

Infraestructura para Datos Masivos



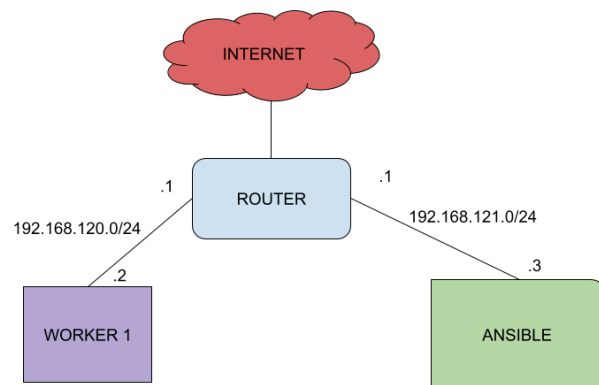
Jaime Rodríguez Pérez
alu0100893861@ull.edu.es

INTRODUCCIÓN

En esta práctica se va a realizar el despliegue de una infraestructura utilizando Ansible y Hadoop. En concreto en esta práctica se trabajará con tres máquinas virtuales desplegadas desde el iaas. Una será el worker, otra el router y la otra el ansible que se encargará de automatizar la gestión de las demás máquinas.

CREACIÓN DE INFRAESTRUCTURA

Como comentamos se van a realizar la configuración de 3 máquinas virtuales en el iaas de la ULL, un router, un worker y un ansible. La estructura quedaría de la siguiente manera:



Para configurar cada una de las máquinas virtuales se le van a especificar las direcciones IPs específicas.

Para configurar el router abrimos la terminal y editamos el archivo de configuración de netplan: `sudo nano /etc/netplan/00-installer-config.yaml`

```
GNU nano 4.8 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
      dhcp-identifier: mac
      dhcp4: false
    ens5:
      addresses:
        - 192.168.120.1/24
    ens7:
      addresses:
        - 192.168.121.1/24
```

Vemos que por cada interfaz añadimos cada una de las direcciones IPS respectivas. Guardamos y cerramos el archivo y aplicamos los cambios con : `sudo netplan apply`

Hacemos esto con cada una de las máquinas virtuales. La configuración del router quedaría de la siguiente manera

```
GNU nano 4.8 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
      dhcp-identifier: mac
      dhcp4: false
      addresses:
        - 192.168.120.2/24
```

Y para el ansible, ya que vamos a instalar ansible en esta máquina se ha tenido que configurar la máquina para que funcione el servidor DNS añadiendo las direcciones IP de los nombres de servidores de la universidad de la Laguna.

```
GNU nano 4.8 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  version: 2
  renderer: networkd
  ethernets:
    #ens3:
    #  dhcp-identifier: mac
    #  dhcp4: true
    ens3:
      addresses:
        - 192.168.121.3/24
      gateway4: 192.168.121.1
      nameservers:
        addresses:
          - 10.4.9.29
          - 10.4.9.30
```

Una vez terminada la configuración de las 3 máquinas y comprobando que se conectan entre ellas realizando un ping desde el router hasta las direcciones ips de las otras dos máquinas, se configura NAT para enmascarar las direcciones de red en la máquina del router.

Usamos el comando

```
- sudo iptables -t nat -A POSTROUTING -o ens3 -J MASQUERADE
```

Este comando va a configurar una regla de enmascaramiento de direcciones de red en IPtables para permitir que los paquetes salientes se enmascaran con la dirección IP de origen del router.

(Si queremos que la configuración NAT no se borre cada vez que reiniciemos la máquina hay que guardar la configuración con el comando `sudo iptables-save`)

```
root@ubuntu:~# sudo iptables-save
# Generated by iptables-save v1.8.4 on Fri Jun 16 16:05:05 2023
*nat
:PREROUTING ACCEPT [18:1127]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -o ens3 -j MASQUERADE
COMMIT
# Completed on Fri Jun 16 16:05:05 2023
```

Por último para activar el envío de paquetes para ipv4 en el router habría que descomentar la línea en el archivo `/etc/sysctl.conf` y descomentar la línea donde pone:

```
net.ipv4.ip_forward = 1
```

Una vez ya tenemos todas las máquinas configuradas procedemos a instalar ansible en la máquina correspondiente.

