Установка и настройка Apache Hadoop в docker (на базе Ubuntu)

**Установка docker**

1. Обновляем ПО

sudo apt update

sudo apt upgrade

2. Устанавливаем curl

sudo apt install curl

3. Устанавливаем docker

способ 1: sudo apt install docker, но в офф репозитории ubuntu содержится не последняя версия docker

способ 2:

2.1 Проводим установку необходимого ПО

sudo apt-get install \

apt-transport-https \

ca-certificates \

curl \

gnupg-agent \

software-properties-common

2.2. Скачиваем и добавляем официальный ключ docker репозитария

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

2.3 Добавляем репозиторий

$ sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

2.4 Индексируем пакеты в репозитарии

$ sudo apt-get update

2.5 Устанавливаем последнюю версию docker

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

Способ 3 (оптимальный):

3.1. Скачиваем скрипт установки с сайта <https://get.docker.com/> и запускаем его

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sh get-docker.sh

4. Установка docker-compose.

4.1. Скачиваем последнюю версию docker-compose в директорию /usr/local/bin/docker-compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

4.2. Устанавливаем бит исполнения

sudo chmod +x /usr/local/bin/docker-compose

**настройка Apache Hadoop на базе docker**

Для утснавки Hadoop в docker контейнерах нам понадобится образ Hadoop, который можно найти на hub.docker.com, но мы будем использовать тестовый подготовленный репозитарий от Big Data Europe. Для скачивания будем использовать Git

1. Установка Git

sudo apt install git

2. Качаем git репозитарий

$ git clone git@github.com:big-data-europe/docker-hadoop.git

Мы видим, что у нас появилась папка docker-hadoop. Переходим в нее и видим ряд директорий, в которых определены dockerfile для создания необходимых images, а в корне видим два файла docker-compose



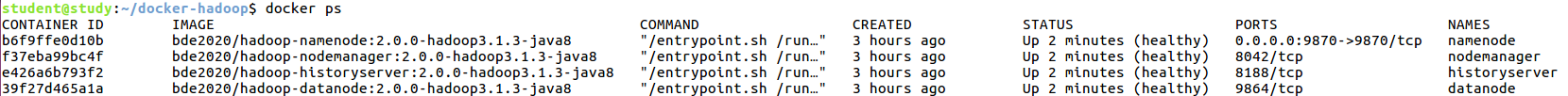
3. Для того, чтобы развернуть простой HDFS кластер запускаем команду

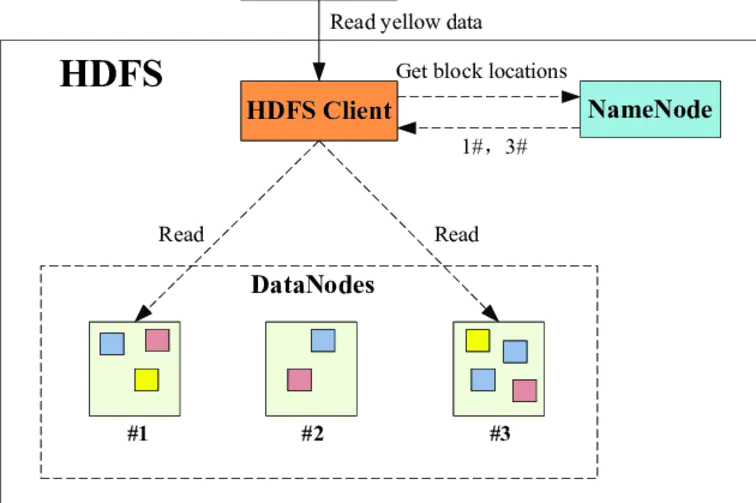
sudo docker-compose up -d

3.1. Чтобы развернуть кластер в swarm используем команду

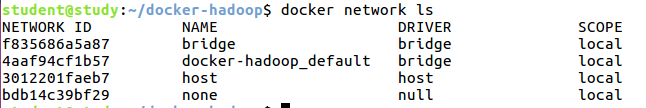
sudo docker stack deploy -c docker-compose-v3.yml hadoop

4. Посмотреть созданные и запущенные контейнеры можно командой docker ps

Видим, что у нас запущено 5 контейнеров, если необходимо большее количество контейнеров в качестве data node, можем описать их по аналогии в файле docker-compose.yaml, который определяет запущенные контейнеры и взаимосвязи между ними. Примерную схему развернутого кластера представляет следующий рисунок, но в нашем случае есть только одна datanode.

