CURSO DE ESPECIALIZACIÓN EN INTELIGENCIA ARTIFICIAL Y BIG DATA

Tarea Docker-Compose Clúster Hadoop

En esta práctica vas a montar un clúster Hadoop distribuido utilizando Docker Compose, construyendo tú mismo cada parte del entorno. Verás cómo se comunican los nodos, cómo se guardan los datos y cómo se ejecutan los jobs en un clúster real mientras lo vas levantando paso a paso.

JUAN FRANCISCO GARCÍA HINOJOSA

ÍNDICE.

1. Introducción **2**

2. Estructura de directorios del proyecto **3**

3. Scripts de arranque del clúster **3**

3.1. Script del NameNode: `start-hdfs.sh` 3

3.2. Script de los DataNodes: `init-datanode.sh` 4

4. Configuración de los archivos XML de Hadoop **5**

4.1. Archivo `core-site.xml` 6

4.2. Archivo `hdfs-site.xml` 6

4.3. Archivo `yarn-site.xml` 6

4.4. Archivo `mapred-site.xml` 7

5. Creación del archivo `docker-compose.yml` **7**

6. Arranque del clúster con Docker Compose **10**

1. Introducción

En esta práctica vamos a trabajar con un clúster Hadoop distribuido montado con **Docker Compose**. Hasta ahora has usado Hadoop en un único nodo, pero en entornos reales se despliega en varios nodos que colaboran entre sí. El objetivo es que puedas ver cómo se organiza un clúster y que aprendas a ponerlo en marcha tú mismo utilizando contenedores.

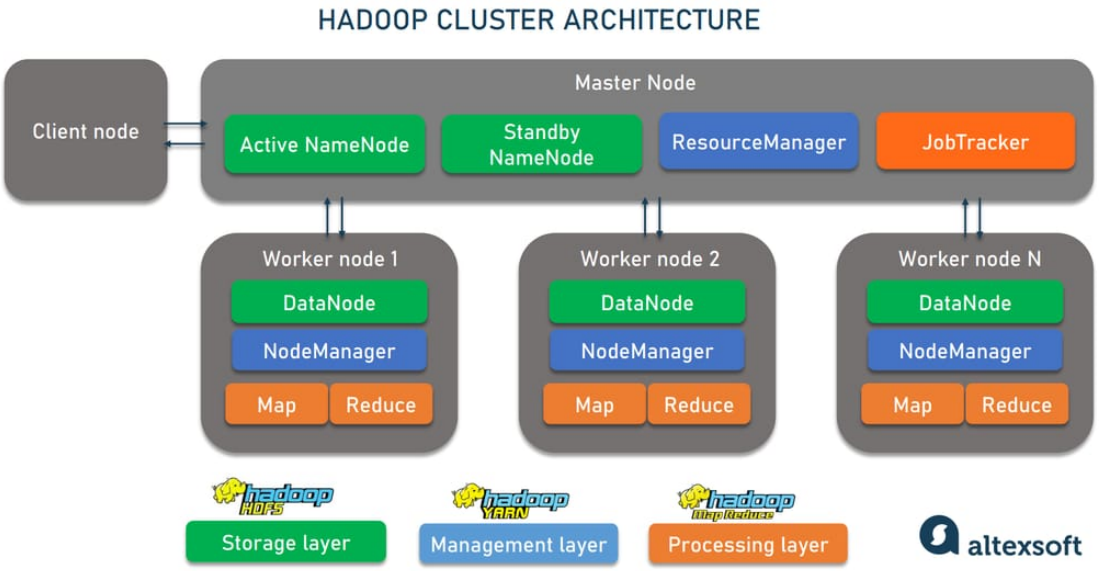

Figura 1: Arquitectura general de un clúster Hadoop

En la imagen puedes ver los componentes principales: el *NameNode*, que gestiona la información del sistema de archivos, los *DataNodes*, que almacenan los datos, y los servicios de YARN, que se encargan de gestionar los recursos y coordinar la ejecución distribuida. Esta es la estructura habitual de un clúster Hadoop moderno.

La idea de esta práctica es que revises los archivos de configuración, prepares los servicios en Docker y consigas que los nodos se comuniquen correctamente. También podrás probar HDFS y ejecutar un ejemplo sencillo como *WordCount*. Lo importante no es memorizar comandos, sino entender qué hace cada parte y comprobar que todo funciona como debe mientras vas montando el entorno.

