



MGM's

Jawaharlal Nehru Engineering College Aurangabad

Affiliated to Dr.B.A.Technological University ,Lonere Maharashtra

ISO 9001:2015,140001:2015Certified,AICTE Approved

Department of Computer Science & Engineering

LAB MANUAL

Programme(UG/PG) : UG

Year : Final Year

Semester : VIII

Course Code : BTCOL707

Course Title : Big Data Analytics Laboratory

Prepared By

Mr. S. N. Jaiswal

Assistant Professor

Department of Computer Science & Engineering

FOREWORD

It is my great pleasure to present this laboratory manual for **Final year** engineering students for the subject of Big Data Analytics Laboratory

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

As you may be aware that MGM has already been awarded with ISO 9001:2015,140001:2015 certification and it is our endure to technically equip our students taking the advantage of the procedural aspects of ISO Certification.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

Dr. H. H. Shinde
Principal

LABORATORY MANUAL CONTENTS

This manual is intended for the Final year students of Computer Science & Engineering in the subject of Big Data Analytics. This manual typically contains practical/Lab Sessions related to Big Data Analytics covering various aspects related the subject to enhanced understanding.

Big Data Analytics provide students the idea of managing big data, analyzing it and carry out meaningful information from big data. The Goal of Big Data Analytics is to learn, use and develop simple application using Apache Hadoop ecosystem..

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Good Luck for your Enjoyable Laboratory Sessions

Mr.S.N.Jaiswal

Subject Teacher

Dr. Vijaya Musande
HOD

LIST OF EXPERIMENTS

Course Code: BTCOL707

Course Title: Big Data Analytics Laboratory

S.No	Name of the Experiment	Page No
1.	Basic CRUD operations in MongoDB. Design a sample database for any application	9
2.	Installation of Cassandra and hands-on practice of it.	14
3.	Cloudera setup and Installation using VM	19
4.	Setup and installation of Apache Hadoop.	26
5.	File management task in Apache Hadoop	34
6.	Map Reduce Paradigm	38
7.	Install, Deploy and configure Apache Spark. Develop application using Apache Spark.	40
8.	Application development using Apache PySpark	43
	Data Analytics using Apache Spark on Amazon food dataset.	-
9.	Install and Run HBase and then use HBase DDL and DML commands	46
10.	Install and configure Apache Kafka. Integrate of Apache kafka with Apache Spark	52
12.	Classification, clustering and frequent item set using Spark MLlib	-
13.	Design of simple project like tweet analysis, fraud detection or any other using different components of Apache Hadoop.	-

Note: Perform any 10 practical's

DOs and DON'Ts in Laboratory:

1. Make entry in the Log Book as soon as you enter the Laboratory.
2. All the students should sit according to their roll numbers starting from their left to right.
3. All the students are supposed to enter the terminal number in the log book.
4. Do not change the terminal on which you are working.
5. All the students are expected to get at least the algorithm of the program/concept to be implemented.
6. Strictly observe the instructions given by the teacher/Lab Instructor.
7. Do not disturb machine Hardware / Software Setup.

Instruction for Laboratory Teachers:

1. Submission related to whatever lab work has been completed should be done during the next lab session along with signing the index.
2. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.
3. Continuous assessment in the prescribed format must be followed.

MGM's



Jawaharlal Nehru Engineering College, Aurangabad

Department of Computer Science and Engineering

Vision of CSE Department

To develop computer engineers with necessary analytical ability and human values who can creatively design, implement a wide spectrum of computer systems for welfare of the society.

Mission of the CSE Department:

- 1.** Preparing graduates to work on multidisciplinary platforms associated with their professional position both independently and in a team environment.
- 2.** Preparing graduates for higher education and research in computer science and engineering enabling them to develop systems for society development.

Programme Educational Objectives

Graduates will be able to

- I.** To analyze, design and provide optimal solution for Computer Science & Engineering and multidisciplinary problems.
- II.** To pursue higher studies and research by applying knowledge of mathematics and fundamentals of computer science.
- III.** To exhibit professionalism, communication skills and adapt to current trends by engaging in lifelong learning.

Programme Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the broadest context of technological change.

LABORATORY OUTCOMES

The practical/exercises in this section are psychomotor domain Learning Outcomes (i.e. subcomponents of the COs), to be developed and assessed to lead to the attainment of the competency.

LO-1: Setup the Hadoop/Cloudera framework

LO-2: Perform file management task in Hadoop

LO-3: Design simple applications using Map Reduce API and Java/Python

LO-4: Implement CRUD operations in MongoDB / Cassandra

LO-5: Develop simple streaming applications using Apache Kafka

LO-6: Analyze the data using Spark Mlib

1. Lab Exercise

Exercise No 1: (2 Hours) – 1 Practical

Aim: - Basic CRUD operations in MongoDB. Design a sample database for any application

Objectives:

1. Student will be able to install MongoDB on either Windows or Linux operating system.
2. Students should be able to design database for any application using MongoDB and perform CRUD operations on it.
3. Student will be able to use MongoDB through Java / Python or any other programming framework.

THEORY:

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document. MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++.

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data

Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair

```

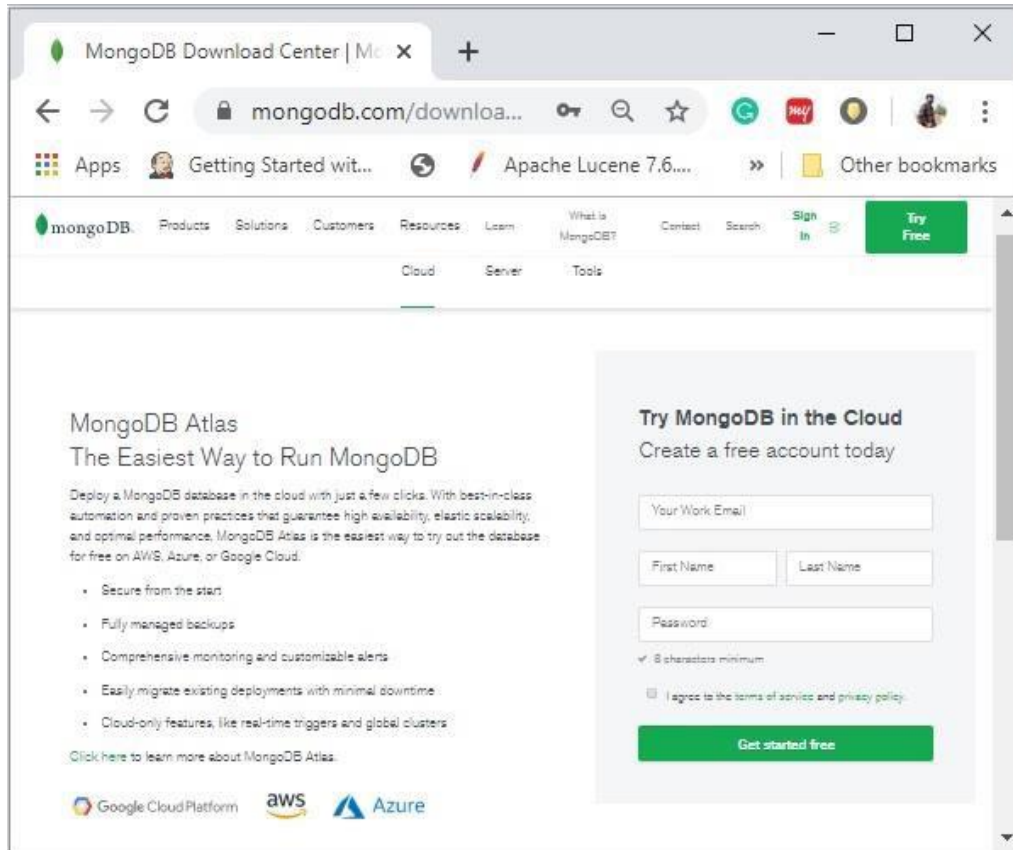
{
  _id:ObjectId(7df78ad8902c)
  title:'MongoDB Overview',
  description:'MongoDB is no sql database',
  by:'Open Source Community',
  url: 'https://www.mongodb.com/',
  tags:['mongodb','database','NoSQL'],
  likes:100,
  comments:[
    {
      user:'user1',
      message:'My first comment',
      dateCreated:new Date(2011,1,20,2,15),
      like:0
    },
    {
      user:'user2',
      message:'My second comments',
      dateCreated:new Date(2011,1,25,7,45),
      like:5
    }
  ]
}

```

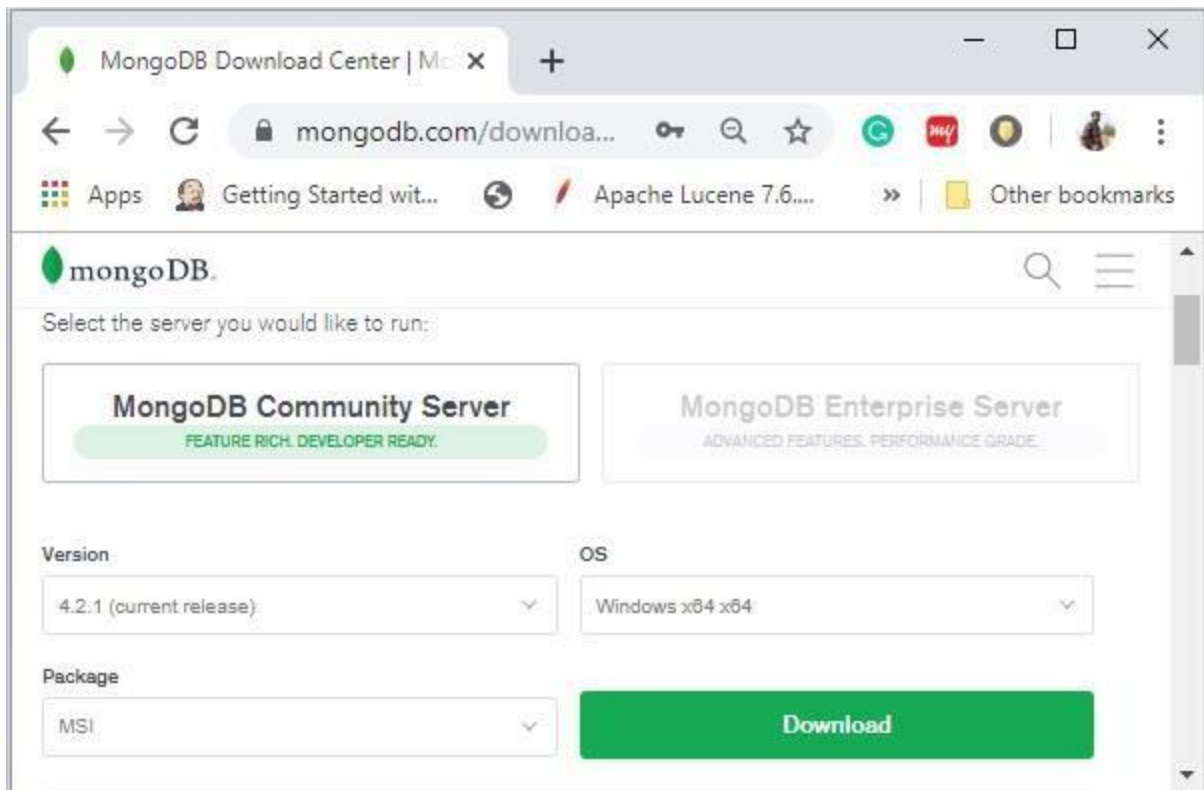
_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide **_id** while inserting the document. If you don't provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

Installation of mongoDB on Windows system:

To install MongoDB on Windows, first download the latest release of MongoDB from <https://www.mongodb.com/download-center>



Enter the required details, select the **Server** tab, in it you can choose the version of MongoDB, operating system and, packaging as:



Now install the downloaded file, by default, it will be installed in the folder **C:\Program Files**.

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is `c:\data\db`. So you need to create this folder using the Command Prompt.

Then you need to specify set the **dbpath** to the created directory in **mongod.exe**. For the same, issue the following commands.

In the command prompt, navigate to the bin directory current in the MongoDB installation folder. Suppose my installation folder is **C:\Program Files\MongoDB**. Run the command as ***mongod.exe --dbpath "C:\data"***

This will show **waiting for connections** message on the console output, which indicates that the **mongod.exe** process is running successfully.

Now to run the MongoDB, you need to open another command prompt and issue the following command on command prompt

C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe

System shows MongoDB prompt >

MongoDB : Basic Commands

- `db.help()` List of commands
- `db.stats()` Statistics of MongoDB
- `use DATABASE_NAME` create database
 - `use myDb` create database with name myDb
 - `db` Check the current database
- `show dbs` Show the list of database in system

Inserting record/document in database

- `db.collection.insertOne(...)`
- `db.collection.insertMany(...)`
- `db.collection.insert()`

db.student.insert(

```
{  
  name: "anilkumar",  
  roll: 406175,  
  dob: "15 Aug 1999",  
  hobby: ["reading", "Swimming"],  
  address :  
{area: "Gulmandi", city: "Aurangabad", pin: 431001}  
}  
)
```

Outcome:

To learn the installation process of mongoDB and perform basic operations on database and collections.

