Steps to install required services on AWS Ec2 instance

1. Login to new EC2 instance using ssh

ssh -i aws-key.pem ubuntu@54.183.30.236

2. Login as root user to install base packages (java 8)
sudo su
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
Note: If you have any other version of Java, it is fine as long as you keep the directory paths proper in the below steps.

3. Check the java version
root@ip-172-31-23-142:/home/ubuntu# Java -version

INSTALLING HADOOP 2.7.3

4. Download the latest stable Hadoop using wget from one of the Apache mirrors.

root@ip-172-31-23-142:/home/ubuntu# wget
https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
--2020-04-05 13:02:47-- https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/hadoop-2.7.3.tar.gz
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 214092195 (204M) [application/x-gzip]
Saving to: 'hadoop-2.7.3.tar.gz'

hadoop-2.7.3.tar.gz
100%[==========>] 204.17M 15.4MB/s   in 19s

2020-04-05 13:03:07 (10.9 MB/s) - 'hadoop-2.7.3.tar.gz' saved [214092195/214092195]

root@ip-172-31-23-142:/home/ubuntu# tar -xvf hadoop-2.7.3.tar.gz

5. Create a directory where the hadoop will store its data. We will set this directory path in hdfs-site.

root@ip-172-31-23-142:/home/ubuntu# Java -version

6. Add the Hadoop related environment variables in your bash file.
root@ip-172-31-23-142:/home/ubuntu#vi ~/.bashrc

Copy and paste these environment variables.
#java variables
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export JRE_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
export PATH:$JAVA_HOME:$JRE_HOME/bin

#Hadoop variables
export HADOOP_HOME=/home/ubuntu/hadoop-2.7.3
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

Save and exit and use this command to refresh the bash settings.
root@ip-172-31-23-142:/home/ubuntu# source ~/.bashrc

7. Setting hadoop environment for password less ssh access. Password less SSH Configuration is a mandatory installation requirement. However it is more useful in a distributed environment.
root@ip-172-31-23-142:/home/ubuntu# ssh-keygen -t rsa -P ''
root@ip-172-31-23-142:/home/ubuntu# cat $HOME/.ssh/id_rsa.pub >> root@ip-172-31-23-142:/home/ubuntu# $HOME/.ssh/authorized_keys

check password less ssh access to localhost
root@ip-172-31-23-142:/home/ubuntu# ssh localhost
#exit from inner localhost shell
root@ip-172-31-23-142:/home/ubuntu# exit

8. Set the hadoop config files. We need to set the below files in order for hadoop to function properly.

- core-site.xml
- hadoop-env.sh
- yarn-site.xml
- hdfs-site.xml
- mapred-site.xml

go to directory where all the config files are present (cd /home/ubuntu/hadoop-2.6.0/etc/Hadoop)

- Copy and paste the below configurations in core-site.xml

##Add the following text between the configuration tabs.
<property>
<name>hadoop.tmp.dir</name>
<value>/home/ubuntu/hadoop tmp/hadoop-${user.name}</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>

- Copy and paste the below configurations in hadoop-env.sh

get the java home directory using:
readlink -f `which java`
Example output: /usr/lib/jvm/java-8-oracle/jre/bin/java (NOTE THE JAVA_HOME PATH. JUST GIVE THE BASE DIRECTORY PATH)
##Need to set JAVA_HOME in hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

- Copy and paste the below configurations in mapred-site.xml

#copy mapred-site.xml from mapred-site.xml.template
cp mapred-site.xml.template mapred-site.xml
vi mapred-site.xml

#Add the following text between the configuration tabs.
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>

- Copy and paste the below configurations in yarn-site.xml

##Add the following text between the configuration tabs.
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>

- Copy and paste the below configurations in hdfs-site.xml

##Add the following text between the configuration tabs.
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property><name>dfs.name.dir</name>
<value>file:///home/ubuntu/hadoopdata/hdfs/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
<value>file:///home/ubuntu/hadoopdata/hdfs/datanode</value>
</property>

9. Formatting the HDFS file system via NameNode (after installing hadoop, for the first time we have to format the HDFS file system to make it work)
hdfs namenode -format

