



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Trabajo Práctico 1

INF356 - 2025-1 - 200

Computación Distribuida para Big Data

22 de abril de 2025 - v1.0

Índice

1. Despliegue del cluster	5
1.1. Implementación del tutorial	5
1.2. Expansión del cluster	15
1.3. Cliente web	15
2. Instalación de Apache Hive	18
2.1. Procedimiento	18
2.2. Prueba	18
3. Exploración del HDFS	19
4. Uso del cluster	20
4.1. Importación	20
4.2. Parsing	20
4.3. Análisis	20

Índice de figuras

1.	Captura de pantalla de la “AWS Management Console - EC2” que muestra las máquinas del cluster creado con el tutorial	14
2.	Captura de pantalla de la “AWS Management Console - EC2” que muestra las máquinas del cluster expandido	16
3.	Captura de pantalla del cliente web de Hadoop	16
4.	Captura de pantalla del cliente web de HDFS	17

Índice de fragmentos de código

1.	Conectarse por primera vez al Master usando la llave PEM con la IP publica	5
2.	Actualizar repositorios locales y descargar actualizaciones	5
3.	Descargar Java / OpenJDK 11 usando APT	6
4.	Crear una llave publica en formato RSA y autorizarla	6
5.	Descargar y descomprimir Hadoop en el Master	6
6.	Editar las Variables de Entorno usando el usuario Root y el editor de texto Vim	6
7.	Variables de Entorno que usaremos para el cluster	6
8.	hadoop-3.3.6/etc/hadoop/core-site.xml	7
9.	hadoop-3.3.6/etc/hadoop/hdfs-site.xml	7
10.	hadoop-3.3.6/etc/hadoop/mapred-site.xml	8
11.	hadoop-3.3.6/etc/hadoop/yarn-site.xml	9
12.	IP's privadas de los Workers en la carpeta Home del Master	10
13.	Borrar el archivo hadoop-3.3.6.tar.gz y comprimimos el directorio hadoop-3.3.6	10
14.	Script para configurar todos los Workers desde el Master	11
15.	Script para configurar todos los Workers desde el Master	12
16.	Formateamos el File System, lo iniciamos y llamamos al negociador de recursos YARN	13
17.	Corroborar estado de nuestro HDFS	13
18.	Estado del HDFS luego de haber ejecutado un Map Reduce	13
19.	Ejecutar el Map Reduce y leer su salida	14
20.	Los archivos que hay que modificar antes de ejecutar el script de instalacion nuevamente	15
21.	Código utilizado en la expansión del cluster	15
22.	Ejemplo de uso de Apache Hive	18
23.	Resultado esperado para ejemplo de uso de Apache Hive Algunos de los valores han sido alterados, deben ser corregidos con el resultado de su procedimiento.	19
24.	Ejemplo de uso de Apache Hive	19
25.	Script de reporte de DFS	19

1. Despliegue del cluster

1.1. Implementación del tutorial

Para la creación del Cluster en Amazon Web Services, fue necesario seguir los pasos indicados en el tutorial entregado en el Slide T01 - Hadoop Cluster - v1.0 entregado en Aula. El instructivo constaba de los siguientes pasos:

- Crear la cuenta de AWS
- Preparar el entorno del Cluster
- Configurar la máquina Master
- Configurar todos los Workers
- Inicializar el Cluster
- Ejecutar el script de Map Reduce con Hadoop

Crear la cuenta de AWS

Para la creación de la cuenta de AWS simplemente seguí las indicaciones en el correo de invitación indicado por el profesor. Posteriormente a eso, me fui directamente a la sección de Modules, en donde estaba el Módulo de Launch AWS Academy Learner Lab, que sirve para entrar a la Console Home de AWS.

Preparar el entorno del Cluster

Una vez en Console Home, creamos una instancia de tipo EC2, para la cual creamos los Key Pairs, el Security Group y la instancia Master.

Para los Key Pairs, creamos una llave de tipo RSA con formato de archivo PEM.

Para el Security Group, creamos tres reglas, una Inbound Rule de tipo SSH con tipo de fuente Anywhere-IPv4 para entrar por SSH, una Inbound Rule de tipo Custom TCP con rango de puerto 8088 y tipo de fuente Anywhere-IPv4 para la consola web de Hadoop, una Inbound Rule de tipo Custom TCP con rango de puerto 9870 y tipo de fuente Anywhere-IPv4 para la consola web de HDFS y finalmente, una Inbound Rule de tipo Custom TCP con rango de puerto de 0 a 65535 y tipo de fuente Anywhere-IPv4.