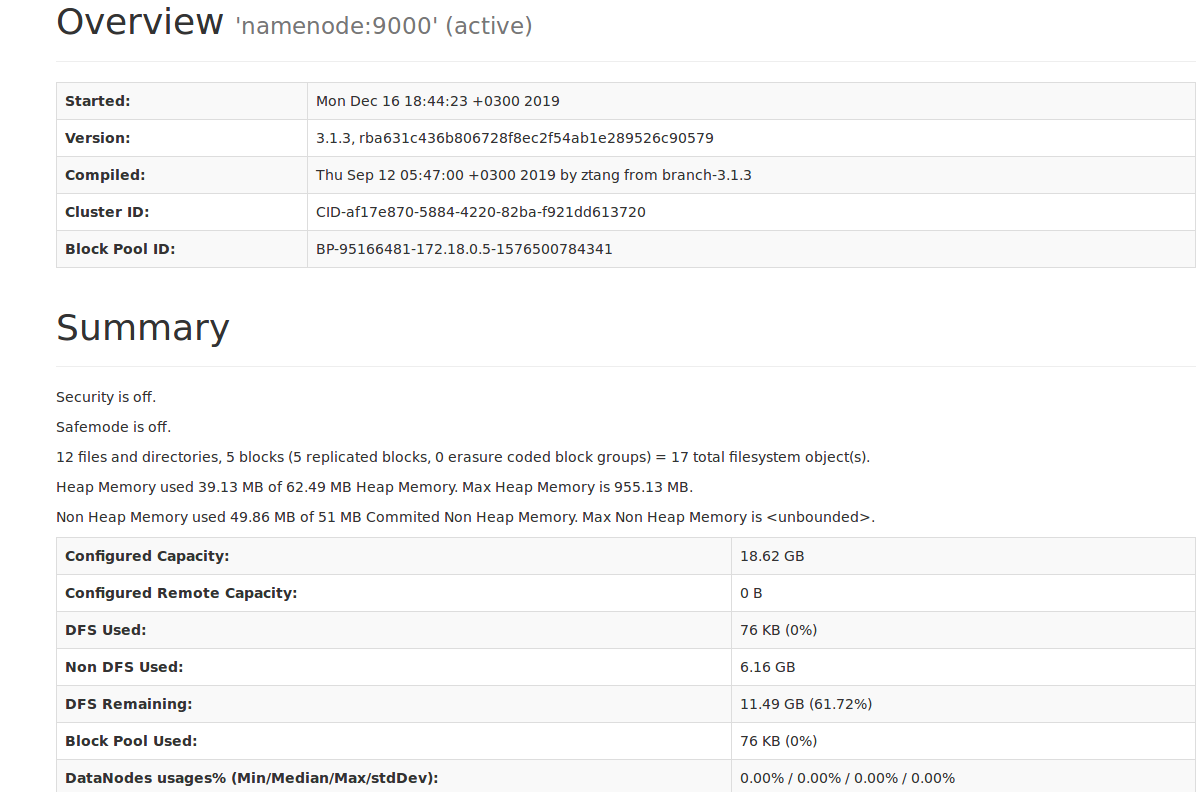


Можно изучить сеть и посмотреть ip адреса выданные каждому контейнеру в кластере командой

и docker network inspect docker-hadoop\_default

Описание сети и взаимосвязей можно найти в файле docker-compose.yml

5. Переходим по URL [http://localhost:9870](http://localhost:9870/), чтобы посмотреть текущий статус кластера полученный с namenode

6. Проведем тестирование нашего Hadoop кластера

зайдем внутрь контейнера выполнив следующую команду $ docker exec -it namenode bash

6.1. Создадим несколько простых текстовых файлов

$ mkdir input

$ echo "Hello World" >input/f1.txt

$ echo "Hello Docker" >input/f2.txt

6.2. Теперь создадим директорию в HDFS

$ hadoop fs -mkdir -p input

6.3. Чтобы отправить все созданные файлы во все datanode в HDFS используем следующую команду

$ hdfs dfs -put ./input/\* input

6.4. Запустим простую программу подсчета слов. Сначала скопируем

$ docker cp ../hadoop-mapreduce-examples-2.7.1-sources.jar container id (который можно посмотреть командой docker ps):hadoop-mapreduce-examples-2.7.1-sources.jar

И запускаем

root@namenode:/# hadoop jar hadoop-mapreduce-examples-2.7.1-sources.jar org.apache.hadoop.examples.WordCount input output

Чтобы вывести результат выполнения программы

root@namenode:/# hdfs dfs -cat output/part-r-00000

**Работа с Apache Hive в HDFS**

Что такое Hive? Hive — это приложение, которое предназначено для аналитики структуированных данных в экосистеме Hadoop. У Hive есть возможность хранения схемы для данных в базах данныхБ называемых Hive Metadatastore и он может применять эту схему для дальнейшей работы с данными в HDFS.

Hive состоит из двух основных частей — Hive Metastore и Hive.

Hive Metastore использует реляционные БД для хранения схем данных, данных о таблицах, базах данных, столбцов, типов данных и как они связаны в HDFS.

Hive является второй важной компонентой в Apache Hive. Он состоит из двух основных элементов — HiveQL Process Engine, который взаимодействует с Metastore, используя HiveQL (SQL like). Второй элемент — это execution engine, который является мостом между HiveQL PE и MapReduce. Он позволяет обрабатывать запросы, которые написаны на HiveQL и использовать MR для представления результата.

