Instalar cluster Hadoop contenerizado

Lo ideal sería construir una imagen base con el sistema operativo, hadoop, configuración y librerías que necesitemos. Luego crear una red para las máquinas Hadoop y desplegarlas indicando los roles que van a desempeñar.

Para ello podemos partir de cero (descargar java, Hadoop, crear variables de entorno, copiar los archivos de configuración, etc) o partir de alguna configuración básica. En esta guía se utiliza esta última.

1. Comprobamos si tenemos instalado docker, para ello: sudo service docker status
2. Si la máquina no tiene instado Docker, entonces se instala con estos comandos:

    2.1. Actualizamos el gestor de paquetes, para ello: sudo yum update

    ```
    [UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo yum update
    ```

    2.2. Instalamos los paquetes necesarios, para ello: sudo yum install -y yum-utils device-mapper-persistent-data lvm2

    ```
    [UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo yum install -y yum-utils de
    vice-mapper-persistent-data lvm2
    ```

    2.3. Eliminamos los paquetes podman y buildha porque entran en conflicto con la instalación de docker, para ello: sudo yum remove podman buildah

    ```
    [UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo yum remove podman buildah
    ```

    Es probable que nos diga que no eliminó nada porque Linux Amazon (el sistema operativo utilizado en AWS worskpaces) no utiliza ni podman ni buildah por defecto, pero es recomendable ejecutarlo porque está basado en RHEL que suelen tener esos paquetes.

    2.4. Instalamos docker, para ello: sudo amazon-linux-extras install docker

    ```
    [UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo amazon-linux-extras install
     docker
    ```

    Puede que nos muestre un mensaje que ya estaba instalado.

3. Inicializamos docker: sudo service docker start

    ```
    [UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo service docker start
    ```

4. Importante asegurarse que esté arrancado antes de utilizarlo, para ello: sudo service docker status

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor prese
t: disabled)
   Active: active (running) since mar 2021-11-23 13:09:14 CET; 40s ago
     Docs: https://docs.docker.com
  Process: 9466 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=
exited, status=0/SUCCESS)
  Process: 9453 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SU
CCESS)
 Main PID: 9474 (dockerd)
    Tasks: 8
   Memory: 38.1M
   CGroup: /system.slice/docker.service
           └─9474 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/cont...

nov 23 13:09:14 a-1ougf97yd7b09 dockerd[9474]: time="2021-11-23T13:09:14.604...c
```

5. Comprobamos si está instalado docker-compose, para ello: docker-compose --version
6. Si no está instalado, instalamos docker-compose:
    6.1. Como usuarios root descargamos Docker-compose de Github, para ello: sudo curl -L "https://github.com/docker/compose/releases/download/1.28.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo curl -L "https://github.com
/docker/compose/releases/download/1.28.2/docker-compose-$(uname -s)-$(uname -m)"
 -o /usr/local/bin/docker-compose
```