CONCLUSIONS:

By following above Steps students will able to install mongoDB on Windows. Students will be able to develop schema for specified application using mongoDB and perform basic operations on it.

2. Lab Exercise

Exercise No 2: (2 Hours) – 1 Practical

Aim: -Installation of Cassandra and hands-on practice of it

Objectives:

1. Student will able to install Cassandra
2. Students should able to perform operations by using different commands.
3. Student will able to use different data types

THEORY:

Cassandra is a distributed database from Apache. Highly scalable and designed to manage very large amounts of structured data. Provides high availability with no single point of failure. Created at Facebook, it differs sharply from relational database management systems. It was open-sourced by Facebook in July 2008. Cassandra was accepted into Apache Incubator in March 2009. It was made an Apache top-level project since February 2010

It is a column-oriented database. Its distribution design is based on Amazon's Dynamo and its data model on Google's Bigtable. Cassandra implements a Dynamo-style replication model with no single point of failure, but adds a more powerful "column family" data model. Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Cassandra is an open-source distributed database software for handling NoSQL databases. CQL (Cassandra Query Language) is used. CQL keeps data in tables arranged in a set of rows with columns that 6key-value pairs. CQL tables are grouped in data containers called keyspaces in Cassandra. Data stored in one keyspace is unrelated to other data in the cluster.

Features of Cassandra:

1. Elastic scalability – it allows to add more hardware to accommodate more customers and more data as per requirement.
2. Always on architecture – Cassandra has no single point of failure
3. Fast linear-scale performance – Cassandra is linearly scalable
4. Flexible data storage – accommodates all data formats

5. Easy data distribution – provides the flexibility to distribute data where you need by replicating data across multiple data centers.
6. Transaction support – supports ACID properties.
7. Fast writes – It performs fast writes and can store hundreds of terabytes of data.

Installation steps to be followed:(Windows)

1. Download and Install Java 8 and set environment variables.
2. Download and install Python 2.7 and set environment variables.
3. Download and Extract Cassandra tar.gz Folder
4. Set CASSANDRA_HOME & add bin folder to path
5. Start Cassandra from Windows CMD
6. Access Cassandra cqlsh from Windows CMD

Apache Cassandra Data Types

1. Built-in data types
2. Collection data types
3. User-defined data types

1. Built In Data Types:

Data Type	Constants	Description
Ascii	Strings	includes character encoding used for strings.
Boolean	Booleans	Stored as 16-bit numbers
Blob	Blobs	“Binary Large Object” and it is utilized for storing binary data.
decimal	integers, floats	Offers precision & scale
double	Integers	It represents a 64-bit floating point
Float	integers, floats	The FLOAT data type stores decimal point values.
Int	Integers	used to store 32-bit signed integers.
smallint	Integers	stores 16-bit signed integers.
bigint	Integers	BIGINT stores 64-bit signed integers.
Text	Strings	use TEXT data types used for text data, represented in UTF8 encoded strings.
varchar	Strings	Use VARCHAR for variables or arbitrary characters
Inet	Strings	Use it to save and manage IP addresses since it supports both numeric and character representation.
counter	Integers	This data type supports two operations: incrementing and decrementing
Time	integers, strings	You can store time values in the following format: hh:mm:ss using the time data type.

Date	integers, strings	store date values in the format: YYYY-MM-DD.
timestamp	integers, strings	combination of time & data

2. Collection data types

Maps	Cassandra can store data in sets of key-value pairs using the <i>Map</i> data type.
Sets	You can store multiple unique values, using the <i>Set</i> data type.
Lists	If you need to store multiple values in a specific order, you can use the <i>List</i> data type.

3. User Defined Data types

User-Defined data types (UDTs) allows to create your own data type based on the requirements you need. A UDT consists of multiple data fields of any data type inside a single column. Once you create your user-defined data type, you can change or even remove the fields inside of it

Shell Commands:

1. HELP – Displays help topics for all cqlsh commands.
2. CAPTURE – Captures the output of a command and adds it to a file.
3. CONSISTENCY – Shows the current consistency level, or sets a new consistency level.
4. COPY – Copies data to and from Cassandra.
5. DESCRIBE – Describes the current cluster of Cassandra and its objects.
6. EXPAND – Expands the output of a query vertically.
7. EXIT – Using this command, you can terminate cqlsh.
8. PAGING – Enables or disables query paging.
9. SHOW – Displays the details of current cqlsh session such as Cassandra version, host, or data type assumptions.
10. SOURCE – Executes a file that contains CQL statements.
11. TRACING – Enables or disables request tracing.

- **Create keyspace**

Syntax:

Create keyspace KeyspaceName *with replication*={ 'class':strategy name, 'replication_factor': No of replications on different nodes};

Example:

Create keyspace Student *with replication*={ 'class': 'SimpleStrategy', 'replication_factor': 1};

- **Selecting Keyspace for Cassandra Table**

Syntax:

USE *keyspace_name*;

or

Specify the Keyspace Name in the Query

```
CREATE TABLE keyspace_name.table_name
```

Example:

```
CREATE TABLE Super_KeySpace.student;
```

1. Commands using In Built Data Type

How to Create Cassandra Table

Syntax:

```
CREATE TABLE tableName (  
    columnName1 dataType,  
    columnName2 dataType,  
    columnName2 dataType PRIMARY KEY  
    (columnName) );
```

example:

```
CREATE TABLE suppliers (  
    supp_id int PRIMARY KEY,  
    supp_city text,  
    supp_email text,  
    supp_fee int,  
    supp_name text,  
    supp_phone int );
```

Updating Data in a Table

```
UPDATE <tablename> SET <column name> = <new value><column  
name> = <value>.... WHERE <condition>
```

Deleting Data from a Table

```
DELETE FROM <identifier> WHERE <condition>;
```

Deleting an Entire Row

```
DELETE FROM emp WHERE emp_id=3;
```

1. Commands using List Data Type

Creating a Table with List

```
CREATE TABLE data(name text PRIMARY KEY, email list<text>);
```

Inserting Data into a List

```
INSERT INTO data(name, email) VALUES  
('ramu',['abc@gmail.com','cba@yahoo.com'])
```

Updating a List

```
UPDATE data SET email = email + ['xyz@tutorialspoint.com'] where  
name = 'ramu'
```

2. Commands using Set Data Type

Creating a Table with Set

```
CREATE TABLE data2 (name text PRIMARY KEY, marks set<varint>);
```

Inserting Data into a Set

```
INSERT INTO data2(name, marks)VALUES ('roshan', {38,39});
```

Updating a Set

```
UPDATE data2 SET marks = marks + {33} where name = 'roshan';
```

3. Commands using Map Data Type

Creating a Table with Map

```
CREATE TABLE data3 (name text PRIMARY KEY, address map<timestamp,  
text>);
```

Inserting Data into a Map

```
INSERT INTO data3 (name, address) VALUES ('robin', {'home': 'hyderabad' ,  
'office': 'Delhi' } );
```

Updating a Map

```
UPDATE data3 SET address = address + {'office':'mumbai'} WHERE name = 'robin';
```

CONCLUSIONS:

By following above Steps students will able to install Cassandra on Windows. Students will be able to perform basic operations using data types and commands of Cassandra.

Additional Task :

1. **Install cassandra on Linux operating system**
2. **Design user interface for cassandra using either Java or Python.**
3. **Compare and contrast NoSQL with SQL**

3. Lab Exercise

Exercise No 3: (2 Hours) – 1 Practical

Aim: - Cloudera setup and Installation using VM

Objectives:

1. Students should able to install Cloudera using VM
2. Students should able to use HDFS file system using Cloudera
3. Student should able to use Map-Reduce paradigm from Cloudera

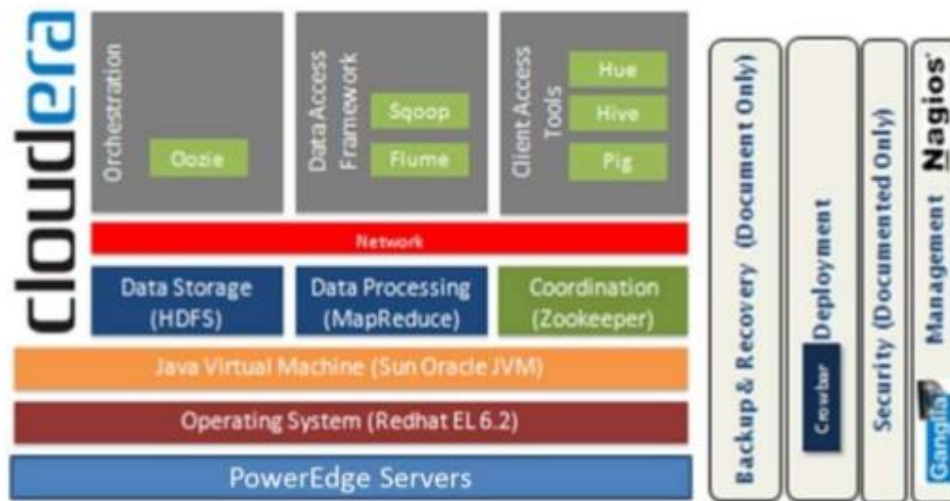
THEORY:

Cloudera, Inc. is a US-based software company that provides a software platform for data engineering, data warehousing, machine learning and analytics that runs in the cloud or on premises.

Cloudera started as a hybrid open-source Apache Hadoop distribution, CDH (Cloudera Distribution Including Apache Hadoop), that targeted enterprise-class deployments of that technology. Cloudera states that more than 50% of its engineering output is donated upstream to the various Apache-licensed open source projects (Apache Spark, Apache Hive, Apache Avro, Apache HBase, and so on) that combine to form the Apache Hadoop platform. Cloudera is also a sponsor of the Apache Software Foundation. Following diagram shows taxonomy of Cloudera Distribution Including Apache Hadoop.

CLUDERA HADOOP ARCHITECTURE

- Cloudera solution taxonomy



Installation steps to be followed: (Windows)

You can install CDH4 in any of the following ways:

1. Installing using Cloudera quickstart vm.
2. Automated method using Cloudera Manager. Cloudera Manager Free Edition automates the installation and configuration of CDH4 on an entire cluster if you have root or password less sudo SSH access to your cluster's machines.
3. Manual methods described below: -
 - a. Download and install the CDH4 "1-click Install" package
 - b. Add the CDH4 repository
 - c. Build your own CDH4 repository
 - d. Install from a CDH4 tarball

Downloading and Installing the Cloudera VM Instructions

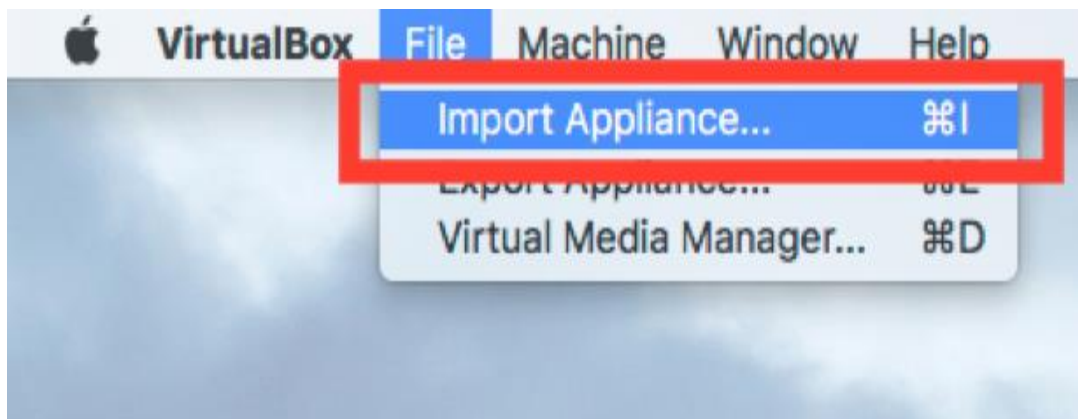
Hardware Requirements:

- A. Quad Core Processor (VT-x or AMD-V support recommended), 64-bit;
- B. 8 GB RAM
- C. 20 GB disk free.
- D. A high speed internet connection because you will be downloading files up to 4 Gb in size.

