Building big data pipeline on AWS documentation

Sr. No.	Updated By	Updated On	Comments
1.	Divya Sistla	22/04/2020	First Draft

Version Information:

Table of Contents

Overview

The purpose is to collect the real time streaming data from COVID19 open API for every 5 minutes into the ecosystem using NiFi and to process it and store it in the data lake on AWS.Data processing includes parsing the data from complex JSON format to csv format then publishing to Kafka for persistent delivery of messages into PySpark for further processing.The processed data is then fed into output Kafka topic which is inturn consumed by Nifi and stored in HDFS .A Hive external table is created on top of HDFS processed data for which the process is Orchestrated using Airflow to run for every time interval. Finally KPIs are visualised in tableau.

Project execution guidelines

- As soon as we login into the EC2 instance , we need to start the following jps services using the commands .
- Note that everytime we stop and start the instance these services need to be restarted if they are not running as Daemon processes.

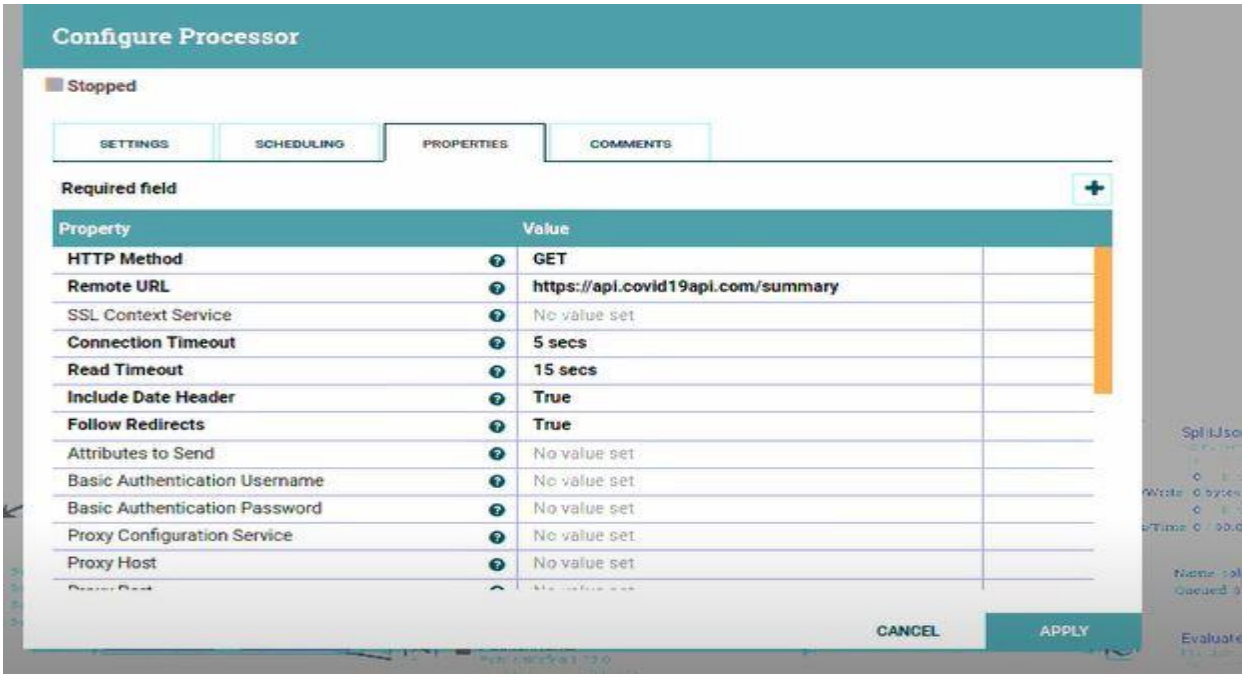
To run Hadoop:
root@ip-172-31-23-142://home/ubuntu/hadoop_2.7.1# sbin/start-all.sh
Or
root@ip-172-31-23-142:/home/ubuntu/hadoop_2.7.1# sbin /start-dfs.sh
root@ip-172-31-23-142:/home/ubuntuhadoop_2.7.1# sbin/start-yarn.sh
To run nifi :
root@ip-172-31-23-142:/home/ubuntu# bin/nifi.sh start
To run NiFi in the Browser:
After running the vncserver, open the browser Then type
http://localhost:9999/nifi
To run kafka:
root@ip-172-31-23-142:/home/ubuntu/kafka/ bin/kafka-server-start.sh config/server.properties
Before run the NiFi GUI browser we start the vnc server
To start vncserver :
root@ip-172-31-23-142:/home/ubuntu# vncserver :1
To run Airflow:
Running Airflow needs two terminals to be opened in parallel .
Need to login into the ec-2 instance into both the terminals .
Then go into the root folder by specifying:
root@ip-172-31-23-142: cd ~
Then into airflow folder by:
root@ip-172-31-23-142: cd Airflow
Then run each of these commands in different terminals and keep them running
root@ip-172-31-23-142: airflow webserver -p 8080
root@ip-172-31-23-142:airflow scheduler

Once all the processes are started , we can check the command jps in another terminal

The next step is to open the VNC server and Firefox browser in it, and give localhost:9999/nifi to open the NiFi UI

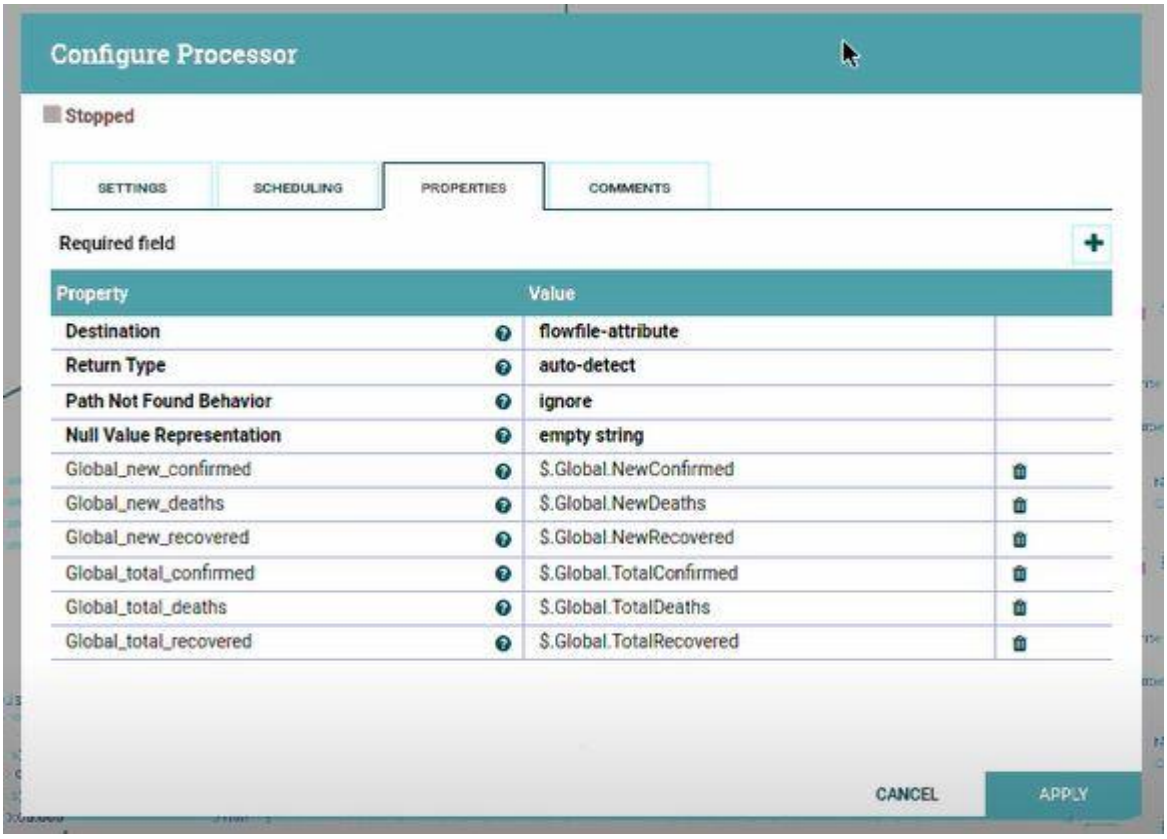
Here, as a first step , we extract the data from Corona Open API endpoint using

