Práctica 2: Acceso y gestión de HDFS

En esta práctica veremos cómo acceder programáticamente y gestionar el HDFS.

Primera parte: ejecución del código `filesystem_cat` (ver transparencias tema 2)

1. Descarga el fichero adjunto `filesystem_cat.py`.
2. Súbelo al contenedor `namenode` en el directorio del usuario `luser`:

```
docker cp filessytem_cat.py namenode:/home/luser
```

1. En el `namenode`, como usuario `luser` (`su - luser`):
2. Para ejecutarlo, hay que hacer unos pasos iniciales:
   - Instalar el paquete `pyarrow`:

```
pip install pyarrow
```

   - Establecer correctamente la variable de entorno CLASSPATH:

```
export CLASSPATH=$(hdfs classpath --glob)
```

3. Comprueba que es capaz de mostrar algún fichero de texto (sin comprimir) que subáis a HDFS, como se hizo en la práctica 1 co los libros, por ejemplo:

```
python3 filesystem_cat.py hdfs://namenode:9000/fichero/a/mostrar
```

Segunda parte: código `copy_half_file` (ver transparencias tema 2)

1. Basándote en el código anterior y en la plantilla adjunta `copy_half_file.py`, escribe un programa que copie la mitad inferior del fichero en HDFS a otro fichero en otro directorio del HDFS
   - Usa `get_file_info()` para obtener la longitud del fichero y `seek()` para saltar a la mitad del mismo.

Tercera parte: Probar el comandos `hdfs dfsadmin`

1. En el NameNode, como usuario `hdadmin`, crea un directorio en HDFS y ponle una cuota de solo 4 ficheros. Comprueba cuántos ficheros puedes copiar a ese directorio. Explica a qué se debe este comportamiento.

Cuarta parte: Probar el comando `hdfs fsck`

1. En el NameNode (como usuario `hdadmin`) haz un chequeo de todo el HDFS, y comprueba si te da errores:

- Intenta determinar las causas de los posibles errores

1. Detén docker datanodes de forma brusca (sin detener los demonios, simplemente haciendo p.e. `docker container stop datanode2 datanode3`) hasta que te queden solo 2 datanodes levantados. Espera unos 10 minutos[1] y comprueba que el comando `hdfs dfsadmin -report` muestra que solo quedan 2 datanodes vivos.
2. Realiza de nuevo el chequeo de disco en el NameNode y comprueba la salida. ¿Cuántos bloques aparecen under-replicated?
3. Prueba a hacer un get del fichero `random_words.txt.bz2` para ver si se hace correctamente.
4. Sigue los pasos que viste en la práctica 1 para añadir un datanode nuevo (vacío). Comprueba, haciendo de nuevo el chequeo, que los datos se replican en el nuevo nodo hasta alcanzar el factor de replicación por defecto.

[1] El tiempo depende de los parámetros `dfs.namenode.stale.datanode.interval` y `dfs.namenode.heartbeat.recheck-interval`. El primero indica el tiempo sin detectar actividad del DataNode para que el NameNode lo considere en estado stale (por defecto, 30 segundos). Un nodo en estado stale tiene menor prioridad en lecturas y escrituras. El segundo parámetro indica el intervalo de chequeo en busca de DataNodes expirados (valor por defecto, 5 minutos). Un DataNode se pasa al estado Dead cuando el tiempo sin detectar actividad es superior a `dfs.namenode.stale.datanode.interval + 2 * dfs.namenode.heartbeat.recheck-interval`.

Entrega

1. Entregar el código desarrollado en la segunda parte, todos los ficheros `*.py`, y un ejemplo de su ejecución.
2. Un documento que muestre que se han ejecutado las tercera y cuarta partes. Mostrar capturas de pantalla, incluyendo una explicación y justificación de lo que aparece en las mismas.