

## Instalar cluster Hadoop contenerizado

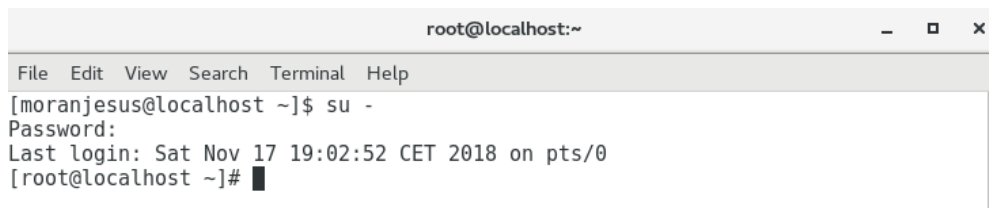
Lo ideal sería construir una imagen base con el sistema operativo, hadoop, configuración y librerías que necesitemos. Luego crear una red para las máquinas Hadoop y desplegarlas indicando los roles que van a desempeñar.

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. Instalar Docker:

#### 1.1. Nos cambiamos a root, para ello: su -

Insertamos la contraseña que añadimos como root durante la instalación



```
root@localhost:~  
File Edit View Search Terminal Help  
[moranjesus@localhost ~]$ su -  
Password:  
Last login: Sat Nov 17 19:02:52 CET 2018 on pts/0  
[root@localhost ~]#
```

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

```
[root@localhost ~]# sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

#### 1.3. Añadimos el repositorio docker, para ello: sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

```
[root@localhost ~]# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

Notar que aunque estemos instalando el paquete de CentOS, sirve para Almalinux. De momento no hay una versión específica para Almalinux

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

```
[root@localhost ~]# sudo yum remove podman buildah
```

#### 1.5. Instalamos docker, para ello: sudo yum install docker-ce docker-ce-cli containerd.io

```
[root@localhost ~]# sudo yum install docker-ce docker-ce-cli containerd.io
```

#### 1.6. Inicializamos docker: sudo systemctl start docker

```
[root@localhost ~]# sudo systemctl start docker
```

También se puede hacer “sudo systemctl enable --now docker” si queremos que, además de arrancar el servicio, quede activado automáticamente al arrancar la máquina. Con el start sólo se arrancará ahora mismo, pero al reiniciar/apagar la máquina, habrá que volver a ejecutar el start para arrancarlo.