    6.2. Le asignamos permisos de ejecución, para ello: sudo chmod +x /usr/local/bin/docker-compose

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ sudo chmod +x /usr/local/bin/doc
ker-compose
```

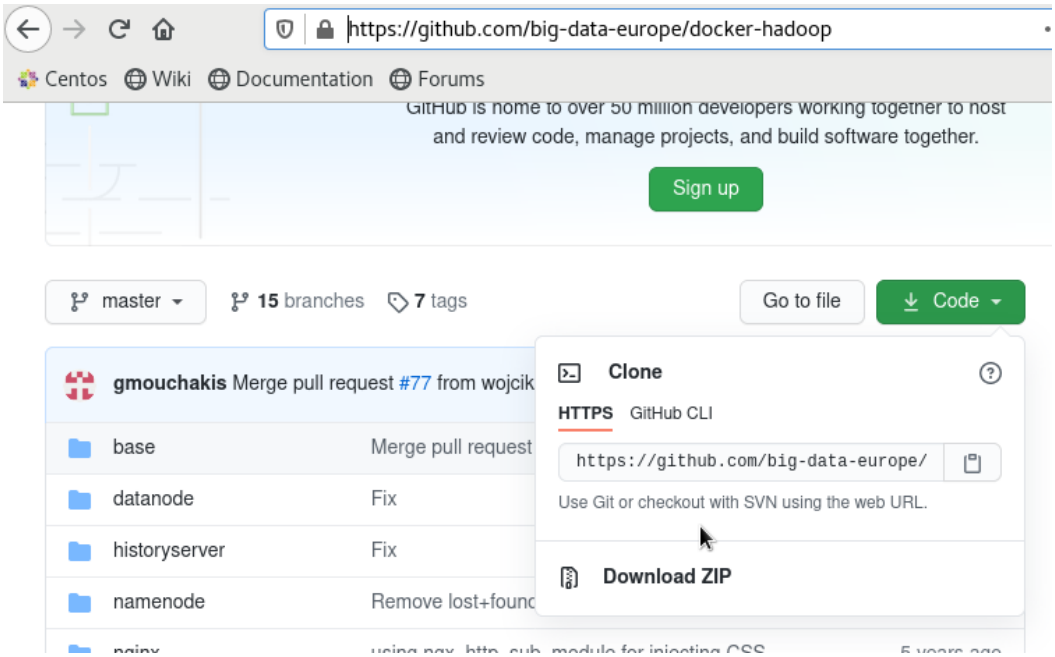
    6.3. Comprobamos que se instaló correctamente, para ello ejecutamos: docker-compose --version

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ docker-compose --version
docker-compose version 1.28.2, build 67630359
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$
```

7. Creamos un enlace simbólico de docker-compose en la carpeta /usr/bin que es donde está guardado docker. Para ello: sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo ln -s /usr/local/bin/docker-c
ompose /usr/bin/docker-compose
```

8. Descargamos las imágenes de Hadoop:
    8.1. Ir a la página: https://github.com/big-data-europe/docker-hadoop



8.2. Clickear en descargar como un zip

8.3. Nos ubicamos en la carpeta de descargas, para ello: c cd /home/jesus.moran/Descargas/

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$ cd /home/jesus.moran/Descargas/
```

Importante: es vuestro caso puede que esa carpeta se llame de forma diferente.

8.4. Descomprimimos el zip, para ello: unzip docker-hadoop-master.zip

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$ unzip docker-hadoop-master.zip
```

8.5. Entramos en la carpeta, para ello: cd docker-hadoop-master

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 Descargas]$ cd docker-hadoop-master
```

9. Desplegamos el cluster de 3 nodos, para ello: sudo docker-compose up -d

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$ sudo docker-compose up -d
```

…

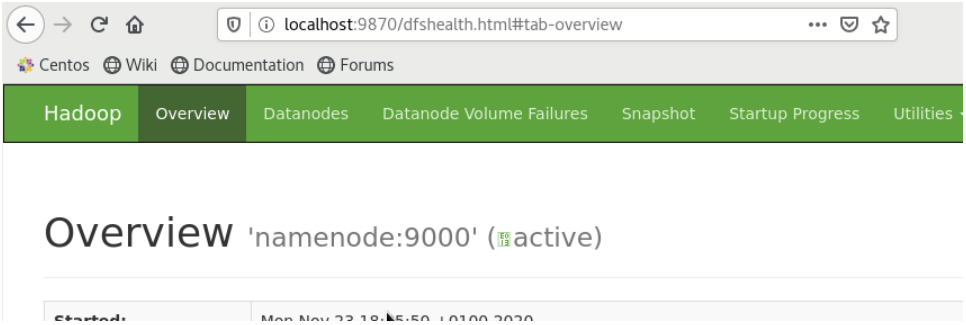
```
Creating namenode         ... done
Creating nodemanager      ... done
Creating resourcemanager  ... done
Creating datanode         ... done
Creating historyserver    ... done
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$
```

10. Comprobamos que se hayan desplegado los contenedores, para ello: docker ps

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$ sudo docker ps
CONTAINER ID   IMAGE                                                COMMAND                  CREA
TED         STATUS                   PORTS
                    NAMES
e8d5471f5a02   bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8     "/entrypoint.sh /run…"   2 mi
nutes ago   Up 2 minutes (healthy)   8188/tcp
                    historyserver
7d3d6262e6ea   bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   2 mi
nutes ago   Up 2 minutes (healthy)   9864/tcp
                    datanode
7208f6543169   bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   "/entrypoint.sh /run…"   2 mi
nutes ago   Up 2 minutes (healthy)   8088/tcp
                    resourcemanager
052b0f3712e1   bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8       "/entrypoint.sh /run…"   2 mi
nutes ago   Up 2 minutes (healthy)   8042/tcp
                    nodemanager
989fe93c8512   bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8          "/entrypoint.sh /run…"   2 mi
nutes ago   Up 2 minutes (healthy)   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp, 0.0.0.0:9870->9870/tc
p, :::9870->9870/tcp   namenode
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 docker-hadoop-master]$
```

