Hadoop con Docker: Introducción a MapReduce

En este ejercicio aprenderás cómo levantar un clúster de Hadoop local utilizando Docker y ejecutar un proceso MapReduce sobre un archivo de texto. Este entorno reproducible permite practicar y entender los conceptos fundamentales de **procesamiento distribuido**, sin necesidad de usar recursos en la nube o clústeres físicos.

Esta ejercicio te permitirá:

- Comprender la arquitectura básica de Hadoop (HDFS + YARN)
- Utilizar contenedores Docker para emular un clúster
- Ejecutar un trabajo MapReduce real (conteo de palabras)
- Interpretar y validar resultados de salida

¿Qué es Hadoop?

Hadoop es un framework de software de código abierto para almacenar datos y ejecutar aplicaciones en clústeres de hardware estándar. Ofrece almacenamiento distribuido (HDFS) y procesamiento paralelo (MapReduce).

Componentes clave:

- **HDFS (Hadoop Distributed File System):** Sistema de archivos distribuido tolerante a fallos.
- **YARN (Yet Another Resource Negotiator):** Planificador de recursos y tareas.
- **MapReduce:** Modelo de programación para procesamiento distribuido de datos.

¿Por qué usar Docker?

Instalar Hadoop puede ser complejo, especialmente para principiantes. Docker permite crear contenedores que simulan un entorno Hadoop real, ahorrando tiempo de configuración y garantizando que todos los estudiantes trabajen en un entorno idéntico.

Prerequisitos

- Tener **Git** y **Docker** instalados.
- Tener conocimientos básicos de consola / terminal.

Paso a paso

1. Clona el repositorio `BigData-Hadoop` desde GitHub

```
git clone https://github.com/therobotacademy/BigData-Hadoop
cd BigData-Hadoop
```

Este repositorio contiene archivos `Dockerfile` y `docker-compose.yml` que definen la infraestructura de Hadoop.

2. Inicia los contenedores necesarios con Docker Compose

```
docker-compose up -d
```

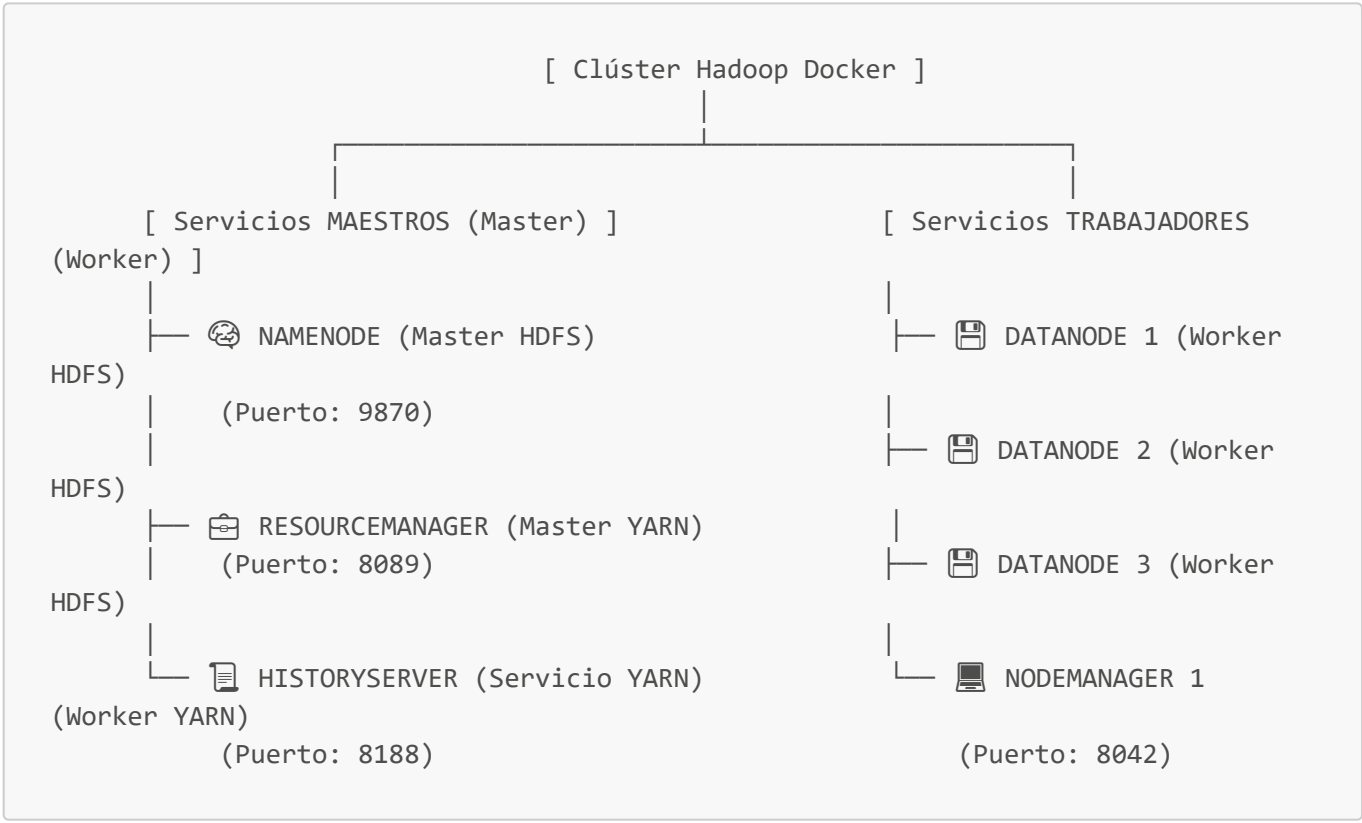
Esto levantará 5 contenedores:

- `namenode` (gestiona HDFS)
- `datanode` (almacena bloques de datos)
- `resourcemanager` (gestiona trabajos YARN)
- `nodemanager` (ejecuta tareas MapReduce)
- `historyserver` (para consultar trabajos pasados)

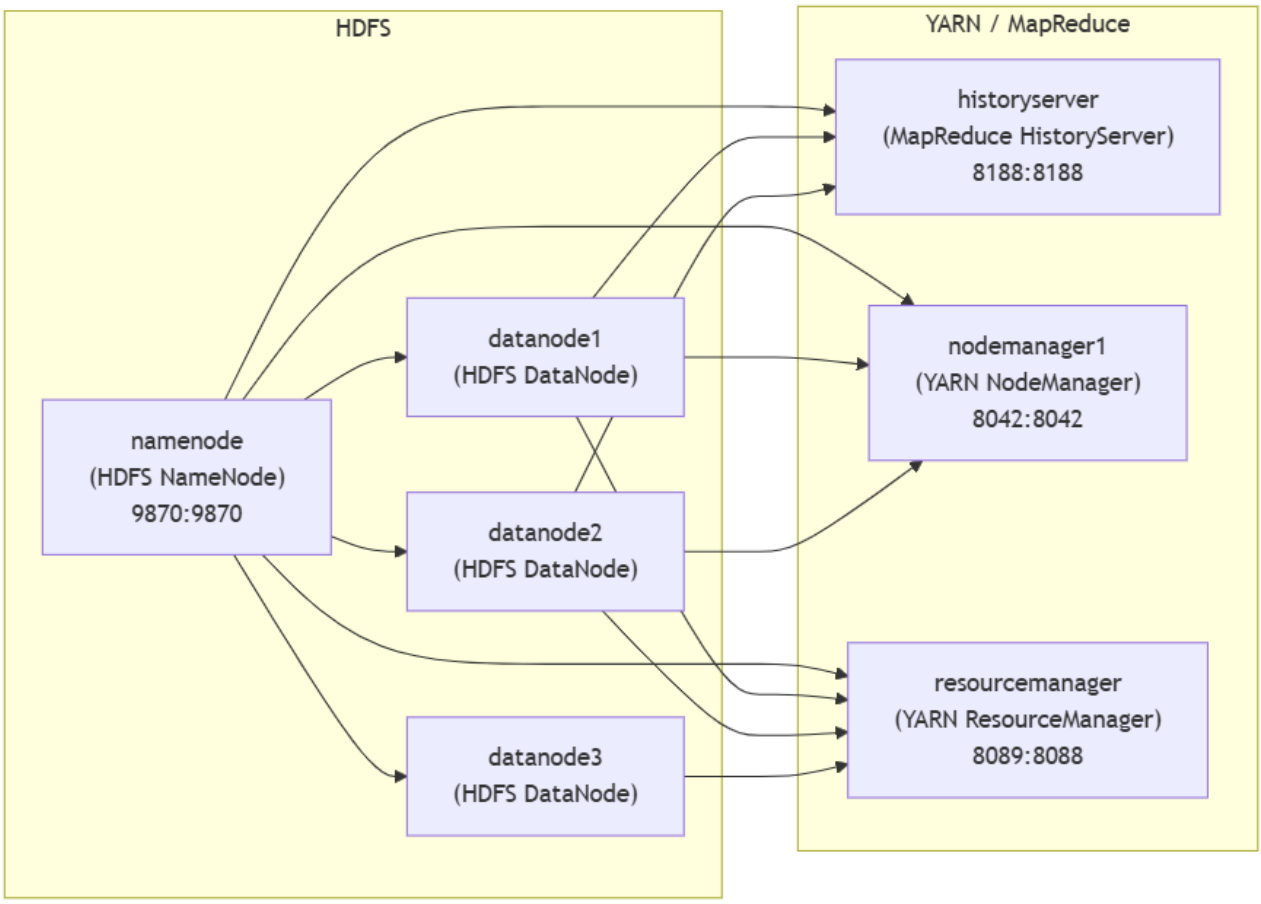
Verifica que estén corriendo:

```
docker ps
```