10. Issue the following commands to start hadoop
cd sbin/
./start-dfs.sh
./start-yarn.sh
#If you have properly done step 5, you can start Hadoop from any directory. (Note the user should be the one where you installed Hadoop)
start-all.sh
OR you can separately start required services as below:

Name node:
hadoop-daemon.sh start namenode
Data node:
hadoop-daemon.sh start datanode
Resource Manager:
yarn-daemon.sh start resourcemanager
Node Manager:
yarn-daemon.sh start nodemanager
Job History Server:
mr-jobhistory-daemon.sh start historyserver
Once hadoop has started point your browser to http://localhost:50070/

10. Check for hadoop processes /daemons running on hadoop with Java Virtual Machine Process Status Tool.
root@ip-172-31-23-142:/home/ubuntu# jps

OR you can check TCP and port details by using
root@ip-172-31-23-142:/home/ubuntu# sudo netstat -plten | grep java

INSTALLING NIFI 1.10.0

1.Download the source file from the apache website

root@ip-172-31-23-142:/home/ubuntu# wget
https://archive.apache.org/dist/nifi/1.10.0/nifi-1.10.0-bin.tar.gz
--2020-04-06 12:41:43-- https://archive.apache.org/dist/nifi/1.10.0/nifi-1.10.0-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... connected.
HTTP request sent, awaiting response... 200 OK

Length: 1372451011 (1.3G) [application/x-gzip]
Saving to: 'nifi-1.10.0-bin.tar.gz'

nifi-1.10.0-bin.tar.gz
100%[==
==============>] 1.28G 14.5MB/s   in 98s

2. Extract the file :
root@ip-172-31-23-142:/home/ubuntu# tar -xvf nifi-1.10.0-bin.tar.gz

3. Configuration

NiFi provides several different configuration options which can be configured on nifi.properties file.
NOTE: Here in nifi.properties to don't get port conflicts i'm changing the port no :8080 to 9999 and host is localhost.
root@ip-172-31-23-142:/home/ubuntu/nifi-1.10.0/conf/# vi nifi.properties

Step 3: Starting Apache Nifi:
On the terminal window,navigate to the Nifi directory and run the following below commands:

Lauches the applicaion run in the foreground and exit by pressing Ctrl-c.
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh run
Launches the application run the background.
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh start
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh status - To check the application status
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh stop - To shutdown the application

Apache Nifi Web User Interface:

After Apache Nifi Started, Web User Interface (UI) to create and monitor our dataflow.
To use Apache Nifi, open a web browser and navigate to http://localhost:8080/nifi
Note: default port is 8080 in our environment we navigate to the http://localhost:9999/nifi

INSTALLING SCALA 2.11.6

root@ip-172-31-23-142:/home/ubuntu# sudo apt install scala
root@ip-172-31-23-142:/home/ubuntu# scala -version
Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL

INSTALLING SPARK 2.4.5

root@ip-172-31-23-142:/home/ubuntu# wget https://downloads.apache.org/spark/spark-2.4.5/spark-2.4.5-bin-hadoop2.7.tgz
root@ip-172-31-23-142:/home/ubuntu# tar -xvf spark-2.4.5-bin-hadoop2.7.tgz
spark-2.4.5-bin-hadoop2.7/
spark-2.4.5-bin-hadoop2.7/licenses/

Invoke Spark shell :

root@ip-172-31-23-142:/home/ubuntu/spark-2.4.5-bin-hadoop2.7# bin/spark-shell

20/04/06 15:12:55 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).
Spark context Web UI available at http://ip-172-31-23-142.us-east-2.compute.internal:4040
Spark context available as 'sc' (master = local[*], app id = local-1586185981625).
Spark session available as 'spark'.
Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/
```

Using Scala version 2.11.12 (OpenJDK 64-Bit Server VM, Java 1.8.0_242)

Type in expressions to have them evaluated.
Type :help for more information.
scala>

root@ip-172-31-23-142:/home/ubuntu/spark-2.4.5-bin-hadoop2.7# bin/pyspark
Python 2.7.12 (default, Oct 8 2019, 14:14:10)
20/04/06 15:13:30 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.5
      /_/
```

Using Python version 2.7.12 (default, Oct 8 2019 14:14:10)
SparkSession available as 'spark'.

