

STEP: - 1

INSTALL REQUIRED TOOLS TO PROCEED WORK

```
sudo apt-get install openjdk-8-jdk
wget https://dlcdn.apache.org/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz
tar xvzf spark-3.1.2-bin-hadoop3.2.tgz
wget https://archive.apache.org/dist/kafka/2.0.0/kafka\_2.11-2.0.0.tgz
tar xvzf kafka_2.11-2.0.0.tgz
```

STEP: - 2

SET PATH

```
#JAVA PATH
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
set PATH="$PATH:JAVA_HOME/bin"
```

```
export HADOOP_HOME=/home/usr/hadoop-2.7.0
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
#spark path
export SPARK_HOME=/home/ubuntu/spark-3.1.2-bin-hadoop3.2
export PATH=$PATH:$SPARK_HOME/bin
```

STEP: - 3

FOLLOW THIS LINK INSTALL JUPYTER NOTEBOOK ON UBUNTU

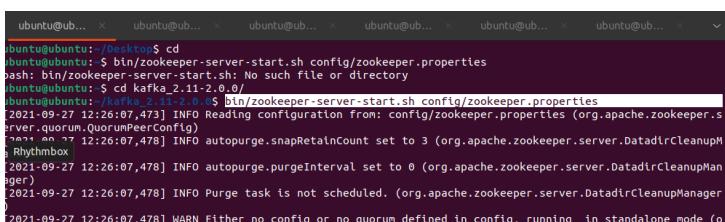
<https://www.digitalocean.com/community/tutorials/how-to-set-up-jupyter-notebook-with-python-3-on-ubuntu-20-04-and-connect-via-ssh-tunneling>

STEP: - 4

GOTO KAFKA DIRECTORY AND RUN ZOOKEEPER AND KAFKA SERVER

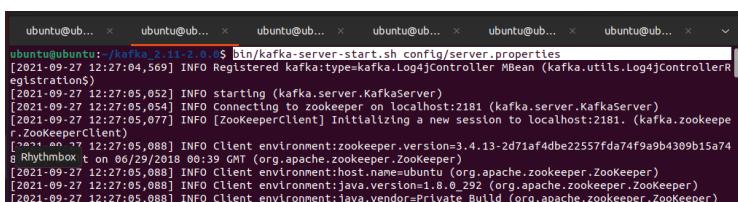
```
cd kafka_2.11-2.0.0/
```

```
bin/zookeeper-server-start.sh config/zookeeper.properties
```



```
ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub...
ubuntu@ubuntu: ~/Desktop$ cd
ubuntu@ubuntu: $ bin/zookeeper-server-start.sh config/zookeeper.properties
bash: bin/zookeeper-server-start.sh: No such file or directory
ubuntu@ubuntu: $ cd kafka_2.11-2.0.0/
ubuntu@ubuntu: ~/kafka_2.11-2.0.0$ bin/zookeeper-server-start.sh config/zookeeper.properties
[2021-09-27 12:26:07,473] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.ZooKeeperQuorumPeerConfig)
[2021-09-27 12:26:07,478] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-09-27 12:26:07,478] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-09-27 12:26:07,478] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)
[2021-09-27 12:26:07,478] WARN Either no config or no quorum defined in config, running in standalone mode (o
```

```
bin/kafka-server-start.sh config/server.properties
```



```
ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub... ~ ubuntu@ub...
ubuntu@ubuntu: ~/kafka_2.11-2.0.0$ bin/kafka-server-start.sh config/server.properties
[2021-09-27 12:27:05,569] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration)
[2021-09-27 12:27:05,562] INFO starting (kafka.server.KafkaServer)
[2021-09-27 12:27:05,564] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2021-09-27 12:27:05,577] INFO [ZooKeeperClient] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)
[2021-09-27 12:27:05,588] INFO Client environment:zookeeper.version=3.4.13-2df1af4dbe22557fd74f9a9b4309b15a748 Rhythmbox t on 06/29/2018 00:39 GMT (org.apache.zookeeper.ZooKeeper)
[2021-09-27 12:27:05,588] INFO Client environment:host.name=ubuntu (org.apache.zookeeper.ZooKeeper)
[2021-09-27 12:27:05,588] INFO Client environment:java.version=1.8.0_292 (org.apache.zookeeper.ZooKeeper)
[2021-09-27 12:27:05,588] INFO Client environment:java.vendor=Private Build (org.apache.zookeeper.ZooKeeper)
```