Запустим контейнер с установленным hive

docker run --hostname=quickstart.cloudera --privileged=true -t -i -p 8888:8888 -v /tmp/test\_volume:/volume cloudera/quickstart:latest /usr/bin/docker-quickstart bash

Приготовим данные с которыми будем работать

[root@quickstart /]# cd /volume/

[root@quickstart volume]# sed 's/::/,/g' movies.dat > movies.t

[root@quickstart volume]# sed 's/::/,/g' ratings.dat > ratings.t

[root@quickstart volume]# sed 's/::/,/g' users.dat > users.t

Переместим данные в HDFS

[root@quickstart volume]# hdfs dfs -mkdir -p /moviedata/ratings

[root@quickstart volume]# hdfs dfs -mkdir -p /moviedata/movies

[root@quickstart volume]# hdfs dfs -mkdir -p /moviedata/users

[root@quickstart volume]# hdfs dfs -put movies.t /moviedata/movies

[root@quickstart volume]# hdfs dfs -put ratings.t /moviedata/ratings

[root@quickstart volume]# hdfs dfs -put users.t /moviedata/users

Начнем с выполнения ряда команд Hive. Для этого будем использовать Beeline, который является командным иинтерфейсом для Hive. Вводим следующую команду для аутентификации в Hive.

beeline -u jdbc:hive2://quickstart:10000/default -n admin -d org.apache.hive.jdbc.HiveDriver

Ключ -u указывает url к БД Hive. Мы используем jbdc драйвер, который определяем ключом -d. Ключ -n определяет пользователя.

Первым делом создадим БД, определив в ней наши таблицы

create database moviedata;

use moviedata;

Команда use определяет БД, с которой мы будем работать. Следующим шагом создаем таблицы

CREATE EXTERNAL TABLE ratings (userid INT, movieid INT, rating INT, tstamp STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/moviedata/ratings';

EXTERNAL говорит Hive, что таблица будет распологаться поверх данных и не нужно загружать их непосредственно в Hive. Поэтому если вы удалите эту таблицу из hive, данные не удалятся.

ROW FORMAT DELIMITED означает, что каждая линия - это ряд в текстовом файле

FIELDS TERMINATED BY ',' Определяет, что столбцы разделяются запятой.

STORED AS TEXTFILE — данные хранятся в текстовом файле

LOCATION просто означает, что данные размещены в HDFS в соотвествующей директории.

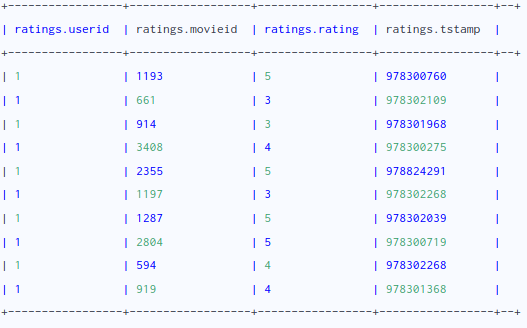
Теперь создадим больше таблиц

CREATE EXTERNAL TABLE movies ( movieid INT, title STRING, genres ARRAY<STRING>) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' COLLECTION ITEMS TERMINATED BY "|" STORED AS TEXTFILE LOCATION '/moviedata/movies/';

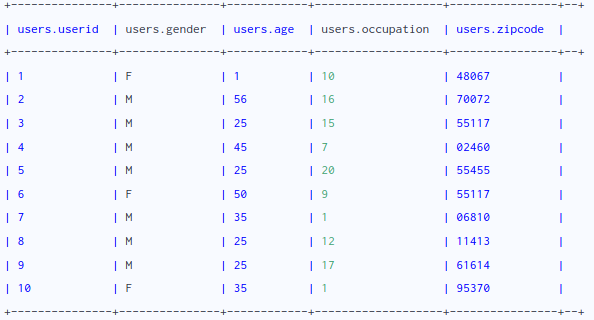
CREATE EXTERNAL TABLE users (userid INT, gender STRING, age INT, occupation INT, zipcode STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE LOCATION '/moviedata/users/';

Мы можем быть уверены, что данные загружены корректно, если увидим следующий вывод

select \* from ratings LIMIT 10;



select \* from users LIMIT 10;

select \* from movies LIMIT 10; - проанализируйте результат выполнения команды самостоятельно

А теперь немного поиграем с запросами

Сначала получим рейтинг кинофильмов. Обратите вниманиена таблицы рейтинга, мы осуществляем поиск по movieid.

select m.title, r.rating FROM MOVIES m JOIN RATINGS r ON (m.movieid = r.movieid) LIMIT 20;

Теперь давайте попробуем упорядочить рейтинги фильмов

select m.title, r.rating FROM MOVIES m JOIN RATINGS r ON (m.movieid = r.movieid) ORDER BY r.rating DESC LIMIT 20;

Попробуем получить среднюю оценку за каждый фильм

select m.title, AVG(r.rating) as AVG\_RATING FROM MOVIES m JOIN RATINGS r ON (m.movieid = r.movieid) GROUP BY m.title ORDER BY AVG\_RATING DESC LIMIT 20;