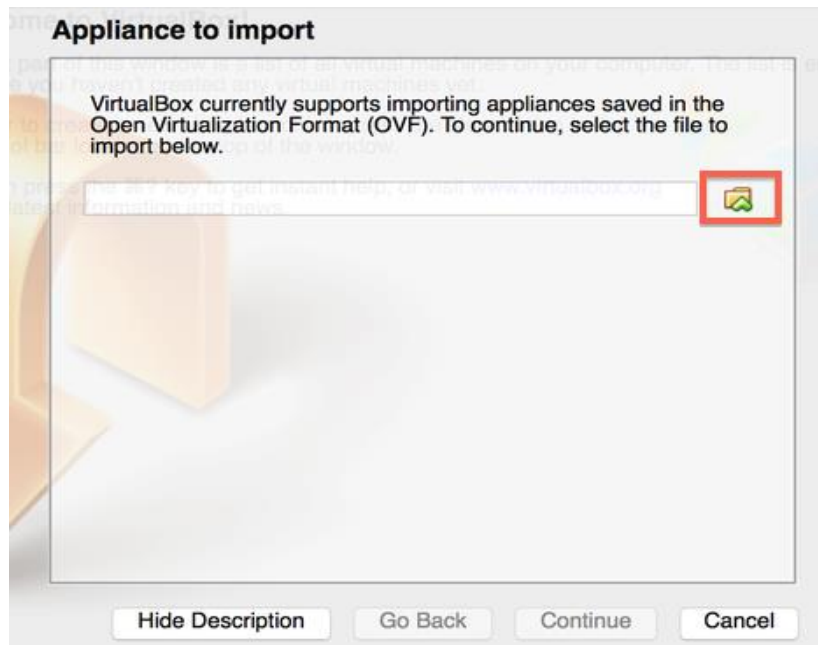
Instructions

Please use the following instructions to download and install the Cloudera Quickstart VM with VirtualBox.. The screenshots are from a Mac but the instructions should be the same for Windows.

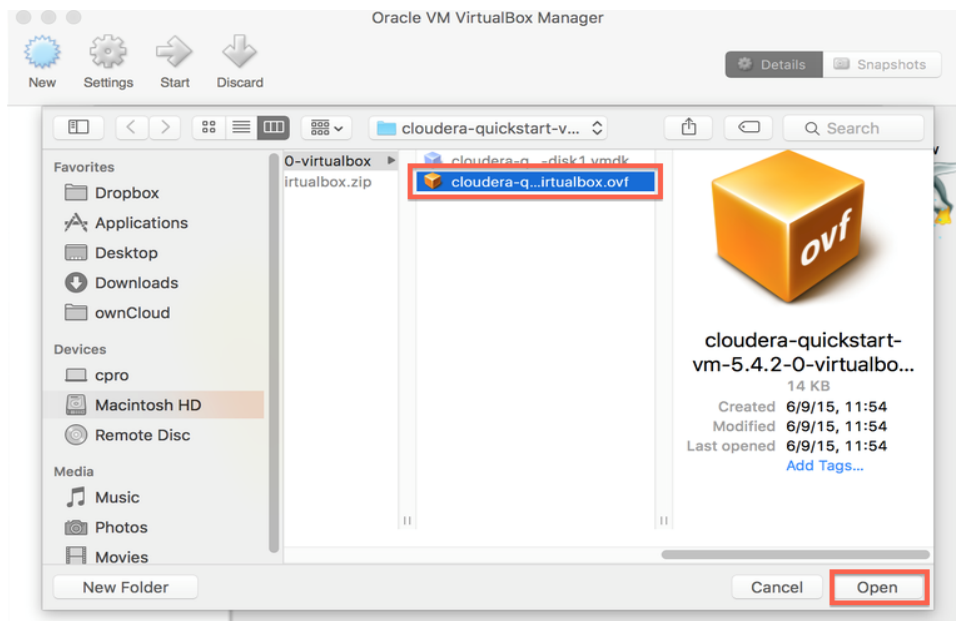
1. **Install VirtualBox.**
 - a. Go to <https://www.virtualbox.org/wiki/Downloads> to download and install VirtualBox for your computer.
2. **Download the Cloudera VM.**
 - a. Download the Cloudera VM from https://downloads.cloudera.com/demo_vm/virtualbox/cloudera-quickstart-vm-5.4.2-0-virtualbox.zip. The VM is over 4GB, so will take some time to download.
3. **Unzip the Cloudera VM:**
 - a. On Windows: Right-click cloudera-quickstart-vm-5.4.2-0-virtualbox.zip and select “Extract All...”
4. **Start VirtualBox.**
5. **Begin importing.**
 - a. Import the VM by going to File -> Import Appliance



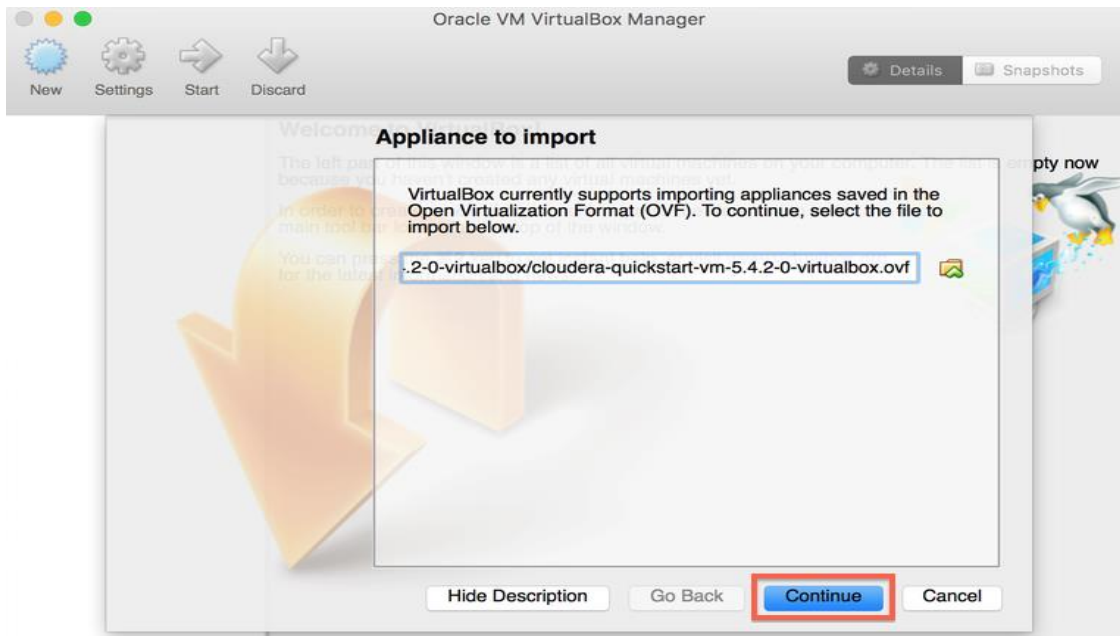
6. Click the Folder icon.



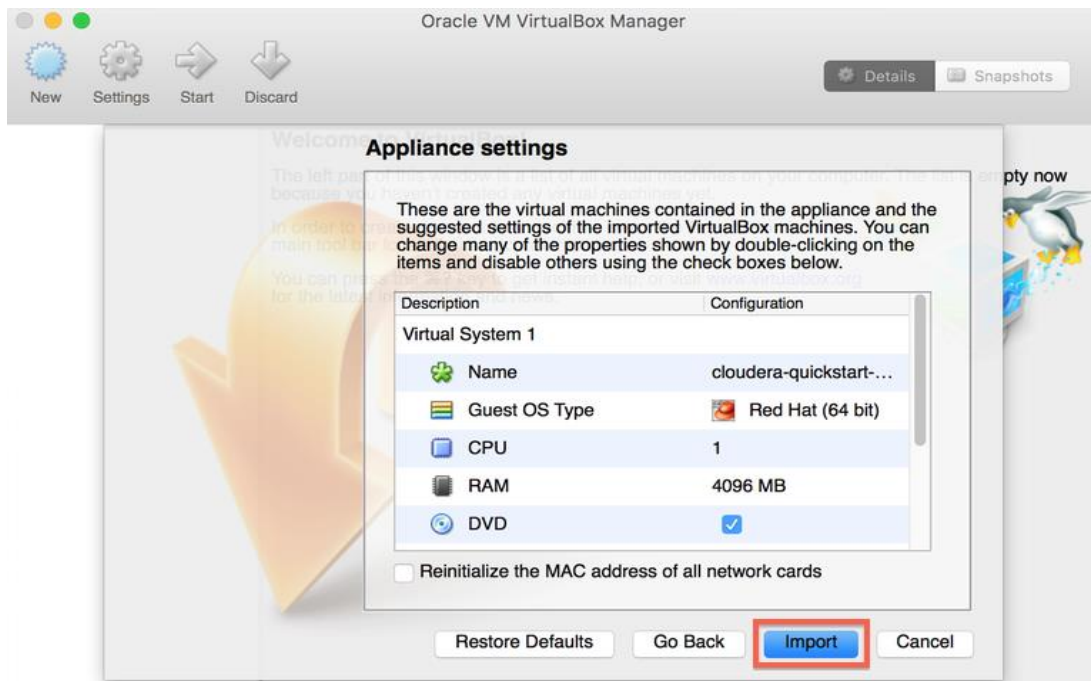
7. Select the **cloudera-quickstart-vm-5.4.2-0-virtualbox.ovf** from the Folder where you unzipped the VirtualBox VM and click Open.



8. Click **Continue** to proceed.



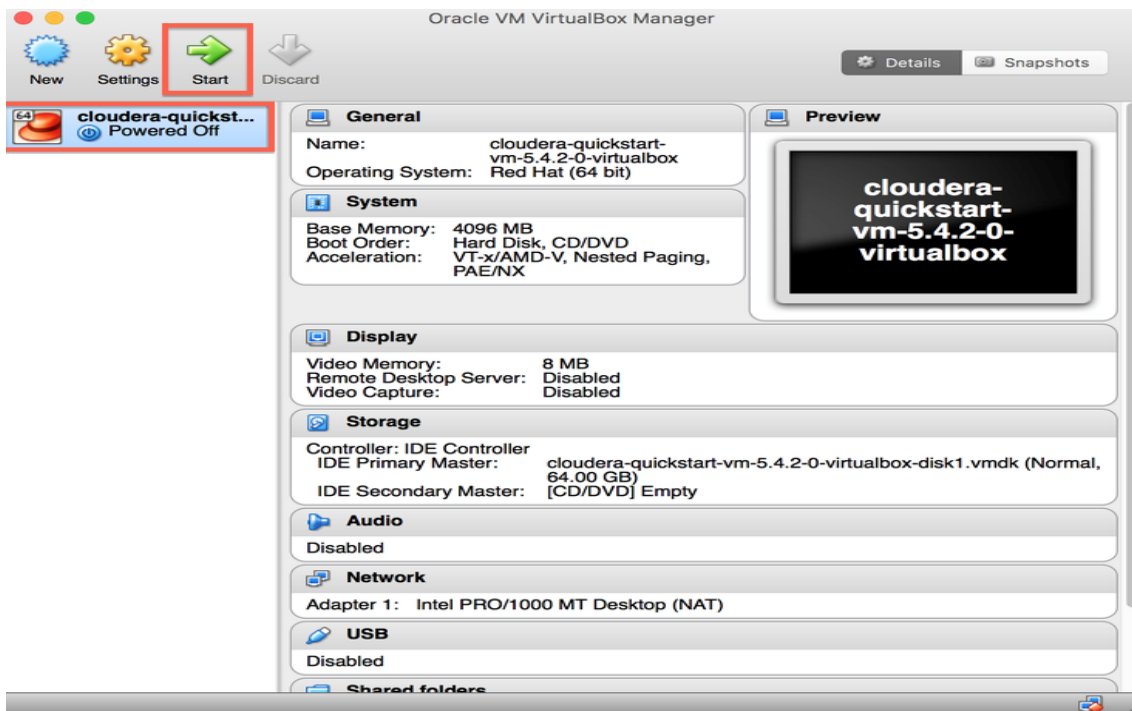
9. Click Import.



