

Version History.....	2
Version	2
Reference Documents.....	2
Comments	2
Installation	3
How to access Hadoop.....	6
Admin Password Reset.....	6
Explore Ambari.....	9
Default Users in Hadoop Sandbox	10
Getting Started with HDP Sandbox.....	10
Concepts	10
Data Management	11
Data Access	11
Data Governance and Integration	12
Security	12
Operations	12
HDFS.....	13
Apache MapReduce	14
Apache Hive	16
Apache ORC.....	16
Apache Spark	17
Apache Zeppelin.....	18
Loading Sensor Data into HDFS.....	18
Hive - Data ETL	21
Data Analytics Studio	21
Create and Load Table	22
Spark - Risk Factor.....	26

Version History

Version

SI No	Date	Author	Comments
1	15-Nov-2020	Ansu John	Initial version on installation

Reference Documents

Document	Version
https://www.ibm.com/downloads/cas/KJLWJG2L	Jan-2017
https://www.cloudera.com/tutorials/learning-the-ropes-of-the-hdp-sandbox.html	14-Nov-2020
https://www.cloudera.com/tutorials/getting-started-with-hdp-sandbox/1.html	15-Nov-2020

Comments

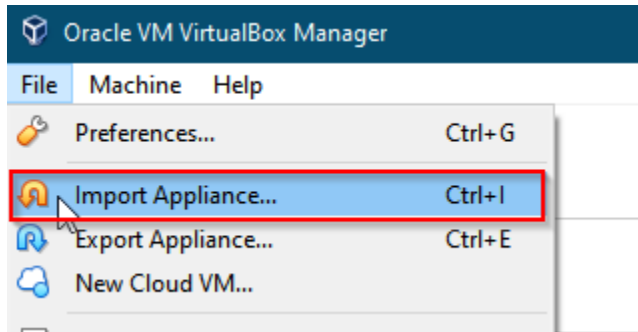
Please contact me @ [LinkedIn](#) for any suggestions.

In case you are interested in data science projects exploring various analytics tools and techniques, please refer to my [GitHub repositories](#).

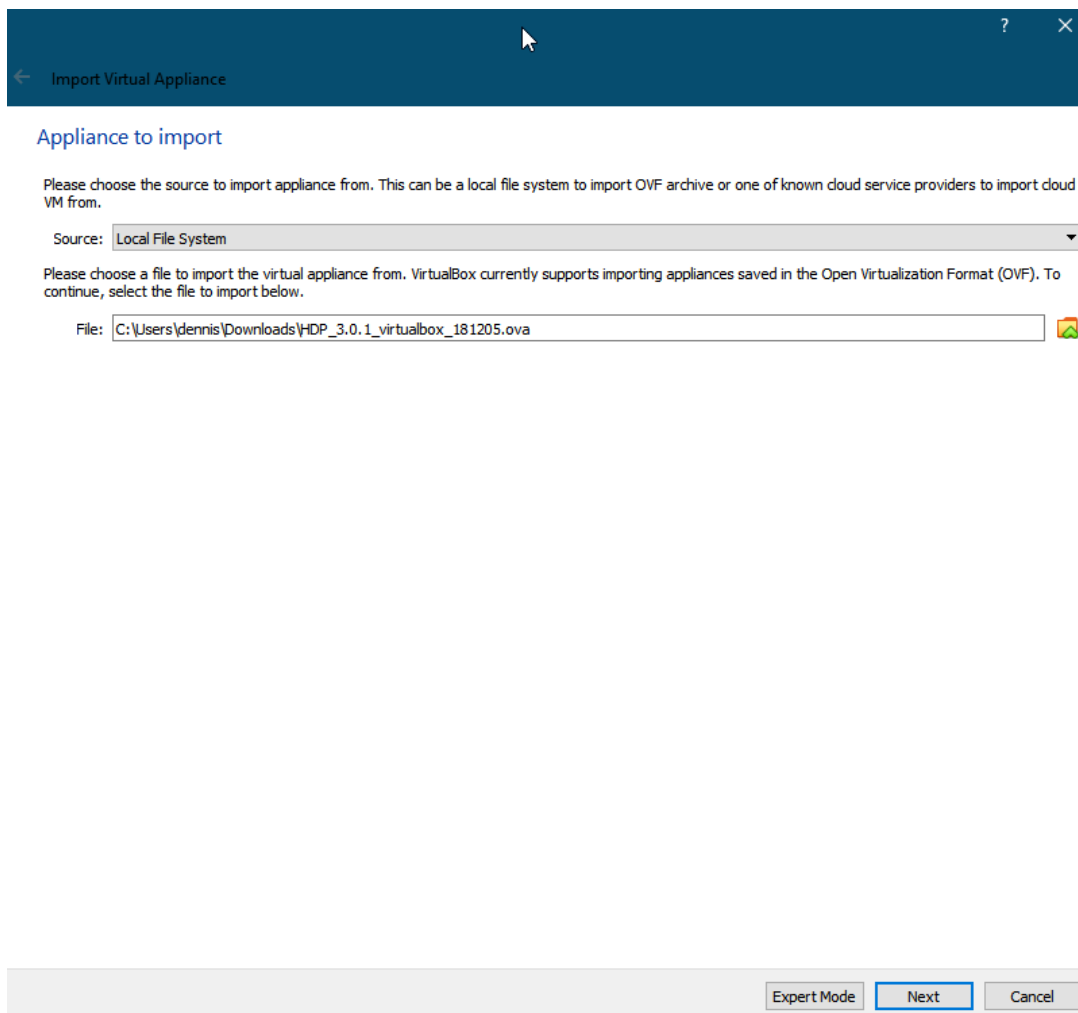
Installation

Prerequisite: Installed Oracle Virtual box and downloaded copy of Hortonworks Sandbox for Hadoop in your computer.

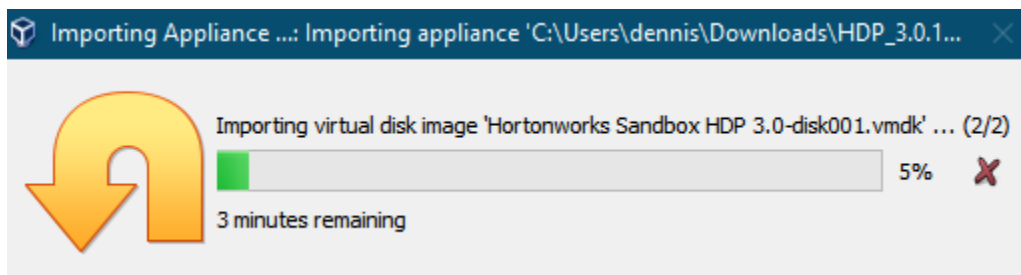
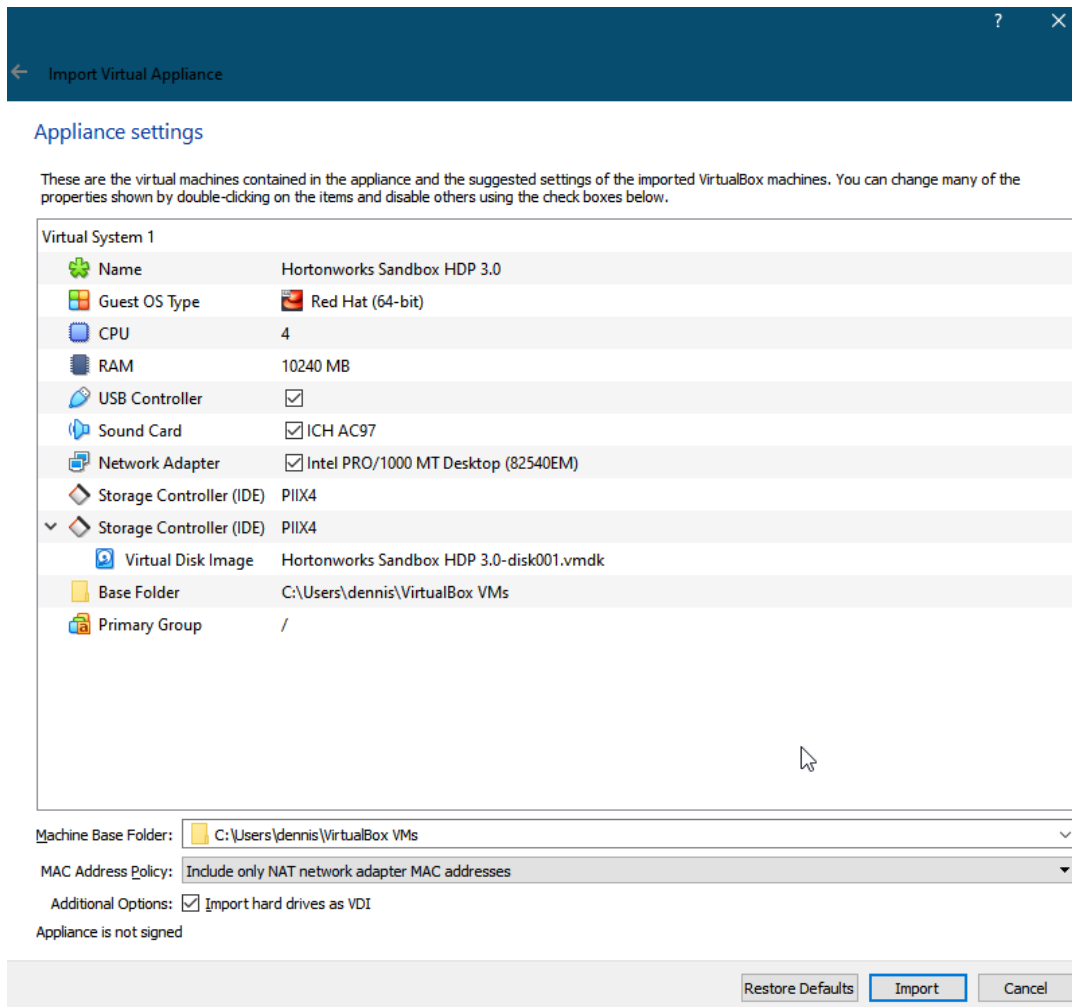
Step 1: Open Oracle Virtual box. Navigate to “File” > “Import Appliance...”



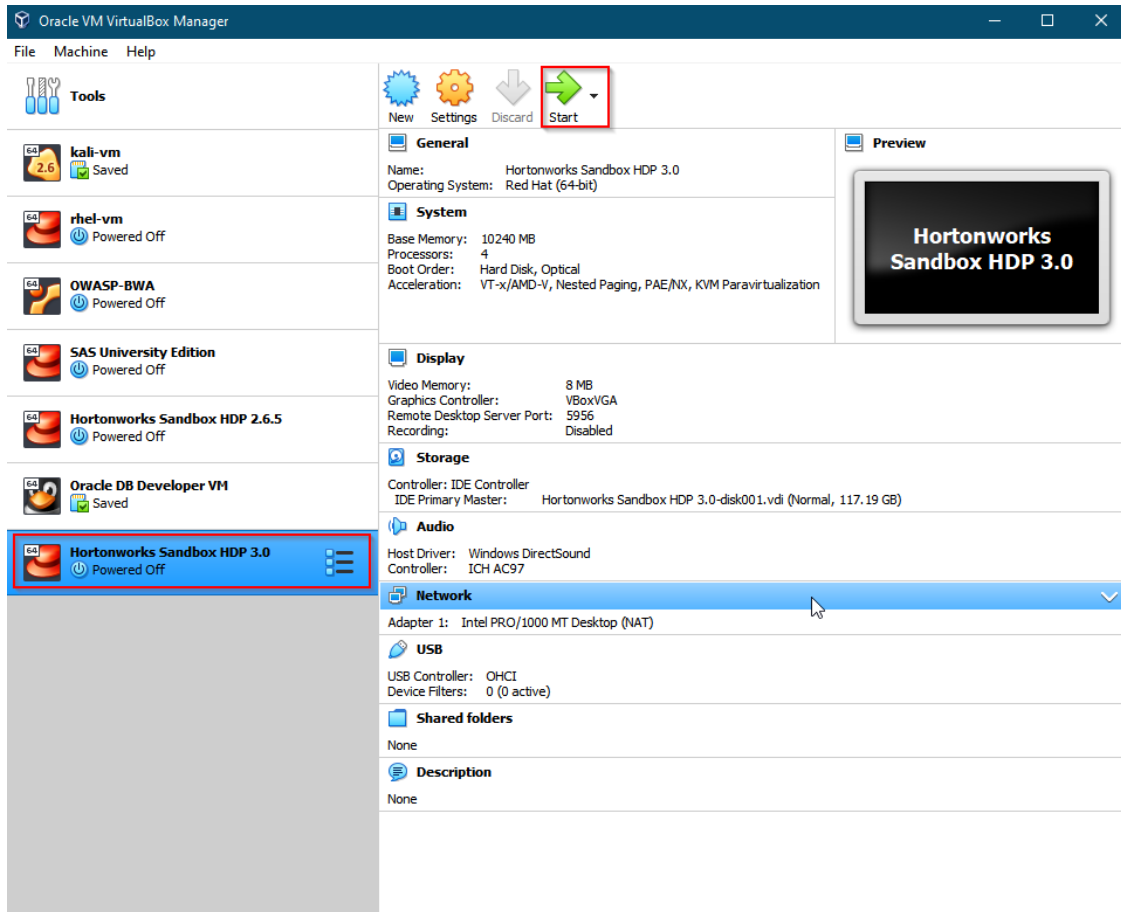
Step 2: Select the Hortonworks Sandbox ova file using the file browser icon from “Import Virtual Appliance” screen.