INSTALLING ZOOKEEPER

root@ip-172-31-23-142:/home/ubuntu# sudo apt-get install zookeeperd

INSTALLING KAFKA 2.4.0

Download Kafka

To install Kafka on your machine, click on the below link –

wget https://downloads.apache.org/kafka/2.4.0/kafka_2.11-2.4.0.tgz
Extract the tar file
root@ip-172-31-23-142:/home/ubuntu# tar -xvf kafka_2.11-2.4.0.tgz

Start Server

You can start the server by giving the following command –

root@ip-172-31-23-142:/home/ubuntu/kafka/ bin/kafka-server-start.sh config/server.properties

To run in the background
$KAFKA_HOME/bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
$KAFKA_HOME/bin/kafka-server-start.sh -daemon config/server.properties

Note before going to start the kafka server the Please specify the kafka_home path in the bashrc file.

INSTALLING HIVE 2.1.0

root@ip-172-31-23-142:/home/ubuntu# wget https://archive.apache.org/dist/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz
root@ip-172-31-23-142:/home/ubuntu# tar -xvf apache-hive-2.1.0-bin.tar.gz
root@ip-172-31-23-142:/home/ubuntu# sudo vi ~/.bashrc

Add the following lines at the end of the file
HIVE Paths
export HIVE_HOME=/home/ubuntu/apache-hive-2.1.0-bin
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/home/ubuntu/apache-hive-2.1.0-bin/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/home/ubuntu/apache-hive-2.1.0-bin/lib/*:.

The following command is used to execute ~/.bashrc file.
root@ip-172-31-23-142:/home/ubuntu# source ~/.bashrc

Configure Hive to work with Hadoop
Move to conf directory under $HIVE_HOME and execute the following commands
root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf # cp hive-env.sh.template hive-env.sh

Edit hive-env.sh and add the below line
export HADOOP_HOME=/home/ubuntu/hadoop/

Hive installation is now complete. We will need an external database server to configure Metastore. We will use Apache Derby for this.

Download and untar Apache Derby
root@ip-172-31-23-142:/home/ubuntu# wget http://archive.apache.org/dist/db/derby/db-derby-10.13.1.1/db-derby-10.13.1.1-bin.tar.gz

root@ip-172-31-23-142:/home/ubuntu# tar xvzf db-derby-10.13.1.1-bin.tar.gz

Update bashrc
root@ip-172-31-23-142:/home/ubuntu# vi ~/.bashrc

Add the following lines at the end of the file

#Derby Paths

export DERBY_HOME=/usr/local/derby

export PATH=$PATH:$DERBY_HOME/bin

Export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar

root@ip-172-31-23-142:/home/ubuntu# source .bashrc

Create a directory to store metastore
root@ip-172-31-23-142:/home/ubuntu# sudo mkdir $DERBY_HOME/data
Configure Metastore of Hive

Move to conf directory of Hive

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf/cp hive-default.xml.template hive-site.xml
Edit hive-site.xml

Add the below lines

```
<property>

<name>javax.jdo.option.ConnectionURL</name>
```

```
<value>jdbc:derby://localhost:1527/metastore_db;create=true </value>

<description>JDBC connect string for a JDBC metastore </description>

</property>
```

Run hive

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/hive

Troubleshooting for Hive installation

Error 1

Sometimes, you might get the below error when you run hive.