### 2. Instalaos docker-compose:

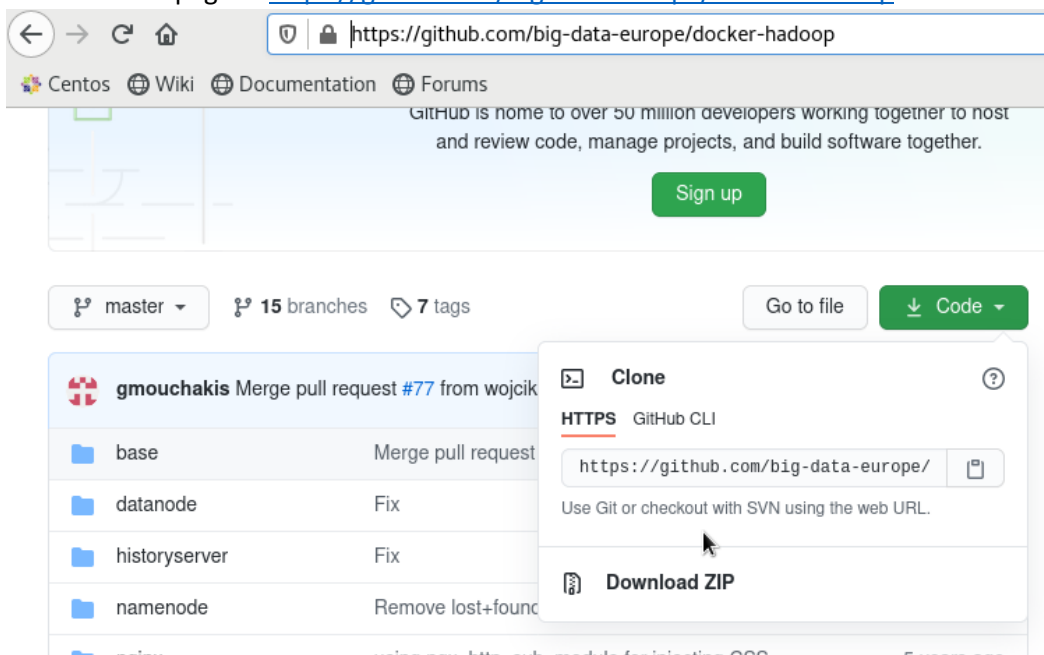
- 2.1. Como usuarios root descargamos Docker-compose de Github, para ello: `sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`

```
[root@localhost ~]# sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

- 2.2. Le asignamos permisos de ejecución, para ello: `sudo chmod +x /usr/local/bin/docker-compose`
- ```
[root@localhost ~]# sudo chmod +x /usr/local/bin/docker-compose
```
- 2.3. Comprobamos que se instaló correctamente, para ello ejecutamos: `docker-compose --version`
- ```
[root@localhost ~]# docker-compose --version
docker-compose version 1.27.4, build 40524192
[root@localhost ~]#
```

3. Descargamos las imágenes de Hadoop:

- 3.1. Ir a la página: <https://github.com/big-data-europe/docker-hadoop>



- 3.2. Clickear en descargar como un zip
- 3.3. Nos ubicamos en la carpeta de descargas, para ello: `cd /home/moranjesus/Downloads`
- 3.4. Descomprimos el zip, para ello: `unzip docker-hadoop-master.zip`
- ```
[root@localhost Downloads]# unzip docker-hadoop-master.zip
```
- 3.5. Entramos en la carpeta, para ello: `cd docker-hadoop-master`
- ```
[root@localhost Downloads]# cd docker-hadoop-master
[root@localhost docker-hadoop-master]#
```

4. Desplegamos el cluster de 3 nodos, para ello: `docker-compose up -d`
- ```
[root@localhost docker-hadoop-master]# docker-compose up -d
```

...

```

Creating historyserver    ... done
Creating nodemanager      ... done
Creating datanode         ... done
Creating resource manager ... done
Creating namenode         ... done
[root@localhost docker-hadoop-master]#

```

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

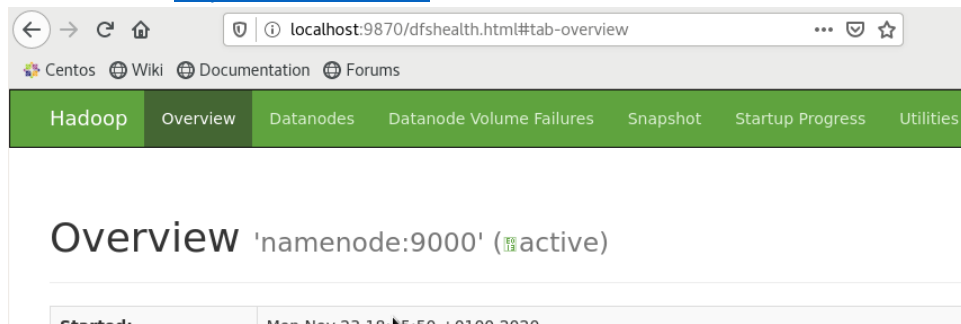
```

[root@localhost docker-hadoop-master]# docker ps
CONTAINER ID   STATUS    PORTS                               COMMAND                                CREATED
1399da4bfa7e   Up 9 minutes (healthy)   8042/tcp                             "/entrypoint.sh /run..."           9 minutes
ago            bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8   nodemanager
e4ef316095a7   Up 9 minutes (healthy)   9864/tcp                             "/entrypoint.sh /run..."           9 minutes
ago            bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8     datanode
a8516291d3bf   Up 9 minutes (healthy)   8088/tcp                             "/entrypoint.sh /run..."           9 minutes
ago            bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8   resource manager
88c9bce49fea   Up 9 minutes (healthy)   8188/tcp                             "/entrypoint.sh /run..."           9 minutes
ago            bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8   historyserver
ddc27b9470d0   Up 9 minutes (healthy)   0.0.0.0:9000->9000/tcp, 0.0.0.0:9870->9870/tcp "/entrypoint.sh /run..."           9 minutes
ago            bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8
[root@localhost docker-hadoop-master]#

```

##### 6. Insertamos datos en HDFS:

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



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