Para ello en la máquina ansible ejecutamos el comando `sudo apt-get update`, para actualizar los paquetes y luego instalamos ansible con

```
sudo apt install ansible
```

Vamos ahora a crear usuarios destinados a gestionar y tener el control. Para ello añadimos un usuario ansible con el comando

```
sudo useradd -m ansible -s /bin/bash
```

```
root@ubuntu:~# sudo useradd -m ansible -s /bin/bash
root@ubuntu:~# sudo passwd ansible
New password:
Retype new password:
passwd: password updated successfully
root@ubuntu:~#
```

Y le creamos una contraseña.

Queremos realizar una conexión ssh desde el nodo control hasta los nodos usuarios o worker. Para hacerlo sin contraseña se debe crear clave pública.

Generamos un par de claves SSH en la máquina ansible

```
sudo ssh-keygen -t rsa
```

```
usuario@ubuntu:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_rsa):
Created directory '/home/usuario/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/usuario/.ssh/id_rsa
Your public key has been saved in /home/usuario/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:0x3dfpWNjHwAUyAwQ/spU32UbgxNYPkiGv4E+snxXIU usuario@ubuntu
The key's randomart image is:
+---[RSA 3072]-----+
|
|  . = . . * 0 = .
|  + + 0 + 0 = 0 0
|  . . . * = = . +
|  0 0 . 0 E 0 * 0 . .
|  0 0 + S . . + . 0
|  . + 0 . . . . .
|  0 B .
|  + +
|
+----[SHA256]-----+
usuario@ubuntu:~$
```

Copiamos esa clave al servidor del Worker

```

usuario@ubuntu:~$ ssh-copy-id ansible@192.168.120.2
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/usuario/.ssh/id_rsa.pub"
The authenticity of host '192.168.120.2 (192.168.120.2)' can't be established.
ECDSA key fingerprint is SHA256:w1NR1Pj6bJ1W14R9URVFbALfb/T4akPSJb0dgPN91zY.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
The authenticity of host '192.168.120.2 (192.168.120.2)' can't be established.
ECDSA key fingerprint is SHA256:w1NR1Pj6bJ1W14R9URVFbALfb/T4akPSJb0dgPN91zY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ansible@192.168.120.2's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ansible@192.168.120.2'"
and check to make sure that only the key(s) you wanted were added.

usuario@ubuntu:~$ ssh ansible@192.168.120.2
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jun 18 02:25:01 UTC 2023

System load:  0.07               Processes:    127
Usage of /:   34.6% of 19.56GB   Users logged in: 1
Memory usage: 13%              IPv4 address for ens3: 192.168.120.2
Swap usage:   0%

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

```

Y probamos el login realizando una conexión ssh de la máquina ansible con usuario ansible al worker y vemos que se realiza con éxito.

Ahora vamos a realizar la configuración básica de ansible y modificar la sección de escalado de permisos en el archivo ansible.cfg

Para ello hacemos uso del comando

```
cp /etc/ansible/ansible.cfg /home/ansible
```

Y luego editamos el fichero para modificar la parte de escalados de permisos.

```

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False

```

Se crea también el fichero inventory.ini o inventory.yaml como se prefiera. Y se realiza un ping ansible

```

ansible@ubuntu:/home/usuario$ ansible -i /etc/ansible/inventory.yaml worker1 -m ping
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host 192.168.120.2 should use /usr/bin/python3, but is using
/usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release will default to using the
discovered platform python for this host. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature will be
removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
192.168.120.2 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}

```

CREACIÓN DE MÁQUINAS VIRTUALES DE FORMA DESATENDIDA EN LA INFRAESTRUCTURA IAAS.ULL.ES

Instalación del SDK ovirt

Primeramente se va a instalar el software development kit de ovirt para permitirnos comunicarnos con la API de ovirt

- sudo apt install pip
- sudo apt install python3-pycurl
- sudo apt install libxml2-dev
- pip3 install ovirt-engine-sdk-python