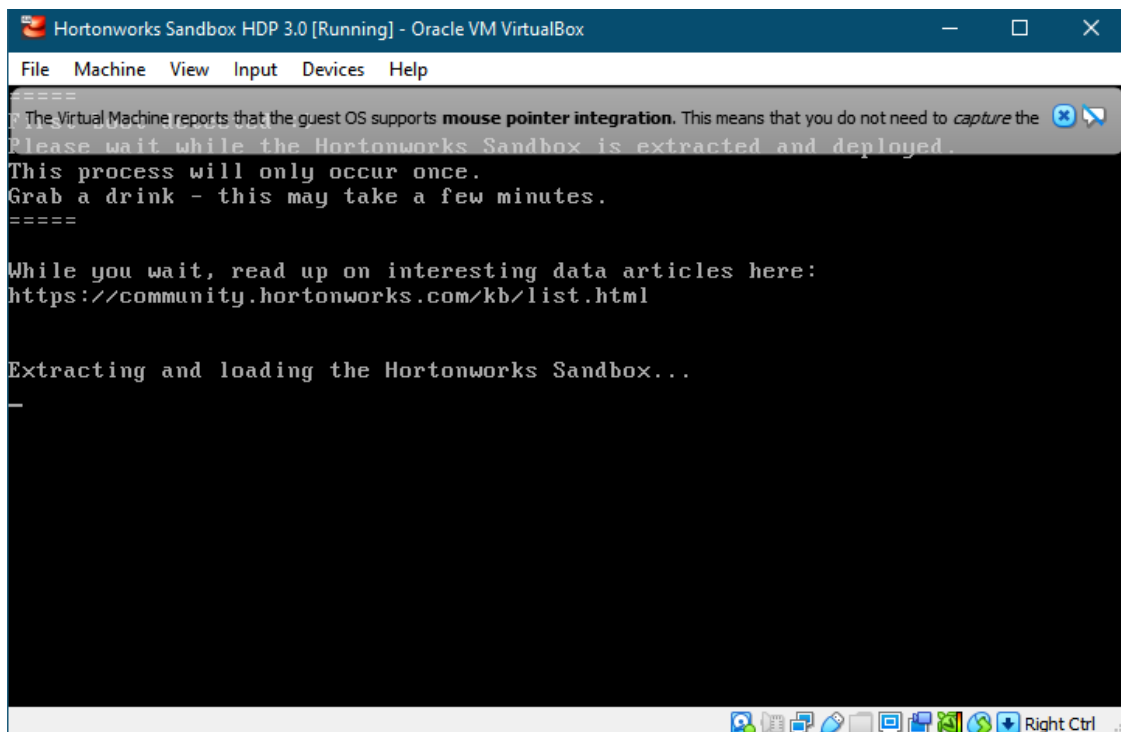
Step 3: Verify the appliance settings and click “Import”.



Step 4: Once the import completes, you will be able to view the “Hortonworks Sandbox” in your Oracle virtual box.

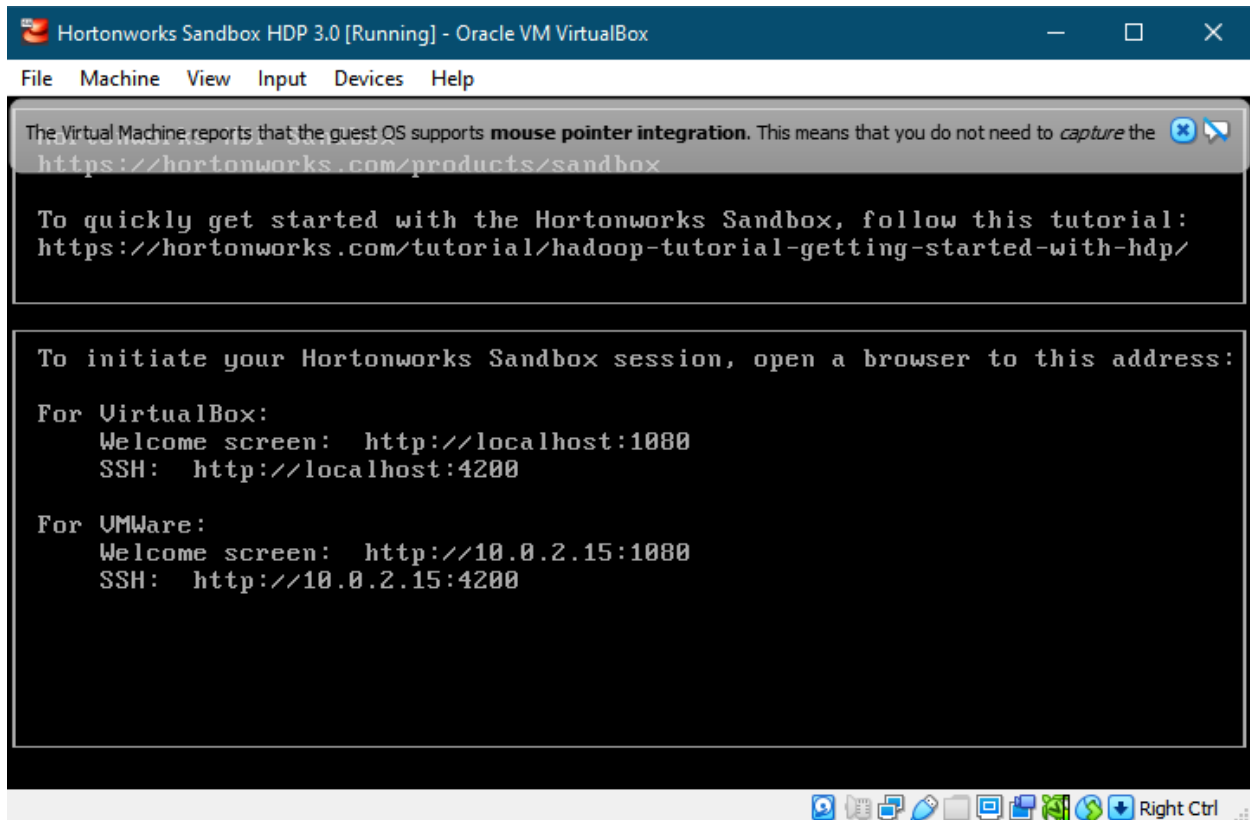


Step 5: Click “Start” to start Hadoop. The following screen will appear.



How to access Hadoop

Once Hortonworks Sandbox starts, it will display the URL to access Hadoop in the screen as shown below.



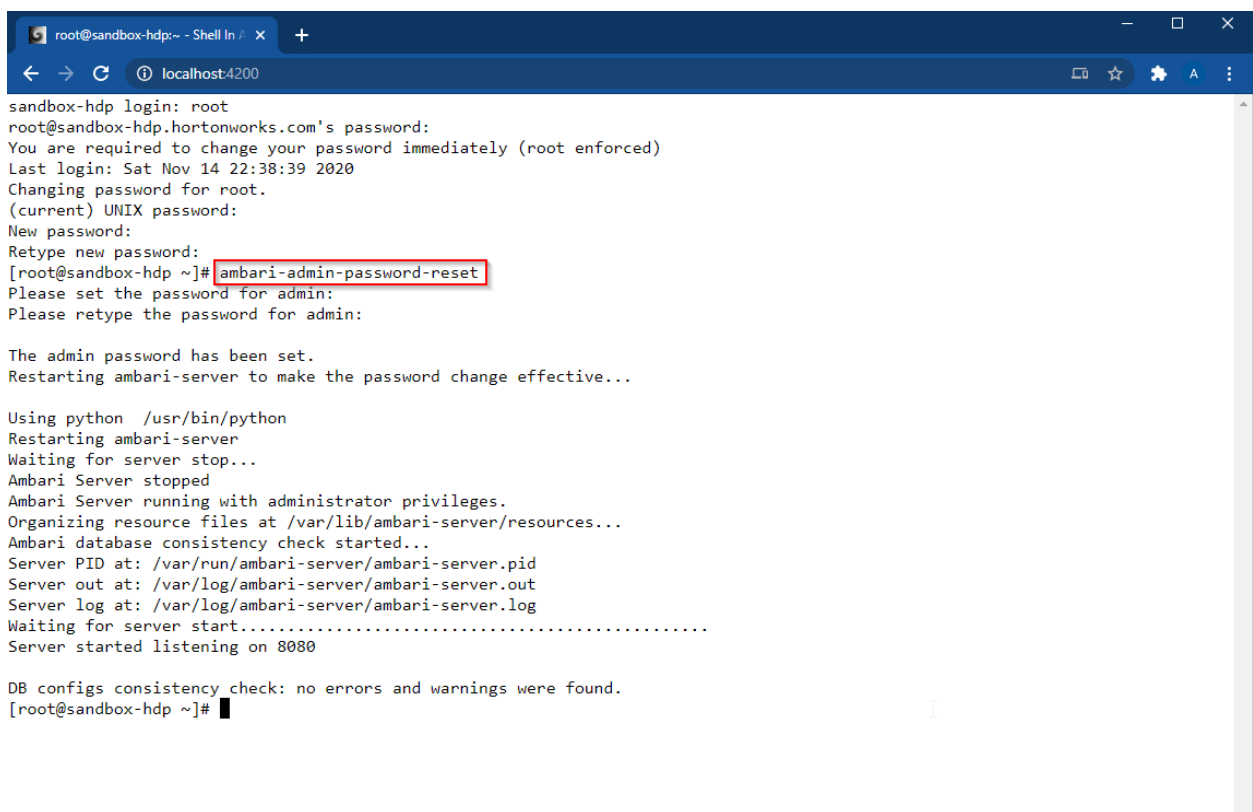
Admin Password Reset

Step 1: Open web browser from your computer and access Shell Web Client (<http://localhost:4200/>). Login using credentials: root / Hadoop. It will prompt you to change your password as shown below.



```
root@sandbox-hdp:~ - Shell In / x +
localhost:4200
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
You are required to change your password immediately (root enforced)
Last login: Sat Nov 14 22:38:39 2020
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox-hdp ~]#
```

Step 2: You will need to reset the Ambari admin password. Type the following commands: `ambari-admin-password-reset` in the SSH interface for this.