11. Insertamos datos en HDFS:

11.1. Comprobamos que está activo el servidor web, para ello desde Firefox del anfitrión entramos en: http://localhost:9870



11.2. Creamos una carpeta llamada libros e introducimos unos libros:
mkdir libros
cd libros
curl https://www.gutenberg.org/files/2000/2000-0.txt > quijote.txt
curl https://www.gutenberg.org/files/345/345-0.txt > dracula.txt

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ mkdir libros
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 ~]$ cd libros/
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 libros]$ curl https://www.gutenberg.
org/files/2000/2000-0.txt > quijote.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 2173k  100 2173k    0     0   161k      0  0:00:13  0:00:13 --:--:--  573k
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 libros]$ curl https://www.gutenberg.
org/files/345/345-0.txt > dracula.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  860k  100  860k    0     0   159k      0  0:00:05  0:00:05 --:--:--  205k
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 libros]$
```

11.3. Copiamos esa carpeta al namenode, para ello: sudo docker cp ../libros namenode:/home/libros/

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 libros]$ sudo docker cp ../libros na
menode:/home/libros/
```

11.4. Nos conectamos al namenode, para ello: sudo docker exec -it namenode bash

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker exec -it namenode bash
root@989fe93c8512:/#
```

11.5. En el contenedor del namenode, nos ubicamos en la carpeta de los libros, para ello: cd /home/libros

```
root@989fe93c8512:/home/libros# cd /home/libros
```

11.6. En el contenedor del namenode, indicamos que se cree una carpeta en HDFS para guardar la información: hdfs dfs -mkdir -p librosEnHDFS
-p es para que cree las carpetas padres si no existen

```
root@989fe93c8512:/home/libros# hdfs dfs -mkdir -p librosEnHDFS
```

11.7. Cargamos los libros en hdfs, para ello: hdfs dfs -put ./ librosEnHDFS

```
root@989fe93c8512:/home/libros# hdfs dfs -put ./ librosEnHDFS
2021-11-23 19:36:08,602 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrust
ed = false, remoteHostTrusted = false
2021-11-23 19:36:08,879 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrust
ed = false, remoteHostTrusted = false
root@989fe93c8512:/home/libros#
```

11.8. Comprobamos que se cargaron, para ello: hdfs dfs -ls librosEnHDFS/libros