El cluster consta de los siguientes servicios (uno por contenedor):



El siguiente diagrama muestra el orden de arranque (cada flecha indica el contenedor que ha de haber arrancado previamente):



3. Accede al contenedor `namenode`

```
docker exec -it namenode bash
```

Aquí puedes ejecutar comandos como si fuera una terminal Linux con Hadoop configurado.

Introducción al sistema de archivos HDFS

HDFS divide los archivos en bloques (por defecto 128MB) y los distribuye en varios nodos con redundancia. Para este laboratorio, crearemos la siguiente estructura de directorios:

```
hdfs dfs -mkdir -p /user/root/input
```

Verifica:

```
hdfs dfs -ls /user/root
```

4. Descarga un archivo de texto para analizar

Usaremos el libro *Don Quijote de La Mancha* (1 MB de texto), útil para tareas de conteo de palabras.

Puedes descargarlo desde este enlace: el_quijote.txt

5. Descarga el archivo `.jar` de MapReduce

Descarga el siguiente `.jar`, que contiene ejemplos precompilados de MapReduce:

hadoop-mapreduce-examples-2.7.1-sources.jar

Mueve ambos archivos al directorio del repositorio y cópialos al contenedor:

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

6. Copia el archivo al sistema de archivos distribuido (HDFS)

Dentro del contenedor `namenode`:

```
cd /tmp
hdfs dfs -put el_quijote.txt /user/root/input
```

Ejecutar un trabajo MapReduce

¿Qué es MapReduce?

MapReduce es un modelo de programación distribuida basado en dos funciones:

- **Map:** transforma pares clave/valor de entrada en nuevos pares intermedios.
- **Reduce:** agrupa esos pares intermedios y realiza cálculos agregados.

Para entender cómo funciona MapReduce sin necesidad de ejecutar código, podemos reproducir su flujo completo utilizando una simple tabla de Excel. Este ejercicio muestra cómo se transforman los datos paso a paso a través de las fases **Map**, **Shuffle** y **Reduce**, que son la base del procesamiento distribuido en Hadoop.

Datos de partida

Imaginemos que tenemos una pequeña tabla con importes de ventas por producto y tienda:

Producto	Tienda	Importe
A	1	20
B	1	30
A	2	25
B	2	40
C	1	50

Nuestro objetivo será calcular el total vendido por cada producto, tal y como lo haría un trabajo MapReduce real.

1️⃣ Fase Map: transformar filas en pares clave–valor

En MapReduce, cada mapper procesa sus datos de entrada de forma independiente y genera pares **(clave, valor)**. En este caso, la **clave** será el producto y el **valor** será el importe de la venta.

Transformación:

- (A, 20)
- (B, 30)
- (A, 25)
- (B, 40)
- (C, 50)

Cada fila produce un par.

2️⃣ Fase Shuffle: agrupar valores por clave

Antes de llegar a los reducers, Hadoop agrupa automáticamente todos los valores asociados a la misma clave. Es el paso que reordena y redistribuye los datos a través del clúster.

Agrupación obtenida:

- (A, [20, 25])
- (B, [30, 40])
- (C, [50])

Aquí vemos cómo todas las ventas del mismo producto se reúnen para ser procesadas juntas.

