Veermata Jijabai Technological Institute, Mumbai 400019

Experiment No.: 07

Aim: Setup and install Apache Kafka and stream realitime data from any social media website like Twitter, Facebook, instagram etc

Name: Kiran K Patil

Enrolment No.: 211070904

Branch: Computer Engineering

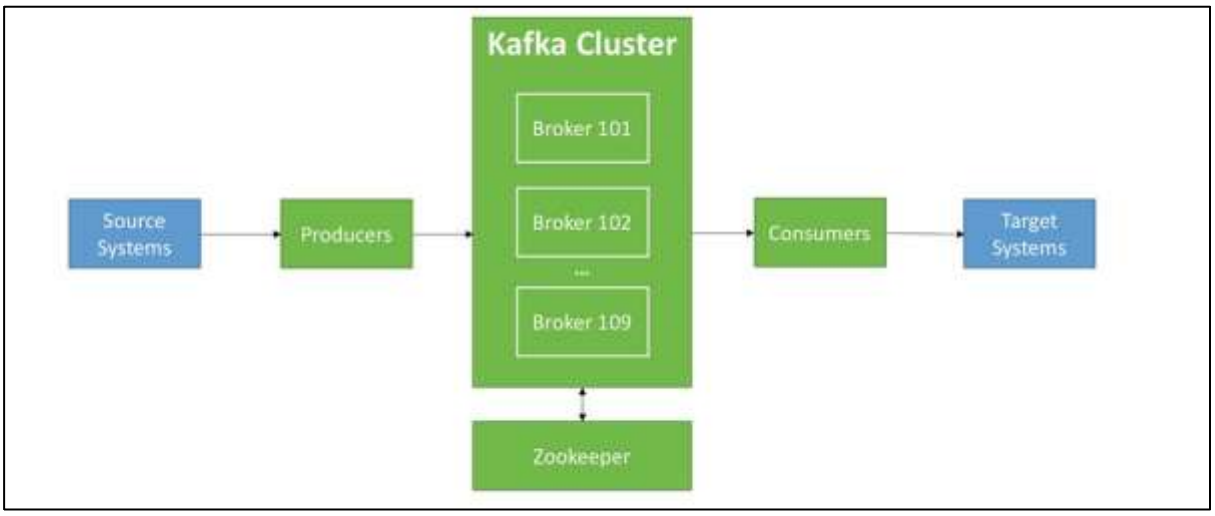
Batch: B

Course: Big Data Analytics Lab

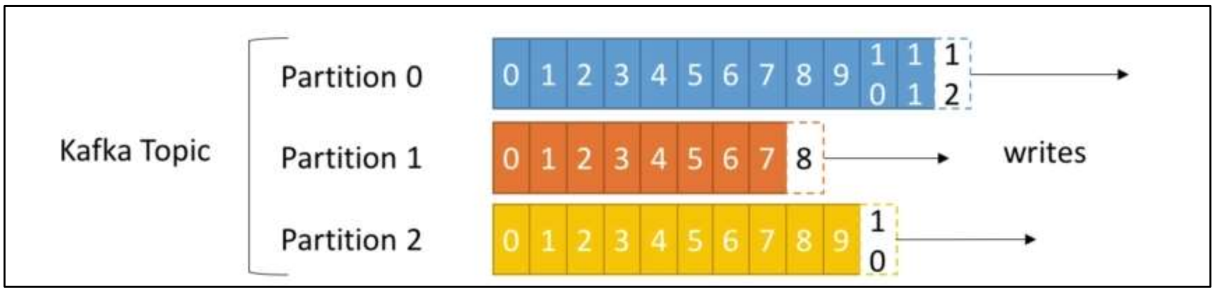
Theory:

Kafka :

Apache Kafka is a distributed streaming platform used for building real-time data pipelines and applications. It handles high-throughput, fault-tolerant data streams, allowing producers to publish messages to topics and consumers to subscribe and process them. Kafka's key components include topics, brokers, partitions, producers, consumers, consumer groups, connectors, and the Streams API. It's widely used across industries for its performance, scalability, and reliability in handling streaming data.



Topics, Partitions and Offsets



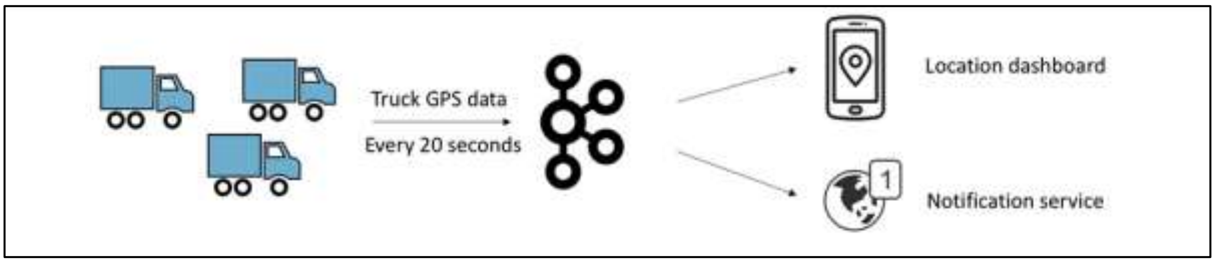
Topics : a particular stream of data,Similar to a table in database (without the constraints). You can have many topics as you want and the topic is identified by its name.

Topics are split in partitions. Each partition is ordered and each message within a partition gets an incremental id, called offset. Offsets only have a meaning for a specific partition. For Example, offset 3 in partition 0 doesn't represent the same as offset 3 in partition 1.

Order of offset is guaranteed only within a partition, not across partitions. Each message within a partition gets an incremental id called offset.

You need to specify how many partitions there should be, when creating a topic (you can change this later). Data in Kafka is kept only for a limited amount of time. Data gets deleted over time, By default it is one week. Once the data is written to a partition, it cannot be changed. (immutability)

Say you have a fleet of trucks, each truck reports its GPS position to Kafka. You can have a topic *TRUCK_GPRS* that contains the position of all trucks. Each truck will send a message to Kafka every 20 seconds, each message will contain the truck ID and the truck position (latitude and longitude).

