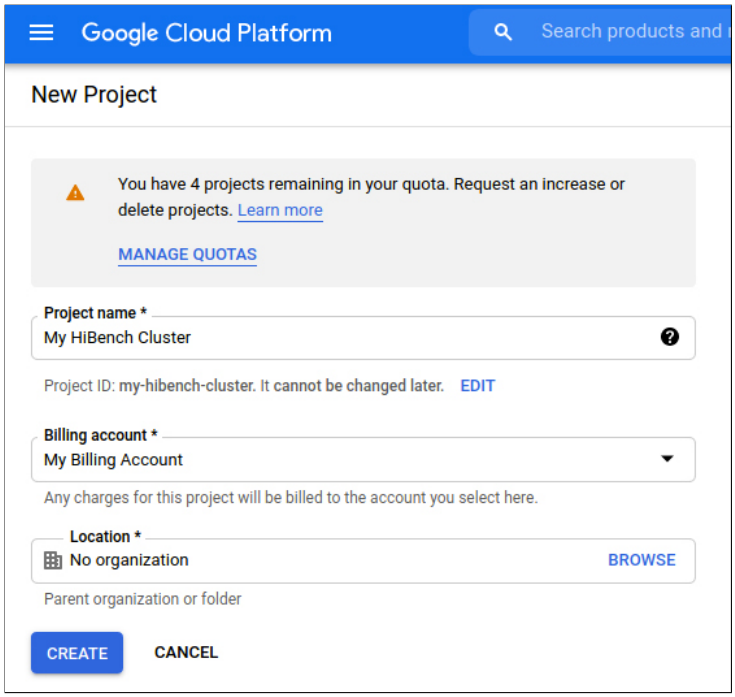

Figure 3: GCP - Create Project

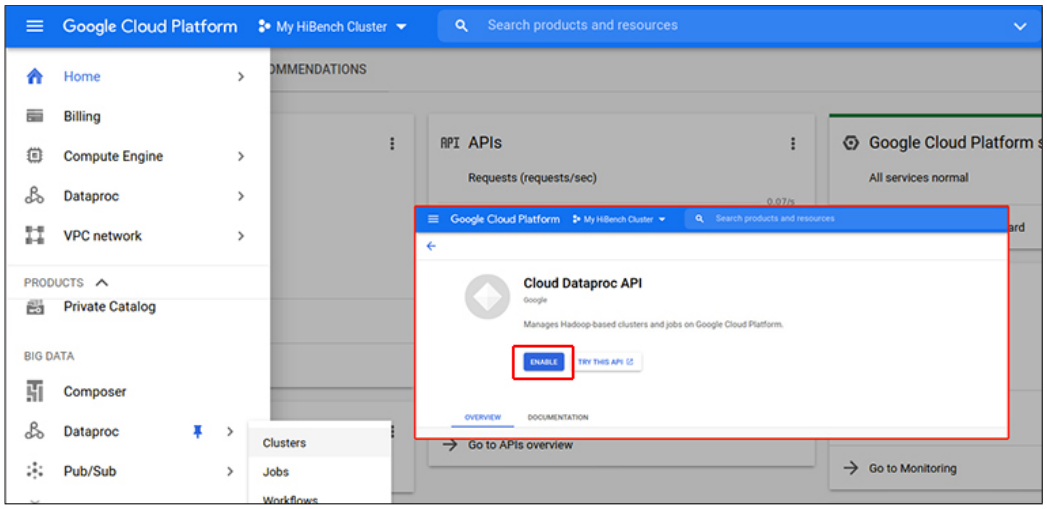

Figure 4: GCP - Enabling Dataproc API

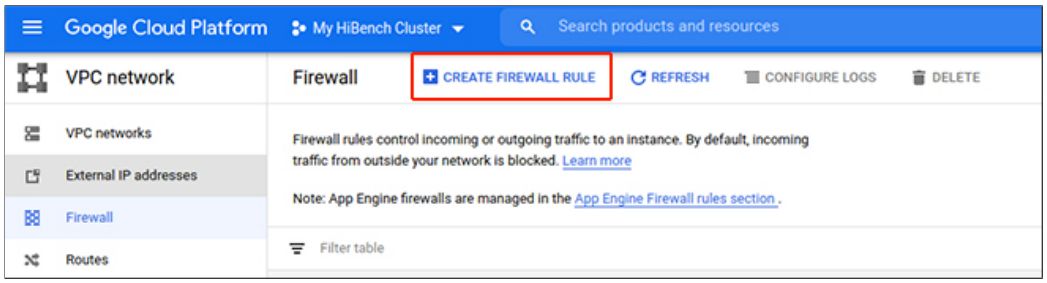

Figure 5: GCP- Create Firewall (Step 1)

Figure 6: Entering following informations:
Name: hadoop-access

Figure 7: Direction of traffic: ingress (for incoming traffic)
Specify a user friendly tag name (hadoopaccess in this case) for determining the access of the dataproc cluster at later stage. To limit the cluster acces only with our local machine, Source IP ranges is given with the IP value of current local machine including subnet mask "/32" at the end. We specify TCP protocol with port number 9870 and 8088, gateways for NameNode's and YARN resource manager's WebUI, respectively.

Figure 8: Clicking on Create button redirects to firewall listings page after successful creation.

Figure 9: To start with the managed Hadoop cluster, click on "Create Cluster"

Figure 10:
Selected configurations:
Setup Cluster Pane
Name: a proper name for the cluster
Location: europe-west3, Frankfurt
Cluster type: Select "Standard (1 master, N workers)"
Autoscaling policy is left blank since a we are not dealing autoscaling.

Figure 11: The pre-installed and pre-configured Dataproc image 1.4 fits our rules since it fits our rule satisfying HiBench's prerequisite of Hadoop 2.X. The OS is ubuntu 18.04.

Figure 12: Regarding our aim to benchmarks managed systems as they come out of the box, we don't want to use any optional components, so leaving the Components boxes blank.

Figure 13: For the master node we select General Purpose *e2-highmem-8* configuration allocating 64GB memory with 8 cores for the namenode. Master node is lasted mostly in memory operations so we allocate large space. Leave the primary

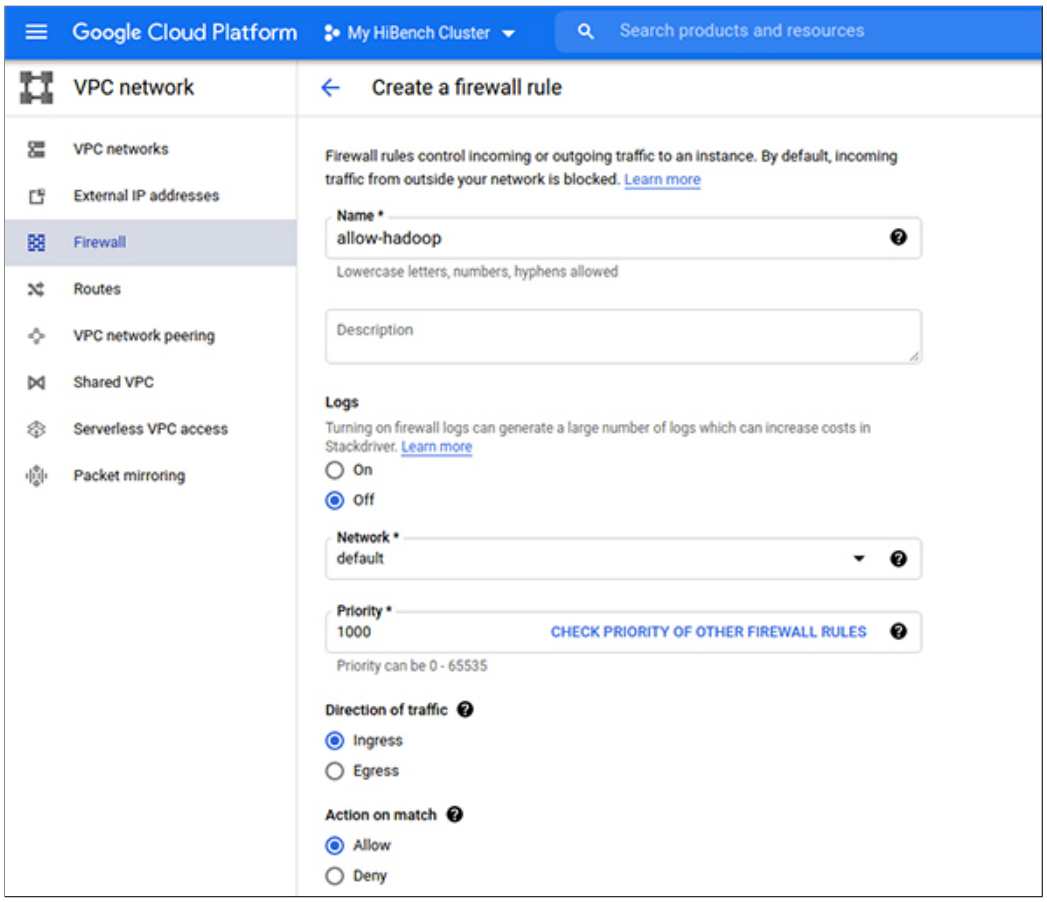

Figure 6: GCP- Create Firewall (Step 2)

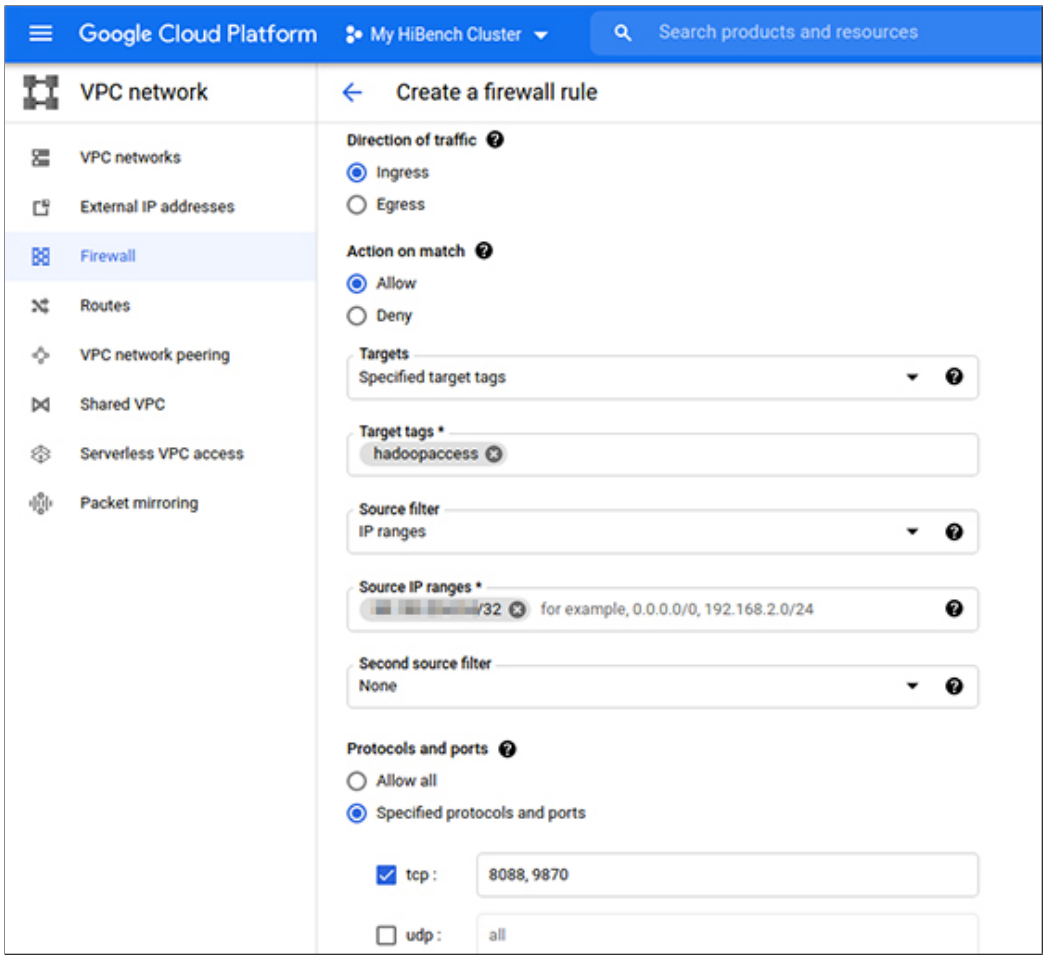

Figure 7: GCP- Create Firewall (Step 3)

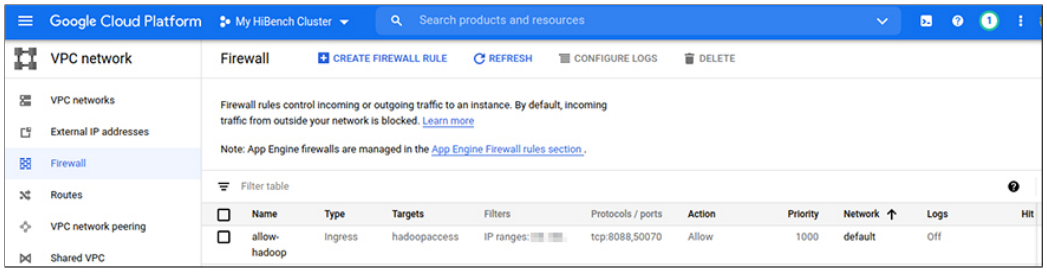

Figure 8: GCP- Create Firewall (Step 4)

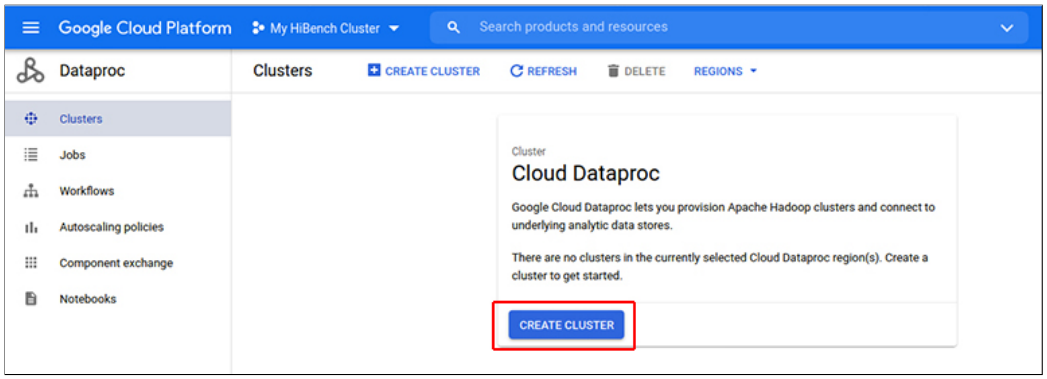

Figure 9: GCP Dataproc - Create Cluster (Step 1)

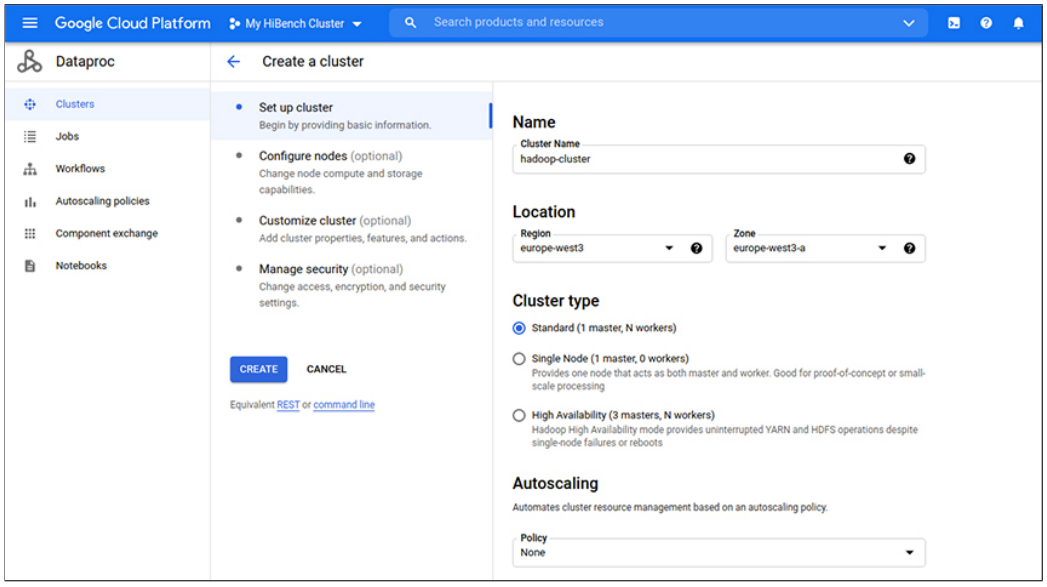

Figure 10: GCP Dataproc - Create Cluster (Step 2)

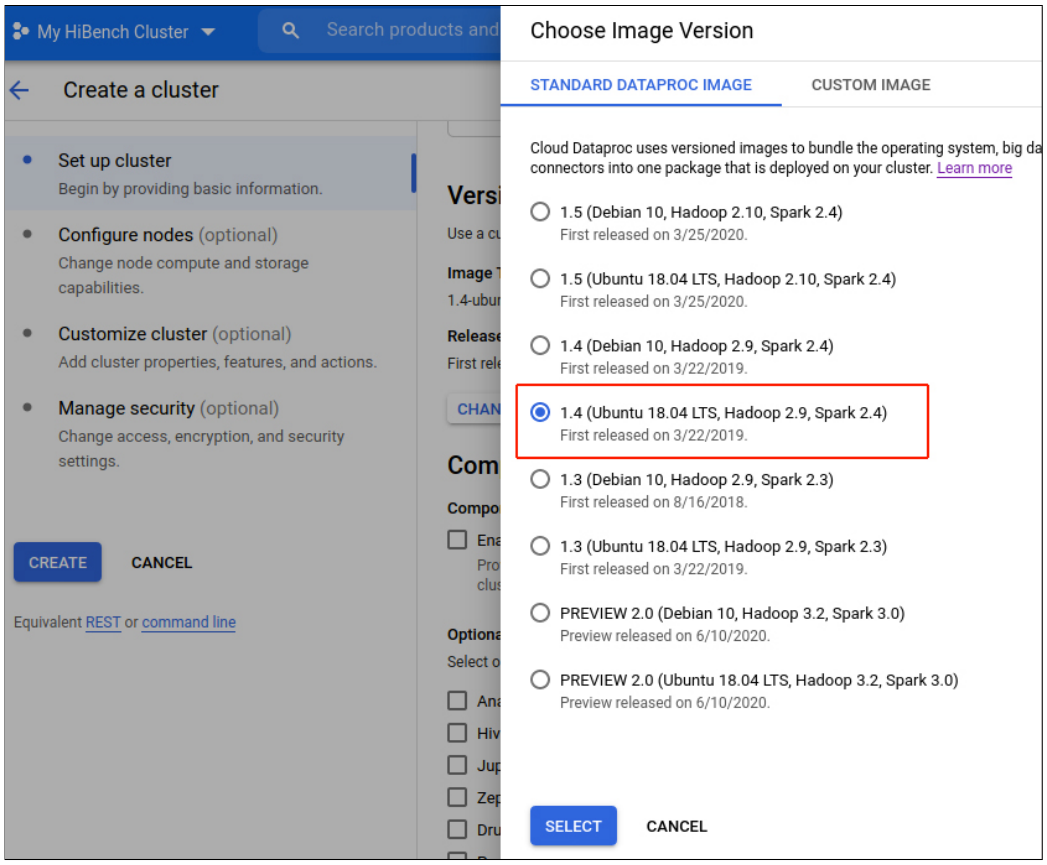

Figure 11: GCP Dataproc - Create Cluster (Step 3)

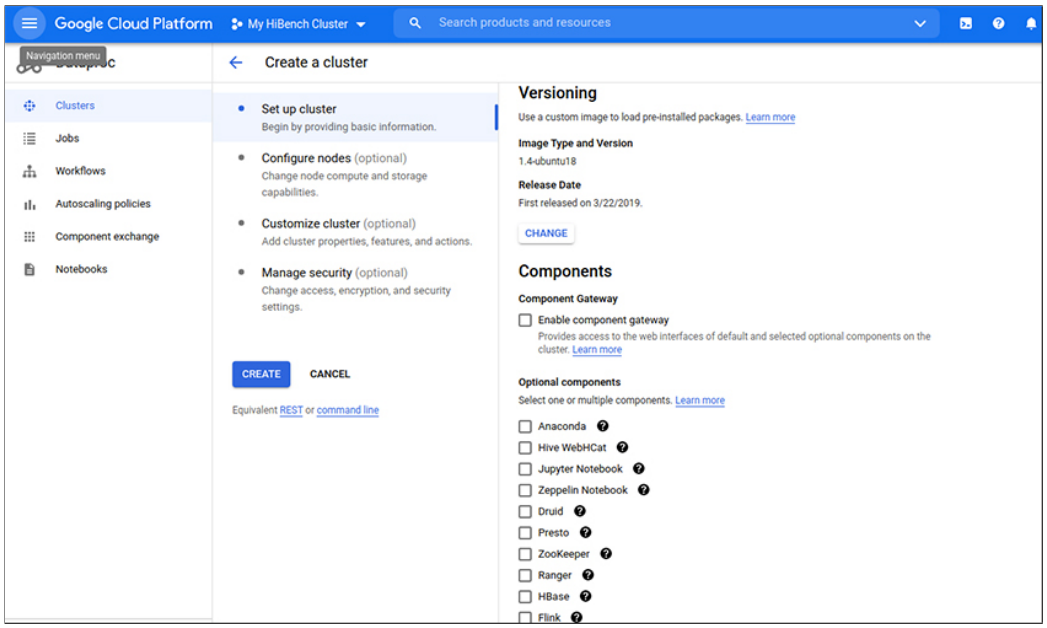

Figure 12: GCP Dataproc - Create Cluster (Step 4)

disk size in its default value of 500 GB.