Primero copiamos desde nuestra máquina local el fichero ipDM-ovejas-centos-8-ip-fijas.yaml.

```
C:\Users\aschafhauser\Downloads>scp IpDM-ovejas-centos-8.yaml usuario@10.6.130.185:/home/usuario
The authenticity of host '10.6.130.185 (10.6.130.185)' can't be established.
ECDSA key fingerprint is SHA256:w1NRlPj6bJ1W14R9URVFbALfb/T4akPSJb0dgPN9lzY.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '10.6.130.185' (ECDSA) to the list of known hosts.
usuario@10.6.130.185's password:
IpDM-ovejas-centos-8.yaml                                100% 3252    48.1KB/s   00:00
```

Y una vez tenemos el archivo en nuestra máquina router copiamos ese mismo archivo a la máquina ansible. Creamos un túnel ssh para hacer la copia.

```
C:\Users\aschafhauser\Downloads>ssh -L 9999:192.168.121.3:22 usuario@10.6.130.185
usuario@10.6.130.185's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jun 18 16:21:53 UTC 2023

System load:  0.0               Users logged in:      1
Usage of /:   35.9% of 19.56GB   IPv4 address for ens3: 10.6.130.185
Memory usage: 18%              IPv4 address for ens3: 10.6.130.95
Swap usage:   0%               IPv4 address for ens5: 192.168.120.1
Processes:   115               IPv4 address for ens7: 192.168.121.1

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

108 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Fri Jun 16 16:50:58 2023
usuario@ubuntu:~$
```

Se pretende conectar al frontend del iaas para la creación de las máquinas virtuales desatendidas. Para ello hay que autenticarse con nuestro usuario de la ULL, queremos pasarle esa contraseña al playboy pero que no sea visible. Para ello ejecutamos el siguiente comando

```
ansible-vault create alu0100893861@ull.yaml
```

Se nos solicita la contraseña del vault y una vez escrita se nos abre un fichero VI donde debemos escribir nuestra contraseña del IAAS.

Terminado este proceso ya tendremos el fichero con la información de nuestra contraseña encriptada.

MODIFICACIÓN DEL PLAYBOOK

Modificamos el fichero para adecuarlo a nuestra información. Para ellos vamos a cambiar

- vars_file → Deberemos poner el nombre del fichero que hemos creado con la contraseña ULL (alu0100893861ull.yaml en nuestro ejemplo)
- ssh_keys → Deberemos cambiarlo por el contenido del fichero /home/ansible/.ssh/id_rsa.pub
- ovirt_login → Nuestras credenciales ULL (alu0101012345@ULL)
- prefix → lpDM1
ull.es
- nodes → Aquí indicamos las máquinas a crear, donde especificamos el nombre que le vamos a asignar a cada máquina y la dirección IP que le vamos a asignar a cada una de ellas (tienen que ser direcciones dentro de la red 192.168.XXX asignadas a cada estudiante)
- profile_name → nombre de la red asignada

EJECUCIÓN DEL PLAYBOOK

Una vez configurado y modificado todo el playbook es hora de probarlo. Para ello ejecutamos el comando

```
ansible-playbook -vvv --ask-vault-pass lpDM-ovejas-centos-8-ip-fijas.yaml
```

```
&& sleep 0
ok: [localhost] => {
  "ansible_facts": {
    "ovirt_auth": {}
  },
  "changed": false,
  "invocation": {
    "module_args": {
      "ca_file": null,
      "compress": true,
      "headers": null,
      "hostname": null,
      "insecure": null,
      "kerberos": false,
      "ovirt_auth": {
        "ca_file": null,
        "compress": true,
        "headers": {
          "filter": true
        },
        "insecure": true,
        "kerberos": false,
        "timeout": 0,
        "token": "fb5FQlAr5TFC-iehIfk.j9R7DydFvTPLxPeMrIp3WA5JjipN16dqB5KT_vncNEu5txMNC04yjrxtCreBd_mGn7w",
        "url": "https://iaas.u11.es/ovirt-engine/api"
      },
      "password": null,
      "state": "absent",
      "timeout": 0,
      "token": null,
      "url": null,
      "username": null
    }
  }
}
META: ran handlers
META: ran handlers

PLAY RECAP *****
localhost                : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

usuario@ubuntu:~$
```

Este comando creará las dos ovejas que hemos configurado con sus respectivas IPs dentro del rango que le dimos de nuestra red.

