**HADOOP AND HIVE USING DOCKER**

**Big Data** is a collection of data that is huge in volume, yet growing exponentially with time. It is a data with so large size and complexity that none of traditional data management tools can store it or process it efficiently. Big data is also a data but with huge size. It is of three types namely,

* Structured data – Structured data is the neatly organized data you keep in databases, datasets and spreadsheets. It’s easy for traditional analytics tools to this data. Eg: Credit card numbers.
* Unstructured data – is the data which does not conforms to a data model and has no easily identifiable structure such that it cannot be used by a computer program easily. It is not organized in a pre-defined manner. Eg: Audio and Video files.
* Semi-Structured data – Semi-structured data refers to data that is not captured or formatted in conventional ways. It does not follow the format of a tabular data model. Eg: Email, Webpages.

**Hadoop**

**Hadoop is an open source framework which uses the mapreduce algorithm and does batch processing. It is a tool for implementing bigdata. It permits the large volumes of data on node system. It consists of four modules namely,**

* **HDFS – Hadoop Distributed File System, it manages large sets of data, scales single apache cluster into 100’s of nodes.**
* **YARN – It is a cluster resource manager that schedules tasks and allocates memory, cpu to various applications.**
* **Mapreduce – Splits the big data processing tasks into smaller ones, distributes small tasks across different nodes, then runs each task.**
* **Hadoop Common(Core) – Set of common libraries and utilities that other 3 modules depend on.**

**Pre-Requisite:**

**Git, Docker, Docker-Compose**

**Procedure:**

**apt-get update**

**apt-get install git**

**apt install docker.io**

**apt-get install docker**

**apt-get install docker-compose**

* **Check the versions of git, docker, docker-compose for confirmation using**

**Git –version**

**Docker –version**

**Docker-compose –version**

* **Git clone (github url) // to clone the repo from github**
* **Cd docker-hadoop**
* **Ls**
* **Cat docker-compose.yml // Base configuration**
* **It will display various services with its property configurations as below,**

**Namenode**

**Datanode**

**Resourcemanager**

**Nodemanager**

**Historyserver**

* **Docker-compose up –d // To run services as detachable one in the background**
* **Docker container ls // displays the containers running for all nodes**
* **Docker exec –it namenode /bin/bash //entering inside the docker container of the namenode**
* **Hdfs dfs –ls / //listing the directories under /**
* **Hdfs dfs –mkdir -p /user/root //we can create a new directory using hdfs dfs**
* **Hdfs dfs –ls /**
* **exit**
* **Clear**
* **Docker container ls**
* **Ifconfig //note the ip and the port number(for specific node namely namenode - 9870)**
* **192.168.56.101:9870 //hit this in the browser – all the datanodes and the namenodes will be present**
* **Docker-compose down (to stop all running containers)**
* **For Mapreduce job, download it using wget “url”**
* **wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-mapreduce-examples/2.7.1/hadoop-mapreduce-examples-2.7.1-sources.jar**
* **Docker container ls**
* **Docker cp hadoop-mapreduce namenode :/tmp**
* **Vim input.txt //create a text file and enter some data**
* **Docker cp input.txt namenode :/tmp //uploading the file to the container**
* **Docker exec –it namenode /bin/bash**
* **Cd /tmp**
* **Ls**
* **Input1.txt //create an other text file and upload it to hdfs**
* **Hdfs dfs –mkdir /user/root/input**
* **Hdfs dfs –put input1.txt /user/root/input (uploading input1.txt)**
* **Hdfs dfs –cat /user/root/input/input1.txt (view)**
* **Hdfs jar xxx.jar org input**
* **Output will be displayed**
* **Exit**
* **Docker container ls (job wil be working fine)**
* **Docker-compose down // to stop the containers**
* **Docker container ls (Check if all the containers are stopped)**

**Base Configuration:**

**Namenode – Master node in Apache hadoop hdfs, maintains and manages the blocks present on the datanode (slavenode). It controls the access to files by client and is configured with RAM.**

**Datanode – It is the slavenode in hdfs, non-expensive, not high quality and not highly available. Its is a block server that stores the data in local file ext3 or ext4.**

**Resourcemanager – Master that attributes all available cluster resources, it helps to manage the distributed applications running on yarn system.**

**Nodemanager – It is a per machine/ per-node framework agent, responsible for containers, monitoring their resource usage, reporting the same to the resource manager.**

**Historyserver - Provides the application history from the event logs stored in the file system.**

services:

  namenode:

    image: bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8

    container\_name: namenode

    restart: always

    ports:

      - 9870:9870

      - 9000:9000

    volumes:

      - hadoop\_namenode:/hadoop/dfs/name

    environment:

      - CLUSTER\_NAME=test

    env\_file:

      - ./hadoop.env

  datanode:

    image: bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8

    container\_name: datanode

    restart: always

    volumes:

      - hadoop\_datanode:/hadoop/dfs/data

    environment:

      SERVICE\_PRECONDITION: "namenode:9870"

    env\_file:

      - ./hadoop.env

  resourcemanager:

    image: bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8

    container\_name: resourcemanager

    restart: always

    environment:

      SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864"

    env\_file:

      - ./hadoop.env

  nodemanager1:

    image: bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8

    container\_name: nodemanager

    restart: always

    environment:

      SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"

    env\_file:

      - ./hadoop.env

  historyserver:

    image: bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8

    container\_name: historyserver

    restart: always

    environment:

      SERVICE\_PRECONDITION: "namenode:9000 namenode:9870 datanode:9864 resourcemanager:8088"

    volumes:

      - hadoop\_historyserver:/hadoop/yarn/timeline

    env\_file:

      - ./hadoop.env

volumes:

  hadoop\_namenode:

  hadoop\_datanode:

  hadoop\_historyserver:

**Docker-hive**

**Apache Hive is an open source data warehouse built on the top of hadoop. It can be used for analyzing and querying large datasets that are store in hadoop files (HDFS). It is a query language used for reading,writing and managing large datasets using sql.**

**Mkdir docker-hive**

**Cd docker-hive**

**Git clone**

**Ls**

**Vim docker-compose.yml**

**Docker-compose up –d //To run in a detachable way**

**Clear**

**Docker container ls**

**Docker exec –it docker\_hive-hive-server\_1 /bin/bash //to enter**

**/opt/hive/bin/beeline –u jdbc:hive2://localhost:10000 (hive client, beeline is a cli used for connecting)**

**Show databases;**

**Create database dbname;**

**Show databases;**

**Use name;**

**create table tablename(x,y);**

**show tables;**

**describe tables;**

**describe formatted tables;**

**quit**

**exit**

**Docker-hive base configuration:**

* **Namenode**
* **Datanode**
* **Hive-server**
* **Hive-metastore – central repo of the hive metadata, it stores the metadata for hive tables and partitions in relational database, it communicates with background such as mysql to retrieve data api.**
* **Hive-metastore-postgresql (object relational database)**

services:

  namenode:

    image: bde2020/hadoop-namenode:2.0.0-hadoop2.7.4-java8

    volumes:

      - namenode:/hadoop/dfs/name

    environment:

      - CLUSTER\_NAME=test

    env\_file:

      - ./hadoop-hive.env

    ports:

      - "50070:50070"

  datanode:

    image: bde2020/hadoop-datanode:2.0.0-hadoop2.7.4-java8

    volumes:

      - datanode:/hadoop/dfs/data

    env\_file:

      - ./hadoop-hive.env

    environment:

      SERVICE\_PRECONDITION: "namenode:50070"

    ports:

      - "50075:50075"

  hive-server:

    image: bde2020/hive:2.3.2-postgresql-metastore

    env\_file:

      - ./hadoop-hive.env

    environment:

      HIVE\_CORE\_CONF\_javax\_jdo\_option\_ConnectionURL: "jdbc:postgresql://hive-metastore/metastore"

      SERVICE\_PRECONDITION: "hive-metastore:9083"

    ports:

      - "10000:10000"

  hive-metastore:

    image: bde2020/hive:2.3.2-postgresql-metastore

    env\_file:

      - ./hadoop-hive.env

    command: /opt/hive/bin/hive --service metastore

    environment:

      SERVICE\_PRECONDITION: "namenode:50070 datanode:50075 hive-metastore-postgresql:5432"

    ports:

      - "9083:9083"

  hive-metastore-postgresql:

    image: bde2020/hive-metastore-postgresql:2.3.0

  presto-coordinator:

    image: shawnzhu/prestodb:0.181

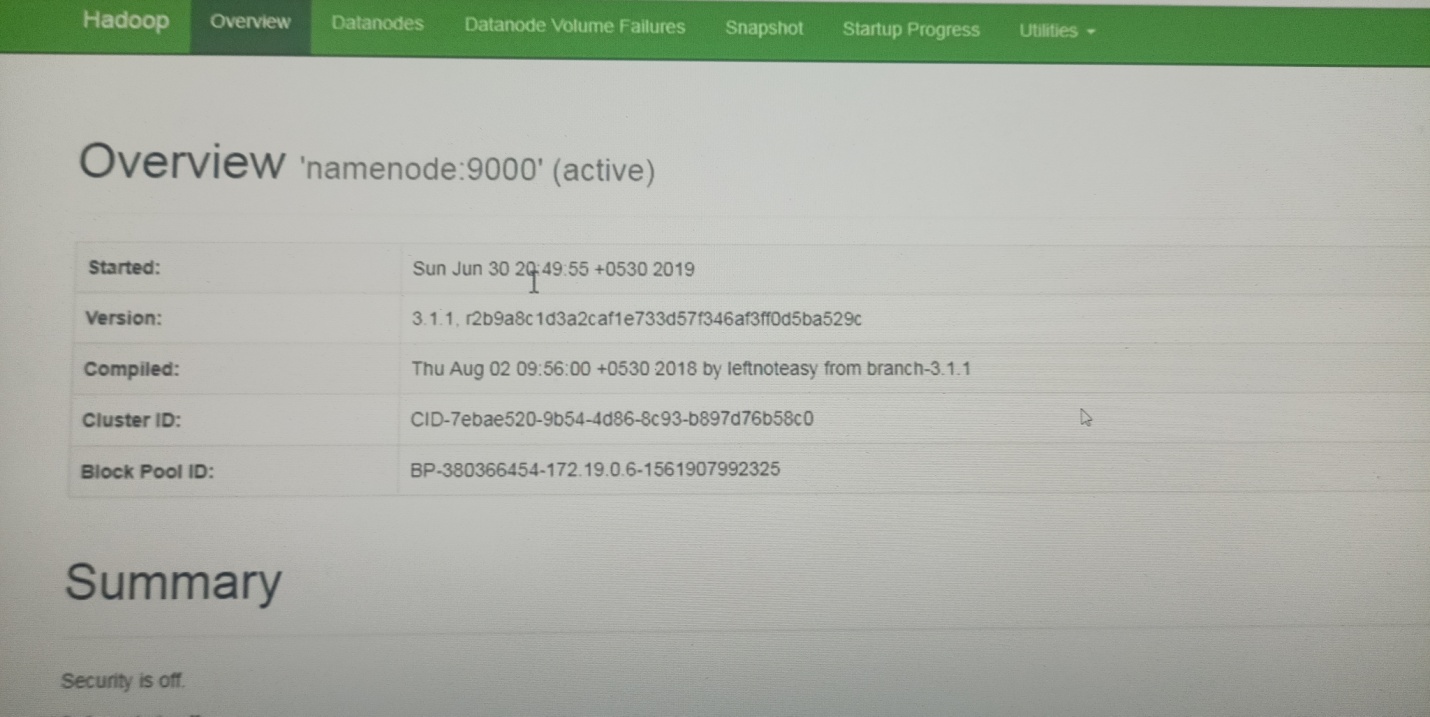
    ports:

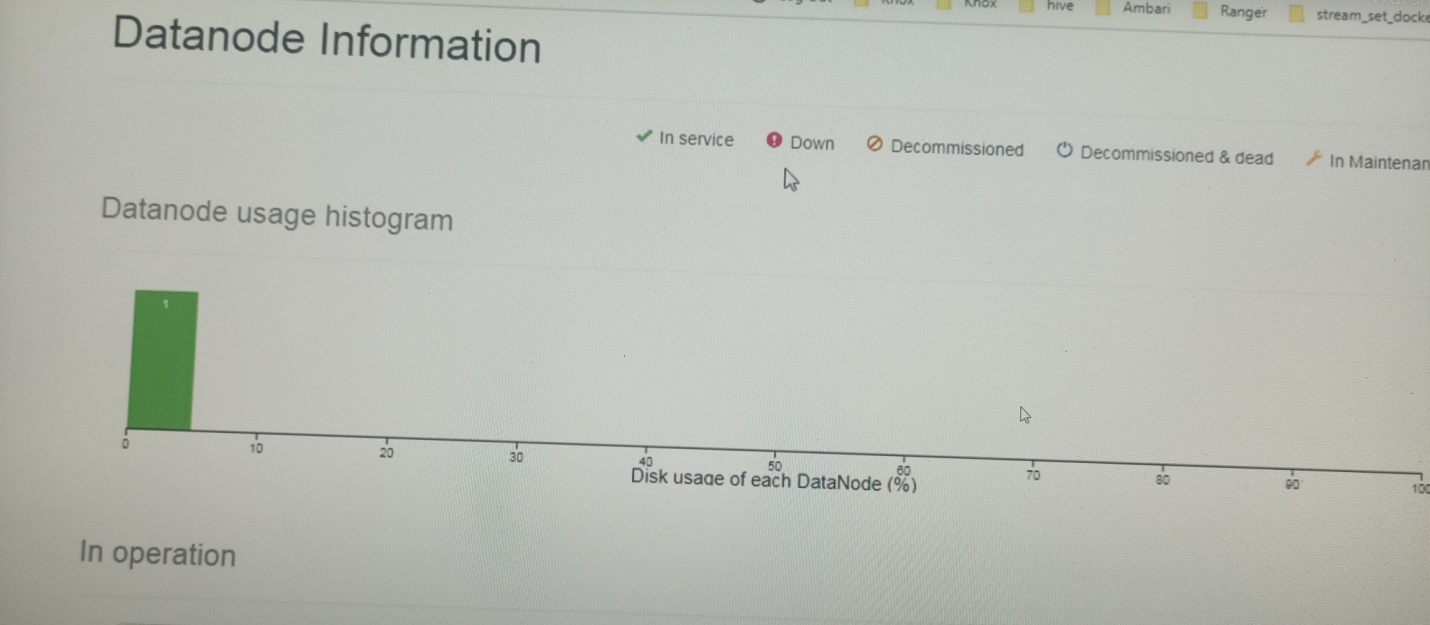
      - "8080:8080"

volumes:

  namenode:

  datanode:

****

****