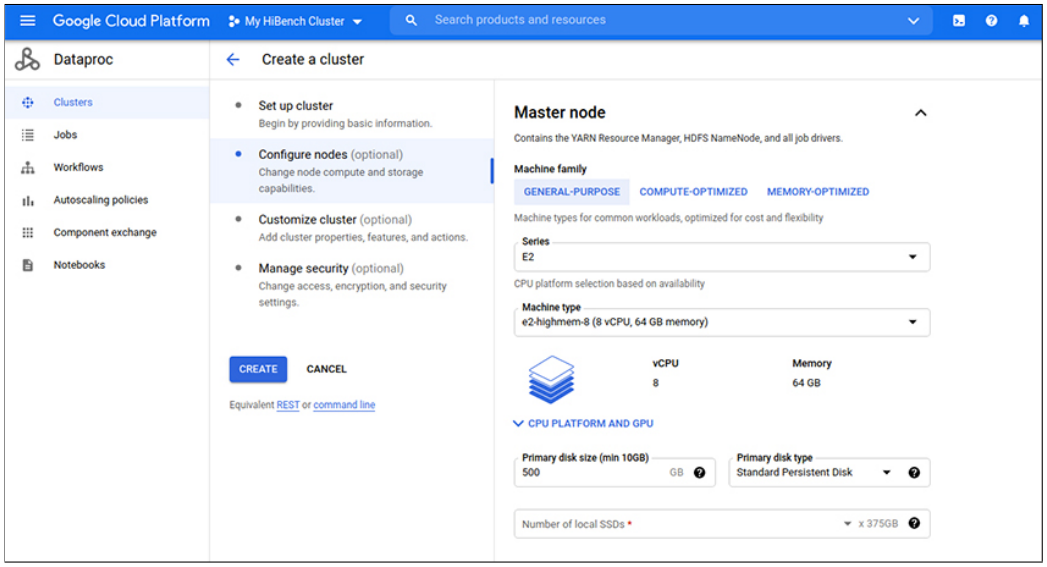

Figure 13: GCP Dataproc - Create Cluster (Step 5)

Figure 14: For worker nodes number we specify 3, for machine type selecting *e2-highmem-4* configuration provides 4 CPUs and 32 GB memory per worker node which totals in 12 processors and 96 GB memory for the cluster.

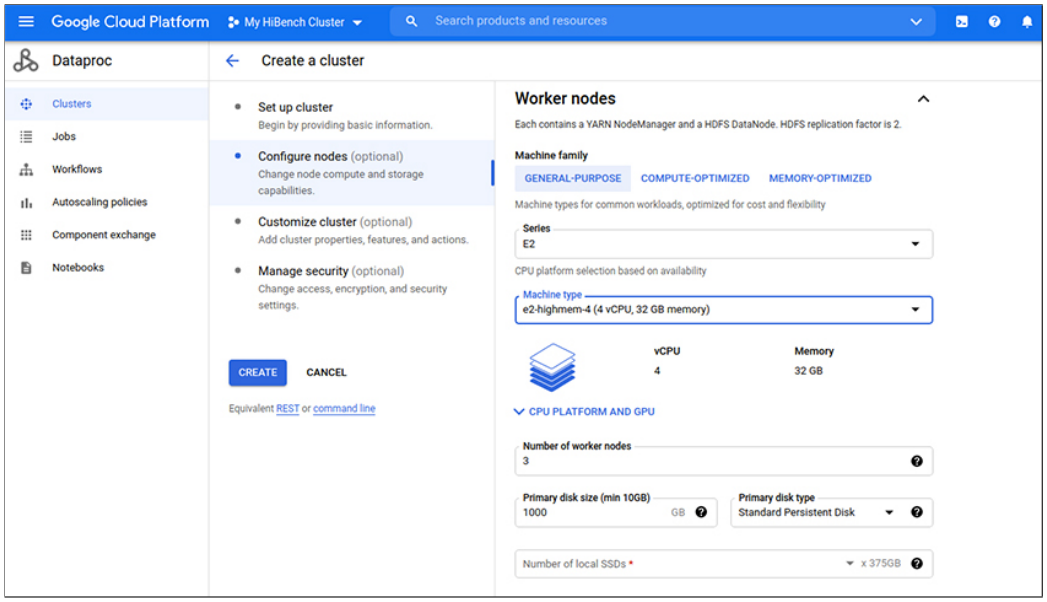

Figure 14: GCP Dataproc - Create Cluster (Step 6)

Figure 15: Below the configuration settings page worker nodes' available systems resources to YARN is listed. The local fraction available to YARN is 0.8 which makes 76.8 of 96 GB available to Hadoop.

Figure 16: For the network configuration we provide the tag we created during firewal creation for Hadoop.

Figure 17: As final step we check Allow API access, and after reviewing our settings, we click on Create.

Figure 18 and Figure 19: The creation process can be followed from Dataproc and VM dashboards.

Figure 20: Once the installation is completed, by navigating to Compute Engine ¿ VM Instances page we can acces master and worker nodes' CLIs by clicking on relevant SSH buttons. Our Dataproc cluster is ready for benchmarking operations.

Figure 21 and Figure 22: Using the external IP of the master node and specified ports 8088 and 9870 in the firewall settings, Hadoop Web UI portals shall be available.

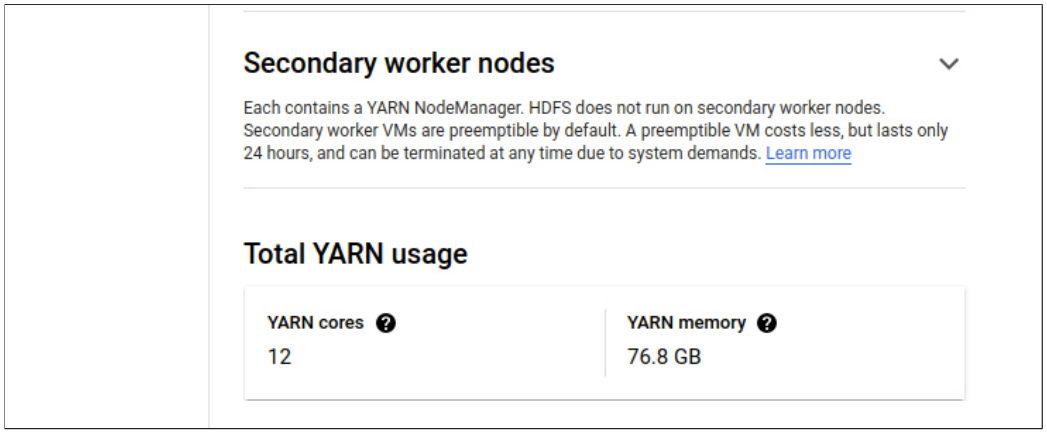

Figure 15: GCP Dataproc - Create Cluster (Step 7)

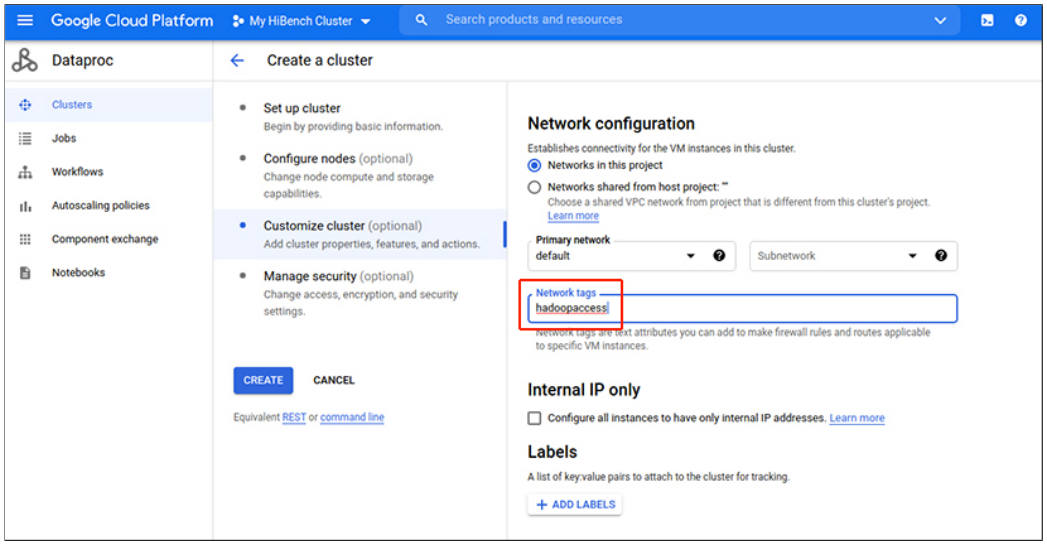

Figure 16: GCP Dataproc - Create Cluster (Step 8)

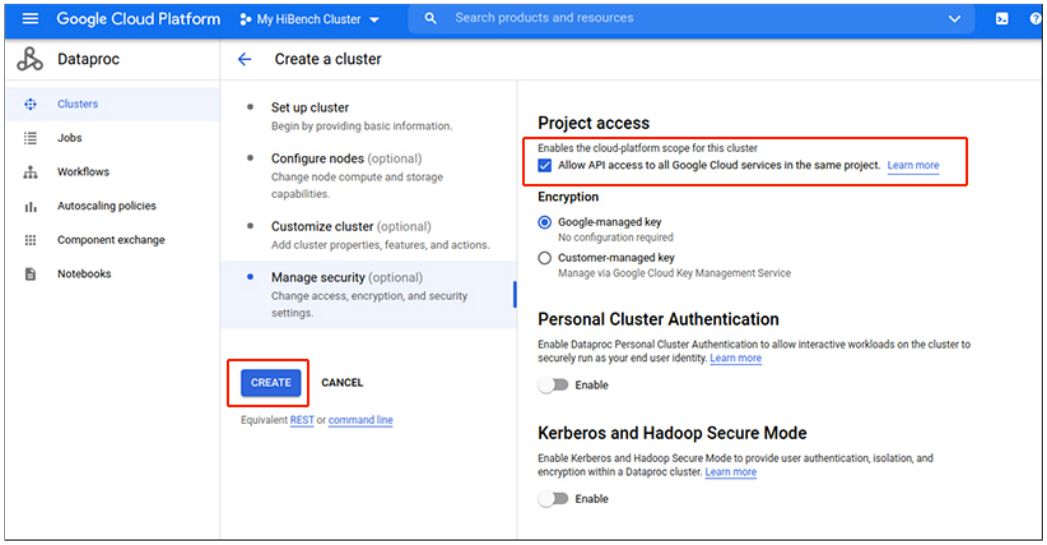

Figure 17: GCP Dataproc - Create Cluster (Step 9)

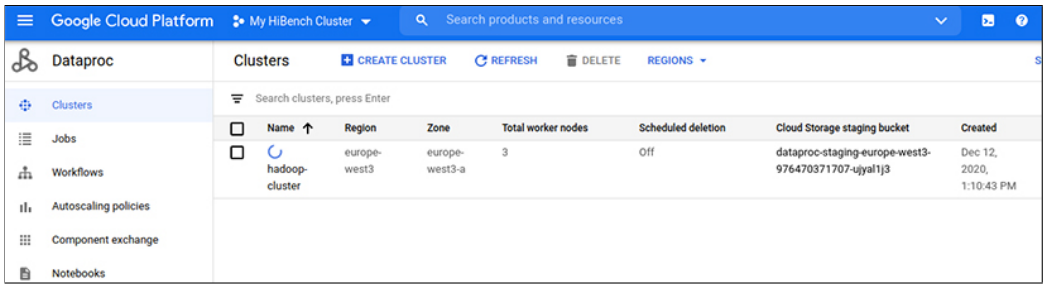

Figure 18: GCP Dataproc - Create Cluster (Step 10)

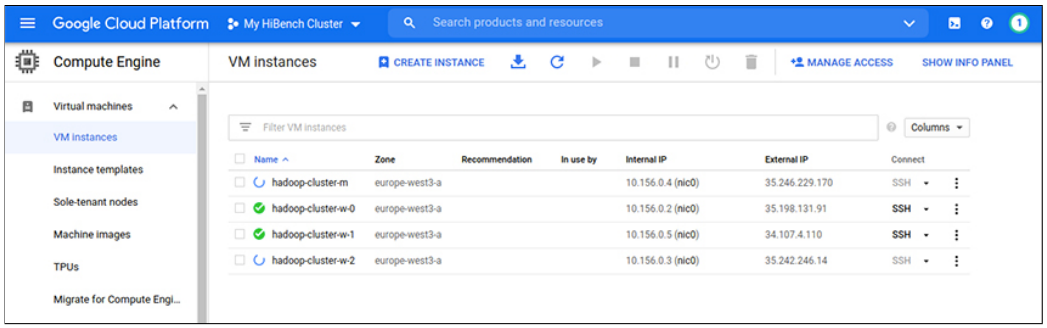

Figure 19: GCP Dataproc - Create Cluster (Step 11)

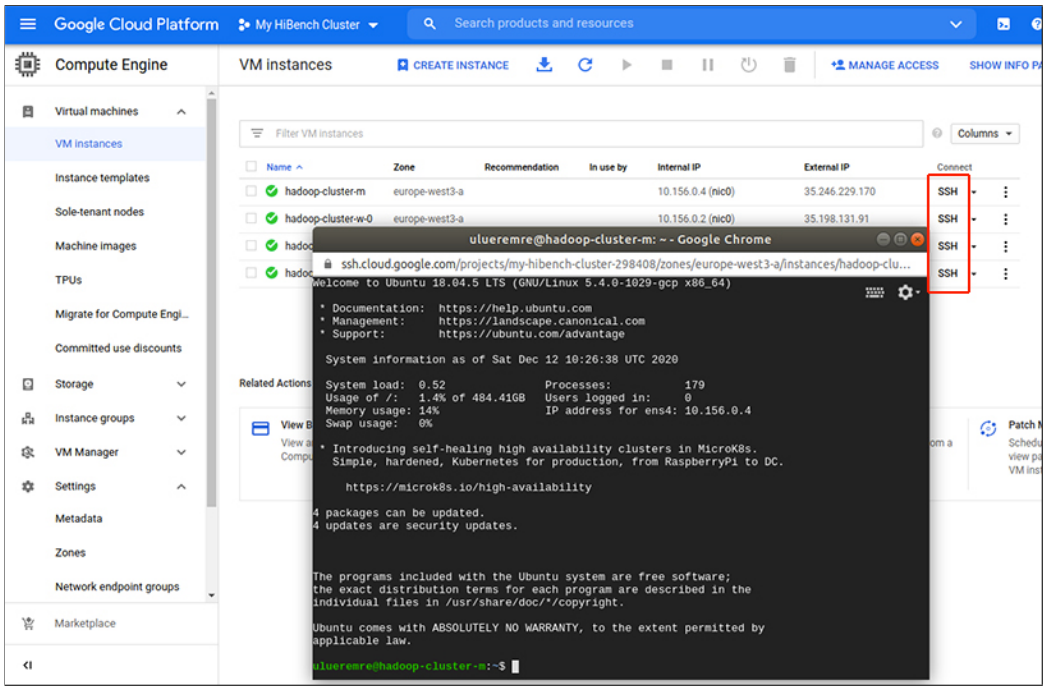

Figure 20: GCP Dataproc - Create Cluster (Step 12)

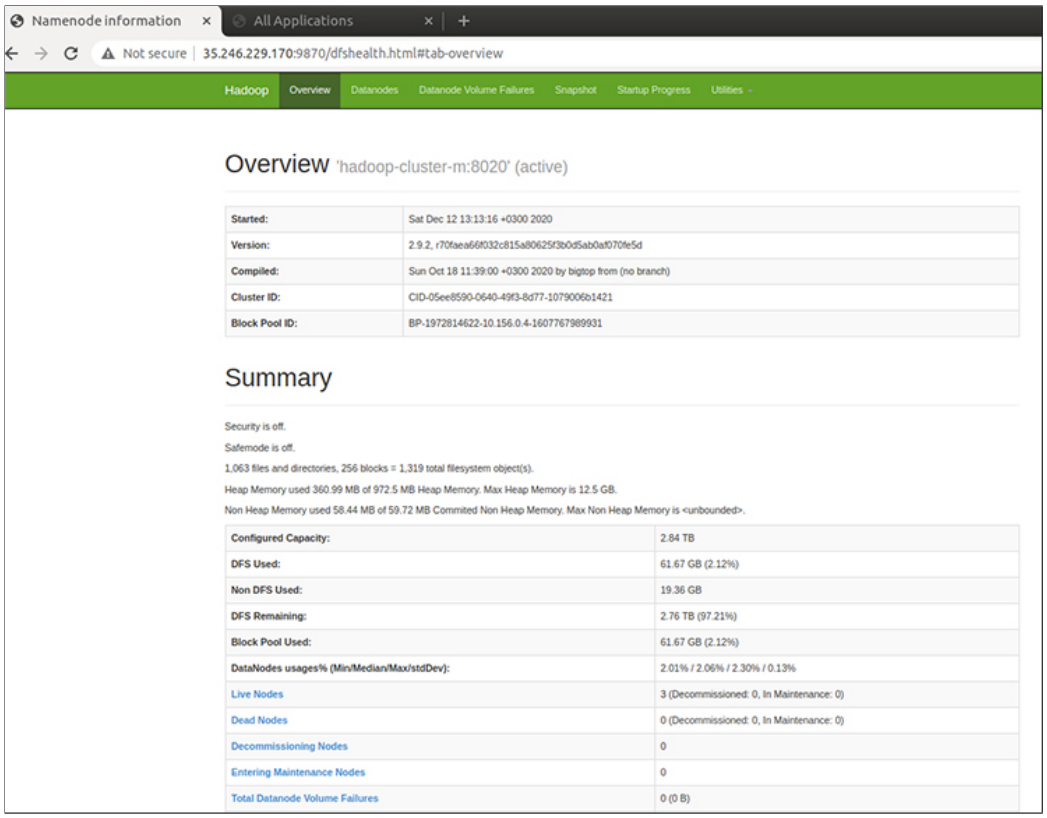

Figure 21: GCP Dataproc - Namenode Manager

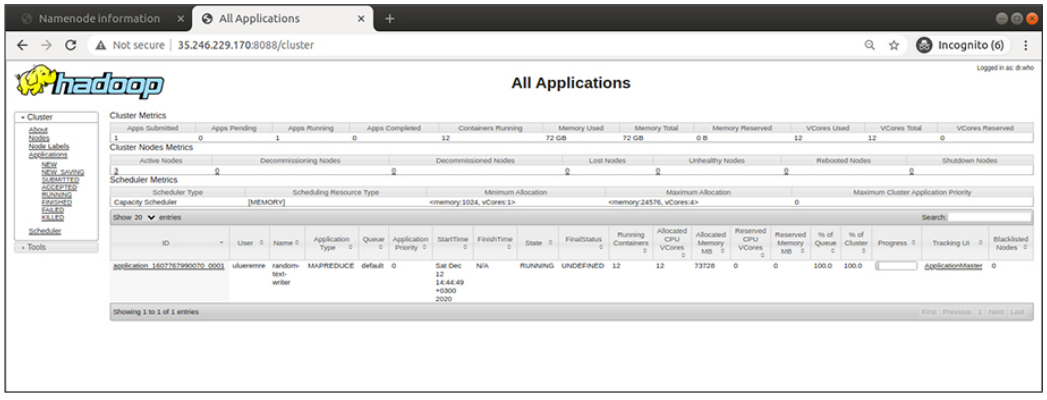

Figure 22: GCP Dataproc - Resource Manager

2 Creating HDInsight Cluster on Azure

Working on Azure requires a Microsoft account; an account can be created at https://login.live.com.

Figure 23: On the Azure portal, we first create a resource group. A resource group is a collection of resources that share the same lifecycle, permissions, and policies.

