# **Docker Setup:**

**Step-1:** open a browser and sign in (https://www.docker.com/), then download the docker-desktop-installer.exe (**Windows-AMD64**) file.

**step-2:** Search for 'turn windows features on or off' from your PC's start menu.

**step-3:** See the options 'Hyper-V' and 'Windows Subsystem for Linux' and check them, then click 'OK'. The pc will required restart after few minutes.

**step-4:** Open command prompt and type-

> wsl --status

> wsl --update

> wsl --set-default-version 2

> wsl --set-default-version 1

> wsl --set-default-version 2

**step-5:** install and execute (docker-desktop-installer.exe) that you have downloaded. PC will take restart again.

step-6: Open command prompt and type-

> docker version

> docker images

> docker search mysql

# **Docker Execution**

**step-1:** open a browser and serach for 'https://docker-curriculum.com/' and see the commands.

**step-2:** run 'ocker-desktop-installer.exe' application and open the docker terminal and execute the following commands:

$ docker run hello-world

$ docker pull busybox

$ docker run busybox

$ docker run busybox echo "hello from busybox"

$ docker images

$ docker ps

$ docker ps -a

**step-3:** Run hadoop in docker terminal:

$ docker pull macio232/hadoop-pseudo-distributed-mode

$ docker run -p 9870:9870 -p 8088:8088 -it --name=testHadoop macio232/hadoop-pseudo-distributed-mode

**step-4:** a console will open (for linux)

# ls

# cd home/

/home# cd hadoop/

/home/hadoop# vi student.txt

**Step-5:** write some thing to the student.txt file

Kabir 24

Bashar 25

Momin 26

Atik 24

Amir 25

type "esc -> : -> wq -> enter" for write and quite for save.

type "esc -> : -> q! -> enter" for quite without save.

**step-6:** now open a local browser and check the ports are active (port:127.0.0.1:9870 and port:127.0.0.1:8088)

**step-7:** create a folder on hadoop ecosystem.

/home/hadoop# hdfs dfs -mkdir /samrat/

/home/hadoop# hdfs dfs -put ‘/home/hadoop/student.txt’ /samrat

/home/hadoop# hive

hive> show databases;

hive> create database samrat-test;

hive> show databases;

hive> create database samrat\_test;

hive> use samrat\_test;

hive> show tables;

hive> create table student(Name string, Age int)

> Row format delimited

> Fields terminated by '\t';

hive> show tables;

hive> slect \* from student;

hive> load data inpath /samrat/student.txt into table student;

hive> slect \* from student;

hive>

**step-8:**

To close the the above docker window, open a new window and type

hive> docker stop testHadoop

step-9: To reopen the the closed docker window, type

hive> docker container start -i testHadoop;

**Guide to Managing Databases and Tables in Hive**

This guide reflects the step-by-step process of creating, managing, and querying databases and tables in Hive, including database creation, partitioning, table types, and data loading operations. It provides a clear and structured introduction to Hive database and table management tasks.

**Create Database Statement:**

hive> CREATE DATABASE IF NOT EXISTS Student;

**Verify the Databases List:**

hive> SHOW DATABASES;

default

Student

**To use The Database:**

hive> use Student;

**To See the Database Location:**

hive> DESCRIBE DATABASE EXTENDED student;

**Hive Table Types**

**Internal or Managed table:** You can drop the table with underlying data.

**External table:** You can drop an external table, only table metadata from Metastore will be removed but the underlying files will not be removed and still they can be accessed via HDFS commands, Spark or any other Hadoop compatible tools.