2. Estructura de directorios del proyecto

La estructura inicial que debes crear para el proyecto es la siguiente:

```
hadoop-cluster/
├── docker-compose.yml
├── config/
│   ├── core-site.xml
│   ├── hdfs-site.xml
│   ├── yarn-site.xml
│   └── mapred-site.xml
├── namenode/
├── datanode1/
├── datanode2/
├── start-hdfs.sh
└── init-datanode.sh
```

3. Scripts de arranque del clúster

Una vez creada la estructura básica de directorios, el siguiente paso consiste en preparar dos scripts sencillos que facilitan el arranque del NameNode y de los DataNodes dentro de los contenedores. La imagen oficial de Hadoop no inicia por sí sola los servicios principales, por lo que estos scripts permiten indicar de forma clara qué debe ponerse en marcha en cada nodo.

Os recuerdo que el uso de scripts no es obligatorio. Todo lo que se quiera ejecutar al arrancar el contenedor podría integrarse directamente en la sección `command` del archivo `docker-compose.yml`. No obstante, separarlo en scripts suele simplificar la lectura y hace más cómodo modificar o revisar cada parte cuando sea necesario.

3.1. Script del NameNode: `start-hdfs.sh`

Este script inicia el NameNode de forma controlada. Antes de arrancarlo, revisa si el directorio interno del NameNode ya contiene la estructura generada por un arranque anterior. Si es la primera vez que se ejecuta, realiza el formateo inicial; en caso contrario, arranca directamente el servicio sin modificar los metadatos existentes.

El comando:

```
hdfs namenode -format -force -nonInteractive
```

realiza el formateo del NameNode. Esto crea el *clusterID*, los directorios internos y los ficheros de metadatos necesarios para que HDFS funcione. La opción `-force` permite formatearlo sin mostrar advertencias, y `-nonInteractive` evita que pida confirmación por consola. Este formateo solo debe ejecutarse la primera vez que se inicializa el clúster.

```
#!/bin/bash

# 'set -e' detiene el script si ocurre algun error durante la ejecucin.
set -e

NN_DIR="/opt/hadoop/data/nameNode"

echo "Iniciando NameNode"
echo "Directorio de metadatos: $NN_DIR"

if [ ! -d "$NN_DIR/current" ]; then
  echo "No se encuentra un NameNode inicializado. Formateando..."
  hdfs namenode -format -force -nonInteractive
else
  echo "NameNode ya inicializado. Se omite el formateo."
fi

echo "Arrancando servicio NameNode..."
hdfs namenode
```

3.2. Script de los DataNodes: `init-datanode.sh`

El script de los DataNodes es más sencillo. Su objetivo es asegurarse de que el directorio asignado existe y arrancar el servicio correspondiente. No realiza formateo ni operaciones adicionales. Al igual que el script del NameNode, termina mostrando los logs para mantener el contenedor activo.

```
#!/bin/bash

# 'set -e' detiene el script si algun comando produce un error.
set -e
```

```
DN_DIR="/opt/hadoop/data/dataNode"

echo "Iniciando DataNode"
echo "Directorio de datos: $DN_DIR"

if [ -d "\$DN_DIR" ]; then
    rm -rf "\$DN_DIR"/*
    echo "Directorio datanode borrado correctamente"
else
    echo "El directorio datanode no existe. Creando..."
    mkdir -p "\$DN_DIR"
fi

echo "Estableciendo propiertario y permisos"
chown -R hadoop:hadoop "\$DN_DIR"
chmod 755 "\$DN_DIR"

echo "Arrancando servicio DataNode..."
hdfs  datanode
```

Tienes la opción de utilizar estos scripts o de hacerlo manualmente una vez que hayas montado el docker-compose. Si decides no utilizar los scripts, cuando levantes el docker compose debes entrar en cada uno de los nodos para crear sus directorios y arrancar el servicio hdfs correspondiente.

4. Configuración de los archivos XML de Hadoop

Los archivos XML del directorio `config/` contienen la configuración que utilizarán el NameNode, los DataNodes y los servicios de YARN. Aunque cada uno cumple una función distinta, todos deben estar coordinados para que el clúster arranque correctamente dentro de Docker. A continuación se explica brevemente para qué sirve cada fichero y se muestra una plantilla que deberás completar.