- **InvokeHttp processor -**Drag and drop a new processor and search for Invokehttp and click on configure and specify details as follows:

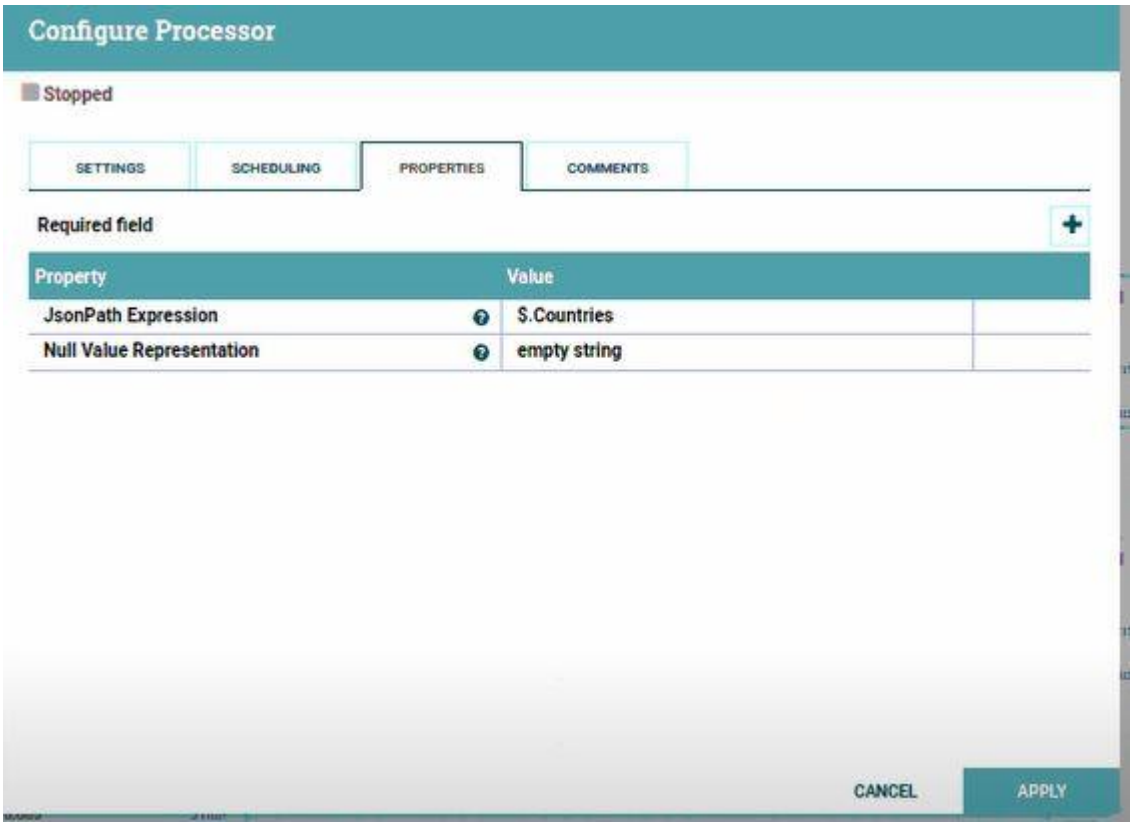


- Then run the processor and the raw data is received in nested JSON format (JSON array and JSON objects) from the COVID 19 API endpoint for every 5 minutes as scheduled in the processor.

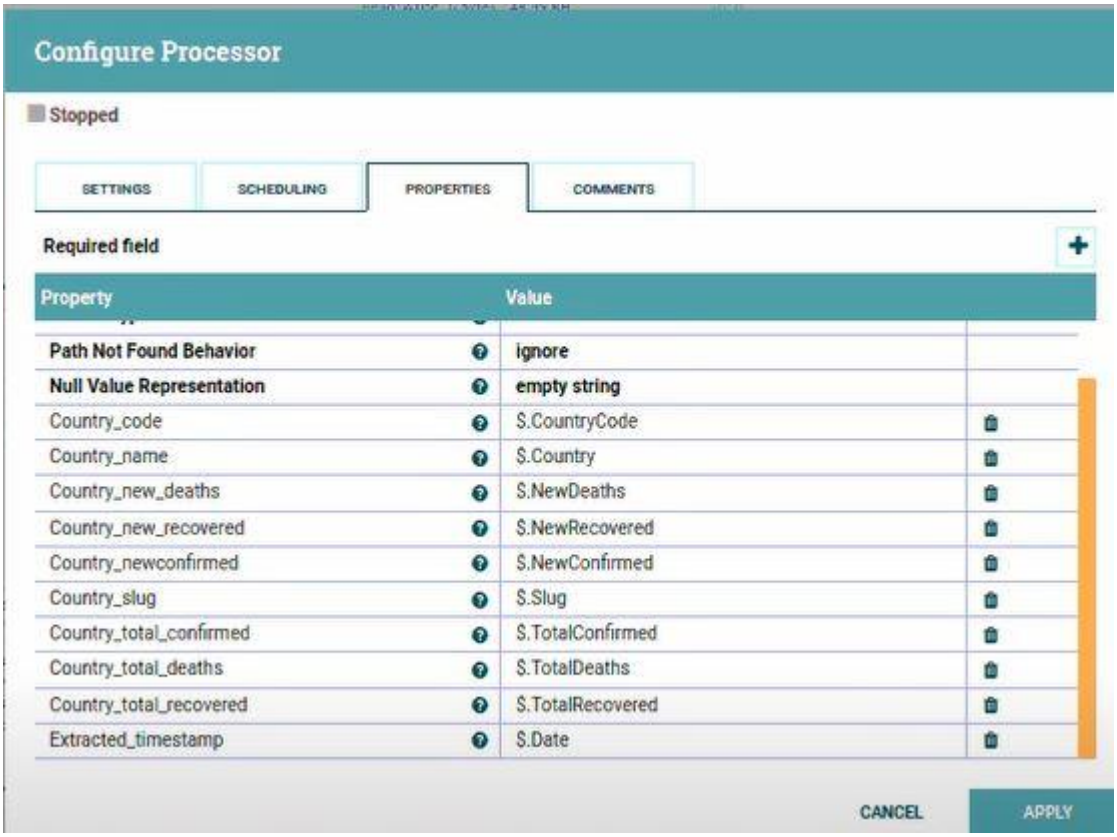
- Then, evaluate the fields which are as JSON object using EvaluateJSON processor with the following configuration



- To evaluate fields which are as JSON arrays , splitJSON processor is used


Configure Processor
Stopped
SETTINGS
SCHEDULING
PROPERTIES
COMMENTS
Required field
Property
Value
JsonPath Expression
$.Countries
Null Value Representation
empty string
CANCEL
APPLY