Para la instancia Master creamos un nodo con Ubuntu 24.04 de tipo t2.micro y con 16gb de almacenamiento. Esta instancia usará la Key y Security Group recién creados (al igual que las instancias Worker más adelante). Una vez configurados los parámetros, lanzamos la instancia y nos conectamos por SSH:

```
ssh ubuntu@34.207.118.42 -i .\Documents\llave.pem
```

Código 1: Conectarse por primera vez al Master usando la llave PEM con la IP pública

Configurar la máquina Master

El primer paso es lógicamente actualizar los paquetes del sistema, para esto es necesario ejecutar el siguiente comando:

```
sudo apt update && sudo apt upgrade -y
```

Código 2: Actualizar repositorios locales y descargar actualizaciones

Una vez hecho esto, instalamos el paquete correspondiente a Java / OpenJDK 11:

```
1 sudo apt-get install openjdk-11-jdk -y
```

Código 3: Descargar Java / OpenJDK 11 usando APT

Una vez que hayamos resuelto todos los paquetes necesarios, tenemos que crear una llave publica con el siguiente comando:

```
1 ssh-keygen -t rsa
2 cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Código 4: Crear una llave publica en formato RSA y autorizarla

Esta llave publica la guardaremos en la carpeta de Authorized Keys para SSH, lo cual permitira al Master conectarse a si mismo por LocalHost.

Descargamos Hadoop 3.3.6 desde la pagina oficial usando el comando WGET, el cual hace una request a la pagina para descargar el archivo comprimido en formato TAR GZ, el cual tendremos que descomprimir y modificar para indicarle parametros especiales. Ejecutamos entonces los siguientes comandos:

```
1 wget https://www.apache.org/dyn/closer.cgi/hadoop/common/hadoop-3.3.6/hadoop-3.3.6-src.tar.gz
2 tar -xvz hadoop-3.3.6.tar.gz
```

Código 5: Descargar y descomprimir Hadoop en el Master

Esto creara una carpeta llamada hadoop-3.3.6 en el directorio Home del Master, esta carpeta albergara los binarios yu scripts ejecutables para desplegar nuestro Distributed File System (DFS). Para poder ejecutar los comandos en dicha carpeta de manera directa, modificaremos las Variables de Entorno de la maquina Master, de esta manera podremos ejecutar los comandos en cualquier directorio. Dichas Variables tambien contienen rutas para la ejecucion de OpenJDK y la ubicacion de nuestra carpeta Hadoop, las cuales debemos obedecer para que funcione correctamente nuestro cluster. Las variables de entorno en Linux se guardan en el directorio /etc/environment y solo puede modificarse por el Super Usuario (Root). Personalmente, a mi me gusta usar Vim, pero para aquellas personas con gustos inferiores existen alternativas:

```
1 sudo vim /etc/environment
```

Código 6: Editar las Variables de Entorno usando el usuario Root y el editor de texto Vim

Las Variables debiesen quedar asi:

```
1 PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:..."
2 JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
3 HADOOP_HOME="/home/ubuntu/hadoop-3.3.6"
4 HADOOP_COMMON_HOME="/home/ubuntu/hadoop-3.3.6"
```

Código 7: Variables de Entorno que usaremos para el cluster

Lo ultimo es modificar los archivos de configuracion de Hadoop en base a lo indicado por el profesor. Estos archivos luego seran enviados a los Workers cuando comprimamos la carpeta hadoop-3.3.6.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4   <property>
5     <name>fs.defaultFS</name>
6     <value>hdfs://172.31.30.146</value>
7   </property>
8 </configuration>
```

Código 8: hadoop-3.3.6/etc/hadoop/core-site.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4   <property>
5     <name>dfs.block.size</name>
6     <value>33554432</value>
7   </property>
8   <property>
9     <name>dfs.replication</name>
10    <value>3</value>
11  </property>
12  <property>
13    <name>dfs.name.dir</name>
14    <value>file:///home/ubuntu/dfs/name</value>
15  </property>
16  <property>
17    <name>dfs.data.dir</name>
18    <value>file:///home/ubuntu/dfs/data</value>
19  </property>
20  <property>
21    <name>dfs.namenode.heartbeat.recheck-interval</name>
22    <value>10000</value>
23  </property>
24 </configuration>
```

Código 9: hadoop-3.3.6/etc/hadoop/hdfs-site.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4   <property>
5     <name>mapreduce.framework.name</name>
6     <value>yarn</value>
7   </property>
8   <property>
9     <name>yarn.app.mapreduce.am.env</name>
10    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
11  </property>
12  <property>
13    <name>mapreduce.map.env</name>
14    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
15  </property>
16  <property>
17    <name>mapreduce.map.java.opts</name>
18    <value>-Xmx1024m</value>
19  </property>
20  <property>
21    <name>mapreduce.map.memory.mb</name>
22    <value>1024</value>
23  </property>
24  <property>
25    <name>mapreduce.reduce.env</name>
26    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
27  </property>
28  <property>
29    <name>mapreduce.reduce.java.opts</name>
30    <value>-Xmx1024m</value>
31  </property>
32  <property>
33    <name>mapreduce.reduce.memory.mb</name>
34    <value>1024</value>
35  </property>
36  <property>
37    <name>mapreduce.job.maps</name>
38    <value>4</value>
39  </property>
40  <property>
41    <name>mapreduce.job.running.map.limit</name>
42    <value>4</value>
43  </property>
44  <property>
45    <name>mapreduce.job.reduces</name>
46    <value>4</value>
47  </property>
48  <property>
49    <name>mapreduce.job.running.reduce.limit</name>
50    <value>4</value>
51  </property>
52 </configuration>