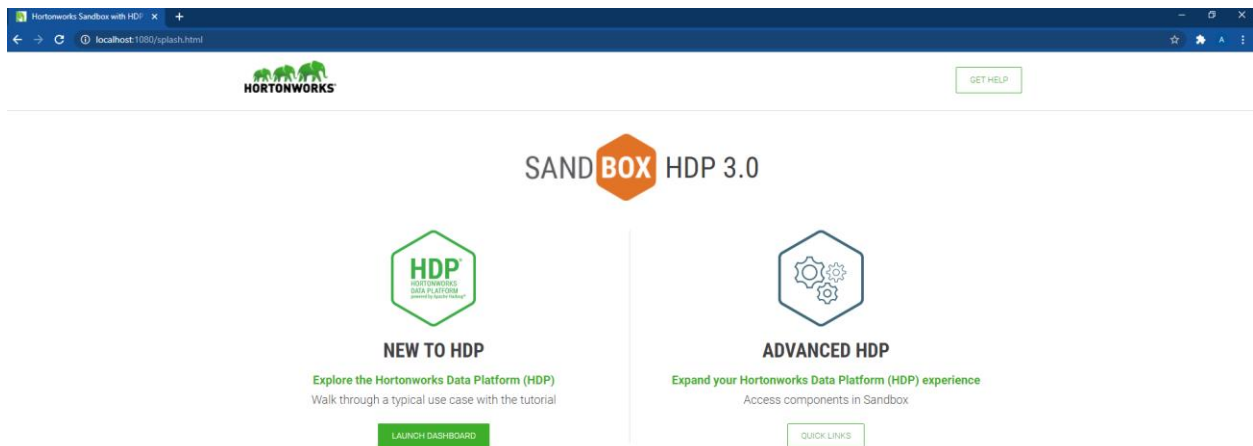
```
root@sandbox-hdp:~ - Shell In / x +
localhost:4200
sandbox-hdp login: root
root@sandbox-hdp.hortonworks.com's password:
You are required to change your password immediately (root enforced)
Last login: Sat Nov 14 22:38:39 2020
Changing password for root.
(current) UNIX password:
New password:
Retype new password:
[root@sandbox-hdp ~]# ambari-admin-password-reset
Please set the password for admin:
Please retype the password for admin:

The admin password has been set.
Restarting ambari-server to make the password change effective...

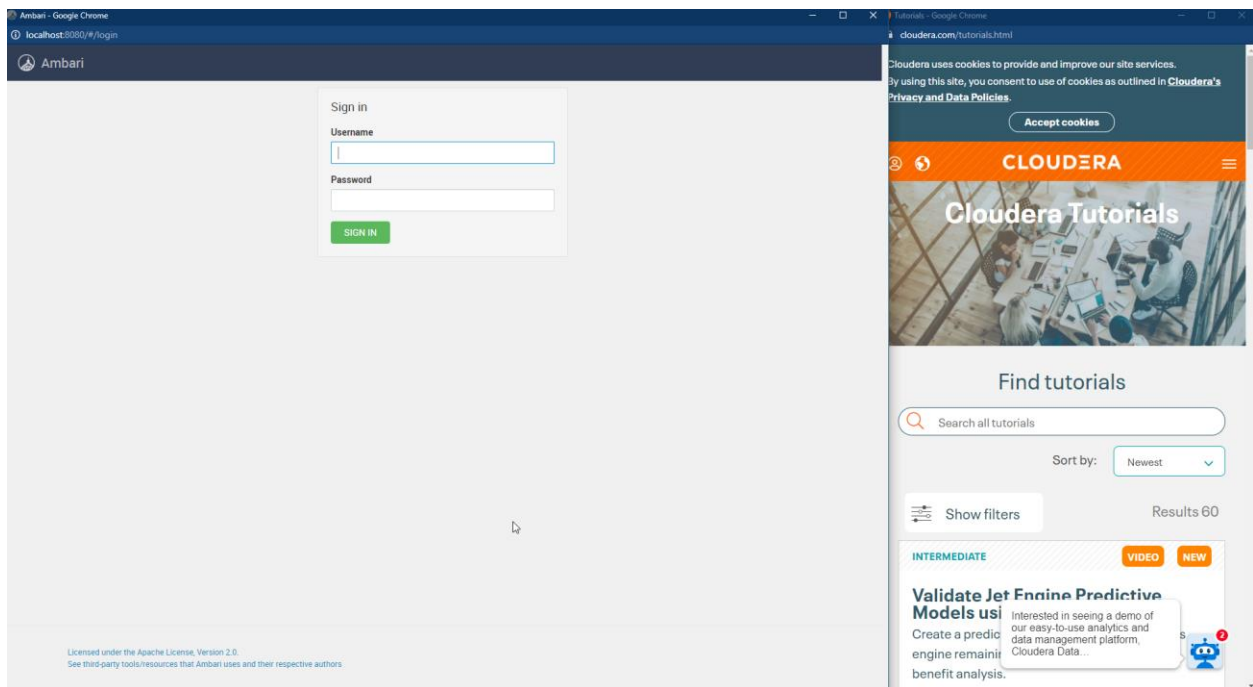
Using python /usr/bin/python
Restarting ambari-server
Waiting for server stop...
Ambari Server stopped
Ambari Server running with administrator privileges.
Organizing resource files at /var/lib/ambari-server/resources...
Ambari database consistency check started...
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Waiting for server start.....
Server started listening on 8080

DB configs consistency check: no errors and warnings were found.
[root@sandbox-hdp ~]#
```

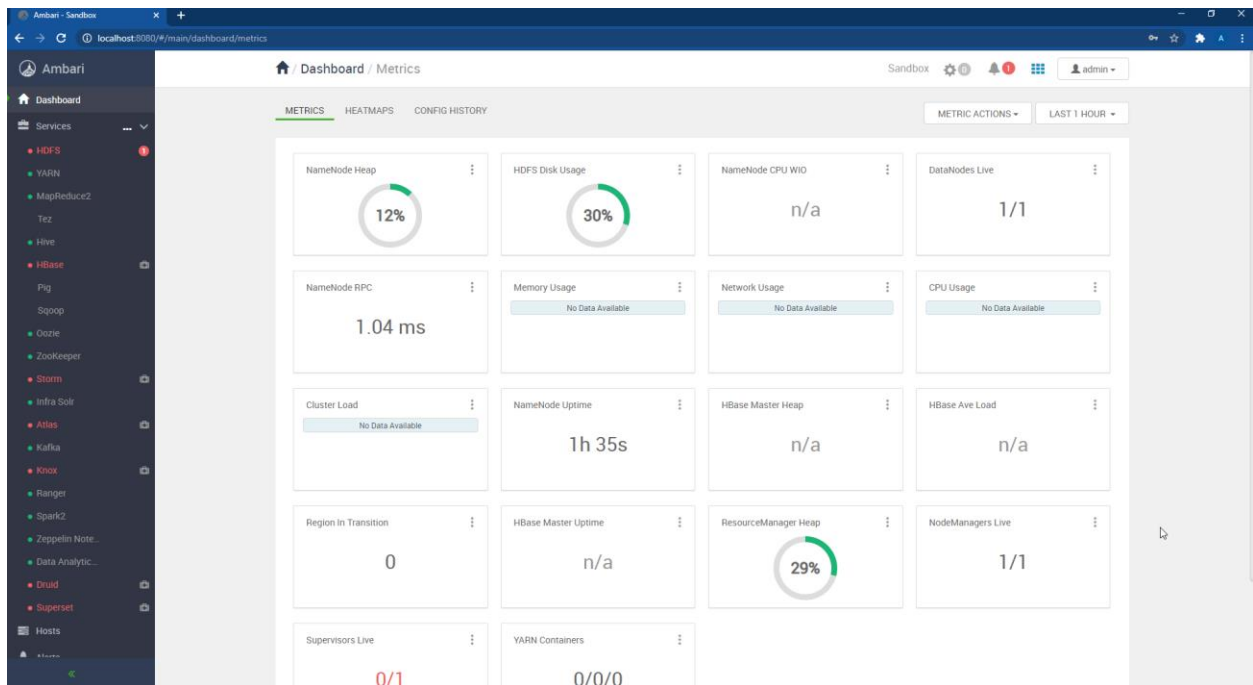
Step 3: Open web browser from your computer and access Web Client (<http://localhost:1080/>) to get the below screen. Click “Launch Dashboard”.



This will open login screen and cloudera tutorial link for you as shown below.

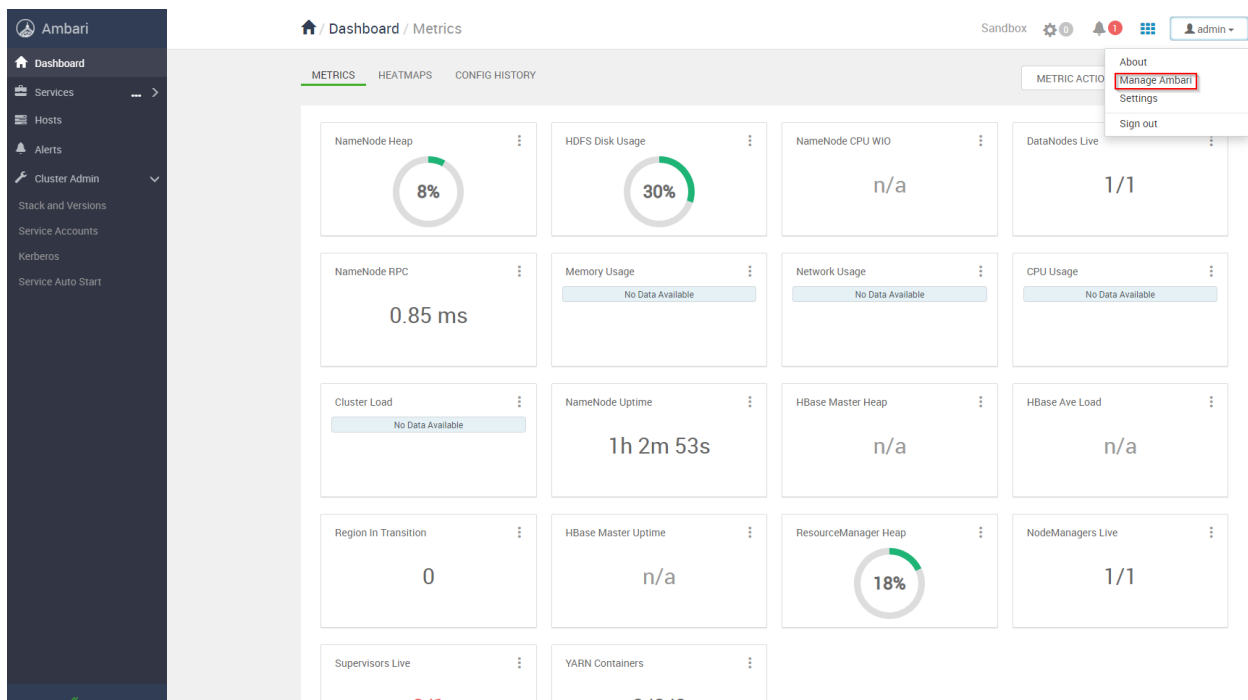


Step 4: Login using credentials: admin / <new password>.



Explore Ambari

Ambari is a completely open source management platform for provisioning, managing, monitoring and securing Apache Hadoop clusters. Login to as admin and Select “Manage Ambari”.