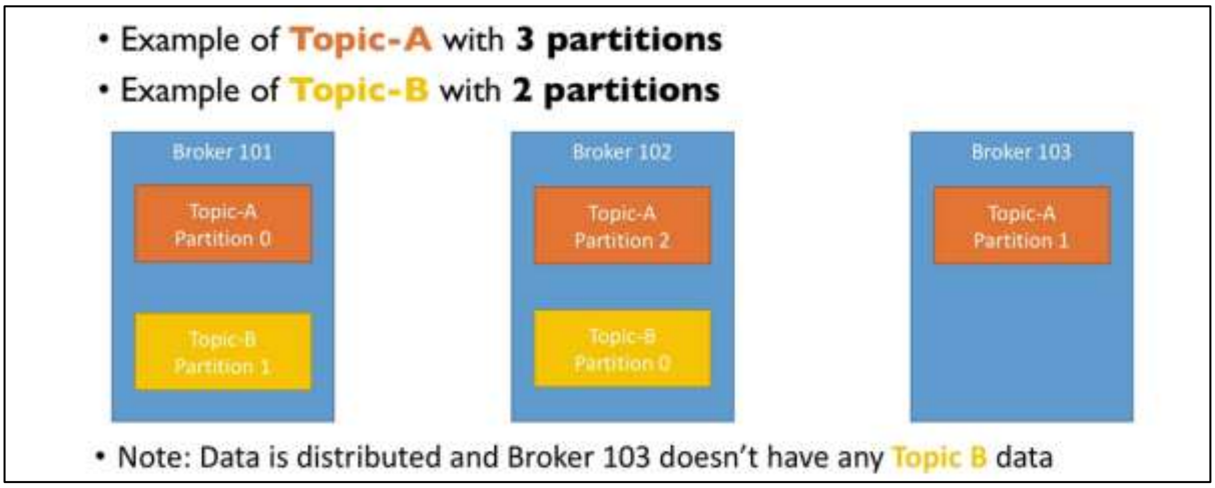


We choose to create that topic with 10 partitions (this is an arbitrary number). We have each truck reporting/sending messages of their own position to Kafka, through the same *TRUCK_GPRS* topic. (you don't have one topic per truck, trucks are going to send data to the same topic).

Then we have consumers for our data. you can have Location dashboard for your Employees so they can look at the truck data, or maybe you want to have a Notification service , for example when a truck is running out of fuel, or whatever you want.

Brokers

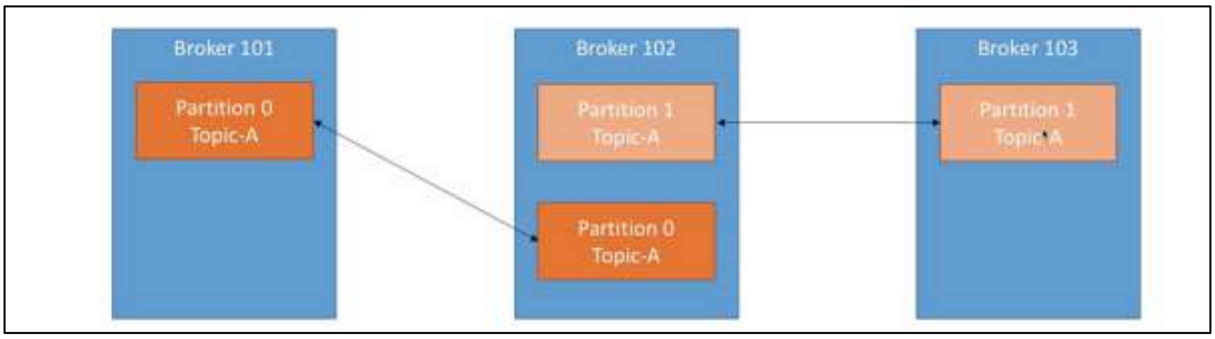
A Kafka cluster is composed of multiple brokers(cluster means that it's composed of multiple brokers and each broker basically is a server). Each broker is identified with its id (id is going to be a number : integer). Each broker will only contain certain topic partitions. After connecting to any broker, (called bootstrap broker), you will be connected to the entire cluster.


• Example of Topic-A with 3 partitions
• Example of Topic-B with 2 partitions
Broker 101
Topic-A Partition 0
Topic-B Partition 1
Broker 102
Topic-A Partition 2
Topic-B Partition 0
Broker 103
Topic-A Partition 1
• Note: Data is distributed and Broker 103 doesn't have any Topic B data

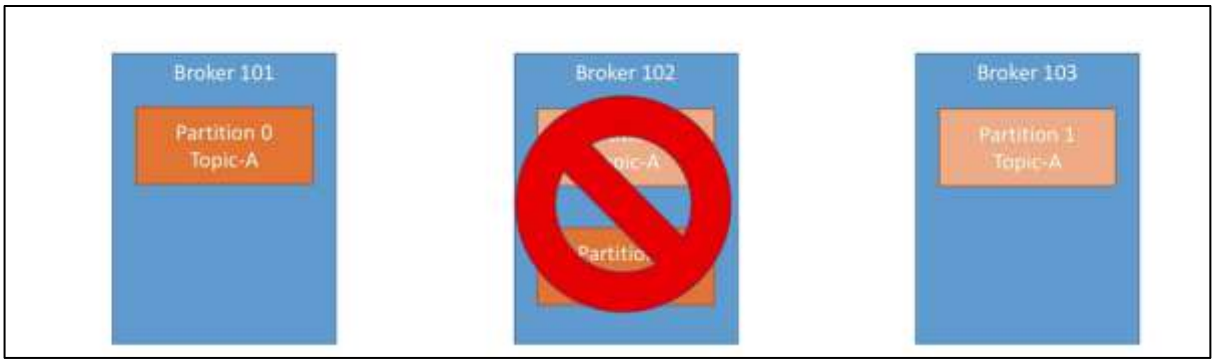
Let's say we have three brokers broker 101, 102, 103. and we are going to create a topic named *Topic_A*, and it has three partitions, and *Topic_B* with two partitions. (there is no relationship between partition number and broker number). Once a topic is created, Kafka will automatically assign the topic and distribute it across all of the brokers. In *Topic_B*, there is less partitions than the number of brokers so broker 103 will not hold any data from *Topic_B*

Topic Replication factor

Kafka is a distributed system. We have 3 brokers or 100 brokers, so this is distributed. When there is a distributed system in the big data world, we need to have replication, as if a machine goes down, then the things still work, and replication does that for us. When you create a topic you need to decide on the replication factor. Topics should have a replication factor greater than 1 (usually between 2 and 3). This way if a broker is down, another broker can serve the data. Let's take an example of *Topic-A* with 2 partitions and replication factor of 2. Because of replication factor, we need to see two replicas of these partitions somewhere (partition 1 of *Topic-A* is going to be replicated on broker 102 and partition 1 of *Topic -A* is also going to be replicated on Broker 103)