Se introdujeron cambios en CentOS-8 que provocan que sea necesario migrar las ovejas creadas a Centos8-stream para ello primero añadimos la máquina a la red cambiándole el gateway por el correspondiente con la máquina router y luego ejecutamos los siguientes comandos.

```
sudo sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
sudo sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g'
/etc/yum.repos.d/CentOS-*
sudo dnf install centos-release-stream -y
sudo dnf swap centos-{linux,stream}-repos
sudo dnf distro-sync -y
```

Configuración del cluster de Hadoop

Primero se les añade un nombre apropiado para cada oveja creada en el playbook.

```
sudo hostnamectl set-hostname master-node  
sudo hostnamectl set-hostname worker-node1  
sudo hostnamectl set-hostname worker-node2
```

Y actualizamos el fichero /etc/hosts con las ips y los nombres de los nodos creados.

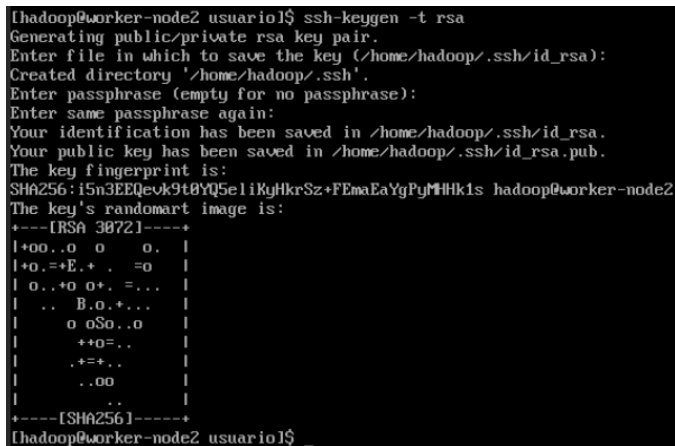
Creamos en todos los nodos un usuario hadoop con privilegios de sudo

```
sudo adduser -m hadoop -G wheel  
sudo passwd hadoop
```

El nodo master va a utilizar ssh para conectarse con los otros dos nodos, para ello configuramos un SSH key autenticación en cada nodo.

En cada nodo ejecutamos

```
ssh-keygen -t rsa
```



```
[hadoop@worker-node2 usuario1]$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):  
Created directory '/home/hadoop/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/hadoop/.ssh/id_rsa.  
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:15n3EEQevk9t0YQ5eliKyHkrSz+FEmaEaYgFyMHHk1s hadoop@worker-node2  
The key's randomart image is:  
+---[RSA 3072]-----+  
|+oo..o o o. |  
|+o.=+E.+ . =o |  
| o..+o o+. =... |  
| .. B.o.+... |  
| o oSo..o |  
| ++o=.. |  
| .+=+.. |  
| ..oo |  
| .. |  
+---[SHA256]-----+  
[hadoop@worker-node2 usuario1]$
```

Luego en el nodo del master ejecutamos los siguientes comandos:

```
ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub localhost  
scp /home/hadoop/.ssh/authorized_keys worker-node1:/home/hadoop/.ssh/
```

Hacemos lo mismo en el nodo del worker 1

```
ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub localhost  
scp /home/hadoop/.ssh/authorized_keys worker-master:/home/hadoop/.ssh/
```

y en el nodo worker 2.

```
ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub localhost  
scp /home/hadoop/.ssh/authorized_keys worker-master:/home/hadoop/.ssh/  
scp /home/hadoop/.ssh/authorized_keys worker-nodo1:/home/hadoop/.ssh/
```

Instalamos Java en cada uno de los nodos

```
sudo dnf install -y java-latest-openjdk java-latest-openjdk-devel
```

Y configuramos la variable JAVA_HOME

```
echo "JAVA_HOME=$(which java)" | sudo tee -a /etc/environment
```

```
[usuario@master-nodo ~]$ echo $JAVA_HOME  
/usr/bin/java
```

Se ha de añadir manualmente la localización de los binarios de hadoop en todos los nodos, esto se hace primero editando el fichero vi /home/hadoop/.bashrc y se le añade las líneas

```
export HADOOP_HOME=$HOME/hadoop  
export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin
```

Y por último editamos el fichero vi ~/.bash_profile y le añadimos:

```
PATH=$HOME/hadoop/bin:$HOME/hadoop/sbin:$PATH
```

Después de realizar todas estas acciones procedemos a instalar Hadoop.