You will be able to view the below screens

1. “Dashboard” will take you to the Ambari Dashboard which is the primary UI for Hadoop Operators
2. “Cluster Management” allows you to grant permission to Ambari users and groups

3. "Administration of Users" allows you to add & remove Ambari users and groups
4. "Ambari User Views" list the set of Ambari Users views that are part of the cluster

Admin / Cluster Information

Cluster Name*
Sandbox

Cluster Blueprint

```
{
  "configurations": {
    {
      "zeppelin-log4j-properties": {
        "properties_attributes": {},
        "properties": {
          "log4j_properties_content": "\nlog4j.rootLogger = INFO, dailyfile\nlog4j.appender.stdout = org.apache.log4j.ConsoleAppender\nlog4j.appender.stdout.layout = org.apache.log4j.PatternLayout\nlog4j.appender.stdout.layout.ConversionPattern=%5p [%d] (%t) %F[%M]%L - %m%n\nlog4j.appender.dailyfile.DatePattern=yyyy-MM-dd\nlog4j.appender.dailyfile.Threshold = INFO\nlog4j.appender.dailyfile = org.apache.log4j.DailyRollingFileAppender\nlog4j.appender.dailyfile.File = ${zeppelin.log.file}\nlog4j.appender.dailyfile.layout = org.apache.log4j.PatternLayout\nlog4j.appender.dailyfile.layout.ConversionPattern=%5p [%d] (%t) %F[%M]%L - %m%n"
        }
      },
      "zoo.cfg": {
        "properties_attributes": {},
        "properties": {
          "autopurge.purgeinterval": "24",
          "dataDir": "/hadoop/zookeeper",
          "autopurge.snapRetainCount": "30",
          "autopurge.maxLogFileSize": "100MB"
        }
      }
    }
  }
}
```

DOWNLOAD

Default Users in Hadoop Sandbox

Admin / Users

USERS GROUPS

Username	Role	Status	Type	Group	Actions
admin	Ambari Administrator	Active	Local	-	
amy_ds	Service Operator	Active	Local	users	
maria_dev	Service Operator	Active	Local	users	
raj_ops	Cluster Administrator	Active	Local	users	

ADD USERS

User	Password	Role	Services
admin	Refer here	Ambari Admin	Ambari
maria_dev	maria_dev	Service Operator	Hive/Tez, Ranger, Falcon, Knox, Sqoop, Oozie, Flume, Zookeeper
raj_ops	raj_ops	Cluster Administrator	Hive, Zeppelin, MapReduce/Tez/Spark, Pig, Solr, HBase/Phoenix, Sqoop, NiFi, Storm, Kafka, Flume
amy_ds	amy_ds	Service Operator	Atlas

Getting Started with HDP Sandbox

Concepts

Apache Hadoop is an open source framework for distributed storage and processing of large sets of data on commodity hardware. Hadoop enables businesses to quickly gain insight from massive amounts of structured and unstructured data. Numerous Apache Software Foundation projects make up the services required by an enterprise to deploy, integrate and work with Hadoop.

The base Apache Hadoop framework is composed of the following [modules](#):

- Hadoop Common – contains libraries and utilities needed by other Hadoop modules.

- Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users’ applications.
- Hadoop MapReduce – a programming model for large scale data processing.

The below five [pillars](#) make Hadoop enterprise ready

Data Management

Store and process vast quantities of data in a storage layer that scales linearly.

- **Hadoop Distributed File System (HDFS)** is the core technology for the efficient scale out storage layer, and is designed to run across low-cost commodity hardware.
- **Apache Hadoop YARN** is the pre-requisite for Enterprise Hadoop as it provides the resource management and pluggable architecture for enabling a wide variety of data access methods to operate on data stored in Hadoop with predictable performance and service levels.

Data Access

Interact with your data in a wide variety of ways – from batch to real-time.

- **Apache Hive** - Built on the MapReduce framework, Hive is a data warehouse that enables easy data summarization and ad-hoc queries via an SQL-like interface for large datasets stored in HDFS.
- **Apache Pig** - A platform for processing and analyzing large data sets.
- **MapReduce** – MapReduce is a framework for writing applications that process large amounts of structured and unstructured data in parallel across a cluster of thousands of machines, in a reliable and fault-tolerant manner.
- **Apache Spark** – Spark is ideal for in-memory data processing. It allows data scientists to implement fast, iterative algorithms for advanced analytics such as clustering and classification of datasets.
- **Apache Storm** - Storm is a distributed real-time computation system for processing fast, large streams of data adding reliable real-time data processing capabilities to Apache Hadoop
- **Apache HBase** - A column-oriented NoSQL data storage system that provides random real-time read/write access to big data for user applications.
- **Apache Tez** - Tez generalizes the MapReduce paradigm to a more powerful framework for executing a complex DAG (directed acyclic graph) of tasks for near real-time big data processing.
- **Apache Kafka** – Kafka is a fast and scalable publish-subscribe messaging system that is often used in place of traditional message brokers because of its higher throughput, replication, and fault tolerance.
- **Apache HCatalog** – A table and metadata management service that provides a centralized way for data processing systems to understand the structure and location of the data stored within Apache Hadoop.
- **Apache Slider** – A framework for deployment of long-running data access applications in Hadoop. Slider leverages YARN’s resource management capabilities to deploy those applications, to manage their lifecycles and scale them up or down.

- **Apache Solr** – Solr is the open source platform for searches of data stored in Hadoop. Solr enables powerful full-text search and near real-time indexing on many of the world’s largest Internet sites.
- **Apache Mahout** – Mahout provides scalable machine learning algorithms for Hadoop which aids with data science for clustering, classification and batch based collaborative filtering.
- **Apache Accumulo** – Accumulo is a high-performance data storage and retrieval system with cell-level access control. It is a scalable implementation of Google’s Big Table design that works on top of Apache Hadoop and Apache ZooKeeper.

Data Governance and Integration

Quickly and easily load data, and manage according to policy.

- **Workflow Management** – Workflow Manager allows you to easily create and schedule workflows and monitor workflow jobs. It is based on the Apache Oozie workflow engine that allows users to connect and automate the execution of big data processing tasks into a defined workflow.
- **Apache Flume** – Flume allows you to efficiently aggregate and move large amounts of log data from many different sources to Hadoop.
- **Apache Sqoop** – Sqoop is a tool that speeds and eases movement of data in and out of Hadoop. It provides a reliable parallel load for various, popular enterprise data sources.

Security

Address requirements of Authentication, Authorization, Accounting and Data Protection. Security is provided at every layer of the Hadoop stack from HDFS and YARN to Hive and the other Data Access components on up through the entire perimeter of the cluster via Apache Knox.

- **Apache Knox** – The Knox Gateway (“Knox”) provides a single point of authentication and access for Apache Hadoop services in a cluster. The goal of the project is to simplify Hadoop security for users who access the cluster data and execute jobs, and for operators who control access to the cluster.
- **Apache Ranger** – Apache Ranger delivers a comprehensive approach to security for a Hadoop cluster. It provides central security policy administration across the core enterprise security requirements of authorization, accounting and data protection.

Operations

Provision, manage, monitor and operate Hadoop clusters at scale.

- **Apache Ambari** – An open source installation lifecycle management, administration and monitoring system for Apache Hadoop clusters.
- **Apache Oozie** – Oozie Java Web application used to schedule Apache Hadoop jobs. Oozie combines multiple jobs sequentially into one logical unit of work.

- **Apache ZooKeeper** – A highly available system for coordinating distributed processes. Distributed applications use ZooKeeper to store and mediate updates to important configuration information.

HDFS

HDFS is a distributed file system that is designed for storing large data files. HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers. An HDFS cluster is comprised of a NameNode, which manages the cluster metadata, and DataNodes that store the data.

Files and directories are represented on the NameNode by inodes. Inodes record attributes like permissions, modification and access times, or namespace and disk space quotas.

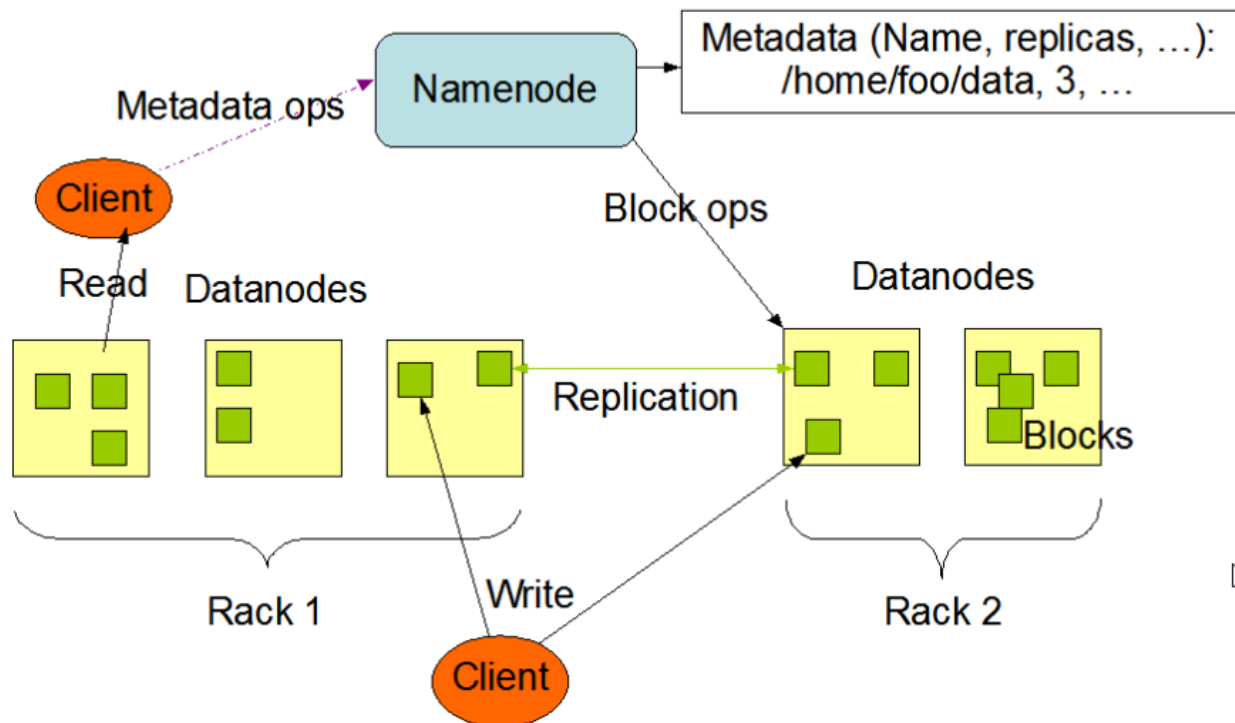
The file content is split into large blocks (typically 128 megabytes), and each block of the file is independently replicated at multiple DataNodes. The blocks are stored on the local file system on the DataNodes.

The Namenode actively monitors the number of replicas of a block. When a replica of a block is lost due to a DataNode failure or disk failure, the NameNode creates another replica of the block. The NameNode maintains the namespace tree and the mapping of blocks to DataNodes, holding the entire namespace image in RAM.

The NameNode does not directly send requests to DataNodes. It sends instructions to the DataNodes by replying to heartbeats sent by those DataNodes. The instructions include commands to:

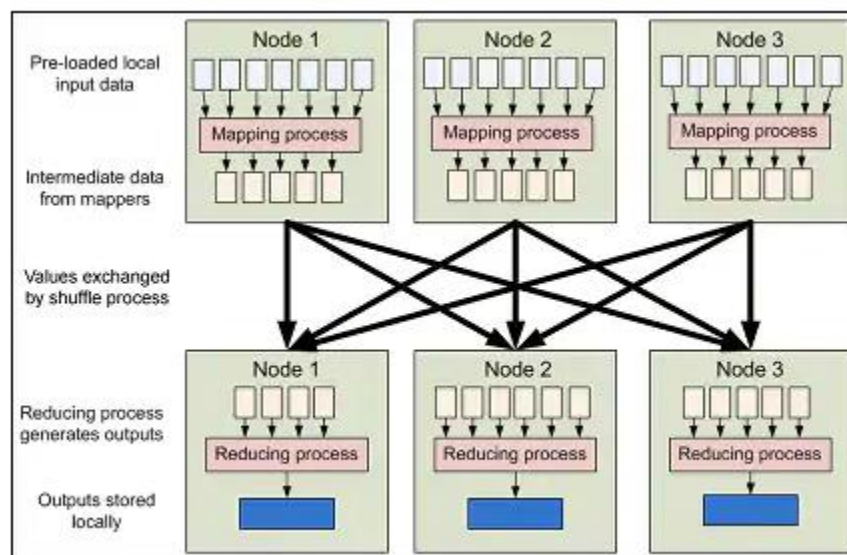
- replicate blocks to other nodes
- remove local block replicas
- re-register and send an immediate block report, or
- shut down the node

HDFS Architecture



Apache MapReduce

MapReduce is the key algorithm that the Hadoop data processing engine uses to distribute work around a cluster. A MapReduce job splits a large data set into independent chunks and organizes them into key, value pairs for parallel processing. This parallel processing improves the speed and reliability of the cluster, returning solutions more quickly and with greater reliability.