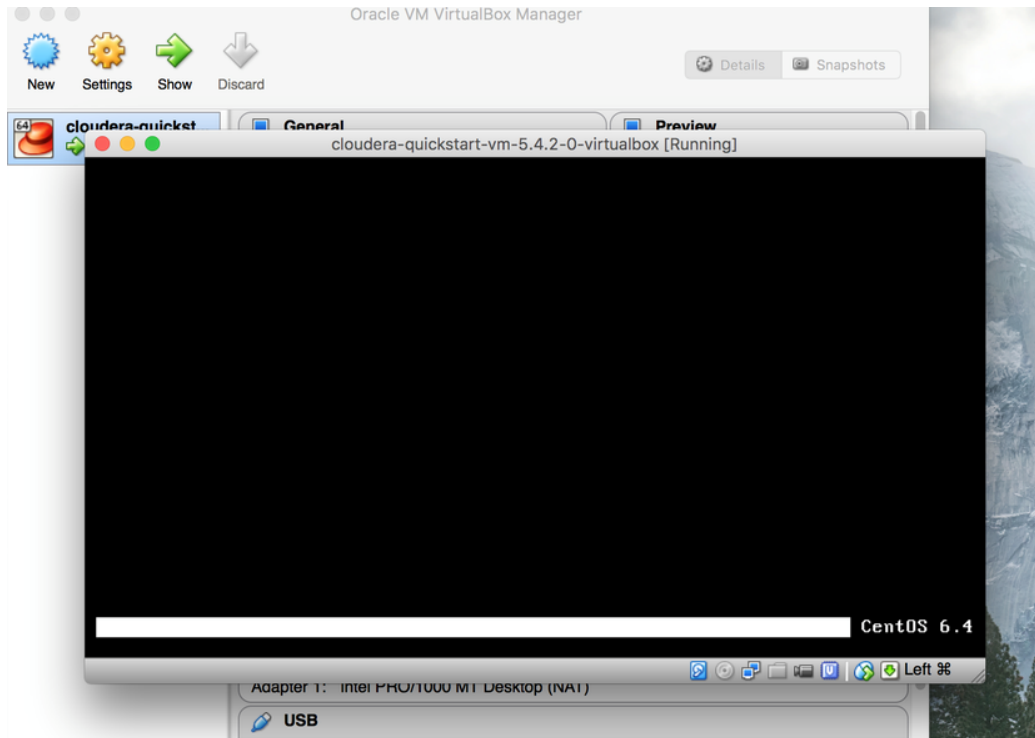
10. The virtual machine image will be imported. This can take several minutes.



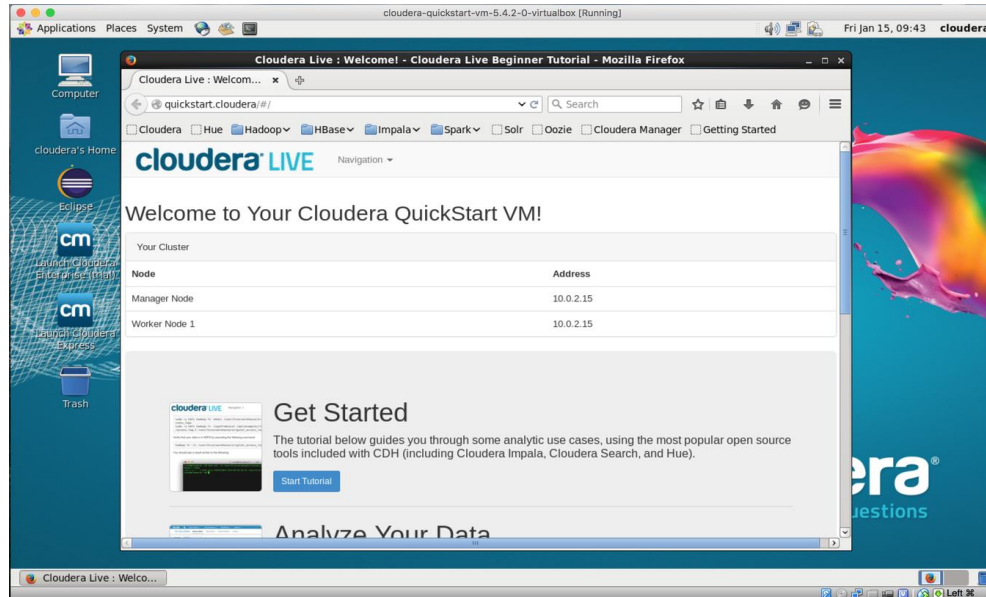
11. **Launch Cloudera VM.** When the importing is finished, the quickstart-vm-5.4.2-0 VM will appear on the left in the VirtualBox window. Select it and click the Start button to launch the VM.



12. **Cloudera VM booting.** It will take several minutes for the Virtual Machine to start. The booting process takes a long time since many Hadoop tools are started.



13. **The Cloudera VM desktop.** Once the booting process is complete, the desktop will appear with a browser.



CONCLUSIONS:

In this Practical we learned how to install and enable Cloudera Distribution Including Apache Hadoop using VM.

4. Lab Exercise

Exercise No 4: (2 Hours) – 1 Practical

Aim: - Setup and installation of Apache Hadoop on Windows.

THEORY:

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.



Prerequisites :

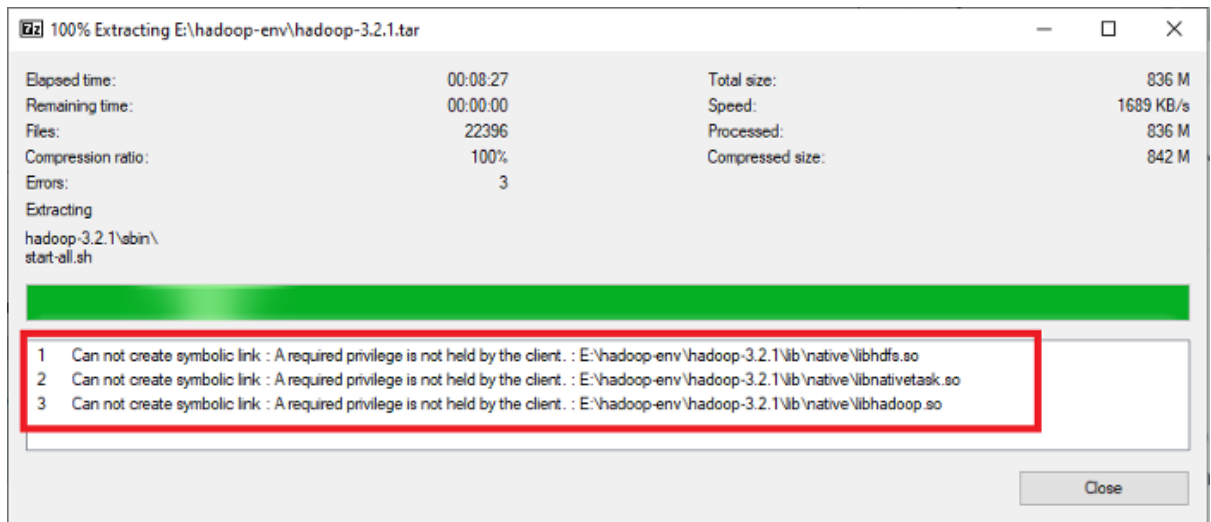
1. Java 8 runtime environment (JRE): [Hadoop 3 requires a Java 8 installation.](#)
2. [Java 8 development Kit \(JDK\)](#)
3. To unzip downloaded Hadoop binaries,
4. Create a folder “D:\hadoopSoft” on local machine to store downloaded files.

Download Hadoop binaries

The first step is to download Hadoop binaries from the [official website](#). The binary package size is about 342 MB. After finishing the file download, unpack the package in two steps. First, extract the hadoop-3.2.1.tar.gz library, and then, unpack the extracted tar file

The tar file extraction may take some minutes to finish. In the end, you may see some warnings about symbolic link creation. Just ignore these warnings since they are not related to windows.

Name	Date modified	Type	Size
 hadoop-3.2.1.tar	9/10/2019 8:11 PM	TAR File	893,250 KB
 hadoop-3.2.1.tar.gz	4/15/2020 8:52 PM	GZ File	350,779 KB

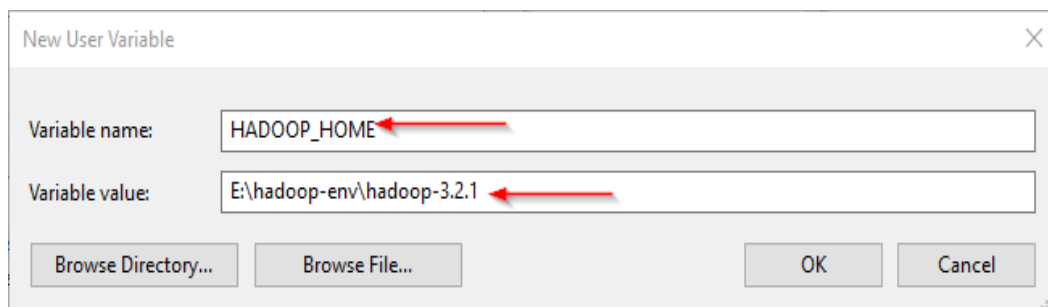
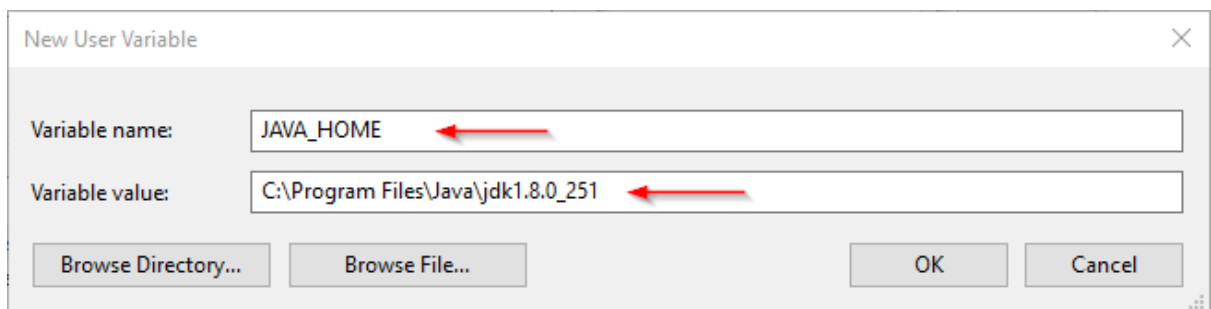


Setting up environment variables

After installing Hadoop and its prerequisites, we should configure the environment variables to define Hadoop and Java default paths.

There are two variables to define:

1. JAVA_HOME: JDK installation folder path
2. HADOOP_HOME: Hadoop installation folder path



Configuring Hadoop cluster

There are four files we should alter to configure Hadoop cluster:

1. %HADOOP_HOME%\etc\hadoop\hdfs-site.xml
2. %HADOOP_HOME%\etc\hadoop\core-site.xml
3. %HADOOP_HOME%\etc\hadoop\mapred-site.xml
4. %HADOOP_HOME%\etc\hadoop\yarn-site.xml

HDFS site configuration

Hadoop is built using a master-slave paradigm. Before altering the HDFS configuration file, create a directory to store all master node (name node) data and another one to store data (data node). In this example, created the following directories:

- D:\hadoop-soft\hadoop-3.2.1\data\dfs\namenode
- D:\hadoop-soft\hadoop-3.2.1\data\dfs\datanode

Open “hdfs-site.xml” file located in “%HADOOP_HOME%\etc\hadoop” directory, and add the following properties within the <configuration></configuration> element:

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>1</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.namenode.name.dir</name>
```

```
<value>file:///D:/hadoop-soft/hadoop-3.2.1/data/dfs/namenode</value>
```

```
</property>
```

```
<property>
```

```
<name>dfs.datanode.data.dir</name>
```

```
<value>file:///D:/hadoop-soft/hadoop-3.2.1/data/dfs/datanode</value>
```

```
</property>
```

Core site configuration

Now, configure the name node URL adding the following XML code into the `<configuration></configuration>` element within “core-site.xml”:

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9820</value>
</property>
```

Map Reduce site configuration

Now, add the following XML code into the `<configuration></configuration>` element within “mapred-site.xml”:

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
<description>MapReduce framework name</description>
</property>
```

Yarn site configuration

Now, add the following XML code into the `<configuration></configuration>` element within “yarn-site.xml”:

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
<description>Yarn Node Manager Aux Service</description>
</property>
```

Formatting Name node

After finishing the configuration, format the name node using the following command:

```
hdfs namenode -format
```

```

2020-04-17 22:14:17,206 INFO namenode.FSImage: Allocated new BlockPoolId: BP-2032026115-192.168.1.105-1587150857190
2020-04-17 22:14:17,207 INFO common.Storage: Will remove files: []
2020-04-17 22:14:17,275 INFO common.Storage: Storage directory E:\hadoop-env\hadoop-3.2.1\data\dfs\namenode has been successfully formatted.
2020-04-17 22:14:17,331 INFO namenode.FSImageFormatProtobuf: Saving image file E:\hadoop-env\hadoop-3.2.1\data\dfs\namenode\current\fsimage.ckpt_000000000000000000 using no compression
2020-04-17 22:14:17,531 INFO namenode.FSImageFormatProtobuf: Image file E:\hadoop-env\hadoop-3.2.1\data\dfs\namenode\current\fsimage.ckpt_000000000000000000 of size 400 bytes saved in 0 seconds.
2020-04-17 22:14:17,555 INFO namenode.NNStorageRetentionManager: Going to retain 1 images with txid >= 0
2020-04-17 22:14:17,580 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 when meet shutdown.
2020-04-17 22:14:17,580 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at
*****/
PS C:\Windows\system32>

```

Starting Hadoop services

Now, open command prompt , and navigate to “%HADOOP_HOME%\sbin” directory. Then the following command to start the Hadoop nodes:

./start-dfs.cmd

Two command prompt windows will open (one for the name node and one for the data node) as follows:

```

Apache Hadoop Distribution - hadoop namenode
2020-04-17 22:44:54,087 INFO namenode.FSDirectory: Initializing quota with 4 threads
2020-04-17 22:44:54,115 INFO namenode.FSDirectory: Quota initialization completed in 27 milliseconds
Name spaces=1
storage space=0
storage types=RAM_DISK=0, SSD=0, DISK=0, ARCHIVE=0, PROVIDED=0
2020-04-17 22:44:54,151 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval 30000 milliseconds
2020-04-17 22:44:55,147 INFO hdfs.StateChange: BLOCK* registerDatanode: from DatanodeRegistration(127.0.0.1:9866, datanodeUId=94e235fa-78fd-413e-95e5-5d84dc62bc9f, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-11d9a063-fc7b-4208-b192-2e4b6bf8736f;nsid=660255427;c=1587150857190) storage 94e235fa-78fd-413e-95e5-5d84dc62bc9f
2020-04-17 22:44:55,152 INFO net.NetworkTopology: Adding a new node: /default-rack/127.0.0.1:9866
2020-04-17 22:44:55,153 INFO blockmanagement.BlockReportLeaseManager: Registered Datanode 94e235fa-78fd-413e-95e5-5d84dc62bc9f (127.0.0.1:9866).
2020-04-17 22:44:55,353 INFO blockmanagement.DatanodeDescriptor: Adding new storage report to DS-debef9eb-f03b-40b4-8bdd-dd36b16ee068 for DN 127.0.0.1:9866
2020-04-17 22:44:55,473 INFO block.StateChange: BLOCK* processReport 0x6718670216280c90: Processing first storage report for DS-debef9eb-f03b-40b4-8bdd-dd36b16ee068 from datanode 94e235fa-78fd-413e-95e5-5d84dc62bc9f
2020-04-17 22:44:55,478 INFO block.StateChange: BLOCK* processReport 0x6718670216280c90: from storage DS-debef9eb-f03b-40b4-8bdd-dd36b16ee068 node DatanodeRegistration(127.0.0.1:9866, datanodeUId=94e235fa-78fd-413e-95e5-5d84dc62bc9f, infoPort=9864, infoSecurePort=0, ipcPort=9867, storageInfo=lv=-57;cid=CID-11d9a063-fc7b-4208-b192-2e4b6bf8736f;nsid=660255427;c=1587150857190), blocks: 0, hasStaleStorage: false, processing time: 5 msecs, invalidatedBlocks: 0

Apache Hadoop Distribution - hadoop datanode
2020-04-17 22:44:55,000 INFO impl.FsDatasetImpl: Total time to add all replicas to map for block pool BP-2032026115-192.168.1.105-1587150857190: 6ms
2020-04-17 22:44:55,013 INFO datanode.VolumeScanner: Now scanning bpid BP-2032026115-192.168.1.105-1587150857190 on volume E:\hadoop-env\hadoop-3.2.1\data\dfs\datanode
2020-04-17 22:44:55,016 INFO datanode.VolumeScanner: VolumeScanner(E:\hadoop-env\hadoop-3.2.1\data\dfs\datanode, DS-debef9eb-f03b-40b4-8bdd-dd36b16ee068): finished scanning block pool BP-2032026115-192.168.1.105-1587150857190
2020-04-17 22:44:55,059 INFO datanode.DirectoryScanner: Periodic Directory Tree Verification scan starting at 4/18/20 1:51 AM with interval of 21600000ms
2020-04-17 22:44:55,069 INFO datanode.VolumeScanner: VolumeScanner(E:\hadoop-env\hadoop-3.2.1\data\dfs\datanode, DS-debef9eb-f03b-40b4-8bdd-dd36b16ee068): no suitable block pools found to scan. Waiting 1814399943 ms.
2020-04-17 22:44:55,075 INFO datanode.DataNode: Block pool BP-2032026115-192.168.1.105-1587150857190 (Datanode UId 94e235fa-78fd-413e-95e5-5d84dc62bc9f) service to localhost/127.0.0.1:9820 beginning handshake with NN
2020-04-17 22:44:55,182 INFO datanode.DataNode: Block pool BP-2032026115-192.168.1.105-1587150857190 (Datanode UId 94e235fa-78fd-413e-95e5-5d84dc62bc9f) service to localhost/127.0.0.1:9820 successfully registered with NN
2020-04-17 22:44:55,183 INFO datanode.DataNode: For namenode localhost/127.0.0.1:9820 using BLOCKREPORT_INTERVAL of 21600000msec CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartbeatInterval=3000
2020-04-17 22:44:55,555 INFO datanode.DataNode: Successfully sent block report 0x6718670216280c90, containing 1 storage report(s), of which we sent 1. The reports had 0 total blocks and used 1 RPC(s). This took 12 msec to generate and 129 msecs for RPC and NN processing. Got back one command: FinalizeCommand/s.
2020-04-17 22:44:55,556 INFO datanode.DataNode: Got finalize command for block pool BP-2032026115-192.168.1.105-1587150857190

```

Next, start the Hadoop Yarn service using the following command:

./start-yarn.cmd

Two command prompt windows will open (one for the resource manager and one for the node manager) as follows:

```
2020-04-17 22:47:05,659 INFO store.AbstractFSNodeStore: Finished create editlog file at: file:/tmp/hadoop-yarn-HFadl/node-attribute/nodeattribute.editlog
2020-04-17 22:47:05,695 INFO event.AsyncDispatcher: Registering class org.apache.hadoop.yarn.server.resourcemanager.nodelabels.NodeAttributesStoreEventType for class org.apache.hadoop.yarn.server.resourcemanager.nodelabels.NodeAttributesManagerImpl
2020-04-17 22:47:05,740 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-04-17 22:47:05,765 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.spl.ResourceTrackerPB to the server
2020-04-17 22:47:05,799 INFO ipc.Server: Starting Socket Reader #1 for port 8031
2020-04-17 22:47:05,825 INFO ipc.Server: IPC Server listener on 8031: starting
2020-04-17 22:47:05,820 INFO ipc.Server: IPC Server Responder: starting
2020-04-17 22:47:05,847 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2020-04-17 22:47:05,853 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-04-17 22:47:05,929 INFO ipc.Server: Starting Socket Reader #1 for port 8030
2020-04-17 22:47:05,956 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.spl.ApplicationMasterProtocolPB to the server
2020-04-17 22:47:06,007 INFO ipc.Server: IPC Server listener on 8030: starting
2020-04-17 22:47:06,000 INFO ipc.Server: IPC Server Responder: starting
2020-04-17 22:47:06,263 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queueCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-04-17 22:47:06,288 INFO ipc.Server: Starting Socket Reader #1 for port 8032
2020-04-17 22:47:06,298 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.spl.ApplicationClientProtocolPB to the server
2020-04-17 22:47:06,320 INFO resourceManager.ResourceManager: Transitioned to active state
2020-04-17 22:47:06,331 INFO ipc.Server: IPC Server Responder: starting
2020-04-17 22:47:06,333 INFO ipc.Server: IPC Server listener on 8032: starting
2020-04-17 22:47:06,961 INFO resourceManager.ResourceTrackerService: NodeManager from node DESKTOP-SSVATPQ (id: 57849 httpPort: 8042) registered with capability: <memory:8192, vCores:8>, assigned nodeId DESKTOP-SSVATPQ:57849
2020-04-17 22:47:06,970 INFO rmmode.RMNodeImpl: DESKTOP-SSVATPQ:57849 Node Transited from NEW to RUNNING
2020-04-17 22:47:07,016 INFO capacity.CapacityScheduler: Added node DESKTOP-SSVATPQ:57849 clusterResource: <memory:8192, vCores:8>
Apr 17, 2020 10:47:04 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory register
INFO: Registering org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver as a provider class
Apr 17, 2020 10:47:04 PM com.sun.jersey.server.impl.application.WebApplicationImpl._initiate
INFO: Initiating Jersey application, version 'Jersey: 1.19 02/11/2015 03:25 AM'
Apr 17, 2020 10:47:04 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver to GuiceManagedComponentProvider with the scope "Singleton"
Apr 17, 2020 10:47:04 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.webapp.GenericExceptionHandler to GuiceManagedComponentProvider with the scope "Singleton"
Apr 17, 2020 10:47:06 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentProvider
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.NMWebServices to GuiceManagedComponentProvider with the scope "Singleton"
2020-04-17 22:47:06,268 INFO handler.ContextHandler: Started o.e.j.w.WebAppContext@716309a9[/,file:///C:/Users/HFadl/AppData/Local/Temp/jetty-0.0.0.0-8042-node-_-any-3353495707710539143.dir/webapp/,AVAILABLE][node]
2020-04-17 22:47:06,284 INFO server.AbstractConnector: Started ServerConnector@406f3a8[HTTP/1.1,(http://1.1)](0.0.0.0:8042)
2020-04-17 22:47:06,285 INFO server.Server: Started @13636ms
2020-04-17 22:47:06,286 INFO webapp.WebApps: Web app node started at 8042
2020-04-17 22:47:06,289 INFO nodemanager.NodeStatusUpdaterImpl: Node ID assigned is : DESKTOP-SSVATPQ:57849
2020-04-17 22:47:06,319 INFO client.NMProxy: Connecting to ResourceManager at /0.0.0.0:8031
2020-04-17 22:47:06,327 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2020-04-17 22:47:06,448 INFO nodemanager.NodeStatusUpdaterImpl: Sending out 0 NM container statuses: []
2020-04-17 22:47:06,476 INFO nodemanager.NodeStatusUpdaterImpl: Registering with RM using containers: []
2020-04-17 22:47:06,999 INFO security.NMContainerTokenSecretManager: Rolling master-key for container-tokens, got key with id -611578912
2020-04-17 22:47:07,001 INFO security.NMTokenSecretManagerInNM: Rolling master-key for container-tokens, got key with id -977682056
2020-04-17 22:47:07,003 INFO nodemanager.NodeStatusUpdaterImpl: Registered with ResourceManager as DESKTOP-SSVATPQ:57849 with total resource of <memory:8192, vCores:8>
```

To make sure that all services started successfully, we can run the following command:

jps

It should display the following services:

```
14560 DataNode
4960 ResourceManager
5936 NameNode
768 NodeManager
14636 Jps
```

Hadoop Web UI

There are three web user interfaces to be used:

- Name node web page: <http://localhost:9870/dfshealth.html>

Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'localhost:9820' (active)

Started:	Fri Apr 17 22:44:51 +0300 2020
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 18:56:00 +0300 2019 by rohitsharmaks from branch-3.2.1
Cluster ID:	CID-11d9a063-4c7b-4208-b192-2e4b6b736f
Block Pool ID:	BP-2032026115-192.168.1.105-1587150857190

Summary

Security is off.
 Safemode is off.
 1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
 Heap Memory used 63.28 MB of 191.5 MB Heap Memory. Max Heap Memory is 889 MB.

Data node web page: <http://localhost:9864/datanode.html>

Hadoop
Overview
Utilities

DataNode on [REDACTED]:9866

Cluster ID:	CID-11d9a063-4c7b-4208-b192-2e4b6b736f
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842


Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
localhost:9820	BP-2032026115-192.168.1.105-1587150857190	RUNNING	1s	12 minutes	0 B (64 MB)

Volume Information

Directory	Storage Type	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
-----------	--------------	---------------	---------------	-------------------	-----------------------------	--------

Yarn web page: <http://localhost:8088/cluster>



All Applications

Cluster

About
Nodes
Node Labels
Applications
NEW
NEW SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores
0	0	0	0	0	0 B	8 GB	0 B	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Repl...
1	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory 1024, vCores 1>	<memory 8192, vCores 4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB
No data available in table															

Showing 0 to 0 of 0 entries

CONCLUSIONS:

Student will able to install Hadoop framework by using above steps.

5. Lab Exercise

Exercise No 5: (2 Hours) – 1 Practical

1. Aim: - File management task in Apache Hadoop

THEORY:

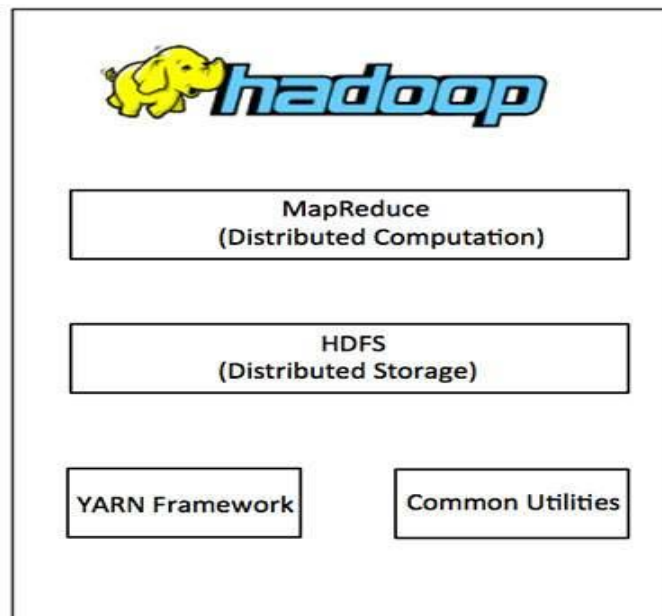
Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel with others. In short, Hadoop is used to develop applications that could perform complete statistical analysis on huge amounts of data.