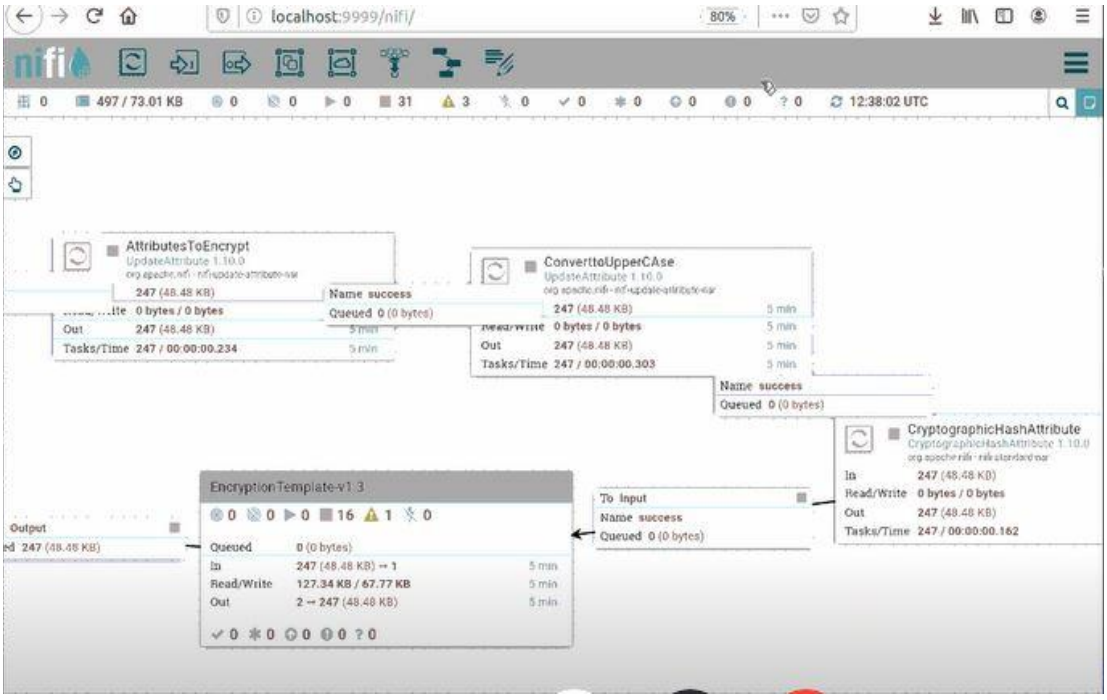
- Then we use evalauteJSON again to extract the individual fields from JSON array which was split by the array



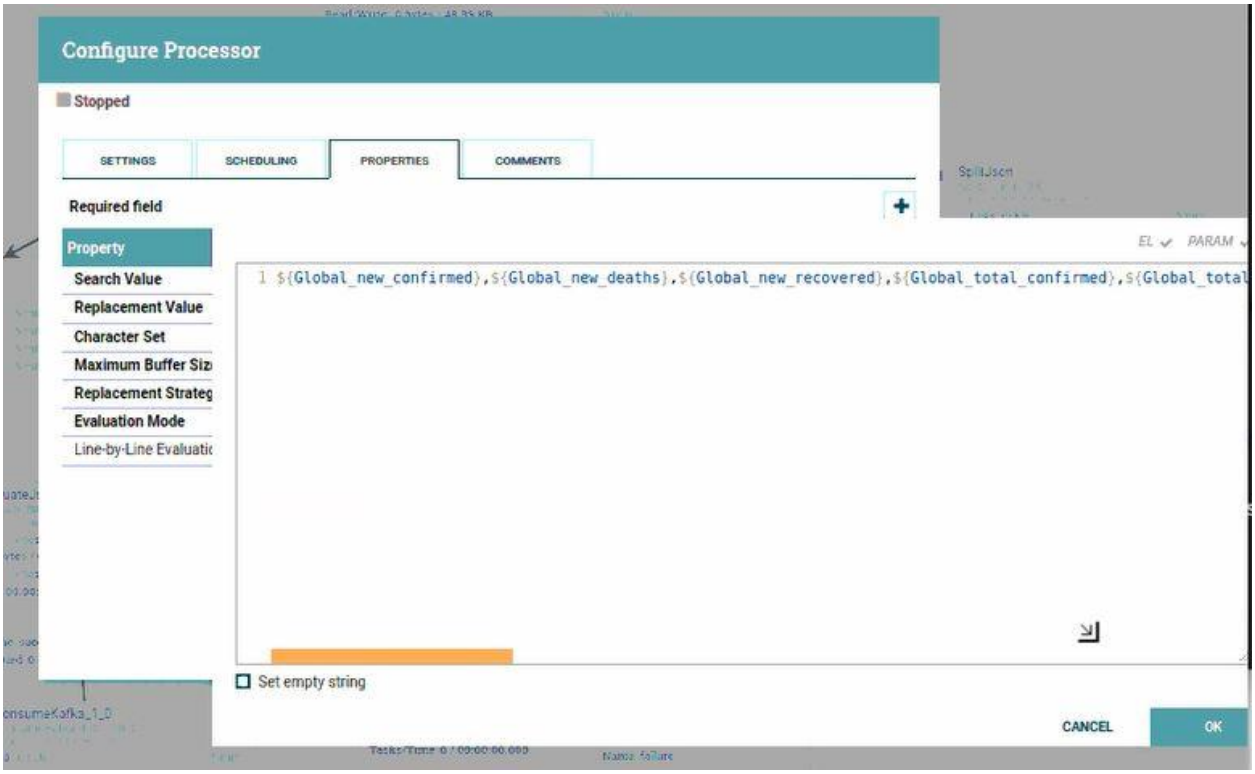
Encryption of Pii fields in the Data using Nifi

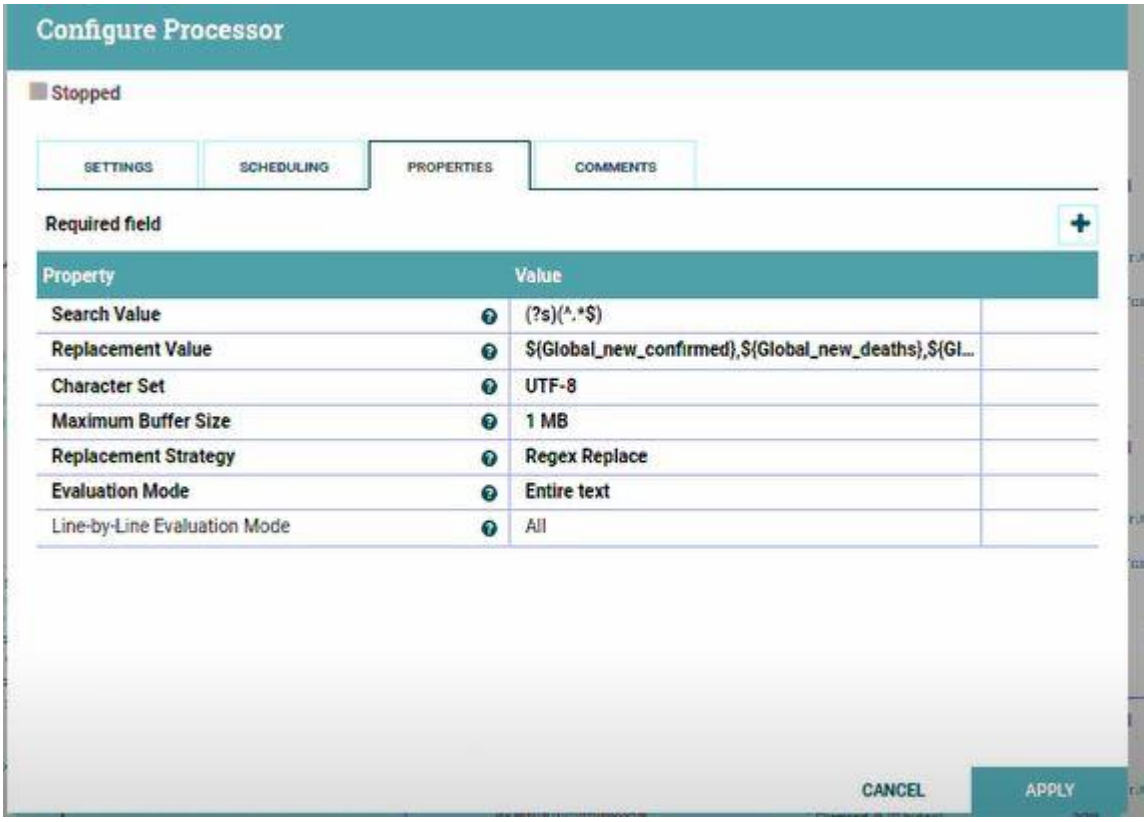
Data encryption of Pii fields is necessary for Data security purposes in large scale distribution environments.So, we've encrypted one of the fields as a sample to show how the encryption can be carried out using NiFi .
We would need three processors for the encryption purpose

- AttributestoEncrpyt
- Cryptographic Hash Attribute
- EncryptContent
- Encryption template which needs to be created using set of processors for encrypting the data which needs to be imported as XML into NiFi from settings->templates->upload template and we need to upload the given XML here->drag and drop the template icon in NiFi and choose the latest uploaded template .