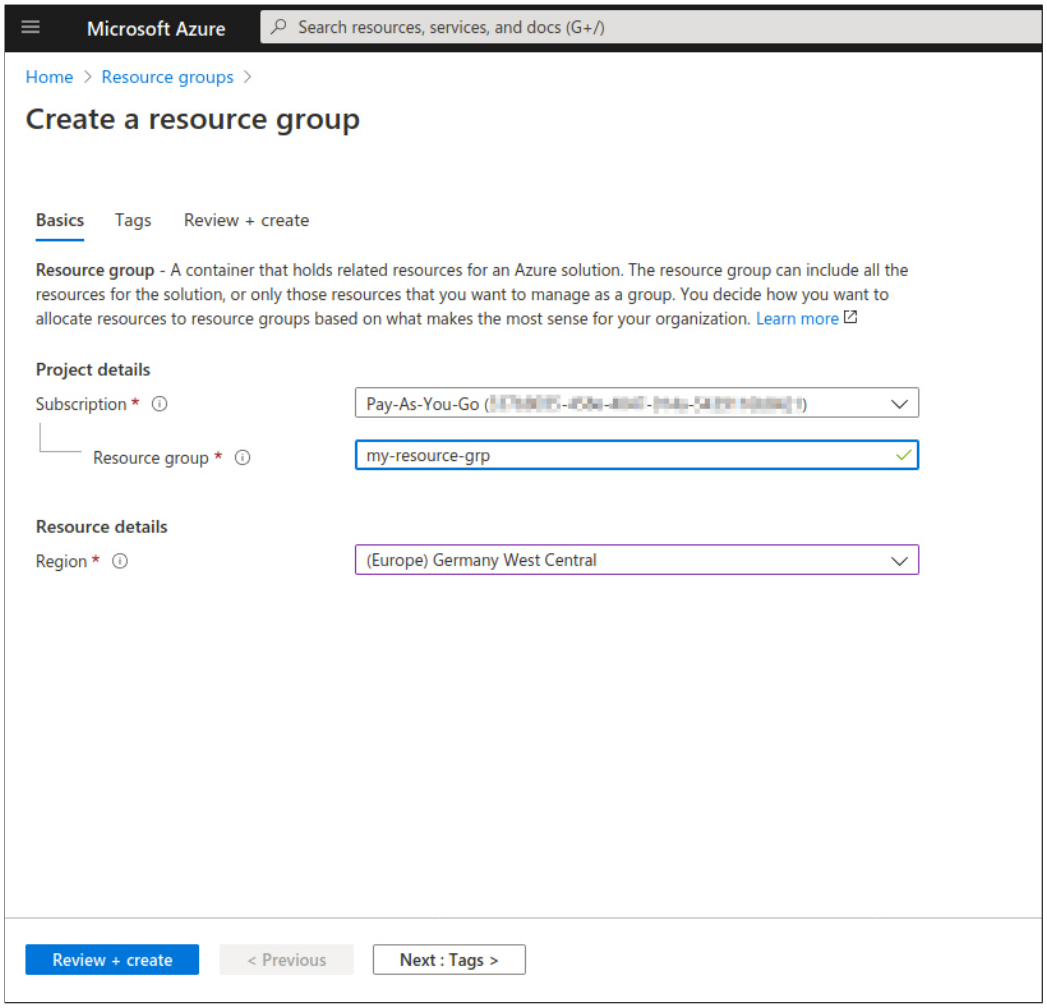

Figure 23: HDInsight setup - Create a resource group

Figure 24: The subscription has to be registered with Microsoft.HDInsight. From the subscription's Resource Provider page we can locate HDInsight and

register it.

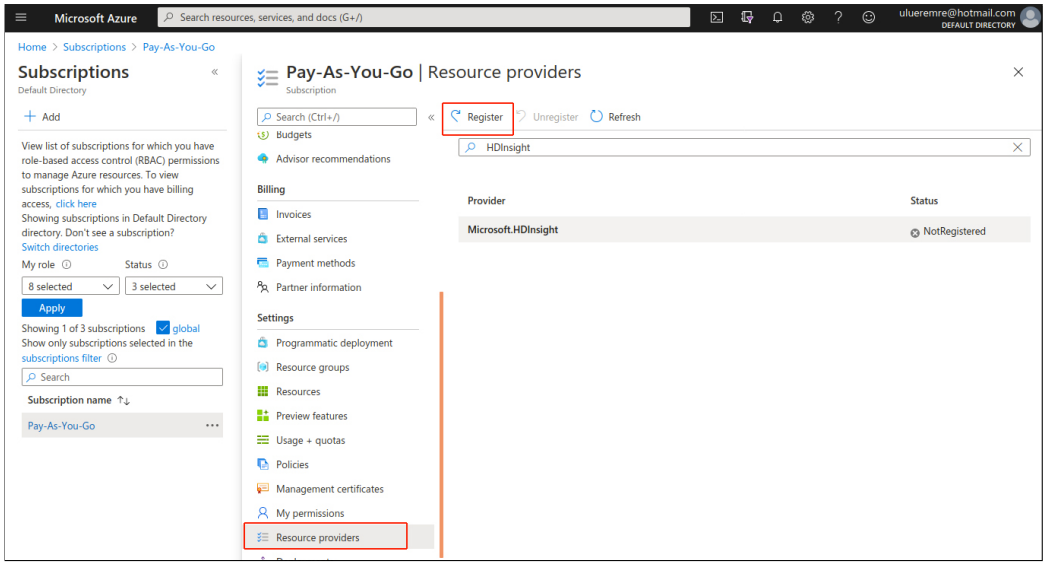

Figure 24: HDInsight setup - Register subscription

Figure 25: Next step is to create an HDI cluster. To do so, we first navigate to HDI console by entering the name HDInsight into the searchbar on the top of the page.

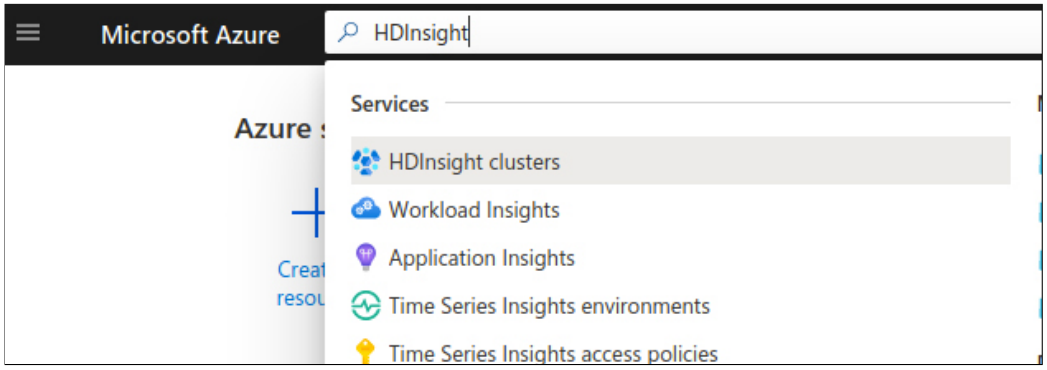

Figure 25: HDInsight setup - Navigate to HDInsight

Figure 26: From the HDI console, click on "Create HDInsight Cluster"

Figure 27: Select subscription and resource group. Give cluster a name and select *Germany West Central* as Region. The cluster type is Hadoop, version

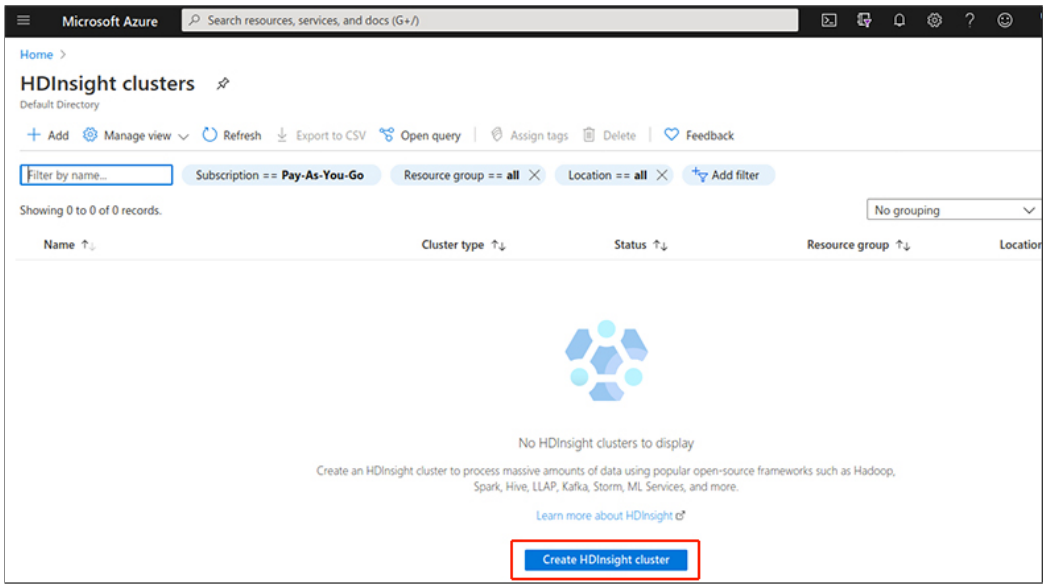

Figure 26: HDInsight setup - Create Cluster

will be *HDI 3.6*. For remote CLI operations, remote logon with an SSH Key is recommended, for the sake of the purpose, we check *Use cluster login as password for SSH*.

Figure 28: Leave the storage settings with their defaults.

Figure 29: Step 3: As with storage options, we leave Network settings with their defaults.

Figure 30: HDInsight assigns an obligatory High Availability option for the master node which doubles the resource usage. By the date the study has been made Zoekeeper configuration has also been assigned to 3 free of charge nodes by HDInsight. We specify the master and worker nodes' machine types by the following choices: *A8m v2* for master node, and *A4m v2* for 3 worker nodes. Note: Due to resource usage regulations it is likely that the user faces issues on core availability on this page. Opening a support ticket for rising the quota limit may solve this issue within 24 to 48 hours.

Figure 31: Review the validated settings and click on Create button. Any issues related with the settings are displayed here so that they can be fixed before creation process.

Figure 32 The creation process takes some time and can be tracked from the HDInsight cluster overview page.

Figure 33 After successful installation a message appears.

Figure 34 HDInsight comes with an HDP (Hortonworks Data Platform) cluster.

