Hadoop. Almacenamiento y procesamiento MapReduce

Contenido

[Recursos útiles 1](#_Toc204155186)

[Objetivo de la práctica 2](#_Toc204155187)

[Instalación Hadoop 2](#_Toc204155188)

[Proyecto Github de referencia 2](#_Toc204155189)

[Práctica contar palabras con MapReduce 4](#_Toc204155190)

[1. Copiar ficheros de libros a namenode 4](#_Toc204155191)

[2. Conectamos al contenedor namenode 4](#_Toc204155192)

[3. Subir archivos desde el contenedor a HDFS 4](#_Toc204155193)

[4. Ejecutamos JAR de contar palabras 5](#_Toc204155194)

[5. Verificando el resultado con un fichero trivial 6](#_Toc204155195)

[Hadoop y Python (pendiente) 7](#_Toc204155196)

[Problemas encontrados: 7](#_Toc204155197)

[Reto para el alumno: 7](#_Toc204155198)

# Recursos útiles

Recuerde el directorio de recursos compartidos disponible en:

<https://nascorformacion0-my.sharepoint.com/:f:/g/personal/juan_pinuela_docente_nascorformacion_com/EtGSBITnqFpJp6-XpG6REOQBFycP_G9zgcB89jxPymjV2Q?e=IBIEER>

Todo los archivos utilizados en la práctica se encuentran disponibles en la subcarpeta “hadoop”, incluyendo los archivos de un repositorio Github habitualmente usado en el contexto de Hadoop

# Objetivo de la práctica

Familiarizar al alumno con el ecosistema Hadoop, desde su despliegue en contenedores Docker hasta la ejecución de tareas MapReduce sobre un conjunto de datos reales. La práctica reproduce un flujo básico de procesamiento distribuido, permitiendo al estudiante experimentar directamente con los comandos HDFS, la interfaz web del clúster y la ejecución de un job MapReduce clásico con Java.

# Instalación Hadoop

## Proyecto Github de referencia

En este caso vamos a utilizar el proyecto disponible en:

<https://github.com/big-data-europe/docker-hadoop>

Entre otras cosas, en el proyecto tenemos el fichero dockerfile necesario para lanzar los principales servicios del ecosistema Hadoop como son:

* **NameNode**: nodo maestro encargado de gestionar el sistema de archivos HDFS.

Texto

El contenido generado por IA puede ser incorrecto.

Accederemos al servicio web de gestión que implementa en:

http://<dockerhadoop\_IP\_address>:9870/dfshealth.html#tab-overview

Donde para descubir la IP del NameNode podemos consultar Portainer o inspeccionar la red con el comando docker network inspect.

docker inspect -f '{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' namenode



Luego, en mi caso, puedo acceder a ver el cluster usando:

<http://localhost:9870/dfshealth.html#tab-overview>

Recordar que es más fácil con Portainer.

O más fácil con Portainer:

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Una vez conocida la IP podemos visualizar el estado de HDFS en el navegador;

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

* **DataNode**: nodos donde se almacenan físicamente los bloques de datos.
* **ResourceManager** y **NodeManager**: encargados de gestionar los recursos del clúster mediante YARN, permitiendo la ejecución de tareas MapReduce.
* **HistoryServer**: permite consultar el histórico de las ejecuciones realizadas

# Práctica contar palabras con MapReduce

## Copiar ficheros de libros a namenode

docker cp C:\docker\hadoop\input namenode:/tmp/input/



## Conectamos al contenedor namenode

docker exec -it namenode /bin/bash

Texto

El contenido generado por IA puede ser incorrecto.

docker exec: Ordena a Docker ejecutar un comando dentro de un contenedor que ya está en funcionamiento.

-it: Son dos opciones combinadas:

* -i (interactive): mantiene la entrada estándar abierta.
* -t (tty): asigna una terminal pseudo-TTY (simula una terminal real).

Juntas, permiten tener una sesión de terminal interactiva dentro del contenedor, igual que en una shell normal.

namenode:Es el nombre del contenedor donde quieres ejecutar el comando. En tu caso, el contenedor que ejecuta el servicio NameNode.

