

Objetivo de la práctica

Comprobar que Hadoop puede ejecutar un job MapReduce de ejemplo (el grep del hadoop-mapreduce-examples-*.jar) y que el job produce la salida esperada en /tmp/salida. Haremos 2 ejecuciones de prueba con dos expresiones distintas (una que devuelve 1 coincidencia tipo dfsadmin y otra que cuenta todas las letras a).

Entorno (tal y como se ve en las capturas)

- Hadoop descomprimido en /home/manuu/Desktop/hadoop
- Jar de ejemplos en
/home/manuu/Desktop/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.6.jar
- Trabajamos en local (desde la carpeta .../mapreduce) y usamos rutas /tmp/entrada y /tmp/salida tal como en tus capturas.

1) Ir al directorio de Hadoop (igual que en la primera captura):

```
(root㉿kali)-[/home/.../hadoop/share/hadoop/mapreduce]
# cd
└─(root㉿kali)-[~]
  └─# cd /home/manuu/Desktop/hadoop/
    └─(root㉿kali)-[/home/manuu/Desktop/hadoop]
      # ls
      LICENSE-binary  NOTICE.txt  etc          jdk          licenses-binary  share
      LICENSE.txt     README.txt  hadoop-3.3.6.tar.gz  lib          openjdk-8u44-linux-x64.tar.gz
      NOTICE-binary   bin        include       libexec      sbin
      └─(root㉿kali)-[/home/manuu/Desktop/hadoop]
        └─# cd share
          └─(root㉿kali)-[/home/manuu/Desktop/hadoop/share]
            # ls
            doc  hadoop
            └─(root㉿kali)-[/home/.../Desktop/hadoop/share/hadoop]
              # ls
              client  common  hdfs  mapreduce  tools  yarn
              └─(root㉿kali)-[/home/.../Desktop/hadoop/share/hadoop]
                # cd mapreduce
                  └─(root㉿kali)-[/home/.../hadoop/share/hadoop/mapreduce]
                    # ls
                    hadoop-mapreduce-client-app-3.3.6.jar           hadoop-mapreduce-client-nativetask-3.3.6.jar
                    hadoop-mapreduce-client-common-3.3.6.jar         hadoop-mapreduce-client-shuffle-3.3.6.jar
                    hadoop-mapreduce-client-core-3.3.6.jar          hadoop-mapreduce-client-uploader-3.3.6.jar
                    hadoop-mapreduce-client-hs-3.3.6.jar           hadoop-mapreduce-examples-3.3.6.jar
                    hadoop-mapreduce-client-hs-plugins-3.3.6.jar      jdiff
                    hadoop-mapreduce-client-jobclient-3.3.6-tests.jar lib-examples
                    hadoop-mapreduce-client-jobclient-3.3.6.jar      sources
                    └─(root㉿kali)-[/home/.../hadoop/share/hadoop/mapreduce]
                      # jar tvf hadoop-mapreduce-examples-3.3.6.jar
                        0 Sun Jun 18 08:51:36 CEST 2023 META-INF/
                        186 Sun Jun 18 08:51:34 CEST 2023 META-INF/MANIFEST.MF
                        0 Sun Jun 18 08:51:36 CEST 2023 org/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/examples/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/examples/dancing/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/examples/terasort/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/examples/pi/
                        0 Sun Jun 18 08:51:36 CEST 2023 org/apache/hadoop/examples/pi/math/
```

¿Qué buscar en la salida para dar OK?

- El job terminó SUCCEEDED (en la salida del comando hadoop jar normalmente aparece Job Finished: ... , state: SUCCEEDED o en YARN).
- En /tmp/salida hay fichero(s) part-r-00000 (o varios) y al catarlos se ve texto con la clave y un número (por ejemplo DFS\t1).
- Al menos **una** coincidencia (si el patrón buscado existe en los archivos de entrada) — puedes indicar a los alumnos qué patrón deben buscar para que haya coincidencias concretas (por ejemplo: A a mayúscula en muchos XML dará matches).