```

Código 10: hadoop-3.3.6/etc/hadoop/mapred-site.xml


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3 <configuration>
4   <property>
5     <name>yarn.resourcemanager.hostname</name>
6     <value>172.31.30.146</value>
7   </property>
8   <property>
9     <name>yarn.nodemanager.aux-services</name>
10    <value>mapreduce_shuffle</value>
11  </property>
12  <property>
13    <name>yarn.nodemanager.resource.memory-mb</name>
14    <value>1024</value>
15  </property>
16  <property>
17    <name>yarn.nodemanager.resource.cpu-vcores</name>
18    <value>1</value>
19  </property>
20  <property>
21    <name>yarn.app.mapreduce.am.resource.mb</name>
22    <value>1024</value>
23  </property>
24  <property>
25    <name>yarn.app.mapreduce.am.command-opts</name>
26    <value>-Xmx1024m</value>
27  </property>
28  <property>
29    <name>yarn.app.mapreduce.am.resource.cpu-vcores</name>
30    <value>1</value>
31  </property>
32 </configuration>
```

Código 11: hadoop-3.3.6/etc/hadoop/yarn-site.xml

Configurar todos los Workers

Los Workers también serán máquinas con Ubuntu 24.04 y de tipo t2.micro, usarán las mismas Key Pairs y el mismo Security Group. Lo único que cambiará entre ellos y el Master será el almacenamiento, las cuales constarán solo de 8gb en total. Una vez creados los Workers desde el interfaz de EC2 de AWS, tendremos que obtener sus IP's públicas y privadas. La primera para conectarnos por primera vez, la segunda para establecer una conexión permanente. Anotamos las IP's privadas de la siguiente manera:

```
1 172.31.30.0
2 172.31.17.157
3 172.31.24.51
4 172.31.27.79
```

Código 12: IP's privadas de los Workers en la carpeta Home del Master

Luego de tener esto, borramos el archivo comprimido de Hadoop que descargamos inicialmente y comprimimos el directorio que modificamos en el paso anterior para luego enviárselo a los Workers. Estos luego descomprimen el archivo de manera local para así no abusar de nuestro ancho de banda limitado:

```
1 tar czf hadoop-3.3.6.tar.gz hadoop-3.3.6
```

Código 13: Borrar el archivo hadoop-3.3.6.tar.gz y comprimimos el directorio hadoop-3.3.6

Y como la tarea de configurar cada Worker manualmente es tedioso, usamos un script de Bash que realiza el trabajo por nosotros. El cual itera los siguientes pasos:

Para cada una de las instancias Worker proceder de la siguiente forma:

1. Instalar la llave pública de la máquina Master en el Worker
2. Crear una llave para el Worker
3. Instalar la llave del Worker en el Master
4. Instalar la llave del Worker en el Worker
5. Actualizar el sistema operativo en el Worker
6. Instalar Java en el Worker (openjdk-11-jdk)
7. Editar el archivo /etc/environment de la misma forma que para Master
8. Reiniciar el worker
9. Transferir el archivo de despliegue de Hadoop de la máquina Master al Worker
10. Expandir el archivo de despliegue en la máquina Worker

El script utilizado quedó de la siguiente fórmula:

```

1 #!/bin/bash
2
3 # === CONFIGURACIÓN ===
4
5 # IPs publicas de los Workers y el Master
6 WORKERS=("172.31.30.0" "172.31.17.157" "172.31.24.51" "172.31.27.79")
7 MASTER="172.31.30.146"
8
9 # User de el Master y los Workers
10 USER="ubuntu"
11 HOME="/home/$USER"
12
13 # Ubicacion de la llave privada
14 KEY="$HOME/.ssh/llave.pem"
15
16 # Variables de Entorno
17 PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games..."
18 JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
19 HADOOP_HOME="/home/ubuntu/hadoop-3.3.6"
20 HADOOP_COMMON_HOME="/home/ubuntu/hadoop-3.3.6"
21
22 # Path del archivo comprimido de Hadoop
23 HADOOP_ARCHIVE="/home/ubuntu/hadoop-3.3.6.tar.gz"
24
25 # Ubicacion de las Variables de Entorno
26 ENV_VARS="/etc/environment"
27
28 for WORKER in "${WORKERS[@]}; do
29     echo ">>> Configurando al Worker de IP $WORKER <<<"
30
31     echo ""
32     echo "1) Instalando llave publica del Master en el Worker"
33     MASTER_KEY="$(cat ~/.ssh/id_rsa.pub)"
34     echo "└─> ssh $USER@$WORKER -i $KEY echo $MASTER_KEY >> ~/.ssh/authorized_keys"
35     ssh $USER@$WORKER -i $KEY "echo $MASTER_KEY >> ~/.ssh/authorized_keys"
36
37     echo ""
38     echo "2) Creando llave publica para el Worker"
39     echo "└─> ssh $USER@$WORKER -i $KEY ssh-keygen -t rsa"
40     ssh $USER@$WORKER -i $KEY "ssh-keygen -t rsa"
41     echo ""
42
43     echo ""
44     echo "3) Instalando llave del Worker en el Master"
45     echo "└─> ssh $USER@$WORKER -i $KEY cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys"
46     ssh $USER@$WORKER -i $KEY "cat ~/.ssh/id_rsa.pub" >> ~/.ssh/authorized_keys
47     echo ""
48
49     echo ""
50     echo "4) Instalando llave del Worker en el mismo Worker"
51     echo "└─> ssh $USER@$WORKER cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys"
52     ssh $USER@$WORKER "cat ~/.ssh/id_rsa.pub" >> ~/.ssh/authorized_keys
53     echo ""
54
55     echo ""
56     echo "5) Actualizando sistema operativo en el Worker"
57     echo "└─> ssh $USER@$WORKER sudo apt update -qq && sudo apt upgrade -y -qq"
58     ssh $USER@$WORKER "sudo apt update -qq && sudo apt upgrade -y -qq"
59     echo ""
60 ...

```

Código 14: Script para configurar todos los Workers desde el Master

```

1  ...
2  echo ""
3  echo "6) Instalando Java en el Worker"
4  echo "└─> ssh $USER@$WORKER sudo apt-get install -y -qq openjdk-11-jdk"
5  ssh $USER@$WORKER "sudo apt-get install -y -qq openjdk-11-jdk"
6  echo ""
7
8  echo ""
9  # Esta seccion me dio problemas asi que este paso lo hice manual al final
10 echo "7) Copiando las Variables de Entorno en el Worker"
11     echo "└─> cat $HOME/environment"
12     cat $HOME/environment
13 echo "└─> scp $HOME/environment $USER@$WORKER:$HOME/"
14 scp $HOME/environment $USER@$WORKER:$HOME/
15 echo "ssh $USER@$WORKER cat $HOME/ambiente"
16 ssh $USER@$WORKER "cat $HOME/ambiente"
17 echo "└─> ssh $USER@$WORKER sudo cp $HOME/ambiente $ENV_VARS"
18 ssh $USER@$WORKER "sudo cp $HOME/ambiente $ENV_VARS"
19 echo "└─> ssh $USER@$WORKER cat $ENV_VARS"
20 ssh $USER@$WORKER "cat $ENV_VARS"
21
22 echo ""
23
24 echo ""
25 echo "8) Copiando archivo de despliegue Hadoop al Worker..."
26 echo "ssh $USER@$WORKER rm $HADOOP_ARCHIVE && ls -la"
27 ssh $USER@$WORKER "rm $HADOOP_ARCHIVE && ls -la"
28 echo "└─> scp $HADOOP_ARCHIVE $USER@$WORKER:$HOME/"
29 scp $HADOOP_ARCHIVE $USER@$WORKER:$HOME/
30 echo "└─> ssh $USER@$WORKER ls -la"
31 ssh $USER@$WORKER "ls -la"
32 echo ""
33
34 echo ""
35 echo "9) Descomprimiendo el archivo de despliegue en el Worker..."
36 echo "└─> ssh $USER@$WORKER tar -xzf $HADOOP_HOME.tar.gz -C $HOME/"
37 ssh $USER@$WORKER "tar -xzf $HADOOP_HOME.tar.gz -C $HOME/"
38 echo ""
39
40 echo ""
41 echo "10) Reiniciando Worker..."
42 echo "└─> ssh $USER@$WORKER sudo reboot now"
43 ssh $USER@$WORKER "sudo reboot now"
44 echo ""
45
46 echo "El Worker de IP $WORKER ha sido configurado correctamente! :)"
47 echo ""
48 done
49
50 echo "Todos los Workers han sido configurados! :D"

```

Código 15: Script para configurar todos los Workers desde el Master

Una vez configurado todo, reiniciamos el nodo Master y reingresamos cuando la maquina vuelva a estar activa.

Inicializar el Cluster

Si las maquinas estan correctamente configuradas, podremos levantar el File System distribuido sin problemas. Para esto es necesario formatear el disco e iniciar los servicios para la distribucion de este y para la gestion de recursos entre los Workers. Cabe notar que por disco me refiero a unidad de almacenamiento, como vimos en clases, un DFS no almacena datos de forma continua, si no que la reparte entre varios Nodos.

Dicho esto, ejecutamos la siguiente serie de comandos:

```
1 hdfs namenode -format
2 start-dfs.sh
3 start-yarn.sh
```

Código 16: Formateamos el File System, lo iniciamos y llamamos al negociador de recursos YARN

Luego, para corroborar de que todo esta en orden, ejecutamos el siguiente comando:

```
1 hdfs fsck / -files -blocks
```

Código 17: Corroborar estado de nuestro HDFS

La salida debiese ser algo del siguiente estilo:

```
1 ...
2 Status: HEALTHY
3 Number of data-nodes: 4
4 Number of racks: 1
5 Total dirs: 11
6 Total symlinks: 0
7
8 Replicated Blocks:
9 Total size: 428156 B
10 Total files: 9
11 Total blocks (validated): 8 (avg. block size 53519 B)
12 Minimally replicated blocks: 8 (100.0 %)
13 Over-replicated blocks: 0 (0.0 %)
14 Under-replicated blocks: 0 (0.0 %)
15 Mis-replicated blocks: 0 (0.0 %)
16 Default replication factor: 3
17 Average block replication: 3.0
18 Missing blocks: 0
19 Corrupt blocks: 0
20 Missing replicas: 0 (0.0 %)
21 Blocks queued for replication: 0
22 ...
23 The filesystem under path '/' is HEALTHY
```

Código 18: Estado del HDFS luego de haber ejecutado un Map Reduce

Notaremos que existe una seccion que muestra la cantidad de Data Nodes, si muestra 4 entonces configuramos correctamente nuestro cluster.

Instances (1/5) [Info](#)

Last updated less than a minute ago [Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Find Instance by attribute or tag (case-sensitive) [All states](#) [Previous](#) **1** [Next](#) [Settings](#)

<input type="checkbox"/>	Name ✎	Instance ID	Instance state ▼	Instance type ▼	Status check	Alarm
<input checked="" type="checkbox"/>	worker-1	i-02726e3b5b124366c	Running 🔍 🔍	t2.micro	Initializing 🕒	View
<input type="checkbox"/>	worker-2	i-0e8beeb4bab32bb293	Running 🔍 🔍	t2.micro	Initializing 🕒	View
<input type="checkbox"/>	worker-3	i-0ad704152037a39af	Running 🔍 🔍	t2.micro	Initializing 🕒	View
<input type="checkbox"/>	worker-4	i-0840fbfdcd68e3f126	Running 🔍 🔍	t2.micro	Initializing 🕒	View
<input type="checkbox"/>	master	i-0a241458b2a86b298	Running 🔍 🔍	t2.micro	2/2 checks passed 🕒	View

Figura 1: Captura de pantalla de la "AWS Management Console - EC2" que muestra las máquinas del cluster creado con el tutorial

Ejecutar el script de Map Reduce con Hadoop

Finalmente, ejecutamos nuestro algoritmo de Map Reduce. Para esto basta con crear un directorio /data, almacenar nuestro archivo a leer y finalmente ejecutar el ejecutable JAR contenido en nuestra carpeta de Hadoop. Una vez hecho esto, podemos analizar la salida del Map Reduce haciéndole un Cat y pasándolo por Pipe a More para desplegar la información por pantalla:

```
1 hdfs dfs -mkdir /data
2 hdfs dfs -put sw-script-e04.txt /data/
3 hadoop jar hadoop-3.3.6/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar \
4 wordcount /data/sw-script-e04.txt /data/sw-script-e04.wordcount
5 hadoop fs -cat /data/sw-script-e04.wordcount/part-r-00000 | more
```

Código 19: Ejecutar el Map Reduce y leer su salida

1.2. Expansión del cluster

Para expandir el cluster a 8 Workers, lo primero seria replicar las instancias desde AWS, estas debiesen tener los mismos atributos que los workers originales. Lo segundo seria actualizar la configuracion desde el Master para incluir las nuevas IP's y habilitar el uso de 4 Nodos desde los XML de Hadoop. Lo tercero seria configurar las conexiones de SSH hacia los workers nuevos, para esto podemos reutilizar el script de Bash pero solo con las IP's privadas nuevas.

```

1 # hadoop-3.3.6/etc/hadoop/mapred-site.xml
2 <property>
3   <name>mapreduce.job.running.map.limit</name>
4   <value>8</value> # Modificar de 4 a 8
5 </property>
6 # ...
7 <property>
8   <name>mapreduce.job.reduces</name>
9   <value>8</value> # Modificar de 4 a 8
10 </property>
11 # ...
12 <property>
13   <name>mapreduce.job.running.reduce.limit</name>
14   <value>4</value> # Modificar de 4 a 8
15 </property>
16
17 # hadoop-3.3.6/etc/hadoop/workers
18 172.31.30.0
19 172.31.17.157
20 172.31.24.51
21 172.31.27.79
22 172.31.84.79
23 172.31.93.162
24 172.31.92.28
25 172.31.81.101
26
27 # /home/ubuntu/install.sh
28 # WORKERS=("172.31.30.0" "172.31.17.157" "172.31.24.51" "172.31.27.79")
29 WORKERS=("172.31.30.0" "172.31.17.157" "172.31.24.51" "172.31.27.79" \
30 "172.31.84.79" "172.31.93.162" "172.31.92.28" "172.31.81.101")
31 MASTER="172.31.30.146"

```

Código 20: Los archivos que hay que modificar antes de ejecutar el script de instalacion nuevamente