Logging initialized using configuration in jar:file:/home/hadoopuser1/Projects/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true

Exception in thread “main” java.lang.RuntimeException: org.apache.hadoop.hive.ql.metadata.HiveException: org.apache.hadoop.hive.ql.metadata.HiveException: MetaException(message:Hive metastore database is not initialized. Please use schematool (e.g. ./schematool -initSchema -dbType …) to create the schema. If needed, don’t forget to include the option to auto-create the underlying database in your JDBC connection string (e.g. ?createDatabaseIfNotExist=true for mysql))

If you encounter this error, the execute the below command

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/schematool -initSchema -dbType derby

If all goes well, you can now execute hive

If not, there is a chance that you might see this error

Error 2

```
Starting metastore schema initialization to 2.1.0

Initialization script hive-schema-2.1.0.derby.sql

Error: FUNCTION 'NUCLEUS_ASCII' already exists. (state=X0Y68,code=30000)

org.apache.hadoop.hive.metastore.HiveMetaException: Schema initialization FAILED!
Metastore state would be inconsistent !!

Underlying cause: java.io.IOException : Schema script failed, errorcode 2

Use –verbose for detailed stacktrace.

*** schemaTool failed ***
```

Running hive, even though it fails, creates a metastore_db directory in the directory from which you ran hive.

You will need to delete this directory

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/ mv metastore_db metastore_db.tmp

Once it's deleted, again run the below command

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/bin/schematool -initSchema -dbType derby
You should see the below output

```
SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/home/hadoopuser1/Projects/hive/lib/log4j-slf4j-impl-2.4.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/home/hadoopuser1/Projects/hadoop/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.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.apache.logging.slf4j.Log4jLoggerFactory]

Metastore connection URL: jdbc:derby:;databaseName=metastore_db;create=true

Metastore Connection Driver : org.apache.derby.jdbc.EmbeddedDriver

Metastore connection User: APP

Starting metastore schema initialization to 2.1.0

Initialization script hive-schema-2.1.0.derby.sql

Initialization script completed

schemaTool completed

root@ip-172-31-23-142:/home/ubuntu/hive/conf$
```

While trying to run hive, you might encounter the below error:

Error 3

```
Logging initialized using configuration in jar:file:/home/hadoopuser1/Projects/hive/lib/hive-common-2.1.0.jar!/hive-log4j2.properties Async: true
```

Exception in thread “main” java.lang.IllegalArgumentException:

java.net.URISyntaxException: Relative path in absolute URI:

${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D

If you do, open hive-site.xml

root@ip-172-31-23-142:/home/ubuntu/apache-hive-2.1.0/conf/ sudo vi hive-site.xml
Edit the following lines

```
<property>

<name>hive.exec.scratchdir</name>

<value>/tmp/hive</value>

<description>HDFS root scratch dir for Hive jobs which gets created with write all (733)

permission. For each connecting user, an HDFS scratch dir:

${hive.exec.scratchdir}/&lt;username&gt; is created, with

${hive.scratch.dir.permission}.</description>

</property>

<property>

<name>hive.exec.local.scratchdir</name>

<value>/tmp/hadoopuser1</value>

<description>Local scratch space for Hive jobs</description>

</property>

<property>

<name>hive.downloaded.resources.dir</name>

<value>/tmp/hadoopuser1_resources</value>
```

```
<description>Temporary local directory for added resources in the remote file system.</description>

</property>

<property>

<name>hive.scratch.dir.permission</name>

<value>733</value>

<description>The permission for the user specific scratch directories that get created.</description>

</property>
```

Create a directory for the hive warehouse into hdfs. This directory will be used by Hive to store all the data into HDFS-

cmd: hadoop dfs -mkdir -p /user/hive/warehouse

Configuring Metastore of Hive

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE_HOME/conf directory. First of all, copy the template file using the following command:

$ cd $HIVE_HOME/conf

$ cp hive-default.xml.template hive-site.xml

Edit hive-site.xml and append the following lines between the <configuration> and </configuration> tags:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby://localhost:1527/metastore_db;create=true </value>
  <description>JDBC connect string for a JDBC metastore </description>
</property>
```

Create a file named jpox.properties and add the following lines into it:

javax.jdo.PersistenceManagerFactoryClass =

org.jpox.PersistenceManagerFactoryImpl

org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

Verifying Hive Installation

Before running Hive, you need to create the /tmp folder and a separate Hive folder in HDFS. Here, we use the /user/hive/warehouse folder. You need to set write permission for these newly created folders as shown below:

chmod g+w

Now set them in HDFS before verifying Hive. Use the following commands:

$HADOOP_HOME/bin/hadoop fs -mkdir /tmp

$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp

$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

The following commands are used to verify Hive installation:

$ cd $HIVE_HOME

$ bin/hive

On successful installation of Hive, you get to see the following response:

Logging initialized using configuration in jar:file:/home/hadoop/hive-0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties

Hive history file=/tmp/hadoop/hive_job_log_hadoop_201312121621_1494929084.txt

…….

hive>

The following sample command is executed to display all the tables:

hive> show tables;

OK

Time taken: 2.798 seconds

hive>

INSTALLING AIRFLOW

Install and update PIP

root@ip-172-31-23-142:/home/ubuntu# sudo apt-get install software-properties-common

root@ip-172-31-23-142:/home/ubuntu#sudo apt-add-repository universe
root@ip-172-31-23-142:/home/ubuntu#sudo apt-get update
root@ip-172-31-23-142:/home/ubuntu#sudo apt-get install python3
root@ip-172-31-23-142:/home/ubuntu#sudo apt-get install python3-pip

Installing PostgreSQL for Airflow

root@ip-172-31-23-142:/home/ubuntu# sudo apt-get install postgresql postgresql-contrib

As we have already installed the Postgresql database using the above-mentioned command. We
will now create a database for airflow and grant access to a sudo user. Let's access to psql, a
command-line tool for Postgres.
sudo -u postgres psql
After logging in successfully, we will get a psql prompt (postgres=#). We will create a new user
and provide privileges to it.

postgres=# CREATE ROLE ubuntu;
CREATE ROLE
postgres=# CREATE DATABASE airflow;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES on database airflow to ubuntu;
GRANT
postgres=# ALTER ROLE ubuntu SUPERUSER;
ALTER ROLE
postgres=#
postgres=# ALTER ROLE ubuntu CREATEDB;

ALTER ROLE
postgres=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public to ubuntu;
GRANT
postgres=# ALTER ROLE ubuntu WITH LOGIN;
ALTER ROLE
postgres=# \c airflow
You are now connected to database "airflow" as user "postgres".
airflow=#
airflow=# \conninfo
You are connected to database "airflow" as user "postgres" via socket in "/var/run/postgresql" at port "5432".
We'll change settings in pg_hb.conf file for required configuration as per Airflow. You can run
command SHOW hba_file to find the location of pg_hba.conf file.Most likely located at pg_hb.conf located at /etc/postgresql/10/main/pg_hba.conf
open this file with vim and change ipv4 address to 0.0.0.0/0 and listen_addresses to listen_addresses = '*'.
We will restart PostgreSQL to load changes.
sudo service postgresql restart

root@ip-172-31-23-142:/etc/postgresql/9.5/main# vi pg_hba.conf

root@ip-172-31-23-142:/etc/postgresql/9.5/main# sudo service postgresql restart

Install Airflow

PostgreSQL is already installed and configured. Next, We will install Airflow and configure it.
Set the AIRFLOW_HOME environment variable to ~/airflow.
export AIRFLOW_HOME=~/airflow

Install Ubuntu dependencies required for Apache Airflow:

root@ip-172-31-23-142:/etc/postgresql/9.5/main#**sudo apt-get install** root@ip-172-31-23-142:/etc/postgresql/9.5/main#**libmysqlclient-dev**
root@ip-172-31-23-142:/etc/postgresql/9.5/main# sudo apt-get install libkrb5-dev
root@ip-172-31-23-142:/etc/postgresql/9.5/main# sudo apt-get install libsasl2-dev
root@ip-172-31-23-142:/home/ubuntu# export LC_ALL=C
root@ip-172-31-23-142:/home/ubuntu# sudo pip install apache-airflow
root@ip-172-31-23-142:/home/ubuntu# sudo pip install apache-airflow[postgres,rabbitmq,hdfs,hive,celery,crypto]

After successfully installing airflow, we will initialize Airflow's database

root@ip-172-31-23-142:/home/ubuntu#airflow initdb

Now airflow.cfg file should be generated in the airflow home directory, we will tweak some
configuration here to get better airflow functionality.
We will be using CeleryExecutor instead of SequentialExecutor which comes by default with
airflow. Change executor = CeleryExecutor
For DB connection we will pass the PostgreSQL database 'airflow', that we have created in
earlier steps.
sql_alchemy_conn = postgresql+psycopg2://ubuntu@localhost:5432/airflow
For removing examples on the home page load_examples variable can set to False
Change broker_url and celery_result_backend to the same config, as shown below
broker_url = amqp://guest:guest@localhost:5672//
celery_result_backend = amqp://guest:guest@localhost:5672//
After doing all these settings just save your configuration and exit.
For Loading new configurations, we should run
root@ip-172-31-23-142:/home/ubuntu#airflow initdb

Installing Rabbitmq

Rabbitmq is a message broker that is required to rerun airflow dags with celery. Rabbitmq can be
installed with the following command.

root@ip-172-31-23-142:/home/ubuntu# sudo apt install rabbitmq-server

We will change configuration NODE_IP_ADDRESS=0.0.0.0 in configuration file located at
/etc/rabbitmq/rabbitmq-env.conf
Now Start RabbitMQ service

root@ip-172-31-23-142:/home/ubuntu# sudo service rabbitmq-server start

Installing Celery

Celery is a python API for rabbitmq, We can install celery using pip

root@ip-172-31-23-142:/home/ubuntu# sudo pip3 install celery

Requirement already satisfied (use --upgrade to upgrade): celery in /usr/local/lib/python3.5/dist-packages
Requirement already satisfied (use --upgrade to upgrade): kombu<4.7,>=4.6.8 in /usr/local/lib/python3.5/dist-packages (from celery)
Requirement already satisfied (use --upgrade to upgrade): billiard<4.0,>=3.6.3.0 in /usr/local/lib/python3.5/dist-packages (from celery)

All the required installation and configuration is done. We will create a dags folder in airflow
home directory .i.e; at /root/airflow/ location

root@ip-172-31-23-142:/home/ubuntu# mkdir -p /root/airflow/dags/

root@ip-172-31-23-142:~/airflow# airflow webserver

[2020-04-06 14:23:38,688] {settings.py:253} INFO - settings.configure_orm(): Using pool settings. pool_size=5, max_overflow=10, pool_recycle=1800, pid=14226
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Airflow 1.10 will be the last release series to support Python 2