2) Preparar directorio de entrada local (captura 2) y primera ejecución (buscar líneas que contengan dfs seguido de letras — en la captura usan `dfs[a-z.]`):

```
(root@kali)-[~/home/.../hadoop/share/hadoop/mapreduce]
# mkdir /tmp/entrada

(root@kali)-[~/home/.../hadoop/share/hadoop/mapreduce]
# cp /home/manuu/Desktop/hadoop/etc/hadoop/*.xml /tmp/entrada

(root@kali)-[~/home/.../hadoop/share/hadoop/mapreduce]
# ls /tmp/entrada
capacity-scheduler.xml  hadoop-policy.xml  hdfs-site.xml  kms-acls.xml  mapred-site.xml
core-site.xml           hdfs-rbf-site.xml  httpfs-site.xml  kms-site.xml  yarn-site.xml

[root@kali]-[~/home/.../hadoop/share/hadoop/mapreduce]
# hadoop jar hadoop-mapreduce-examples-3.3.6.jar grep /tmp/entrada /tmp/salida 'dfs[a-z.]'
2025-10-03 14:11:05,089 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-10-03 14:11:05,185 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-10-03 14:11:05,186 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-10-03 14:11:05,377 INFO input.FileInputFormat: Total input files to process : 10
```

Observa la salida: verás logs de metrics, mappers y reducers en ejecución (como en tus capturas).

3) Tras finalizar, mira /tmp/salida localmente (captura 3):

```
(root@kali)-[~/home/.../hadoop/share/hadoop/mapreduce]
# cd /tmp/salida/

(root@kali)-[/tmp/salida]
# ls
_SUCCESS  part-r-00000

[root@kali]-[/tmp/salida]
# cat part-r-00000
1          dfsadmin
```

4) Borrar salida anterior y ejecutar la segunda prueba (en la captura 5 ejecutas con la expresión 'a'):

```
(root㉿kali)-[~/tmp/salida]
# cd /home/manuu/Desktop/hadoop/share/hadoop/mapreduce/
[root@kali ~]# hadoop jar hadoop-mapreduce-examples-3.3.6.jar grep /tmp/entrada /tmp/salida 'a'
2025-10-03 14:15:44,934 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-10-03 14:15:44,968 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-10-03 14:15:44,968 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-10-03 14:15:45,062 INFO input.FileInputFormat: Total input files to process : 10
```

```
(root㉿kali)-[~/home/.../hadoop/share/hadoop/mapreduce]
# cat /tmp/salida/part-r-00000
1560    a
```

Interpretación: la ejecución con patrón 'a' cuenta todas las ocurrencias de la letra a en los XML -> el total mostrado es 1560 y la clave es a.

Notas y cosas a tener en cuenta

- Las ejecuciones en tus capturas se hacen **desde** el directorio .../mapreduce y llaman hadoop jar hadoop-mapreduce-examples-3.3.6.jar ... (no hace falta usar la ruta completa del jar si estás en esa carpeta).
- La primera expresión ('dfs[a-z.]+') es **case-sensitive** y por eso encontró dfsadmin (todo en minúsculas).
- La segunda prueba con 'a' cuenta todas las letras a en los XML (muchas apariciones) y dio 1560.
- Si un alumno no obtiene part-r-00000 es porque el job falló o la salida ya existía: deben borrar /tmp/salida antes de ejecutar (rm -rf /tmp/salida), como se muestra en la práctica.
- Si están usando HDFS en vez de modo local, la práctica es análoga pero usando hdfs dfs -put para subir los XML y consultando hdfs dfs -cat /tmp/salida/part-* — **pero** las capturas que enviaste usan el filesystem local, así la práctica queda tal cual.

Criterios de evaluación (automático/manual)

- **Check 1 (Setup):** hadoop-mapreduce-examples-3.3.6.jar existe → OK.
- **Check 2 (Entrada):** /tmp/entrada contiene archivos XML → OK.
- **Check 3 (Job 1):** after hadoop jar ... 'dfs[a-z.]+' → /tmp/salida/part-r-00000 exists and contains a line with dfsadmin and count 1.
- **Check 4 (Job 2):** after hadoop jar ... 'a' → /tmp/salida/part-r-00000 exists and contains 1560 a.
(Puntos: cada check vale X; si fail, incluir salida de error)