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Hadoop Architecture

At its core, Hadoop has two major layers namely –

- Processing/Computation layer (MapReduce), and
- Storage layer (Hadoop Distributed File System).



v

MapReduce

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

Hadoop Distributed File System

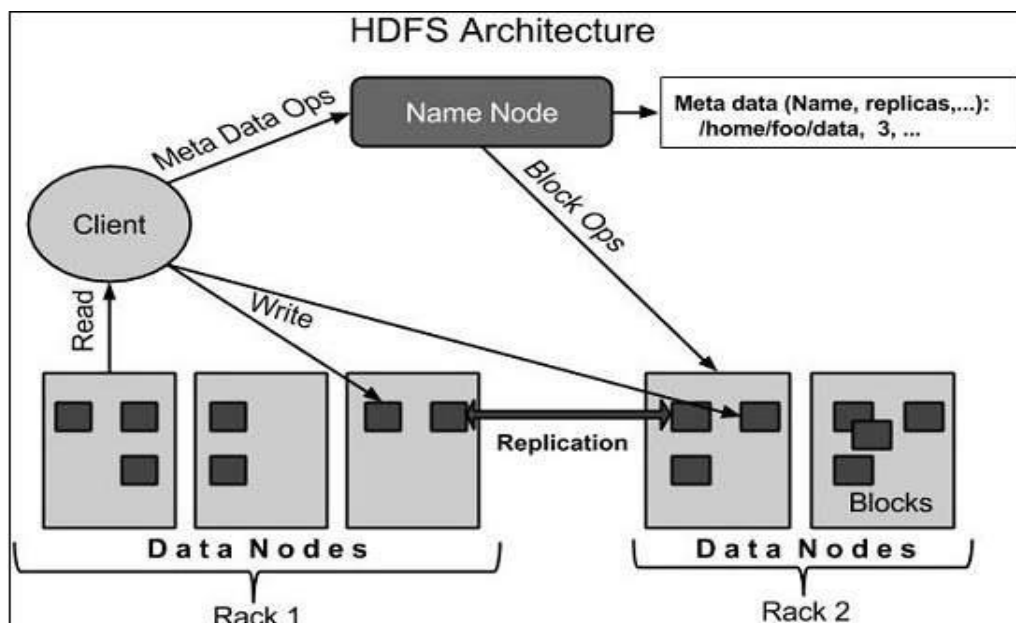
The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules –

- **Hadoop Common** – These are Java libraries and utilities required by other Hadoop modules.
- **Hadoop YARN** – This is a framework for job scheduling and cluster resource management.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



Starting HDFS

`.\start-dfs.cmd`

Listing Files in HDFS

After loading the information in the server, find the list of files in a directory, status of a file, using 'ls'. Given below is the syntax of ls that you can pass to a directory or a filename as an argument.

`$ $HADOOP_HOME/bin/hadoop fs -ls <args>`

Sr.No	Command & Description
1	-ls <path> Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry.
2	-lsr <path> Behaves like -ls, but recursively displays entries in all subdirectories of path.
3	-du <path> Shows disk usage, in bytes, for all the files which match path; filenames are reported with the full HDFS protocol prefix.
4	-dus <path> Like -du, but prints a summary of disk usage of all files/directories in the path.
5	-mv <src><dest> Moves the file or directory indicated by src to dest, within HDFS.
6	-cp <src> <dest> Copies the file or directory identified by src to dest, within HDFS.
7	-rm <path> Removes the file or empty directory identified by path.
8	-rmr <path> Removes the file or directory identified by path. Recursively deletes any child entries (i.e., files or subdirectories of path).
9	-put <localSrc> <dest> Copies the file or directory from the local file system identified by localSrc to dest

	within the DFS.
10	-copyFromLocal <localSrc> <dest> Identical to <code>-put</code>
11	-moveFromLocal <localSrc> <dest> Copies the file or directory from the local file system identified by localSrc to dest within HDFS, and then deletes the local copy on success.

CONCLUSIONS:

Students will be able to perform file handling commands on the Hadoop platform.

6. Lab Exercise

Exercise No 6: (2 Hours) – 1 Practical

1. Aim: - Map Reduce Paradigm

.THEORY:

MapReduce is a framework used to write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

What is MapReduce?

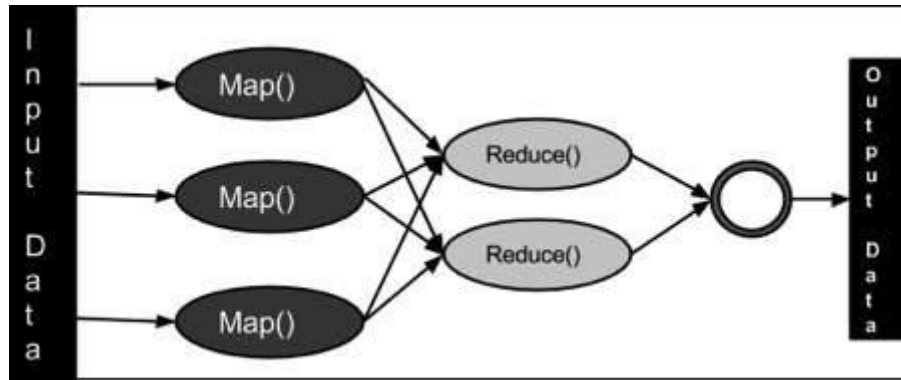
MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
 - **Map stage** – The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
 - **Reduce stage** – This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



CONCLUSIONS:

Students able to write simple programs using map-reduce paradigm.

7. Lab Exercise

Exercise No 7: (2 Hours) – 1 Practical

Aim: - Install, Deploy and configure Apache Spark. Develop application using Apache Spark.

THEORY:

Apache Spark is an open-source framework that processes large volumes of stream data from multiple sources. Spark is used in distributed computing with machine learning applications, data analytics, and graph-parallel processing.

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It also supports a rich set of higher-level tools including [Spark SQL](#) for SQL and structured data processing, [MLlib](#) for machine learning, [GraphX](#) for graph processing, and [Spark Streaming](#).

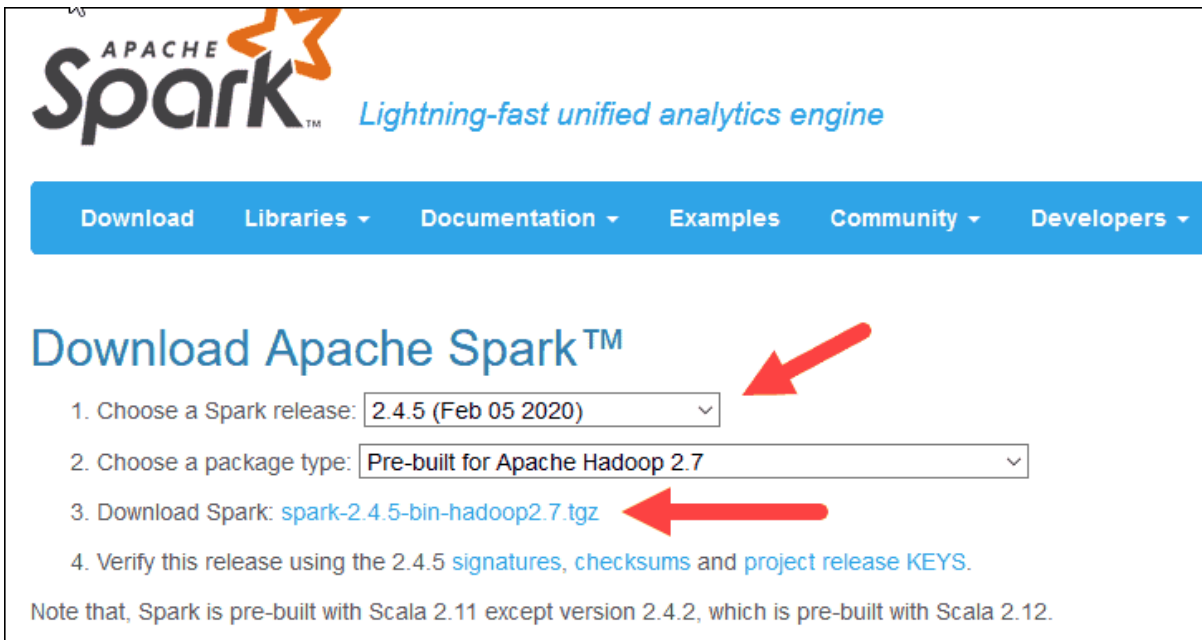
Installation of Apache Spark:

Prerequisites

- A system running Windows 10
- A user account with administrator privileges (required to install software, modify file permissions, and modify system PATH)
- Command Prompt
- A tool to extract .tar files
- Java 8
- Python 3.8

Download Apache Spark

1. Open a browser and navigate to <https://spark.apache.org/downloads.html>.
2. Under the *Download Apache Spark* heading, there are two drop-down menus. Use the current non-preview version.
 - In our case, in *Choose a Spark release* drop-down menu select **2.4.5 (Feb 05 2020)**.
 - In the second drop-down *Choose a package type*, leave the selection **Pre-built for Apache Hadoop 2.7**.
3. Click the *spark-2.4.5-bin-hadoop2.7.tgz* link.
4. A page with a list of mirrors loads where you can see different servers to download from. Pick any from the list and save the file to your Downloads folder.



APACHE Spark™ Lightning-fast unified analytics engine

Download Libraries ▾ Documentation ▾ Examples Community ▾ Developers ▾

Download Apache Spark™

1. Choose a Spark release: 2.4.5 (Feb 05 2020) ▾
2. Choose a package type: Pre-built for Apache Hadoop 2.7 ▾
3. Download Spark: [spark-2.4.5-bin-hadoop2.7.tgz](#)
4. Verify this release using the 2.4.5 [signatures](#), [checksums](#) and [project release KEYS](#).

Note that, Spark is pre-built with Scala 2.11 except version 2.4.2, which is pre-built with Scala 2.12.

Install Apache Spark

Installing Apache Spark involves **extracting the downloaded file** to the desired location.

1. Create a new folder named *Spark* in the root of your C: drive.
2. In Explorer, locate the Spark file you downloaded.
3. Right-click the file and extract it to *C:\Spark* using the tool you have on your system
4. Now, your *C:\Spark* folder has a new folder *spark-2.4.5-bin-hadoop2.7* with the necessary files inside.

Add winutils.exe File

Download the **winutils.exe** file for the underlying Hadoop version for the Spark installation you downloaded.

1. Navigate to this URL <https://github.com/cdarlint/winutils> and inside the **bin** folder, locate **winutils.exe**, and click it.
2. Find the **Download** button on the right side to download the file.
3. Now, create new folders **Hadoop** and **bin** on C: using Windows Explorer or the Command Prompt.

8. Lab Exercise

Exercise No 4: (2 Hours) – 1 Practical

Aim: - Application development using Apache PySpark.

THEORY:

Apache Spark is a new and open-source framework used in the big data industry for real-time processing and batch processing. It supports different languages, like Python, Scala, Java, and R.

Apache Spark is initially written in a Java Virtual Machine (JVM) language called Scala, whereas Pyspark is like a Python API which contains a library called Py4J. This allows dynamic interaction with JVM objects.

Pyspark = Python + Apache Spark

Installation of PySpark is similar to that of Apache Spark as explain in 7Th experiment.

Starting of PySpark

Open Command Prompt and type the following command

Pyspark

Following message is displayed and pyspark is ready to execute task.

```
Type "help", "copyright", "credits" or "license" for more information.
20/06/17 17:36:48 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

  ____      _
 / ___|  _ \| | | |
 \___ \| | | | |_| |
  ___) | |_| | __| |
 |____|_||_|_|_|_|_|

version 2.4.6

Using Python version 3.7.4 (default, Aug 9 2019 18:34:13)
SparkSession available as 'spark'.
>>> _
```

Who uses PySpark?

PySpark is very well used in Data Science and Machine Learning community as there are many widely used data science libraries written in Python including NumPy, TensorFlow also used due to its efficient processing of large datasets. PySpark has been used by many organizations like Walmart, Trivago, Sanofi, Runtastic, and many more.

