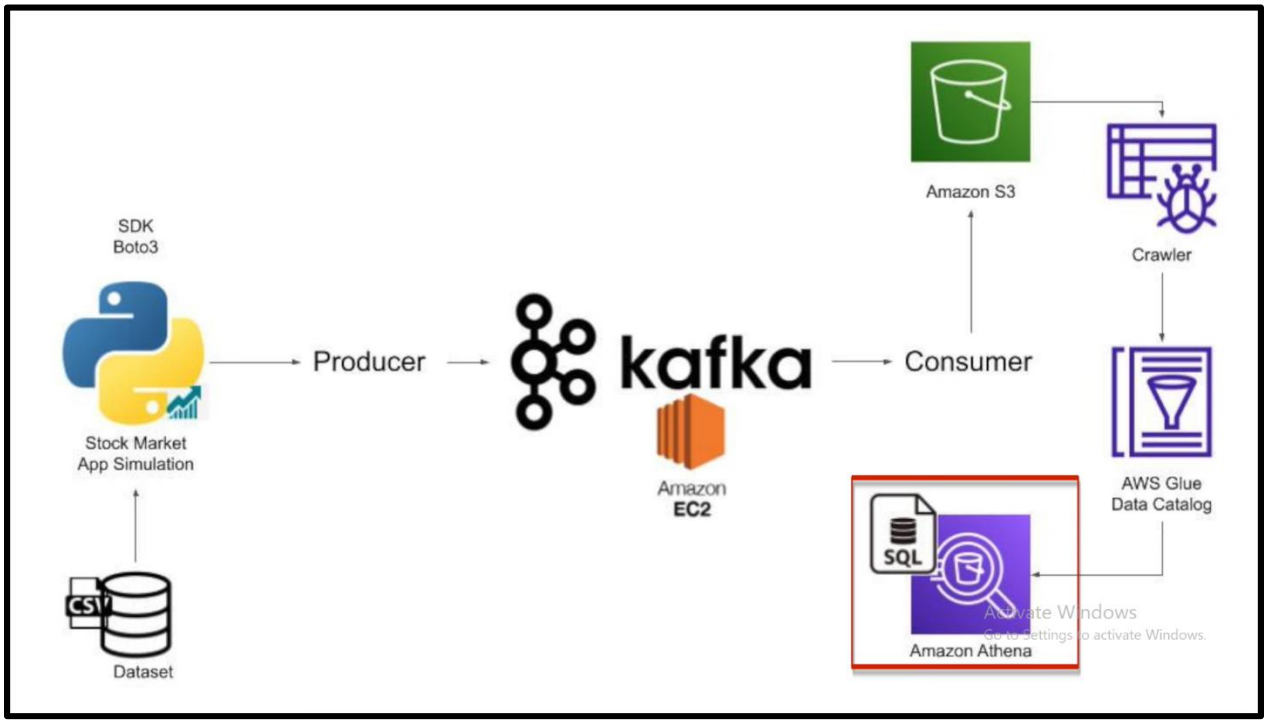
KAFKA STREAMING PROJECT



Step 1: Use python to produce stock data - - - - - - by looping and giving to producer

Step 2: Use Kafka cluster to consume the data to S3 bucket of amazon- - - - - - - giving path to save the data to consumer in s3 bucket.

Step 3: Crawl that data to build a Glue catalog

Step 4: Analyze the data using Amazon Athena

Kafka real time:

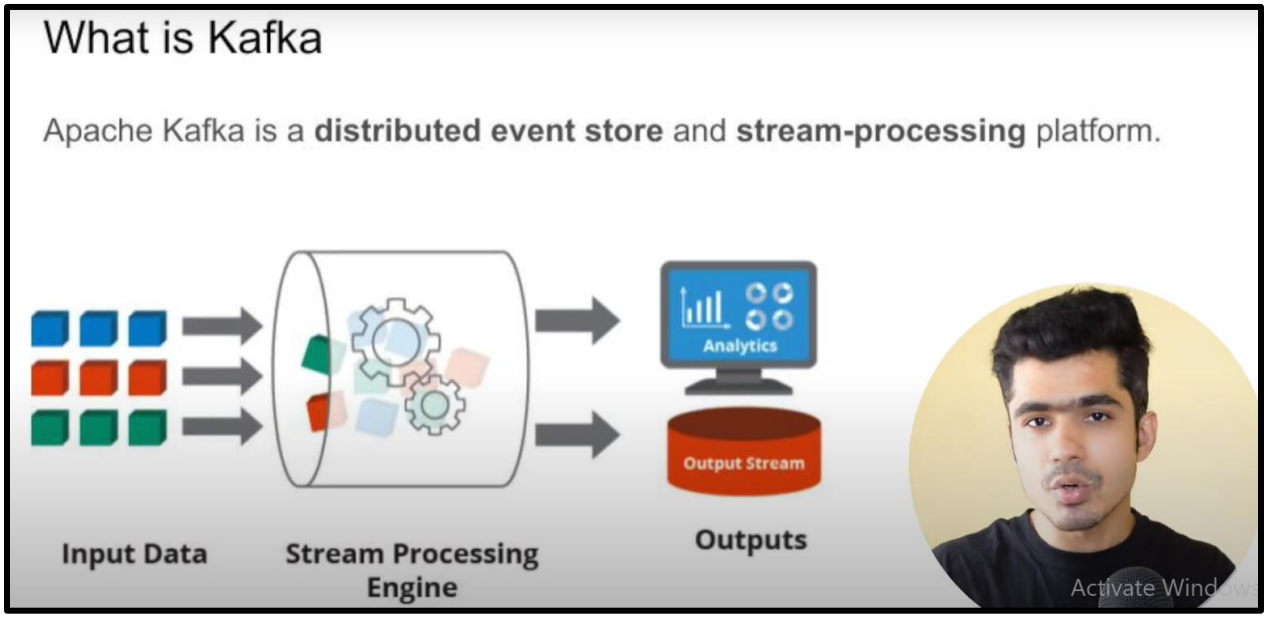
Introduction

In this project, you will execute an End-To-End Data Engineering Project on Real-Time Stock Market Data using Kafka.

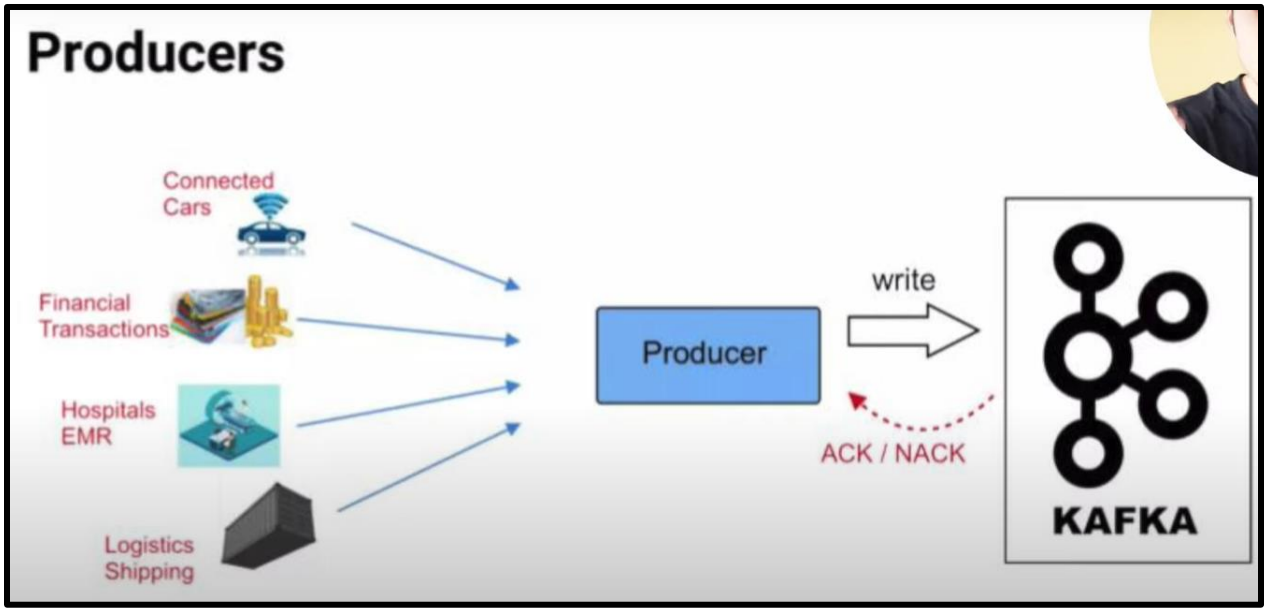
We are going to use different technologies such as Python, Amazon Web Services (AWS), Apache Kafka, Glue, Athena, and SQL.

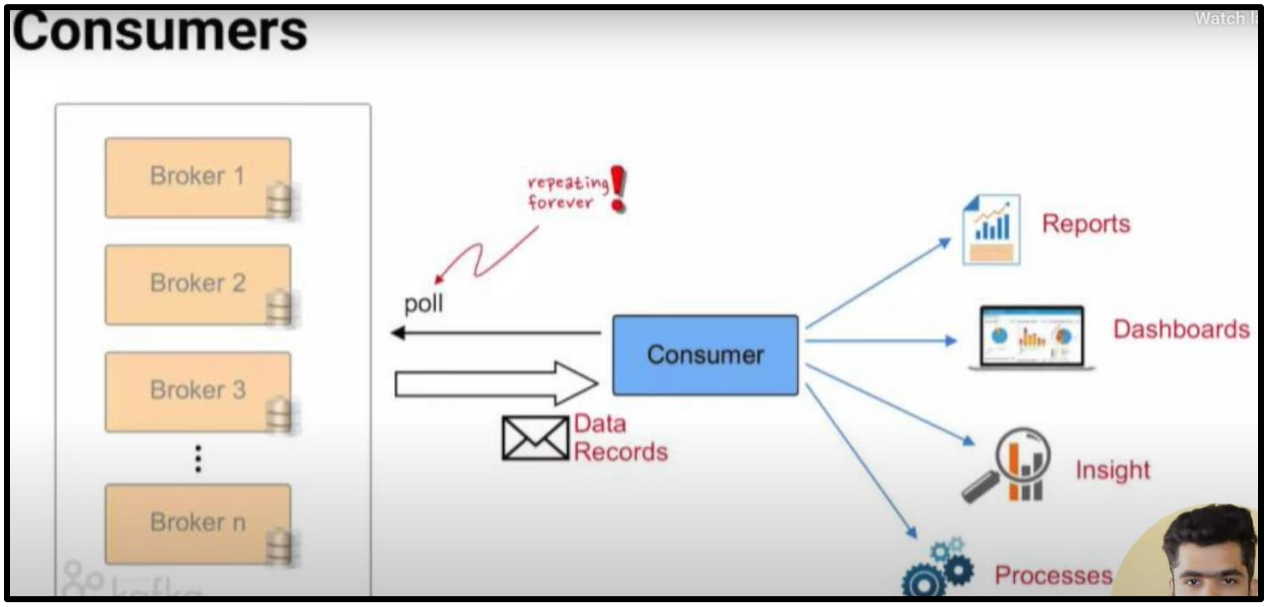
Technologies Used

- Programming Language - Python
- Amazon Web Service (AWS)
- S3 (Simple Storage Service)
- Athena
- Glue Crawler
- Glue Catalog
- EC2
- Apache Kafka



Multiple brokers inside one kafka cluster


Producers
Connected Cars
Financial Transactions
Hospitals EMR
Logistics Shipping
Producer
write
ACK / NACK
KAFKA

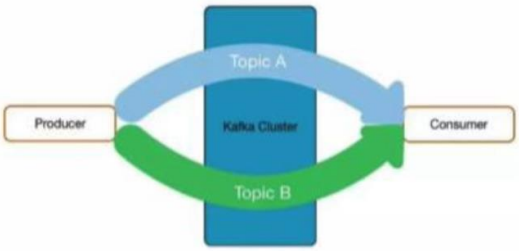

Consumers
Broker 1
Broker 2
Broker 3
Broker n
repeating forever
poll
Consumer
Data Records
Reports
Dashboards
Insight
Processes

ZooKeeper Basics

- **Open Source** Apache Project
- Distributed **Key Value Store**
- Maintains **configuration information**
- Stores **ACLs** and **Secrets**
- Enables highly reliable **distributed coordination**
- Provides **distributed synchronization**