```

[root@localhost docker-hadoop-master]# docker exec -it namenode bash
root@ddc27b9470d0:/#

```

###### 6.3. Descargamos varios libros, para ello:

```

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
root@4b72dbd0d4ce:/# mkdir libros
root@4b72dbd0d4ce:/# cd libros
root@4b72dbd0d4ce:/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 1198k      0  0:00:01  0:00:01 --:--:-- 1198k
root@4b72dbd0d4ce:/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 682k      0  0:00:01  0:00:01 --:--:-- 682k
root@4b72dbd0d4ce:/libros#

```

###### 6.4. 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@ddc27b9470d0:/libros# hdfs dfs -mkdir -p librosEnHDFS
root@ddc27b9470d0:/libros#

```

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

```

root@ddc27b9470d0:/libros# hdfs dfs -put ./ librosEnHDFS
2020-11-23 18:10:24,845 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localHostTrusted = false, remoteHostTrusted = false
2020-11-23 18:10:25,127 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localHostTrusted = false, remoteHostTrusted = false
root@ddc27b9470d0:/libros# █

```

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

```

root@ddc27b9470d0:/libros# hdfs dfs -ls librosEnHDFS/libros
Found 2 items
-rw-r--r--  3 root supergroup      883160 2020-11-23 18:10 librosEnHDFS/libros/d
racula.txt
-rw-r--r--  3 root supergroup      2198927 2020-11-23 18:10 librosEnHDFS/libros/q
uijote.txt
root@ddc27b9470d0:/libros# █

```

## 7. 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@ddc27b9470d0:/libros# hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-${HADOOP_VERSION}.jar wordcount /user/root/librosEnHDFS/libros/ /user/root/salidaEnHDFS █

```

```

2020-11-23 18:27:55,358 INFO mapreduce.Job: Counters: 54
  File System Counters
    FILE: Number of bytes read=262244
    FILE: Number of bytes written=1213814
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=3082341
    HDFS: Number of bytes written=636546
    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)=267772
    Total time spent by all reduces in occupied slots (ms)=122600
    Total time spent by all map tasks (ms)=66943
    Total time spent by all reduce tasks (ms)=15325
    Total vcore-milliseconds taken by all map tasks=66943
    Total vcore-milliseconds taken by all reduce tasks=15325
    Total megabyte-milliseconds taken by all map tasks=274198528
    Total megabyte-milliseconds taken by all reduce tasks=125542400
  Map-Reduce Framework
    Map input records=53834
    Map output records=548684
    Map output bytes=5204173
    Map output materialized bytes=263653
    Input split bytes=254
    Combine input records=548684
    Combine output records=59084
    Reduce input groups=57977
    Reduce shuffle bytes=263653
    Reduce input records=59084
    Reduce output records=57977
    Spilled Records=118168
    Shuffled Maps =2
    Failed Shuffles=0
    Merged Map outputs=2
    GC time elapsed (ms)=807
    CPU time spent (ms)=5510
    Physical memory (bytes) snapshot=519704576
    Virtual memory (bytes) snapshot=18100113408
    Total committed heap usage (bytes)=392372224
    Peak Map Physical memory (bytes)=207884288
    Peak Map Virtual memory (bytes)=4953128960
    Peak Reduce Physical memory (bytes)=113123328
    Peak Reduce Virtual memory (bytes)=8193855488