PySpark Modules & Packages

- PySpark RDD ([pyspark.RDD](#))
- PySpark DataFrame and SQL ([pyspark.sql](#))
- PySpark Streaming ([pyspark.streaming](#))
- PySpark MLib ([pyspark.ml](#), [pyspark.mllib](#))
- PySpark GraphFrames ([GraphFrames](#))
- PySpark Resource ([pyspark.resource](#)) It's new in PySpark 3.0
-

PySpark RDD – Resilient Distributed Dataset

PySpark RDD (Resilient Distributed Dataset) is a fundamental data structure of PySpark that is fault-tolerant, immutable distributed collections of objects, which means once you create an RDD you cannot change it. Each dataset in RDD is divided into logical partitions, which can be computed on different nodes of the cluster.

RDD Creation

In order to create an RDD, first, you need to create a SparkSession which is an entry point to the PySpark application. SparkSession can be created using a builder() or newSession() methods of the SparkSession.

```
spark = SparkSession.builder()  
    .master("local[1]")  
    .appName("SparkByExamples.com")  
    .getOrCreate()
```

using parallelize()

SparkContext has several functions to use with RDDs. For example, it's *parallelize()* method is used to create an RDD from a list.

```
#Create RDD from parallelize
```

```
dataList = [("Java", 20000), ("Python", 100000), ("Scala", 3000)]
```

```
rdd=spark.sparkContext.parallelize(dataList)
```

Using textFile()

RDD can also be created from a text file using ***textFile()*** function of the SparkContext.

//Create RDD from external Data source

```
rdd2 = spark.sparkContext.textFile("/path/textFile.txt")
```

RDD Operations

On PySpark RDD, following operations are perform.

RDD transformations – Transformations are lazy operations. When you run a transformation(for example update), instead of updating a current RDD, these operations return another RDD.

RDD actions – operations that trigger computation and return RDD values to the driver.

RDD Transformations

Transformations on Spark RDD returns another RDD and transformations are lazy meaning they don't execute until you call an action on RDD. Some transformations on RDD's are flatMap(), map(), reduceByKey(), filter(), sortByKey() and return new RDD instead of updating the current.

RDD Actions

RDD Action operation returns the values from an RDD to a driver node. In other words, any RDD function that returns non RDD[T] is considered as an action.

Some actions on RDD's are count(), collect(), first(), max(), reduce() and more.

CONCLUSIONS:

Student will able to install Pyspark, create RDD using different techniques and perform relevant operations on it.

9. Lab Exercise

Exercise No 9: (2 Hours) – 1 Practical

Aim: - Install and Run HBase and then use HBase DDL and DML commands

THEORY:

What is HBase?

Apache HBase is an open-source, NoSQL, distributed big data store. It enables random, strictly consistent, real-time access to petabytes of data. HBase is very effective for handling large, sparse datasets.

HBase integrates seamlessly with Apache Hadoop and the Hadoop ecosystem and runs on top of the Hadoop Distributed File System (HDFS) or Amazon S3 using Amazon Elastic MapReduce (EMR) file system, or EMRFS. HBase serves as a direct input and output to the Apache MapReduce framework for Hadoop, and works with Apache Phoenix to enable SQL-like queries over HBase tables.



How does HBase work?

HBase is a column-oriented, non-relational database. This means that data is stored in individual columns, and indexed by a unique row key. This architecture allows for rapid retrieval of individual rows and columns and efficient scans over individual columns within a table. Both data and requests are distributed across all servers in an HBase cluster, allowing you to query results on petabytes of data within milliseconds. HBase is most effectively used to store non-relational data, accessed via the HBase API. Apache Phoenix is commonly used as a SQL layer on top of HBase allowing you to use familiar SQL syntax to insert, delete, and query data stored in HBase.

Benefits of HBase

SCALABLE

HBase is designed to handle scaling across thousands of servers and managing access to petabytes of data. With the elasticity of Amazon EC2, and the scalability of Amazon S3, HBase is able to handle online access to massive data sets.

FAST

HBase provides low latency random read and write access to petabytes of data by distributing requests from applications across a cluster of hosts. Each host has access to data in HDFS and S3, and serves read and write requests in milliseconds.

FAULT-TOLERANT

HBase splits data stored in tables across multiple hosts in the cluster and is built to withstand individual host failures. Because data is stored on HDFS or S3, healthy hosts will automatically be chosen to host the data once served by the failed host, and data is brought online automatically.

Apache HBase Installation

The objective of this tutorial is to describe step by step process to install Apache HBase (Version 2.3.0) using Apache Hadoop 3.1.2 on Ubuntu 18.04.4 LTS (Bionic Beaver). Once the installation is completed then you can work with Apache HBase.

Platform

- **Operating System (OS).** We have used Ubuntu 18.04.4 LTS version, You can use other flavors of Linux systems such as Redhat, CentOS, etc.
- **Hadoop.** We have already installed Hadoop 3.1.2 version on which we will run Apache HBase (Please refer to the "Hadoop Installation on Single Node" tutorial and install Hadoop first before proceeding for Apache HBase installation.)
- **HBase.** We have used the Apache HBase-2.3.0 version for installation.

Download Software

- **HBase.**

<https://downloads.apache.org/hbase/2.3.0/hbase-2.3.0-bin.tar.gz>

Steps to Install Apache HBase version(2.3.0) on Ubuntu 18.04.4 LTS

Please follow the below steps to install HBase.

Step 1. Please verify if Hadoop is installed.

Step 2. Please verify if Java is installed.

Step 3. Please download HBase 2.3.0 from the below link.

On Linux: `$wget https://downloads.apache.org/hbase/2.3.0/hbase-2.3.0-bin.tar.gz`

On Windows: <https://downloads.apache.org/hbase/2.3.0/hbase-2.3.0-bin.tar.gz>

```
cloudduggu@ubuntu: ~  
cloudduggu@ubuntu:~$ wget https://downloads.apache.org/hbase/2.3.0/hbase-2.3.0-bin.tar.gz  
--2020-08-05 06:43:38-- https://downloads.apache.org/hbase/2.3.0/hbase-2.3.0-bin.tar.gz  
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 2a01:4f8:10a:201a::2  
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 271784798 (259M) [application/x-gzip]  
Saving to: 'hbase-2.3.0-bin.tar.gz'  
  
hbase-2.3.0-bin.tar.gz      100%[=====>] 259.19M  68.3KB/s   in 65m 35s  
  
2020-08-05 07:49:14 (67.5 KB/s) - 'hbase-2.3.0-bin.tar.gz' saved [271784798/271784798]  
cloudduggu@ubuntu:~$
```

Step 4. Let us extract the tar file using the below command and rename the folder to HBase to make it meaningful.

\$tar -xzf hbase-2.3.0-bin.tar.gz

\$mv hbase-2.3.0 hbase

```
cloudduggu@ubuntu: ~  
cloudduggu@ubuntu:~$ tar -xzf hbase-2.3.0-bin.tar.gz  
cloudduggu@ubuntu:~$ mv hbase-2.3.0 hbase  
cloudduggu@ubuntu:~$
```

Step 5. Now edit (hbase-env.sh) configuration file which is present under(/home/cloudduggu/hbase/conf) and add JAVA path as mentioned below.

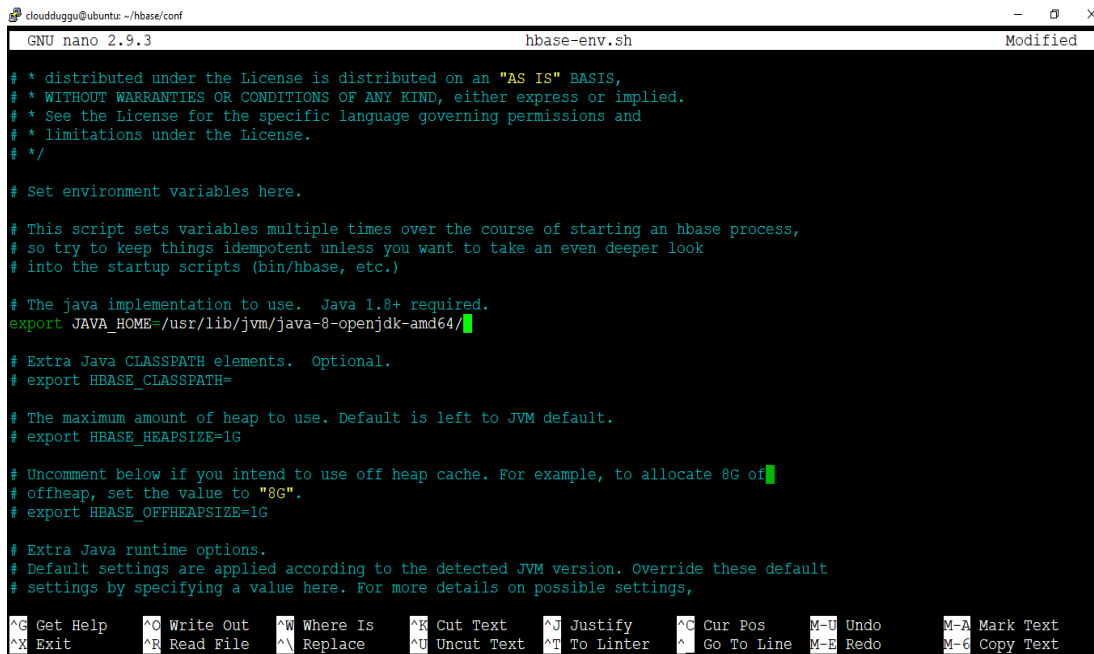
cloudduggu@ubuntu:~ /hbase/conf\$ nano hbase-env.sh

Now put JAVA path.

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

cloudduggu@ubuntu: ~/hbase/conf

```
cloudduggu@ubuntu:~/hbase/conf$ nano hbase-env.sh
```



```
GNU nano 2.9.3 hbase-env.sh Modified

# * distributed under the License is distributed on an "AS IS" BASIS,
# * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# * See the License for the specific language governing permissions and
# * limitations under the License.
# */

# Set environment variables here.

# This script sets variables multiple times over the course of starting an hbase process,
# so try to keep things idempotent unless you want to take an even deeper look
# into the startup scripts (bin/hbase, etc.)

# The java implementation to use.  Java 1.8+ required.
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

# Extra Java CLASSPATH elements.  Optional.
# export HBASE_CLASSPATH=

# The maximum amount of heap to use. Default is left to JVM default.
# export HBASE_HEAPSIZE=1G

# Uncomment below if you intend to use off heap cache. For example, to allocate 8G of
# offheap, set the value to "8G".
# export HBASE_OFFHEAPSIZE=1G

# Extra Java runtime options.
# Default settings are applied according to the detected JVM version. Override these default
# settings by specifying a value here. For more details on possible settings,
```

Save the changes by pressing CTRL + O and exit from the nano editor by pressing CTRL + X.

Step 6. After this edit (.bashrc) file to update the environment variable of Apache HBase so that it can be accessed from any directory.

cloudduggu@ubuntu:~\$ nano .bashrc

Add below path.

export HBASE_HOME= /home/cloudduggu/hbase

export PATH=\$PATH:\$HBASE_HOME/bin

```
cloudduggu@ubuntu: ~  
GNU nano 2.9.3 .bashrc Modified  
  
export SPARK_HOME=/home/cloudduggu/spark  
export HIVE_HOME=/home/cloudduggu/hive  
export PATH=$PATH:$HIVE_HOME/bin  
  
export PIG_HOME=/home/cloudduggu/pig  
export PATH=$PATH:/home/cloudduggu/pig/bin  
export PIG_CLASSPATH=$HADOOP_HOME/etc/Hadoop  
  
export SQOOP_HOME=/home/cloudduggu/sqoop  
export PATH=$PATH:$SQOOP_HOME/bin  
  
export FLUME_HOME=/home/cloudduggu/flume  
export PATH=$PATH:$FLUME_HOME/bin  
  
export HBASE_HOME=/home/cloudduggu/hbase  
export PATH=$PATH:$HBASE_HOME/bin  
  
^G Get Help      ^O Write Out     ^M Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos       M-U Undo        M-A Mark Text  
^X Exit          ^R Read File     ^\ Replace       ^U Uncut Text    ^T To Spell      _ Go To Line    M-E Redo        M-D Copy Text
```

Step 7. Now start Apache HBase and verify it using the below commands.

cloudduggu@ubuntu:~/hbase\$ bin/start-hbase.sh

cloudduggu@ubuntu:~/hbase\$ jps

```
cloudduggu@ubuntu: ~/hbase  
cloudduggu@ubuntu:~/hbase$ bin/start-hbase.sh  
/home/cloudduggu/hadoop/libexec/hadoop-functions.sh: line 2358: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_USER:  
bad substitution  
/home/cloudduggu/hadoop/libexec/hadoop-functions.sh: line 2453: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_OPTS:  
bad substitution  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/home/cloudduggu/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/  
/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/home/cloudduggu/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/  
impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
running master, logging to /home/cloudduggu/hbase/bin/../logs/hbase-cloudduggu-master-ubuntu.out  
/home/cloudduggu/hadoop/libexec/hadoop-functions.sh: line 2358: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_USER:  
bad substitution  
/home/cloudduggu/hadoop/libexec/hadoop-functions.sh: line 2453: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_OPTS:  
bad substitution  
cloudduggu@ubuntu:~/hbase$ jps  
10384 NodeManager  
10935 HMaster  
9928 SecondaryNameNode  
9546 NameNode  
9707 DataNode  
11166 Jps  
10223 ResourceManager
```

Step 8. After this we can see Hbase services are running from the JPS command, now let us start the HBase shell using the below command.

cloudduggu@ubuntu:~/hbase\$ bin/hbase shell

```
clouduggu@ubuntu: ~/hbase
clouduggu@ubuntu:~/hbase$ bin/hbase shell
/home/clouduggu/hadoop/libexec/hadoop-functions.sh: line 2358: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_USER:
bad substitution
/home/clouduggu/hadoop/libexec/hadoop-functions.sh: line 2453: HADOOP_ORG.APACHE.HADOOP.HBASE.UTIL.GETJAVAPROPERTY_OPTS:
bad substitution
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/clouduggu/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl
/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/clouduggu/hbase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.30.jar!/org/slf4j/
impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell
Version 2.3.0, re0e1382705c59d3fb3ad8f5bfff720a9dc7120fb0, Mon Jul 6 22:27:43 UTC 2020
Took 0.0067 seconds
hbase(main):001:0> whoami
clouduggu (auth:SIMPLE)
groups: clouduggu, adm, cdrom, sudo, dip, plugdev, lpadmin, sambashare
Took 0.1609 seconds
hbase(main):002:0> █
```

DDL commands are table management commands which are used to create a table, list table, describe the table, and so on.