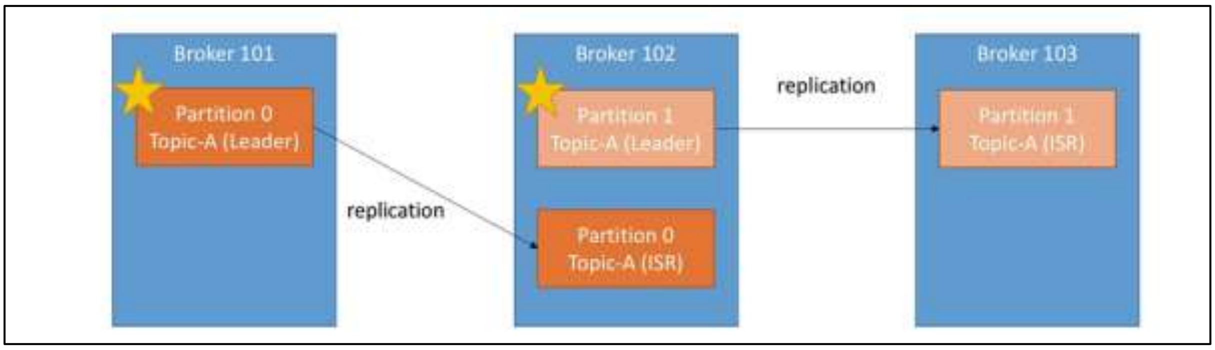


Let's say that we lose broker 2. Broker 101 and 103 can still serve the data. The replication basically allowed us to ensure that data would not be lost.



Concept of leader for a partition.

At any time only ONE broker can be a leader for a given partition. Only that leader can receive and serve data for a partition. The other brokers will synchronize the data. Therefore each partition has one leader and multiple ISR (in-sync replica) Let's take our previous example.

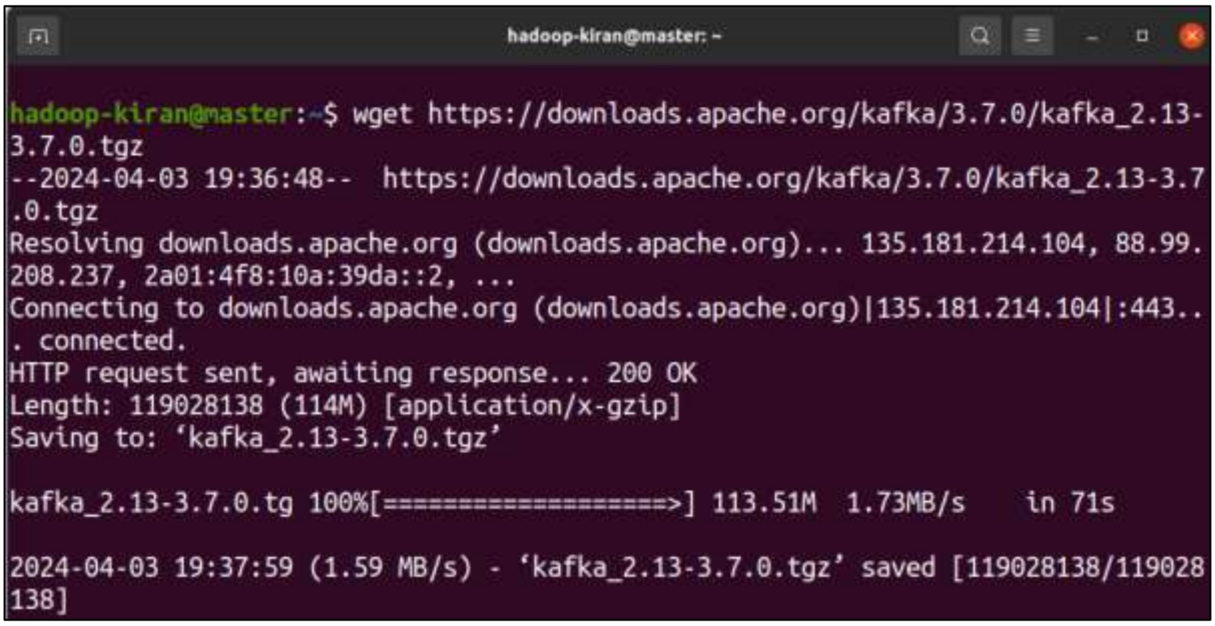


For partition 0 broker 101 is going to be the leader while broker 102 is going to be a replica or ISR. Similarly, for partition 102, broker 102 will be leader and broker 103 will be ISR. Zookeeper (will be discussed bellow) will decide leaders and ISRs. If broker 101 on the left is lost, then the partition 0 on broker 102 will become the leader because it was in-sync replica. When broker 101 comes back, it will try to become leader again after replicating the data. This is happening in the background, handled by Kafka.

Implementation:

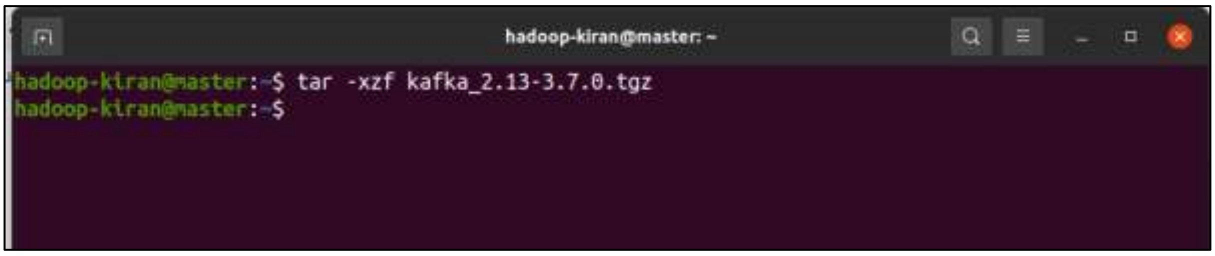
1. Download kafka

wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7.0.tgz.asc



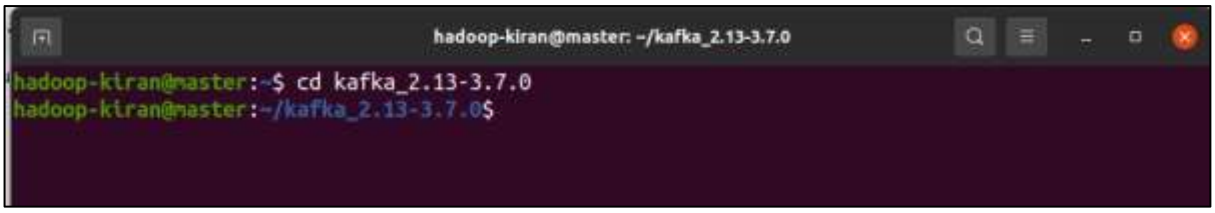
```
hadoop-kiran@master:~$ wget https://downloads.apache.org/kafka/3.7.0/kafka_2.13-
3.7.0.tgz
--2024-04-03 19:36:48--  https://downloads.apache.org/kafka/3.7.0/kafka_2.13-3.7
.0.tgz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.
208.237, 2a01:4f8:10a:39da::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443..
. connected.
HTTP request sent, awaiting response... 200 OK
Length: 119028138 (114M) [application/x-gzip]
Saving to: ‘kafka_2.13-3.7.0.tgz’

kafka_2.13-3.7.0.tg 100%[===================>] 113.51M  1.73MB/s    in 71s

2024-04-03 19:37:59 (1.59 MB/s) - ‘kafka_2.13-3.7.0.tgz’ saved [119028138/119028
138]
```