**Temporary table:** For temporary purpose.

**Transactional Table:** For transactional data purpose.

**Table Creation**

hive> CREATE External TABLE IF NOT EXISTS Student.Info(Id int,Name string,

Age int)

> PARTITIONED BY (Gender string)

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ','

> STORED AS TEXTFILE

> LOCATION '/data/output/';

**To See the Table Structure:**

hive> DESCRIBE student.info;

**Insert Data (single record) into Table:**

hive> INSERT INTO student.info PARTITION(Gender=’M’) values(7,'Maruf',23);

hive> INSERT INTO student.info PARTITION(Gender=’F’) values(8,'Rina',50);

**For Bulk Data (multiple record) Load:**

hive> LOAD DATA INPATH '/samrat/data.txt' INTO TABLE Student.Info

PARTITION(Gender=’F’);

hive> LOAD DATA INPATH '/path/to/HDFS/dir/file.csv' OVERWRITE INTO TABLE Student.Info PARTITION (Gender='M');

hive> LOAD DATA INPATH '/path/to/HDFS/dir/file.csv' OVERWRITE INTO TABLE Student.Info PARTITION (Gender='F');

**To Retrieves the all data:**

hive> SELECT \* FROM Info;

**Conditional Data Retrieve:**

hive> SELECT \* FROM Info WHERE age=23;

**To See All Partitions:**

hive> USE student;

hive> show partitions Info;

**To Drop a Partition:**

hive>ALTER TABLE Student.Info DROP PARTITION (gender="F");

**To Repair a Partition:**

hive> Msck repair table Student.Info;

# **Hadoop Word Cound using Docker**

Open the Docker terminal and run the following commands:

1. **Start the container:**

C:\Users\asus> docker run -p 9870:9870 -p 8088:8088 -it

--name=testHadoop macio232/hadoop-pseudo-distributed-mode

Or

C:\Users\asus> docker container start -i testHadoop

1. **Navigate to the Hadoop Data Directory**

root@2a78d9e418fb:/# cd /home/hadoop/data

1. **Confirm You Are in the Correct Directory**

root@2a78d9e418fb:/home/hadoop/data# pwd

1. **Create a .java file for word count**

root@2a78d9e418fb:/home/hadoop/data# vi WordCount.java

**Copy and paste the following java code to the WordCount.java file**

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

import java.util.StringTokenizer;

public class WordCount {

    public static class TokenizerMapper

         extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);

        private Text word = new Text();

        public void map(Object key, Text value, Context context

                ) throws IOException, InterruptedException {

            StringTokenizer itr = new StringTokenizer(value.toString());

            while (itr.hasMoreTokens()) {

                word.set(itr.nextToken());

                context.write(word, one);

            }

        }

    }

    public static class IntSumReducer

         extends Reducer<Text,IntWritable,Text,IntWritable> {

        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,

                           Context context

                           ) throws IOException, InterruptedException {

            int sum = 0;

            for (IntWritable val : values) {

                sum += val.get();

            }

            result.set(sum);

            context.write(key, result);

        }

    }

    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "word count");

        job.setJarByClass(WordCount.class);

        job.setMapperClass(TokenizerMapper.class);

        job.setCombinerClass(IntSumReducer.class);

        job.setReducerClass(IntSumReducer.class);

        job.setOutputKeyClass(Text.class);

        job.setOutputValueClass(IntWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));

        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);

    }

}

**Summary of Commands:**

**i**: Enter insert mode.

**Esc**: Exit insert mode (back to normal mode).

**:wq**: Save and quit.

**:q!**: Quit without saving

1. **Compile the Java Code:**

/home/hadoop/data# javac -classpath `hadoop classpath` -d . WordCount.java

1. **Package the compiled classes into a JAR file:**

/home/hadoop/data# jar cf wordcount.jar WordCount\*.class

1. **Create a directory for the input data inside**

/home/hadoop/data# mkdir input

1. **Create a sample text file:**

/home/hadoop/data# echo "Hello Hadoop Hello Docker" > input/file01.txt

1. **Put the input data into HDFS**

/home/hadoop/data# hdfs dfs -mkdir -p /user/hadoop/input   
/home/hadoop/data# hdfs dfs -put ./input/\* /user/hadoop/input/

1. **Run the Hadoop job using:**

/home/hadoop/data# hadoop jar wordcount.jar WordCount /user/hadoop/input /user/hadoop/output

1. **After the job completes, view the results:**

/home/hadoop/data# hdfs dfs -cat /user/hadoop/output/part-r-00000