Topics

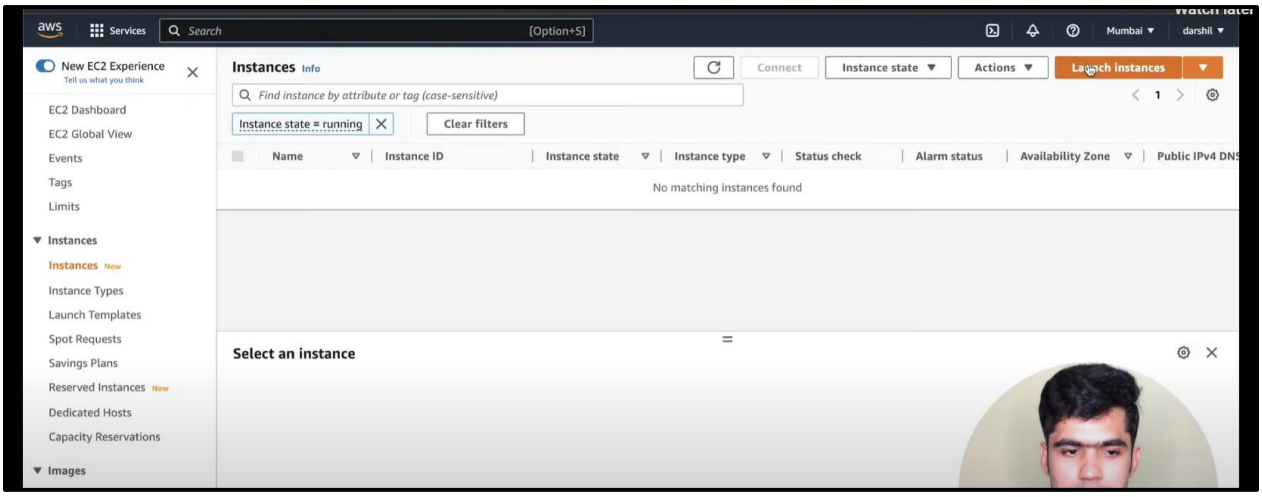
- **Topics**: Streams of "related" Messages in Kafka
  - Is a Logical Representation
  - Categorizes Messages into Groups
- Developers define Topics
- Producer ⟷ Topic: N to N Relation
- Unlimited Number of Topics

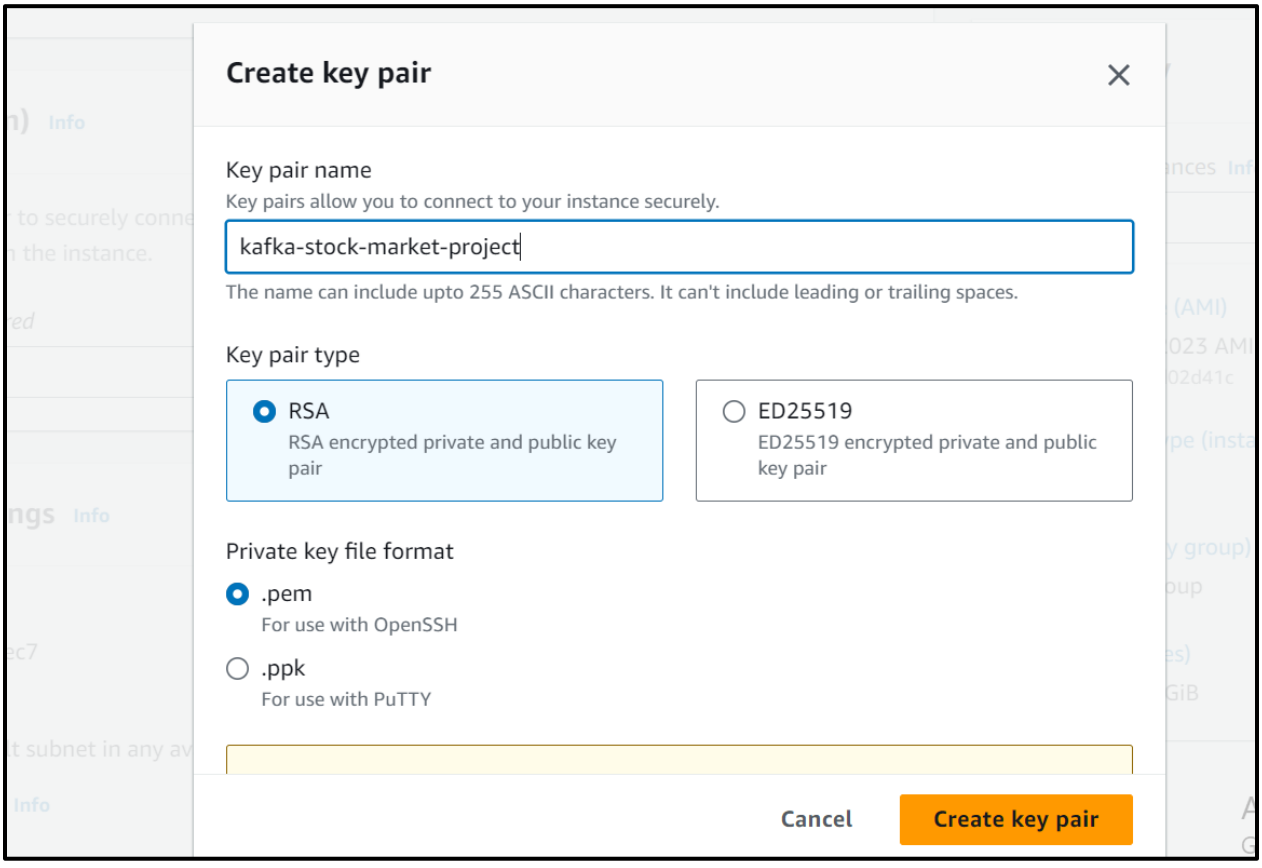


Topic is Basically, logical representation inside the kafka broker

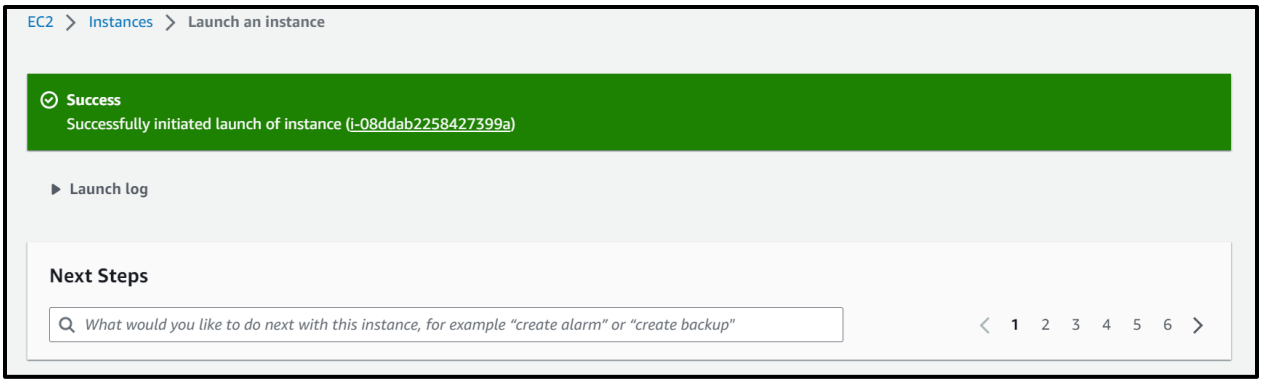
Attach some photos here

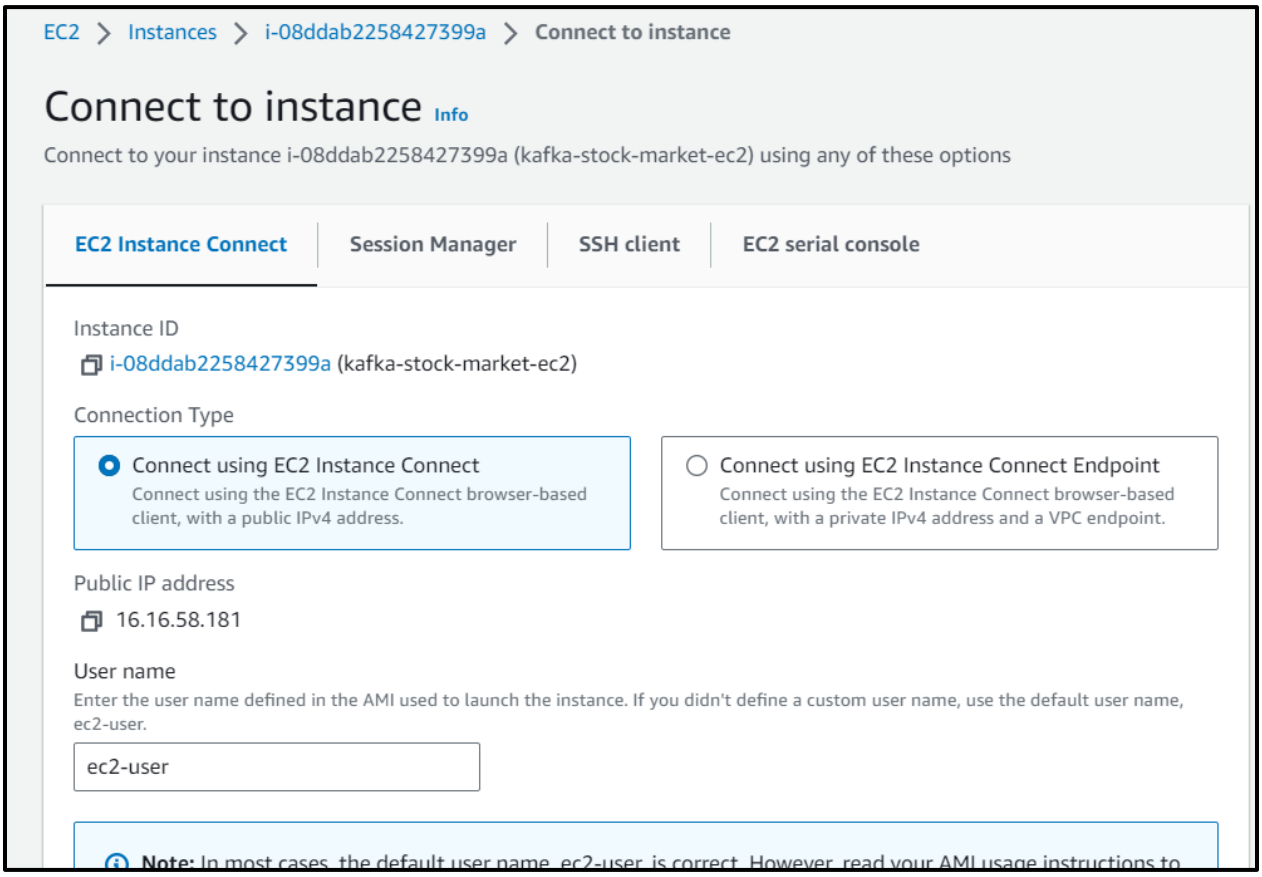
Creating a EC2 instance:



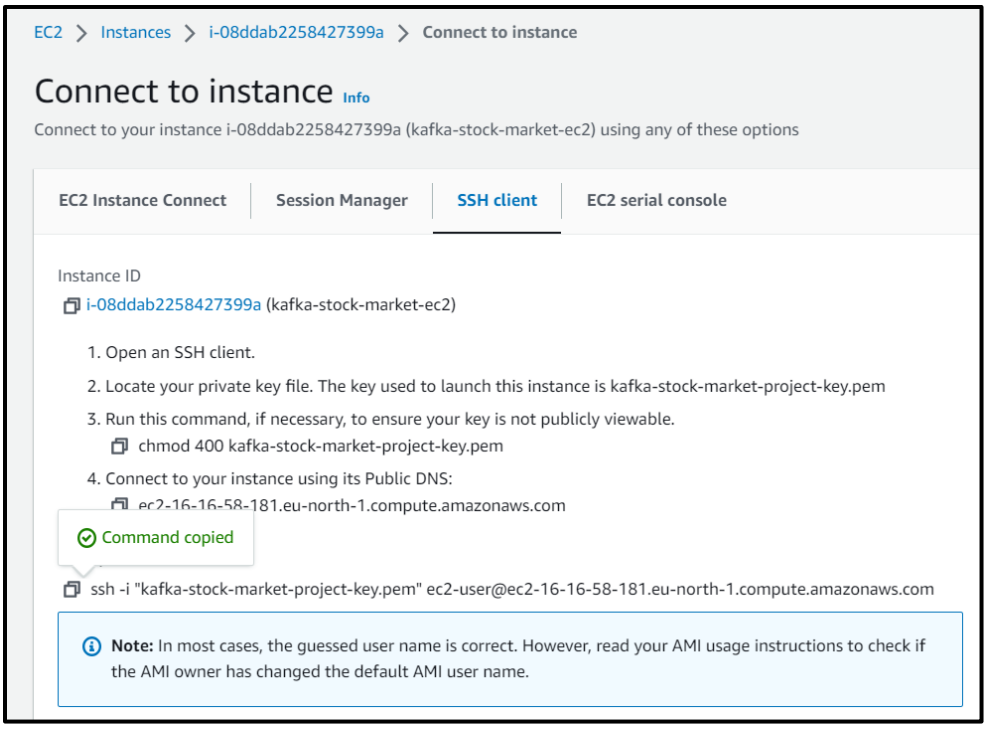


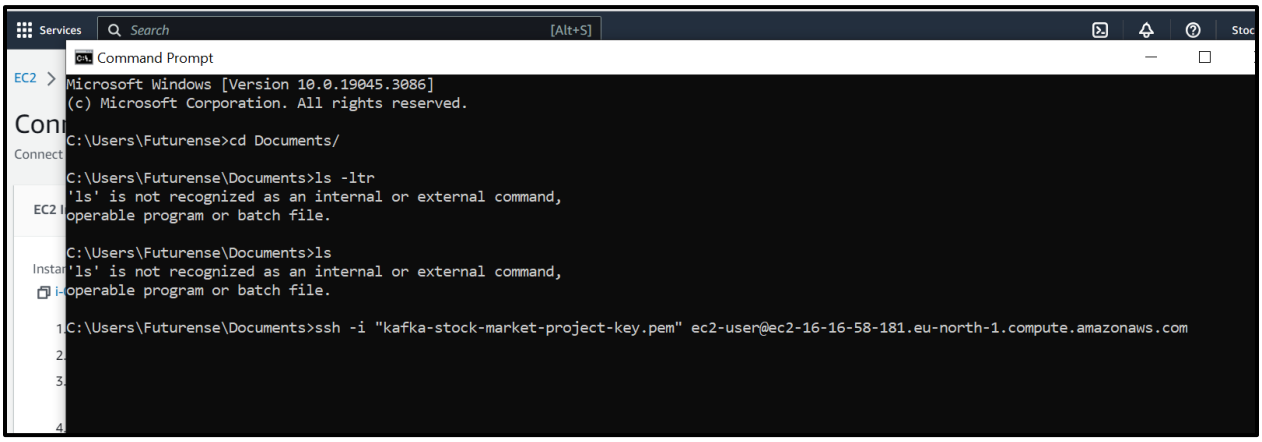
Launch the instance:





Here to connect from our local cmd we need ssh instance details: as given below





Get connected to our EC2 instance:

Download the kafka cluster inside our EC2 machine:

```
C:\Users\Futurense\Documents\kafka>ssh -i "kafka-stock-market-project-key.pem" ec2-user@ec2-16-16-58-181.eu-north-1.compute.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Aug  5 15:14:40 2023 from 122.171.17.148
[ec2-user@ip-172-31-22-150 ~]$ wget https://downloads.apache.org/kafka/3.5.1/kafka_2.12-3.5.1.tgz
--2023-08-05 15:18:29--  https://downloads.apache.org/kafka/3.5.1/kafka_2.12-3.5.1.tgz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.95.219, 2a01:4f8:10a:201a::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 106956505 (102M) [application/x-gzip]
Saving to: ‘kafka_2.12-3.5.1.tgz’

kafka_2.12-3.5.1.tgz             100%[=============================================================>] 102.00M  85.8MB/s    in 1.2s

2023-08-05 15:18:30 (85.8 MB/s) - ‘kafka_2.12-3.5.1.tgz’ saved [106956505/106956505]

[ec2-user@ip-172-31-22-150 ~]$
```

Now uncompress it: tar -xvf kafka_2.12-3.3.1.tgz

```
[ec2-user@ip-172-31-22-150 ~]$ ls
kafka_2.12-3.5.1  kafka_2.12-3.5.1.tgz
[ec2-user@ip-172-31-22-150 ~]$
```

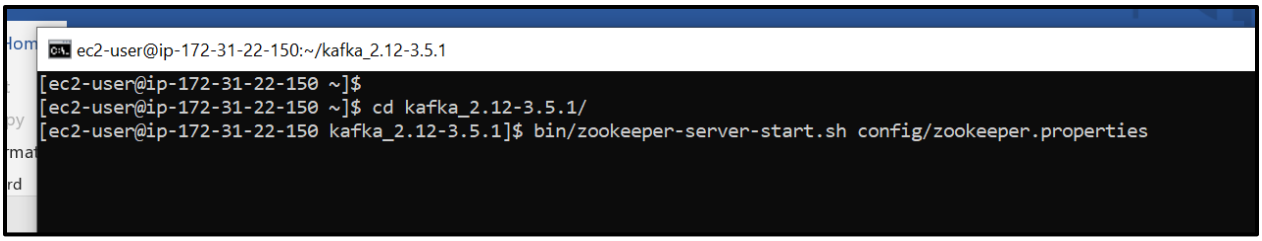
Apache Kafka will run on the top of the JVM (Java Virtual Machine):

Install java

```
[ec2-user@ip-172-31-22-150 ~]$ sudo yum install java
Last metadata expiration check: 0:42:52 ago on Sat Aug  5 14:51:56 2023.
Dependencies resolved.
================================================================================================================================================
 Package                                 Architecture        Version                                 Repository                Size
================================================================================================================================================
Installing:
 java-17-amazon-corretto                 x86_64              1:17.0.8+7-1.amzn2023.1                 amazonlinux              188 k
Installing dependencies:
```

```
[ec2-user@ip-172-31-22-150 ~]$ java --version
openjdk 17.0.8 2023-07-18 LTS
OpenJDK Runtime Environment Corretto-17.0.8.7.1 (build 17.0.8+7-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.8.7.1 (build 17.0.8+7-LTS, mixed mode, sharing)
[ec2-user@ip-172-31-22-150 ~]$
```

Now start the Zookeeper in kafka cluster:

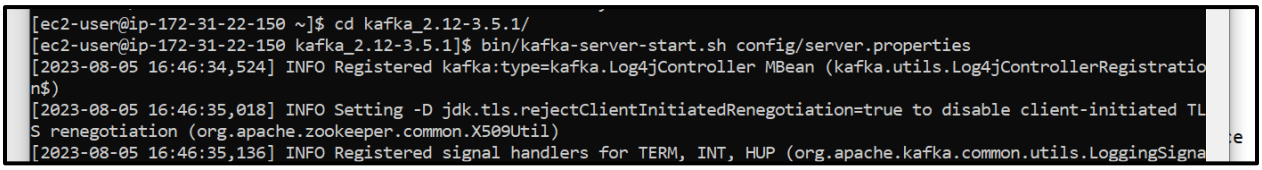


ssh -i "kafka-stock-market-project-key.pem" ec2-user@ec2-16-16-58-181.eu-north-1.compute.amazonaws.com

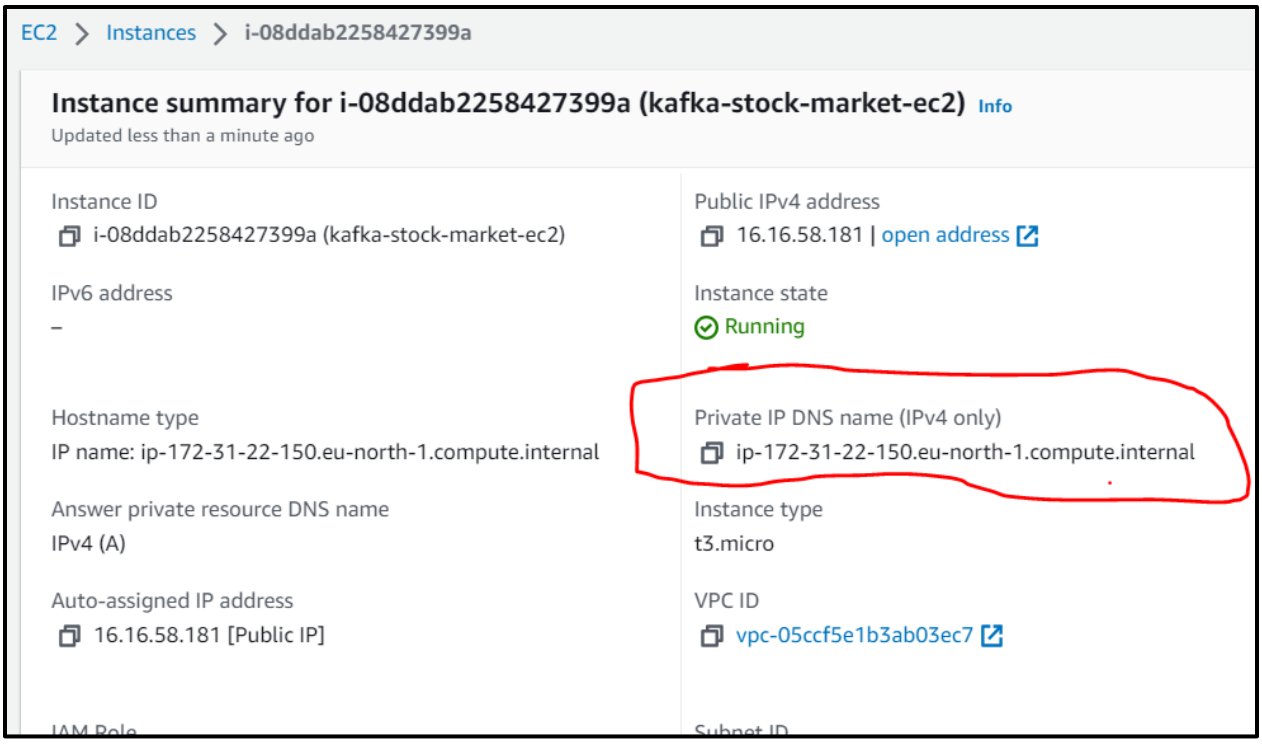
Increase the memory of th kafka server:

export KAFKA_HEAP_OPTS="-Xmx256M -Xms128M"

Now run the kafka server after increasing the memory:



Here we can not access the kafka server as it is a private address



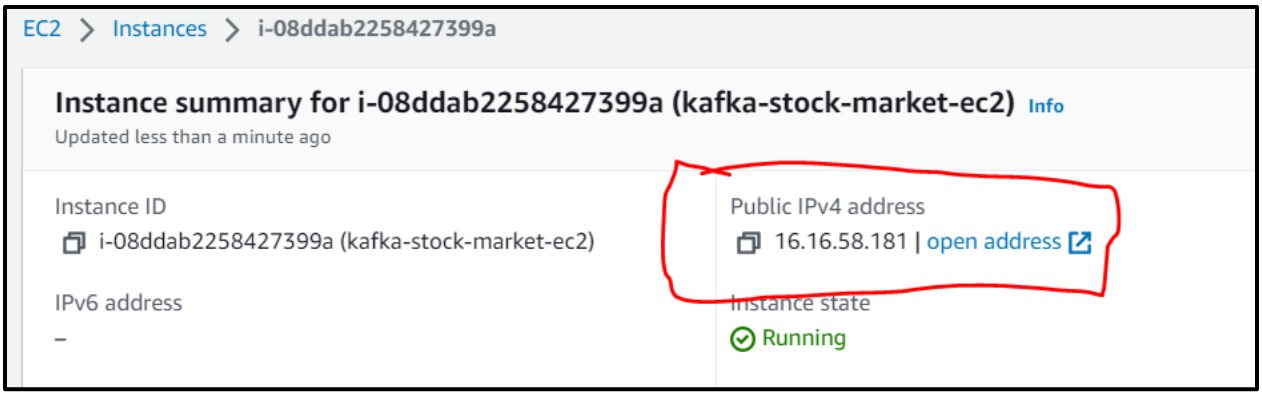
To access the server we need to change that to public ip

So, first stop the servers both zookeeper and kafka server

Now in the ec2 instance: run the below command

sudo nano config/server.properties

Change the configuration according to below:



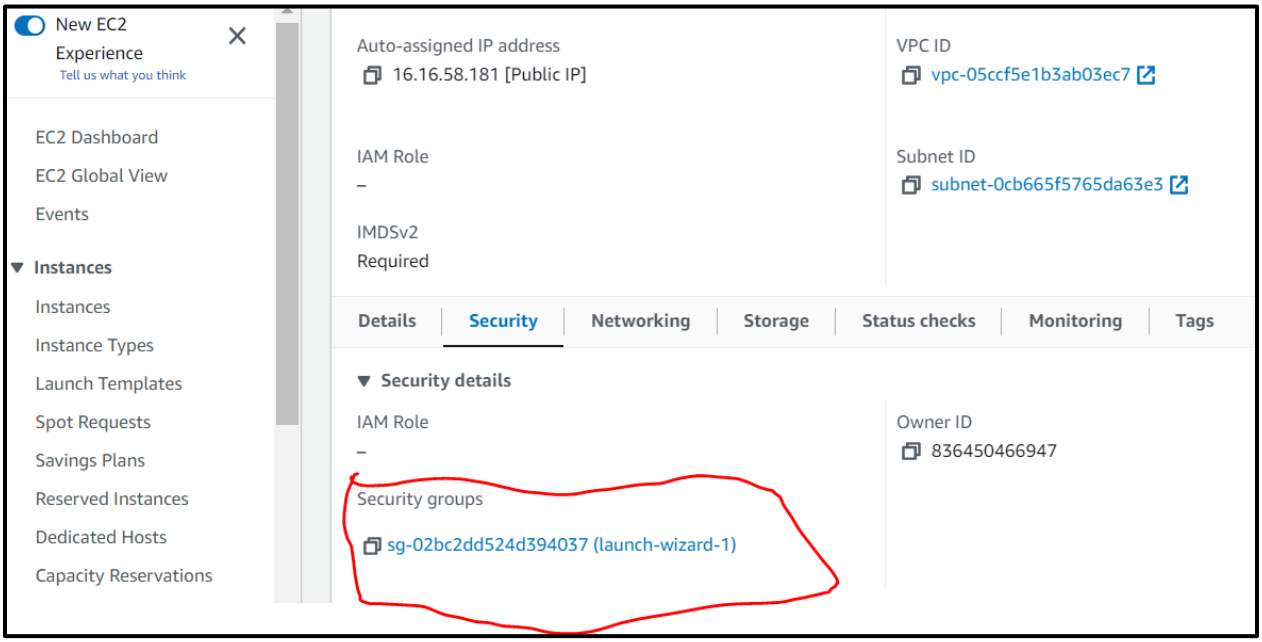
Enter the below the above address

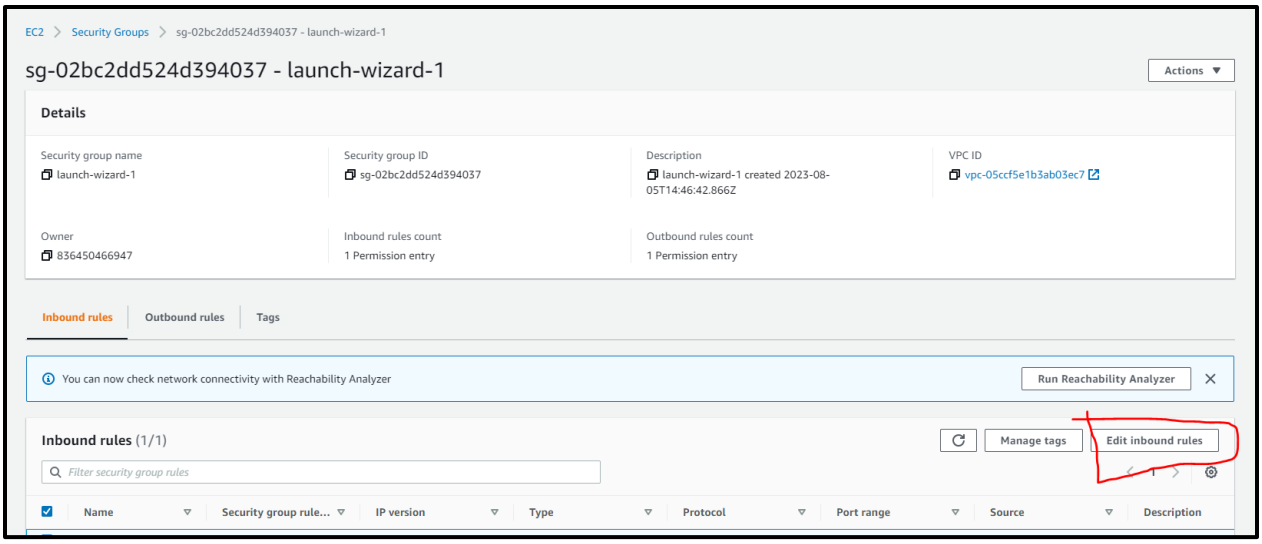
```
# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
advertised.listeners=PLAINTEXT://65.2.168.105:9092
```

Now again run the zookeeper server:

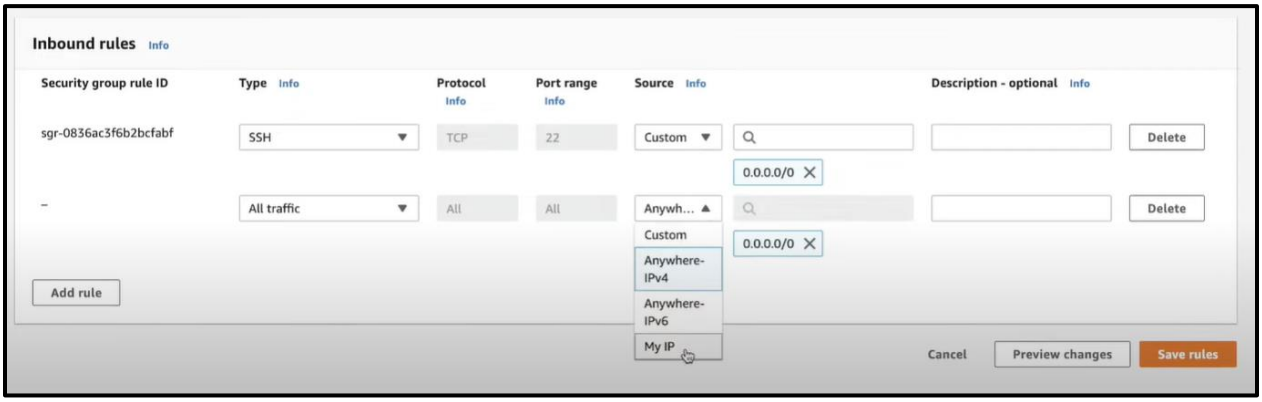
Run the kafka server as well

Now we have to give permission access to our local machine to the ec2 instance



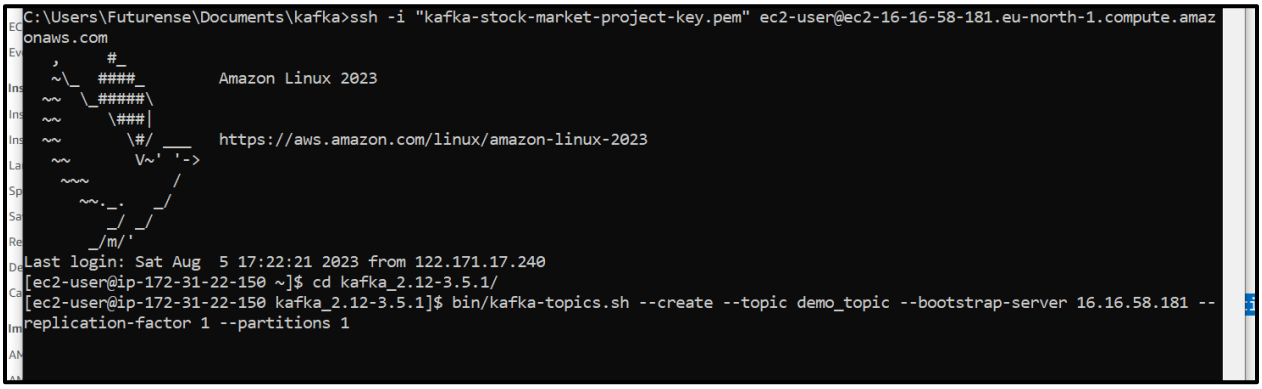


Here select ‘Anywhere IPv4’ is recommended for this project: as below

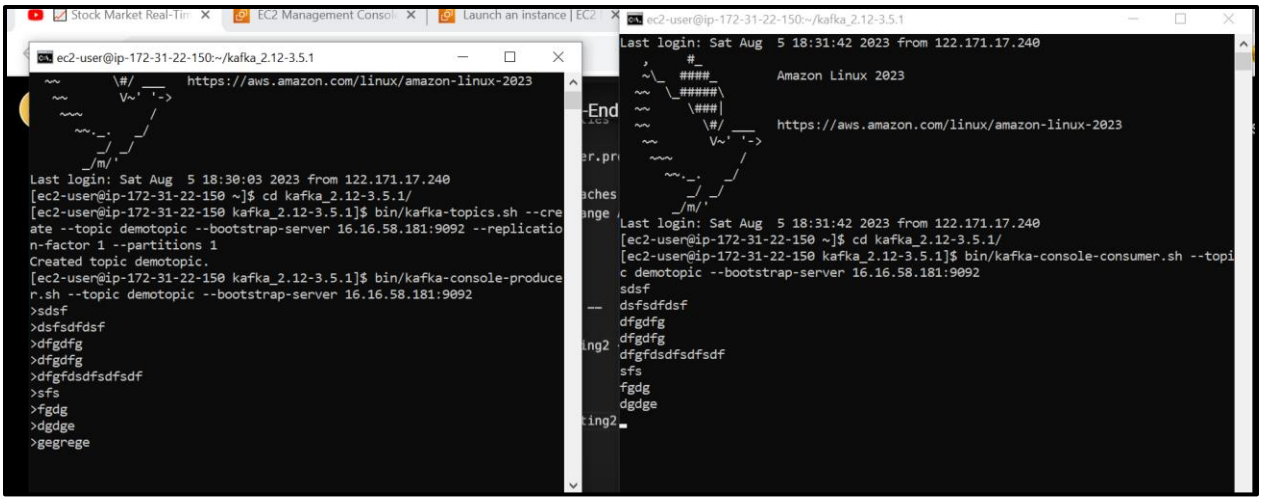


Now we have to make topic, producer and consumer

Create topic inside the kafka:

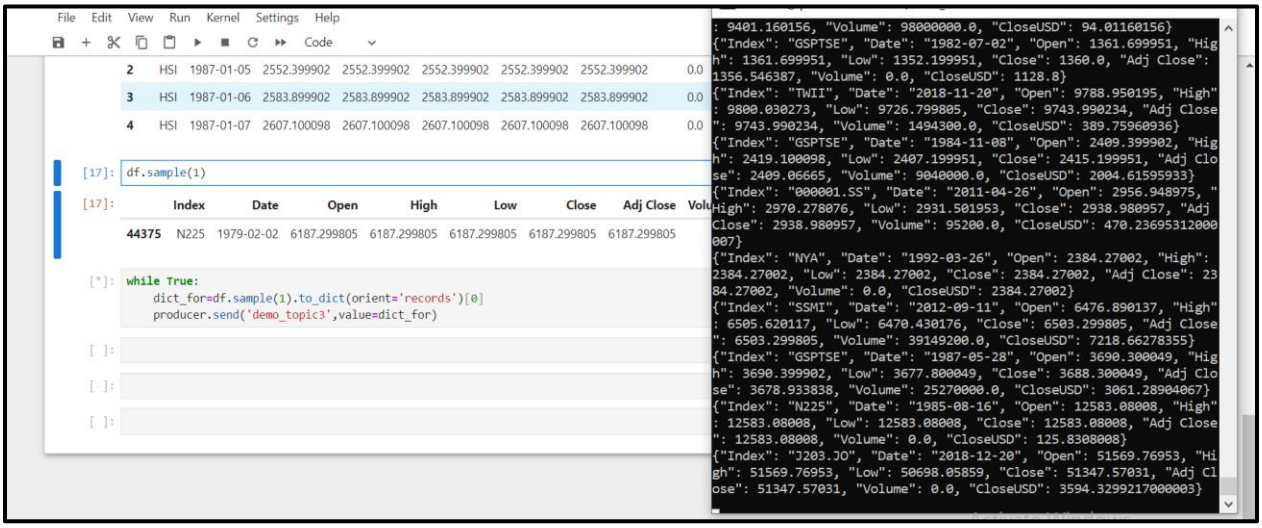


Create a producer in another terminal:



Connecting Jupyter notebook with ec2 instance and sending data

Looping through the csv file samples from the jupyter notebook:



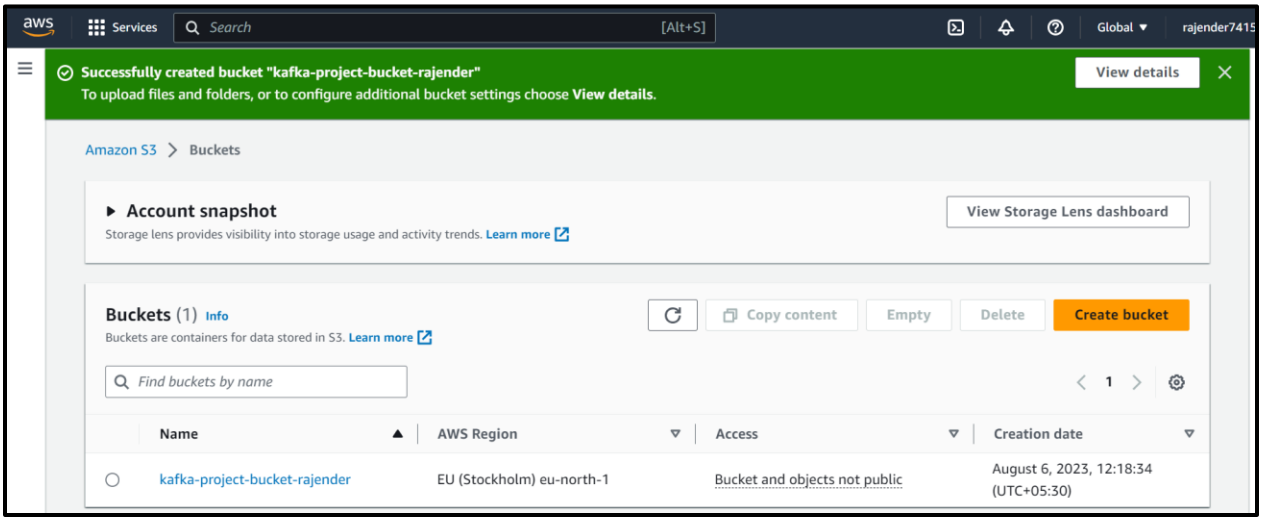
While running the kafka server will go down