Para ello primero hemos de instalar tar y wget para descargar y descomprimir el paquete.

```
sudo dnf install tar  
sudo dnf install wget  
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.gz  
tar xzf hadoop-3.3.4.tar.gz  
mv hadoop-3.3.4 hadoop
```

Configurar Hadoop

En el nodo del master escribimos el siguiente comando para encontrar la ruta de instalación de java.

```
update-alternatives --display java
```

Una vez encontramos el valor vamos a `~/hadoop/etc/hadoop/hadoop-env.sh` y editamos el fichero añadiendo la ruta a la variable `JAVA_HOME`.

```
# variable is REQUIRED on ALL platforms except OS X!  
export JAVA_HOME=/usr/lib/jvm/java-20-openjdk-20.0.1.0.9-8.rolling.el8.x86_64
```

Ahora editamos el fichero `core-site.xml` añadiendo la localización del master-node al puerto 9000. Editamos el fichero `~/hadoop/etc/hadoop/core-site.xml` y añadimos el código

```
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
<configuration>  
  <property>  
    <name>fs.default.name</name>  
    <value>hdfs://master-node:9000</value>  
  </property>  
</configuration>
```

El siguiente fichero de configuración es `~/hadoop/etc/hadoop/hdfs-site.xml` con el código:

```
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
<configuration>  
  <property>  
    <name>dfs.namenode.name.dir</name>  
    <value>/home/hadoop/data/nameNode</value>  
  </property>  
  <property>  
    <name>dfs.datanode.data.dir</name>  
    <value>/home/hadoop/data/dataNode</value>  
  </property>  
  <property>  
    <name>dfs.replication</name>  
    <value>2</value>  
  </property>  
</configuration>
```

Otro fichero que tendremos que editar es el fichero ~/hadoop/etc/hadoop/mapred-site.xml, añadiendo el código:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
    <value>512</value>
</property>
<property>
    <name>mapreduce.map.memory.mb</name>
    <value>256</value>
</property>
<property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>256</value>
</property>
</configuration>
```

El siguiente fichero que editamos en el master node es el ~/hadoop/etc/hadoop/yarn-site.xml y le añadimos el código :

```
<?xml version="1.0"?>
<configuration>
<property>
    <name>yarn.acl.enable</name>
    <value>0</value>
</property>
```

```

<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>192.168.120.3</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>2048</value>
</property>
<property>
  <name>yarn.scheduler.maximum-allocation-mb</name>
  <value>2048</value>
</property>
<property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>1024</value>
</property>
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>>false</value>
</property>
</configuration>

```

Por último editamos el fichero ~/hadoop/etc/hadoop/workers para así incluir a los dos nodos en el caso. Le añadimos al fichero el nombre de los nodos:

```

worker-node1
worker-node2

```

A continuación vamos a duplicar la configuración de hadoop a los workers-nodes para ello primero copiamos los ficheros de hadoop del master y los copiamos en los workers.

```

scp hadoop-*.tar.gz worker-node1:/home/hadoop/
scp hadoop-*.tar.gz worker-node2:/home/hadoop/

```

Ahora conectándonos por ssh a cada uno de los nodos extraemos el fichero tar.

```

tar xzf hadoop-3.3.1.tar.gz
mv hadoop-3.3.1 hadoop

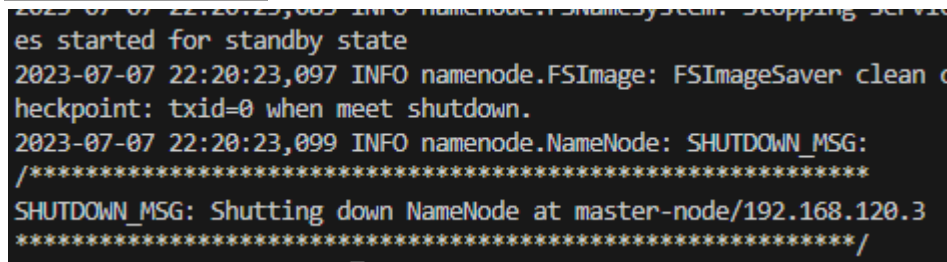
```

Y duplicamos la configuración de los nodos con el siguiente comando:

```
for node in worker-node1 worker-node2; do
scp ~/hadoop/etc/hadoop/* $node:/home/hadoop/hadoop/etc/hadoop/;
done
```

A continuación, en master-node como usuario de hadoop, escribimos el siguiente comando para formatear el sistema de archivos de hadoop:

```
hdfs namenode -format
```



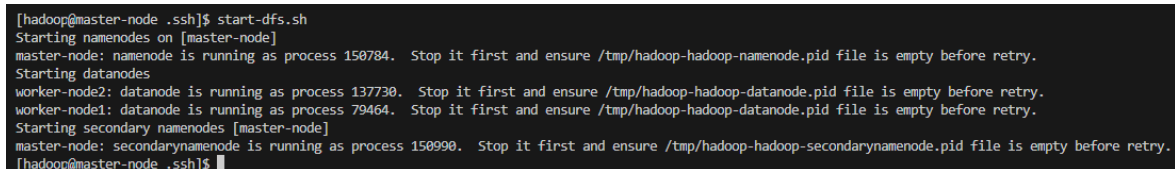
```
2023-07-07 22:20:23,005 INFO namenode.FSImageSaver: Stopping services started for standby state
2023-07-07 22:20:23,097 INFO namenode.FSImageSaver: clean checkpoint: txid=0 when meet shutdown.
2023-07-07 22:20:23,099 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at master-node/192.168.120.3
*****/
```

Con esta salida, podemos ver que ya tenemos el hadoop configurado y listo para funcionar.

Vamos por lo tanto a empezar un cluster de Hadoop.

Para iniciar usamos el comando desde el usuario hadoop en el nodo master:

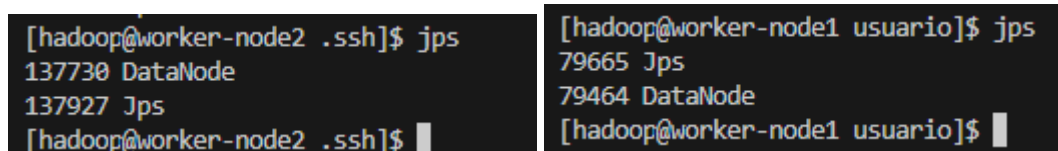
```
start-dfs.sh
```



```
[hadoop@master-node .ssh]$ start-dfs.sh
Starting namenodes on [master-node]
master-node: namenode is running as process 150784. Stop it first and ensure /tmp/hadoop-hadoop-namenode.pid file is empty before retry.
Starting datanodes
worker-node2: datanode is running as process 137730. Stop it first and ensure /tmp/hadoop-hadoop-datanode.pid file is empty before retry.
worker-node1: datanode is running as process 79464. Stop it first and ensure /tmp/hadoop-hadoop-datanode.pid file is empty before retry.
Starting secondary namenodes [master-node]
master-node: secondarynamenode is running as process 150990. Stop it first and ensure /tmp/hadoop-hadoop-secondarynamenode.pid file is empty before retry.
[hadoop@master-node .ssh]$
```

Obtendremos una salida como la que vemos en la figura anterior.

También podemos usar el comando jps para chequear que todos los procesos se están ejecutando.



```
[hadoop@worker-node2 .ssh]$ jps
137730 DataNode
137927 Jps
[hadoop@worker-node2 .ssh]$

[hadoop@worker-node1 usuario]$ jps
79665 Jps
79464 DataNode
[hadoop@worker-node1 usuario]$
```

Si queremos obtener más información podemos usar el comando

```
hdfs dfsadmin -report
```

Para probar hds primero, se crea un directorio de inicio, todos los demás comandos usarán una ruta relativa a este directorio de inicio predeterminado:

En el nodo maestro:

```
hdfs dfs -mkdir -p /user/hadoop
```

Se crea un directorio de libros en el sistema de archivos hadoop.

```
hdfs dfs -mkdir books
```

Haciendo funcionar hadoop.

Descargamos algunos libros para poder realizar el conteo de palabras más adelante.

