TECNOLOXÍAS DE COMPUTACIÓN PARA DATOS MASIVOS

Entrega 1

ANDRÉS CAMPOS CUIÑA

FECHA DE ENTREGA: 17/10/2021

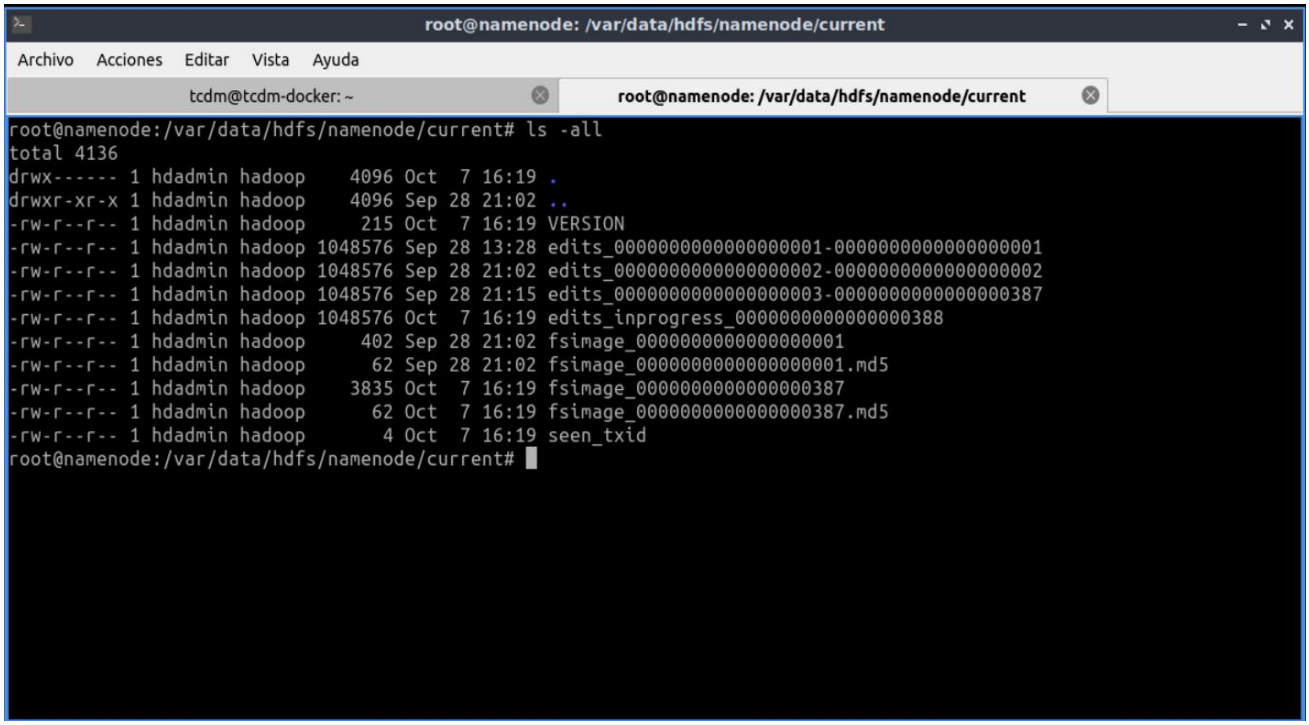
ÍNDICE

1 AÑADIR AL CLUSTER UN SERVIDOR DE BACKUP Y UN TIMELINESERVER

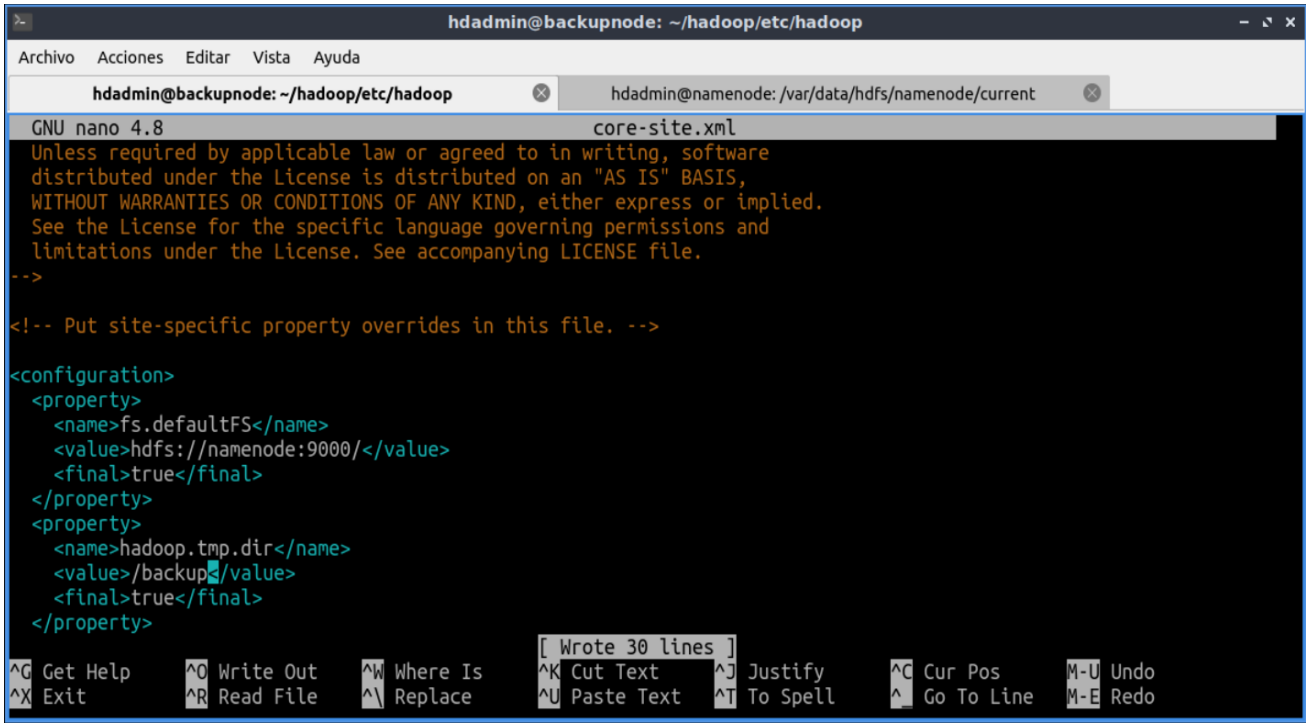
Procederemos a añadir al cluster configurado anteriormente un servidor de BackUp y un servidor TimeLineServer. Se irán explicando los pasos llevados a cabo a continuación.

1.1 SERVIDOR DE BACKUP

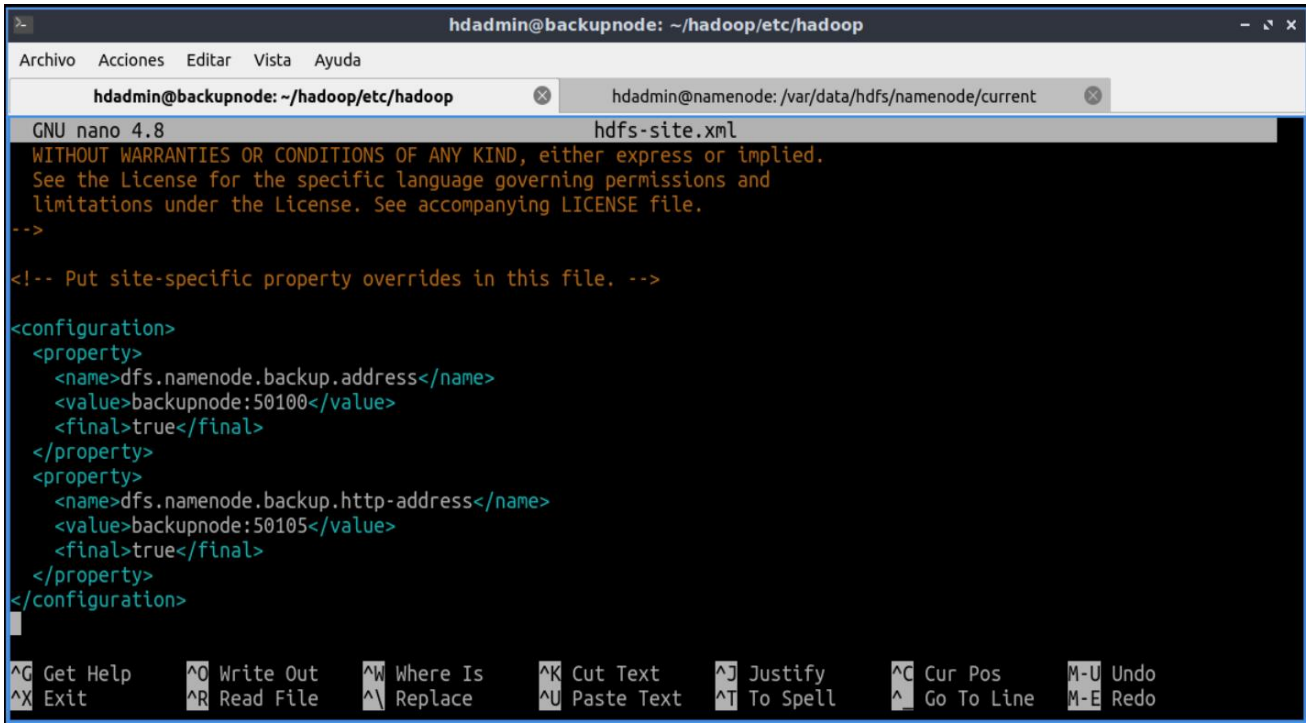
En primer lugar, iniciamos el clúster y en el NameNode obtenemos una captura en la que se muestren los ficheros del directorio de metadatos del NameNode. Para esto lo primero que hacemos es desplazarnos al directorio donde se encuentran estos ficheros mediante `cd /var/data/hdfs/namenode/current` y después comprobamos que ficheros hay mediante el comando `ls -all`. El resultado es el siguiente:



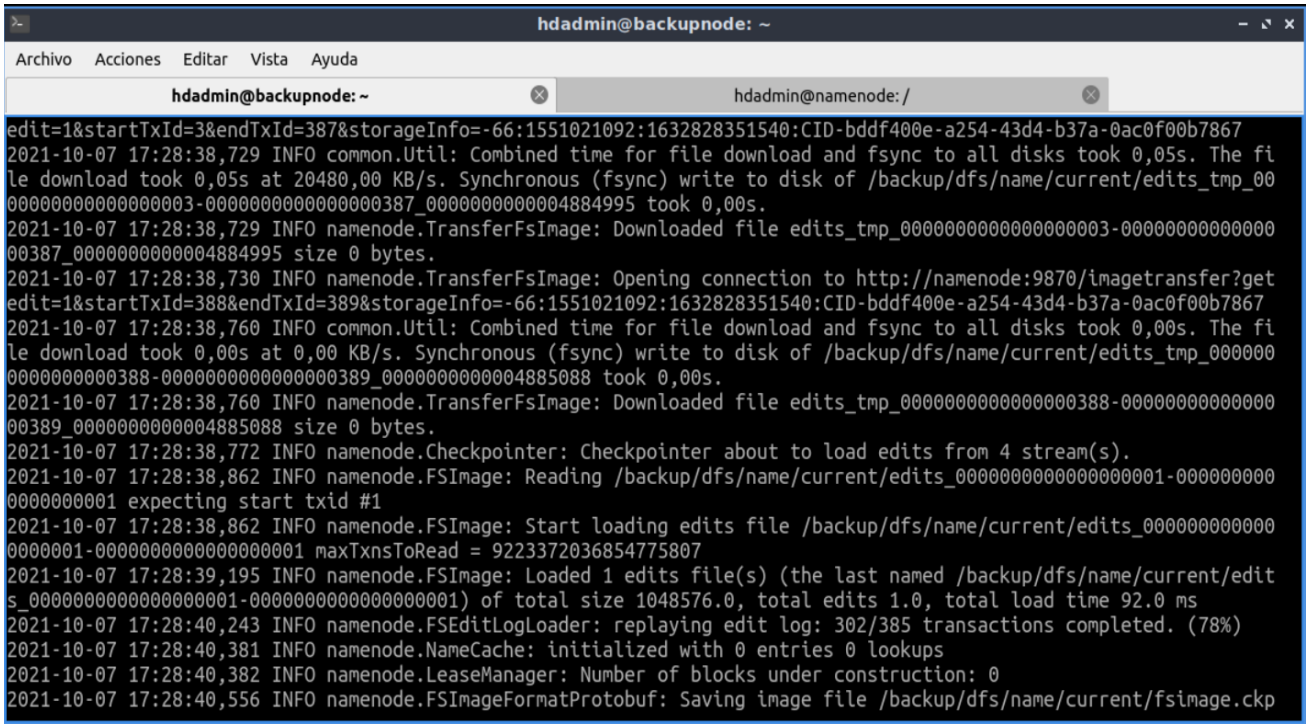
Para añadir el servidor de BackUp seguimos los pasos descritos en el guión de esta entrega. En primer lugar, con el cluster funcionando, ejecutamos el comando `docker container run -ti --name backupnode --network=hadoop-cluster --hostname backupnode --cpus=1 --memory=3072m --expose 50100 -p 50105:50105 hadoop-base /bin/bash` para iniciar un nuevo Docker a partir de la imagen hadoop-base.
Una vez que ya se ha creado este nuevo contenedor, de nombre backupnode, accedemos a él y creamos en el directorio raíz el directorio /backup mediante el comando `mkdir backup`, hacemos que el propietario sea hdadmin mediante el comando `chown hdadmin:hadoop /backup` y por último creamos dentro de este directorio la carpeta dfs/name (hacemos que el usuario hdadmin sea el propietario de estas carpetas también).
Después, cambiamos al usuario hdadmin (mediante `su – hdadmin`) y después modificamos el fichero core-site.xml para añadir las propiedades que se nos especifican en el enunciado:

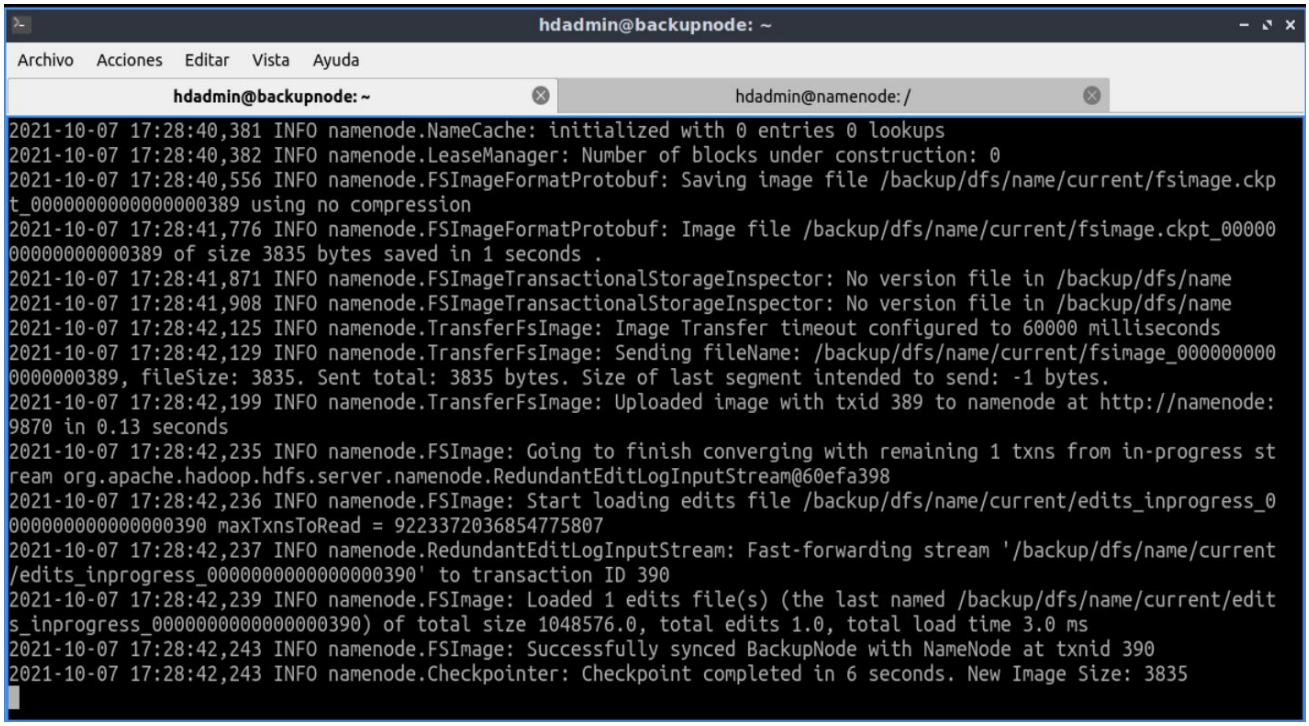


También añadimos las propiedades necesarias al fichero **hdfs-site.xml**:

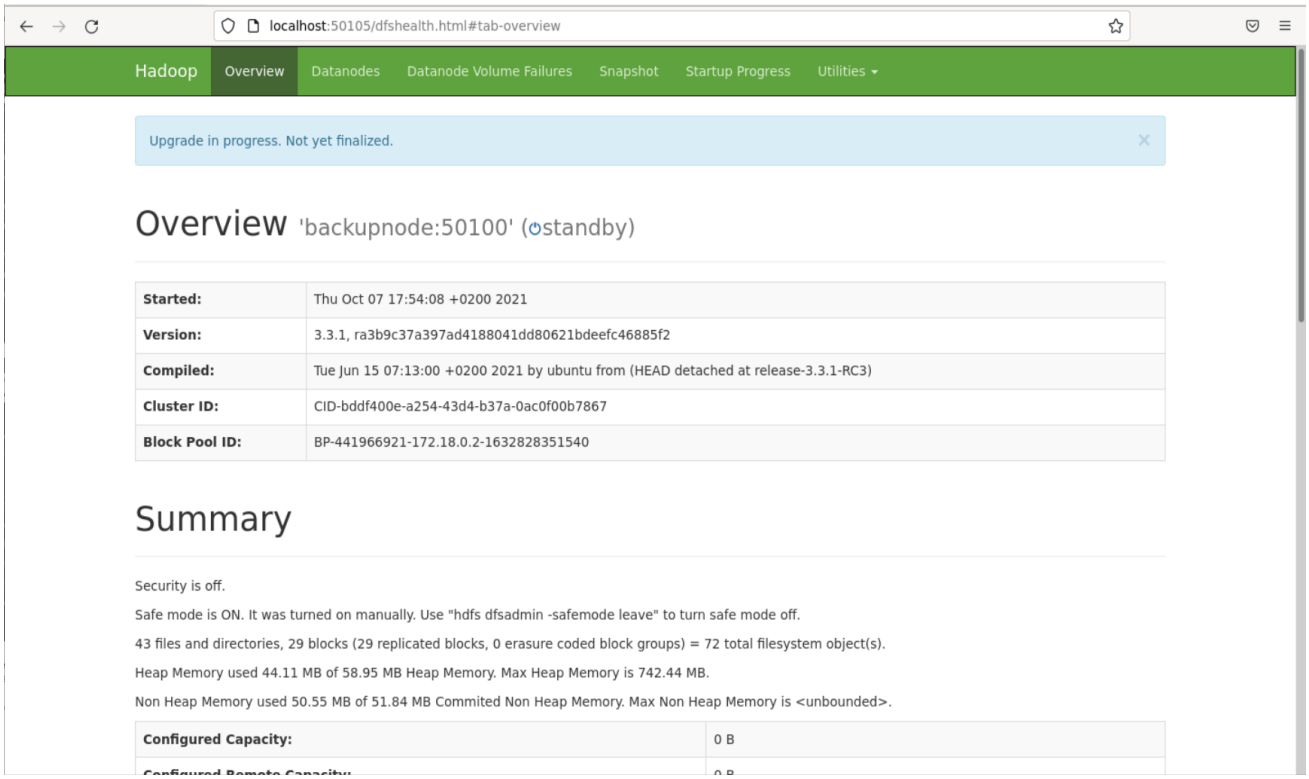


Ahora iniciamos el servidor de backup ejecutando el comando `hdfs namenode -backup`. Unos ejemplos de los mensajes que nos da el servicio de backup, destacando aquellos en los que se ve como se hace el checkpoint, al iniciarse son los siguientes:

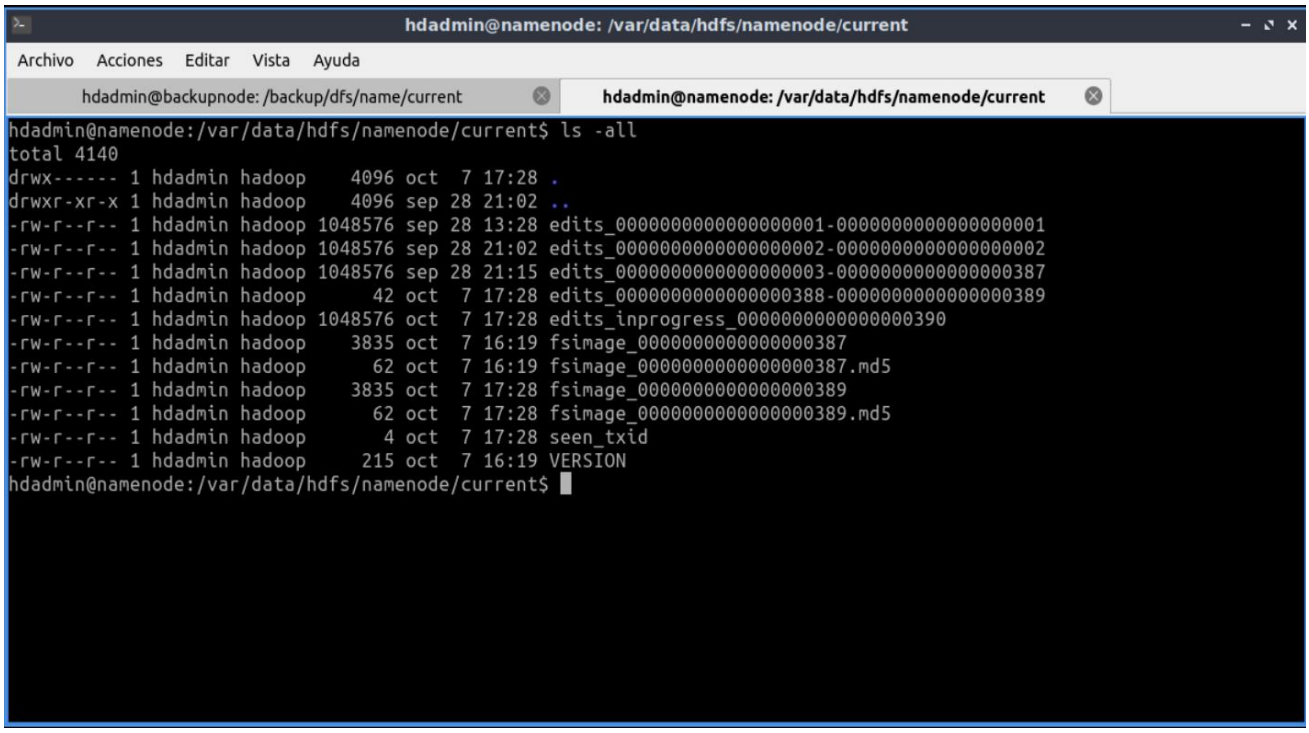




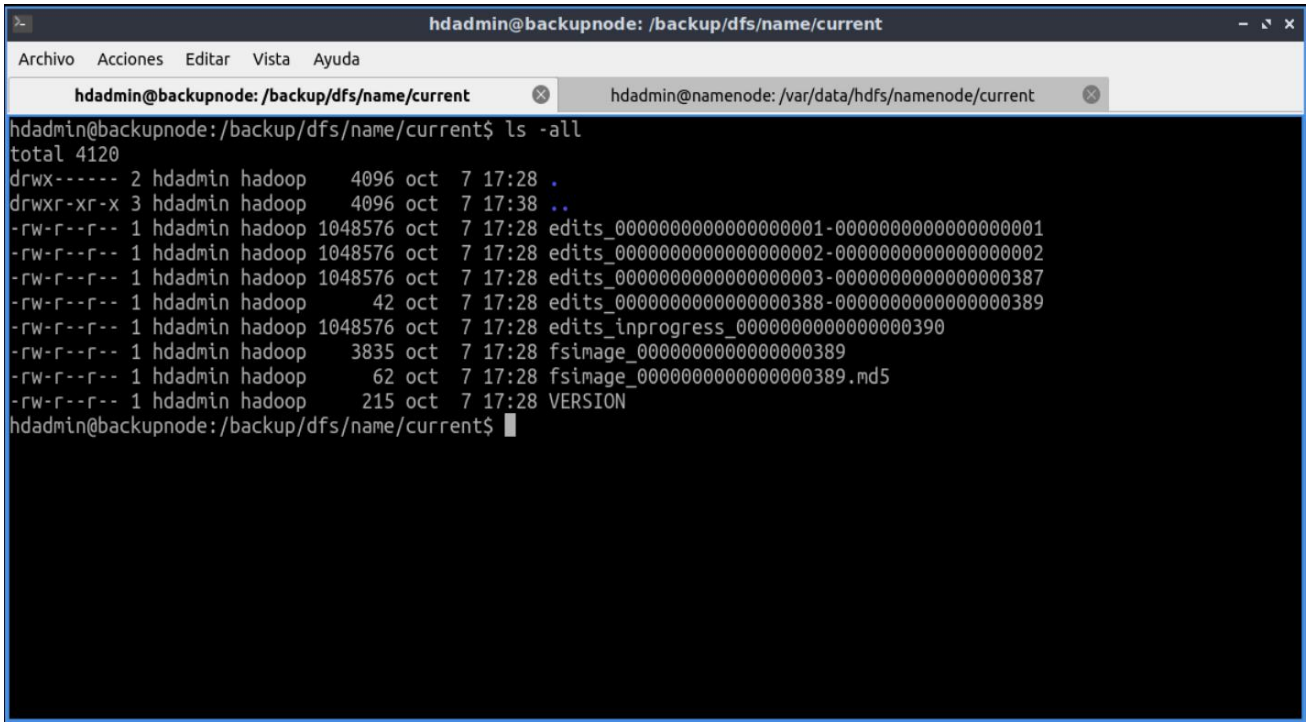
Como se puede ver en la última línea en los mensajes del servicio de backup, se nos dice que un checkpoint ha sido correctamente completado en 6 segundos.
A continuación, podemos una captura de pantalla de la interfaz web del nodo de backup:



Finalmente, compararemos el contenido del directorio del backup con el directorio con los metadatos del NameNode, antes y después de que el servicio de backup se haya completado. En primer lugar, vemos el contenido del directorio con los metadatos del NameNode después de haber ejecutado el servicio de backup (anteriormente ya se mostró la captura con los contenidos de este directorio antes de ejecutar el servicio de backup):



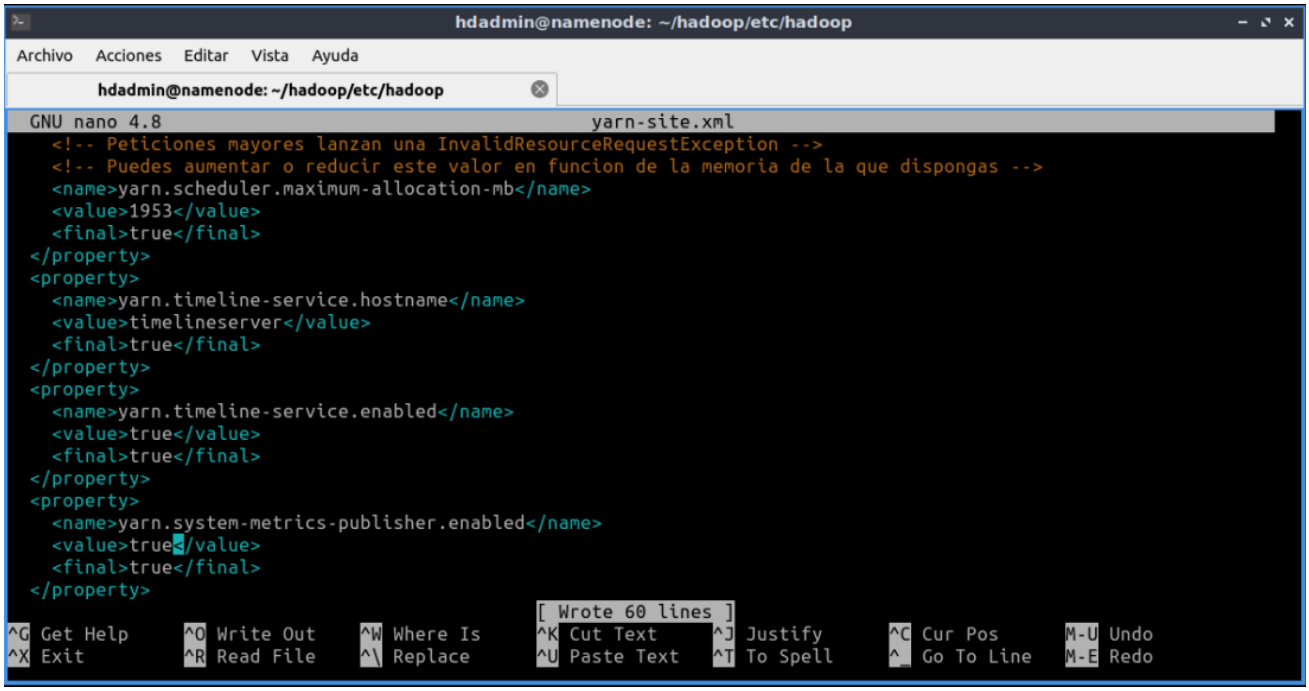
En la siguiente captura se puede ver el contenido del directorio de backup después de haber ejecutado el servicio de backup (antes de la ejecución este directorio se encontraba vacío):



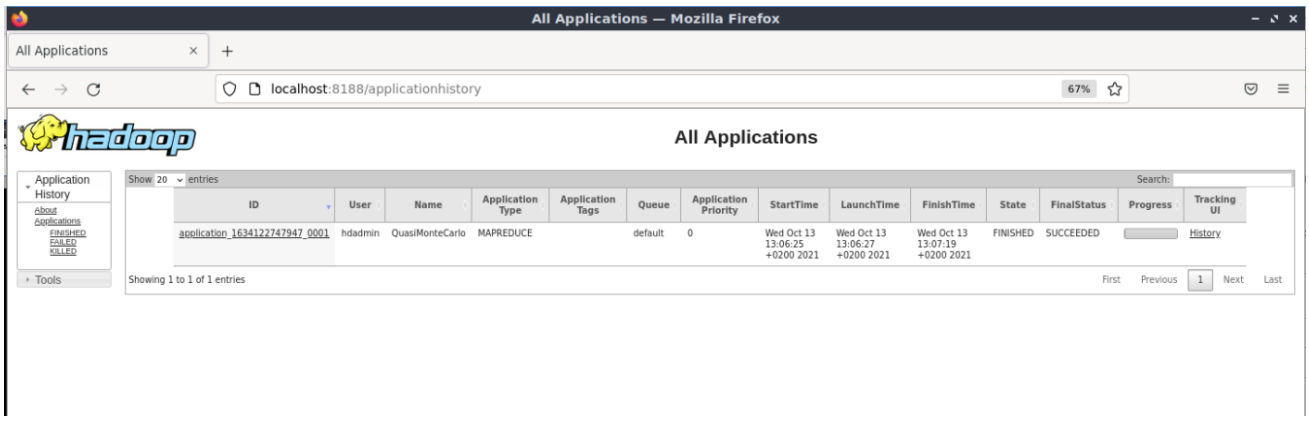
Para poder entender los contenidos de estos directorios es mejor primero dejar claro que los ficheros cuyo nombre empieza por ‘fsimage’ son los ficheros que contienen la estructura de directorios completa (namespace) de HDFS con información sobre la localización de cada fichero. Los ficheros cuyo nombre empieza por ‘edits’ tienen los cambios realizados al HDFS desde que la última FsImage fue creada. Por lo que con la última FsImage y los ficheros EditLogs podemos restaurar el estado del HDFS en caso de fallo. Entonces, en las capturas mostrada anteriormente podemos ver como al ejecutar el servicio de backup se añade un fichero FsImage en el directorio del NameNode y este mismo fichero se almacena en el directorio de backup, además de los ficheros EditLogs que están presentes en ambos directorios. En el directorio del NameNode se almacenan dos ficheros FsImage mientras que en el directorio del backup sólo se almacena el que fue creado después de ejecutar el servicio de backup.