The *Map* function divides the input into ranges by the InputFormat and creates a map task for each range in the input. The JobTracker distributes those tasks to the worker nodes. The output of each map task is partitioned into a group of key-value pairs for each reduce.

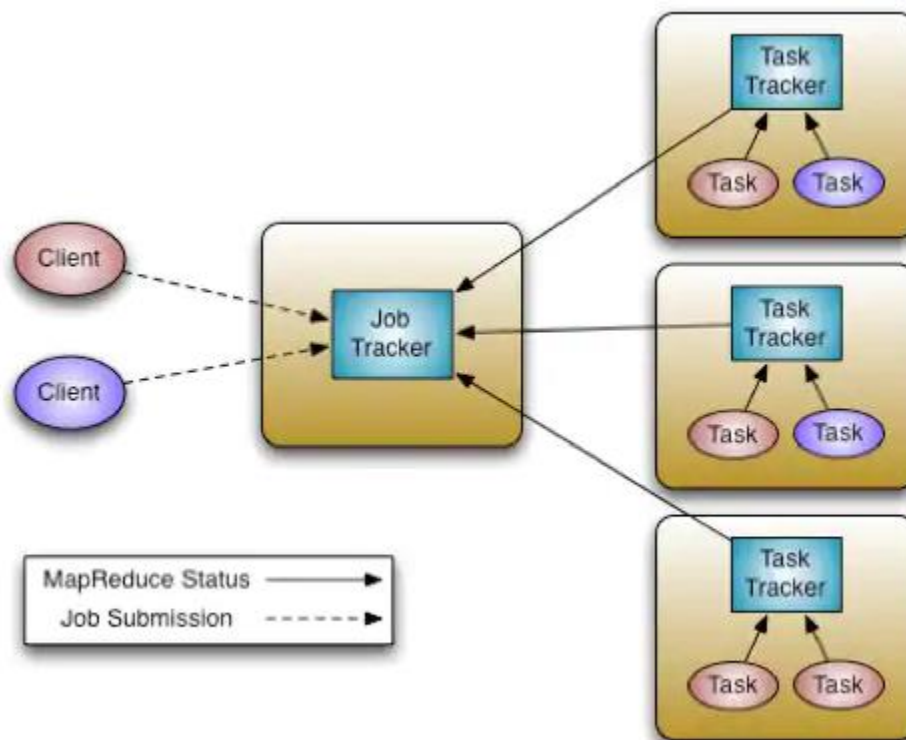
- `map(key1,value) -> list<key2,value2>`

The *Reduce* function then collects the various results and combines them to answer the larger problem that the master node needs to solve. Each reduce pulls the relevant partition from the machines where the maps executed, then writes its output back into HDFS. Thus, the reduce is able to collect the data from all of the maps for the keys and combine them to solve the problem.

- `reduce(key2, list<value2>) -> list<value3>`

The current Apache Hadoop MapReduce System is composed of the JobTracker, which is the master, and the per-node slaves called TaskTrackers.

- The JobTracker is responsible for resource management (managing the worker nodes i.e. TaskTrackers), tracking resource consumption/availability and also job life-cycle management (scheduling individual tasks of the job, tracking progress, providing fault-tolerance for tasks etc).
- The TaskTracker has simple responsibilities – launch/teardown tasks on orders from the JobTracker and provide task-status information to the JobTracker periodically.



Apache Hive

Hive is an SQL like query language that enables those analysts familiar with SQL to run queries on large volumes of data. Hive has three main functions: data summarization, query and analysis. Hive provides tools that enable easy data extraction, transformation and loading (ETL).

The tables in Hive are similar to tables in a relational database, and data units are organized in a taxonomy from larger to more granular units. Databases are comprised of tables, which are made up of partitions. Data can be accessed via a simple query language and Hive supports overwriting or appending data.

Within a particular database, data in the tables is serialized and each table has a corresponding Hadoop Distributed File System (HDFS) directory. Each table can be sub-divided into partitions that determine how data is distributed within sub-directories of the table directory. Data within partitions can be further broken down into buckets.

Apache ORC

Apache ORC is a fast columnar storage file format for Hadoop workloads.

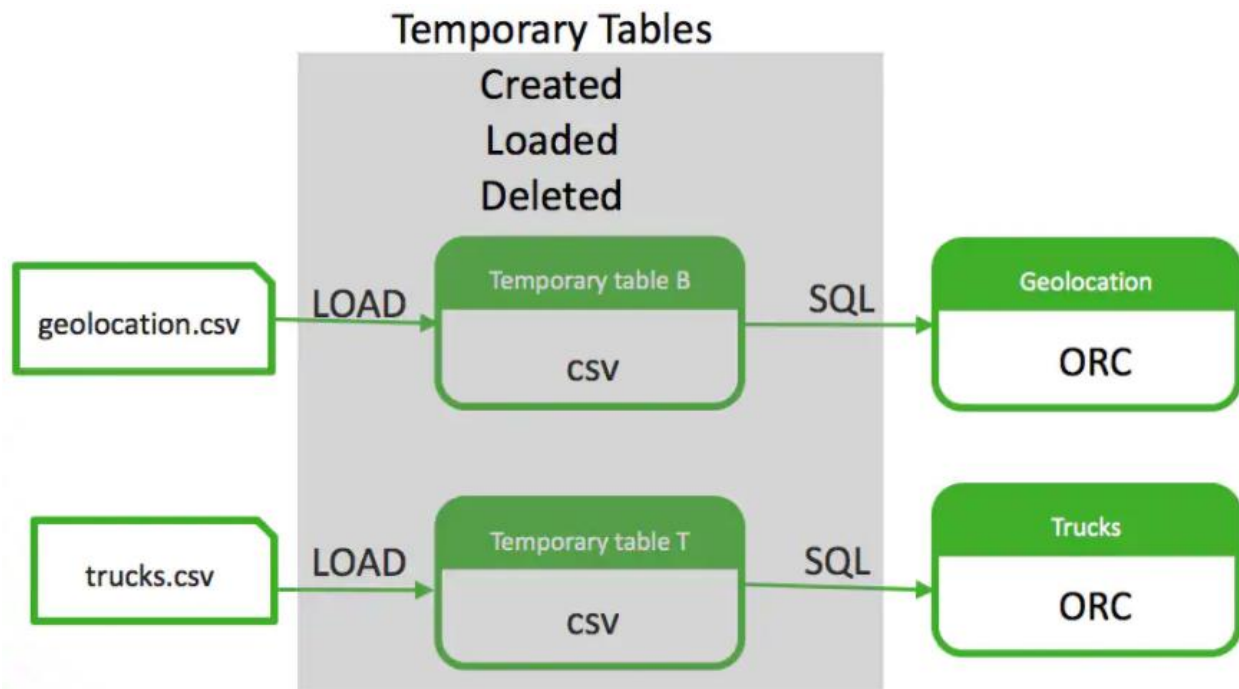
The Optimized Row Columnar (Apache ORC project) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using ORC files improves performance when Hive is reading, writing, and processing data.

To create a table using the ORC file format, use STORED AS ORC option. For example:

```
CREATE TABLE <tablename> ... STORED AS ORC ...
```

Following is a visual representation of the Upload table creation process:

1. The target table is created using ORC file format (i.e. Geolocation)
2. A temporary table is created using TEXTFILE file format to store data from the CSV file
3. Data is copied from temporary table to the target (ORC) table
4. Finally, the temporary table is dropped



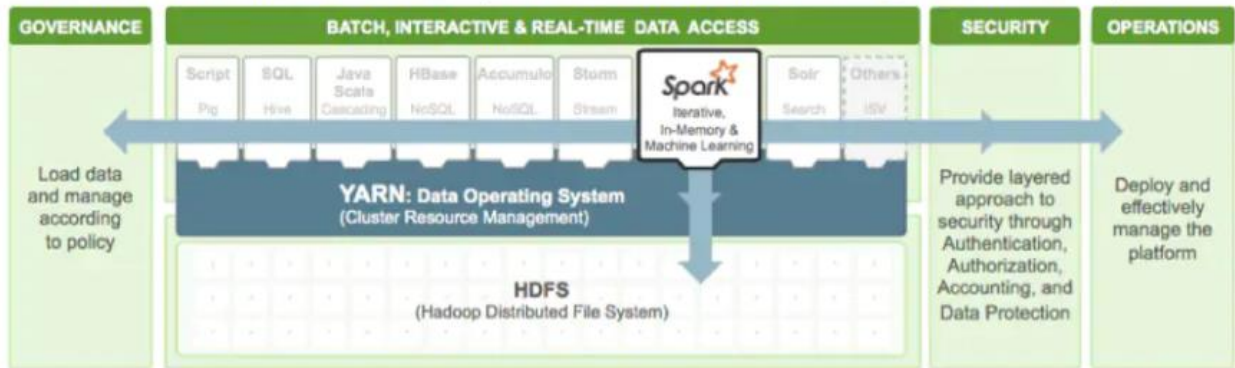
You can review the SQL statements issued by selecting the **Queries** tab and reviewing the four most recent jobs, which was a result of using the **Upload Table**.

QUERY	STATUS	USER	TABLES READ	TABLES WRITTEN	START TIME	DURATION	DAG ID	APPLICATION	ACTIONS
CREATE TABLE 'default'. 'trucks' ('drived'...	SUCCESS	hive	Not Available	trucks (default)	18 minutes ago	00:00:01	Not Available	Not Available	🔗 🔍
DROP TABLE 'default'. 'dwkdyorpozvzhtytw'...	SUCCESS	hive	dwkdyorpozvzhtytw...	dwkdyorpozvzhtytw...	3 hours ago	00:00:07	Not Available	Not Available	🔗 🔍
FROM 'default'. 'dwkdyorpozvzhtytw'...	SUCCESS	hive	dwkdyorpozvzhtytw...	geolocation (default)	3 hours ago	00:07:52	Not Available	Not Available	🔗 🔍
CREATE TABLE 'default'. 'dwkdyorpozvzht'...	SUCCESS	hive	Not Available	dwkdyorpozvzhtytw...	3 hours ago	00:00:00	Not Available	Not Available	🔗 🔍
CREATE TABLE 'default'. 'geolocation' ('x'...	SUCCESS	hive	Not Available	geolocation (default)	3 hours ago	00:00:01	Not Available	Not Available	🔗 🔍

Apache Spark

Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs in Scala, Java, Python and R that allow data workers to efficiently execute machine learning algorithms that require fast iterative access to datasets. Spark on Apache Hadoop YARN enables deep integration with Hadoop and other YARN enabled workloads in the enterprise.