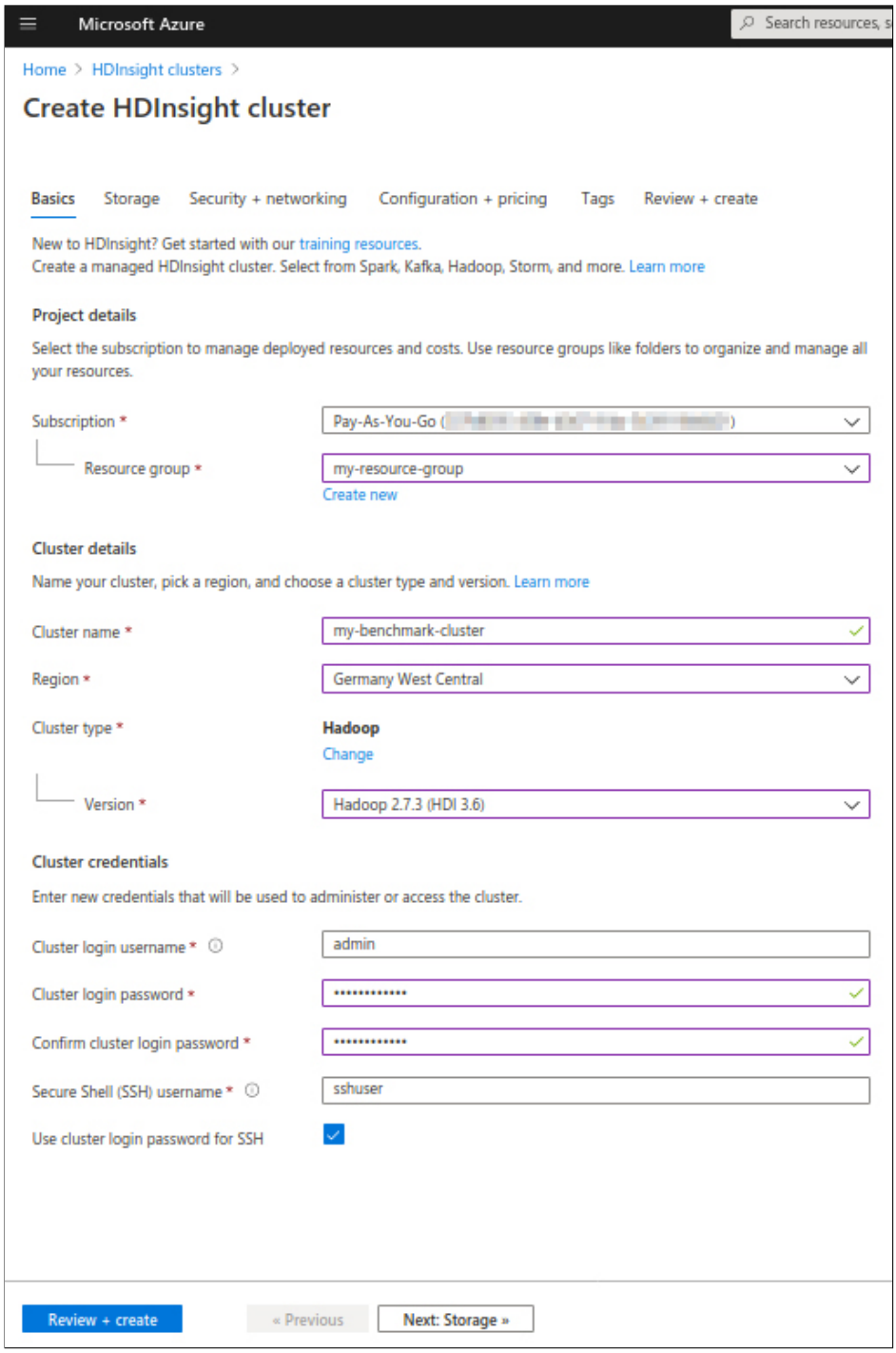

Figure 27: HDInsight setup - Create Cluster

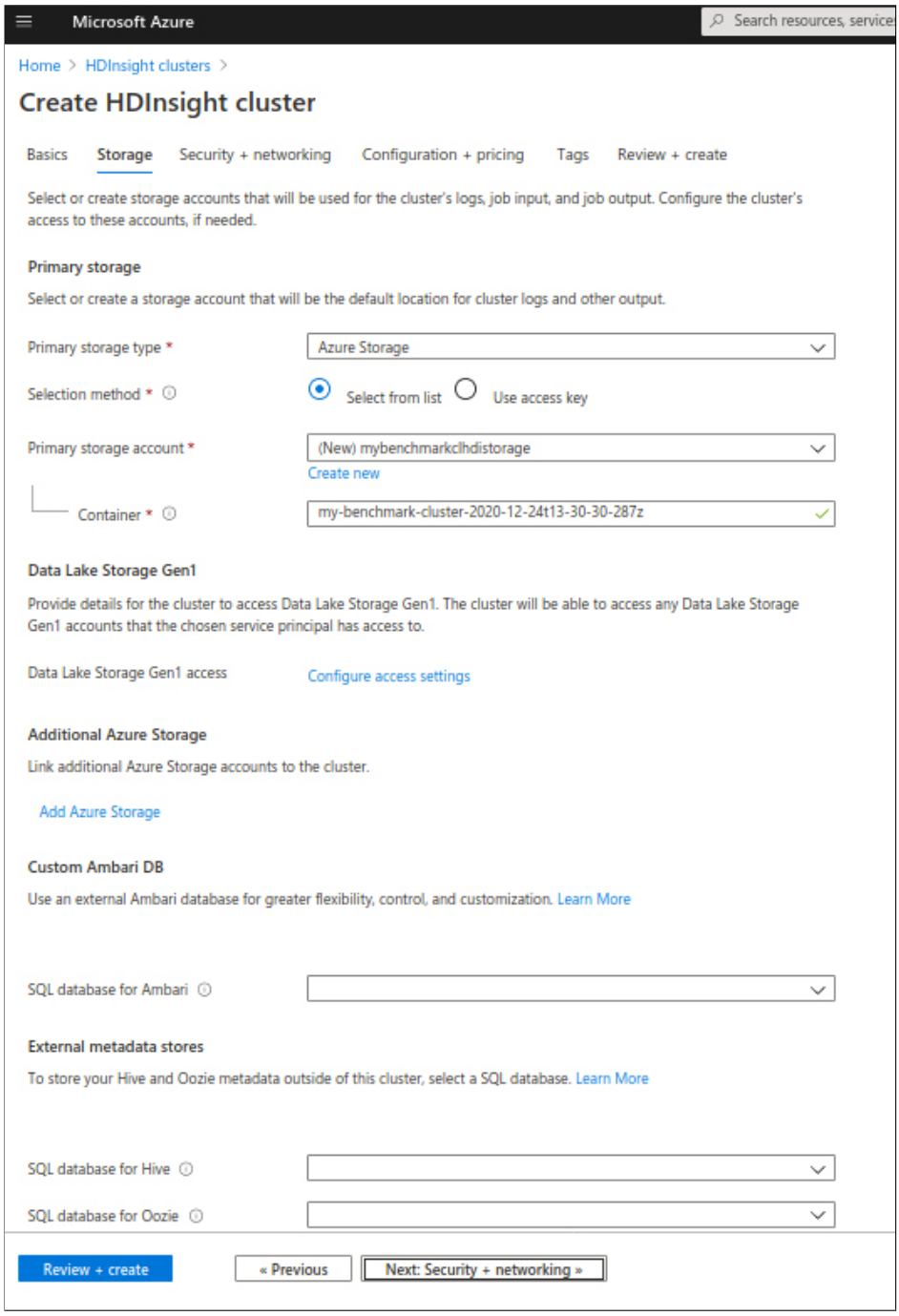

Figure 28: HDInsight setup - Storage

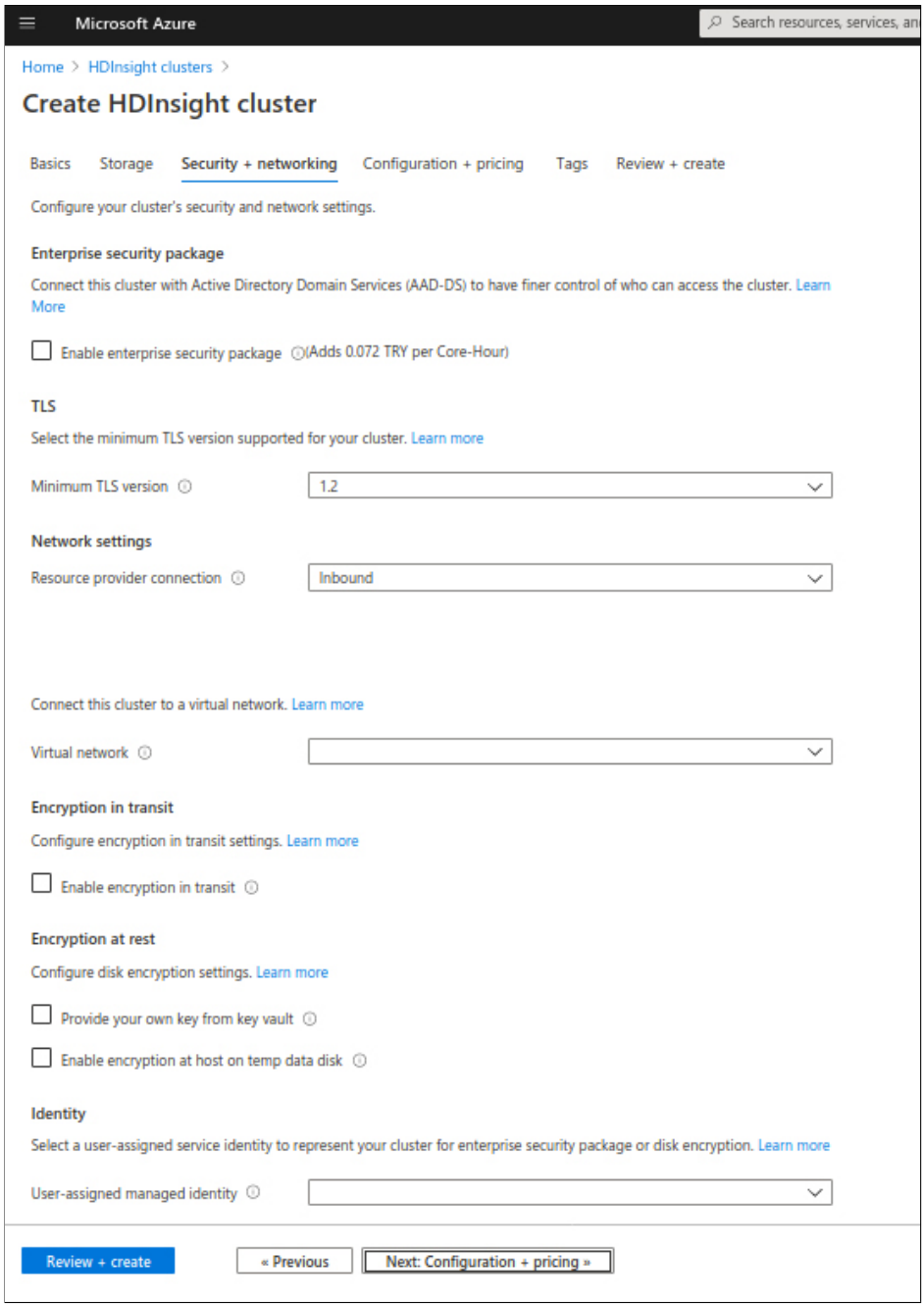

Figure 29: HDInsight setup - Networking

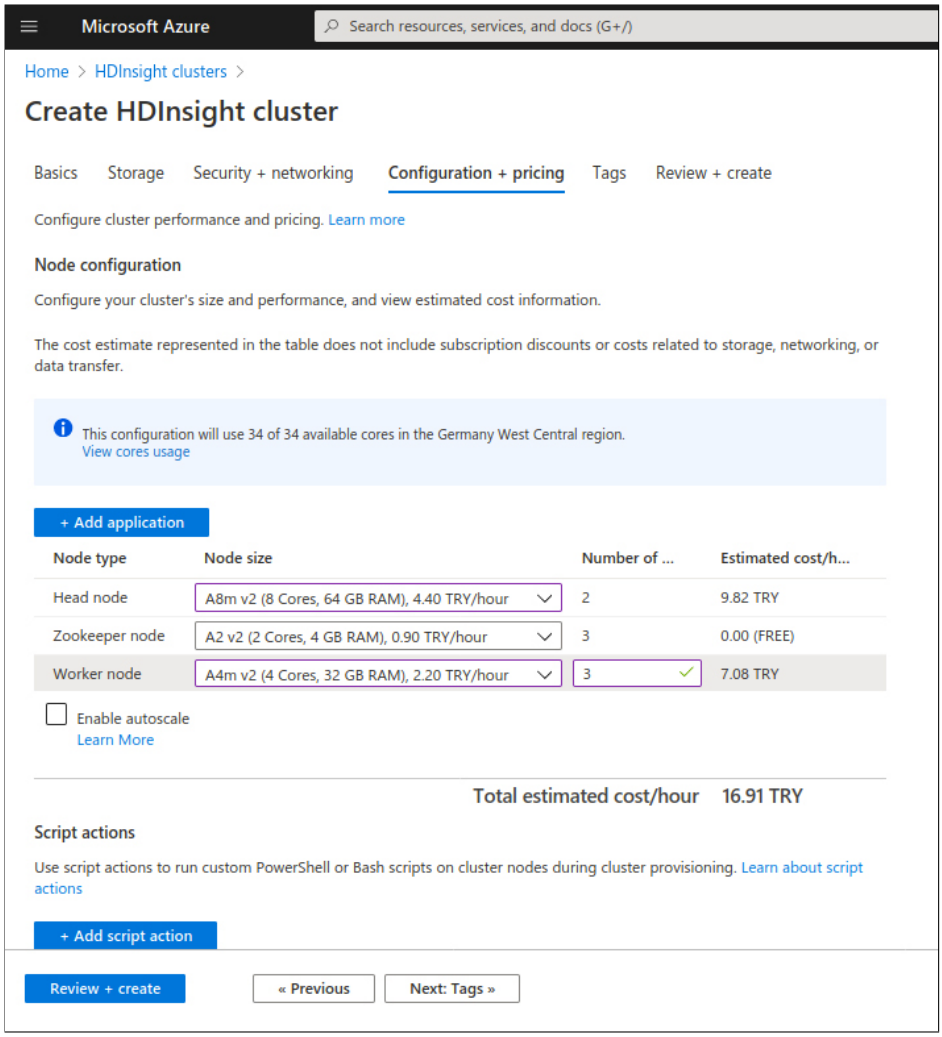

Figure 30: HDInsight setup - Configuration

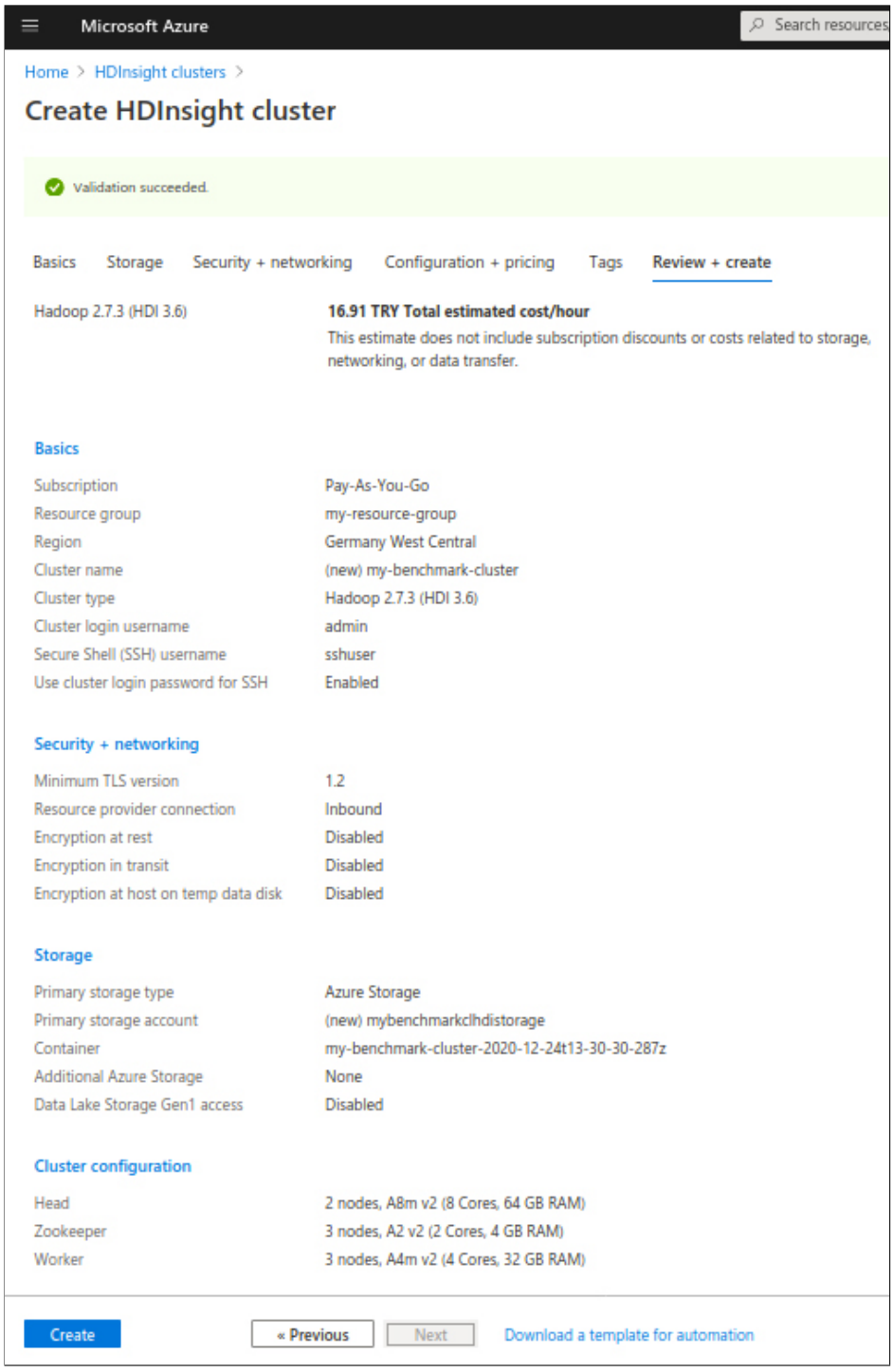

Figure 31: HDInsight setup - Review

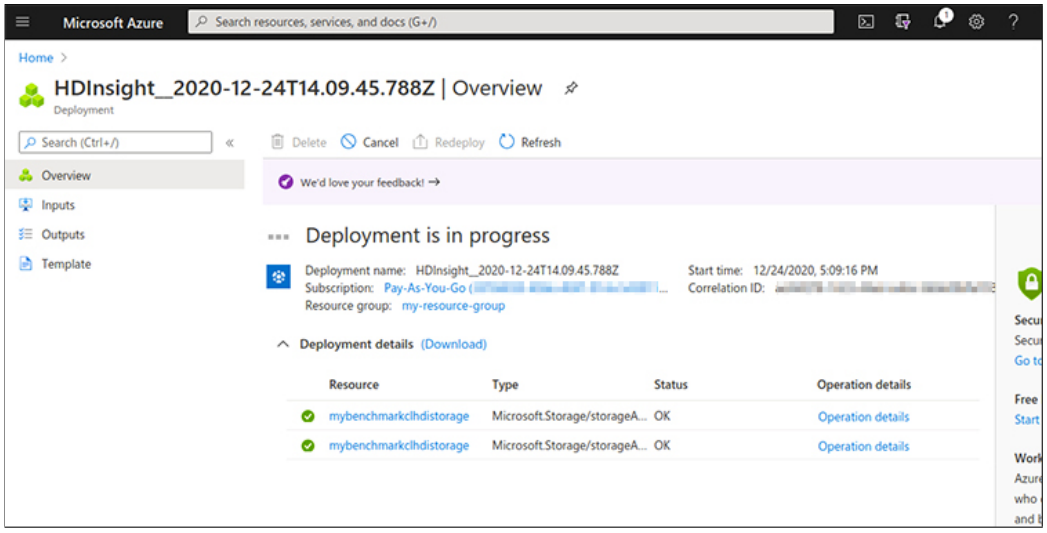

Figure 32: HDInsight setup - In progress

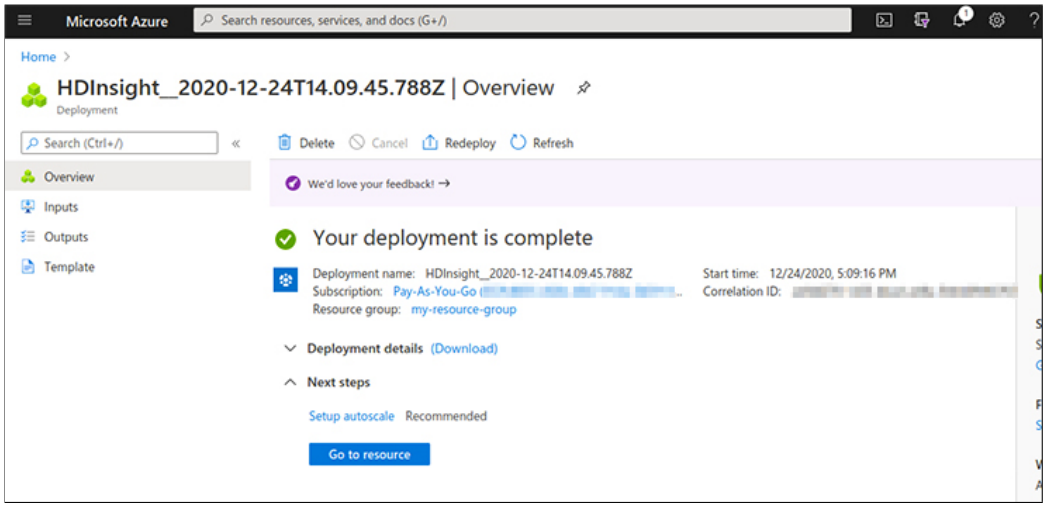

Figure 33: HDInsight setup - Complete

The HDInsight overview page gives access to Ambari web UI.

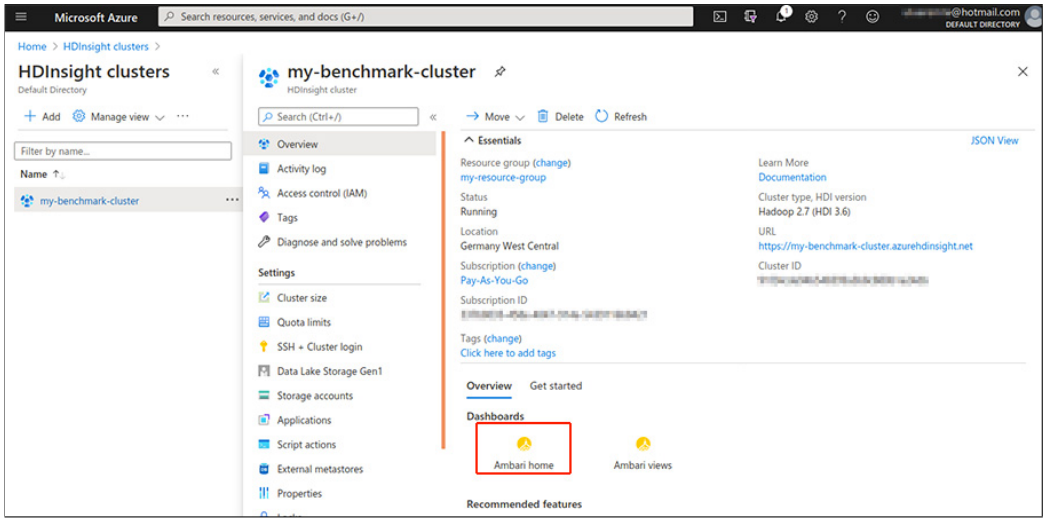

Figure 34: HDInsight setup - Overview

Figure 35 Ambari WebUI gives an overview to the current state of the cluster.

Figure 36 To conduct HiBench and system utilization data we need to connect to the cluster's master and worker nodes. From within the newly created HDI clusters's management portal, clicking on *SSH + Cluster Login*, then selecting the host name will provide us with an ssh link to connect to the master node.

Figure 37 Using the password created on the HDInsight installation process connects the CLI to the master node.

Figure 38 To connect to the worker nodes we list the nodes within the cluster to find out the machine names. In this case our worker node 0, 1, and 2 names are wn0-my-ben, wn2-my-ben, and wn4-my-ben respectively.

Figure 39 In new CLI windows for each worker node we first connect to the master node, then from within the master node we connect to the respective worker nodes. The master node screen on top of the other windows is where HiBench benchmarks are executed. Within the worker nodes scripts capturing system utilization during benchmarking are executed.

3 Creating e-MapReduce Cluster on Alibaba Cloud

An Alibaba Cloud account is required and can be created at the url https://www.alibabacloud.com.