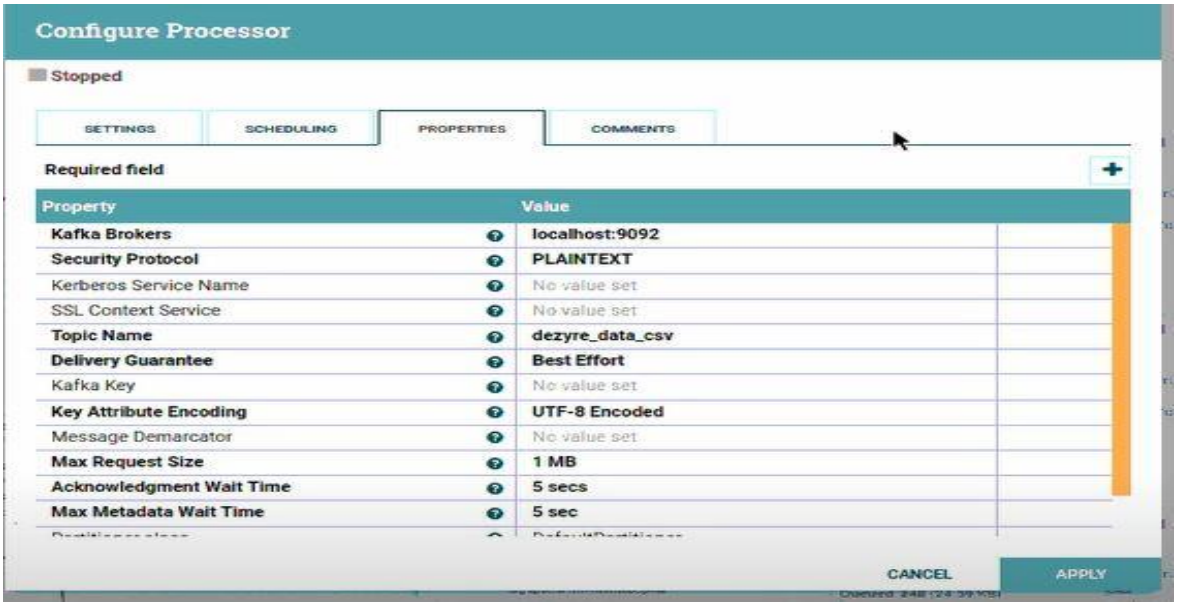


- Now, parse the values into CSV format using ReplaceText processor with commas as follows:

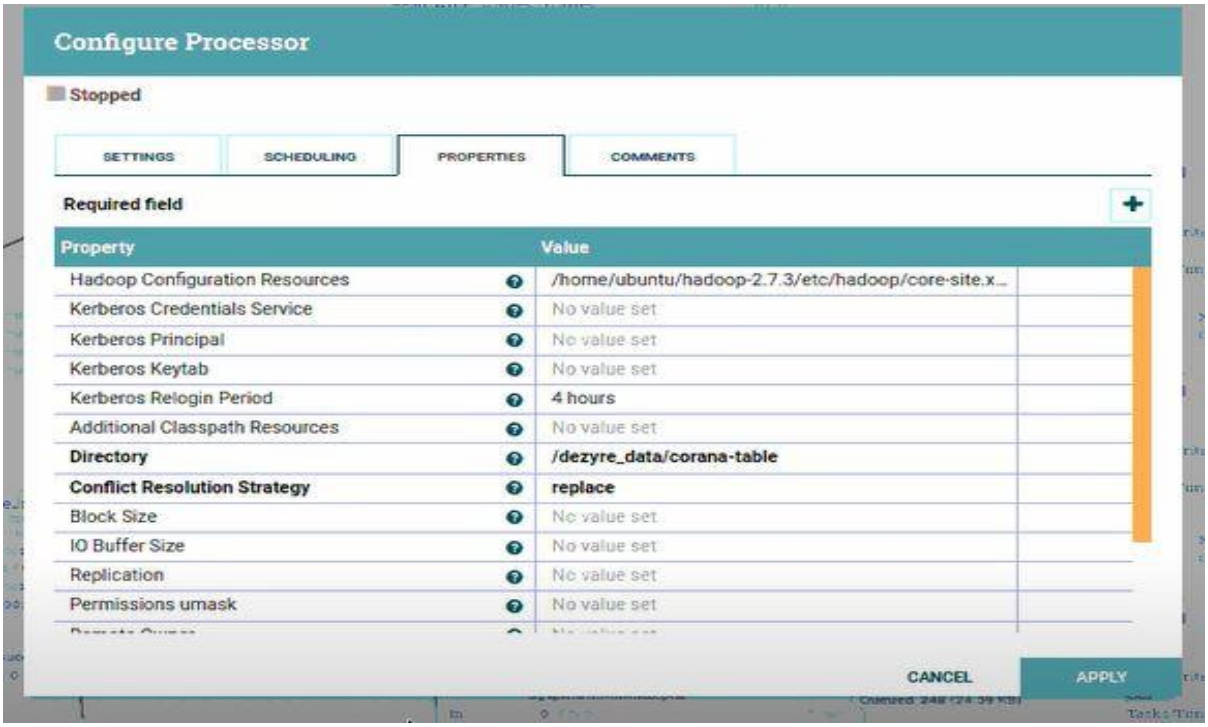




- Now the parsed data in parallel goes into HDFS as well as Kafka for ease of use . For that we use PublishKafka and PutHDFS processors and are configured as follows :
  PublishKafka



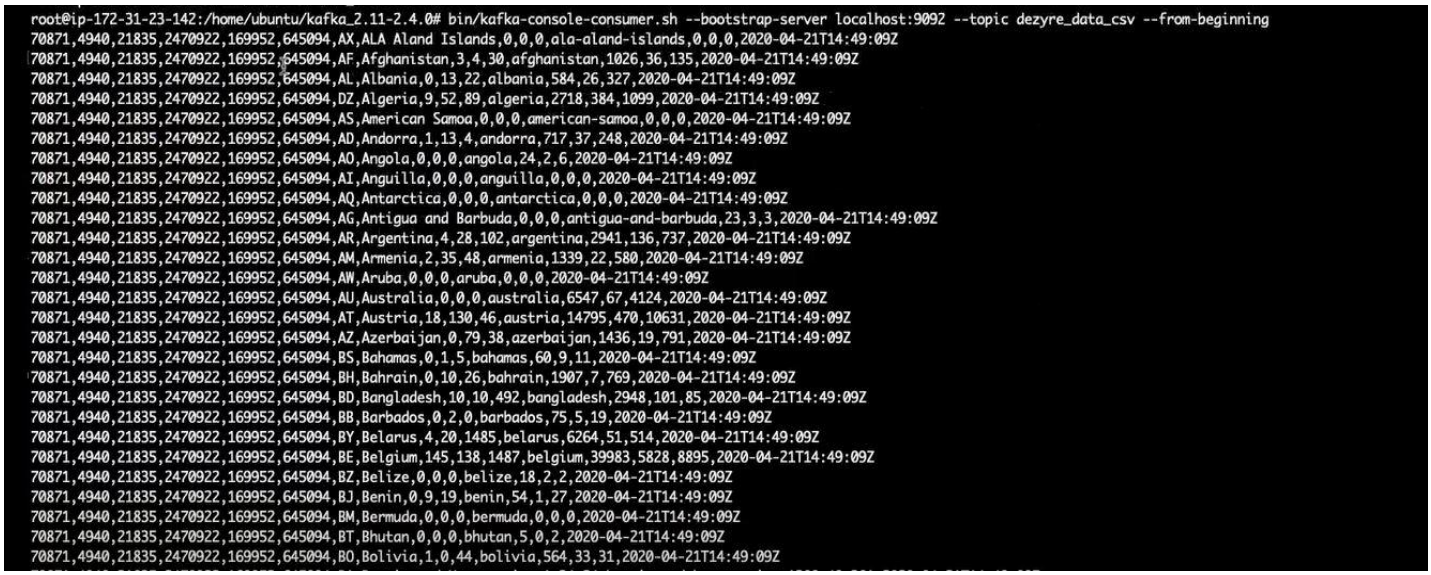
PutHDFS



- Once the data is published to Kafka topic , we can view the data in the console using the following command:
- Navigate to bin folder where Kafka was installed .Then run the following command

  bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic topic name

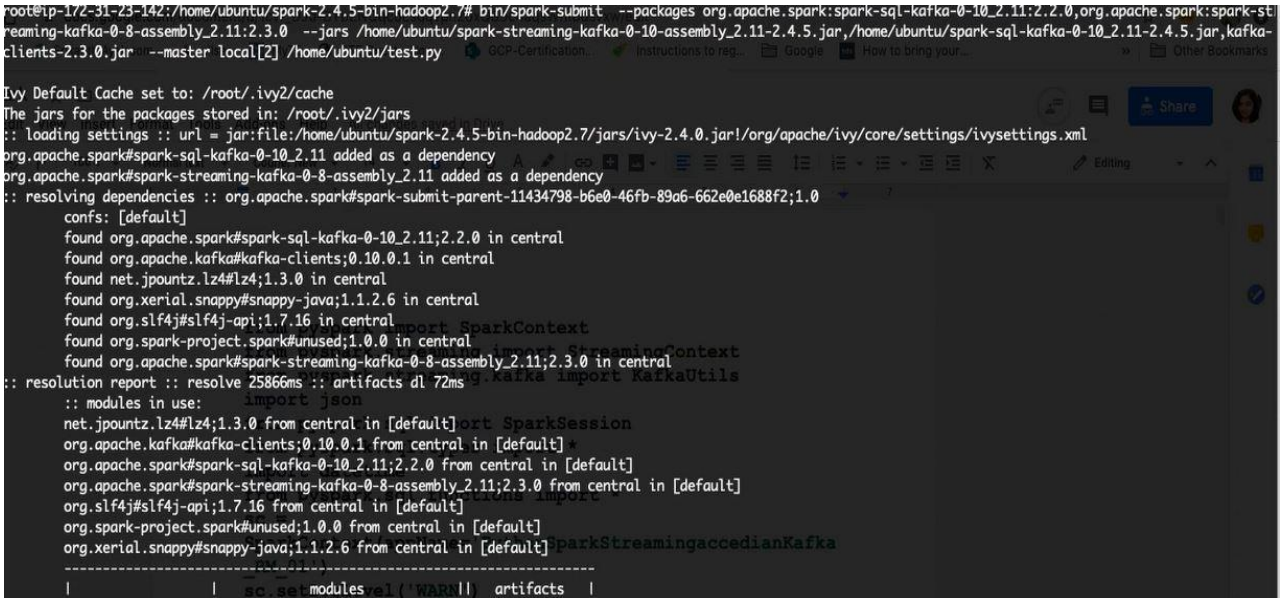
- The above command shows the data which is published into the topic using PublishKafka in NiFi



- The data which is now in Kafka topic , we extract this streaming data messages into PySpark for streaming data processing
- Python code can be found in the link below:
- https://docs.google.com/document/d/17gIkv58tIzTpw44ijJKcKUkiDDXMyVlw5v_8ESl4M54/edit?usp=sharing

- Before submitting the python code , ensure that you have downloaded the required jars in the below command and copy them into your ec2-instance from your local using scp command .
- Then run the python code in the spark bin folder using:

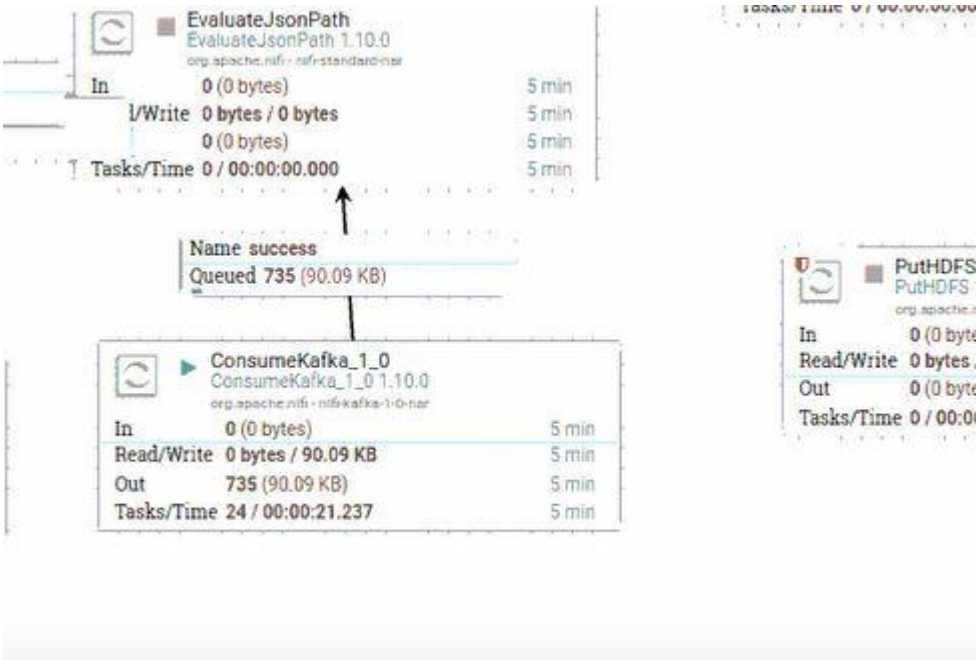
  bin/spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.2.0,org.apache.spark:spark-streaming-kafka-0-8-assembly_2.11:2.3.0 --jars /home/ubuntu/spark-streaming-kafka-0-10-assembly_2.11-2.4.5.jar,/home/ubuntu/spark-sql-kafka-0-10_2.11-2.4.5.jar,kafka-clients-2.3.0.jar --master local[2] /home/ubuntu/test.py



- The above python code is writing data to Kafka output topic in JSON format and it can be verified in console via Kafka console consumer

- bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic topic name

  - Now the data in the output Kafka topic is ready to consume by consumers like Nifi.
  - We use ConsumeKafka processor to get the data for further processing



- Once the data is consumed by Nifi , We parse the data into CSV again and store it in HDFS.
- Then create a Hive external table using the below script in hive.
- We login into Hive by entering into hive terminal by going into the folder where apache-hive is installed and give the following command
- bin/hive which enters into Hive shell

```
releases.
hive> use default;
OK
Time taken: 0.777 seconds
hive> show tables;
OK
airflow_hive
sample
Time taken: 0.163 seconds, Fetched: 2 row(s)
hive> select * from sample;
OK
Time taken: 1.134 seconds
hive> select * from airflow_hive;
OK
1       bigdata
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive>
```

https://docs.google.com/document/d/1f79QCJc6n7nJaIqfxyF18u0-SNnULRy2GGiP6Qc/edit?usp=sharing

We create external table to point to HDFS location.

- The Hive script to create table can be found at:

https://docs.google.com/document/d/1f79QCJc6n7nJaIqfxyF18u0-SNnULRy2GGiP6Q91F1c/edit?usp=sharing