3️⃣ Fase Reduce: agregación final

El reducer recibe la clave y la lista de valores generada en el paso anterior. Ahora solo tiene que aplicar la operación correspondiente: en este caso, **sumar** los importes.

Resultados finales:

- (A, 45)
- (B, 70)
- (C, 50)

Estos serían los totales de ventas por producto.

Ejecutar el ejemplo de WordCount

```
hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar
org.apache.hadoop.examples.WordCount /user/root/input /user/root/output
```

Este trabajo leerá el archivo de entrada, aplicará la función de conteo, y guardará la salida.

Verificar resultados

```
hdfs dfs -cat /user/root/output/part-r-00000
```

Esto mostrará líneas como:

```
"aventura"    47
"caballero"   91
"molino"      12
```

Para mover el resultado a tu máquina local:

```
hdfs dfs -cat /user/root/output/part-r-00000 > /tmp/quijote_wc.txt
exit
docker cp namenode:/tmp/quijote_wc.txt .
```

Visualiza el estado del clúster

Puedes acceder al dashboard de Hadoop desde tu navegador:

🌐 http://localhost:9870

Allí puedes consultar el sistema de archivos, nodos activos, y tareas ejecutadas.

Apagar el clúster

Cuando termines:

```
docker-compose down
```

Ejercicios

- 🧪 **Comparativa:** procesa archivos de distintos tamaños y analiza los tiempos.
- 🌐 **Escalado horizontal:** agrega más nodos `datanode` y `nodemanager` al `docker-compose.yml`.
- 📊 **Visualización:** exporta el conteo y crea una nube de palabras usando Python (matplotlib + wordcloud).

📓 Conceptos de Hadoop

Hadoop es un framework de código abierto diseñado para el almacenamiento distribuido y el procesamiento paralelo de grandes volúmenes de datos. Fue creado por Doug Cutting y Mike Cafarella, inspirado en los

papers de Google sobre GFS (Google File System) y MapReduce.

Características principales:

- Escalable horizontalmente (añadiendo más nodos).
- Tolerante a fallos.
- Divide y conquista: divide grandes tareas en pequeñas sub-tareas distribuidas.

◇ HDFS (Hadoop Distributed File System)

Es el **sistema de archivos distribuido** utilizado por Hadoop. Diseñado para almacenar archivos de gran tamaño y permitir acceso eficiente desde múltiples nodos.

¿Cómo funciona?

- **Bloques:** divide cada archivo en bloques (típicamente 128 MB).
- **Replicación:** cada bloque se replica (por defecto 3 veces) en distintos DataNodes.
- **Tolerancia a fallos:** si un nodo falla, los bloques aún están disponibles en otros.

Componentes clave:

- **NameNode:** gestiona la estructura del sistema de archivos (metadatos).
- **DataNode:** almacena físicamente los bloques de datos.

◇ NameNode

El **NameNode** es el servidor maestro de HDFS. Mantiene la estructura del sistema de archivos (directorios, archivos, permisos, ubicación de bloques).

📌 **Nota importante:** El NameNode es crítico. Si se pierde, se pierde el acceso a todo el sistema HDFS (aunque los datos estén físicamente en los DataNodes).

◇ DataNode

El **DataNode** es el nodo esclavo que almacena los bloques de datos físicos. Los DataNodes responden a peticiones del NameNode y permiten lectura/escritura de bloques.

- Cada DataNode puede tener múltiples bloques.
- Ejecuta tareas de almacenamiento, lectura y envío de datos.

◇ YARN (Yet Another Resource Negotiator)

YARN es el sistema de gestión de recursos y ejecución de tareas en Hadoop. Se introdujo en Hadoop 2.0 para separar el procesamiento del almacenamiento.

Componentes principales:

- **ResourceManager (RM):** gestiona los recursos globales del clúster y asigna contenedores para tareas.
- **NodeManager (NM):** gestiona los recursos y tareas en un nodo específico.

YARN permite que múltiples frameworks (como MapReduce, Spark, Tez) usen Hadoop simultáneamente.

◇ MapReduce

MapReduce es un modelo de programación distribuido para el procesamiento de grandes volúmenes de datos en paralelo.

Fases:

1. **Map:** toma una entrada y la transforma en pares clave-valor intermedios.
2. **Shuffle & Sort:** agrupa los pares por clave.
3. **Reduce:** aplica una función de agregación o transformación a cada grupo de claves.