4.1. Archivo `core-site.xml`

Este archivo define la configuración general del sistema de archivos de Hadoop. El valor más importante es `fs.defaultFS`, que indica la dirección del NameNode usando el nombre del contenedor y el puerto correspondiente. Todos los nodos del clúster usarán esta ruta para localizar el NameNode y comunicarse con él.

```
<configuration>
    <!-- Aqui debe definirse fs.defaultFS apuntando al NameNode -->
</configuration>
```

4.2. Archivo `hdfs-site.xml`

Este archivo contiene la configuración específica del sistema HDFS. Aquí se definen las rutas donde el NameNode guarda sus metadatos y donde los DataNodes almacenan sus bloques. También aparece el factor de replicación, que determina cómo se distribuyen los datos entre los nodos.

```
<configuration>
    <!-- Factor de replicacion -->
    <!-- Directorio del NameNode -->
    <!-- Directorio de los DataNodes -->
</configuration>
```

4.3. Archivo `yarn-site.xml`

Este fichero especifica la configuración de YARN, el componente que gestiona la ejecución de trabajos distribuidos. Aquí debe indicarse el contenedor que actúa como ResourceManager, así como los puertos utilizados para la comunicación y las interfaces web.

```
<configuration>
    <!-- Opciones necesarias para NodeManager -->
    <!-- Host del ResourceManager -->
    <!-- Puertos de comunicacion web de YARN -->
</configuration>
```

4.4. Archivo `mapred-site.xml`

Este archivo indica qué motor de ejecución debe utilizar Hadoop para los trabajos MapReduce. En un clúster real debe configurarse para que utilice YARN como framework, de manera que las tareas se distribuyan entre varios nodos en lugar de ejecutarse en modo local.

```
<configuration>
    <!-- Configuracion para que MapReduce utilice YARN -->
    <!-- Si fuese necesario, parametros adicionales
        relacionados con el JobHistory-->
</configuration>
```

5. Creación del archivo `docker-compose.yml`

Una vez definidos los directorios, los scripts de arranque y los ficheros de configuración, el siguiente paso consiste en preparar tu archivo `docker-compose.yml`. Este archivo será el encargado de organizar los contenedores del NameNode y de los DataNodes, asignar los volúmenes, montar la configuración y permitir que todos los nodos se comuniquen entre sí.

A continuación se muestran ejemplos orientativos que te pueden servir como guía para completar tu propio archivo, sin proporcionar una solución cerrada. La idea es que identifiques qué necesita cada servicio y vayas montando las piezas según lo necesite tu clúster.

Ejemplo orientativo de definición de un NameNode

Un servicio suele comenzar indicando la imagen a utilizar, el nombre del contenedor y algunas variables de entorno. Un ejemplo podría ser:

```
services:
  namenode:
    image: apache/hadoop:3.4.1
    container_name: namenode
    hostname: namenode
    user: root
```

Aquí puedes observar los elementos básicos: el nombre del servicio, la imagen de Hadoop y el nombre del contenedor. A partir de aquí puedes ir completando el resto de opciones.

Ejemplo orientativo de variables de entorno

Cada servicio del clúster necesita algunas variables de entorno básicas para que Hadoop pueda localizar sus directorios internos y sus ficheros de configuración. Estas variables permiten que cada nodo utilice la instalación interna de Hadoop y cargue correctamente los archivos XML del directorio `config/`.

Un ejemplo mínimo de cómo definir variables de entorno en un servicio podría ser:

```
environment:
  - HADOOP_HOME=/opt/hadoop
  - HADOOP_CONF_DIR=/opt/hadoop/etc/hadoop
```

Estas variables indican dónde está instalado Hadoop dentro del contenedor y dónde se encuentran los ficheros de configuración. Todos los nodos del clúster (NameNode y DataNodes) deben utilizar esta misma configuración para trabajar de forma coherente.

Si decides añadir más variables de entorno (por ejemplo, valores relacionados con el usuario o la JVM), puedes hacerlo utilizando la misma estructura. Lo importante es que todos los servicios apunten a las mismas rutas internas para evitar inconsistencias al arrancar el clúster.

Ejemplo orientativo de volúmenes

Cada nodo necesita acceder a un directorio propio donde almacenar sus datos, así como al directorio `config` donde están los ficheros XML. Un montaje típico de volúmenes podría tener una forma similar a esta:

```
volumes:
  - ./namenode:/opt/hadoop/data/nameNode
  - ./config:/opt/hadoop/etc/hadoop
```

El primer volumen corresponde a los datos del NameNode y el segundo proporciona la configuración interna de Hadoop. Los DataNodes seguirían un esquema parecido, adaptado a su propio directorio de datos.