2. Extract the downloaded file



```
hadoop-kiran@master:~$ tar -xzf kafka_2.13-3.7.0.tgz
hadoop-kiran@master:~$
```

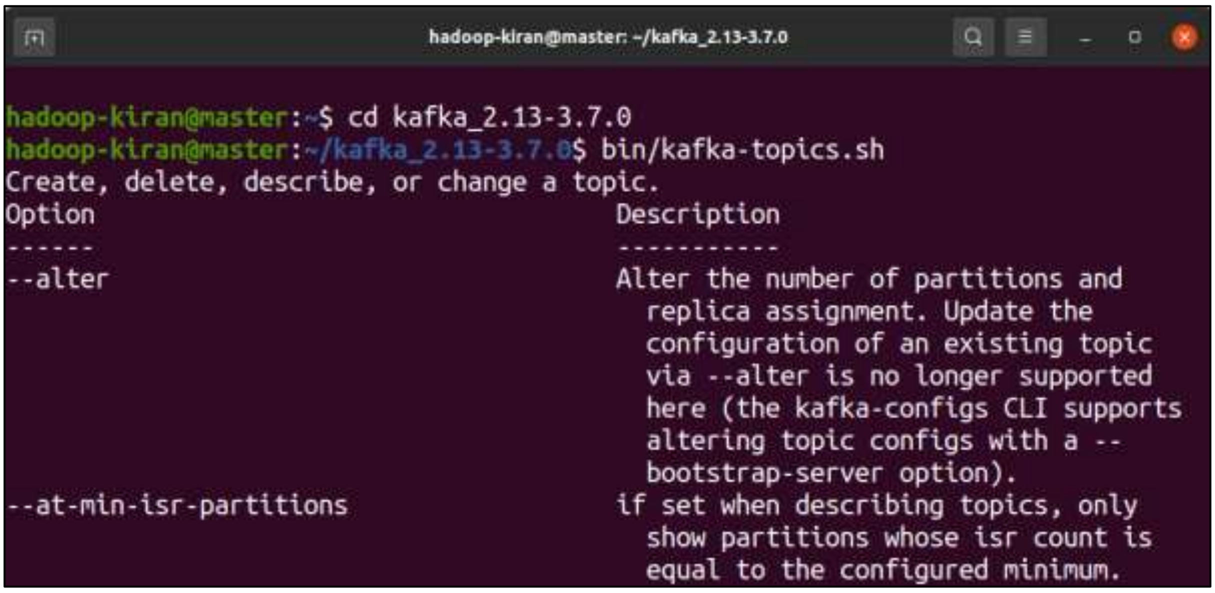
3. Move to kafka directory



```
hadoop-kiran@master:~$ cd kafka_2.13-3.7.0
hadoop-kiran@master:~/kafka_2.13-3.7.0$
```

4. Check the installation

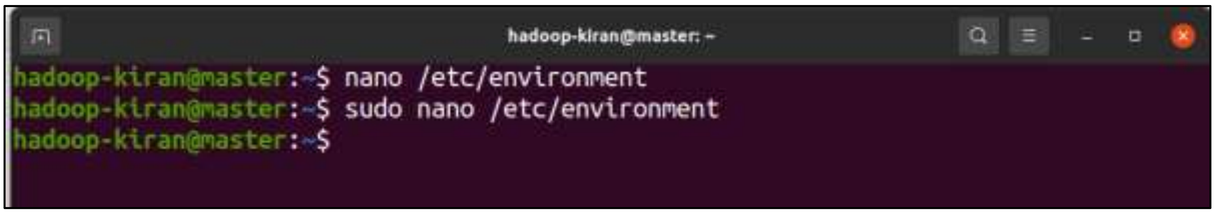
bin/kafka-topics.sh



hadoop-kiran@master: ~/kafka_2.13-3.7.0

```
hadoop-kiran@master:~$ cd kafka_2.13-3.7.0
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh
Create, delete, describe, or change a topic.
Option                                  Description
------                                  -----------
--alter                                 Alter the number of partitions and
                                          replica assignment. Update the
                                          configuration of an existing topic
                                          via --alter is no longer supported
                                          here (the kafka-configs CLI supports
                                          altering topic configs with a --
                                          bootstrap-server option).
--at-min-isr-partitions                 if set when describing topics, only
                                          show partitions whose isr count is
                                          equal to the configured minimum.
```