Apache HBase supports the below list of DDL commands.

1. [Create Table](#)
2. [List](#)
3. [Describe](#)
4. [Disable](#)
5. [Disable all](#)
6. [Show filters](#)
7. [Drop Table](#)
8. [Drop all](#)
9. [Is enabled](#)
10. [Alter](#)

DML commands are used to perform data manipulation operations such as putting data into a table, retrieving data from a table and deleting schema, and so on.

Apache HBase supports the below list of DML commands.

1. [Put](#)
2. [Count](#)
3. [Get](#)
4. [Scan](#)
5. [Delete](#)
6. [Delete all](#)
7. [Truncate](#)

CONCLUSIONS:

Student will able to install HBase, perform DML and DLL commands on it.

10. Lab Exercise

Exercise No 10: (2 Hours) – 1 Practical

Aim: - Install and configure Apache Kafka. Integrate of Apache kafka with Apache Spark

THEORY:

What is Apache Kafka?

Apache Kafka is an open-source platform that ingests and processes streaming data in real-time. Streaming data is generated every second simultaneously by tens of thousands of data sources. This constant stream of records can be processed by a streaming platform like Kafka both sequentially and incrementally. Apache Kafka uses a Publish and Subscribe model to read and write streams of records. Kafka is also described as a distributed messaging software and as with other message queues, Kafka reads and writes messages asynchronously. However, unlike other messaging systems, Kafka has built-in partitioning, replication, higher throughput and is more fault-tolerant which make it ideal for high volume message processing.

What is Kafka used for?

Kafka Use Cases are many and found across industries like Financial Services, Manufacturing, Retailing, Gaming, Transportation & Logistics, Telecom, Pharma, Life Sciences, Healthcare, Automotive, Insurance and more. In fact, Kafka is used wherever there is high-volume streaming data to be ingested and used for insights. Kafka use cases include data integration, event streaming, data processing, developing business applications and microservices. Kafka can be used in Cloud, Multi-cloud, Hybrid deployments.

Kafka Use Case 1: Tracking Website Activity

Kafka can be used to track user activity by building user activity tracking pipelines with real-time publish-subscribe feeds. Pages viewed, user searches, user registrations, subscribes, comments, transactions etc. can be written to Topics and made available for processing in real-time for data warehouse analytics on platforms like [Snowflake](#), [Amazon Redshift](#), [Amazon S3](#), [ADLS Gen2](#), [Azure Synapse](#), [SQL Server](#) etc.

Kafka Use Case 2: Operational Metrics

Kafka can be used to report on operational metrics by usage in operational data pipelines. It can collect data from distributed applications to enable alerting and reporting for operational metrics, by creating centralized data feeds of data from operations. [CDC to Kafka explained](#)

Kafka Use Case 3: Aggregating Logs

Kafka collects logs from multiple services and makes them available in standard format to multiple consumers. Kafka supports low-latency processing and many data sources, which is great for distributed data consumption.

Kafka Use Case 4: Stream Processing

Unlike data warehousing systems where data is stored and then processed, you can now process data as it arrives -this is real-time data processing. The real-time processing of

streams of data when done continuously, simultaneously and in a sequence of records, is Kafka stream processing. Real-time stream processing means Kafka reads data from a Topic (Source), carries out some transformation or analysis and writes the outcome to another Topic (Sink) for applications and users to consume. The high durability of Kafka is advantageous for real-time stream processing.

How does Kafka Work?

Kafka is known as an event streaming platform since you can:

- Publish i.e. write streams of events and Subscribe i.e. read streams of events, involving a constant data import and export from other applications.
- Store event streams reliably for as long as you like.
- Process event streams as they happen or later.

Kafka is highly elastic, extremely scalable, secure and has a data distributed publish-subscribe messaging system. This distributed system has servers and clients that work through a TCP Network Protocol. This TCP (Transmission Control Protocol) helps in transmitting packets of data from source to destination, between processes, applications, and servers. The protocol establishes connection before the communication happens between the two computing systems in the network. Kafka can be deployed on virtual machines, bare-metal hardware, on-premise containers and in cloud environments

Kafka Architecture -the 1000 ft view

- The Brokers (Servers or Nodes) handle client requests for production, consumption and metadata and enable replication of data in the cluster. The number of Brokers can be more than one in a cluster.
- Zookeeper maintains the state of a cluster, configuration of topics, electing a leader, ACL lists, broker lists etc.
- Producer is an application that produces and delivers records to a broker.
- Consumer is the system that consumes the records from the broker.

Kafka Producers and Consumers

Kafka Producers are basically client applications that publish or write events to Kafka while Kafka Consumers are those systems that subscribe to or read and process those events. Kafka Producers and Kafka Consumers are completely decoupled and there are no dependencies between them. This is one of the reasons Apache Kafka is so highly scalable. The capability to process events exactly-once is one of the guarantees that comes with Kafka.

Kafka Topics

A Kafka Topic is essentially a category to help in organizing and managing messages. A Kafka Topic will have a unique name within the Kafka cluster. Messages are sent to and read from individual Topics. The Kafka Producers writes data to a Topic and the Kafka Consumer reads the data from a Topic. A Kafka Topic can have zero to multiple subscribers. Every Kafka Topic is partitioned and replicated across different Kafka Brokers (Kafka servers or nodes in a Kafka Cluster). This distributed placement enables client applications to read and write data to and from multiple brokers simultaneously. When a new event is written to a Kafka Topic, it is attached to a Topic partition. Events that involve the same event key such as a specific customer id or payment id will be written to the same partition always, and the Consumer of that topic -partition will always get to read the events in the same sequence as

they were written. Partitions are integral to Kafka's functioning since they enable topic parallelization and high message throughput.

Kafka Records

Records are essentially information about events that are stored on the Topic as a record. Applications can connect and transfer a record into the Topic. Data is durable and can be stored in a topic for a long time till the specified retention period is over. Records can consist of various types of information – it could be information about an event on a website like a purchase transaction, feedback on social media or some readings from a sensor-driven device. It could be an event that flags off another event. These records on a topic can be processed and reprocessed by applications that are connected to the Kafka system. Records can essentially be described as byte arrays that are capable of storing objects in any format. An individual record will have 2 mandatory attributes -the Key and the Value and 2 optional attributes -the Timestamp and the Header.

Apache Zookeeper and Kafka

Apache Zookeeper is a software that monitors and maintains order within a Kafka system and behaves as a centralized, distributed coordination service for the Kafka Cluster. It maintains data about configuration and naming and is responsible for synchronization within all the distributed systems. Apache Zookeeper tracks Kafka cluster node statuses, Kafka messages, partitions, and topics among other things. Apache Zookeeper enables multiple clients to do reads and writes concurrently, to issue updates and act as a shared registry within the system. Apache ZooKeeper is integral to the development of distributed applications. It is used by Apache Hadoop, HBase and other platforms for functions like co-ordination of nodes, configuration management, leader election etc.

Kafka Zookeeper Synergy and Apache Zookeeper Use Cases

Apache Zookeeper co-ordinates the Kafka Cluster. Kafka needs Zookeeper to be installed before it can be used in production. This is essential even if the system consists of a single broker, topic, and partition. Basically, Zookeeper has five use cases – electing a controller, cluster membership, configuration of topics, access control lists (ACLs) and monitoring quotas. These are some Apache Zookeeper Use Cases:

Apache Zookeeper elects a Kafka Controller

The Kafka Controller is the Broker or Server that maintains the leader/follower relationship among the partitions. Every Kafka cluster has only one Controller. In the event of a node shutting down, it is the Controller's job to ensure other replicas assume responsibility as partition leaders to replace the partition leaders within the node that is shutting down.

Zookeeper Software keeps a list of Kafka Cluster Members

Apache Zookeeper functions as a registry and keeps a note of all the active brokers on a Kafka cluster.

Apache Zookeeper maintains Configuration of Topics

The Zookeeper software keeps a record of all topic configurations including a topics list, number of topic partitions for individual topics, overrides for topic configurations, preferred leader nodes and replica locations among other things.

Zookeeper Software maintains Access Control Lists or ACLs

The ACLs or Access Control Lists for all topics are also kept by the Zookeeper software. Details like read/write permissions for every topic, list of Consumer groups, group members and the latest offset individual consumer groups have got from the partition are all available.

Apache Zookeeper keeps tabs on Client Quotas

Apache Zookeeper can access the amount of data each client has permission to read or write.

Installing Kafka – some steps

Installation of Apache Kafka on a Windows OS

Prerequisites: Need to have Java installed before installing Kafka.

JRE Server Download: <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

Kafka Installation – files required

The following files will need to be downloaded for Kafka Installation:

- ZooKeeper Download: <http://zookeeper.apache.org/releases.html>
- Kafka Download: <http://kafka.apache.org/downloads.html>

Install Apache ZooKeeper

- a. Download and extract Apache ZooKeeper from the link above.
- b. Go to the ZooKeeper config directory, Change the dataDir path from “dataDir=tmp/zookeeper” to “:\zookeeper-3.6.3\data” in the zoo_sample.cfg file. Please note the Zookeeper folder name might differ as per the downloaded version.
- c. Set up System Environment Variables, add new “ZOOKEEPER_HOME = C:\zookeeper-3.6.3”.
- d. Edit the System Variable named Path and add ;%ZOOKEEPER_HOME%\bin;.
- e. From cmd run – “zkserver”. Now ZooKeeper is up and running on port default port 2181, which can be changed in the zoo_sample.cfg file.

Install Kafka

- a. Download and extract Apache Kafka from the link above.
- b. In the Kafka config directory. Change log.dir path from “log.dirs=tmp/kafka-logs” to “C:\kafka-3.0.0\kafka-logs” in the server.properties. Please note the Kafka folder name might differ as per the downloaded version.
- c. In case ZooKeeper is running on a different machine, then edit these properties of server.properties
Here we need to define the private IP of server
listeners = PLAINTEXT://172.31.33.3:9092## Here we need to define the Public IP of server}
advertised.listeners=PLAINTEXT://35.164.149.91:9092
- d. Add below properties in server.properties
advertised.host.name=172.31.33.3
advertised.port=9092
- e. Kafka runs on default port 9092 and connects to ZooKeeper’s default port, 2181.

Running a Kafka Server

Note: make sure Apache ZooKeeper is up and running before starting a Kafka server. Otherwise, open cmd in C:\zookeeper-3.6.3\bin\windows and run Zookeeper server start batch > zookeeper-server-start.bat ../../config/zookeeper.properties

- a. From the Kafka installation directory C:\kafka-3.0.0\bin\windows, open cmd and run the command below.
>kafka-server-start.bat ../../config/server.properties.
- b. Now Kafka Server is up and running, it's time to create new Kafka Topics to store messages.

Create Kafka Topics

- a. Create a new Topic like my_new_topic.
- b. From C:\kafka-3.0.0\bin\windows open cmd and run below command:
>kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic my_new_topic

Useful Commands for Kafka Installation

- a. Open a new command prompt in the location to view a list of Topics.C:\kafka-3.0.0\bin\windows>kafka-topics.bat --list --zookeeper localhost:2181
- b. Producer and Consumer commands for testing messages:>kafka-console-producer.bat --broker-list localhost:9092 --topic my_new_topic>kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic my_new_topic (above Kafka version >2.0)

CONCLUSIONS:

Student will able to install Apache Kafka on windows.