```

1 for linea in $(cat archivo)
2 do
3   echo $linea
4 done

```

Código 21: Código utilizado en la expansión del cluster

1.3. Cliente web

Aqui podemos apreciar todas las instancias desde la interfaz de EC2 de AWS, las instancias expandidas tienen el nombre de Worker y un numero que va desde 5 hasta 8. Sus direcciones IP privadas aparecen en el apartado de expansion y tienen las mismas características que los otros Nodos.

La figura 3 muestra la consola web de Hadoop y la figura 4 la consola web del HDFS.

Instances (1/9) [Info](#)

Find Instance by attribute or tag (case-sensitive) [All states](#)

Last updated 4 minutes ago [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security group name
worker-5	i-02f1765577eebf37	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-54-157-252-105.co...	54.157.252.105	-	-	disabled	security_group
worker-6	i-03f1af215b3a3a16	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-53-95-95-120.comp...	53.95.95.120	-	-	disabled	security_group
worker-7	i-0e001149f9255a9f	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-13-215-100-100.co...	13.215.100.100	-	-	disabled	security_group
worker-8	i-0796c3f946116202	Running	t2.micro	Initializing	View alarms +	us-east-1c	ec2-3-52-145-145.com...	3.52.145.145	-	-	disabled	security_group
worker-1	i-02720e3b3b124366c	Running	t2.micro	2/2 checks pass	View alarms +	us-east-1d	ec2-54-159-51-45.com...	54.159.51.45	-	-	disabled	security_group
worker-2	i-0e85beb4bab32bb293	Running	t2.micro	2/2 checks pass	View alarms +	us-east-1d	ec2-34-229-165-77.co...	34.229.165.77	-	-	disabled	security_group
worker-3	i-0ad704152037a39af	Running	t2.micro	2/2 checks pass	View alarms +	us-east-1d	ec2-54-147-155-78.co...	54.147.155.78	-	-	disabled	security_group
worker-4	i-0640bfad08e3f1126	Running	t2.micro	2/2 checks pass	View alarms +	us-east-1d	ec2-3-53-231-55.comp...	3.53.231.55	-	-	disabled	security_group
master	i-0a241455b2a55b295	Running	t2.micro	2/2 checks pass	View alarms +	us-east-1d	ec2-34-229-144-172.co...	34.229.144.172	-	-	disabled	security_group

i-0796c3f946116202 (worker-8)

[Details](#) [Status and alarms](#) [Monitoring](#) [Security](#) [Networking](#) [Storage](#) [Tags](#)

Instance summary [Info](#)

Instance ID
i-0796c3f946116202

IPv6 address
-

Hostname type
IP name: ip-172-31-01-101.ec2.internal

Answer private resource DNS name
IPv4 (A)

Auto-assigned IP address
3.52.145.145 (Public IP)

IAM Role
-

IMDSv2

Public IPv4 address
3.52.145.145 | [open address](#)

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-01-101.ec2.internal

Instance type
t2.micro

VPC ID
vpc-b4f91fa9355c9917

Subnet ID
subnet-0ca44fc25513730b0

Instance ARN

Private IPv4 addresses
172.31.01.101

Public IPv4 DNS
ec2-3-52-145-145.compute-1.amazonaws.com | [open address](#)

Elastic IP addresses
-


AWS Compute Optimizer finding