Because we are running kafka server in a small machine with one broker and one single partition

Now flush the producer: **producer.flush()**

Step last:

Create a S3 bucket in AWS:



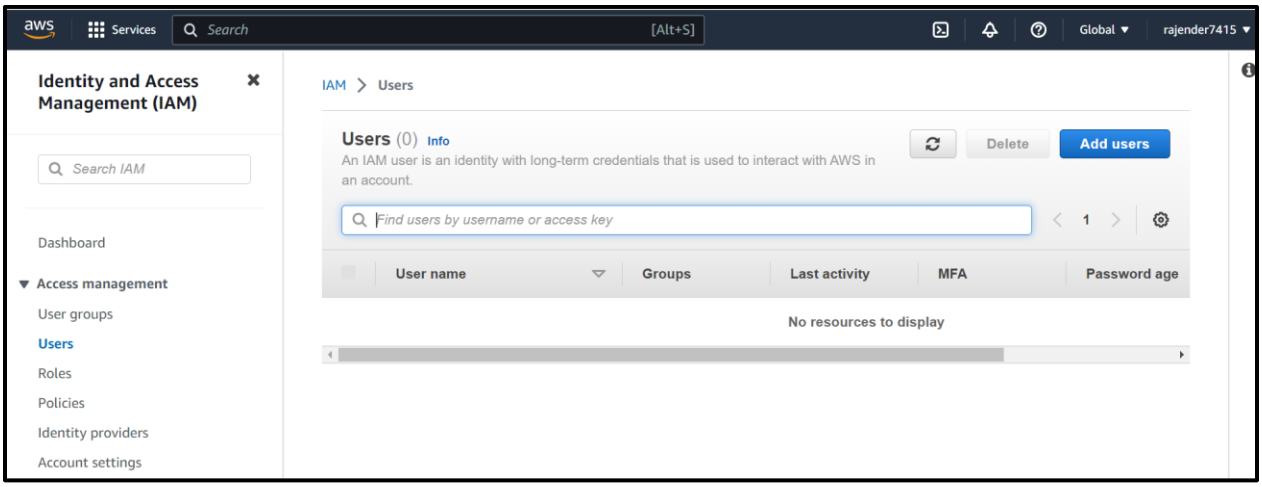
Here region should be same

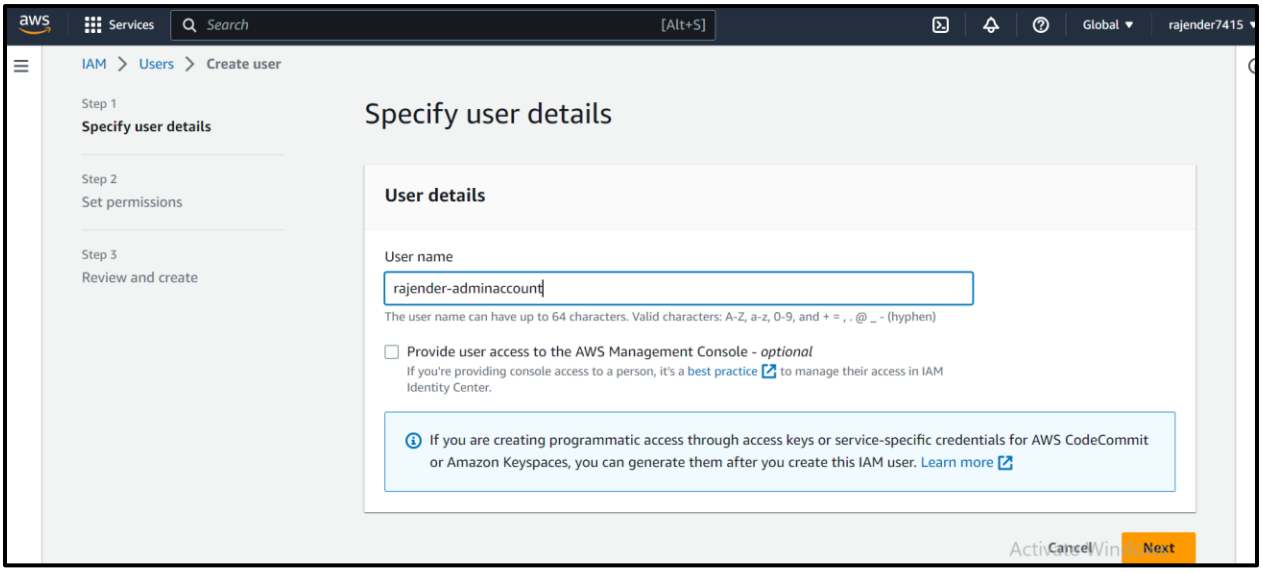
And all options should be by default

Now upload csv file data to the S3 bucket from jupyter notebook:

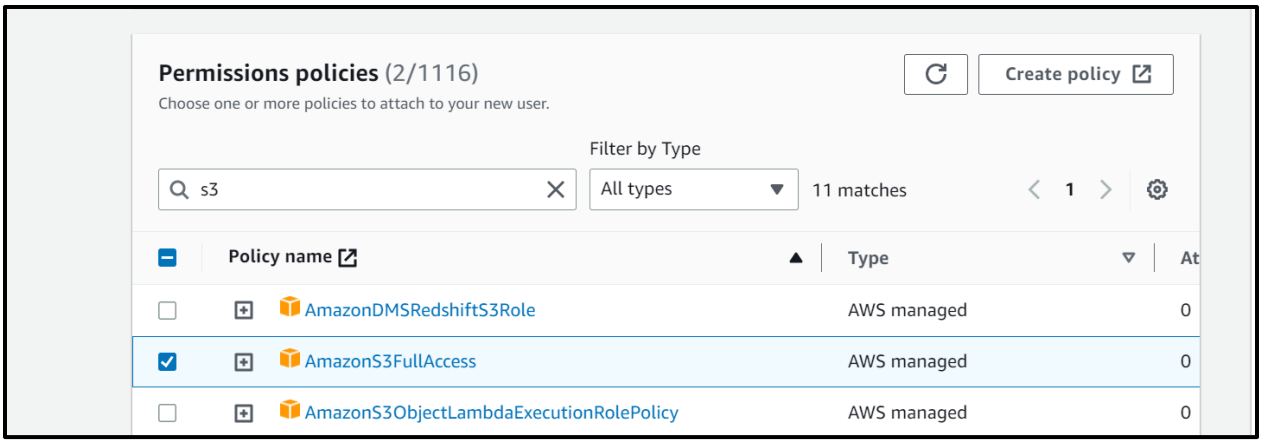
But we need to access to that s3 bucket

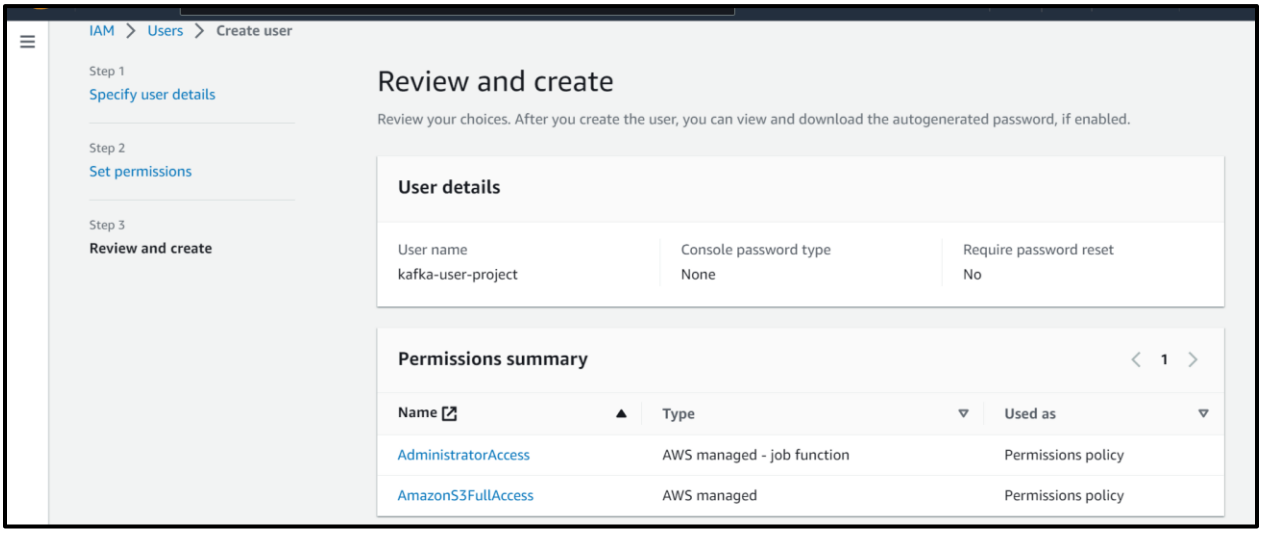
Go tot IAM > go to users



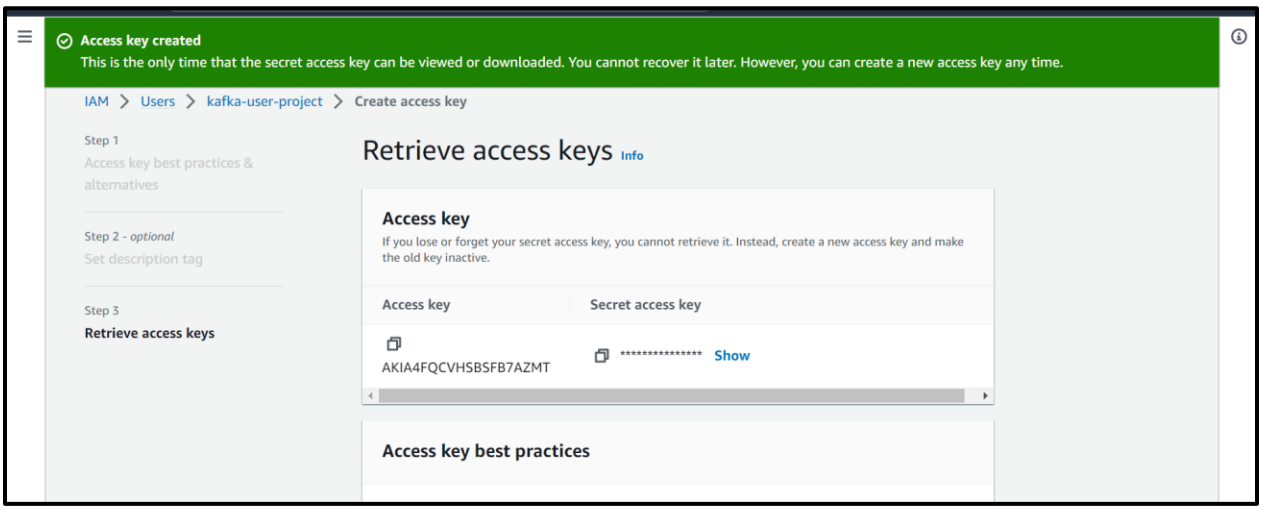


Give access administrator access and Amazons3FullAccess





Here two things: Access key ID, secret access key



Act ID

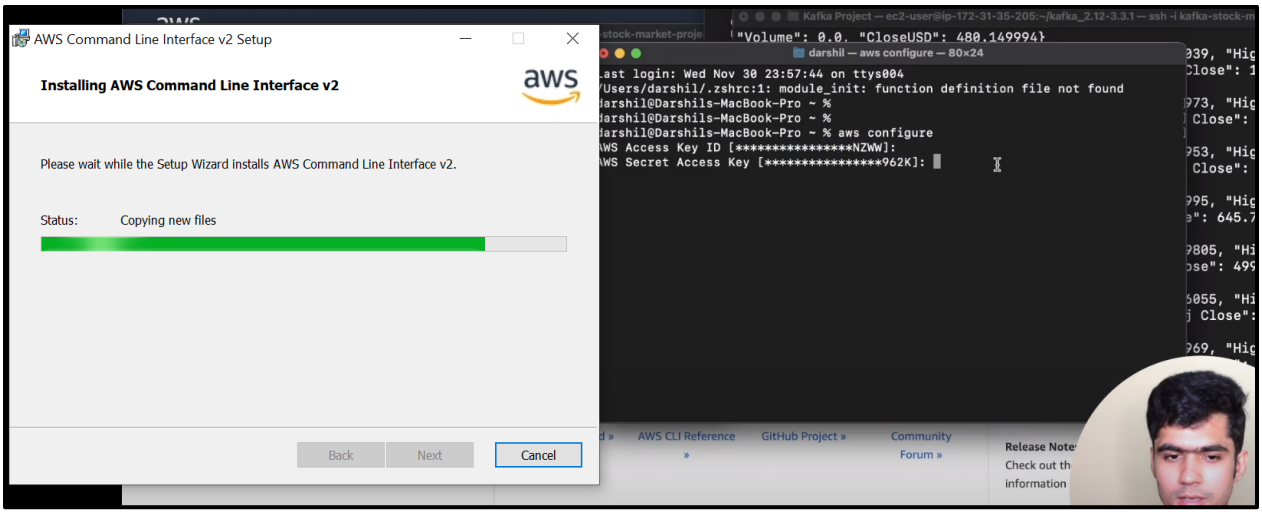
AKIA4FQCVHSBSFB7AZMT

key

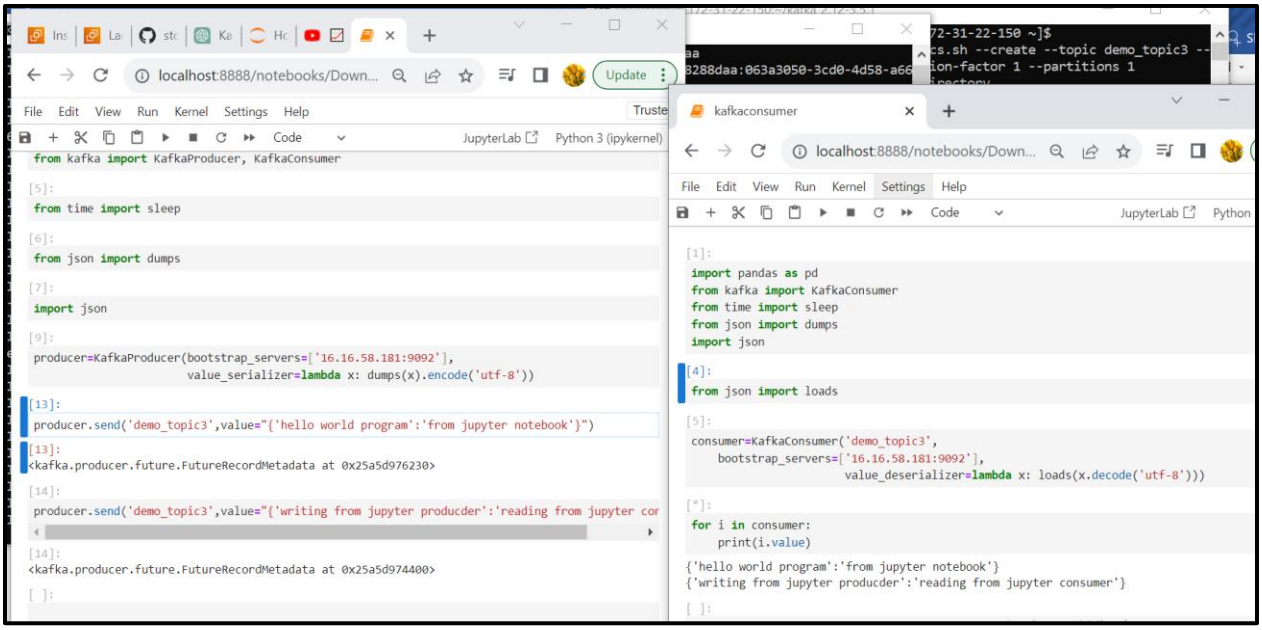
bXGUwPS34Bc4JnLdRjfVfQNNesBTVkIloOrsEwkn

Download the aws command line interface (CLI) from internet and install it:

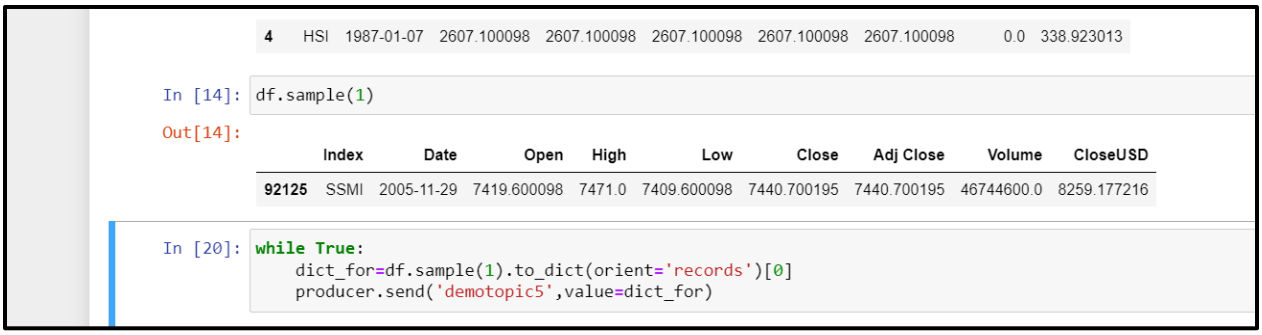
Now run the cmd:



Doing it in jupyter notebooks (producer and consumer)



In producer:



4	HSI	1987-01-07	2607.100098	2607.100098	2607.100098	2607.100098	2607.100098	0.0	338.923013

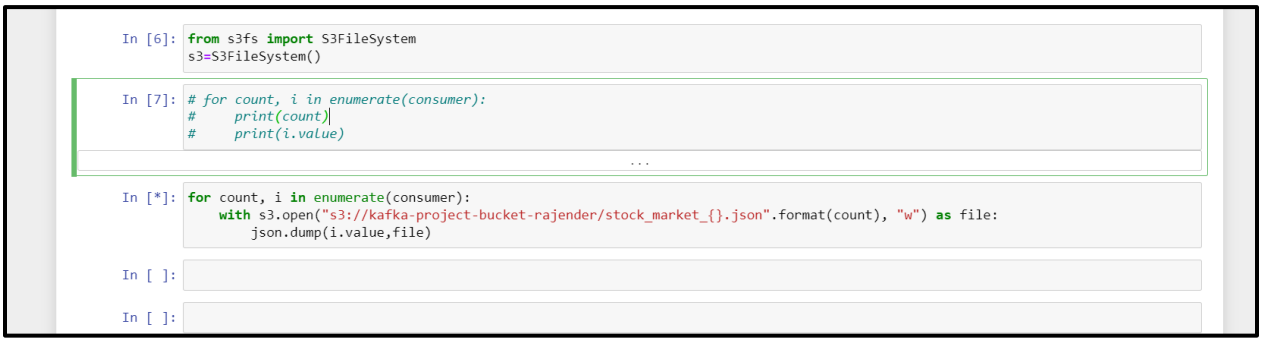
```
In [14]: df.sample(1)
```

Out[14]:

	Index	Date	Open	High	Low	Close	Adj Close	Volume	CloseUSD
92125	SSMI	2005-11-29	7419.600098	7471.0	7409.600098	7440.700195	7440.700195	46744600.0	8259.177216

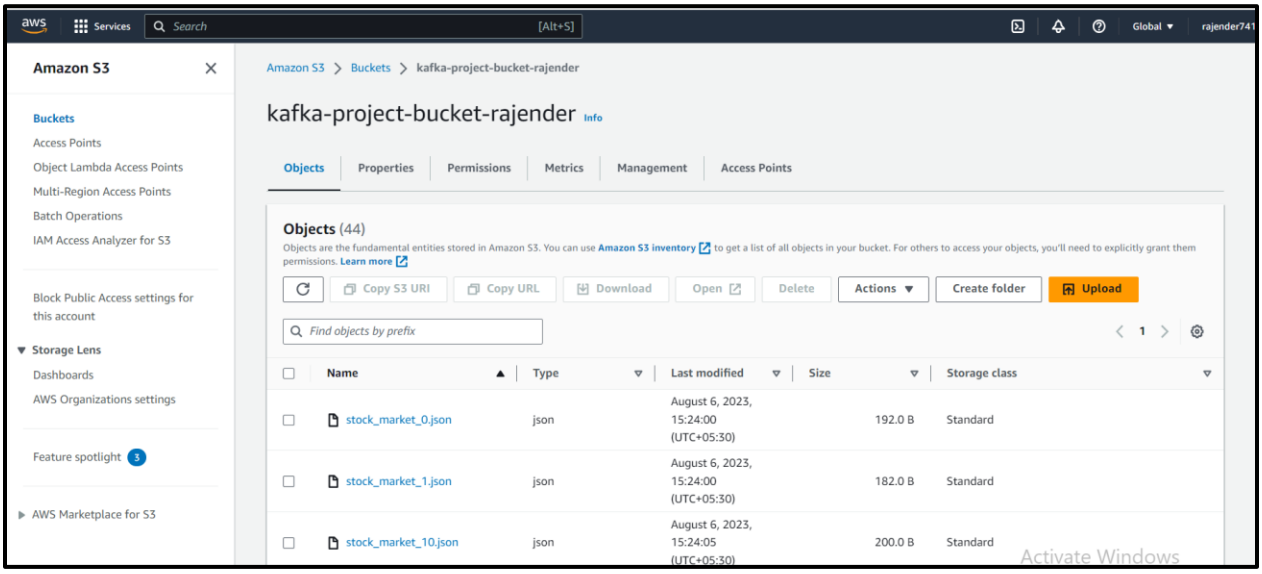
```
In [20]: while True:
             dict_for=df.sample(1).to_dict(orient='records')[0]
             producer.send('demotopic5',value=dict_for)
```

In consumer:



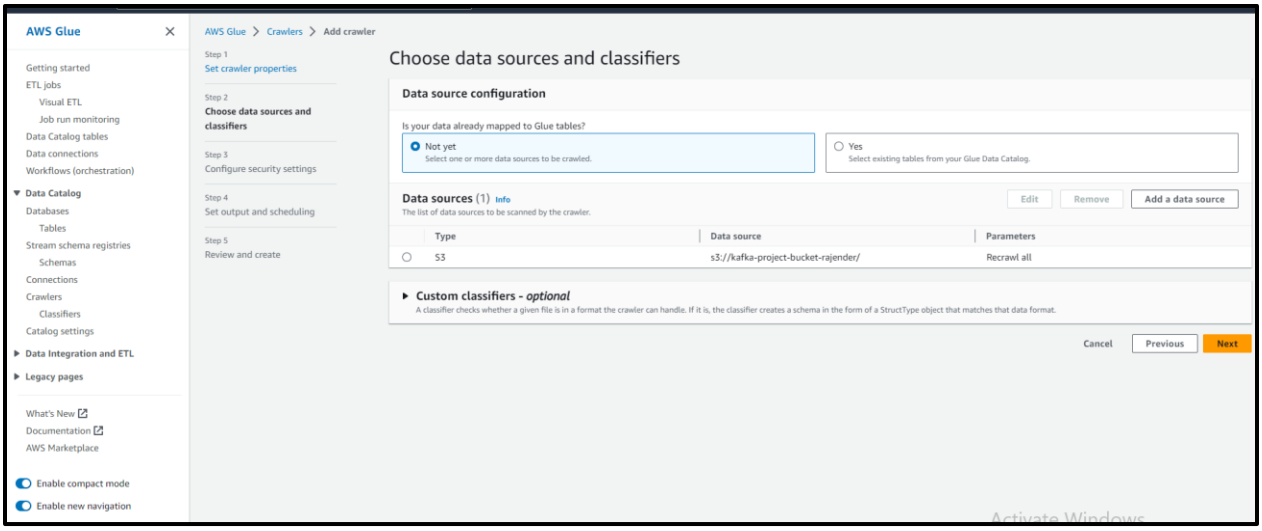
```
In [6]: from s3fs import S3FileSystem
        s3=S3FileSystem()

In [7]: # for count, i in enumerate(consumer):
        #     print(count)
        #     print(i.value)

In [*]: for count, i in enumerate(consumer):
            with s3.open("s3://kafka-project-bucket-rajender/stock_market_{}.json".format(count), "w") as file:
                json.dump(i.value,file)

In [ ]:

In [ ]:
```

See the output:



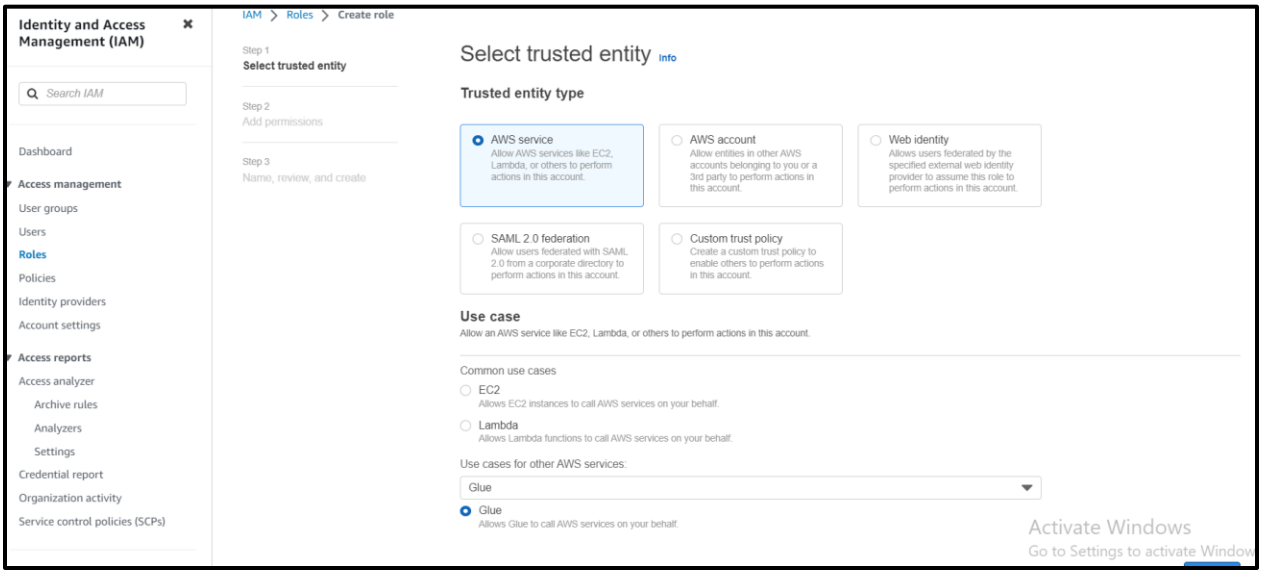
Now we have to create the crawler:

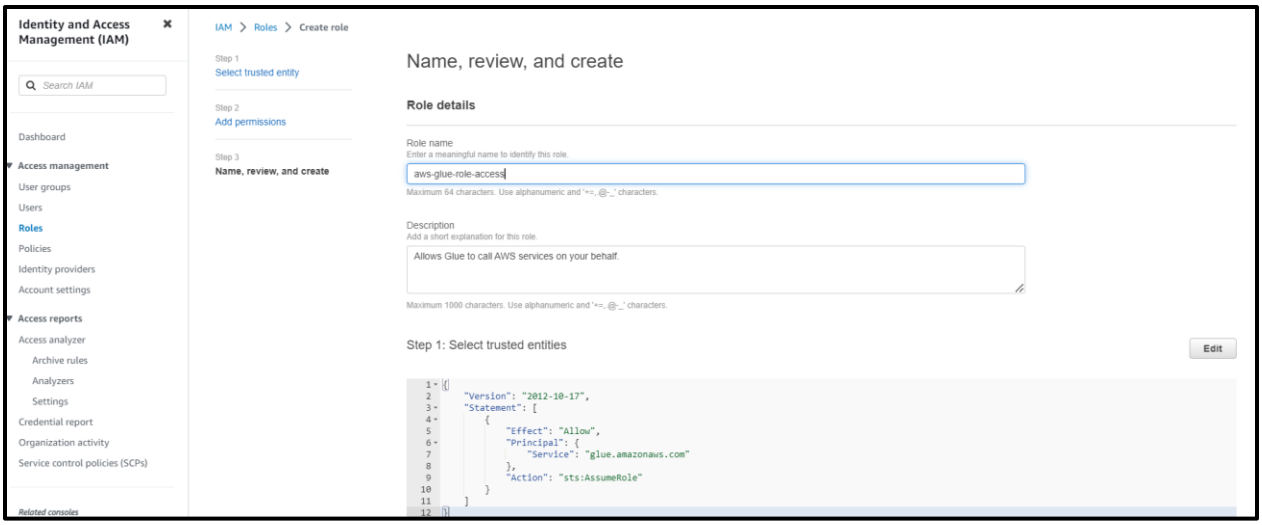
- The crawler will crawl the schema of the files
- The scrawler helps to query the data using Athena
- It will take the data source so that it can be useful in Athena



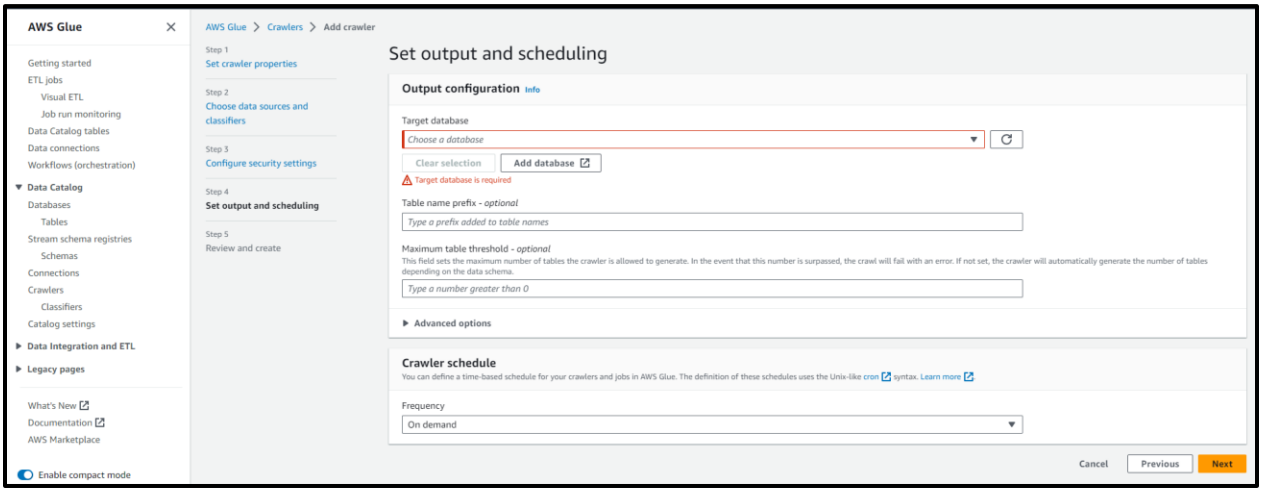
Now during the creation of the crawler, it will ask for the role -- - - - - which is glue/crawler wants to talk to s3 we need role

Create a role inside IAM:

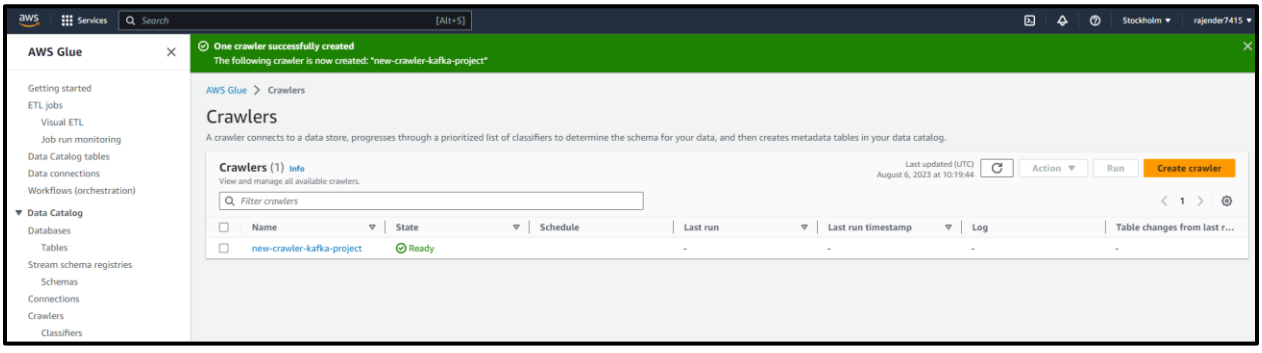


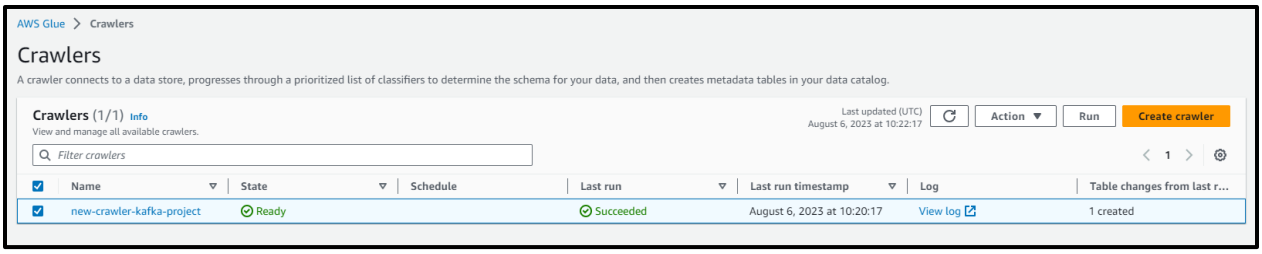


Create a database for the crawler:

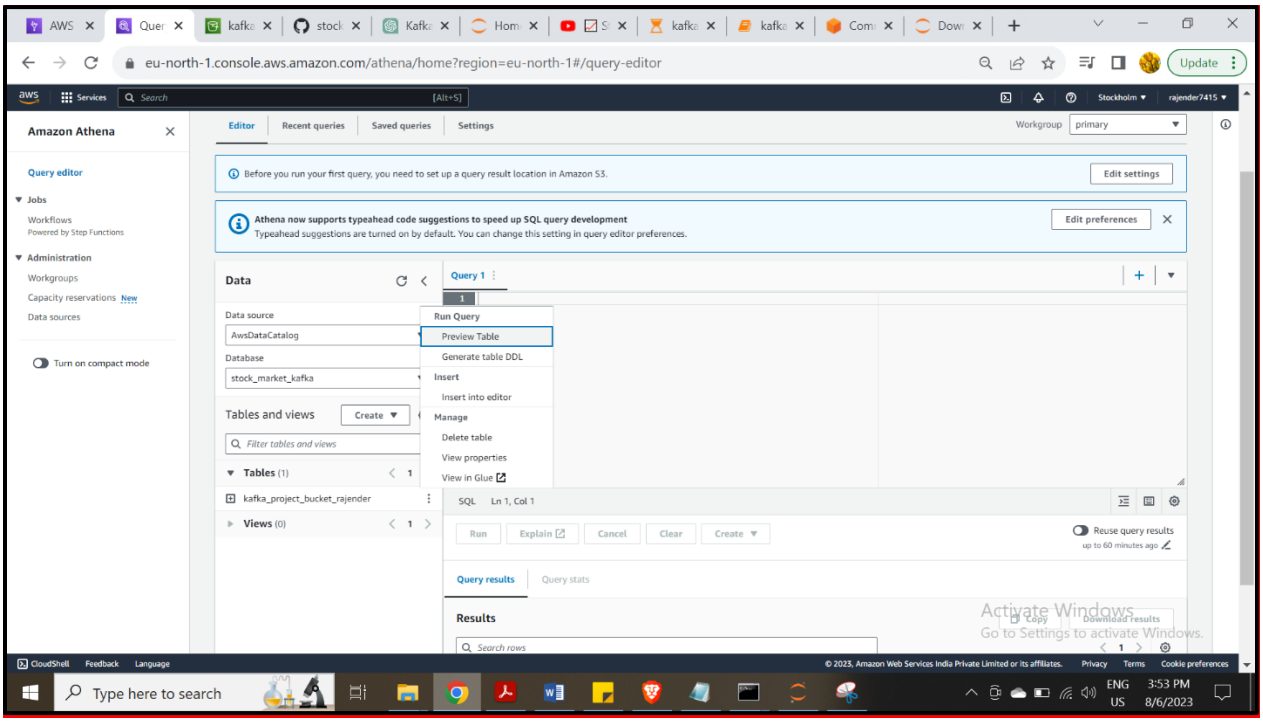


Crawler should be ready before querying:



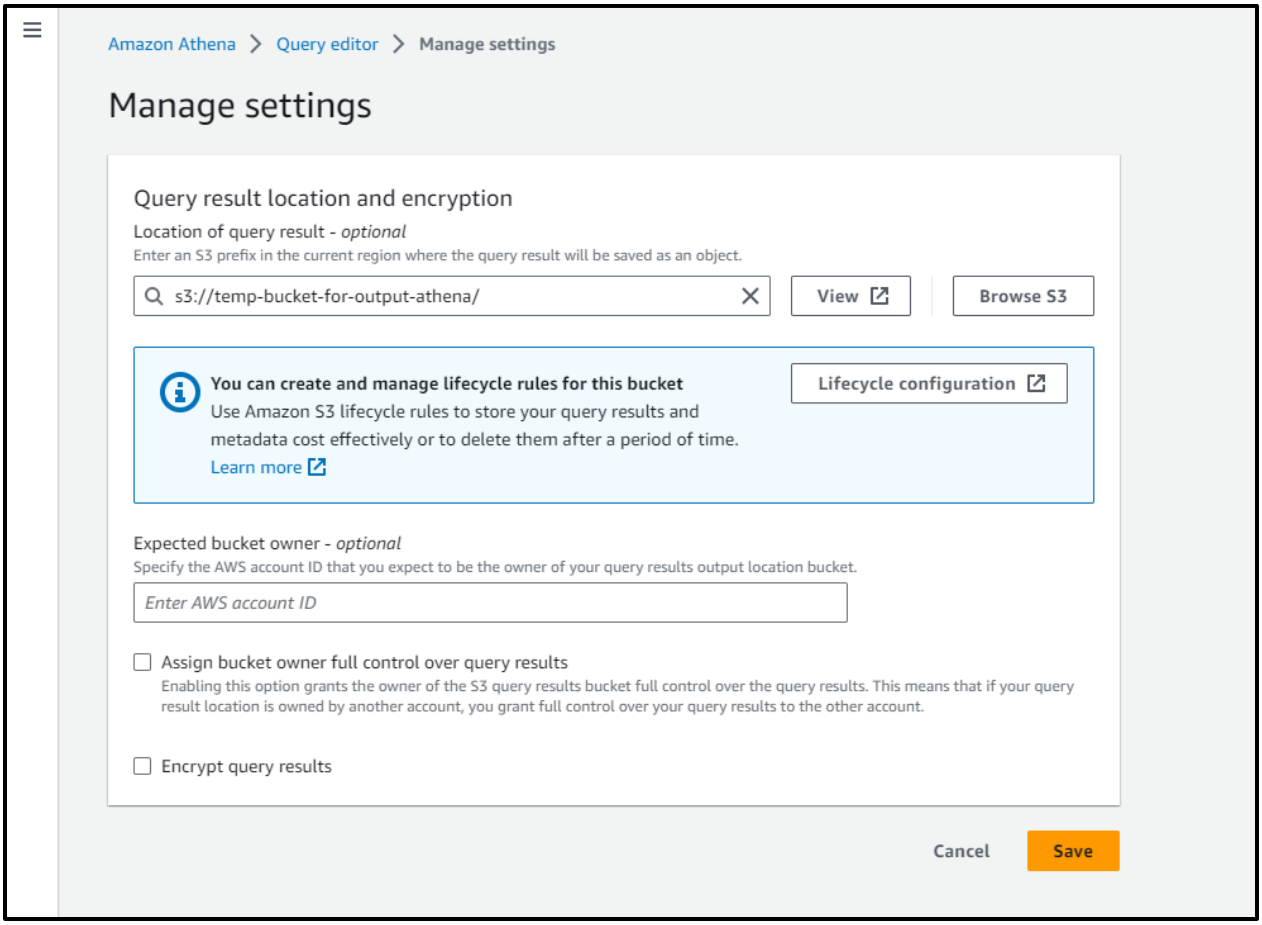


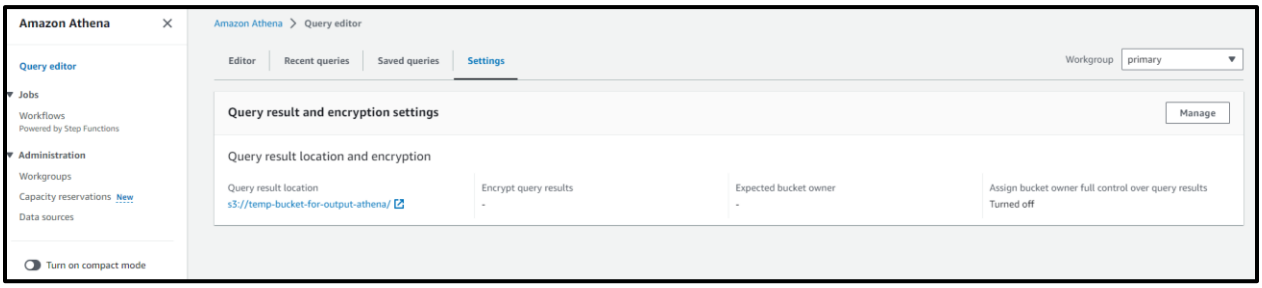
Now go to Athena- - - - - here we can see the database and we can start querying

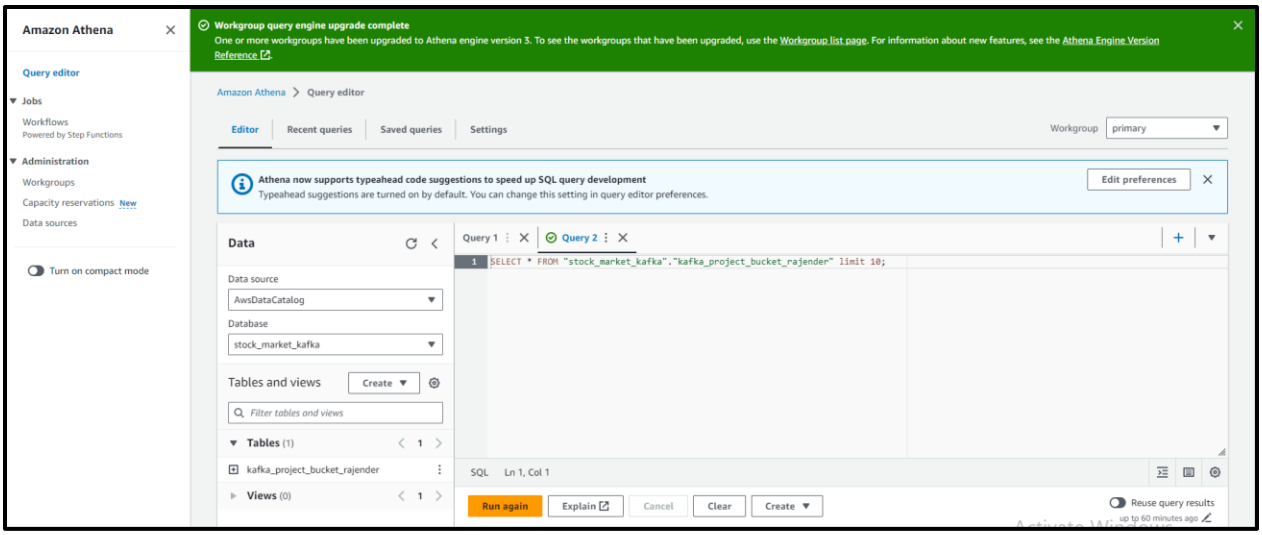


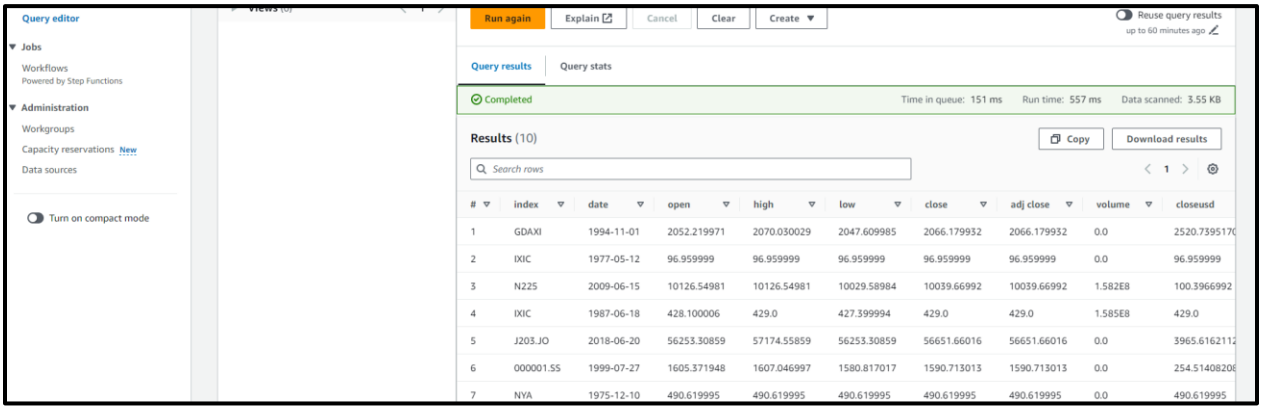
Creating a temp bucket for output storage:

Make sure the temp bucket in the same region



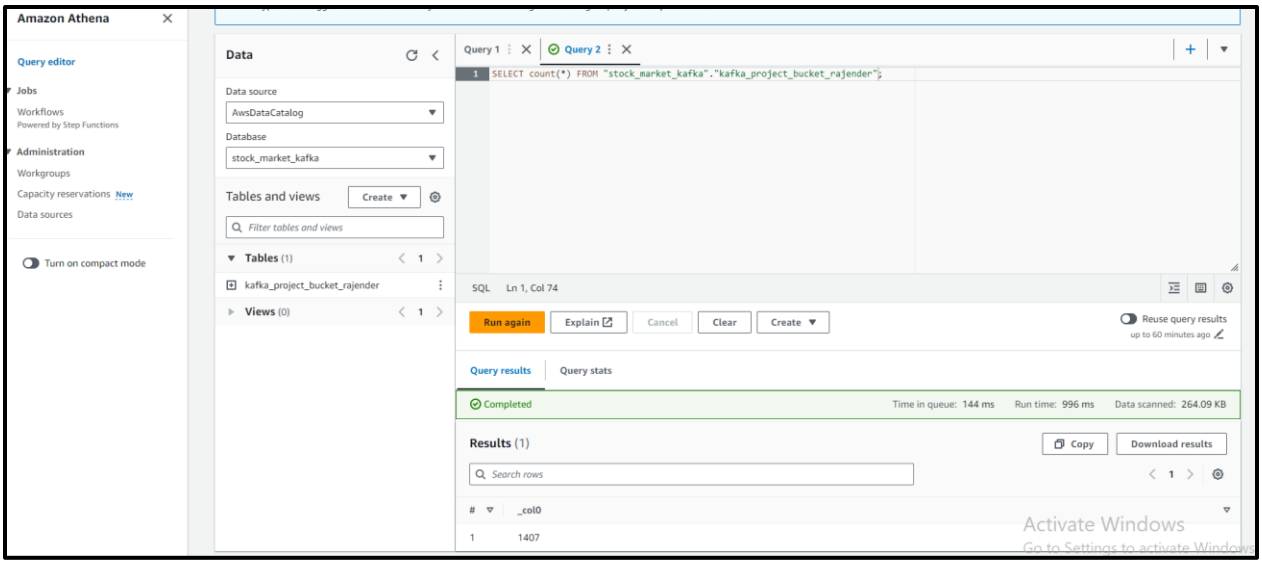


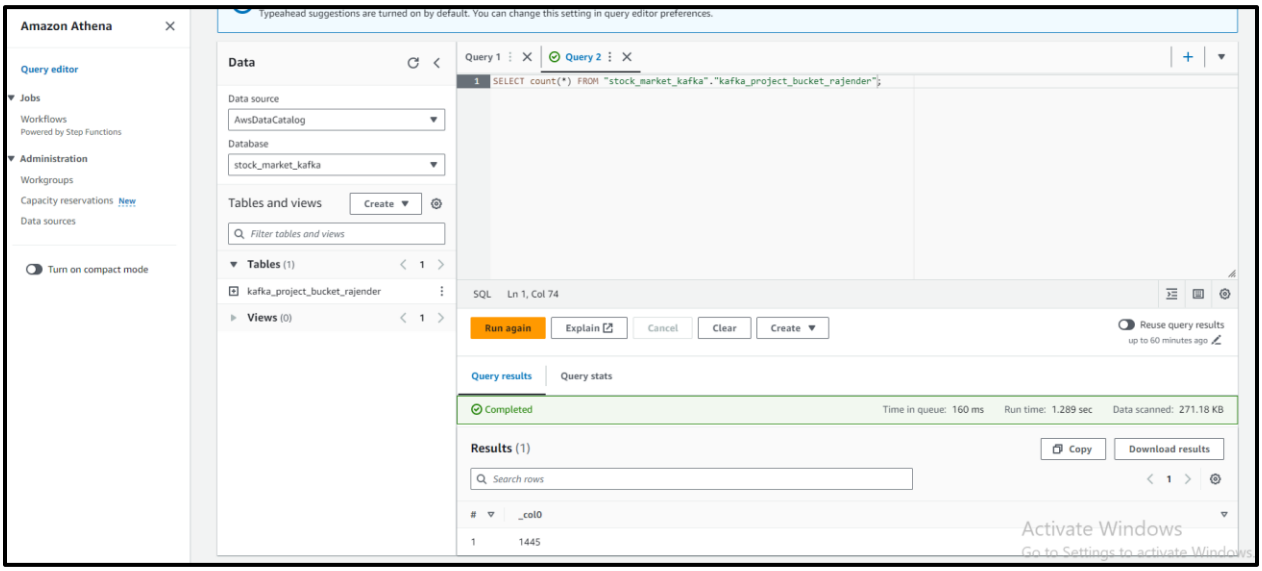




Now add sleep to the producer:

```
In [*]: while True:
            dict_for=df.sample(1).to_dict(orient='records')[0]
            producer.send('demotopic5',value=dict_for)
            sleep(2)
```





See the number of files keep getting increased