STEP: - 5

CREATE A TOPIC project3 WITH REPLICATION FACTOR 1 AND PARTITION 1

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic project3 --replication-factor 1 --partitions 1
```

STEP: - 6

CREATE A producer.py FILE TO FETCH DATA FROM API AND PASS TO TOPIC project3

1. CREATE ACCOUNT ON api.tiingo.com

<https://api.tiingo.com/>

2. GOTO DOCUMENTATION SECTION AND COLLECT YOUR API KEY

The screenshot shows the Tiingo API documentation website. The top navigation bar includes links for Home, Documentation (which is highlighted), Products, About, and Pricing. Below the navigation, there's a note about navigating using the sidebar and a link to third-party packages. The main content area is titled '1.1 GENERAL - OVERVIEW' and '1.1.2 Authentication'. It explains the need for an account and provides instructions for finding an API token. A code block shows an example token: '14b8a64181f2b1007a7ef97847216c6d5ffa16cf'. A note at the bottom says to see how to use the token for requests.

3. COLLECT FOREX DATA API FROM REST SECTION

The screenshot shows the '1. General' section of the Tiingo API documentation. The left sidebar has a '1. General' category with sub-links: 1.1 Overview, 1.2 Connecting, and 1.3 Changelog. Below it is a '2. REST' category with sub-links: 2.1 End-of-Day, 2.2 News, 2.3 Crypto, 2.4 Forex, 2.5 IEX, and 2.6 Fundamentals. The main content area is titled '1.1 GENERAL - OVERVIEW' and '1.1.1 Introduction'. It discusses the performance and consistency of Tiingo's APIs and introduces the two types of endpoints: RESTful and Websocket.

4. SCROLL DOWN AND COLLECT YOUR API FROM EXAMPLES SECTION

The screenshot shows the Tiingo API documentation page. The left sidebar has sections like 1. General, 1.1 Overview, 1.2 Connecting, 1.3 Changelog, and 2. REST. The main content area is titled "To request top-of-book and last price data for a stock, use the following REST endpoints." It includes code snippets for multiple base and quote pairs and specific tickers. Below this, there are tabs for Response, Request, and Examples. Under Examples, there are tabs for Python, Node, and PHP. A code example in Python is shown:

```

import requests
headers = {
    'Content-Type': 'application/json'
}
requestResponse = requests.get("https://api.tiingo.com/tiingo/fx/top?tickers=audusd,eurusd&token=14b8af4181fb1007a/ef07847216c6d5ffa16cf", headers=headers)
print(requestResponse.json())

```

The Response tab shows JSON data for two tickers: audusd and eurusd. The audusd entry is:

```

{
  "ticker": "audusd",
  "quoteTimestamp": "2019-07-01T21:00:01.289000+00:00",
  "bidPrice": 0.6963,
  "bidSize": 1000000.0,
  "askPrice": 0.69645,
  "askSize": 1200000.0,
  "midPrice": 0.696375
},
{
  "ticker": "eurusd",
  "quoteTimestamp": "2019-07-01T21:00:01.181000+00:00",
  "bidPrice": 1.12849,
  "bidSize": 1000000.0
}

```

5. producer.py

#This file is going to be used for producer

#first create a topic name as 'project3'

from kafka import KafkaProducer

import requests

from json import dumps

import time

kafka_data_producers = KafkaProducer(bootstrap_servers=['localhost:9092'], value_serializer=lambda x: dumps(x).encode('utf-8'))

while True:

 response_data = requests.get("https://api.tiingo.com/tiingo/fx/top?tickers=audusd,eurusd&token=<PASS UR API KEY HERE>")

 #response_data=response_data.json()

 data = {'Lagos' : response_data.json()}

 data=data['Lagos'][0]

 kafka_data_producers.send('project3', value=data)

 print(data)

 print()

 time.sleep(10)

6. RUN producer.py file

```

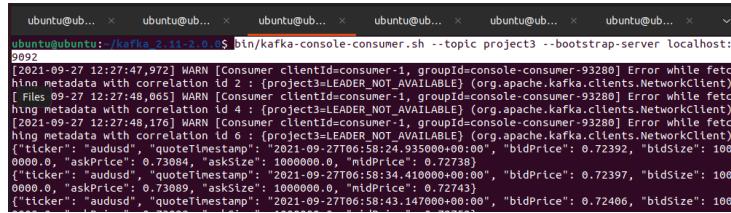
ubuntu@ub... ~ % cd Desktop
ubuntu@ub... $ python3 producer.py
[{'ticker': 'audusd', 'quoteTimestamp': '2021-09-27T06:58:24.935000+00:00', 'bidPrice': 0.72392, 'bidSize': 1000000.0, 'askPrice': 0.73084, 'askSize': 1000000.0, 'midPrice': 0.72738}
[{'ticker': 'audusd', 'quoteTimestamp': '2021-09-27T06:58:34.410000+00:00', 'bidPrice': 0.72397, 'bidSize': 1000000.0, 'askPrice': 0.73089, 'askSize': 1000000.0, 'midPrice': 0.72743}
[{'ticker': 'audusd', 'quoteTimestamp': '2021-09-27T06:58:43.147000+00:00', 'bidPrice': 0.72406, 'bidSize': 1000000.0, 'askPrice': 0.73098, 'askSize': 1000000.0, 'midPrice': 0.72752}

```

STEP: - 7

START CONSUMER API IN NEW TERMINAL