Hortonworks Investment Themes for Spark: Vertical and Horizontal Integration with Hadoop



Apache Zeppelin

A completely open web-based notebook that enables interactive data analytics

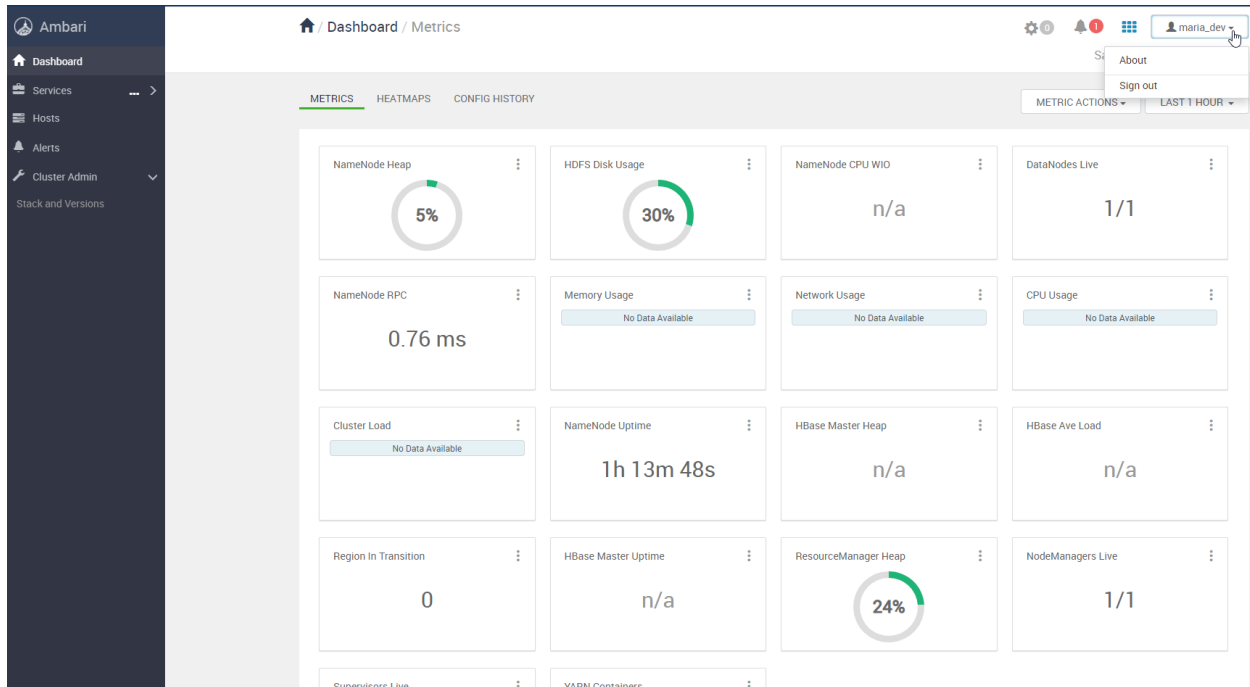
Apache Zeppelin is a web-based notebook that enables interactive data analytics. With Zeppelin, you can make beautiful data-driven, interactive and collaborative documents with a rich set of pre-built language back-ends (or interpreters) such as Scala (with Apache Spark), Python (with Apache Spark), SparkSQL, Hive, Markdown, Angular, and Shell.

With a focus on Enterprise, Zeppelin has the following important features:

- Livy integration (REST interface for interacting with Spark)
- Security:
 - Execute jobs as authenticated user
 - Zeppelin authentication against LDAP
 - Notebook authorization

Loading Sensor Data into HDFS

1: Logon to Ambari using: maria_dev/maria_dev



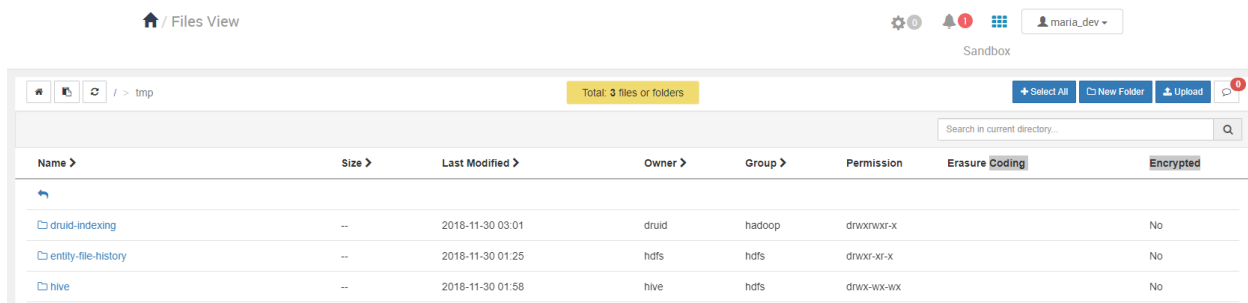
2. Go to Ambari Dashboard and open Files View.

The Ambari Files View displays a list of files and folders in the HDFS file system. The top navigation bar shows the current page as Files View. A dropdown menu is open, showing options for Views, Files View, and Workflow Manager. The table below lists the files and folders, including their Name, Size, Last Modified, Owner, Group, Permission, Erasure Coding, and Encrypted status.

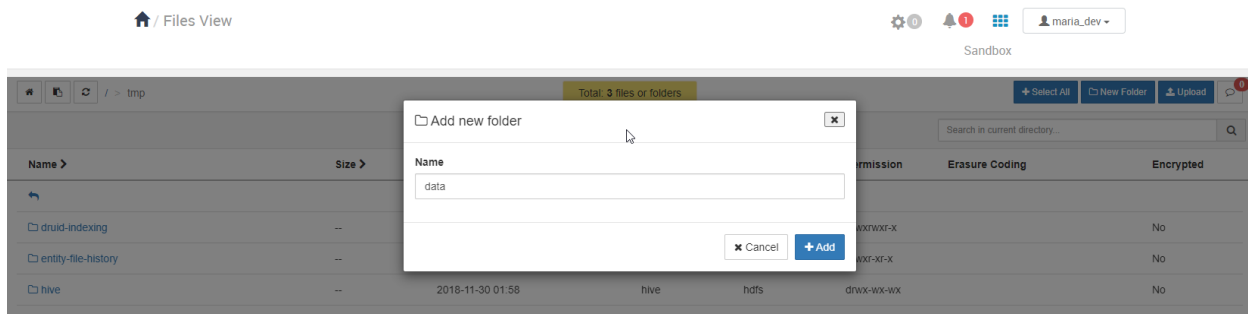
Name	Size	Last Modified	Owner	Group	Permission	Erasure Coding	Encrypted
app-logs	--	2018-11-30 01:56	yam	hadoop	drwxr-xr-x		No
apps	--	2018-11-30 03:01	hdfs	hdfs	drwxr-xr-x		No
ats	--	2018-11-30 01:25	yam	hadoop	drwxr-xr-x		No
atsv2	--	2018-11-30 01:26	hdfs	hdfs	drwxr-xr-x		No
hdp	--	2018-11-30 01:26	hdfs	hdfs	drwxr-xr-x		No
ivy2-recovery	--	2018-11-30 01:55	ivy	hdfs	drwxr-xr-x		No
mapred	--	2018-11-30 01:26	mapred	hdfs	drwxr-xr-x		No
mr-history	--	2018-11-30 01:26	mapred	hadoop	drwxr-xr-x		No
ranger	--	2018-11-30 02:54	hdfs	hdfs	drwxr-xr-x		No
spark2-history	--	2020-11-15 07:54	spark	hadoop	drwxr-xr-x		No
tmp	--	2018-11-30 03:01	hdfs	hdfs	drwxr-xr-x		No
user	--	2018-11-30 03:21	hdfs	hdfs	drwxr-xr-x		No
warehouse	--	2018-11-30 01:51	hdfs	hdfs	drwxr-xr-x		No

3. Start from the top root of the HDFS file system, you will see all the files the logged in user (maria_dev in this case) has access to see:

4. Navigate to /tmp/ directory by clicking on the directory links.



5. Create directory data. Click the **New Folder** button to create that directory. Then navigate to it. The directory path you should see: /tmp/data

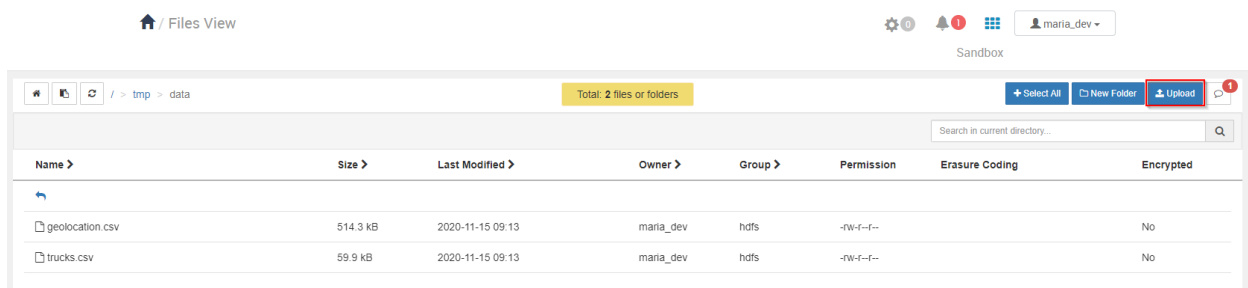


6. If you're not already in your newly created directory path /tmp/data, go to the data folder. Then click on the **Upload** button to upload the corresponding geolocation.csv and trucks.csv files into it.

7. An Upload file window will appear, click on the cloud symbol.

8. Another window will appear, navigate to the destination the two csv files were downloaded. Click on one at a time, press open to complete the upload. Repeat the process until both files are uploaded.

9. Both files are uploaded to HDFS as shown in the Files View UI as shown below. You can also perform the following operations on a file or folder by clicking on the entity's row: Open, Rename, Permissions, Delete, Copy, Move, Download and Concatenate.



10. Click on the data folder's row, which is contained within the directory path /tmp/.

11. Click Permissions. Make sure that the background of all the write boxes are checked (blue).

The screenshot shows the 'Files View' interface in the Hortonworks Sandbox. The 'Permissions' tab is selected, and a 'Directory Edit Permissions' dialog box is open. The dialog box has three sections: 'User', 'Group', and 'Other'. Each section has three buttons: 'Read', 'Write', and 'Execute'. The 'Write' buttons are highlighted in blue, indicating they are checked. Below the dialog box, a table lists files and folders with their permissions.

Name	Size	Last Modified	Owner	Group	Permission	Erasure Coding	Encrypted
data	--	2020-11-15 09:13	maria_dev	hdfs	drwxrwxrwx		No
druid-indexing	--	2018-11-30 03:01	druid	hadoop	drwxrwxr-x		No
entity-file-history	--	2018-11-30 01:25	hdfs	hdfs	drwxr-xr-x		No
hive	--	2018-11-30 01:58	hive	hdfs	drwx-wx-wx		No

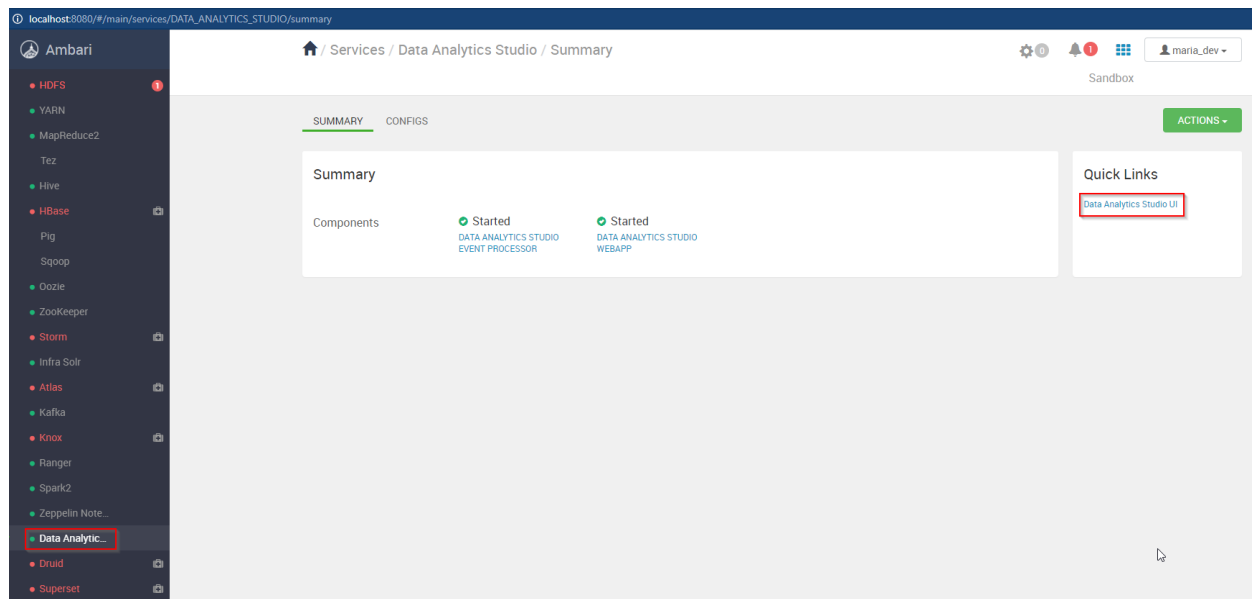
Hive - Data ETL

Data Analytics Studio

Apache Hive presents a relational view of data in HDFS. Hive can represent data in a tabular format managed by Hive or just stored in HDFS irrespective in the file format the data is in. Hive can query data from RCFile format, text files, ORC, JSON, parquet, sequence files and many of other formats in a tabular view. Through the use of SQL you can view your data as a table and create queries like you would in an RDBMS.