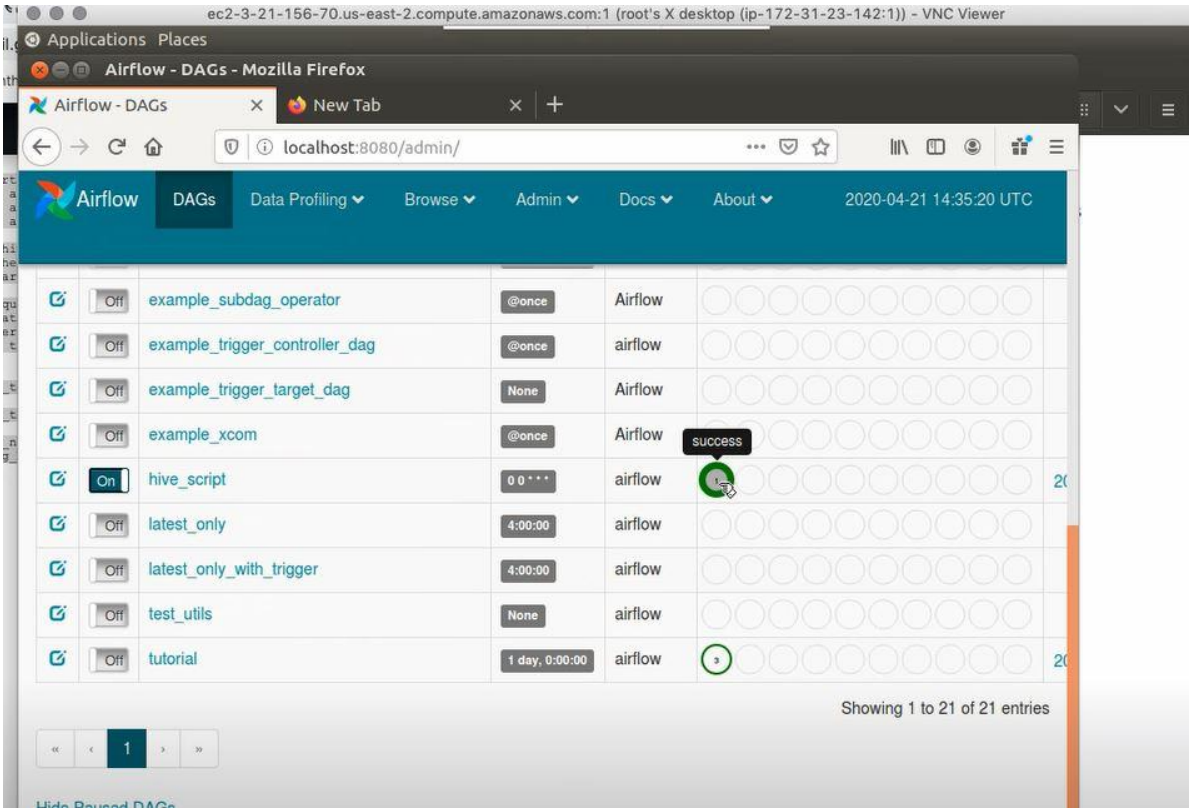
- We create an external table to point to the HDFS location. Further processed tables could also be created as a managed table.

Dataflow Orchestration using Airflow

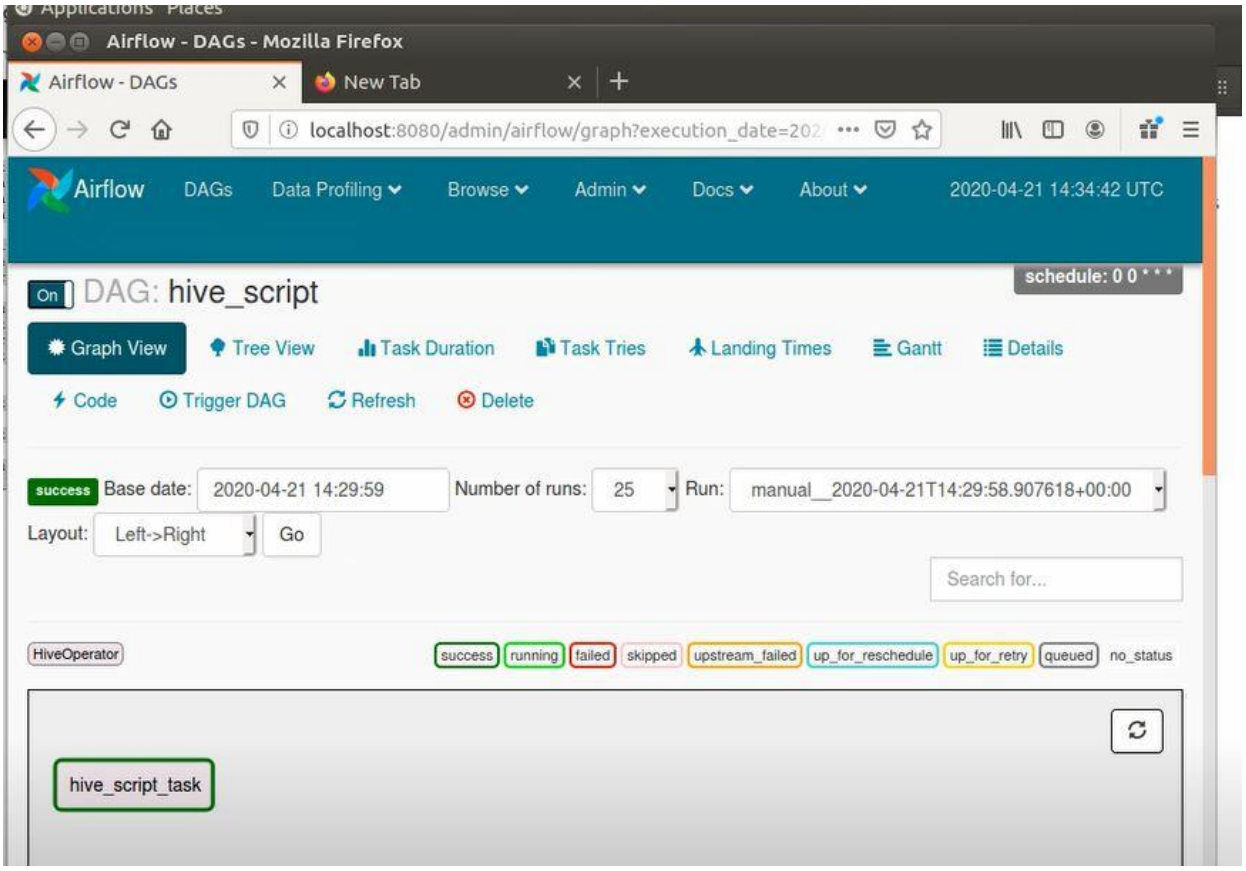
As a first step, we need to start the Airflow webserver and scheduler in separate terminals and keep them running

To run the airflow:
root@ip-172-31-23-142:~/airflow: airflow webserver -p 8080
root@ip-172-31-23-142:~/airflow: airflow scheduler
root@ip-172-31-23-142:~/airflow: airflow worker

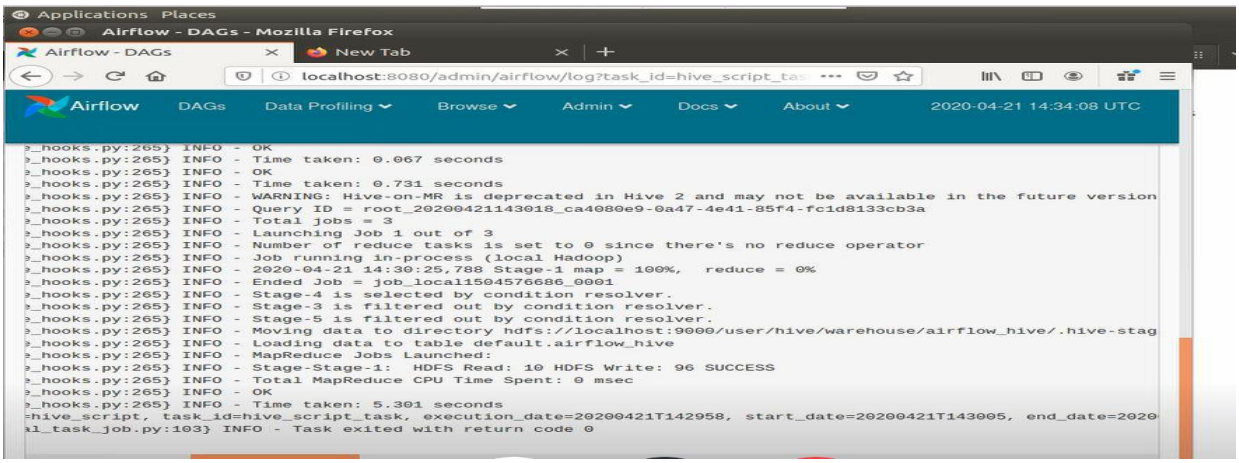
To start the Airflow GUI:
Goto the browser and then type http://localhost:8080/admin
Note: will give the python script dag file in the ~/airflow/dags folder.
Then navigate to the Airflow UI in the VNC server browser by giving localhost:8080/admin which opens the airflow UI as follows :



Here in the above screen as you our dag appears in the browser the will the file and trigger the dag option as below pic.