Ejemplo típico: **conteo de palabras**.

- `Map`: ("palabra", 1)
- `Reduce`: ("palabra", suma de ocurrencias)

MapReduce garantiza procesamiento paralelo, balanceado y escalable.

◇ ResourceManager

El **ResourceManager** es el componente principal de YARN. Se encarga de:

- Asignar recursos a las tareas Map y Reduce.
- Coordinar la ejecución de trabajos.
- Decidir qué nodos usar según disponibilidad.

◇ NodeManager

Cada nodo tiene un **NodeManager** que informa al ResourceManager sobre su estado y ejecuta contenedores de tareas (Map y Reduce) en su máquina local.

Es el agente de ejecución del clúster.

◇ HistoryServer

El **HistoryServer** permite visualizar trabajos MapReduce ya finalizados. Es útil para:

- Ver logs históricos.
- Analizar tiempos y errores.
- Realizar debugging de trabajos MapReduce.

Se accede desde la interfaz web de Hadoop.

◇ Docker

Docker es una plataforma de virtualización ligera que permite crear, ejecutar y administrar contenedores.

En este laboratorio usamos Docker para:

- Simular un clúster de Hadoop completo en una sola máquina.
- Evitar instalaciones manuales complejas.
- Asegurar entornos idénticos para todos los estudiantes.

◇ Docker Compose

Herramienta que permite definir y administrar múltiples contenedores Docker desde un único archivo (`docker-compose.yml`).

En nuestro caso, levanta todos los componentes de Hadoop automáticamente.

Ejemplo:

```
version: '2'
services:
  namenode:
    image: bde2020/hadoop-namenode
    ports:
      - "9870:9870"
    environment:
      - CLUSTER_NAME=test
```

◇ WordCount (Ejemplo de MapReduce)

El ejemplo **WordCount** es el clásico "Hola Mundo" de MapReduce. Cuenta cuántas veces aparece cada palabra en un archivo de texto.

- Mapper: divide texto en palabras → ("palabra", 1)
- Reducer: suma valores de cada clave → ("palabra", n)

Es útil para entender el flujo completo de MapReduce, desde la ingesta de datos hasta la agregación de resultados.

◇ Interfaz Web de Hadoop

Hadoop incluye una interfaz web para monitorear:

- Archivos en HDFS.
- Nodos activos.
- Progreso de trabajos MapReduce.

📍 Accede desde tu navegador: http://localhost:9870

◇ Comandos de HDFS más usados

Comando	Descripción
`hdfs dfs -mkdir`	Crea un directorio en HDFS
`hdfs dfs -put archivo destino`	Sube archivo desde el sistema local a HDFS
`hdfs dfs -ls`	Lista archivos y directorios
`hdfs dfs -cat archivo`	Muestra el contenido de un archivo en HDFS

Comando	Descripción
`hdfs dfs -rm`	Elimina un archivo de HDFS

◇ Comando `hadoop jar`

Permite ejecutar un `.jar` con clases Hadoop, especificando la clase principal del trabajo.

Ejemplo:

```
hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar
org.apache.hadoop.examples.WordCount input output
```

◇ Shuffle & Sort (entre Map y Reduce)

Proceso intermedio de MapReduce que:

- Agrupa todas las claves iguales.
- Reorganiza los datos para que el reducer trabaje con todos los valores de cada clave.

Este paso es transparente para el usuario pero crítico para el rendimiento.

◇ Replicación de datos

HDFS replica los bloques de datos automáticamente. La configuración por defecto suele ser **3 réplicas** por bloque.

Esto garantiza:

- Alta disponibilidad.
- Tolerancia a fallos.
- Balanceo de carga en la lectura.

◇ Escalabilidad horizontal

Hadoop está diseñado para escalar **añadiendo más nodos** (no aumentando el tamaño de un solo servidor). Esto lo diferencia de bases de datos tradicionales y es una de sus fortalezas principales.

◇ Tolerancia a fallos

Si un nodo de datos falla:

- El NameNode detecta la pérdida.
- Redistribuye los bloques a otros nodos.
- No se interrumpe el procesamiento.

Hadoop fue diseñado desde cero para funcionar en entornos donde los fallos son comunes.

Etiquetas: Docker · Hadoop · MapReduce · HDFS · Big Data