```
root@989fe93c8512:/home/libros# hdfs dfs -ls librosEnHDFS/libros
Found 2 items
-rw-r--r--   3 root supergroup     881473 2021-11-23 19:36 librosEnHDFS/libros/dracula.txt
-rw-r--r--   3 root supergroup    2226045 2021-11-23 19:36 librosEnHDFS/libros/quijote.txt
root@989fe93c8512:/home/libros#
```

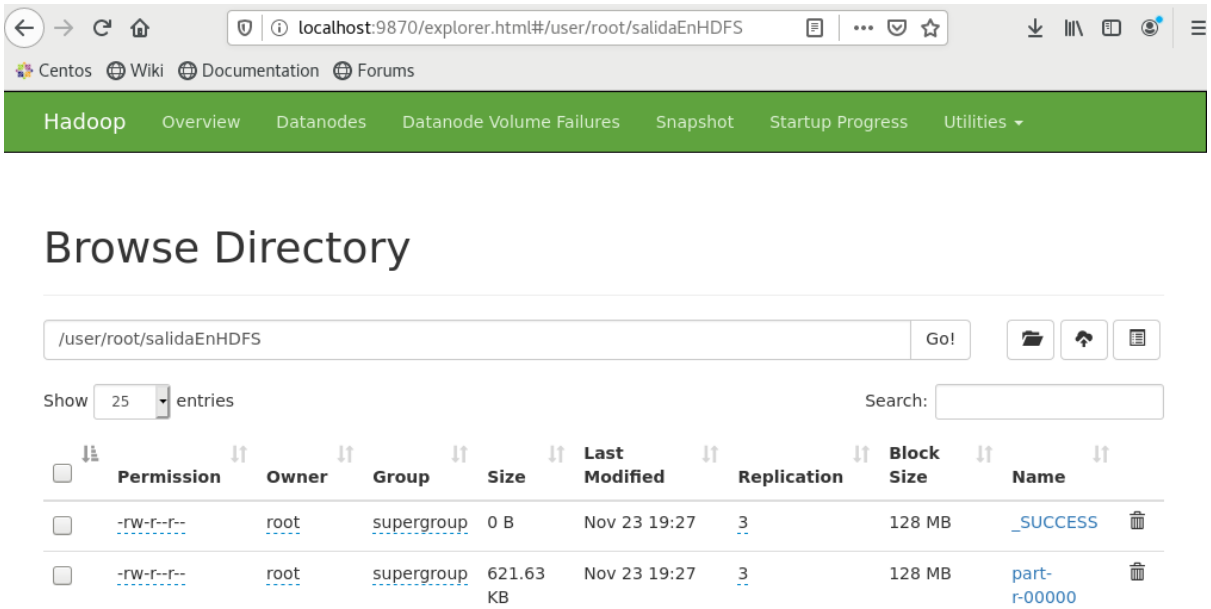
12. Ejecutamos un programa, para ello: hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-${HADOOP_VERSION}.jar wordcount /user/root/librosEnHDFS/libros/ /user/root/salidaEnHDFS

```
root@989fe93c8512:/home/libros# hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examp
les-${HADOOP_VERSION}.jar wordcount /user/root/librosEnHDFS/libros/ /user/root/salidaEnHDFS
```

```
File System Counters
        FILE: Number of bytes read=261055
        FILE: Number of bytes written=1211459
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=3107772
        HDFS: Number of bytes written=634327
        HDFS: Number of read operations=11
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
        HDFS: Number of bytes read erasure-coded=0
Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Rack-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=71168
        Total time spent by all reduces in occupied slots (ms)=32352
        Total time spent by all map tasks (ms)=17792
        Total time spent by all reduce tasks (ms)=4044
        Total vcore-milliseconds taken by all map tasks=17792
        Total vcore-milliseconds taken by all reduce tasks=4044
        Total megabyte-milliseconds taken by all map tasks=72876032
        Total megabyte-milliseconds taken by all reduce tasks=33128448
Map-Reduce Framework
        Map input records=53932
        Map output records=554129
        Map output bytes=5256412
        Map output materialized bytes=262487
        Input split bytes=254
        Combine input records=554129
        Combine output records=58803
        Reduce input groups=57678
        Reduce shuffle bytes=262487
        Reduce input records=58803
        Reduce output records=57678
        Spilled Records=117606
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=851
        CPU time spent (ms)=7400
        Physical memory (bytes) snapshot=1027985408
        Virtual memory (bytes) snapshot=18453131264
        Total committed heap usage (bytes)=888668160
        Peak Map Physical memory (bytes)=449466368
        Peak Map Virtual memory (bytes)=5043752960
        Peak Reduce Physical memory (bytes)=234344448
        Peak Reduce Virtual memory (bytes)=8376659968
```