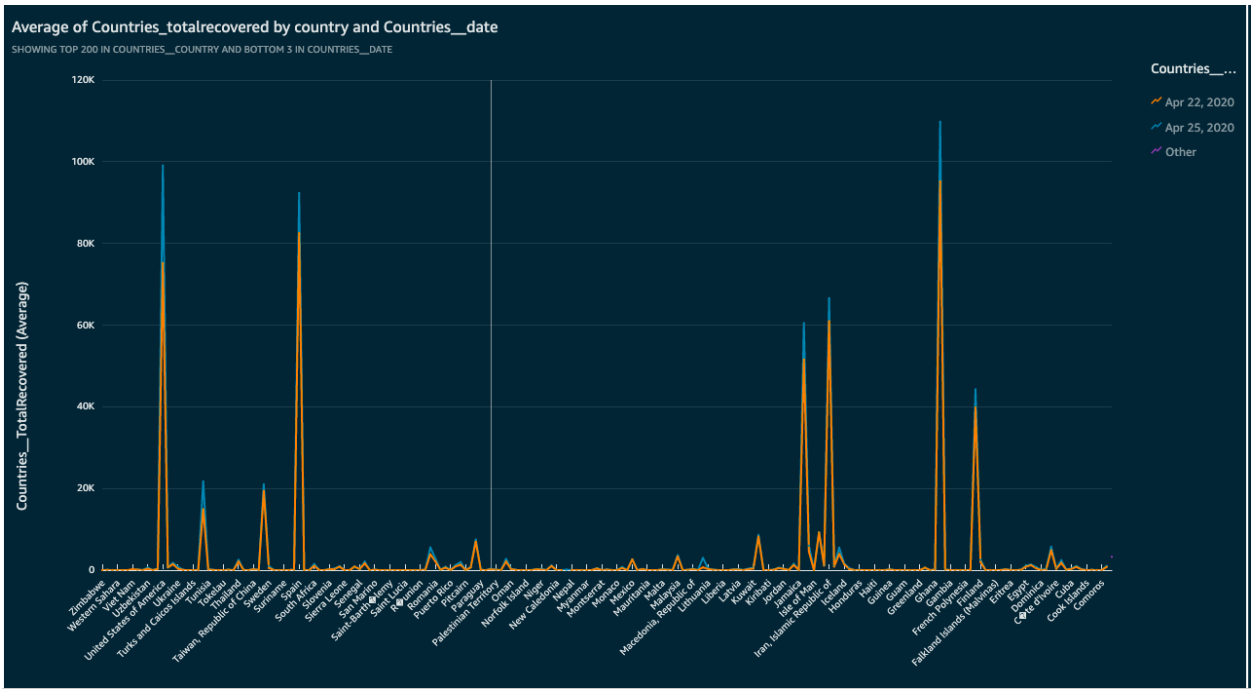
And the logs for the task can be traced by clicking on the hive_script_task and view logs option in the pop up .The logs will be shown as follows:

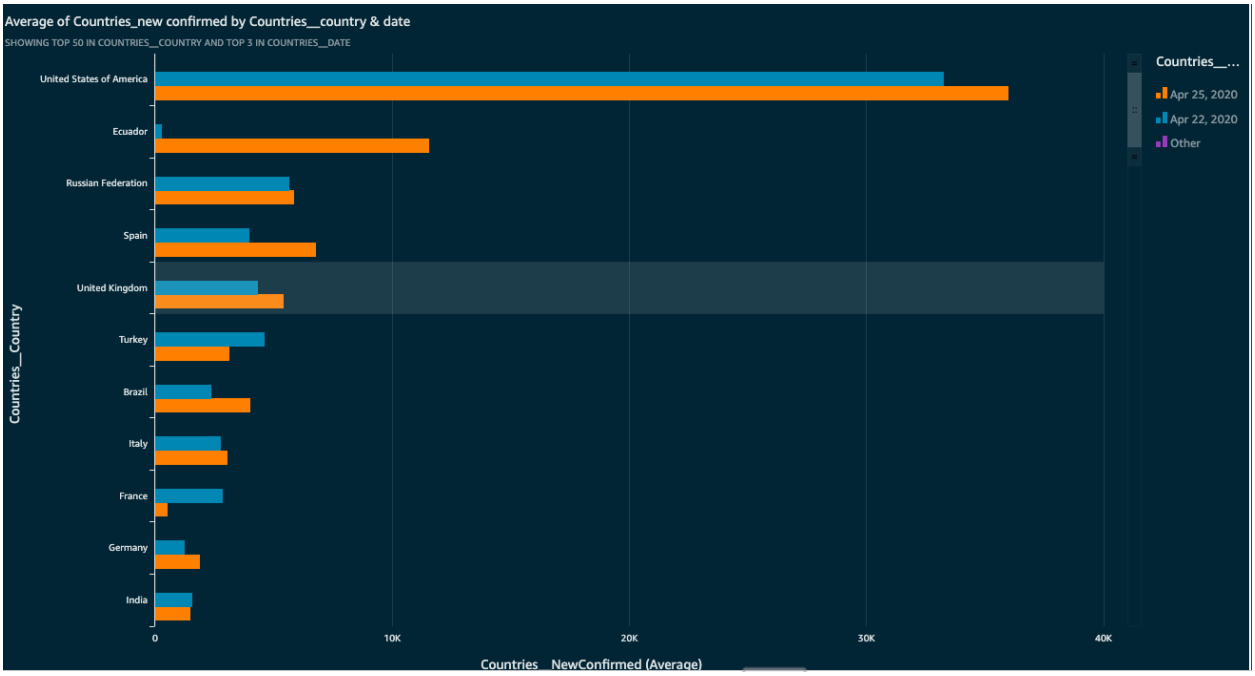


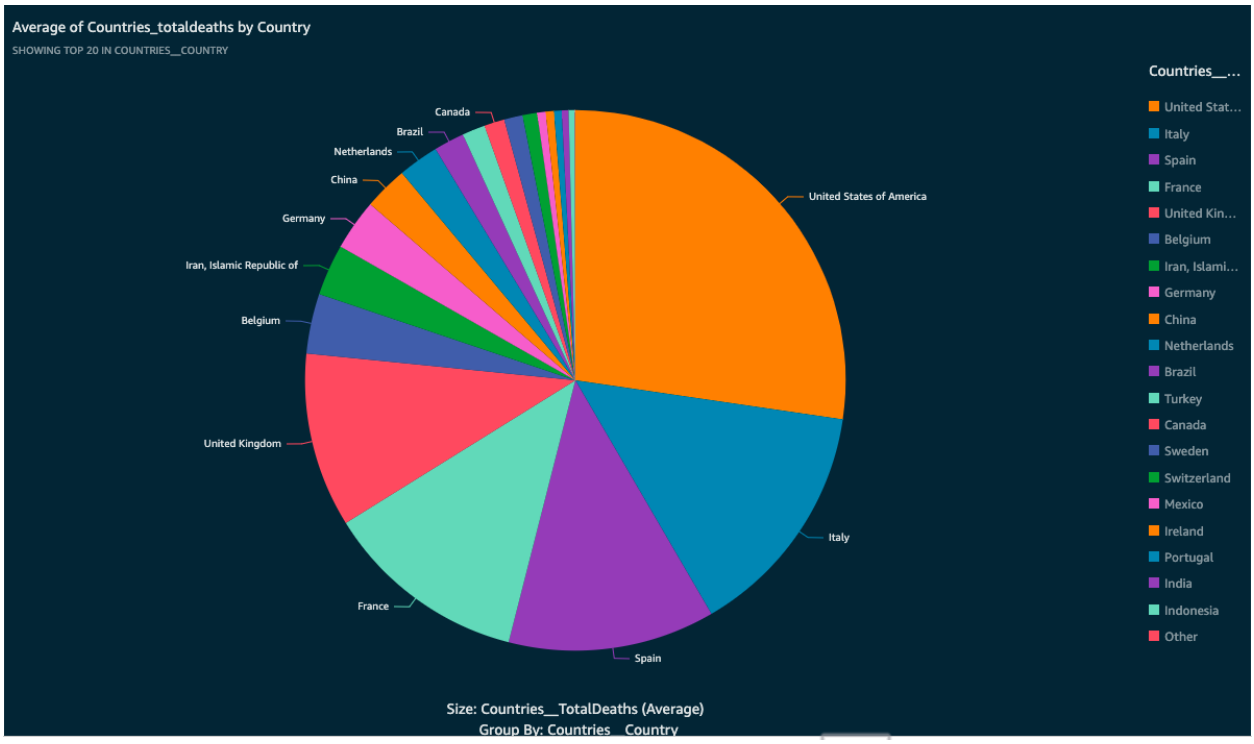
- The DAG should be deployed as a python file with the required tasks specified (Hive task, python task email task etc)
- Below is the link for the airflow python code which needs to be placed in airflow/dags folder

https://docs.google.com/document/d/1cprVhn9lOVUimvs2A3nUPFP8i8BvwrZCONbah-5LmYc/edit?usp=sharing