1.2 SERVIDOR TIMELINESERVER

En primer lugar, accedemos al NameNode y como usuario hdadmin detenemos el servicio ResourceManager mediante el comando yarn –daemon stop resourcemanager. Ahora editamos el fichero yarn-site.xml para añadir las siguientes propiedades:



Reiniciamos el servicio ResourceManager mediante el comando yarn –daemon start resourcemanager. Utilizamos el comando docker container run -ti --name timelineserver --network=hadoop-cluster --hostname timelineserver --cpus=1 --memory=3072m --expose 10200 -p 8188:8188 hadoop-base /bin/bash para iniciar un nuevo Docker en el que levantaremos el servicio TimeLineServer. En este nuevo Docker levantamos el servicio TimeLineServer mediante yarn –daemon start timelineserve (como usuario hdadmin). Desde el NameNode ejecutamos la aplicación del cálculo de pi con YARN (como usuario hdadmin).De esta manera, el servidor web del TimeLineServer recoge la información de la ejecución de esta tarea, como se puede ver en la siguiente captura:

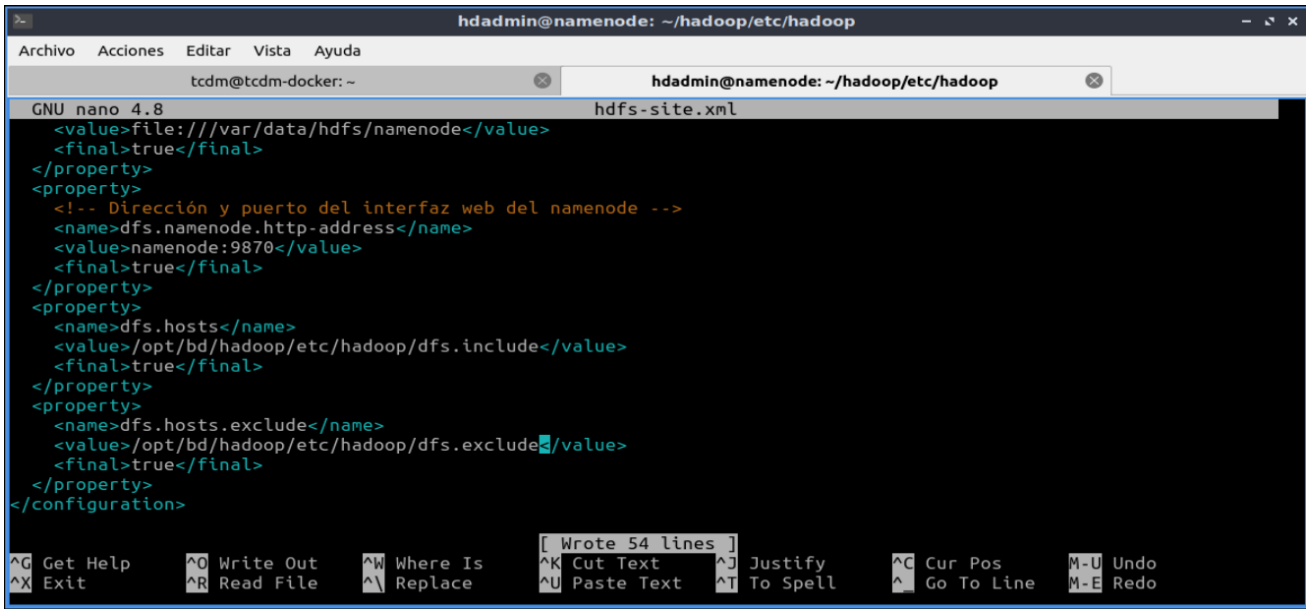


2 AÑADIR UN NUEVO DATANODE/NODEMANAGER

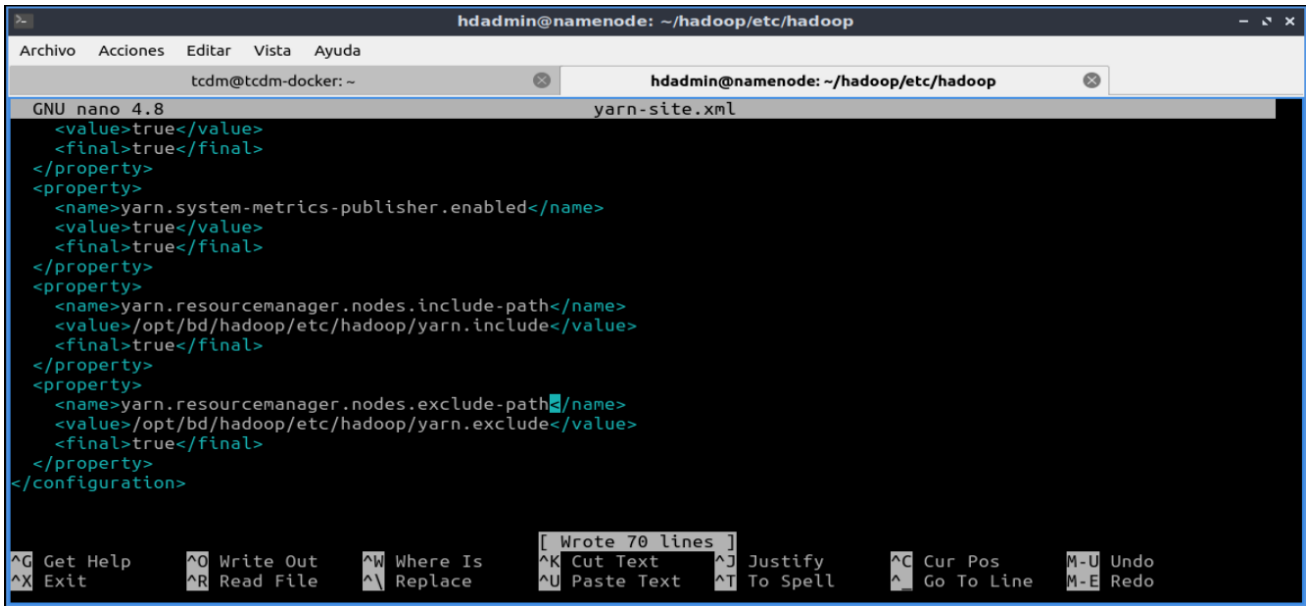
2.1 CREACIÓN DE FICHEROS DE NODOS INCLUIDOS Y EXCLUIDOS

lo que haremos es crear los ficheros de incluidos y de excluidos. Para ello primero detenemos los demonios del NameNode y del ResourceManager. Después creamos los cuatro ficheros que s indican en el enunciado. En los ficheros **dfs.include** y **yarn.include** incluimos los cuatro datanodes.

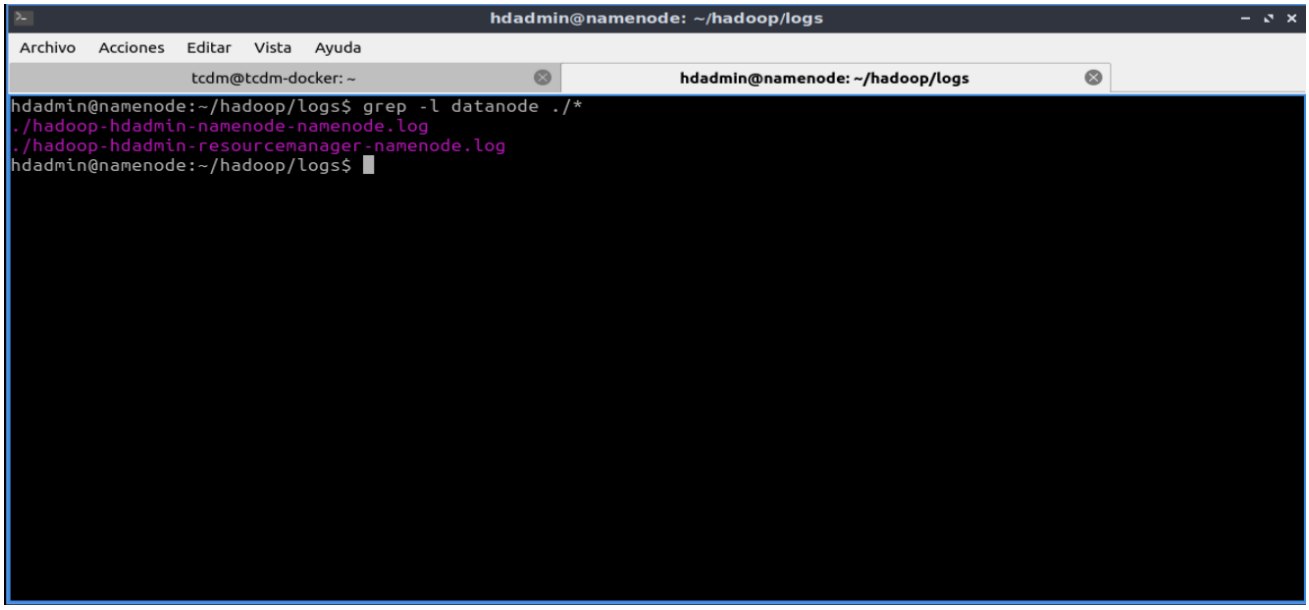
A continuación, editamos el fichero **hdfs.site.xml** para añadir las siguientes propiedades:



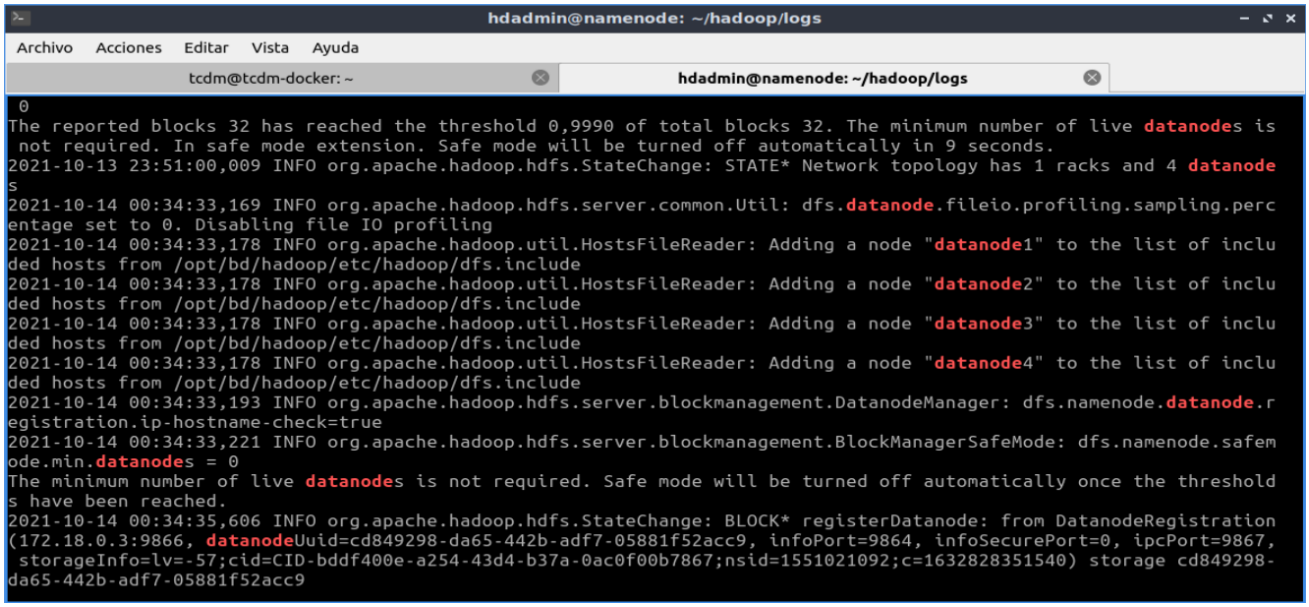
En el fichero **yarn-site.xml** añadimos otras dos propiedades:

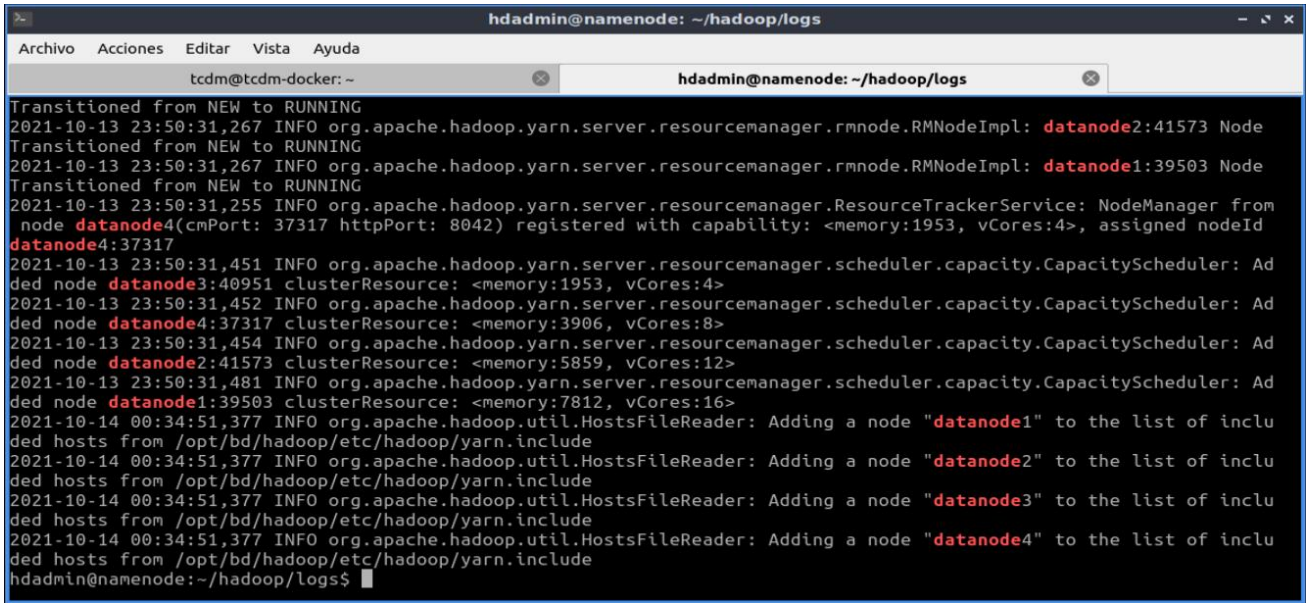


Para terminar, reiniciamos los demonios. Mediante el comando `grep -l datanode ./*` obtenemos los ficheros en el directorio de logs en los que aparezcan nombrados los datanodes. El resultado de este comando es el siguiente:



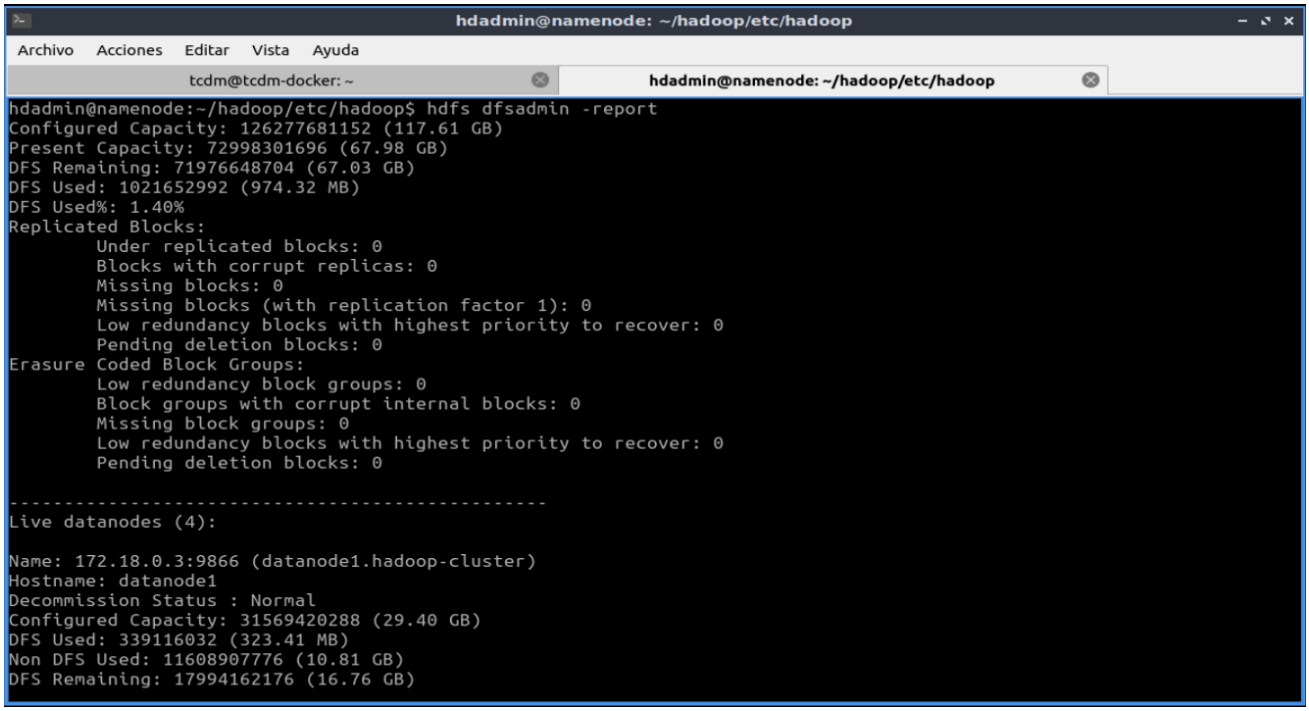
Por lo tanto, consultando en estos dos ficheros podemos ver como se han incluido al HDFS y al YARN los nodos datanode{1..4}:



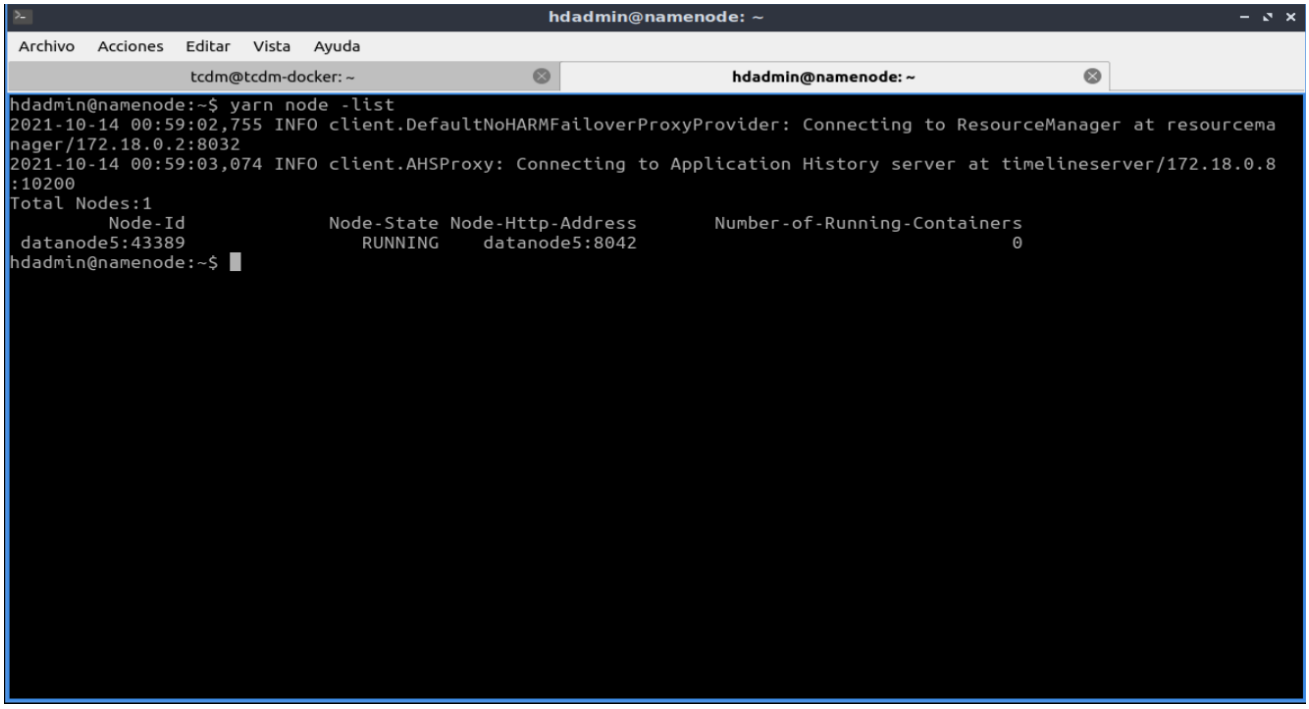


2.2 AÑADIR UN DATANODE/NODEMANAGER

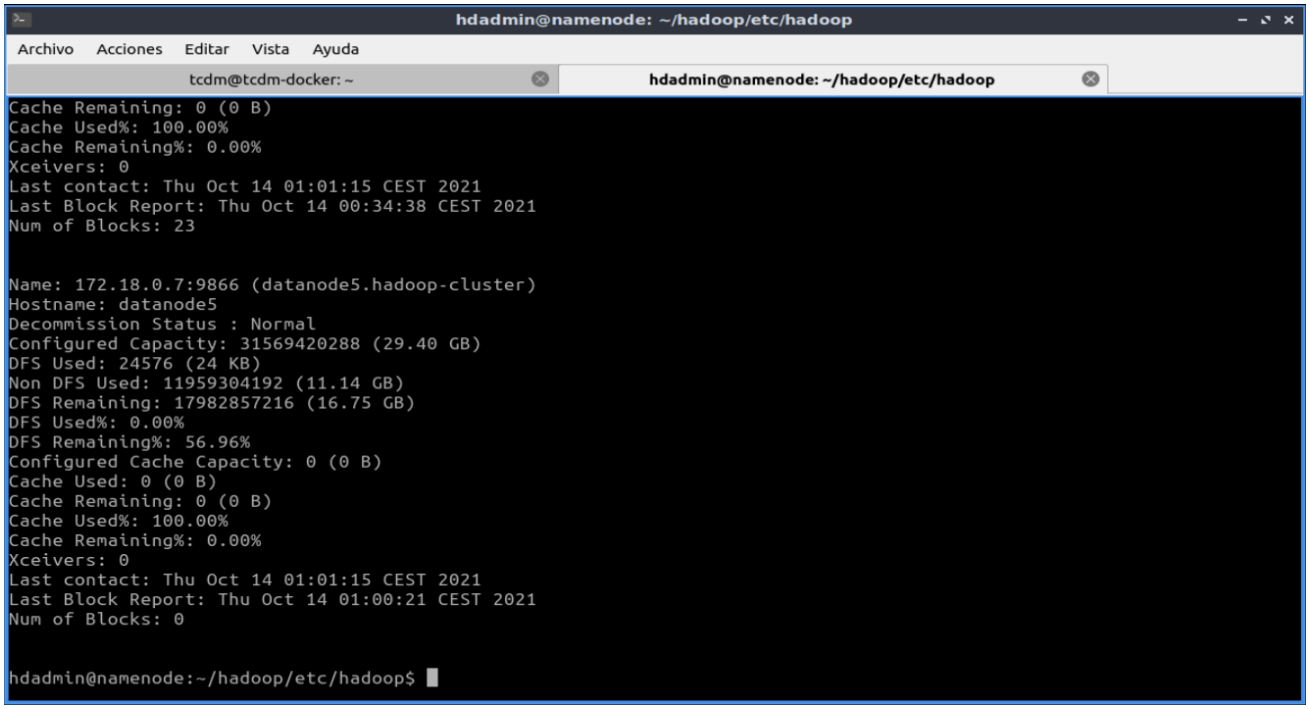
Después de haber añadido el datanode5 al fichero **dfs.include** ejecutamos el comando `hdfs dfsadmin -report` para comprobar como el nuevo contenedor no se ha añadido al HDFS. Como se muestra en la siguiente captura solo hay datanodes en activo:



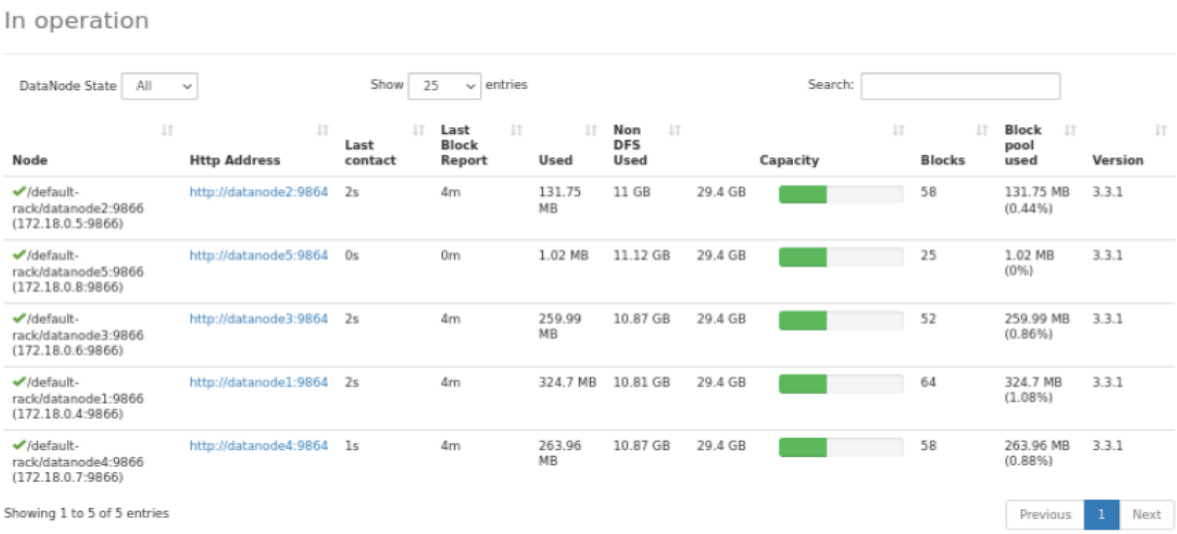
Sin embargo, si ejecutamos el comando `yarn node -list` vemos como sí que se ha añadido al YARN:



Tras añadir el datanode5 al fichero **dfs.include** y actualizar el NameNode vemos como ahora sí que se ha añadido al HDFS, tal y como se muestra en la siguiente captura donde podemos ver la información del datanode5 añadido:



Finalmente, forzamos el balanceo del clúster ejecutando el comando hdfs balancer. Antes de ejecutar el balanceo el nodo datanode5 tiene 25 bloques:



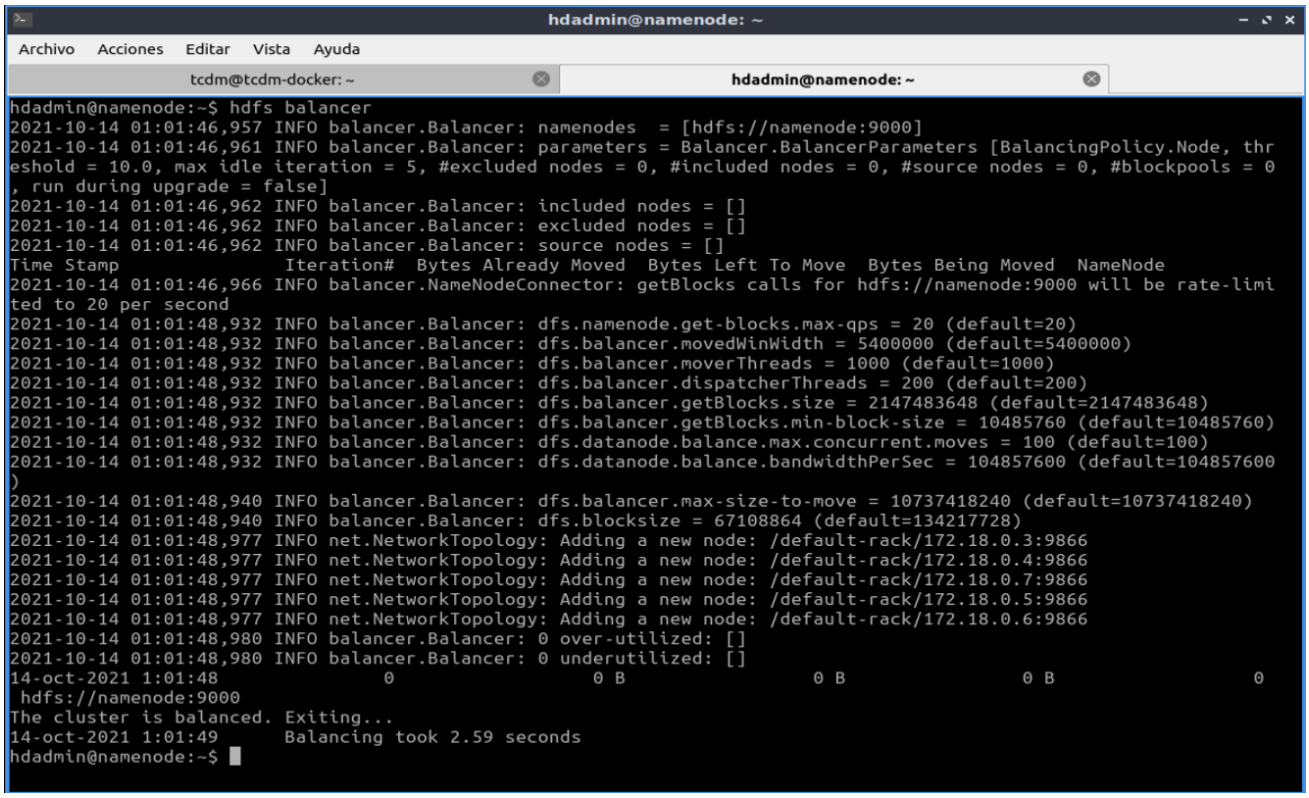
In operation

DataNode State: All | Show 25 entries | Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
/default-rack/datanode2:9866 (172.18.0.5:9866)	http://datanode2:9864	2s	4m	131.75 MB	11 GB	29.4 GB	58	131.75 MB (0.44%)	3.3.1
/default-rack/datanode5:9866 (172.18.0.8:9866)	http://datanode5:9864	0s	0m	1.02 MB	11.12 GB	29.4 GB	25	1.02 MB (0%)	3.3.1
/default-rack/datanode3:9866 (172.18.0.6:9866)	http://datanode3:9864	2s	4m	259.99 MB	10.87 GB	29.4 GB	52	259.99 MB (0.86%)	3.3.1
/default-rack/datanode1:9866 (172.18.0.4:9866)	http://datanode1:9864	2s	4m	324.7 MB	10.81 GB	29.4 GB	64	324.7 MB (1.08%)	3.3.1
/default-rack/datanode4:9866 (172.18.0.7:9866)	http://datanode4:9864	1s	4m	263.96 MB	10.87 GB	29.4 GB	58	263.96 MB (0.88%)	3.3.1