Ejemplo orientativo de ejecución del script de arranque

Si decides utilizar los scripts creados en secciones anteriores, puedes indicarlo con la opción `command`:

```
command: [ "/bin/bash", "/start-hdfs.sh" ]
```

Para los DataNodes la estructura es similar, utilizando su script correspondiente.

Ejemplo orientativo de dependencia entre servicios

Es recomendable que los DataNodes esperen a que el NameNode esté iniciado antes de arrancar. Para ello se utiliza la opción `depends_on`:

```
    depends_on:
      - namenode
```

Esto ayuda a que cada servicio se inicie en el orden adecuado.

Ejemplo orientativo de exposición de puertos

El NameNode dispone de una interfaz web que debe ser accesible desde el exterior. Un ejemplo de cómo exponer ese puerto sería:

```
    ports:
      - "9870:9870"
```

En cambio, los DataNodes no requieren exposición de puertos hacia el exterior.

Ejemplo orientativo de definición de red interna

Para que los contenedores puedan comunicarse entre sí, deben estar conectados a una red común. Una definición mínima de red podría ser:

```
networks:
  hdfs_network:
    driver: bridge
```

Los servicios deberán enlazarse a esta red:

```
    networks:
      - hdfs_network
```

Esto permite que los nodos se comuniquen utilizando los nombres definidos en el archivo, como `namenode`, `datanode1` o `datanode2`.

Estos fragmentos muestran únicamente la estructura de las partes que necesita un `docker-compose.yml` para levantar un clúster Hadoop en contenedores. Tu tarea consiste en combinar estos elementos, completar las rutas, añadir los servicios necesarios y ajustar los parámetros hasta construir un archivo funcional que permita iniciar el NameNode y los DataNodes correctamente.

6. Arranque del clúster con Docker Compose

Una vez preparado el archivo `docker-compose.yml`, puedes poner en marcha el clúster. El objetivo de este apartado es comprobar que los contenedores arrancan correctamente y que los nodos del clúster pueden comunicarse entre sí.

1. Levantar los servicios

Desde el directorio principal del proyecto, inicia los contenedores utilizando:

```
docker compose up -d
```

Si el archivo está bien construido, se iniciará un contenedor para el NameNode y otro para cada DataNode. Todos deberían aparecer en estado `Up`.

2. Comprobar el estado de los contenedores

Puedes verificar si algún servicio ha fallado o se ha reiniciado usando:

```
docker ps
```

Si algún contenedor no aparece como en ejecución, es recomendable revisar sus logs:

```
docker logs namenode
docker logs datanode1
docker logs datanode2
```

Los mensajes de salida te ayudarán a detectar problemas de configuración, directorios incorrectos o fallos al iniciar los demonios de Hadoop.

3. Interfaz web del NameNode

Si el NameNode se ha iniciado correctamente, su interfaz web estará accesible en:

`http://localhost:9870`

En esta interfaz debería aparecer información sobre el sistema de archivos, incluyendo la lista de DataNodes activos. Si la configuración es correcta, los nodos definidos en tu archivo `docker-compose.yml` deberían mostrarse como nodos vivos.

4. Interfaz web de YARN

Además del NameNode, YARN también dispone de su propia interfaz web para consultar el estado del ResourceManager y de las aplicaciones distribuidas. Está disponible en:

`http://localhost:8088`

Si todo funciona como debe, podrás ver información sobre los nodos disponibles, la capacidad del clúster y los trabajos que se ejecuten más adelante.

5. Entrega de la práctica

Al finalizar la práctica, debes entregar todo el proyecto que has ido construyendo. La entrega deberá incluir, como mínimo:

- El archivo `docker-compose.yml` completo y funcional.
- La carpeta `config/` con todos los ficheros XML que hayas preparado (`core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, `mapred-site.xml` y cualquier otro que hayas añadido).
- Los scripts de arranque utilizados (`start-hdfs.sh` e `init-datanode.sh`, en caso de usarlos).
- Los directorios de datos vacíos (`namenode/`, `datanode1/`, `datanode2/`) para que se vea la estructura completa del proyecto.
- Un breve fichero de texto (`README.txt` o similar) donde expliques cómo arrancar el clúster y cómo has probado el comando `WordCount`.

El proyecto se entregará comprimido en un único fichero (por ejemplo, en formato `.zip` o `.tar.gz`) manteniendo la estructura de carpetas mostrada en esta guía.