[Opt-in to AWS Compute Optimizer for recommendations.](#) | [Learn more](#)

Auto Scaling Group name
-

Managed

Activate Windows
Go to Settings to activate Windows.

Figura 2: Captura de pantalla de la "AWS Management Console - EC2" que muestra las máquinas del cluster expandido

 **Nodes of the cluster** Logged in as: dr.who

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources	Physical Mem Used %	Physical VCores Used %
0	0	0	0	0	<memory:0 B, vCores:0>	<memory:4 GB, vCores:4>	<memory:0 B, vCores:0>	72	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
4	0	0	0	0	0	0

Scheduler Metrics

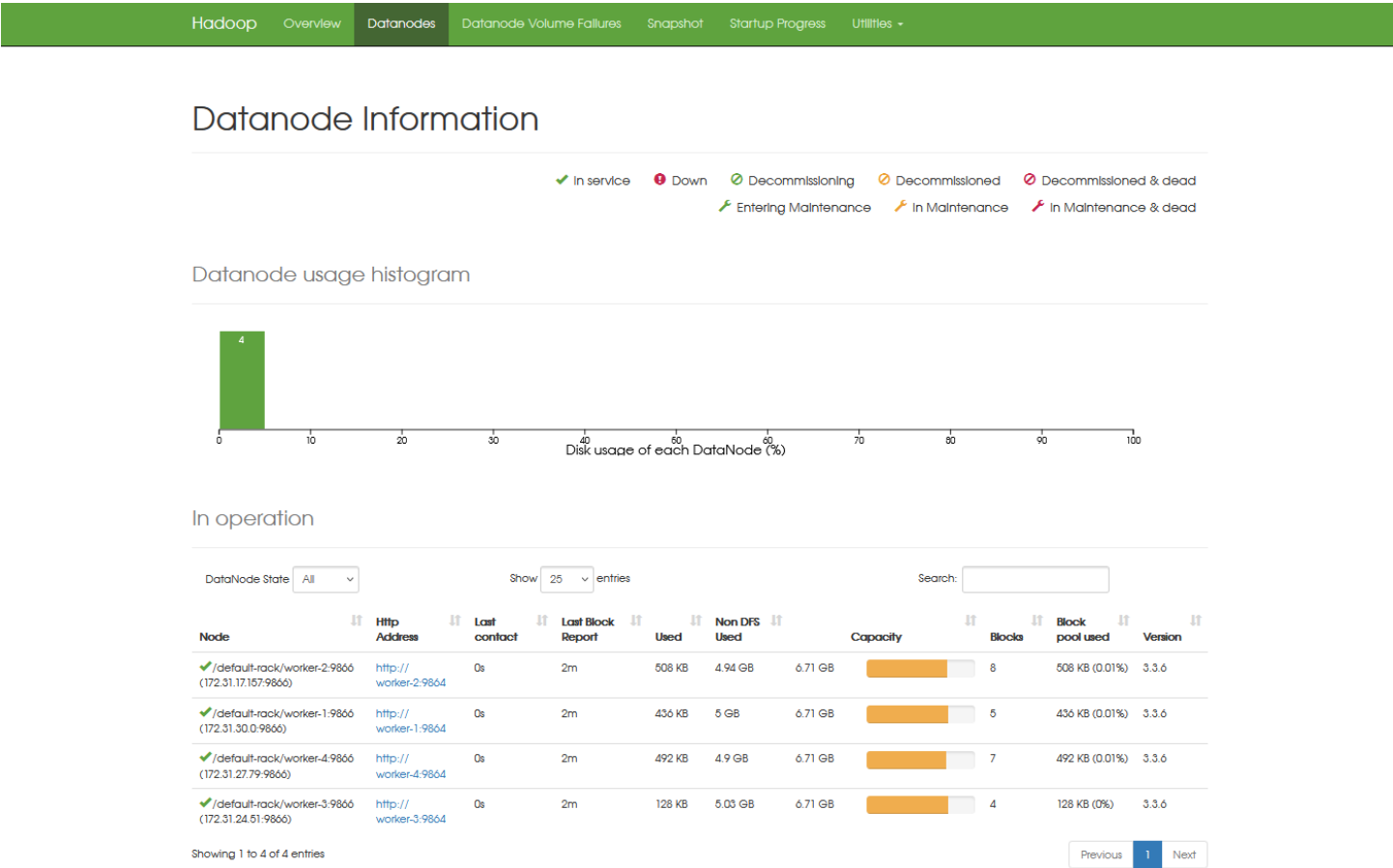
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority	Scheduler Busy %
Capacity Scheduler	(memory-mb (unit-MB), vcores)	<memory:1024, vCores:1>	<memory:1024, vCores:1>	0	0

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %	VCores Used	VCores Avail	Phys VCores Used %	Version
/ default-rack		RUNNING	worker-4:44587	worker-4:8042	Wed Apr 23 02:44:21 +0000 2025	0			0 B	1 GB	69	0	1	0	3.3.6
/ default-rack		RUNNING	worker-3:39227	worker-3:8042	Wed Apr 23 02:44:21 +0000 2025	0			0 B	1 GB	69	0	1	0	3.3.6
/ default-rack		RUNNING	worker-2:41109	worker-2:8042	Wed Apr 23 02:44:21 +0000 2025	0			0 B	1 GB	72	0	1	0	3.3.6
/ default-rack		RUNNING	worker-1:34189	worker-1:8042	Wed Apr 23 02:44:21 +0000 2025	0			0 B	1 GB	77	0	1	0	3.3.6

Showing 1 to 4 of 4 entries First Previous 1 Next Last

Figura 3: Captura de pantalla del cliente web de Hadoop



2. Instalación de Apache Hive

2.1. Procedimiento

Desarrolle un procedimiento para instalar **Apache Hive** en su cluster. Utilice la versión 4.0.1. Tenga en consideración:

- Las máquinas **t2.micro** son muy limitadas para levantar el servicio de Hive. Para esta sección se sugiere subir la máquina maestra a tipo **t3.medium** y los trabajadores a **t3.small**. Para cambiar el tipo de máquina no es necesario volver a desplegarla, basta con detener la máquina, cambiar su tipo desde la consola, y volver a iniciarla.
- Lograr la configuración correcta para Hive es un procedimiento que requiere bastante conocimiento y pruebas. El archivo zip de las instrucciones incluye el archivo **hive-example.xml** con la configuración a utilizar.

En esta sección indique el procedimiento desarrollado y agregue cualquier código que haya sido utilizado para instalar y probar Apache Hive. Incluya el documento de configuración utilizado y explique cada uno de los parámetros definidos en este.

2.2. Prueba

Para probar que la instalación de Hive funciona correctamente, puede utilizar el procedimiento disponible en el fragmento de código 22. Este ejemplo asume que el archivo utilizado para probar el cluster **sw-script-e04.txt** se encuentra disponible en el DFS. El resultado esperado para la prueba indicada se entrega en el fragmento 23