```
Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
File Input Format Counters
        Bytes Read=3107518
File Output Format Counters
        Bytes Written=634327
```

13. Vemos en el servidor web de HDFS que se generó la salida:



14. Vemos por consola los primeros datos, para ello: hdfs dfs -head salidaEnHDFS/part-r-00000

```
root@989fe93c8512:/home/libros# hdfs dfs -head salidaEnHDFS/part-r-00000
2021-11-23 19:40:29,449 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrust
ed = false, remoteHostTrusted = false
!Mal    1
"'Are   1
"'E's   1
```

15. Salimos del contenedor, para ello: Control + D

```
root@989fe93c8512:/# exit
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$
```

16. Ahora vamos a copiar un programa que tenemos en el anfitrión (nuestro ordenador) al contenedor nodemanager (o cualquier otro del cluster) para poder ejecutarlo en el cluster:
    16.1. Desde el anfitrión entramos al nodemanager, para ello: sudo docker exec -it nodemanager bash

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker exec -it nodemanager bash
root@052b0f3712e1:/#
```

    16.2. Comprobamos que el contenedor tiene salida al exterior, para ello:
        16.2.1. Comprobamos que se puede hacer ping a una IP, para ello: ping 1.1.1.1

```
root@052b0f3712e1:/# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=52 time=1.07 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=52 time=1.03 ms
^C
--- 1.1.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.038/1.057/1.076/0.019 ms
root@052b0f3712e1:/#
```

        16.2.2. Dado que deja hacer ping a una IP, comprobamos si se puede hacer ping a un nombre de dominio, para ello: ping www.google.es

```
root@052b0f3712e1:/# ping www.google.es
ping: www.google.es: Temporary failure in name resolution
```

Hay dos opciones: que funcione correctamente, o que falle como en la captura anterior. En el caso de las máquinas Amazon Linux de AWS Workspace es probable que tengan la red bien configurada pero que fallen en la resolución de nombres porque utilizan un DNS que está inaccesible desde el contenedor (pero sí desde el anfitrión)

16.2.3. Si tenemos el problema de que no se pueden resolver los nombres de dominio, entonces hay que ponerle un DNS que sea accesible, por ejemplo el de IBM que es 9.9.9.9. Hay varias formas de hacerlo como modificando las imágenes, indicando en las opciones de docker que los contenedores hereden el DNS de IBM, indicando en el comando de creación de contenedores el DNS que queremos utilizar, etc. En este caso como es para algo que sólo vamos a necesitar una vez, vamos a modificarlo directamente en el contenedor cambiando el arhicov resolv.conf. Para ello:

16.2.4. Hacemos una copia del resolv.conf con el comando: cp /etc/resolv.conf /home/resolv.conf.bak

```
root@052b0f3712e1:/# cp /etc/resolv.conf /home/resolv.conf.bak
root@052b0f3712e1:/#
```

16.2.5. Asegurarse que se hizo una copia mediante: cat /home/resolv.conf.bak

```
root@052b0f3712e1:/# cat /home/resolv.conf.bak
search eu-west-1.compute.internal
nameserver 127.0.0.11
options timeout:2 attempts:5 ndots:0
root@052b0f3712e1:/#
```

16.2.6. Añadimos el DNS de IBM en el archivo resolv.conf. El problema es que Docker tiene bloqueado el archivo porque quiere actualizarlo automáticamente cuando se actualice la información del anfitrión. No obstante, aquí sólo lo necesitamos para una simple instalación, entonces lo modificamos con el siguiente comando que permite modificarlo incluso estando resolv.conf bloqueado: echo "$(sed -n 'H;${x;s/^\n//;s/nameserver .*$/nameserver 9.9.9.9\n&/;p}' /etc/resolv.conf)" > /etc/resolv.conf

```
root@052b0f3712e1:/# echo "$(sed -n 'H;${x;s/^\n//;s/nameserver .*$/nameserver 9.9.9.9\n&/;p}' /etc/resolv.c
onf)" > /etc/resolv.conf
root@052b0f3712e1:/#
```