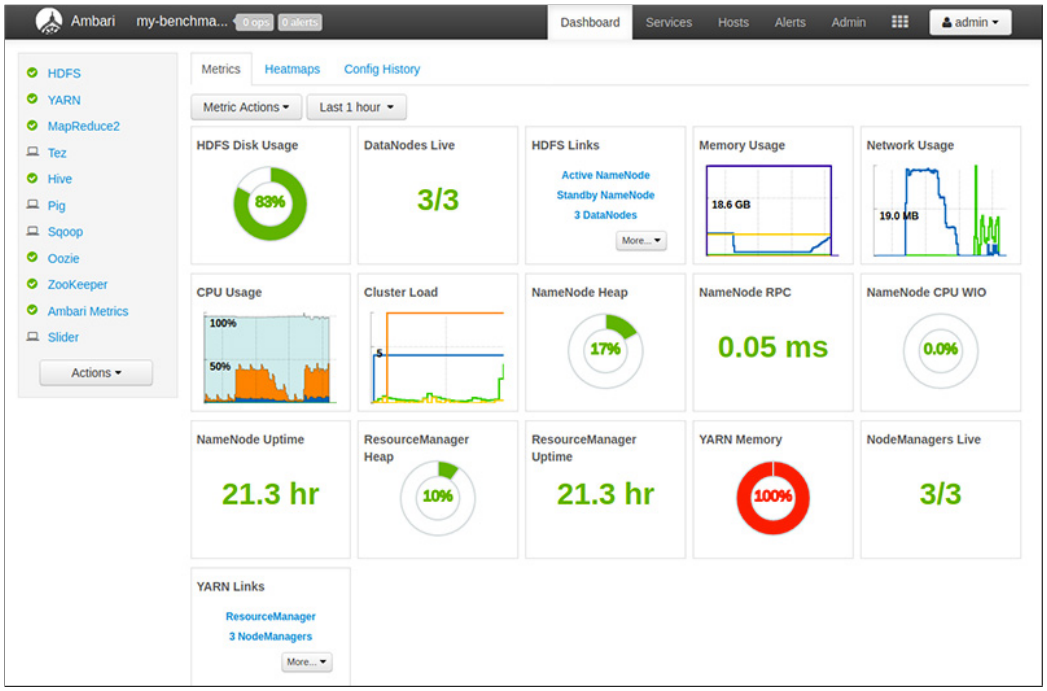

Figure 35: HDInsight - Ambari Dashboard

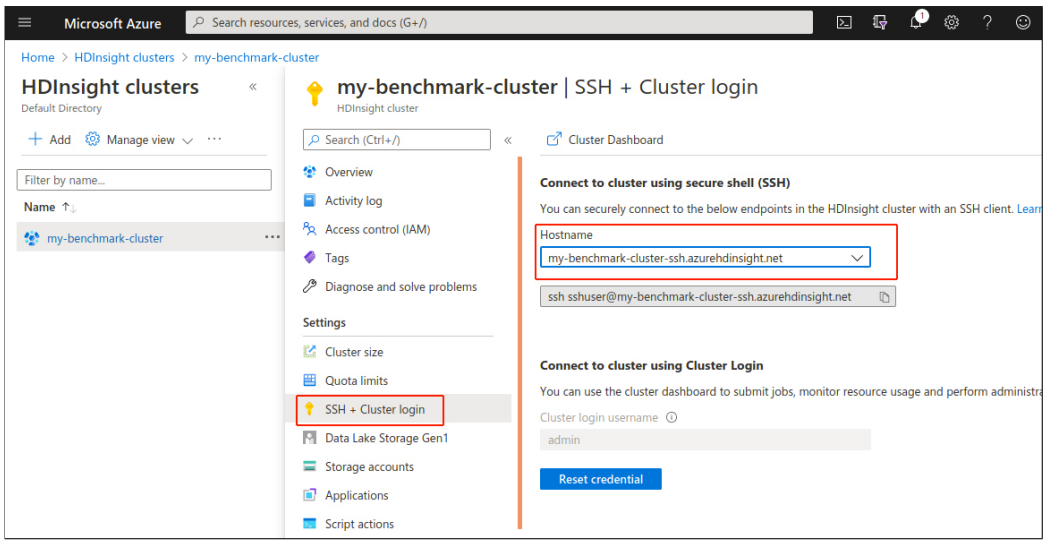

Figure 36: HDInsight - Cluster login

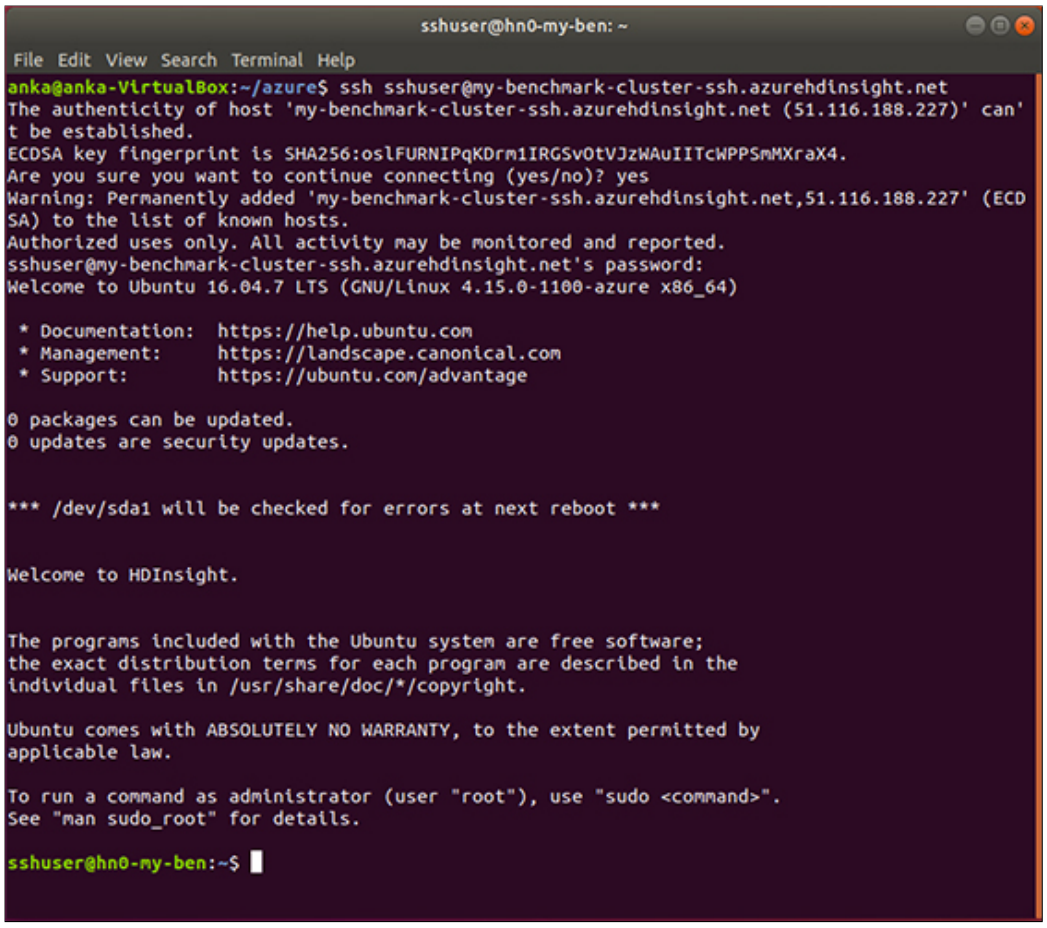

Figure 37: HDInsight - Connecting to master node

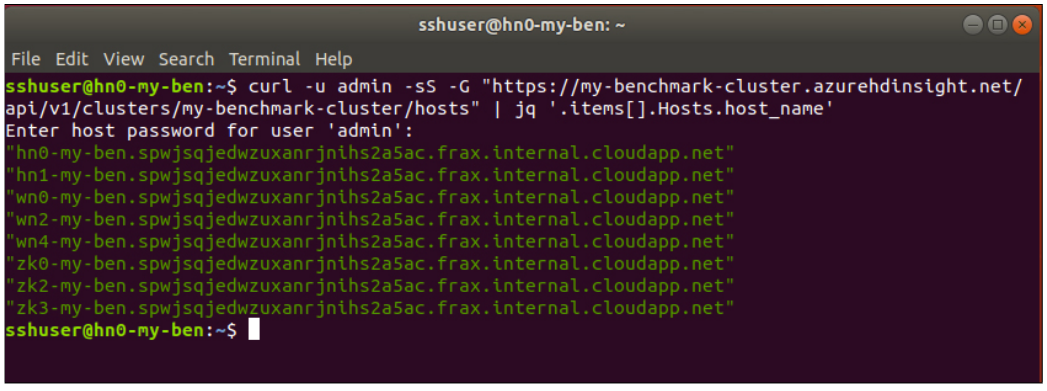

Figure 38: HDInsight - Listing worker nodes

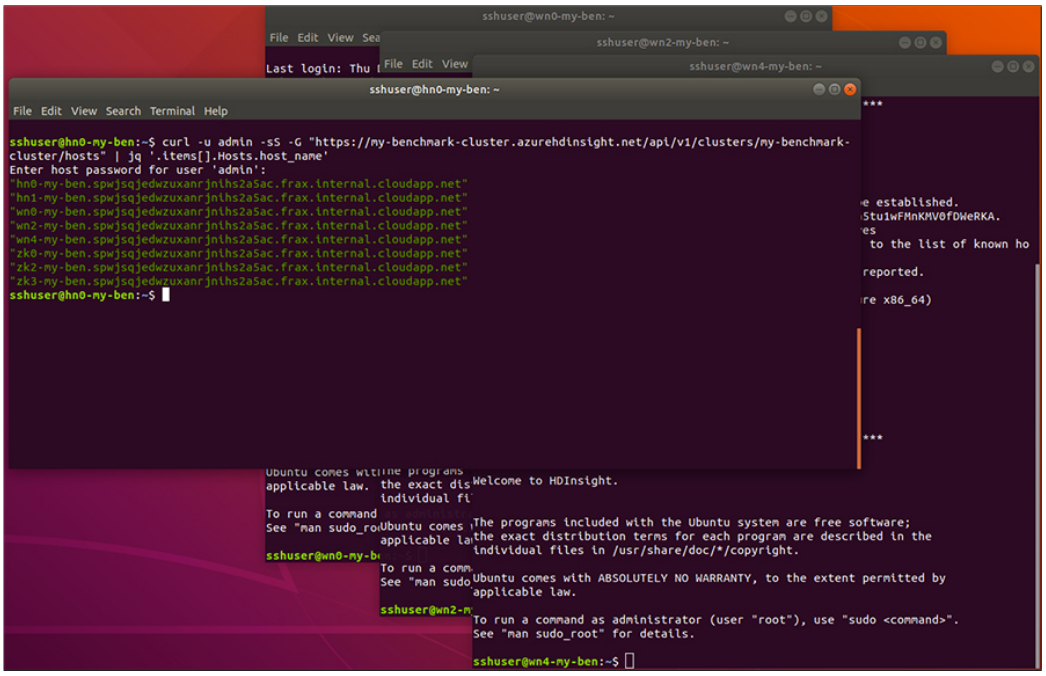

Figure 39: HDInsight - Ready to benchmark

Figure 40 First we create an access key by clicking AccessKey Management link on the upper right hand from the user menu and following the instructions.

Figure 41, 42, 43 For e-MapReduce to run, OSS (Object Storage Service) also needs to be activated.

Figure 44 In OSS, setup a bucket for e-MapReduce to store logs.

Figure 45 Alibaba leverages OSS to store access logs. Within the created bucket's settings enable logging access to the bucket.

Figure 46 Before starting with e-MapReduce, select *Germany, Frankfurt* as region, where cluster will be created.

Figure 47 From Alibaba Cloud dashboard, select e-MapReduce.

Figure 48 From Alibaba Cloud EMR console, click on Cluster Wizard to start cluster installation.

Figure 49 *EMR-3.32.0* is the image version supporting Hadoop 2, and subject to selection.

Figure 50 Make selections below regarding Hardware settings. Existing VPC switch and a security group are required, these can be created by clicking on relevant links on the below screenshot (CreateVPC/V switch and Create Security Group). High availability of Master Node is not required in our case, so we leave

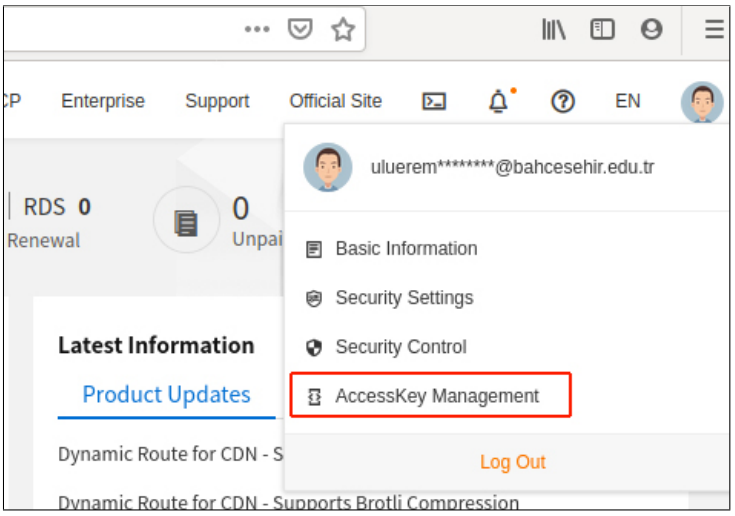

Figure 40: e-MapReduce - Create access key

it unchecked as its default.

Figure 51 For Master Instance selections we chose memory optimized pre-configurations due to namenode's high memory load. Select *ecs.se1.2xlarge* as machine type, specify 500 GB for Master Node data storage.

Figure 52 For the worker nodes (inside the tab Core Instances) select *ecs.se1.xlarge*, in order to leverage 4 CPUs and 32 GB RAM per worker node. Storage capacity is set to 1000 GB (280 GB * 4 disks, considering that 120 GB system disk size reserved for each node) for data. Leaving system storage capacity with its default of 120 GB. Number of worker nodes is specified as 3.

Figure 53 Final step before confirming installation is to give a name for the cluster and specify password for operations. Enable the checkbox *Assign Public IP Address* for later accessing the cluster over internet. Enable Remote Logon and select ssh key created before, if there is no existent ssh key pair, *Create Key Pair* next to the key pair selection box to do so.

Figure 54 Review settings. On the bottom of the page activate checkbox to accept terms and click on *Create*.

Figure 55 The creation process is displayed on e-MapReduce cluster page.

Figure 56 Within Alibaba Cloud's *Elastic Compute Service*, *Instances* page, single VMs of the cluster are also available.

Figure 57 During the SSH key pair creation, respective pem file is downloaded to the local machine. Using the pem file a passwordless secure connection from Linux machine is possible.

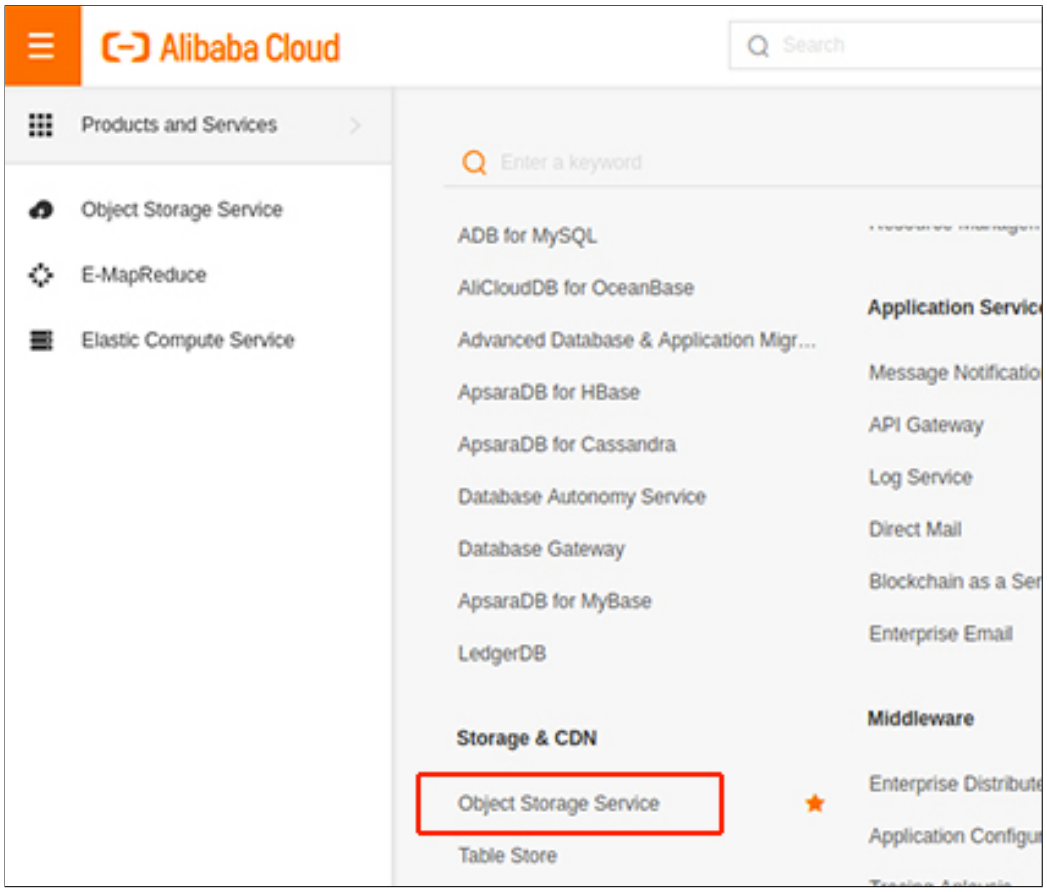

Figure 41: e-MapReduce - Activate OSS (Step 1)

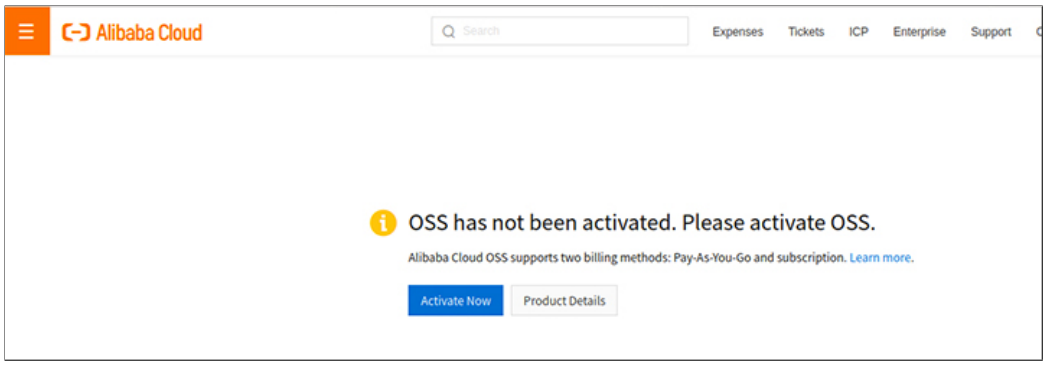

Figure 42: e-MapReduce - Activate OSS (Step 2)

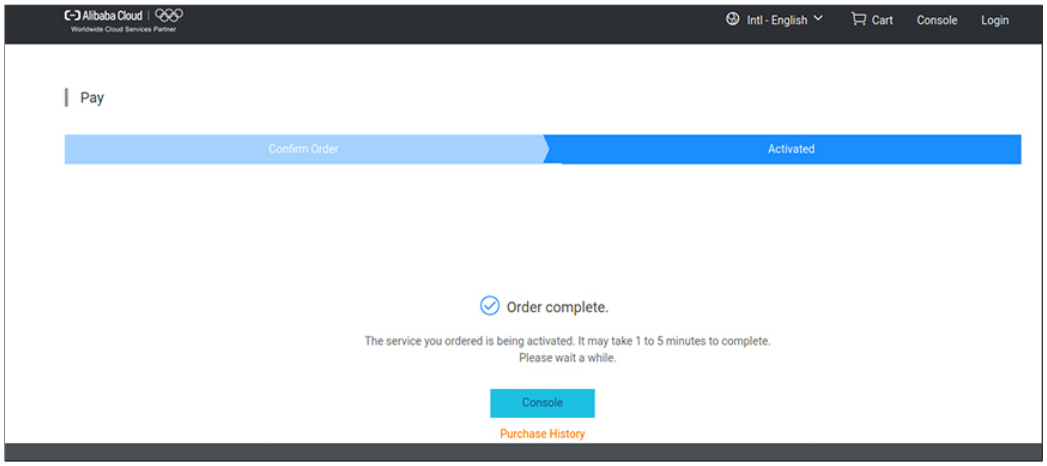

Figure 43: e-MapReduce - Activate OSS (Step 3)

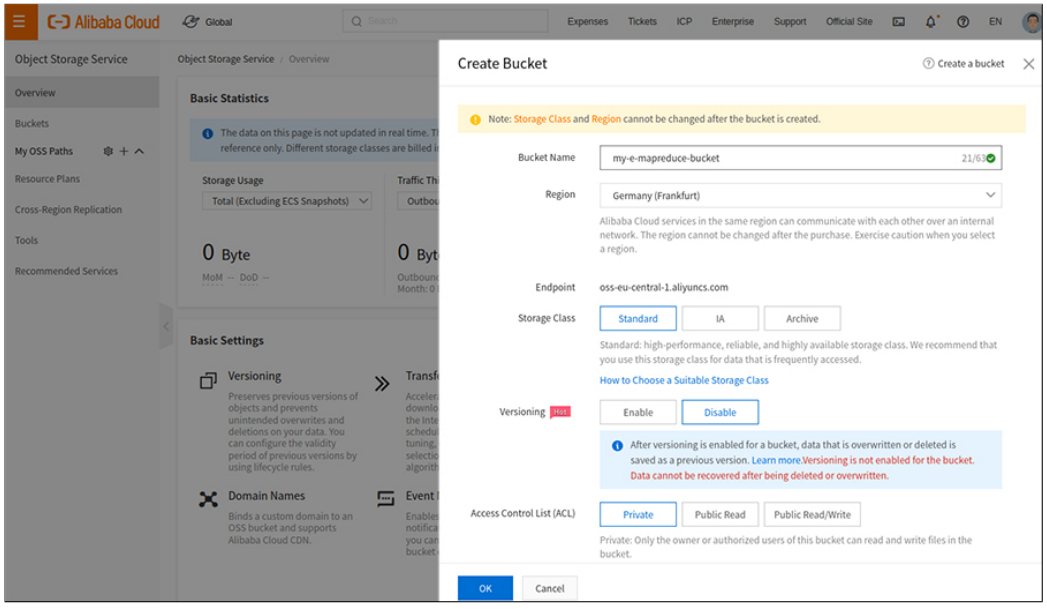

Figure 44: e-MapReduce - Create OSS bucket

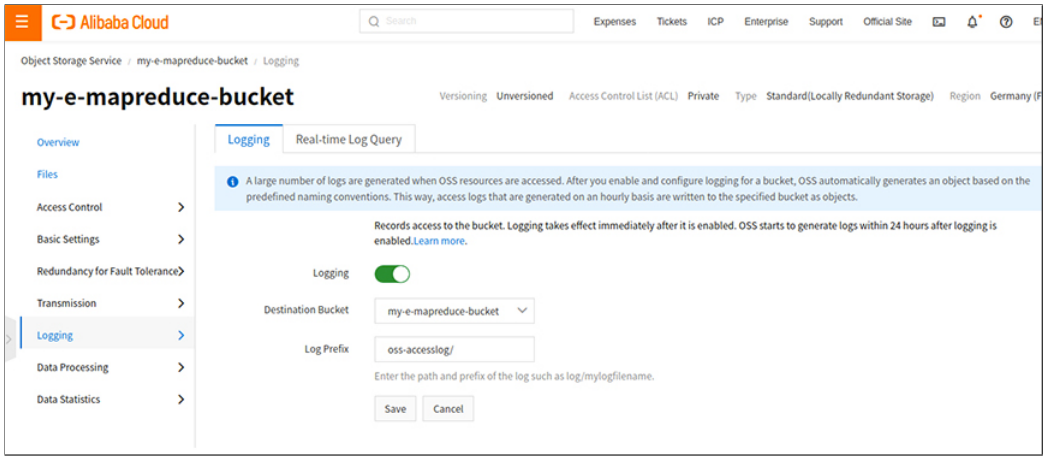

Figure 45: e-MapReduce - Enabling access logging on OSS

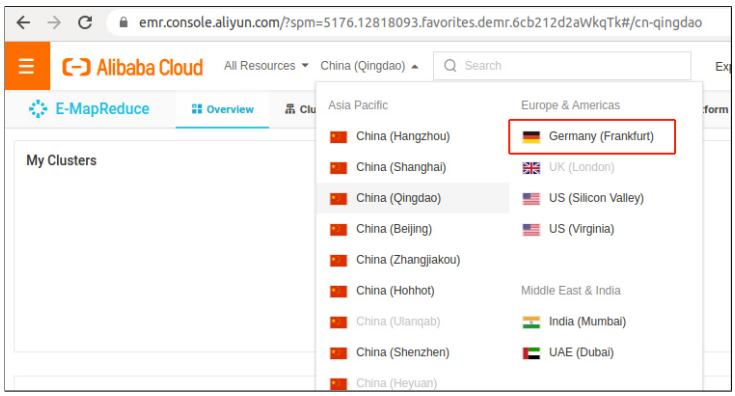

Figure 46: e-MapReduce - Select Region

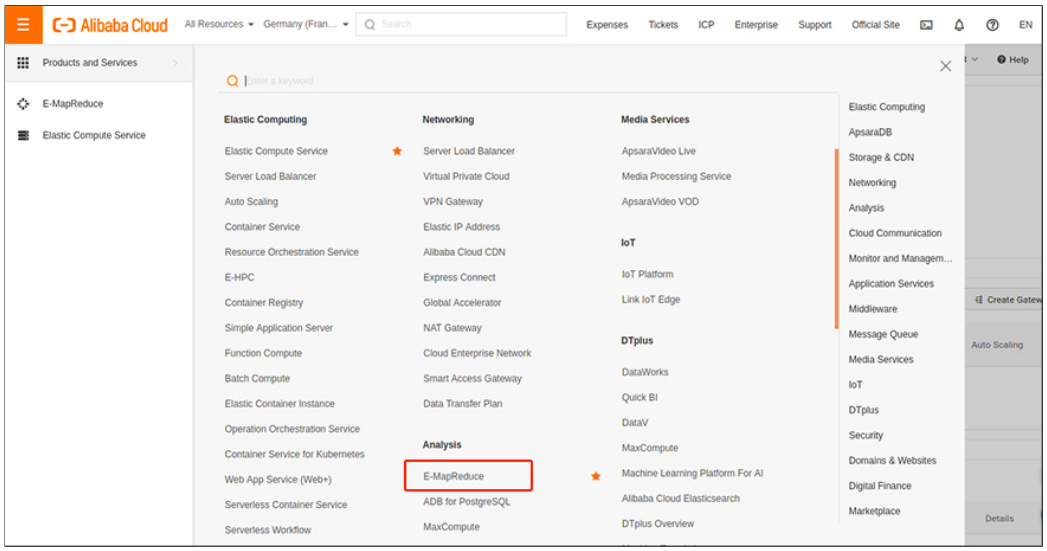

Figure 47: e-MapReduce - Create cluster (Step 1)

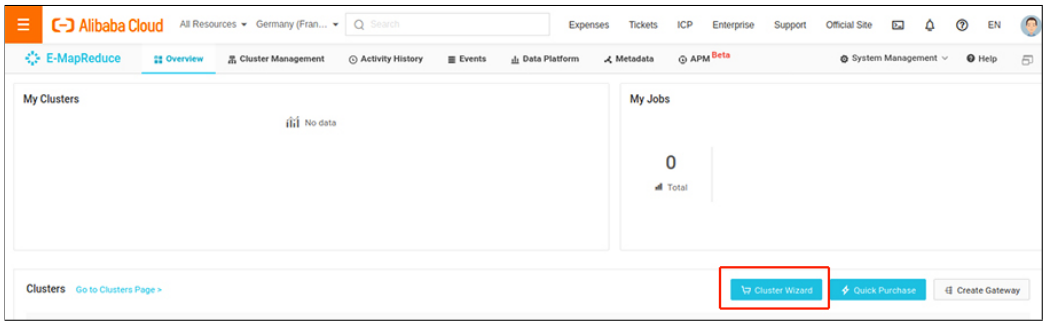

Figure 48: e-MapReduce - Create cluster (Step 2)