```
wget -O heldwin.txt https://www.gutenberg.org/files/2701/2701-0.txt  
wget -O lewis.txt https://www.gutenberg.org/cache/epub/11/pg11.txt  
wget -O bram.txt https://www.gutenberg.org/cache/epub/345/pg345.txt
```

Después de descargar los libros vamos a ponerlos en el directorio recién creado books.

Para ello usamos el comando put

```
hdfs dfs -put bram.txt heldwin.txt lewis.txt books
```

```
[hadoop@master-node ~]$ hdfs dfs -put bram.txt heldwin.txt lewis.txt books  
[hadoop@master-node ~]$ hdfs dfs -ls books  
Found 3 items  
-rw-r--r--  2 hadoop supergroup      881691 2023-07-10 15:55 books/bram.txt  
-rw-r--r--  2 hadoop supergroup    1276235 2023-07-10 15:55 books/heldwin.txt  
-rw-r--r--  2 hadoop supergroup     174280 2023-07-10 15:55 books/lewis.txt  
[hadoop@master-node ~]$
```

Podemos listar el contenido con ls , para ver los libros que tenemos en el directorio books.

Se puede también imprimir el libro directamente en la terminal

```
hdfs dfs -cat books/bram.txt
```

O ver la documentación de hdfs con el comando -help.

```
hdfs dfs -help
```


Yarn

Apache Hadoop YARN (Yet Another Resource Negotiator) es una parte fundamental de la infraestructura de Apache Hadoop. YARN es un administrador de recursos y programación que permite a Hadoop procesar y gestionar eficientemente los recursos de un clúster distribuido.

Para iniciar yarn en el nodo master iniciamos con el comando

```
start-yarn.sh
```

En el jps ahora de los nodos vemos NodeManager mientras que en el Master vemos el proceso ResourceManager.

Mapreduce

MapReduce es un modelo de programación y un componente clave de Apache Hadoop para el procesamiento distribuido de grandes volúmenes de datos. Fue desarrollado originalmente por Google y luego adoptado por el proyecto Apache Hadoop.

El modelo de programación MapReduce se basa en dos operaciones principales: "map" (mapeo) y "reduce" (reducción). Aquí se explica cómo funciona:

Map: La fase de mapeo toma una entrada de datos y la divide en fragmentos más pequeños. Cada fragmento es procesado por una función de mapeo definida por el usuario que extrae información relevante y la transforma en pares clave-valor. Estos pares clave-valor intermedios son luego agrupados por clave antes de pasar a la fase de reducción.

Shuffle and Sort (Mezcla y Ordenamiento): En esta fase, los pares clave-valor intermedios generados por el mapeo son mezclados y ordenados por clave. Esto asegura que todos los pares de la misma clave sean enviados al mismo nodo para su posterior procesamiento en la fase de reducción.

Reduce: La fase de reducción toma los pares clave-valor de la fase anterior y los procesa mediante una función de reducción definida por el usuario. Esta función combina los valores asociados con la misma clave para generar los resultados finales.

En el nodo principal, envíe un trabajo con el jar de muestra a YARN:

```
yarn jar ~/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar wordcount "books/*" output
```

El último argumento (output) es la carpeta donde van a ser guardados los trabajos.

Una vez finalizado podemos ver el resultado consultado el siguiente comando:

```
hdfs dfs -ls output
```

```
[hadoop@master-node ~]$ hdfs dfs -ls output
Found 2 items
-rw-r--r--  2 hadoop supergroup      0 2023-07-10 15:59 output/_SUCCESS
-rw-r--r--  2 hadoop supergroup 516541 2023-07-10 15:59 output/part-r-00000
[hadoop@master-node ~]$
```

Y si queremos imprimir el resultado usamos el comando

```
hdfs dfs -cat output/part-r-00000 | less
```

```
""Tyke 1
""Wilhelmina"--I 1
""Yes, 1
"A 8
"ABRAHAM 1
"ART." 1
"ARTHUR." 1
"About 1
"Afraid 1
"Again 1
"Agreed!" 1
"Ah 2
"Ah, 16
"Aha! 1
"Aha!" 2
"Alas! 1
"All 7
"Already?" 1
"Am 2
"Amen" 1
"An 1
"An' 1
"And 49
"And, 1
"Answer 1
"Any 1
"Arabian 1
"Are 5
"Arthur 1
"Arthur! 2
"As 6
```