16.2.7. Comprobamos que se añadió el nuevo DNS al archivo, para ello: cat /etc/resolv.conf

```
root@052b0f3712e1:/# cat /etc/resolv.conf
search eu-west-1.compute.internal
nameserver 9.9.9.9
nameserver 127.0.0.11
options timeout:2 attempts:5 ndots:0
root@052b0f3712e1:/#
```

16.3. Le instalamos python, para ello:

16.3.1. Actualizamos los paquetes del contenedor, para ello: apt-get update

```
root@052b0f3712e1:/# apt-get update
```

16.3.2. Instalamos python en el contenedor, para ello: apt-get install python3-pip

```
root@052b0f3712e1:/# apt-get install python3-pip
```

De igual forma se pueden instalar otras herramientas que puede ser útiles como por ejemplo el editor nano (apt-get install nano). Notar que si se quiere hacer una instalación personalizada para desplegar el cluster varias veces, entonces lo ideal sería modificar las imágenes y crear contenedores con las nuevas imágenes. Si en cambio lo único que queremos es probar si nos funciona un programa y tener un cluster para desarrollar o probar, puede ser suficiente haciendo la instalación de python manualmente.

16.3.3. Ahora que ya hemos instalado python, volvemos a poner el resolv.conf que teníamos, para ello: cp /home/resolv.conf.bak /etc/resolv.conf

```
root@052b0f3712e1:/# cp /home/resolv.conf.bak /etc/resolv.conf
```

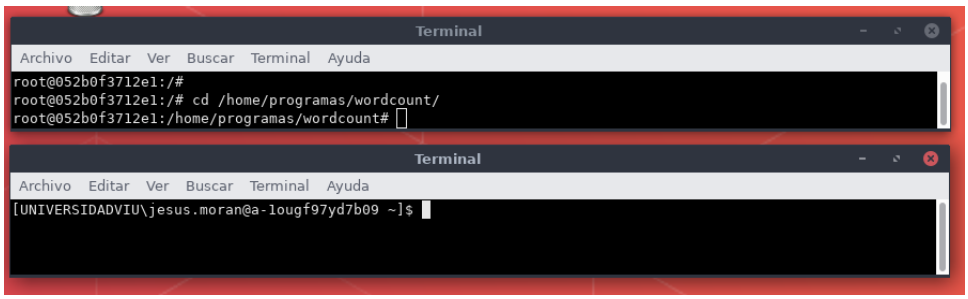
16.4. En la terminal del contenedor creamos una carpeta programas en home, para ello: mkdir -p /home/programas/wordcount

```
root@052b0f3712e1:/# mkdir -p /home/programas/wordcount
```

16.5. En la terminal del contenedor nos ubicamos en la carpeta que acabamos de crear: cd /home/programas/wordcount/

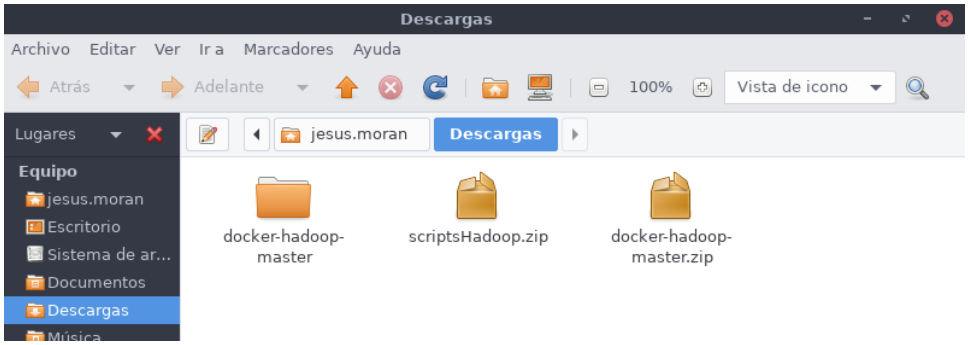
```
root@052b0f3712e1:/# cd /home/programas/wordcount/
root@052b0f3712e1:/home/programas/wordcount#
```