Visualisation of Data in AWS Quicksight






Average of Countries_totaldeaths by Country
SHOWING TOP 20 IN COUNTRIES__COUNTRY
Canada
Brazil
Netherlands
China
Germany
Iran, Islamic Republic of
Belgium
United Kingdom
France
Spain
Italy
United States of America
Countries__...
United Stat...
Italy
Spain
France
United Kin...
Belgium
Iran, Islami...
Germany
China
Netherlands
Brazil
Turkey
Canada
Sweden
Switzerland
Mexico
Ireland
Portugal
India
Indonesia
Other
Size: Countries__TotalDeaths (Average)
Group By: Countries__Country

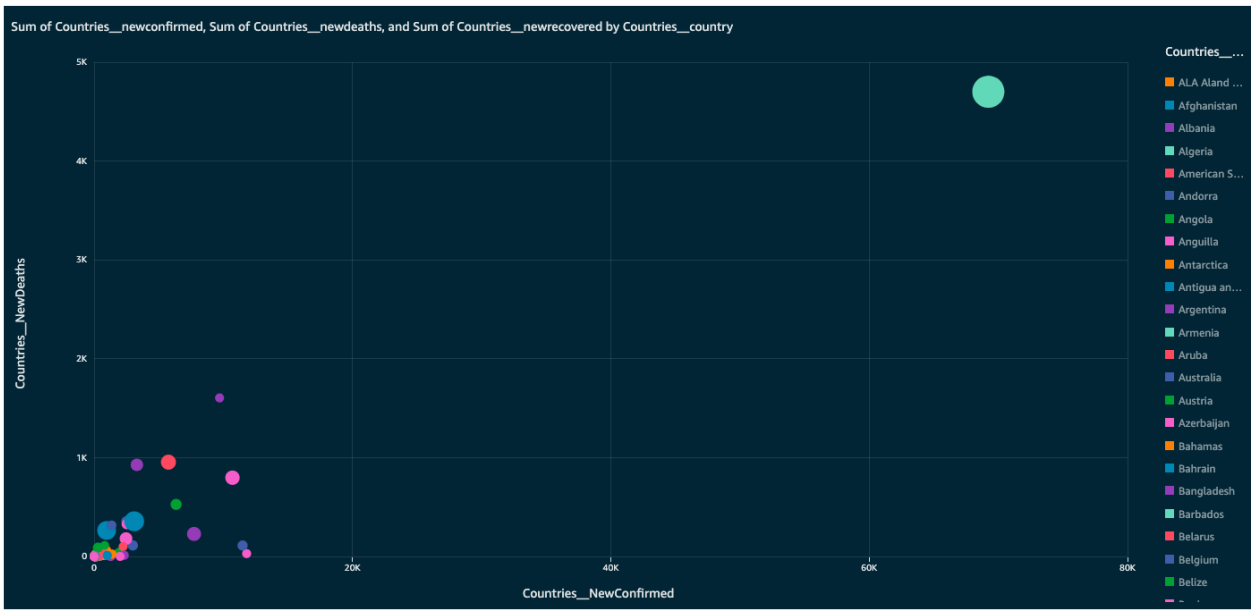
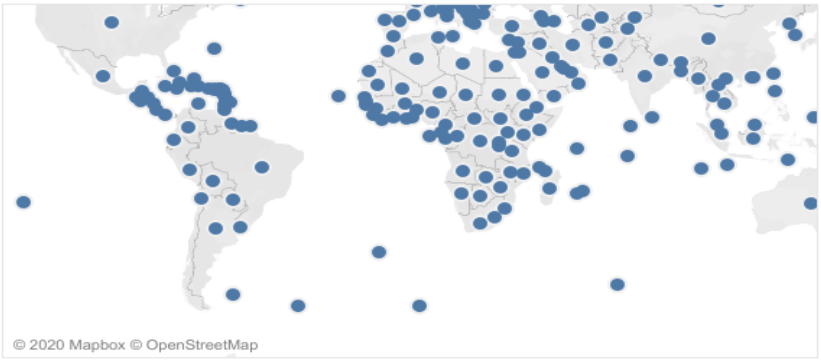
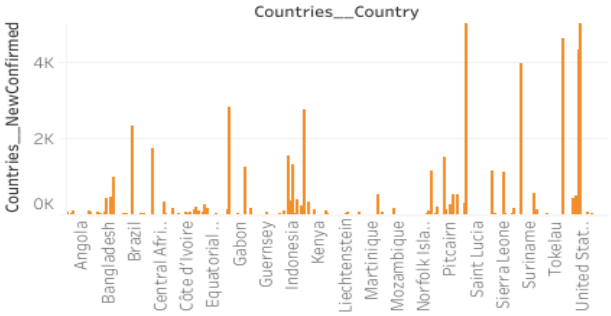

Sum of Countries__newconfirmed, Sum of Countries__newdeaths, and Sum of Countries__newrecovered by Countries__country
Countries__NewDeaths
5K
4K
3K
2K
1K
0
0
20K
40K
60K
80K
Countries__NewConfirmed
Countries__...
ALA Aland ...
Afghanistan
Albania
Algeria
American S...
Andorra
Angola
Anguilla
Antarctica
Antigua an...
Argentina
Armenia
Aruba
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
Barbados
Belarus
Belgium
Belize

Tableau visualisations(Tableau online)

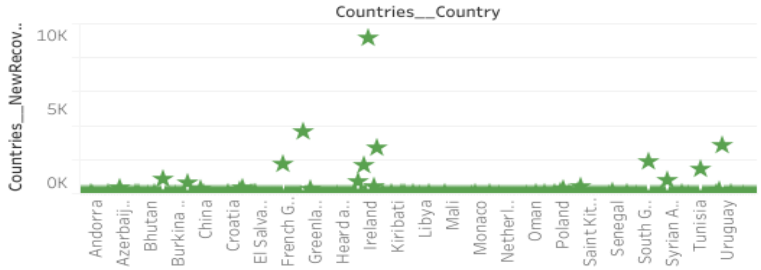
Countrywise total deaths



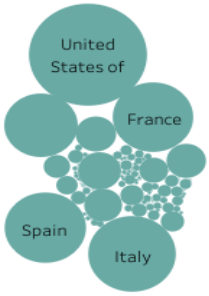
Countrywise new confirmed cases



Countrywise new deaths



Country analysis



Tools used

1. Nifi -nifi-1.10.0
2. Hadoop -hadoop_2.7.3
3. Hive-apache-hive-2.1.0
4. Spark-spark-2.4.5
5. Zookeeper-zookeeper-2.3.5
6. Kafka-kafka_2.11-2.4.0
7. Airflow-airflow-1.8.1
8. Tableau-

Known errors and resolutions

- Everytime consume kafka processor is run, the group id attribute needs to be changed and can be given some random text value
- When we see the below error while starting Hive :

  Caused by: java.net.URISyntaxException: Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D

  Then we need to add the following property at the beginning of hive-site.xml file

  ```
  <property>

   <name>system:java.io.tmpdir</name>

  <value>/user/local/hive/tmp/java</value>

  </property> <property>

  <name>system:user.name</name>

  <value>${user.name}</value>

  </property>
  ```

THE END