```
  ____________       _____________
 ____    |__( )_________  __/__  /________      __
____  /| |_  /__  ___/_  /_ __  /_  __ \_ | /| / /
___  ___ |  / _  /   _  __/ _  / / /_/ /_ |/ |/ /
 _/_/  |_/_/  /_/    /_/    /_/  \____/____/|__/
```

[2020-04-06 14:23:39,189] {__init__.py:51} INFO - Using executor LocalExecutor
[2020-04-06 14:23:39,190] {dagbag.py:403} INFO - Filling up the DagBag from /root/airflow/dags
Running the Gunicorn Server with:
Workers: 4 sync
Host: 0.0.0.0:8080
Timeout: 120
Logfiles: - -
==========
==
[2020-04-06 14:23:40,064] {settings.py:253} INFO - settings.configure_orm(): Using pool settings. pool_size=5, max_overflow=10, pool_recycle=1800, pid=14237
[2020-04-06 14:23:40 +0000] [14237] [INFO] Starting gunicorn 19.10.0
[2020-04-06 14:23:40 +0000] [14237] [INFO] Listening at: http://0.0.0.0:8080 (14237)
[2020-04-06 14:23:40 +0000] [14237] [INFO] Using worker: sync
[2020-04-06 14:23:40 +0000] [14242] [INFO] Booting worker with pid: 14242
[2020-04-06 14:23:40 +0000] [14243] [INFO] Booting worker with pid: 14243
[2020-04-06 14:23:40 +0000] [14250] [INFO] Booting worker with pid: 14250

Commands to be executed to start Airflow:

1.root@ip-172-31-23-142:~/airflow# airflow webserver -D
2.root@ip-172-31-23-142:~/airflow# airflow scheduler -D
3.root@ip-172-31-23-142:~/airflow# airflow worker -D