16.6. Abrimos una nueva terminal en el anfitrión (el ordenador desde el que estamos trabajando), por lo que se tendrán dos terminales (la del contenedor) y la del anfitrión



Fijarse que el prompt (lo primero que sale escrito), cambia. En la terminal del contenedor nos sale root@números y en la del anfitrión sale UNIVERSIDADVIU/nombre@letras (puede que sea distinto, pero tenemos que tener clara cuál es la del anfitrión y cuál la del contenedor)

16.7. En la máquina anfitrión descargamos del campus virtual los programas de Hadoop. Por defecto se descargaran en la carpeta Descargas:



16.8. Desde la terminal del anfitrión, nos ubicamos en la carpeta de descargas: cd ~/Descargas/

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 ~]$ cd ~/Descargas/
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 Descargas]$
```

16.9. Desde la terminal del anfitrión descomprimimos los scripts, para ello: unzip ./scriptsHadoop.zip

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 Descargas]$ unzip ./scriptsHadoop.zip
```

16.10. En la terminal del anfitrión, nos ubicamos en la carpeta de wordcount, para ello: cd ./scriptsHadoop/wordcountPython

```
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 Descargas]$ cd ./scriptsHadoop/wordcountPython
[UNIVERSIDADVIU\jesus.moran@a-lougf97yd7b09 wordcountPython]$
```

16.11. En la terminal del anfitrión copiamos el programa Mapper al contenedor, para ello: sudo docker cp mapperWordCount.py nodemanager:/home/programas/wordcount

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 wordcountPython]$ sudo docker cp mapperWordCount.py nodemanager:
/home/programas/wordcount
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 wordcountPython]$
```

16.12. En la terminal del anfitrión copiamos el programa Reducer al contenedor, para ello: sudo docker cp reducerWordCount.py nodemanager:/home/programas/wordcount

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 wordcountPython]$ sudo docker cp reducerWordCount.py nodemanager
:/home/programas/wordcount
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 wordcountPython]$
```

16.13. En la terminal del contenedor listamos los archivos que hay en la carpeta que contiene el programa, para ello: ls -l

Nota: si no estamos ubicados en la carpeta que contiene el programa, hay que ubicarse en esa carpeta: cd /home/programas/wordcount/

```
root@052b0f3712e1:/home/programas/wordcount# ls -l
total 8
-rwxr-xr-x 1 68479 66049 260 Nov 20 11:58 mapperWordCount.py
-rwxr-xr-x 1 68479 66049 828 Nov 20 12:00 reducerWordCount.py
root@052b0f3712e1:/home/programas/wordcount#
```

16.14. Ejecutamos el programa, para ello desde la terminal del contenedor ejecutamos: hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-${HADOOP_VERSION}.jar -files ./mapperWordCount.py,./reducerWordCount.py -mapper mapperWordCount.py -reducer reducerWordCount.py -combiner ./reducerWordCount.py -input /user/root/librosEnHDFS/libros/ -output /user/root/salidaWordcountPython

```
root@052b0f3712e1:/home/programas/wordcount# hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming
-${HADOOP_VERSION}.jar -files ./mapperWordCount.py,./reducerWordCount.py -mapper mapperWordCount.py -reducer
 reducerWordCount.py -combiner ./reducerWordCount.py -input /user/root/librosEnHDFS/libros/ -output /user/ro
ot/salidaWordcountPython
        Map-Reduce Framework
                Map input records=53932
                Map output records=554129
                Map output bytes=4148154
                Map output materialized bytes=374754
                Input split bytes=342
                Combine input records=554129
                Combine output records=68803
                Reduce input groups=57678
                Reduce shuffle bytes=374754
                Reduce input records=68803
                Reduce output records=57678
                Spilled Records=137606
                Shuffled Maps =3
                Failed Shuffles=0
                Merged Map outputs=3
                GC time elapsed (ms)=933
                CPU time spent (ms)=8630
                Physical memory (bytes) snapshot=1049358336
                Virtual memory (bytes) snapshot=23474995200
                Total committed heap usage (bytes)=984612864
                Peak Map Physical memory (bytes)=299458560
                Peak Map Virtual memory (bytes)=5033308160
                Peak Reduce Physical memory (bytes)=214790144
                Peak Reduce Virtual memory (bytes)=8376233984
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=3111614
        File Output Format Counters
                Bytes Written=1047819
2021-11-24 15:30:31,328 INFO streaming.StreamJob: Output directory: /user/root/salidaWordcountPython
root@052b0f3712e1:/home/programas/wordcount#
```