Showing 1 to 5 of 5 entries | Previous 1 Next

La salida el balanceador de carga es la siguiente:

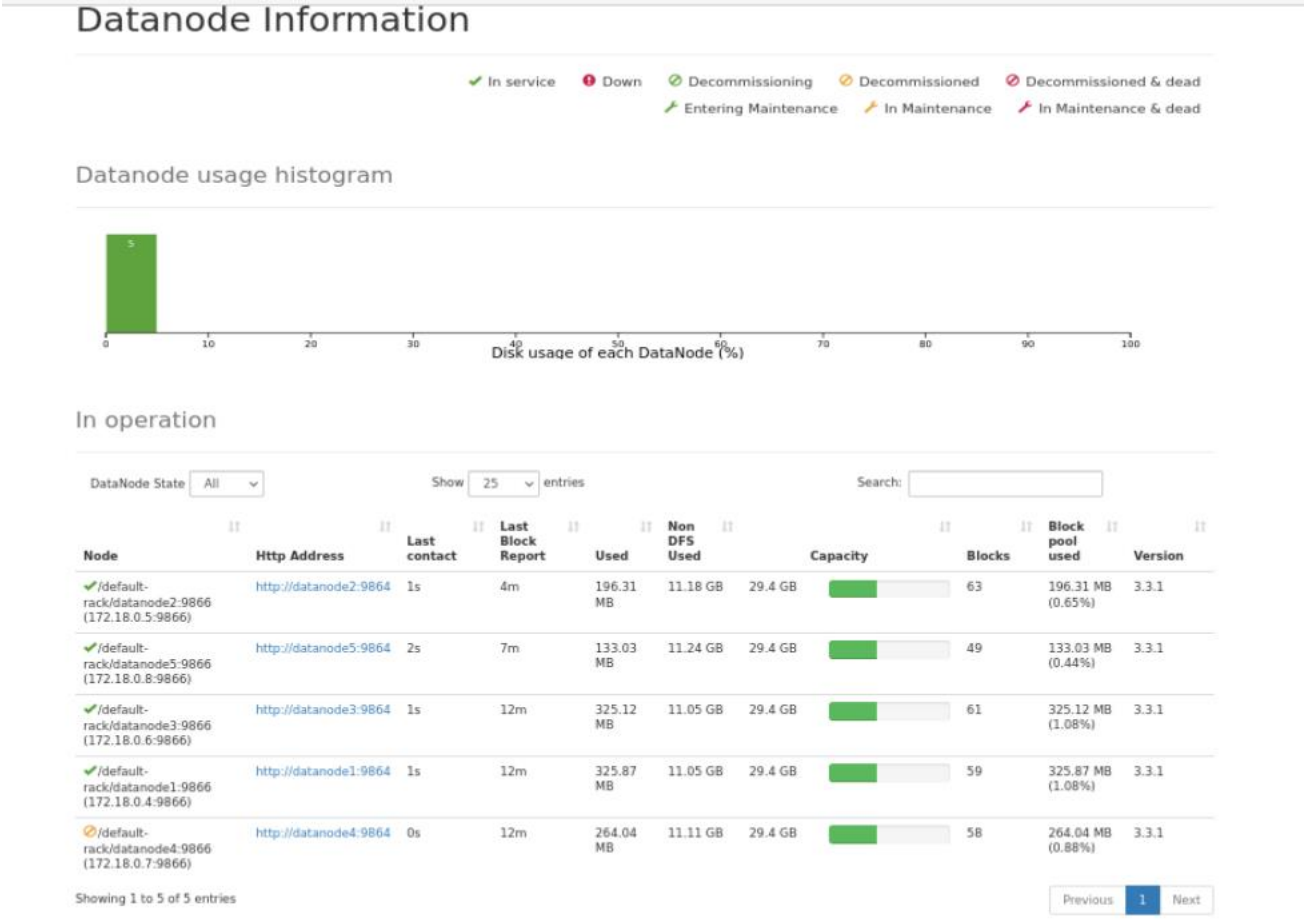


```
hdadmin@namenode: ~
Archivo  Acciones  Editar  Vista  Ayuda
tcdm@tcdm-docker: ~        hdadmin@namenode: ~
hdadmin@namenode:~$ hdfs balancer
2021-10-14 01:01:46,957 INFO balancer.Balancer: namenodes  = [hdfs://namenode:9000]
2021-10-14 01:01:46,961 INFO balancer.Balancer: parameters = Balancer.BalancerParameters [BalancingPolicy.Node, thr
eshold = 10.0, max idle iteration = 5, #excluded nodes = 0, #included nodes = 0, #source nodes = 0, #blockpools = 0
, run during upgrade = false]
2021-10-14 01:01:46,962 INFO balancer.Balancer: included nodes = []
2021-10-14 01:01:46,962 INFO balancer.Balancer: excluded nodes = []
2021-10-14 01:01:46,962 INFO balancer.Balancer: source nodes = []
Time Stamp               Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Moved  NameNode
2021-10-14 01:01:46,966 INFO balancer.NameNodeConnector: getBlocks calls for hdfs://namenode:9000 will be rate-limi
ted to 20 per second
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.namenode.get-blocks.max-qps = 20 (default=20)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.balancer.movedWinWidth = 5400000 (default=5400000)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.balancer.moverThreads = 1000 (default=1000)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.balancer.dispatcherThreads = 200 (default=200)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.balancer.getBlocks.size = 2147483648 (default=2147483648)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.balancer.getBlocks.min-block-size = 10485760 (default=10485760)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.datanode.balance.max.concurrent.moves = 100 (default=100)
2021-10-14 01:01:48,932 INFO balancer.Balancer: dfs.datanode.balance.bandwidthPerSec = 104857600 (default=104857600
)
2021-10-14 01:01:48,940 INFO balancer.Balancer: dfs.balancer.max-size-to-move = 10737418240 (default=10737418240)
2021-10-14 01:01:48,940 INFO balancer.Balancer: dfs.blocksize = 67108864 (default=134217728)
2021-10-14 01:01:48,977 INFO net.NetworkTopology: Adding a new node: /default-rack/172.18.0.3:9866
2021-10-14 01:01:48,977 INFO net.NetworkTopology: Adding a new node: /default-rack/172.18.0.4:9866
2021-10-14 01:01:48,977 INFO net.NetworkTopology: Adding a new node: /default-rack/172.18.0.7:9866
2021-10-14 01:01:48,977 INFO net.NetworkTopology: Adding a new node: /default-rack/172.18.0.5:9866
2021-10-14 01:01:48,977 INFO net.NetworkTopology: Adding a new node: /default-rack/172.18.0.6:9866
2021-10-14 01:01:48,980 INFO balancer.Balancer: 0 over-utilized: []
2021-10-14 01:01:48,980 INFO balancer.Balancer: 0 underutilized: []
14-oct-2021 1:01:48              0                 0 B                 0 B               0 B                   0
 hdfs://namenode:9000
The cluster is balanced. Exiting...
14-oct-2021 1:01:49      Balancing took 2.59 seconds
hdadmin@namenode:~$
```

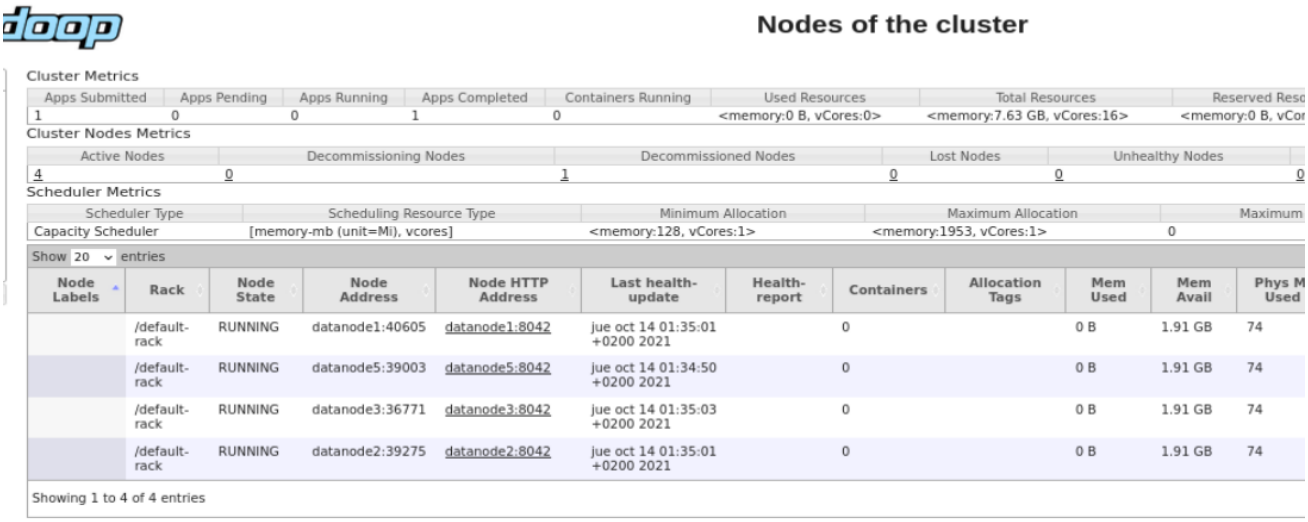
Como se puede ver se han movido 0 bytes de datos. Supongo que al estar todos los nodos muy poco llenos el balanceador considera que no es necesario enviar datos al datanode5. Si los nodos datanode{1..4} estuviesen muy llenos, sí que habrían un desbalanceo considerable en el clúster y esta operación hubiese movido algunos datos al datanode5.