```

1 CREATE DATABASE sw_dialogs;
2
3 USE sw_dialogs;
4
5 CREATE TABLE sw04_dialogs (
6     line      INT,
7     character STRING,
8     dialog    STRING
9 )
10 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
11 WITH SERDEPROPERTIES (
12     "separatorChar" = " ",
13     "quoteChar" = "\""
14 )
15 STORED AS TEXTFILE
16 TBLPROPERTIES ("skip.header.line.count"="1");
17
18 LOAD DATA INPATH
19 '/data/sw-script-e04.txt'
20 INTO TABLE
21 sw04_dialogs;
22
23 SELECT character, COUNT(*) AS lines
24 FROM sw04_dialogs
25 GROUP BY character
26 ORDER BY lines DESC
27 LIMIT 12;

```

Código 22: Ejemplo de uso de Apache Hive

```

1 +-----+-----+
2 | character | lines |
3 +-----+-----+
4 | LUKE      | 524   |
5 | HAN       | 135   |
6 | THREEPIO  | 119   |
7 | BEN       | 82    |
8 | LEIA      | 57    |
9 | VADER     | 41    |
10 | RED LEADER | 73    |
11 | BIGGS     | 34    |
12 | TARKIN    | 28    |
13 | OWEN      | 25    |
14 | TROOPER   | 19    |
15 | WEDGE     | 14    |
16 +-----+-----+
17 12 rows selected (67.678 seconds)

```

Código 23: Resultado esperado para ejemplo de uso de Apache Hive Algunos de los valores han sido alterados, deben ser corregidos con el resultado de su procedimiento.

3. Exploración del HDFS

Desarrolle y explique un script que utilizando bash permita obtener la lista de bloques en las que está guardado un archivo en el DFS, incluyendo las direcciones IP privadas de las máquinas que guardan cada copia del bloque. La salida del script se debe ver como lo indicado en el fragmento 24.

```

1 File:           /mydata/myfile.txt
2 Blocks:         2
3 Avg. Replication: 3.0
4
5 0 - blk_1073742511_1694 - 172.31.40.100 172.31.32.111 172.31.35.247
6 1 - blk_1073742511_1695 - 172.31.32.111 172.31.210.111 172.31.35.247

```

Código 24: Ejemplo de uso de Apache Hive

Muestre el script desarrollado en el Código 25 y explique cada una de las partes del script.

```

1 for linea in $(cat archivo)
2 do
3     echo $linea
4 done

```

Código 25: Script de reporte de DFS

En esta sección indique el procedimiento desarrollado y agregue cualquier código que haya sido utilizado para desarrollar el script. Utilizando la AWS-CLI, descargue en el nodo maestro del cluster el archivo `s3://utfsm-inf356-dataset/vlt_observations_000.csv`¹

¹Este archivo es público y está guardado en un bucket S3, por lo que debe utilizar la opción `-no-sign-request`. Este archivo pesa 371.1[MB] y es un archivo `csv` como el mismo formato al descargar https://archive.eso.org/eso/eso_archive_main.html con todos los campos. Una vez descargado, coloque el archivo en la carpeta `data` del DFS y muestre el resultado del script desarrollado previamente.

4. Uso del cluster

4.1. Importación

Desarrolle un procedimiento que permita importar el archivo **vlt_observations_000.csv** a Hive desde el DFS respetando las columnas del archivo. Indique el código utilizado.

4.2. Parsing

Desarrolle una propuesta para asignar un tipo apropiado a los datos importados y desarrolle un procedimiento que permita dar este formato creando una nueva tabla. Indique el código utilizado.

4.3. Análisis

Piense 3 métricas sobre los datos interpretados, describa cada una de estas métricas y provea el código para obtener el resultado. Las métricas deben tener como mínimo 2 elementos de análisis (agrupamiento, contar, promedio, etc.). Ejemplos de métricas posibles son:

- Cantidad de observaciones por cada tipo de observación (agrupa y cuenta)
- Ángulo promedio de declination de las observaciones del set para cada instrumento (agrupa y promedia)
- Seeing promedio por hora de observación (agrupa y promedia)

Provea un análisis sobre el desempeño del cluster al realizar estas operaciones. Incluya mediciones como tiempo de cómputo, máquinas usadas, cantidad de trabajos, cantidad de mappers y reducers, etc.