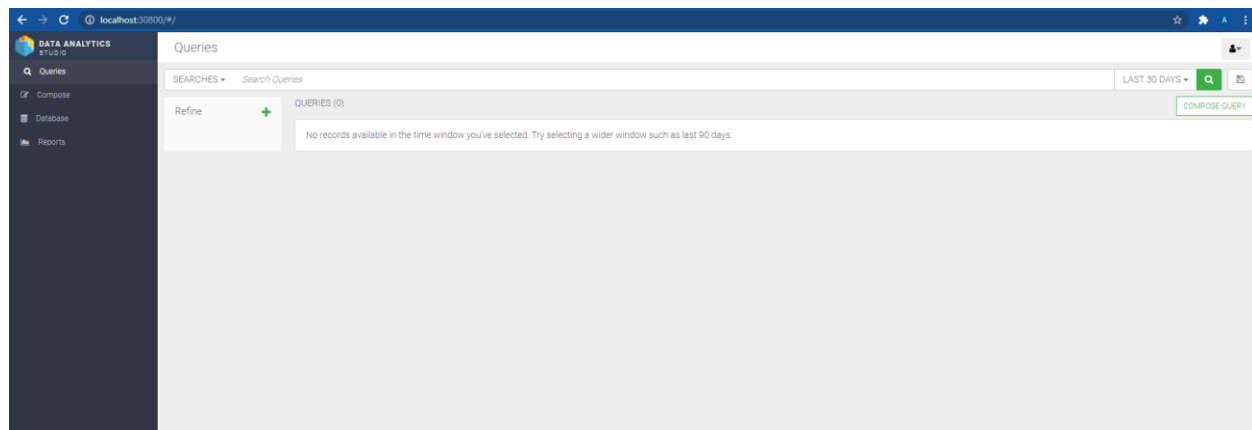
To make it easy to interact with Hive we use a tool in the Hortonworks Sandbox called Data Analytics Studio. DAS provides an interactive interface to Hive. We can create, edit, save and run queries, and have Hive evaluate them for us using a series of Tez jobs.

Let's now open DAS and get introduced to the environment. From Ambari Dashboard Select "Data Analytics Studio" and click on "Data Analytics Studio UI" or directly open <http://localhost:30800/>



There are 4 tabs to interact with Data Analytics Studio:

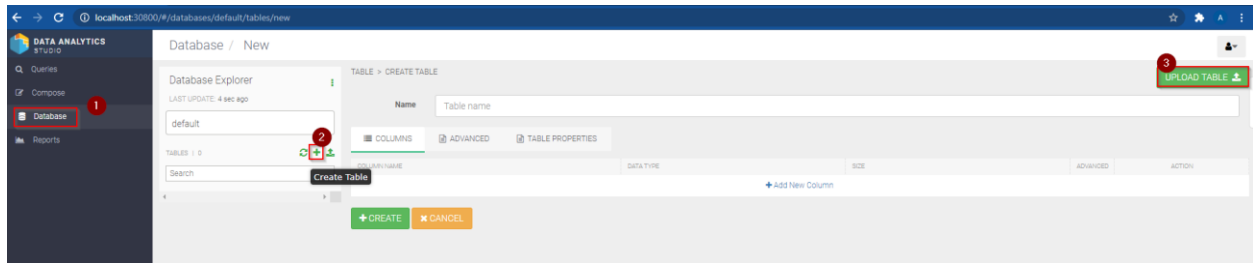
1. **Queries:** This view allows you to search previously executed SQL queries. You can also see commands each user issues.
2. **Compose:** From this view you can execute SQL queries and observe their output. Additionally, visually inspect the results of your queries and download them as csv files.
3. **Database:** Database allows you to add new Databases and Tables. Furthermore, this view grants you access to advanced information about your databases.
4. **Reports:** This view allows you keep track of Read and Write operations, and shows you a Join Report of your tables.



Create and Load Table

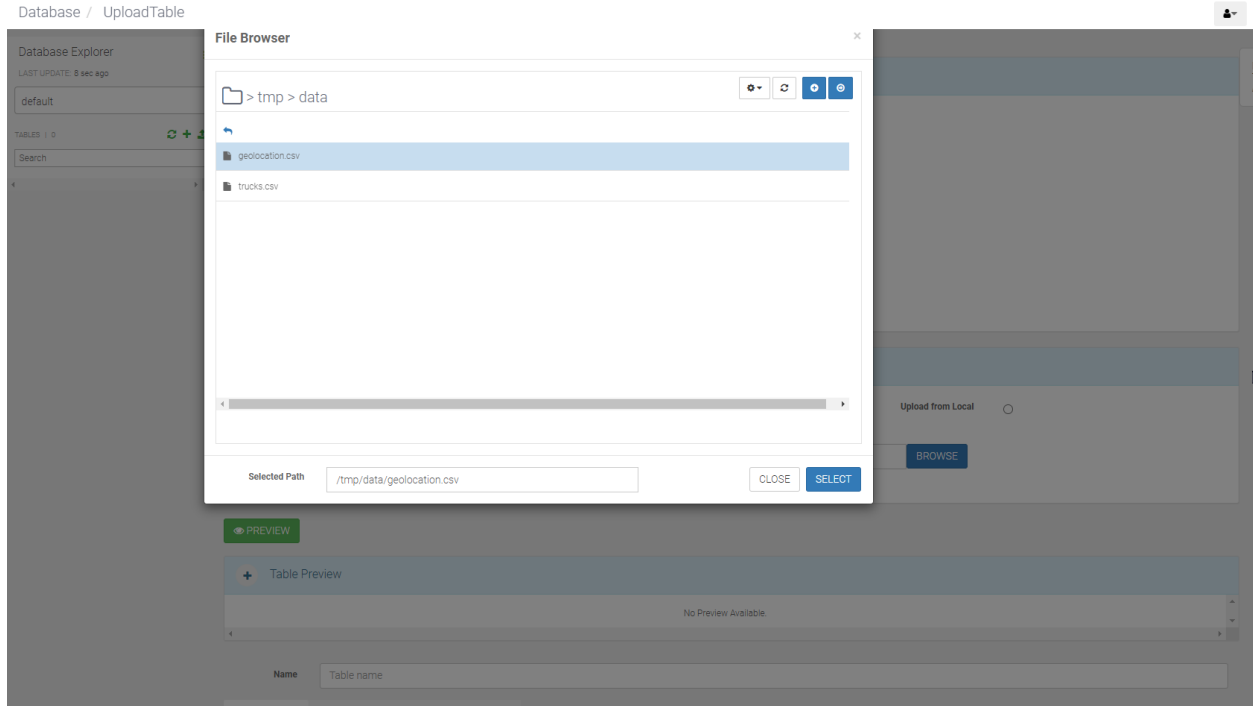
1. Open DAS Main Menu
2. Select Database
3. Select "+" next to Tables to add a new Table

4. Select Upload Table



5. Complete form as follows:

- Select checkbox: Is first row Header: True
- Select Upload from HDFS
- Set Enter HDFS Path to /tmp/data/geolocation.csv



6. Click Preview to view the below screen

- Verify that the first row contains the names of the columns in the table preview

Database / UploadTable

Database Explorer
LAST UPDATE: 13 sec ago

TABLES | 0

TABLE > UPLOAD TABLE

Select File Format

File type: Clear
Field Delimiter: Clear
Escape Character: Clear
Quote Character: Clear
Is first row header? ☒
Contains endlines? ☐

Select File Source

Upload from HDFS ☒ Upload from Local ☐
Enter Hdfs Path:

PREVIEW

Table Preview

TRUCKID	DRIVERID	EVENT	LATITUDE	LONGITUDE	CITY	STATE	VELOCITY	EVENT_IND	IDLING_IND
A54	A54	normal	38.440467	-122.714431	Sanita Rose	California	17	0	0
A20	A20	normal	36.977173	-121.899402	Aptos	California	27	0	0
A40	A40	overspeed	37.957702	-121.29078	Stockton	California	77	1	0
A31	A31	normal	39.409608	-123.355566	Willits	California	22	0	0

- And that column datatypes are properly identified as shown below

Name:

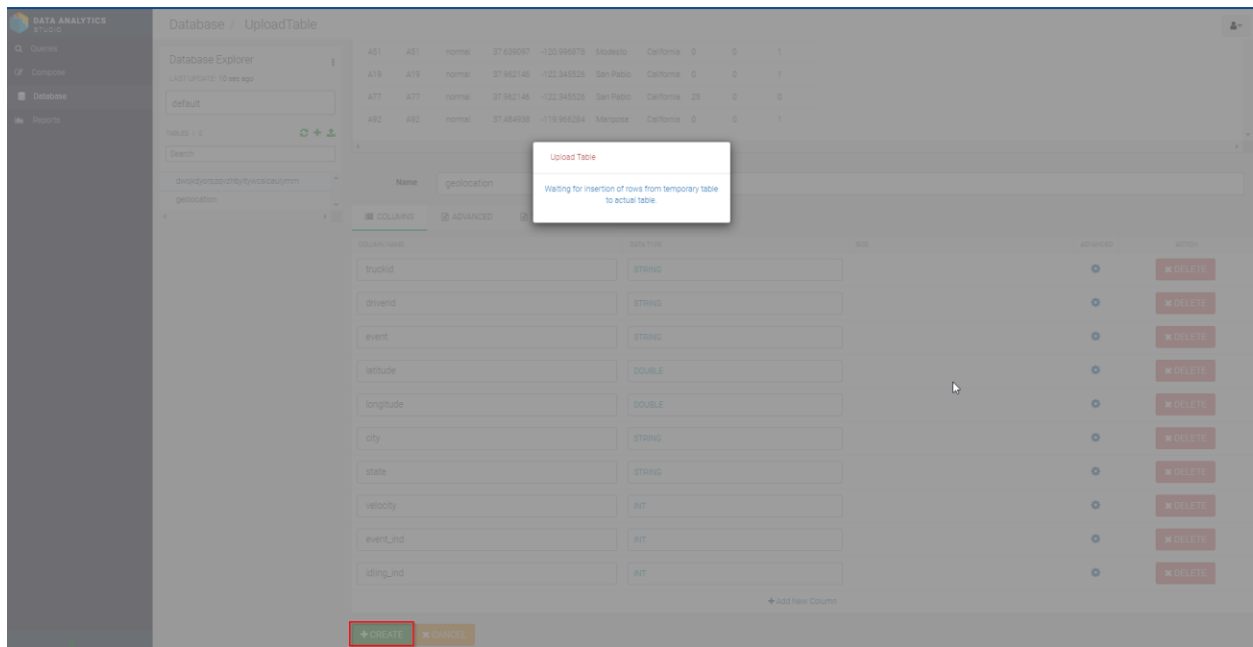
COLUMNS
ADVANCED
TABLE PROPERTIES

COLUMN NAME	DATA TYPE	SIZE	ADVANCED	ACTION
<input type="text" value="truckid"/>	<input type="text" value="STRING"/>			<input type="button" value="X DELETE"/>
<input type="text" value="driverid"/>	<input type="text" value="STRING"/>			<input type="button" value="X DELETE"/>
<input type="text" value="event"/>	<input type="text" value="STRING"/>			<input type="button" value="X DELETE"/>
<input type="text" value="latitude"/>	<input type="text" value="DOUBLE"/>			<input type="button" value="X DELETE"/>
<input type="text" value="longitude"/>	<input type="text" value="DOUBLE"/>			<input type="button" value="X DELETE"/>
<input type="text" value="city"/>	<input type="text" value="STRING"/>			<input type="button" value="X DELETE"/>
<input type="text" value="state"/>	<input type="text" value="STRING"/>			<input type="button" value="X DELETE"/>
<input type="text" value="velocity"/>	<input type="text" value="INT"/>			<input type="button" value="X DELETE"/>
<input type="text" value="event_ind"/>	<input type="text" value="INT"/>			<input type="button" value="X DELETE"/>
<input type="text" value="idling_ind"/>	<input type="text" value="INT"/>			<input type="button" value="X DELETE"/>