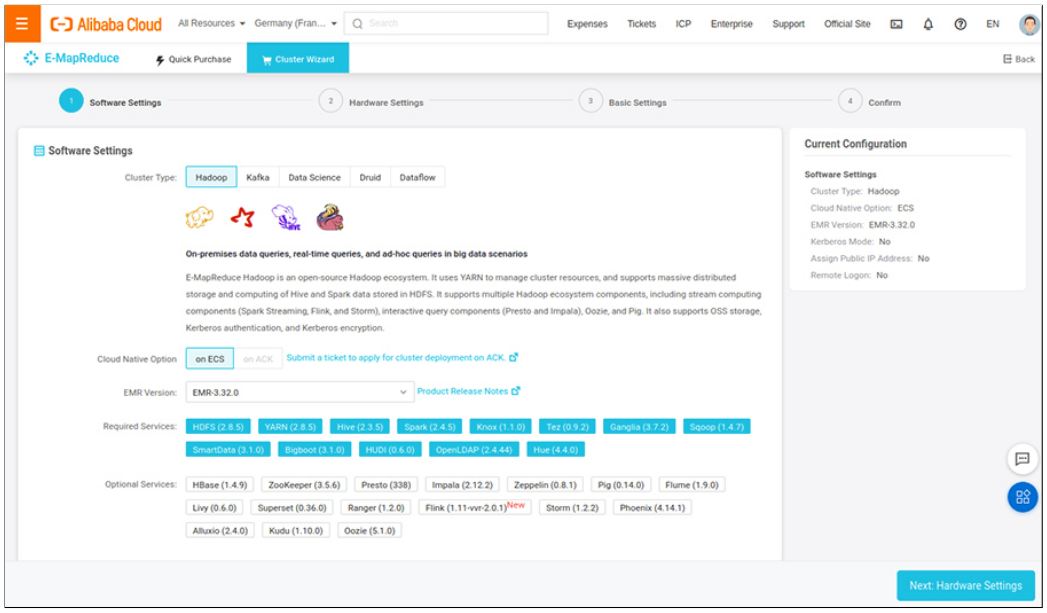

Figure 49: e-MapReduce - Create cluster (Step 3)

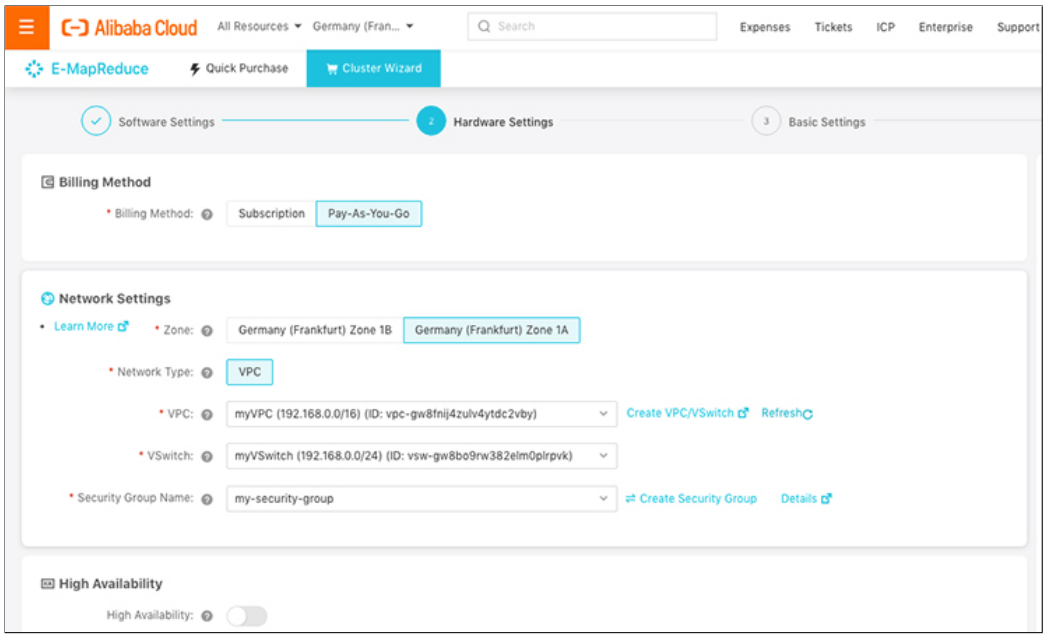

Figure 50: e-MapReduce - Create cluster (Step 4)

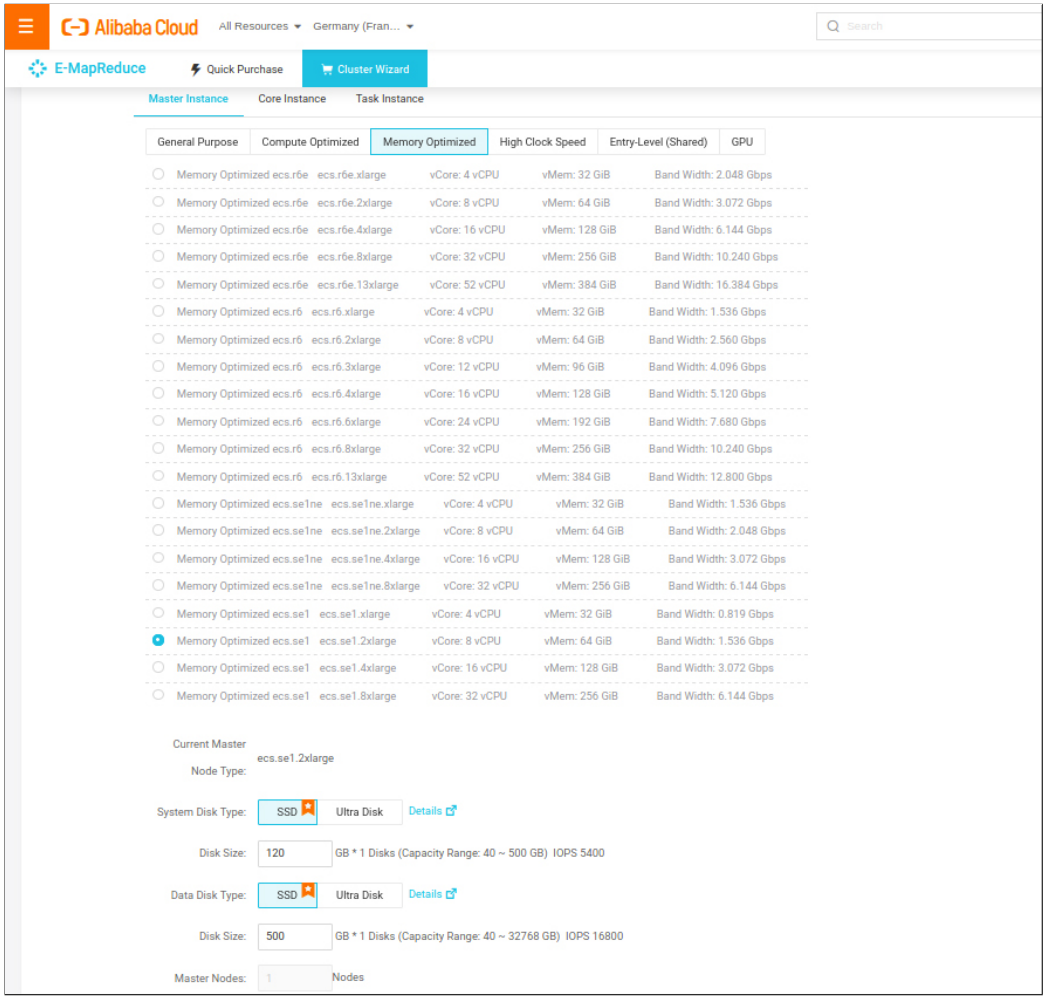

Figure 51: e-MapReduce - Create cluster (Step 5)

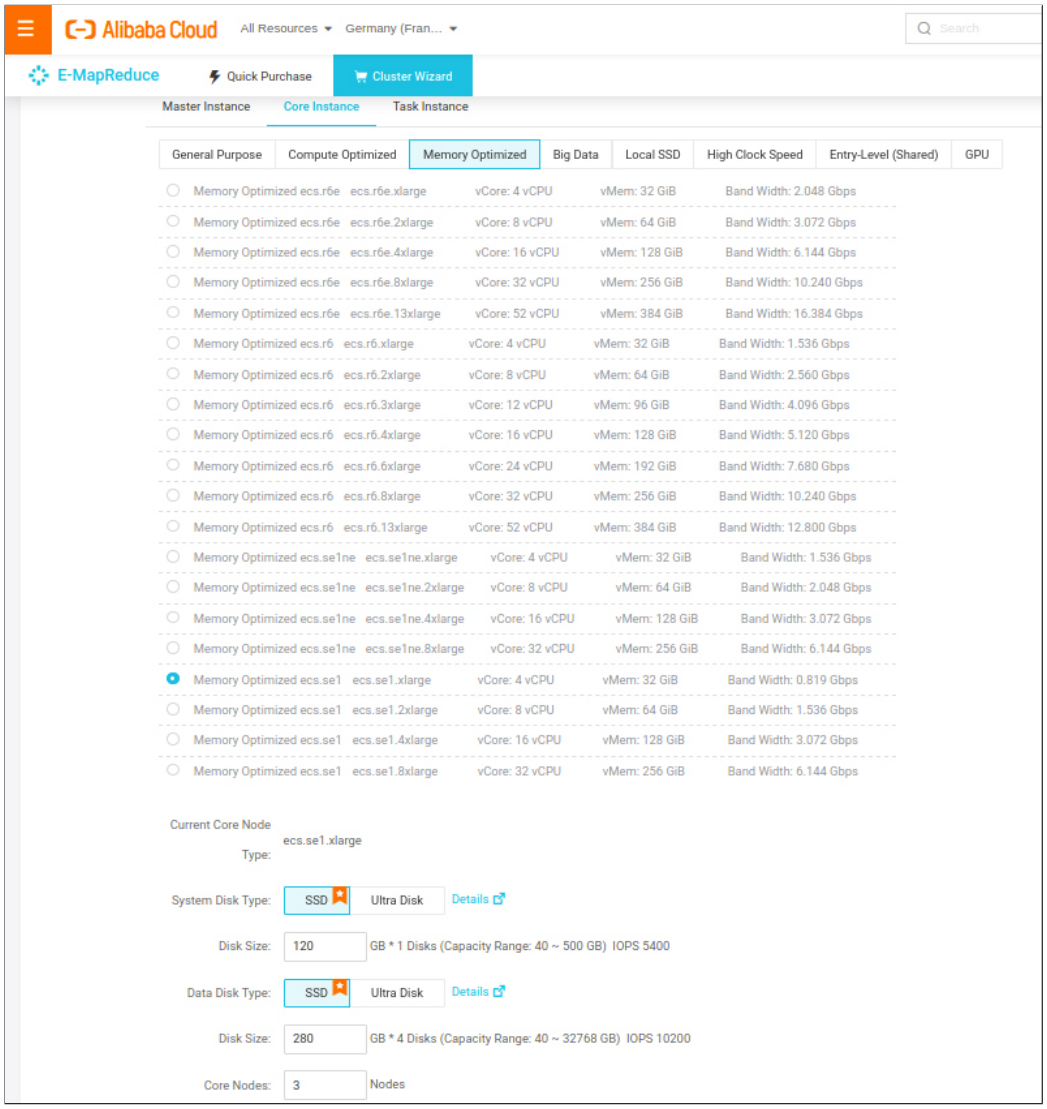

Figure 52: e-MapReduce - Create cluster (Step 6)

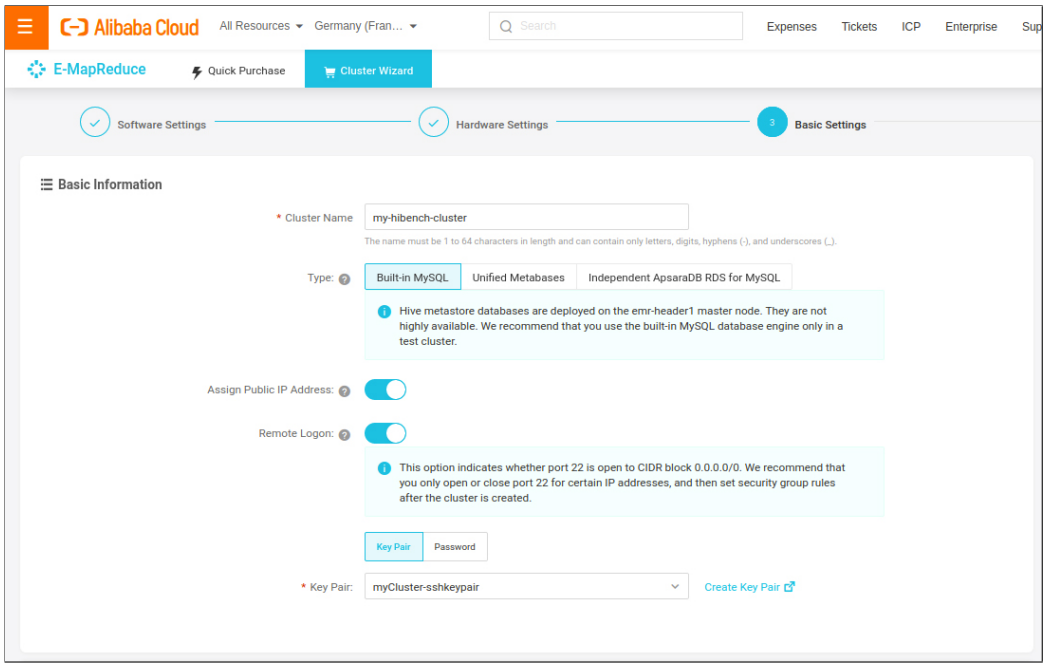

Figure 53: e-MapReduce - Create cluster (Step 7)

```
$ chmod 400 ssh-keypair-name.pem
```

Within the terminal shh connection is made by following command (instead of 0.0.0.0 the public IP address of the master node is entered):

```
$ ssh -i ssh-keypair-name.pem root@0.0.0.0
```

Entering following command from master node to check the Hadoop cluster:

```
$ hdfs dfsadmin -report
```

Figure 58 Worker nodes of the cluster are not accessible directly over the internet; connection to worker nodes is made over master node with following commands:

```
$ su hadoop
$ ssh emr-worker-1
```

Figure 59 Handling broken pipe error: The CLI may fall into timeout causing connection break. To prevent this, we set up the ssh configuration file to send every 30 seconds an empty packet to the server, this will hold the connection live.

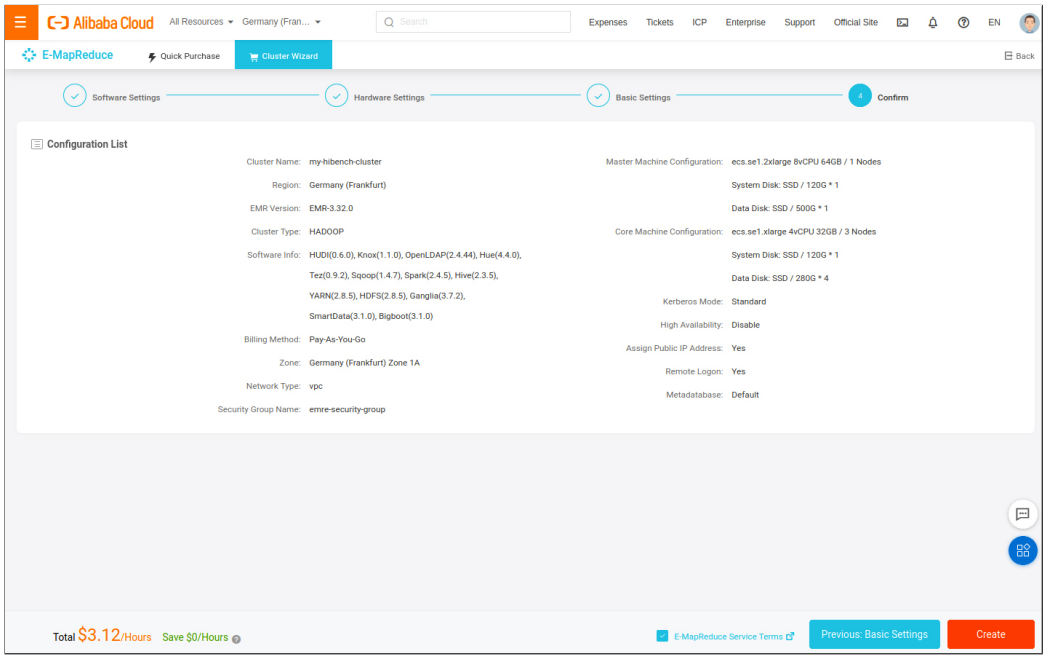

Figure 54: e-MapReduce - Create cluster (Step 8)

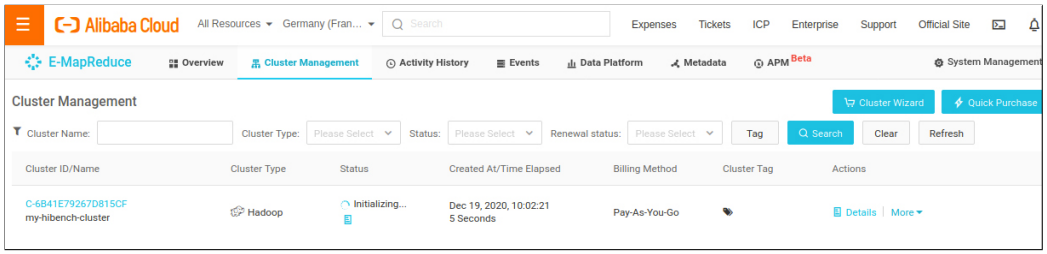

Figure 55: e-MapReduce - Create cluster (Step 9)

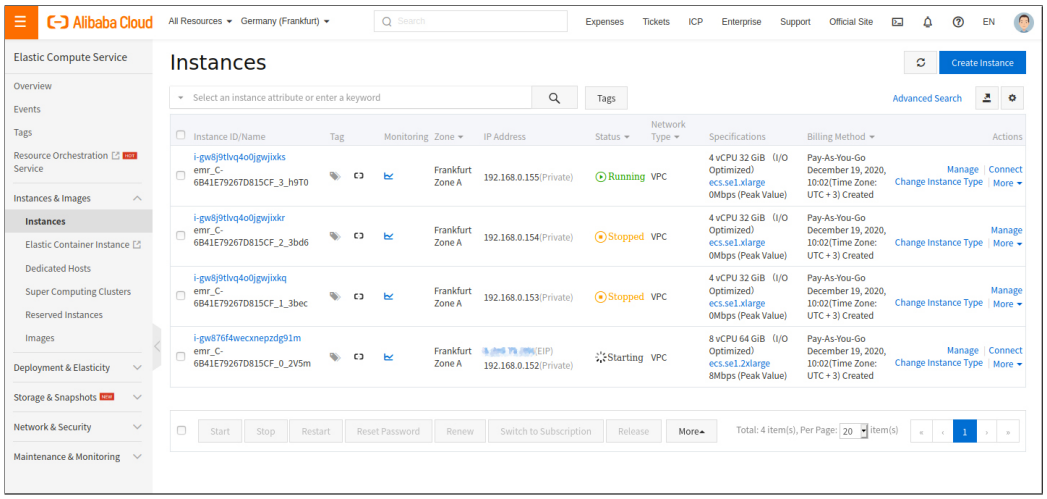

Figure 56: e-MapReduce - Create cluster (Step 10)

```
$ sudo nano ~/.ssh/config
```

Figure 60 depicts CLI connection on master and worker nodes of our e-MapReduce cluster, meaning that our system is ready to go with the benchmark processes.