16.15. Comprobamos que se generó la salida, para ello: hdfs dfs -head salidaWordcountPython/part-00000

```
root@052b0f3712e1:/home/programas/wordcount# hdfs dfs -head salidaWordcountPython/part-00000
2021-11-24 15:32:08,556 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = fa
lse, remoteHostTrusted = false
!Mal    1
"'Are   1
```

16.16. Salimos del contenedor pulsando control+D

```
root@052b0f3712e1:/home/programas/wordcount# exit
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$
```

17. Apagamos el cluster, para ello desde una terminal del anfitrión y situados en la carpeta de los contenedores, ejecutamos: sudo docker-compose stop

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker-compose stop
Stopping historyserver   ... done
Stopping nodemanager     ... done
Stopping datanode        ... done
Stopping namenode        ... done
Stopping resourcemanager ... done
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$
```

18. Si queremos volver a arrancar el cluster, desde una terminal del anfitrión y situados en la carpeta de los contenedores, ejecutamos: sudo docker-compose start

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker-compose start
Starting namenode        ... done
Starting datanode        ... done
Starting resourcemanager ... done
Starting nodemanager1    ... done
Starting historyserver   ... done
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$
```

18.1. Si entramos en el nodemanager veremos que tenemos los archivos y las instalaciones que hicimos. No obstante, es recomendable hacer copias de seguridad de todos los archivos tanto en AWS workspaces como en otro medio por si fallase el contenedor o AWS workspaces:

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker exec -it nodemanager bash
root@e99644dfcf65:/# python3 --version
Python 3.5.3
root@e99644dfcf65:/#
```

```
root@e99644dfcf65:/# ls -l /home/programas/wordcount/
total 8
-rwxr-xr-x 1 68479 66049 260 Nov 20 11:58 mapperWordCount.py
-rwxr-xr-x 1 68479 66049 828 Nov 20 12:00 reducerWordCount.py
root@e99644dfcf65:/#
```

```
root@e99644dfcf65:/home/programas/wordcount# hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming
-${HADOOP_VERSION}.jar -files ./mapperWordCount.py,./reducerWordCount.py -mapper mapperWordCount.py -reducer
 reducerWordCount.py -combiner ./reducerWordCount.py -input /user/root/librosEnHDFS/libros/ -output /user/ro
ot/salidaWordcountPython_tras_reiniciar
```

```
root@e99644dfcf65:/home/programas/wordcount# hdfs dfs -head salidaWordcountPython_tras_reiniciar/part-00000
2021-11-24 15:59:27,622 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = fa
lse, remoteHostTrusted = false
!Mal    1
"'Are   1
"'E's   1
```

19. Si en algún momento queremos eliminar el cluster, desde una terminal del anfitrión y situados en la carpeta de los contenedores, ejecutamos: sudo docker-compose down

```
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$ sudo docker-compose down
Stopping historyserver   ... done
Stopping datanode        ... done
Stopping resourcemanager ... done
Stopping nodemanager     ... done
Stopping namenode        ... done
Removing historyserver   ... done
Removing datanode        ... done
Removing resourcemanager ... done
Removing nodemanager     ... done
Removing namenode        ... done
Removing network docker-hadoop-master_default
[UNIVERSIDADVIU\jesus.moran@a-1ougf97yd7b09 docker-hadoop-master]$
```

Importante: si volvemos a crear contenedores, los datos de hdfs podrían estar disponibles porque sólo se eliminaronn los contenedores. Si queremos eliminar también los volúmenes (los que contienen los datos), entonces tendremos que ejecutar: sudo docker-compose down -v