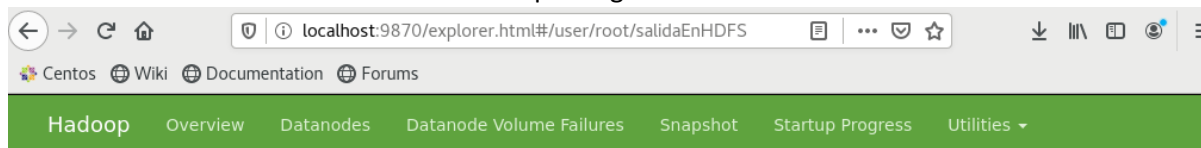
```

```

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=3082087
File Output Format Counters
  Bytes Written=636546

```

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



## Browse Directory

/user/root/salidaEnHDFS

Go!

Show

25

entries

Search:

| <div></div> | Permission | Owner | Group      | Size      | Last Modified | Replication | Block Size | Name         |             |
|-------------|------------|-------|------------|-----------|---------------|-------------|------------|--------------|-------------|
| <div></div> | -rw-r--r-- | root  | supergroup | 0 B       | Nov 23 19:27  | 3           | 128 MB     | _SUCCESS     | <div></div> |
| <div></div> | -rw-r--r-- | root  | supergroup | 621.63 KB | Nov 23 19:27  | 3           | 128 MB     | part-r-00000 | <div></div> |

9. Vemos por consola los primeros datos, para ello: `hadoop fs -cat salidaEnHDFS/part-r-00000`

```

| head
root@ddc27b9470d0:/libros# hadoop fs -cat salidaEnHDFS/part-r-00000 | head
2020-11-23 18:33:46,175 INFO sasl.SaslDataTransferClient: SASL encryption trust
check: localhostTrusted = false, remoteHostTrusted = false
!Mal      1
"'Are      1
"'E's      1
"'I        1
"'Ittin'   1
"'Little   1
"'Lucy,    1
"'Maybe   1
"'Miss     1
"'My       2
cat: Unable to write to output stream.
root@ddc27b9470d0:/libros#

```

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

```

root@ddc27b9470d0:/libros# exit
[root@localhost docker-hadoop-master]#

```

11. 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:

11.1. Desde el anfitrión entramos al nodemanager, para ello: `sudo docker exec -it nodemanager bash`

```

[root@localhost docker-hadoop-master]# sudo docker exec -it nodemanager bash
root@e2c7a0d1e076:/#

```

11.2. Le instalamos python, para ello:

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

```
root@e2c7a0d1e076:/# apt-get update
```

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

```
root@e2c7a0d1e076:/# 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.

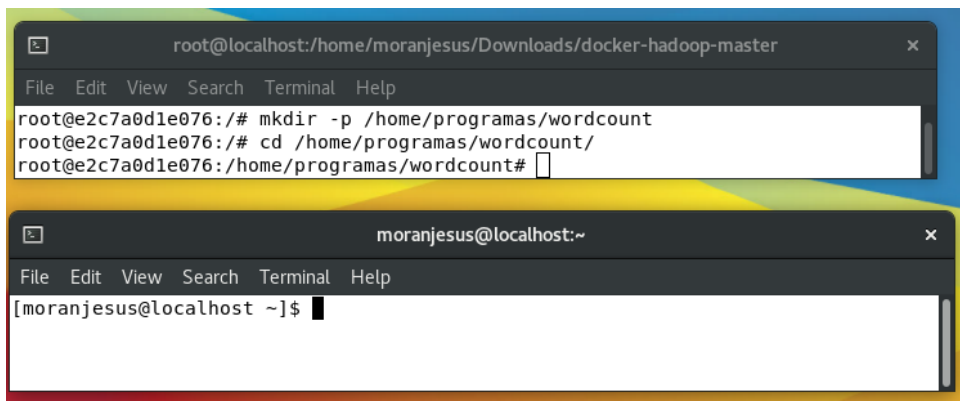
- 11.3. En la terminal del contenedor creamos una carpeta programas en home, para ello: mkdir -p /home/programas/wordcount

```
root@e2c7a0d1e076:/# mkdir -p /home/programas/wordcount
```

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

```
root@e2c7a0d1e076:/home/programas/wordcount#
```

- 11.5. 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 propt (lo primero que sale escrito), cambia. En la terminal del contenedor nos sale root@números y en la del anfitrión sale usuario@localhost (puede ser moranjesus@localhost, root@localhost o con el usuario que estemos)

- 11.6. En la terminal del anfitrión nos ponemos como usuario root, para ello: su -

- 11.7. En la terminal del anfitrión nos ubicamos en la carpeta que tenemos el programa: cd /home/moranjesus/Desktop/wordcountPython

```
root@localhost:/home/moranjesus/Desktop/wordcountPython
[moranjesus@localhost ~]$ su -
Password:
[root@localhost ~]# cd /home/moranjesus/Desktop/wordcountPython
[root@localhost wordcountPython]#
```

