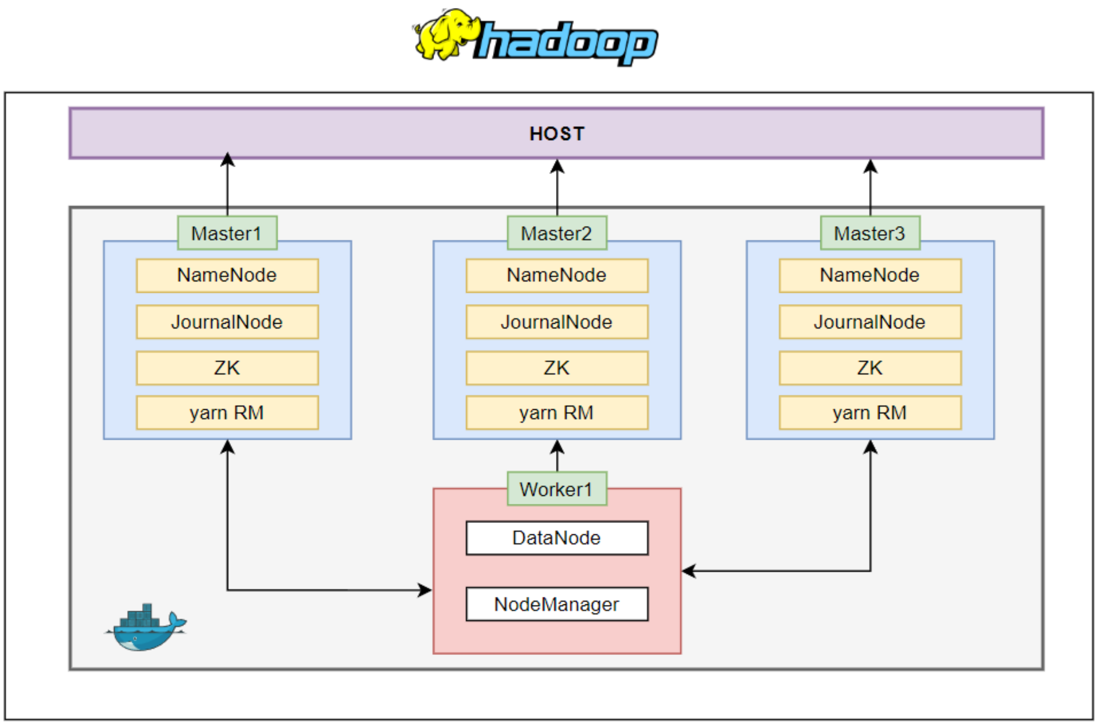
ITI - Data Management Track

Project: Building Highly Available Hadoop Cluster with Docker



By: Youssef Etman – Ibrahim Elsadek

Capacity: Individual

Deadline: Wed 2-MAR-2025 at 11:59 pm, all required files should be uploaded on G drive.

Discussion: **Offline** on Sun 6-MAR-2025

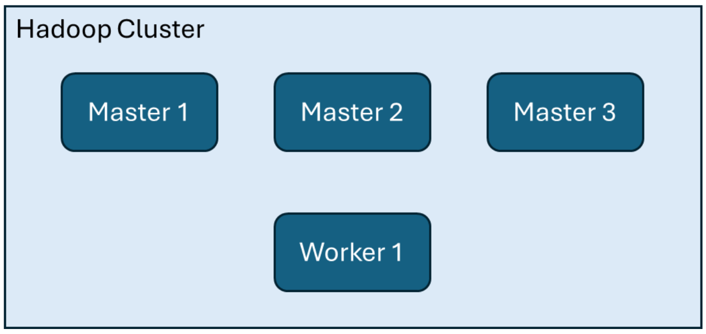
The project will be divided into 2 parts:

1. Manually setting up a highly available Hadoop cluster **(HDFS HA + YARN HA)**.
2. Automating the previous process via Docker.

Part 1: Hadoop

A) Introduction:

The cluster that we want to build is as follows:



3 Masters, 1 Worker

- Each Master Node should run:
  - Zookeeper Service.
  - HDFS Journal Node.
  - HDFS Namenode.
  - YARN Resource Manager.

- Each Worker Node should run:
  - HDFS Datanode.
  - YARN Nodemanager.

Of course, only one master will have an active NameNode and only one master will have an active ResourceManager.

B) Instructions & hints:

- Your first step should be setting up 4 Ubuntu containers within the same docker network. (Recommended version: ubuntu 22.04)
- The second step would be to manually install & configure the required services on each node. (Recommended Hadoop version: 3.3.6)
- A friendly advice to follow the documentation first and not to depend on AI chatbots.
- You can start here: Apache Hadoop 3.4.1 – HDFS High Availability Using the Quorum Journal Manager

C) <u>Delivery:</u>

- Please upload the following to G drive: History of commands run on each of the 4 nodes in order to set up the cluster.
- On the discussion day you will be asked to:
  1. Open HDFS UI on all of the three masters.
  2. Stop the namenode service on the currently active namenode to test the automatic failover of HDFS.
  3. Stop the Resource Manager service on the currently active RM to test the automatic failover of YARN.
  4. Ingest data into HDFS.
  5. Run MapReduce job on ingested data.
  6. Horizontally scaling your cluster by adding a second worker node and making sure that its now a member of HDFS data nodes (From HDFS UI) and a member of YARN node managers (From YARN UI).

Part 2: Docker:

After having 4 running containers from the previous part. Now we need to containerize our cluster to have a single image that could be used later to run multiple instances (master or worker) from Hadoop.

Expecting in this part to convert your commands and instructions you applied in part one and create a docker file from it, then build docker compose file for multiple containers using the image generated from that docker file.

A) <u>Specs:</u>

1. **Docker file:**
   - clear steps for building image
   - versioned package installations
   - optimized layer arrangement
   - maximum utilization of docker file instructions instead of Linux commands (ex: WORKDIR /app instead of cd /app for changing current dir.)
   - all services/machines, both master and workers should start from one docker file not a file for each one, means you need to handle this in docker file or in entrypoint/CMD
   - A simple java code for map-reduce job will be used as our production code that we will use it to run an on-demand job while containers are running.

2. **Docker compose file:**

   Should include 4 services (3 master and 1 worker) with a good setup for network – volumes – dependencies – health check

   - Hostnames and container names should be assigned with an appropriate convention
   - All runtime env variables should be defined in docker compose file and build time env variables in dockerfile
   - Masters should be accessible from hosts, but workers should be only accessible from masters' machines
   - Ports mapping should have no conflict with other host's used ports
   - Data, code files and configuration files should be kept in an appropriate way
   - Build a dependency between services to have a well arch cluster
   - Limit services resources by assigning appropriate resources.

B) <u>Delivery:</u>

We should have the files below uploaded:

- history files for all commands you have run during setup
- docker file
- compose file
- entrypoint or cmd scripts if exist

I'll use your docker file to build the image and your compose-file to start the whole cluster without any additional operation.

C) <u>Extra part as bonus :)</u>

- With your second laptop or with another teammate, make one laptop as a client only for docker and the other laptop is the engine and the server which has all things (hint: look at docker API)
- With your teammate, make your laptops as a cluster one has all masters, and the other is the worker (hint: using docker swarm)