4 Running HiBench on GCP Dataproc

On all master and worker nodes, we update repositories and dependencies.

```
$ sudo apt update
$ sudo apt upgrade
```

For data collection of worker nodes' system resource utilization, install sysstat on all 3 worker nodes.

```
$ sudo apt install sysstat
$ sar -V
    sysstat version 11.6.1
    (C) Sebastien Godard (sysstat <at> orange.fr)
```

On worker nodes, create a directory for storing resouce utilization outputs:

```
$ mkdir data
```

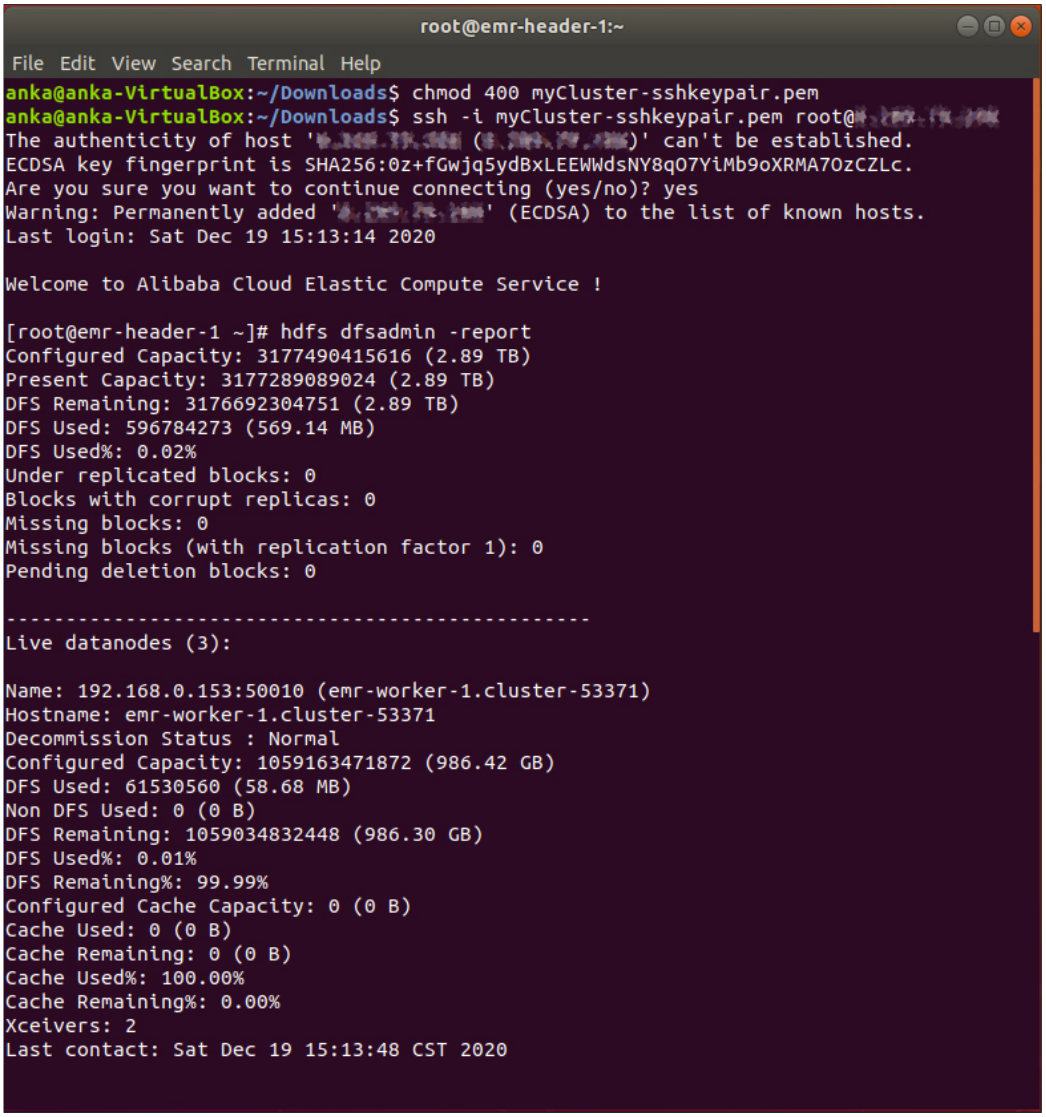

Figure 57: e-MapReduce - Connecting to the cluster

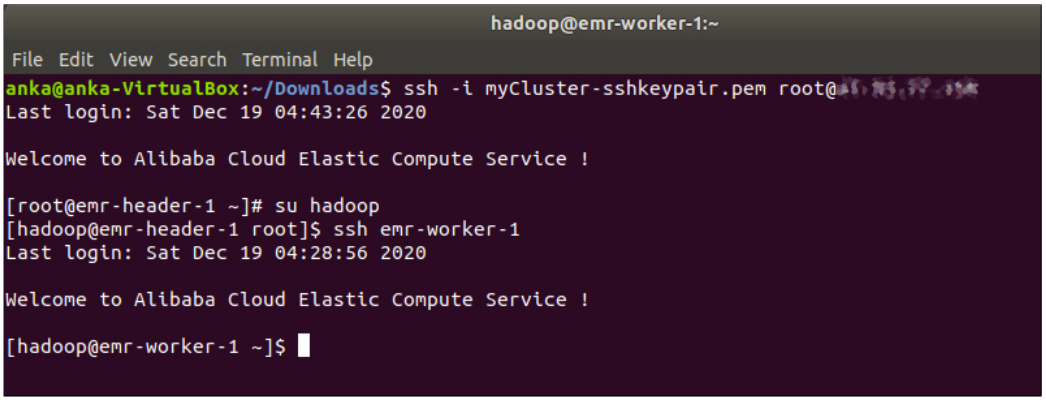

Figure 58: e-MapReduce - Connecting to worker nodes

Figure 59: e-MapReduce - Handling broken pipe error

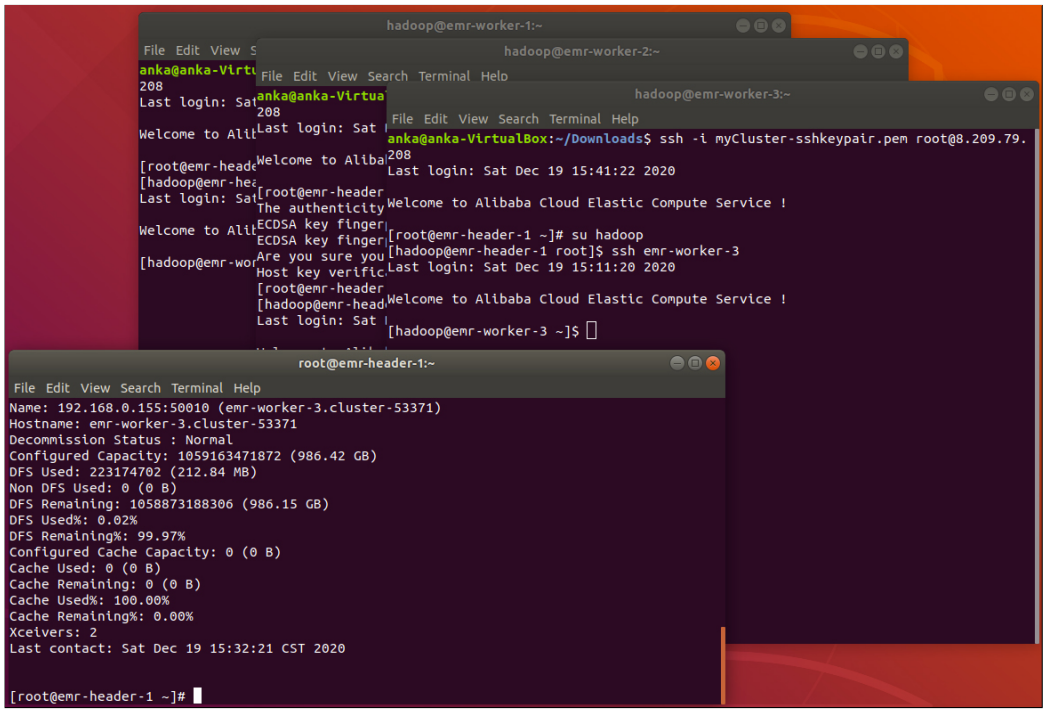

Figure 60: e-MapReduce - Ready to benchmark

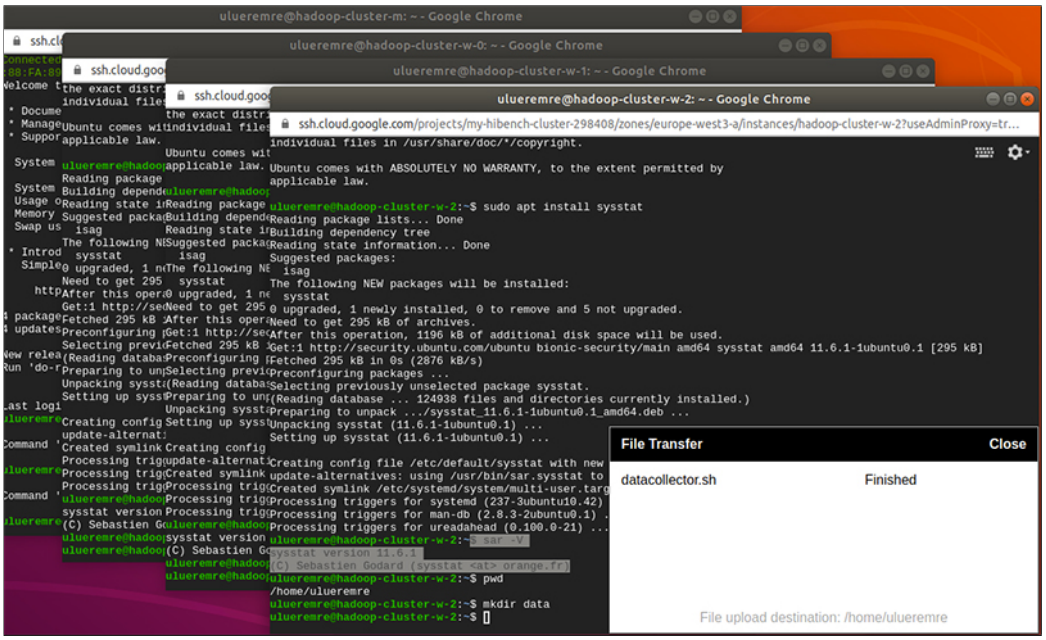

Figure 61: HiBench Dataproc - Uploading datacollector.sh

Figure 61 Upload datacollector script on each worker node.
Make datacollector.sh executable.

```
$ sudo chmod +x datacollector.sh
```

HiBench and related processes are executed on the master node. Apache Maven is required to build HiBench. The first step is to install maven:

```
$ sudo apt install maven
```

Figure 62 Verify maven installation.

```
$ mvn --version
```

HiBench works with python 2, we check if python2 is installed

```
$ python2 -V
    Python 2.7.17
```

Download HiBench 7.1.1

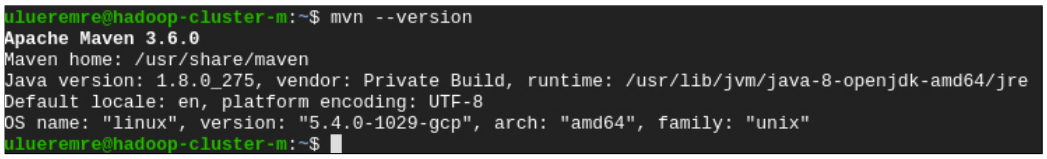

Figure 62: HiBench Dataproc - Verify Maven installation

```
$ wget https://github.com/Intel-bigdata/HiBench/archive/
  HiBench-7.1.tar.gz
```

Untar the downloaded file.

```
$ tar -zxf HiBench-7.1.tar.gz
```

Rename the extracted folder to a more user friendly name

```
$ mv HiBench-HiBench-7.1 HiBench
```

Navigate to HiBench folder

```
$ cd HiBench
```

Build HiBench7.1for Hadoop

```
$ mvn -Phadoopbench -Dspark=2.4 -Dscala=2.12 clean package
```

Figure 63 Upon successful HiBench compilation, an informative success message occurs. During the compilation failures might occur, re-running the above command would mostly fix this issue.
To modify HiBench's configuration files, navigate to HiBench's conf folder. In here, we will modify Hadoop and HiBench configurtions.

```
$ cd conf/
$ cp hadoop.conf.template hadoop.conf
$ sudo nano hadoop.conf
```

hibench.hdfs.master value can be found in /usr/lib/hadoop/etc/hadoop/core-site.xml file (fs.default.name). Here we need to specify the hibench.hadoop.examples.test.jar manually for HiBench to run. Otherwise HiBench raises an Assertion error.

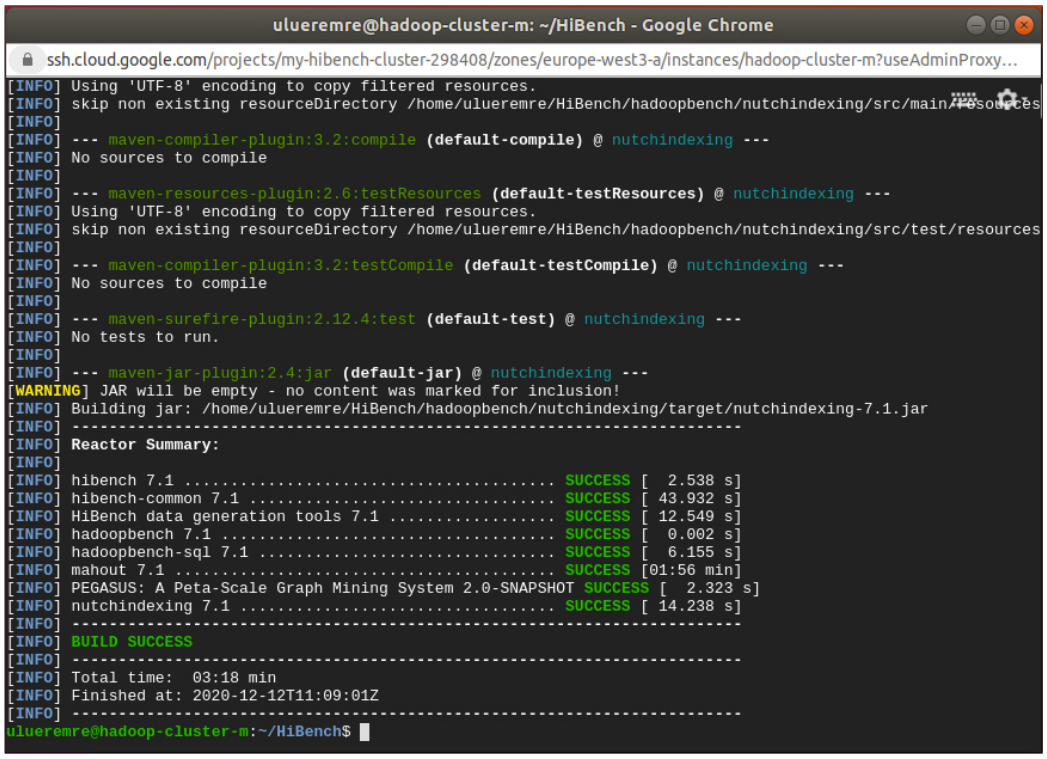

Figure 63: HiBench Dataproc - Success compiled

```
$ sudo nano hadoop.conf
    # Hadoop home
    hibench.hadoop.home     /usr/lib/hadoop

    # The path of hadoop executable
    hibench.hadoop.executable     /usr/lib/hadoop/bin/hadoop

    # Hadoop configuration directory
    hibench.hadoop.configure.dir  /usr/lib/hadoop/etc/hadoop

    # The root HDFS path to store HiBench data
    # hibench.hdfs.master value can be found in
    # /usr/lib/hadoop/etc/hadoop/core-site.xml file
    # (fs.defaultFS)
    hibench.hdfs.master
                           hdfs://hadoop-cluster-m

    hibench.hadoop.examples.test.jar /usr/lib/hadoop-mapreduce/
    hadoop-mapreduce-client-jobclient-2.9.2-tests.jar

    # Hadoop release provider. Supported value:
    # apache, cdh5, hdp
    hibench.hadoop.release    apache
```

Figure 64 depicts specified Hadoop settings for HiBench.
Figure 65 HiBench related configuration settings like data scale and mappers/reducers count are made within *Hibench/conf/hibench.conf* file. Having 12 cores on the worker nodes, we specify a default of 12 mappers and 12 reducers to allocate during benchmark. For each data scale, before we run the benchmarks, *hibench.scale.profile* value has to be updated for the respective data scale:

```
$ sudo nano hibench.conf
```

As a showcase, continue with the manual implementation of UseCase 1.
The benchmarks are executed from within root of HiBench folder. Within Use Case 1 and Use Case 2, benchmark tasks have been executed in an iterative process. We donot put every line of code in here, but to give the idea, the approach is given below:

- Within hibench.conf configuration file, we set up numbers of 12 mappers and 12 reducers.

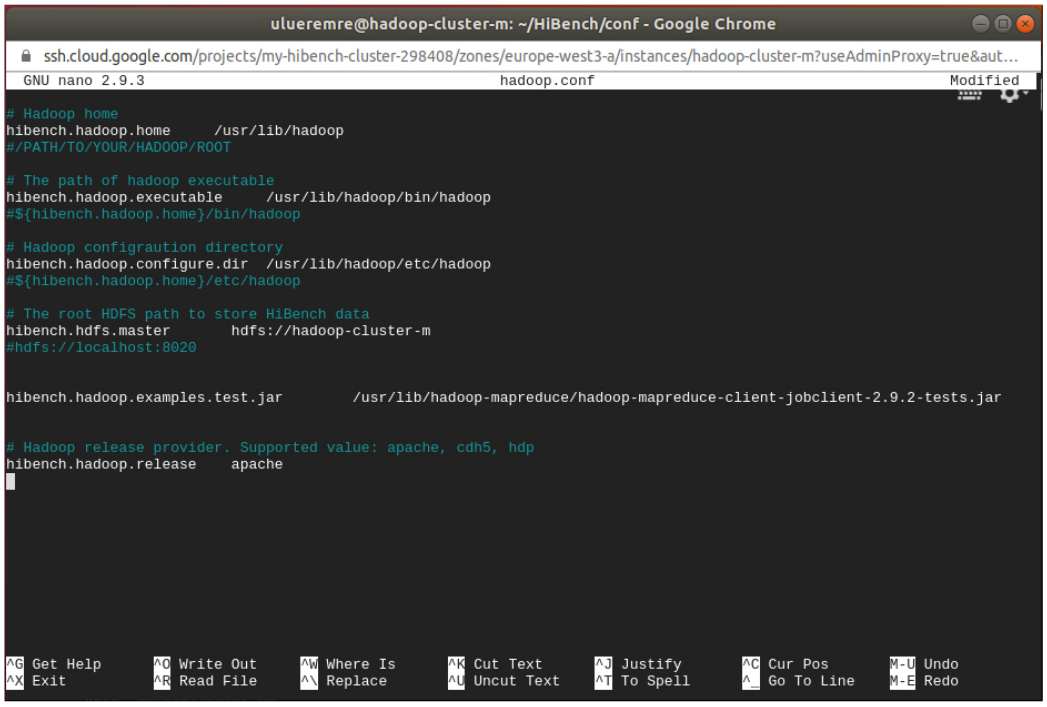

Figure 64: HiBench Dataproc - Hadoop configurations

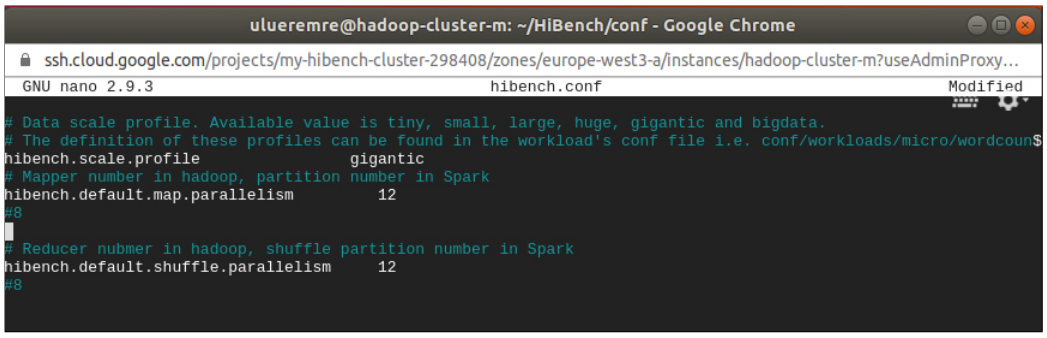

Figure 65: HiBench Dataproc - HiBench configurations

- Set up hibench.conf configuration file with the respective data scale (tiny, small, large, huge, and gigantic. Due to account limitations we ignore the largest scale bigdata)
- On each worker node we located previously written bash script responsible for system activity collection in a specific directory (data directory). Datacollection script is started manually on all worker nodes short before a benchmark is run (not during preparation of the workload)
- On master node within HiBench, start the preparation script (no running data collectors on worker nodes)
- Once preparation of the workload is finished, start datacollector script o worker nodes
- On the master node start benchmark (HiBench/bin/workloads/¡benchmark-class¿/¡benchmarkname¿/hadoop/run.sh)
- Once the benchmark completes, stop data collection process on all worker nodes.

Example codes below firstly create the workload for the respective benchmark; 2, 3, and 4 are run across worker nodes one after another starting to capture system utilization by leveraging datacollector.sh script. Immediately in step 5, the respectie benchmark is executed. After the benchmark completion, data collecting activities on the worker nodes are terminated.

```
# 1. Prepare workload for benchmark MICRO-TERASORT
$ bin/workloads/micro/terasort/prepare/prepare.sh

# 2. Collect system utilization on Worker 0
$ ./datacollector.sh -p gcp-uc1-w0-g-tera

# 3. Collect system utilization on Worker 1
$ ./datacollector.sh -p gcp-uc1-w1-g-tera

# 4. Collect system utilization on Worker 2
$ ./datacollector.sh -p gcp-uc1-w2-g-tera

# 5. Run the benchmark MICRO-TERASORT
$ bin/workloads/micro/terasort/hadoop/run.sh
```

```
# 6. STOP data collection at worker nodes by pressing Ctrl+C)
```

The screenshot in Figure 66 depicts an ongoing benchmark execution with simultaneous data collection on the worker nodes. The CLI to the master node runs TeraSort benchmark while in parallel, in the worker nodes w0, w1, and w2 *datacollector.sh* script is active.