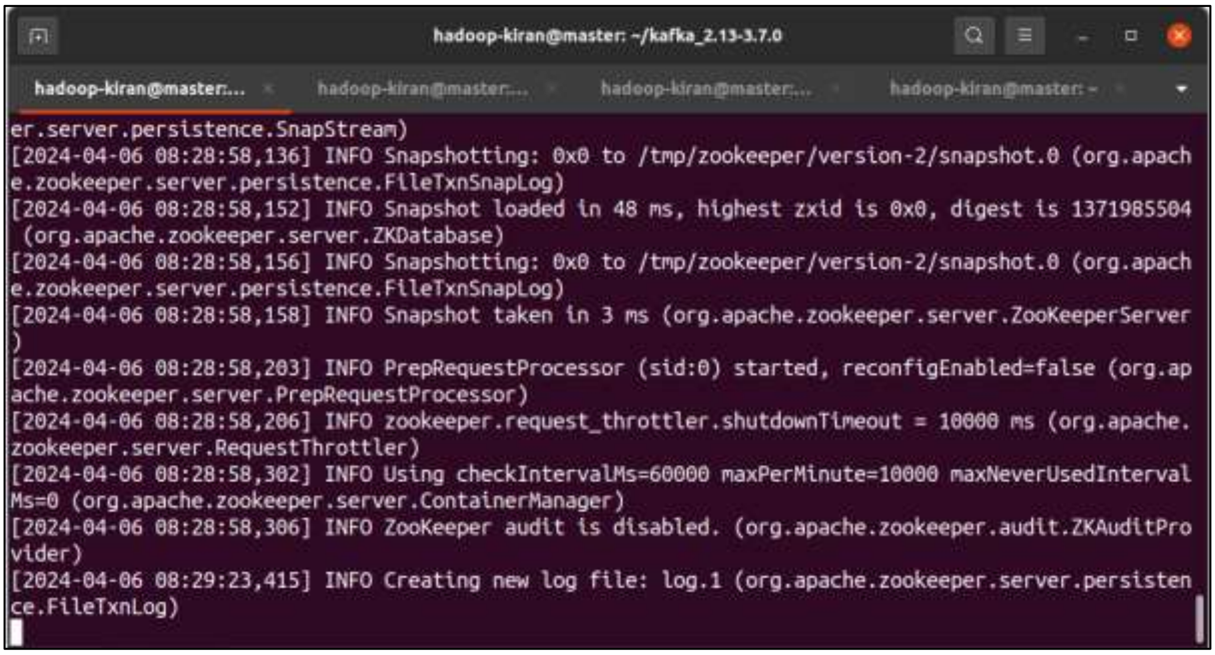
5. Add path to /etc/environment file



hadoop-kiran@master: ~

```
hadoop-kiran@master:~$ nano /etc/environment
hadoop-kiran@master:~$ sudo nano /etc/environment
hadoop-kiran@master:~$
```

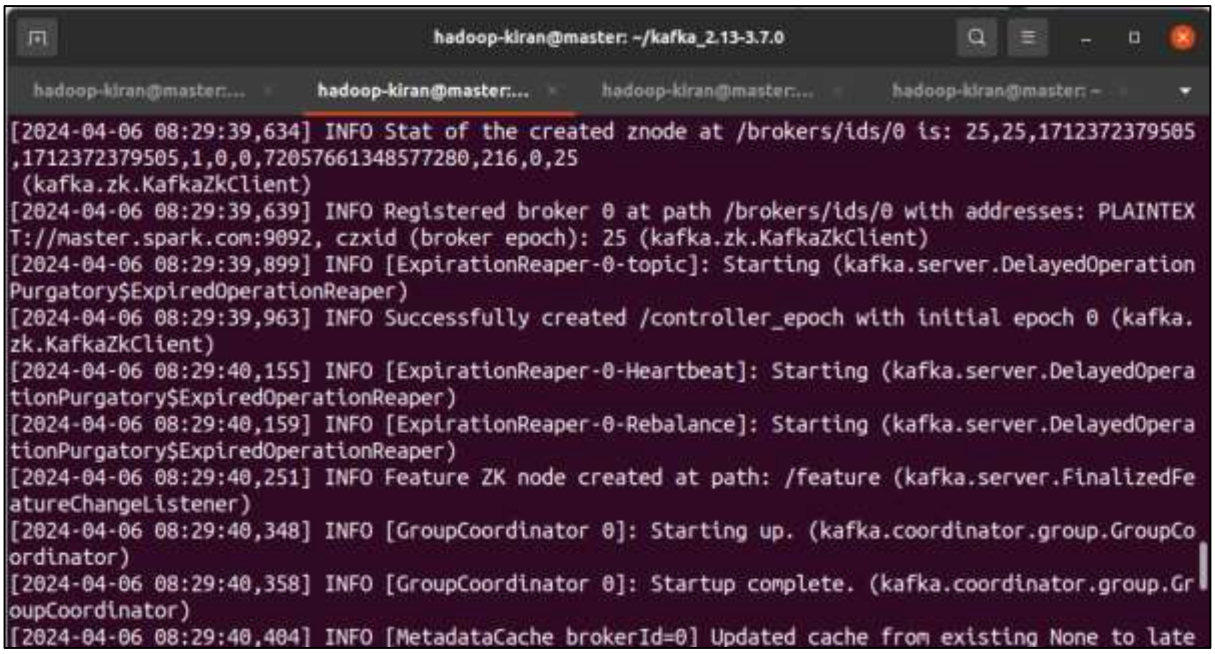
6. start zookeeper



hadoop-kiran@master: ~/kafka_2.13-3.7.0

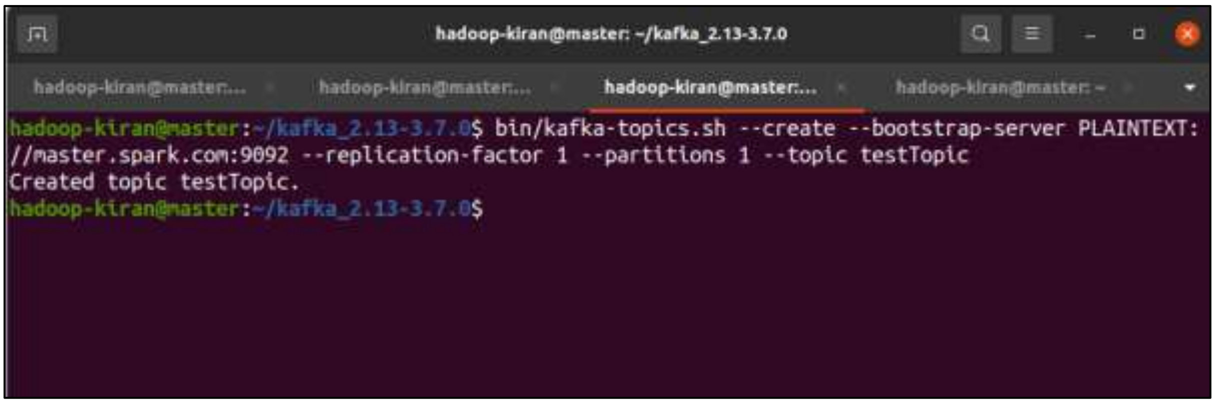
```
er.server.persistence.SnapStream)
[2024-04-06 08:28:58,136] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apach
e.zookeeper.server.persistence.FileTxnSnapLog)
[2024-04-06 08:28:58,152] INFO Snapshot loaded in 48 ms, highest zxid is 0x0, digest is 1371985504
 (org.apache.zookeeper.server.ZKDatabase)
[2024-04-06 08:28:58,156] INFO Snapshotting: 0x0 to /tmp/zookeeper/version-2/snapshot.0 (org.apach
e.zookeeper.server.persistence.FileTxnSnapLog)
[2024-04-06 08:28:58,158] INFO Snapshot taken in 3 ms (org.apache.zookeeper.server.ZooKeeperServer
)
[2024-04-06 08:28:58,203] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.ap
ache.zookeeper.server.PrepRequestProcessor)
[2024-04-06 08:28:58,206] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.
zookeeper.server.RequestThrottler)
[2024-04-06 08:28:58,302] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedInterval
Ms=0 (org.apache.zookeeper.server.ContainerManager)
[2024-04-06 08:28:58,306] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditPro
vider)
[2024-04-06 08:29:23,415] INFO Creating new log file: log.1 (org.apache.zookeeper.server.persisten
ce.FileTxnLog)
```

