Lab Sheet: Single-Node Hadoop Cluster with Docker

Objective

In this lab, you will deploy a single-node Hadoop cluster using a pre-built Docker image. This setup allows you to run and test Hadoop in an isolated Docker container, making it easier to explore Hadoop features without affecting your main system.

Prerequisites

Docker: Ensure Docker is installed and running on your system. You can check if Docker is installed by running:
bash
Copy code
`docker --version`

Step 1: Pull the Hadoop Docker Image

1. **Choose a Hadoop Docker Image**:
    - Many Docker images are available for Hadoop single-node setups. For this lab, we'll use the `bde2020/hadoop-namenode` image from the Big Data Europe repository on Docker Hub.

Pull the Docker image:

```
docker pull bde2020/hadoop-namenode:latest
```

    -
2. **Verify the Download**:

List Docker images to confirm the Hadoop image is downloaded:

```
docker images
```

    -

Step 2: Start the Hadoop Container

1. **Run the Container**:

Start a container with the necessary configurations to act as a single-node Hadoop cluster:

```
docker run -it --name hadoop-cluster -p 9870:9870 -p 8088:8088 -p
50070:50070 bde2020/hadoop-namenode:latest /bin/bash
```

   ○
2. **Start Hadoop Services**:

Once inside the container's shell, start the Hadoop services:

```
start-all.sh
```

   ○
   ○ This will initialize HDFS and YARN.

Step 3: Access Hadoop Web Interfaces

- **HDFS Web Interface**:
  - Open http://localhost:9870 in your browser to view the HDFS NameNode status.
- **YARN Web Interface**:
  - Access the YARN ResourceManager at http://localhost:8088.

Step 4: Running a Sample MapReduce Job

1. **Upload Sample Data to HDFS**:

Inside the Docker container, create an input directory and upload sample files to HDFS:

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

hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/hadoop/input
```

   ○
2. **Run the WordCount Job**:

Run a built-in Hadoop example to test the setup:
bash

```
hadoop jar
```

```
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
wordcount /user/hadoop/input /user/hadoop/output
```

3. **Check the Output**:

Display the results of the WordCount job:

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

Step 5: Exiting the Container

1. **Stop the Container**:

If you want to stop the Hadoop container without deleting it, run:

```
docker stop hadoop-cluster
```

2. **Restart the Container**:

To restart the container later, use:

```
docker start -i hadoop-cluster
```

Task 1: Customize HDFS Configurations

- Edit the `core-site.xml` and `hdfs-site.xml` configuration files inside the container to change settings like replication factor or default block size.
- Run a MapReduce job and observe how changes in configurations affect performance.

Task 2: Write a Custom MapReduce Job

- Write a Java or Python-based MapReduce job to analyze custom data.
- Add the input data to HDFS and run the job, then observe the output.

Task 3: Push Your Work to GitHub

- Document your configuration changes, custom MapReduce code, and insights from the output.
- Commit and push your work to a GitHub repository, including instructions to replicate your setup.