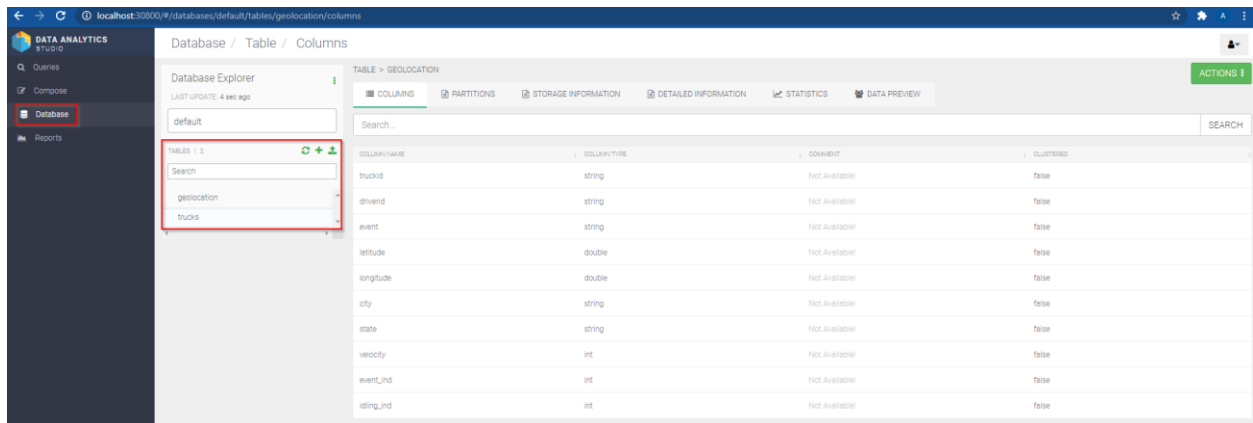
[+ Add New Column](#)

7. Click "Create"

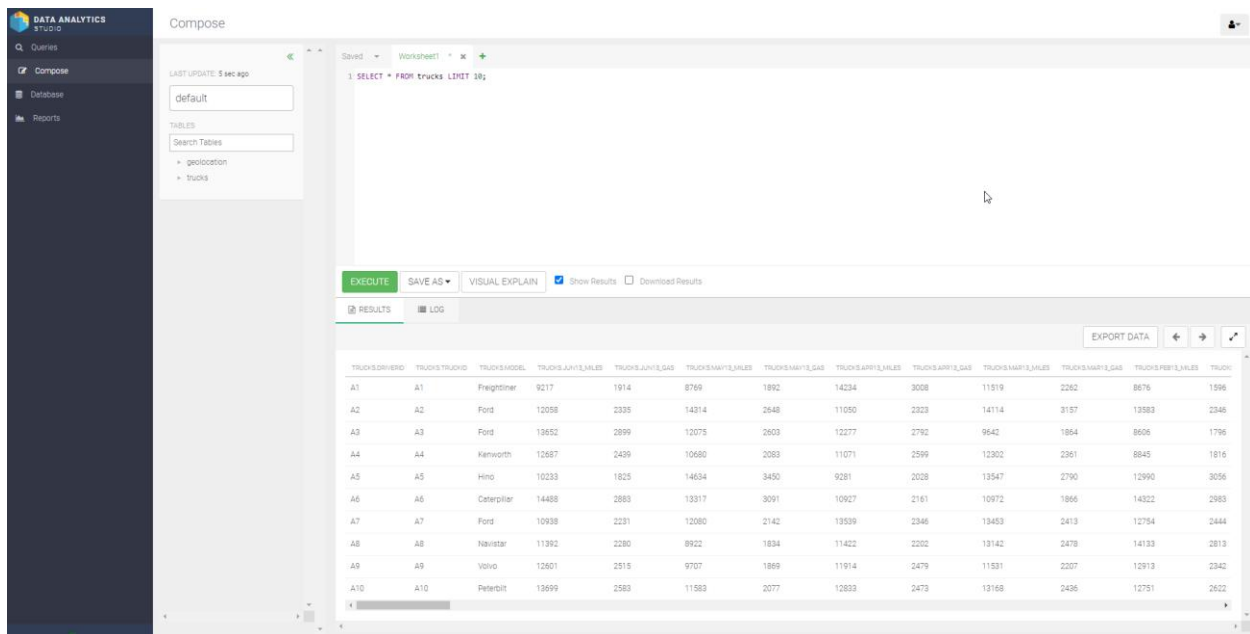
24



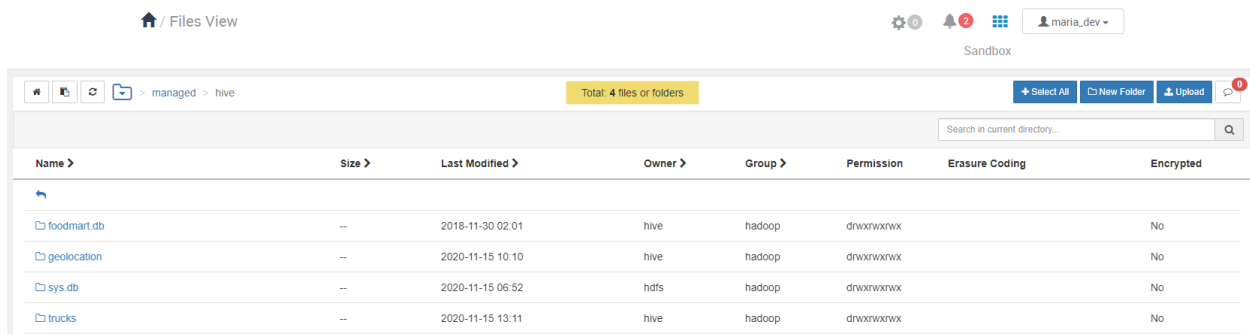
8. Verify that the new tables are created.



9. Query the database tables

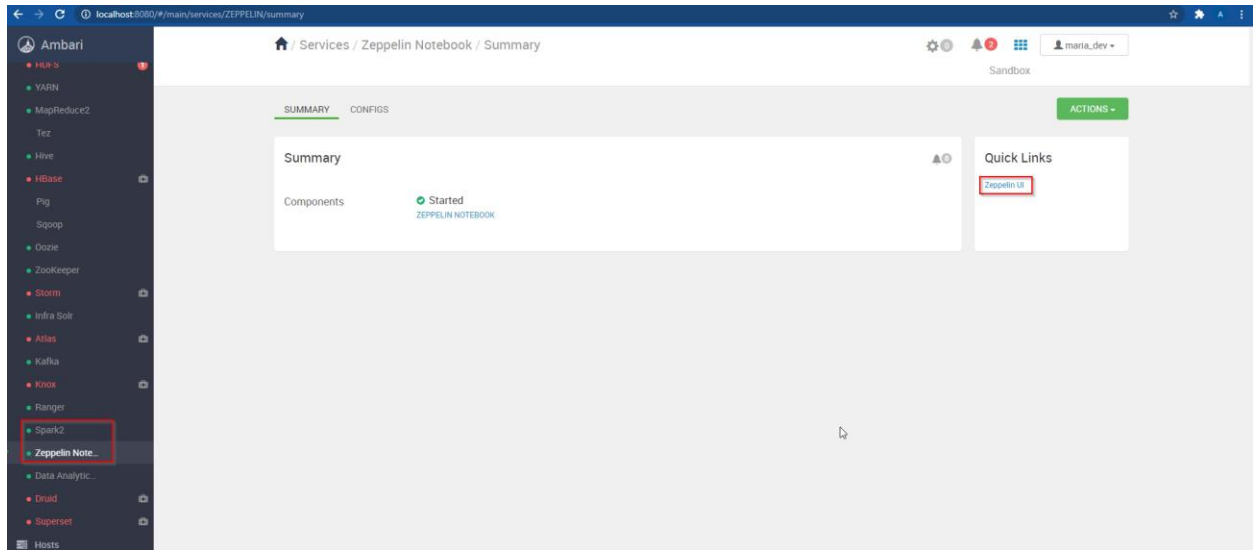


By default, when you create a table in Hive, a directory with the same name gets created in the **/warehouse/tablespace/managed/hive** folder in HDFS. Using the Ambari Files View, navigate to that folder. You should see both a geolocation and trucks directory:

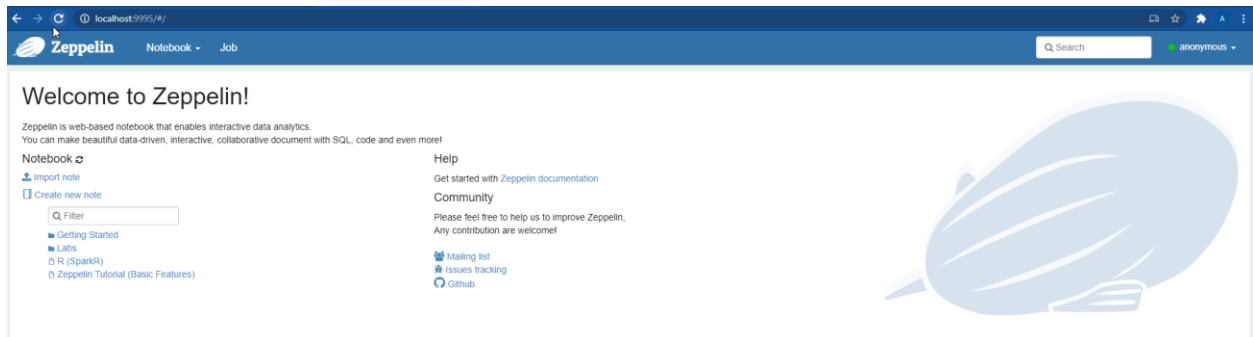


Spark

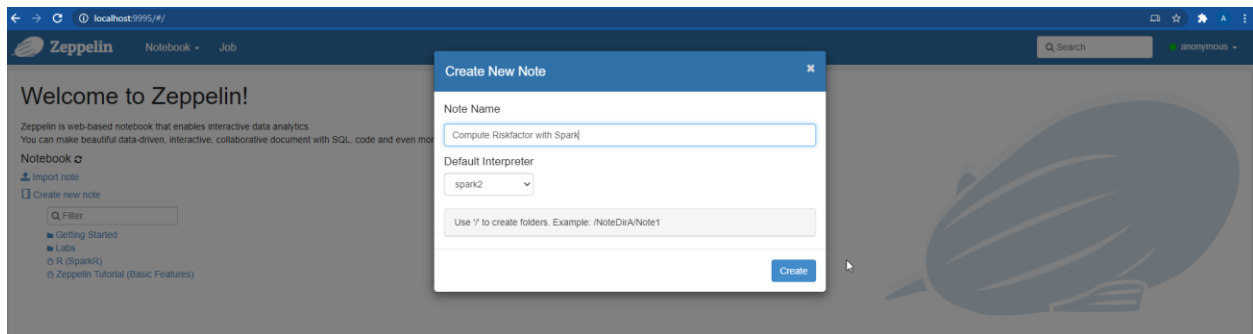
- Log on to Ambari Dashboard as maria_dev. At the bottom left corner of the services column, check that Spark2 and Zeppelin Notebook are running. If these services are disabled, start these services.



- Open Zeppelin via <http://localhost:9995/>



- Create new note and give name



- Create sparksession, write and execute code like in jupyter notebook using the zeppelin notebook.