7. start kafka



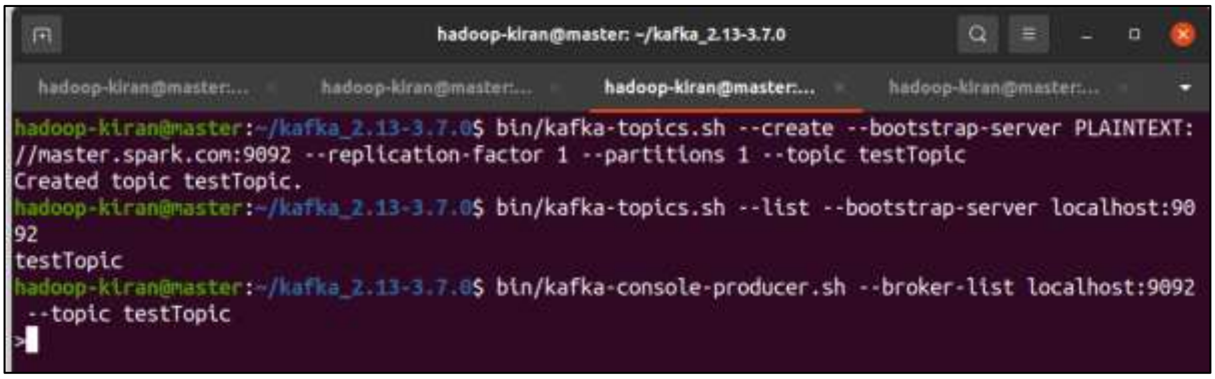
```
[2024-04-06 08:29:39,634] INFO Stat of the created znode at /brokers/ids/0 is: 25,25,1712372379505
,1712372379505,1,0,0,72057661348577280,216,0,25
 (kafka.zk.KafkaZkClient)
[2024-04-06 08:29:39,639] INFO Registered broker 0 at path /brokers/ids/0 with addresses: PLAINTEX
T://master.spark.com:9092, czxid (broker epoch): 25 (kafka.zk.KafkaZkClient)
[2024-04-06 08:29:39,899] INFO [ExpirationReaper-0-topic]: Starting (kafka.server.DelayedOperation
Purgatory$ExpiredOperationReaper)
[2024-04-06 08:29:39,963] INFO Successfully created /controller_epoch with initial epoch 0 (kafka.
zk.KafkaZkClient)
[2024-04-06 08:29:40,155] INFO [ExpirationReaper-0-Heartbeat]: Starting (kafka.server.DelayedOpera
tionPurgatory$ExpiredOperationReaper)
[2024-04-06 08:29:40,159] INFO [ExpirationReaper-0-Rebalance]: Starting (kafka.server.DelayedOpera
tionPurgatory$ExpiredOperationReaper)
[2024-04-06 08:29:40,251] INFO Feature ZK node created at path: /feature (kafka.server.FinalizedFe
atureChangeListener)
[2024-04-06 08:29:40,348] INFO [GroupCoordinator 0]: Starting up. (kafka.coordinator.group.GroupCo
ordinator)
[2024-04-06 08:29:40,358] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.group.Gr
oupCoordinator)
[2024-04-06 08:29:40,404] INFO [MetadataCache brokerId=0] Updated cache from existing None to late
```

8. start a new topic



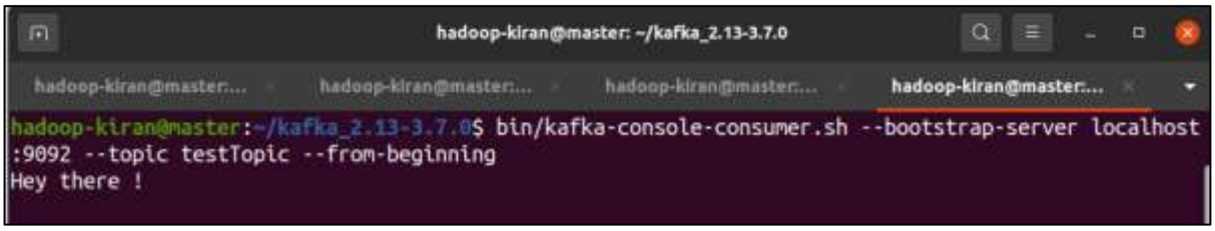
```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh --create --bootstrap-server PLAINTEXT:
//master.spark.com:9092 --replication-factor 1 --partitions 1 --topic testTopic
Created topic testTopic.
hadoop-kiran@master:~/kafka_2.13-3.7.0$
```

9. list the topics and start the producer



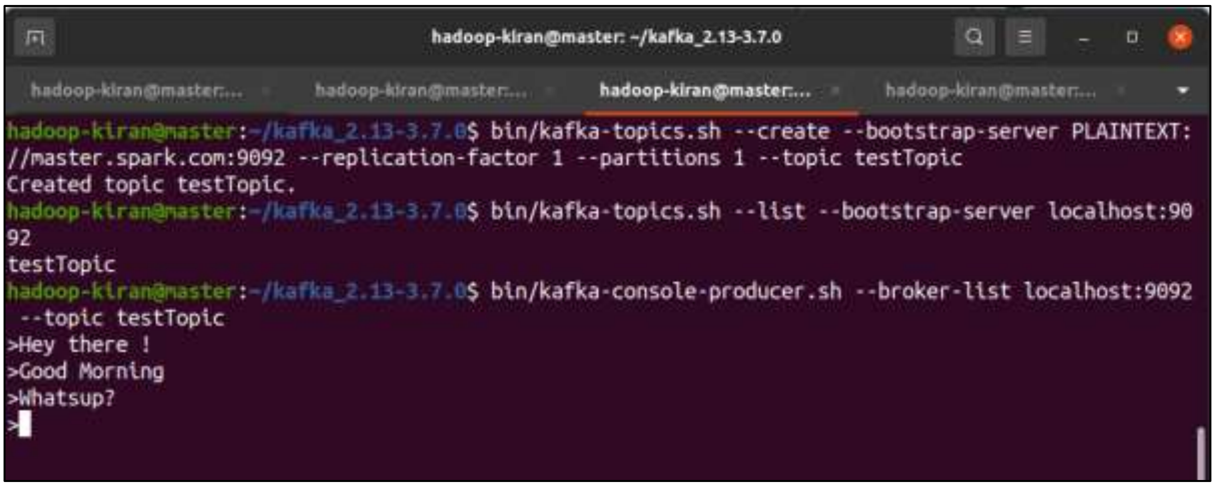
```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh --create --bootstrap-server PLAINTEXT:
//master.spark.com:9092 --replication-factor 1 --partitions 1 --topic testTopic
Created topic testTopic.
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh --list --bootstrap-server localhost:90
92
testTopic
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-console-producer.sh --broker-list localhost:9092
 --topic testTopic
>
```