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

```
[root@localhost wordcountPython]# docker cp mapperWordCount.py nodemanager:/home/programas/wordcount
```

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

```
[root@localhost wordcountPython]# docker cp reducerWordCount.py nodemanager:/home/programas/wordcount
```

- 11.10. 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@e2c7a0d1e076:/home/programas/wordcount# ls -l
total 8
-rwxr-xr-x. 1 1000 1000 260 Nov 20 11:58 mapperWordCount.py
-rwxr-xr-x. 1 1000 1000 828 Nov 20 12:00 reducerWordCount.py
root@e2c7a0d1e076:/home/programas/wordcount#
```

- 11.11. 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@e2c7a0d1e076:/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/root/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)=443
  CPU time spent (ms)=6230
  Physical memory (bytes) snapshot=756670464
  Virtual memory (bytes) snapshot=23059623936
  Total committed heap usage (bytes)=558116864
  Peak Map Physical memory (bytes)=221962240
  Peak Map Virtual memory (bytes)=4982099968
  Peak Reduce Physical memory (bytes)=119529472
  Peak Reduce Virtual memory (bytes)=8193589248

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
2021-11-22 15:00:21,281 INFO streaming.StreamJob: Output directory: /user/root
/salidaWordcountPython
root@e2c7a0d1e076:/home/programas/wordcount# █

```

11.12. Comprobamos que se generó la salida, para ello: `hadoop fs -head`

`salidaWordcountPython/part-00000`

```

root@e2c7a0d1e076:/home/programas/wordcount# hadoop fs -head salidaWordcountPy
thon/part-00000
2021-11-22 15:07:54,131 INFO sasl.SaslDataTransferClient: SASL encryption trus
t check: localhostTrusted = false, remoteHostTrusted = false
!Mal      1
''Are     1
''E's     1
''I       1
'''

```

11.13. Salimos del contenedor pulsando control+D

```

root@e2c7a0d1e076:/home/programas/wordcount# exit
[root@localhost docker-hadoop-master]# █

```

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

```

[root@localhost docker-hadoop-master]# docker-compose stop
Stopping datanode      ... done
Stopping nodemanager   ... done
Stopping historyserver ... done
Stopping namenode      ... done
Stopping resourcemanager ... done
[root@localhost docker-hadoop-master]# █

```

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



```
[root@localhost docker-hadoop-master]# docker-compose start
Starting namenode      ... done
Starting datanode      ... done
Starting resourcemanager ... done
Starting nodemanager1  ... done
Starting historyserver ... done
[root@localhost docker-hadoop-master]# █
```

Los contenedores tendrán las instalaciones y datos que introducimos anteriormente.

```
[root@localhost docker-hadoop-master]# sudo docker exec -it nodemanager bash
root@702fd0eaab29:/# python3 --version
Python 3.5.3
root@702fd0eaab29:/# █
```

---

```
root@702fd0eaab29:/# ls -l /home/programas/wordcount/
total 8
-rwxr-xr-x. 1 1000 1000 260 Nov 20 11:58 mapperWordCount.py
-rwxr-xr-x. 1 1000 1000 828 Nov 20 12:00 reducerWordCount.py
root@702fd0eaab29:/# █
```

---

No obstante, es recomendable que se hagan copias de seguridad de los datos incluso fuera de la máquina virtual por si hay algún fallo.

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

```
[root@localhost docker-hadoop-master]# docker-compose down
Stopping nodemanager      ... done
Stopping datanode         ... done
Stopping resourcemanager  ... done
Stopping historyserver    ... done
Stopping namenode         ... done
Removing nodemanager      ... done
Removing datanode         ... done
Removing resourcemanager  ... done
Removing historyserver    ... done
Removing namenode        ... done
Removing network docker-hadoop-master_default
[root@localhost docker-hadoop-master]# █
```

---

Nota: el anterior comando elimina los contenedores y la red, pero no los volúmenes. Es decir, si volvemos a crear los contenedores, los datos de HDFS podrían estar disponibles. Si queremos eliminar también los volúmenes, tenemos que ejecutar: `docker-compose down -v`