/bin/bash: Es el comando que deseas ejecutar dentro del contenedor. En este caso, abrir una shell de bash (una de las Shell más usadas en Linux / Unix). Esto es así porque los contenedores Docker suelen ejecutarse con Linux internamente.

Usar /bin/bash abre una terminal dentro del contenedor, desde donde puedes interactuar con el sistema como si fuera una máquina virtual.

## Subir archivos desde el contenedor a HDFS

# Crear carpeta en HDFS

hdfs dfs -mkdir -p /input

# Subir archivos a HDFS desde la carpeta temporal del contenedor

hdfs dfs -put /tmp/input/\* /input

# Verificar los archivos en HDFS

hdfs dfs -ls /input

Texto

El contenido generado por IA puede ser incorrecto.

## Ejecutamos JAR de contar palabras

En el material asociado a la práctica tenemos un ejemplo de MapReduce hecho en Java de nombre “hadoop-mapreduce-examples-2.7.1-sources.jar”

Copiamos el JAR al contenedor.

docker cp hadoop-mapreduce-examples-2.7.1-sources.jar namenode:/tmp/

**Paso final: Ejecutar ejemplo de MapReduce con el siguiente comando:**

>> docker exec -it namenode /bin/bash

:/# cd tmp

:/# hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar org.apache.hadoop.examples.WordCount input output

Se genera una nueva carpeta output en HDFS que podemos ver haciendo:

:/# hdfs dfs -ls

Found 2 items

drwxr-xr-x - root supergroup 0 2025-07-22 10:46 input

drwxr-xr-x - root supergroup 0 2025-07-22 12:01 output

Y los resultados están en:

:/# hdfs dfs -ls /output

Found 2 items

-rw-r--r-- 3 root supergroup 0 2025-07-22 12:05 /output/\_SUCCESS

-rw-r--r-- 3 root supergroup 434093 2025-07-22 12:05 /output/part-r-00000

Nos lo traemos a nuestro contenedor con otro comando HDFS

:/# hdfs dfs -get /output/part-r-00000 /tmp/resultado.txt

Y a nuestro equipo con:

>> C:\Docker\hadoop\jar>docker cp namenode:/tmp/resultado.txt C:\docker\hadoop\resultado\resultado.txt

## Verificando el resultado con un fichero trivial

**- Copiar datos desde Windows al contenedor NameNode:**

C:\Docker\hadoop\input>docker cp ./basico1.txt namenode:/tmp/input\_basico/

**- Dentro del contenedor**

hdfs dfs -mkdir -p /input\_basico

hdfs dfs -put /tmp/input\_basico/\* /input\_basico

hdfs dfs -ls /input\_basico

**- Ejecutar MapReduce clásico**

hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar org.apache.hadoop.examples.WordCount /input\_basico /output\_basico

**- Consultar el resultado**

hdfs dfs -ls /output\_basico

hdfs dfs -cat /output\_basico/\*

Texto

El contenido generado por IA puede ser incorrecto.

# Hadoop y Python (pendiente)

Como extensión opcional de la práctica, se plantea un reto adicional: ejecutar el mismo trabajo de MapReduce, pero usando scripts en Python mediante Hadoop Streaming.

Esta parte ha resultado especialmente desafiante, ya que los scripts de usuario deben ejecutarse en los nodos de cómputo del clúster (NodeManagers), donde debe estar disponible un intérprete Python y una configuración correcta del entorno. A pesar de haber instalado Python en el namenode, la ejecución distribuida requiere que también esté presente y operativo en los nodos que efectivamente ejecutan las tareas, lo cual puede implicar la personalización de imágenes Docker o la instalación manual en contenedores específicos.

## Problemas encontrados:

* Fallos por falta de intérprete (code 127).
* Scripts no disponibles en los nodos si no se usa correctamente el parámetro -file.
* Necesidad de permisos de ejecución y shebang correcto (#!/usr/bin/env python3).

## Reto para el alumno:

Se anima a los estudiantes a resolver estos problemas como ejercicio complementario. ¿Eres capaz de personalizar la imagen del NodeManager para soportar scripts en Python? ¿Puedes ejecutar correctamente un mapper.py y reducer.py con Hadoop Streaming? El código de mapper.py y reducer.py está en la carpeta compartida asociada a la práctica.