10. start the Consumer



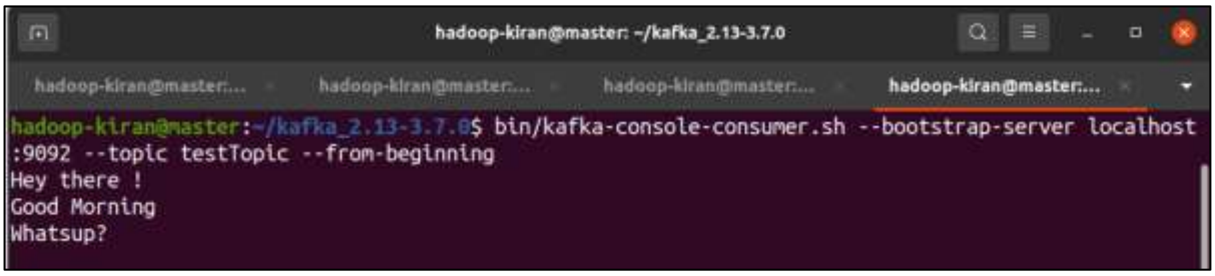
```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-console-consumer.sh --bootstrap-server localhost
:9092 --topic testTopic --from-beginning
Hey there !
```

11. start sending the message



```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh --create --bootstrap-server PLAINTEXT:
//master.spark.com:9092 --replication-factor 1 --partitions 1 --topic testTopic
Created topic testTopic.
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-topics.sh --list --bootstrap-server localhost:90
92
testTopic
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-console-producer.sh --broker-list localhost:9092
 --topic testTopic
>Hey there !
>Good Morning
>Whatsup?
>
```

Can verify the messages send by the sender are received at the consumer side



```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ bin/kafka-console-consumer.sh --bootstrap-server localhost
:9092 --topic testTopic --from-beginning
Hey there !
Good Morning
Whatsup?
```

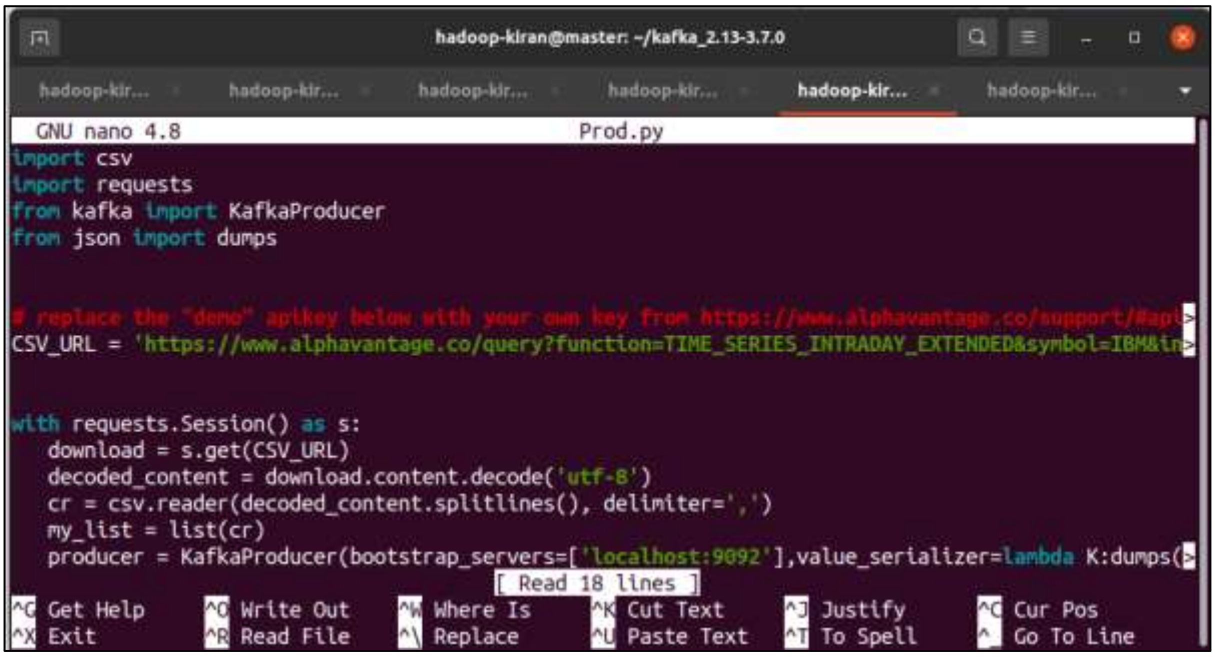
12. Streaming data with Producer Consumer from Python

Prod.py

```
import csv
import requests
from kafka import KafkaProducer
from json import dumps

with requests.Session() as s:
  with open('intraday_5min_IBM.csv', mode ='r')as file:
    cr = csv.reader(file)
```

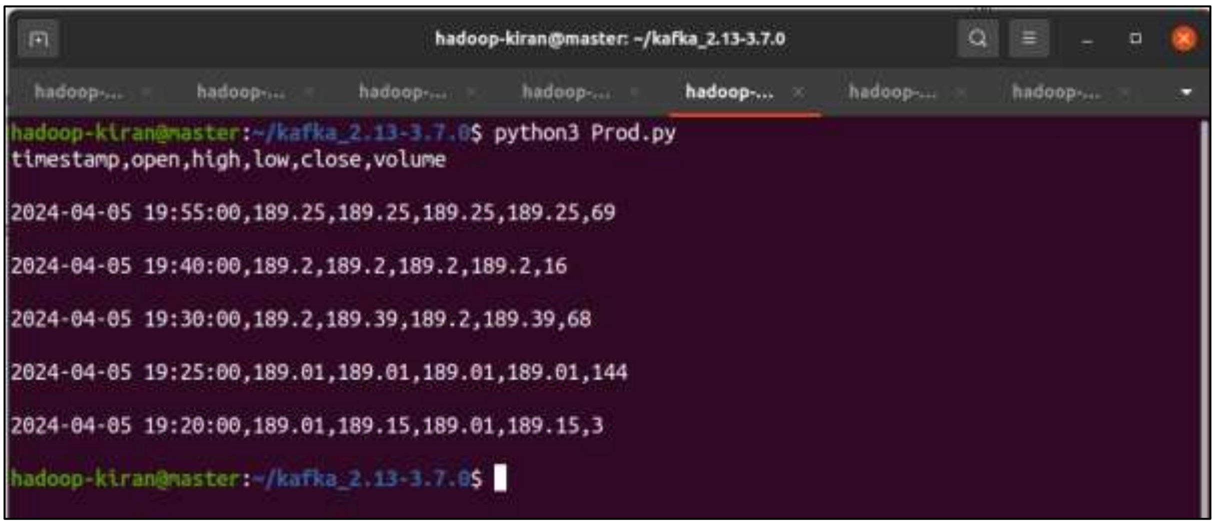
```
   my_list = list(cr)
   producer = KafkaProducer(bootstrap_servers=['master.spark.com:9092'],value_serializer=lambda
K:dumps(K).encode('utf-8'))


with open('intraday_5min_IBM.csv', mode ='r')as file:
  for row in file:
   producer.send('testTopic',row)
```