3 RETIRAR UN DATANODE/NODEMANAGER

Como podemos ver, tras incluir el nodo datanode4 en los ficheros **dfs.exclude** y **yarn.exclude** y actualizar el NameNode y el ResourceManager podemos ver como el nodo aparece como decomisionado en la interfaz web del HDFS:

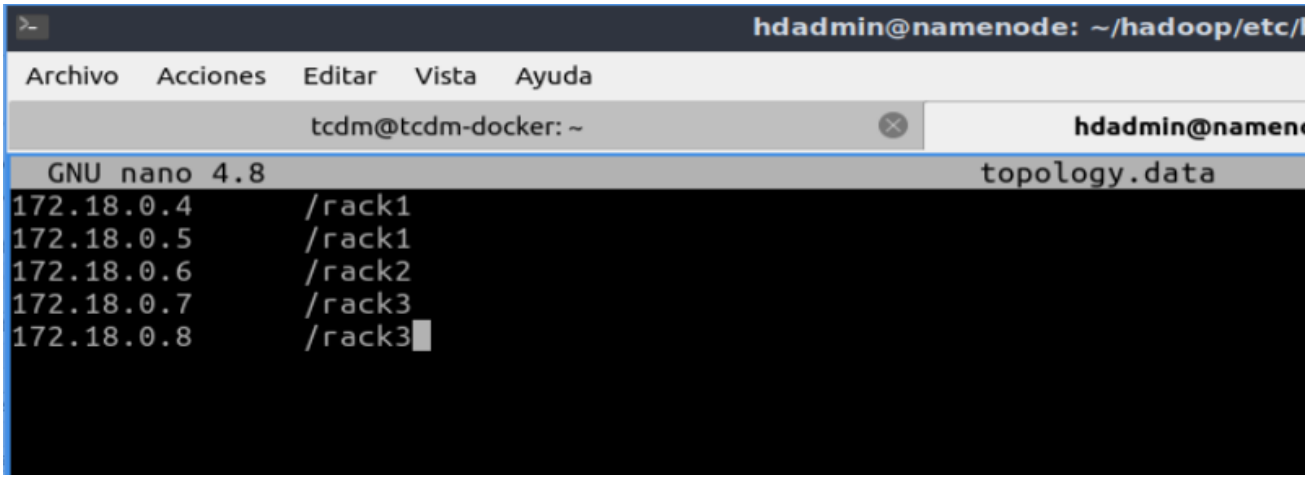


También podemos ver como en la interfaz web del YARN ya no aparece entre los nodos disponibles:

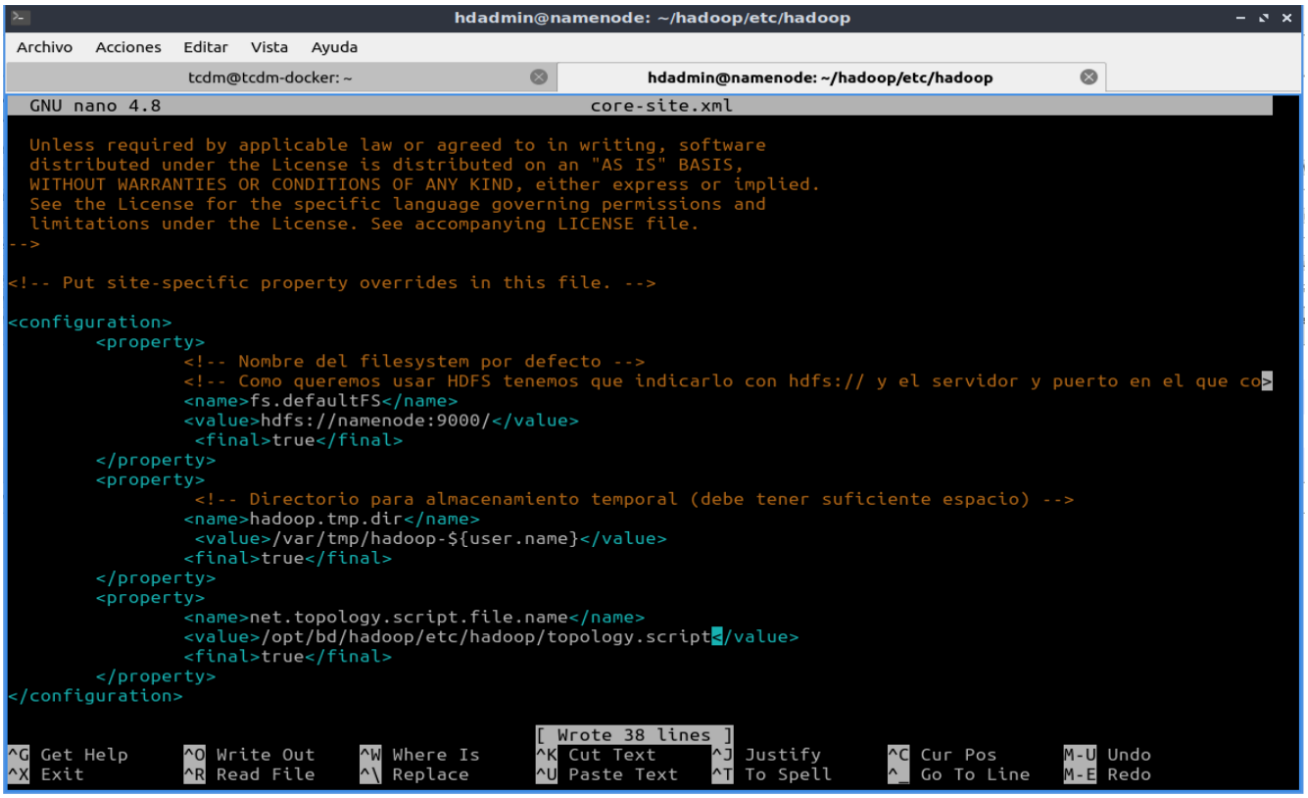


4 RACK AWARENESS

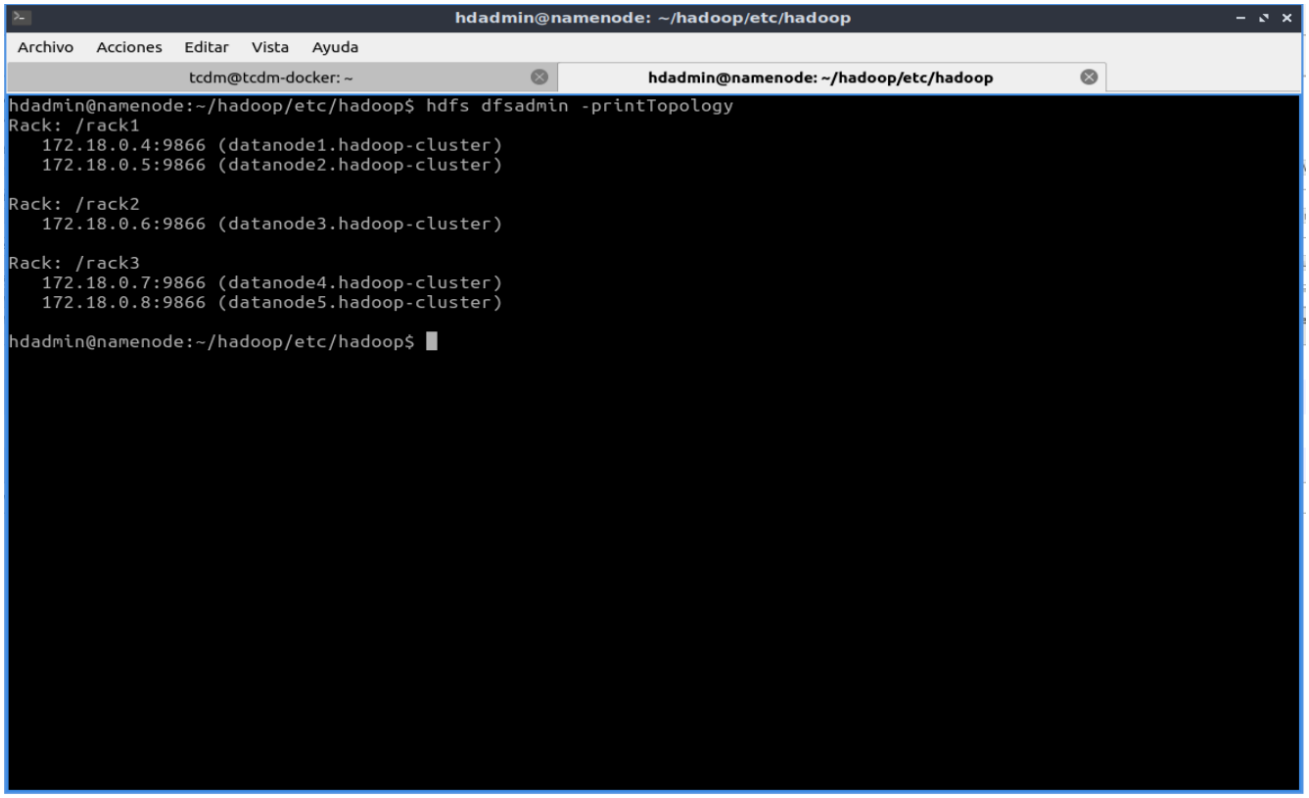
Para terminar, llevamos a cabo los pasos indicados en el enunciado. El contenido del fichero **topology.data** es el siguiente:



La propiedad mediante la cual indicamos el path al script creado:



Y por último, la salida del comando `hdfs dfsadmin -printTopology` en la que se ve que la distribución por racks se llevó a cabo de manera correcta:


hdadmin@namenode: ~/hadoop/etc/hadoop
Archivo Acciones Editar Vista Ayuda
tcdm@tcdm-docker: ~
hdadmin@namenode: ~/hadoop/etc/hadoop
hdadmin@namenode:~/hadoop/etc/hadoop$ hdfs dfsadmin -printTopology
Rack: /rack1
172.18.0.4:9866 (datanode1.hadoop-cluster)
172.18.0.5:9866 (datanode2.hadoop-cluster)
Rack: /rack2
172.18.0.6:9866 (datanode3.hadoop-cluster)
Rack: /rack3
172.18.0.7:9866 (datanode4.hadoop-cluster)
172.18.0.8:9866 (datanode5.hadoop-cluster)
hdadmin@namenode:~/hadoop/etc/hadoop$