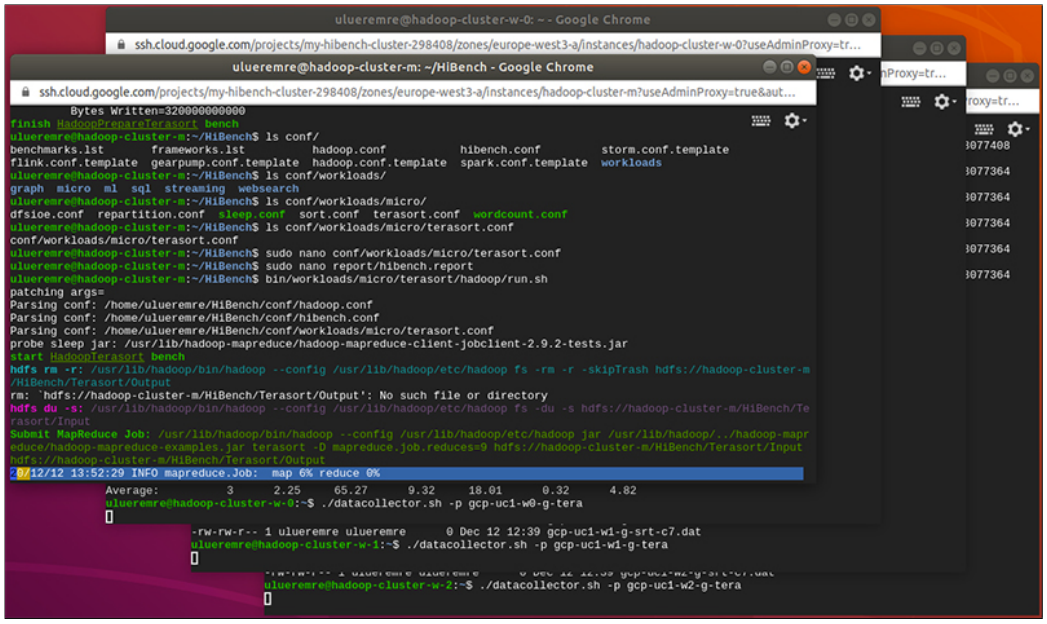

Figure 66: HiBench Dataproc - Profile from an ongoing benchmark execution

5 Running HiBench on Azure HDInsight

Apache Maven is required to build HiBench. The first step is to install maven on the master node:

```
$ sudo apt install maven
```

Verify maven installation.

```
$ mvn -version
    Apache Maven 3.3.9
    Maven home: /usr/share/maven
    Java version: 1.8.0_275, vendor: Private Build
    Java home: /usr/lib/jvm/java-8-openjdk-amd64/jre
    Default locale: en_US, platform encoding: ANSI_X3.4-1968
    OS name: "linux", version: "4.15.0-1100-azure",
        arch: "amd64", family: "unix"
```

Download HiBench 7.1.1

```
$ wget https://github.com/Intel-bigdata/HiBench/archive/
  HiBench-7.1.tar.gz
```

Untar the downloaded file.

```
$ tar -zxf HiBench-7.1.tar.gz
```

Rename the extracted folder to a more user friendly name.

```
$ mv HiBench-HiBench-7.1 HiBench
```

Navigate to HiBench folder.

```
$ cd HiBench
```

Build HiBench7.1 for Hadoop.

```
$ mvn -Phadoopbench -Dspark=2.4 -Dscala=2.12 clean package
```

To modify HiBench configuration files, navigate to HiBench's conf folder. For HDInsight we also need to specify hibench.hadoop.examples.test.jar path and hadoop release (hpd)

```
$ cd conf/
$ cp hadoop.conf.template hadoop.conf
$ sudo nano hadoop.conf
# Hadoop home
hibench.hadoop.home     /usr/hdp/current/hadoop-client

# The path of hadoop executable
hibench.hadoop.executable    /usr/hdp/current/
```

```
                                hadoop-client/bin/hadoop

# Hadoop configuration directory
hibench.hadoop.configure.dir    /usr/hdp/current/
                                    hadoop-client/etc/hadoop

    hibench.hadoop.examples.test.jar /usr/lib/
                                hadoop-mapreduce/
        hadoop-mapreduce-client-jobclient-2.9.2-tests.jar

# The root HDFS path to store HiBench data
# hibench.hdfs.master value can be found in
# core-site.xml file
hibench.hdfs.master       hdfs://cluster-92af-m

# Hadoop release provider.
# Supported value: apache, cdh5, hdp
hibench.hadoop.release    hdp
```

HiBench related configuration settings like data scale and mappers/reducers count are made within *Hibench/conf/hibench.conf* file. For each data scale, before we run the benchmarks, *hibench.scale.profile* value has to be updated for the respective data scale:

```
$ sudo nano hibench.conf
    # Data scale profile. Available value is tiny, small,
    # large, huge, gigantic and bigdata.
    # The definition of these profiles can be found in
    # the workload's conf file
    # i.e. conf/workloads/micro/wordcount.conf
    hibench.scale.profile                tiny
    # Mapper number in hadoop, partition number in Spark
    hibench.default.map.parallelism         12

    # Reducer nubmer in hadoop, shuffle partition number
    # in Spark
    hibench.default.shuffle.parallelism     12
```

Figure 67 taken during benchmark execution depicts the process on master node and worker nodes.

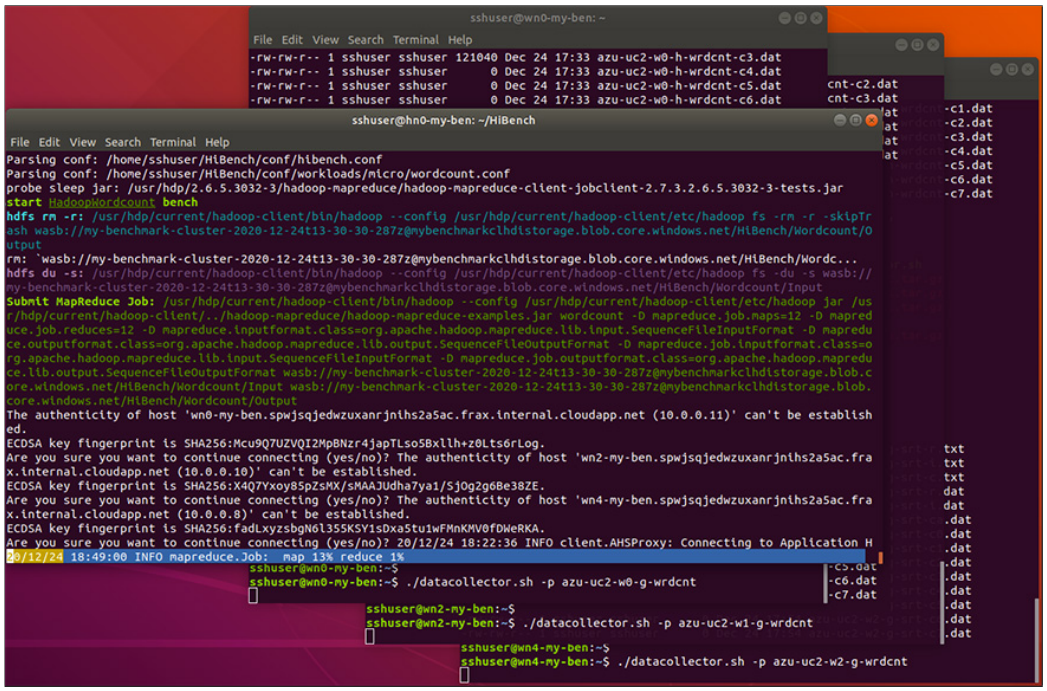

Figure 67: Azure HDInsight - HiBench execution process configurations

6 Running HiBench on Alibaba Cloud e-MapReduce

Aliyun Linux has sysstat package preinstalled, so we don't need to install it.

```
# sar -V
    sysstat version 10.1.5
    (C) Sebastien Godard (sysstat <at> orange.fr)
```

On each worker node, create a directory for storing resouce utilization outputs:

```
# mkdir data
```

On each worker node creata datacollector.sh file.

```
# touch datacollector.sh
```

Upload datacollector script to each worker node. Make datacollector.sh executable

```
# sudo chmod +x datacollector.sh
```

HiBench and related processes are executed on the master node. Apache Maven is required to build HiBench. The first step is to install maven. Because Aliyun Linux is CentOS based, yum package manager has to be leveraged:

```
# sudo yum install maven
```

Figure 68 Verify maven installation

```
# mvn -version
```

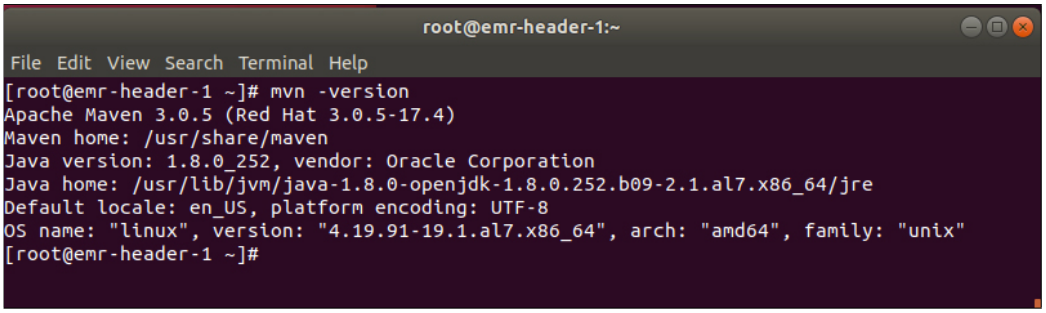

Figure 68: Alibaba Cloud e-MapReduce - Maven version

HiBench works with python 2, we check if python2 is installed

```
# python2 -V
    Python 2.7.5
```

Download HiBench 7.1.1

```
# wget https://github.com/Intel-bigdata/HiBench/archive/
HiBench-7.1.tar.gz
```

Untar the downloaded file.

```
# tar -zxf HiBench-7.1.tar.gz
```

Rename the extracted folder to a more user friendly name

```
# mv HiBench-HiBench-7.1 HiBench
```

Navigate to HiBench folder

```
# cd HiBench
```

Build HiBench 7.1 for Hadoop

```
# mvn -Phadoopbench -Dspark=2.4 -Dscala=2.12 clean package
```

Figure 69 Upon successful HiBench compilation, an informative success message occurs. During the compilation failures might occur, re-running the above command would mostly fix this issue.

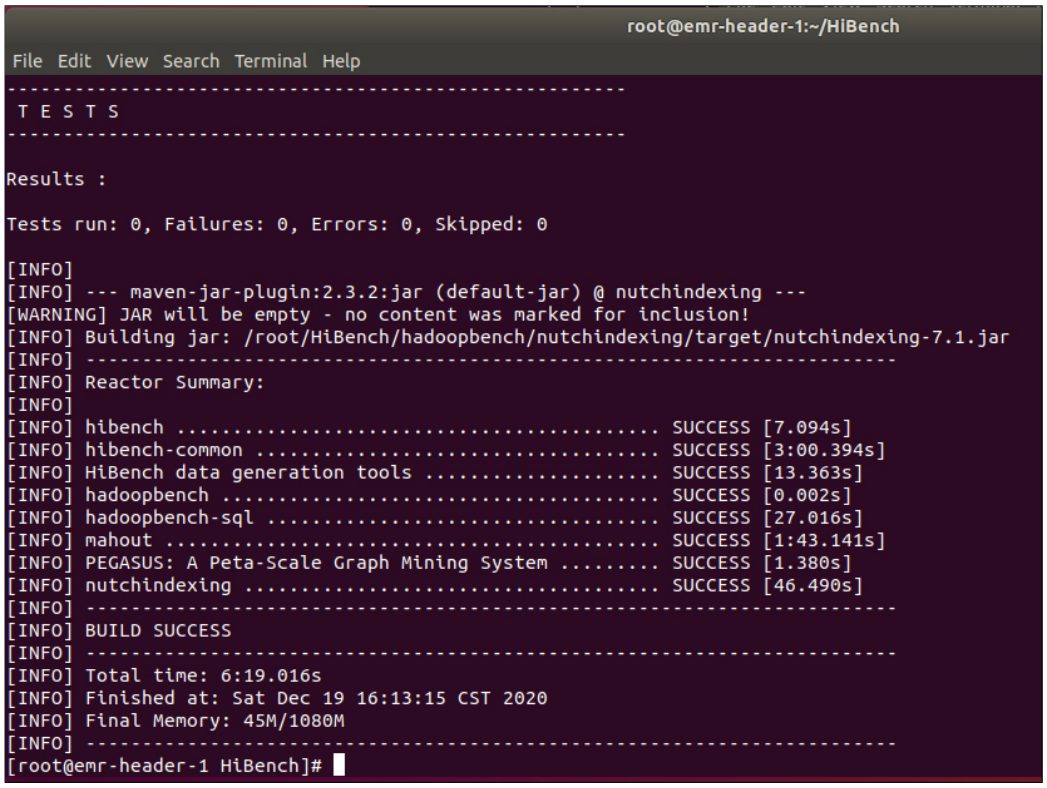

Figure 69: Alibaba Cloud e-MapReduce - HiBench success

To modify HiBench configuration files, navigate to HiBench's conf folder

```
# cd conf/
# cp hadoop.conf.template hadoop.conf
# sudo vi hadoop.conf
```

hibench.hdfs.master value can be found in /usr/lib/hadoop/etc/hadoop/core-site.xml file (fs.default.name). Here we need to specify the hibench.hadoop.examples.test.jar manually for HiBench to run. Otherwise HiBench raises an Assertion error.

Figure 70 Editing the hibench configuration file (using vi editor).

```
# sudo vi hadoop.conf
```

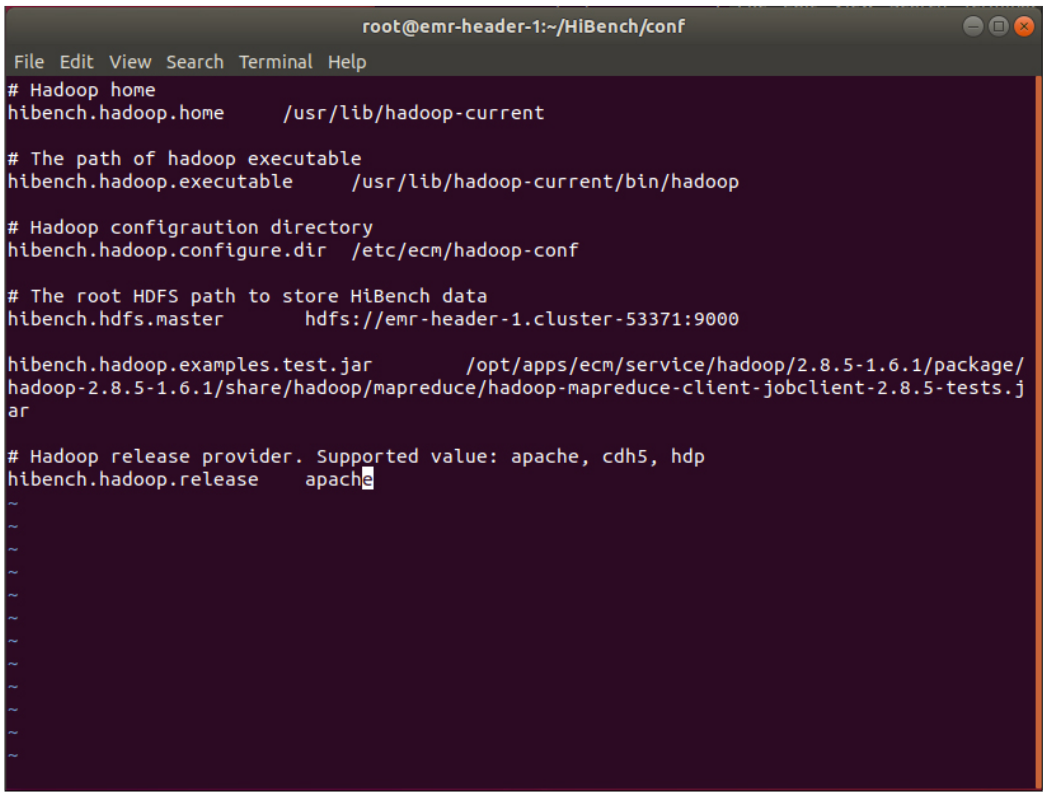

Figure 70: Alibaba Cloud e-MapReduce - Hadoop configurations

Figure 71 HiBench related configuration settings like data scale and mappers/reducers count are made from the file *Hibench/conf/hibench.conf*. Having 12 cores on worker nodes, we specify 12 mappers and 12 reducers used during benchmark. As the data scale change the changes will be made from this file.

```
$ sudo vi hibench.conf
```

Figure 72 depicts the execution of a HiBench benchmark (Wordcount) and resource utilization capturing process on the worker nodes.

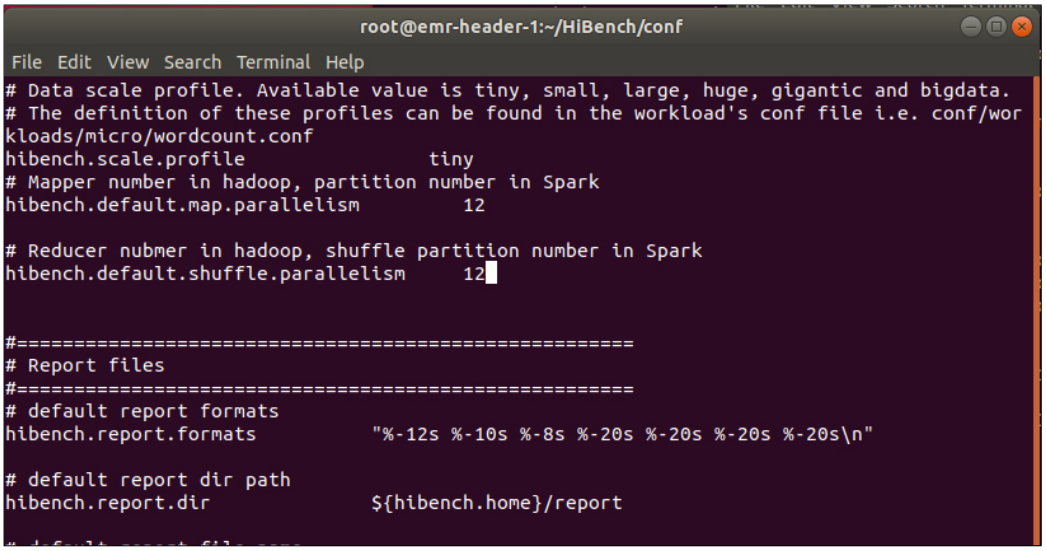

Figure 71: Alibaba Cloud e-MapReduce - HiBench configurations

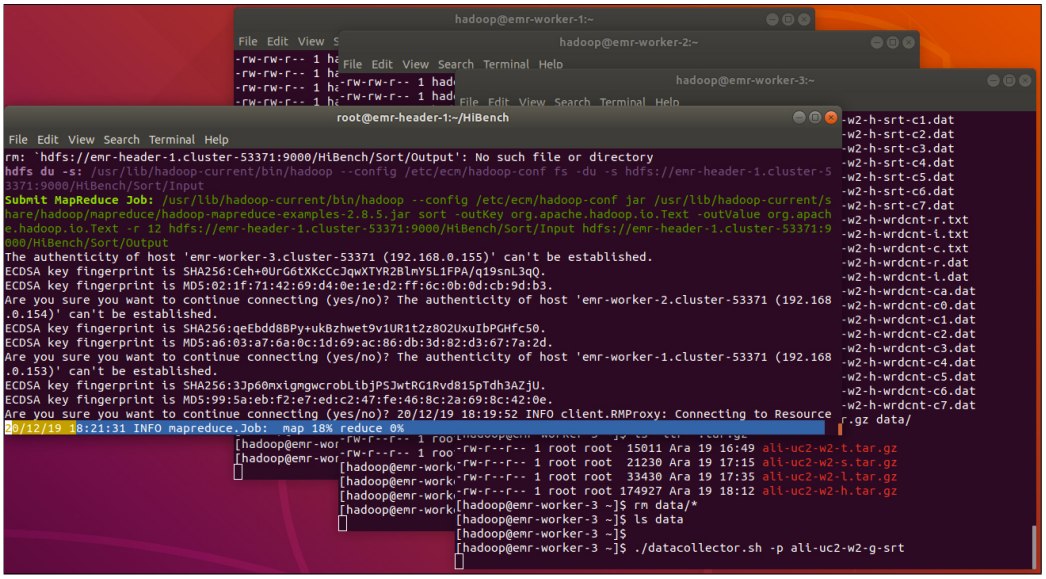

Figure 72: Alibaba Cloud e-MapReduce - HiBench running

List of Figures