Prod.py



```
  GNU nano 4.8                                    Prod.py
import csv
import requests
from kafka import KafkaProducer
from json import dumps


# replace the "demo" apikey below with your own key from https://www.alphavantage.co/support/#api>
CSV_URL = 'https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY_EXTENDED&symbol=IBM&in>


with requests.Session() as s:
   download = s.get(CSV_URL)
   decoded_content = download.content.decode('utf-8')
   cr = csv.reader(decoded_content.splitlines(), delimiter=',')
   my_list = list(cr)
   producer = KafkaProducer(bootstrap_servers=['localhost:9092'],value_serializer=lambda K:dumps(>
                                        [ Read 18 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit         ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell     ^_ Go To Line
```

Executing Prod.py

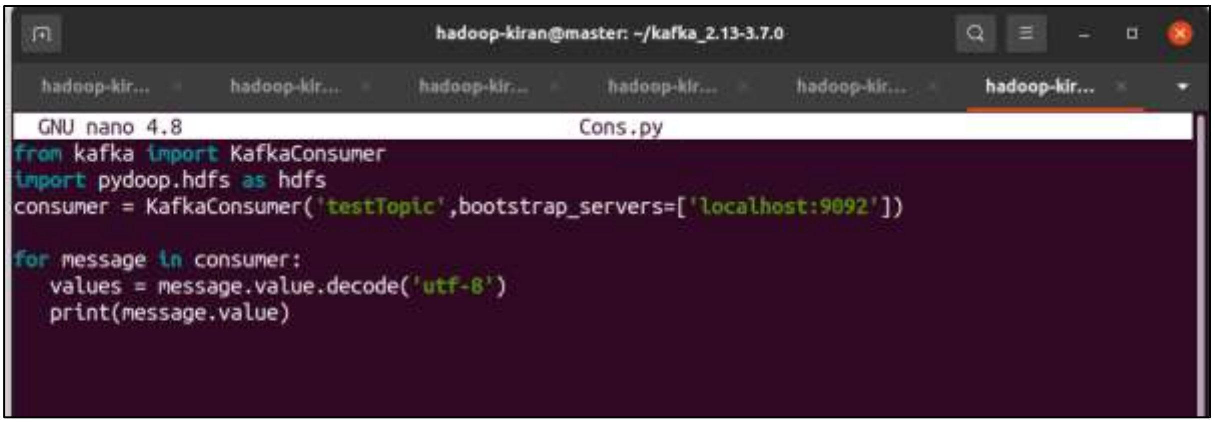


```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ python3 Prod.py
timestamp,open,high,low,close,volume

2024-04-05 19:55:00,189.25,189.25,189.25,189.25,69

2024-04-05 19:40:00,189.2,189.2,189.2,189.2,16

2024-04-05 19:30:00,189.2,189.39,189.2,189.39,68

2024-04-05 19:25:00,189.01,189.01,189.01,189.01,144

2024-04-05 19:20:00,189.01,189.15,189.01,189.15,3

hadoop-kiran@master:~/kafka_2.13-3.7.0$ 
```

Cons.py

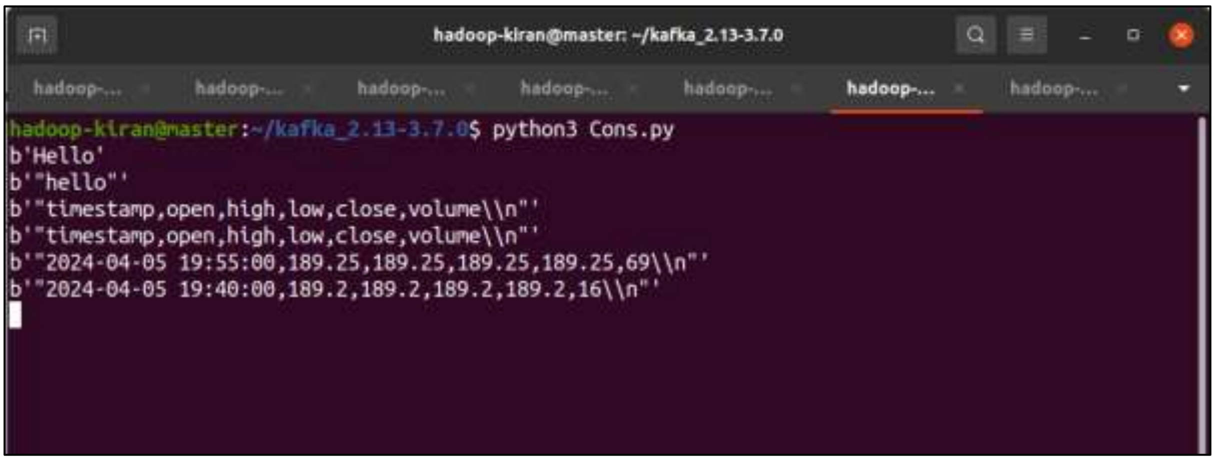
```
from kafka import KafkaConsumer

consumer = KafkaConsumer('testTopic',bootstrap_servers=['master.spark.com:9092'])


for message in consumer:
   values = message.value.decode('utf-8')
   print(message.value)
```

13. cons .py



```
 GNU nano 4.8                          Cons.py
from kafka import KafkaConsumer
import pydoop.hdfs as hdfs
consumer = KafkaConsumer('testTopic',bootstrap_servers=['localhost:9092'])

for message in consumer:
   values = message.value.decode('utf-8')
   print(message.value)
```

Executing Cons.py can verify streamed data to kafaka



```
hadoop-kiran@master:~/kafka_2.13-3.7.0$ python3 Cons.py
b'Hello'
b'"hello"'
b'"timestamp,open,high,low,close,volume\\n"'
b'"timestamp,open,high,low,close,volume\\n"'
b'"2024-04-05 19:55:00,189.25,189.25,189.25,189.25,69\\n"'
b'"2024-04-05 19:40:00,189.2,189.2,189.2,189.2,16\\n"'
```

Conclusion:

In conclusion, Apache Kafka is a powerful distributed streaming platform designed for handling real-time data feeds with high throughput, fault tolerance, and scalability. Thus we have setup and installed and streamed data into Apache Kafka.