```
bin/kafka-console-consumer.sh --topic project3 --bootstrap-server localhost:9092
```



A terminal window showing the output of the command 'bin/kafka-console-consumer.sh --topic project3 --bootstrap-server localhost:9092'. The logs are from 2021-09-27 12:27:47.9721, indicating errors while fetching metadata for consumer group 'project3-LEADER' due to it being unavailable. It also shows some successful fetches for the 'audusd' ticker.

```
ubuntu@ub... ~ % ubuntu@ub... ~ %
ubuntu@ubuntu:~/kafka_2.5.1-2.0.1$ bin/kafka-console-consumer.sh --topic project3 --bootstrap-server localhost:9092
[2021-09-27 12:27:47.9721] WARN [Consumer clientId=consumer-1, groupId=console-consumer-93280] Error while fetching metadata with correlationId 1 : [project3-LEADER,NOT AVAILABLE] (org.apache.kafka.clients.NetworkClient)
[2021-09-27 12:27:48.005] WARN [Consumer clientId=consumer-1, groupId=console-consumer-93280] Error while fetching metadata with correlationId 4 : [project3-LEADER,NOT AVAILABLE] (org.apache.kafka.clients.NetworkClient)
[2021-09-27 12:27:48.176] WARN [Consumer clientId=consumer-1, groupId=console-consumer-93280] Error while fetching metadata with correlationId 6 : [project3-LEADER,NOT AVAILABLE] (org.apache.kafka.clients.NetworkClient)
[2021-09-27 12:27:48.176] WARN [Consumer clientId=consumer-1, groupId=console-consumer-93280] Error while fetching metadata with correlationId 6 : [project3-LEADER,NOT AVAILABLE] (org.apache.kafka.clients.NetworkClient)
[{"ticker": "audusd", "quoteTimestamp": "2021-09-27T06:58:24.935000+00:00", "bidPrice": 0.72392, "bidSize": 100
0000.0, "askPrice": 0.73084, "askSize": 1000000.0, "midPrice": 0.72738}
[{"ticker": "audusd", "quoteTimestamp": "2021-09-27T06:58:34.410000+00:00", "bidPrice": 0.72397, "bidSize": 100
0000.0, "askPrice": 0.73089, "askSize": 1000000.0, "midPrice": 0.72743}
[{"ticker": "audusd", "quoteTimestamp": "2021-09-27T06:58:43.147000+00:00", "bidPrice": 0.72406, "bidSize": 100
0000.0, "askPrice": 0.73094, "askSize": 1000000.0, "midPrice": 0.72747}
```

STEP: - 8

CREATE ANOTHER .py FILE TO STRUCTURE THE JSON DATA INTO SPARK DATAFRAME AND DO SOME OPERATION OR A USECASE

1. CREATE TOPIC askPriceOutput

```
bin/kafka-console-consumer.sh --topic askPriceOutput --bootstrap-server localhost:9092
```

2. usecase2.py

```
# Import Required Libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
# Created spark session
spark = SparkSession \
    .builder \
    .appName("activeCitiesInUs") \
    .getOrCreate()

# Created kafka consumer using spark readStream
raw_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "project3") \
    .option("startingOffsets", "latest") \
    .load() \
    .selectExpr("CAST(value AS STRING)")

# Created Schema for Structured Streaming
schema = StructType([
    StructField("ticker", StringType()),
    StructField("quoteTimestamp", StringType()),
    StructField("bidPrice", FloatType()),
    StructField("bidSize", FloatType()),
    StructField("askPrice", FloatType()),
    StructField("askSize", FloatType()),
    StructField("midPrice", FloatType())
])
schema_df = raw_df.select(from_json(raw_df.value, schema).alias("data"))
output_df = schema_df.select(to_json(struct(col("data.quoteTimestamp"), col("data.askPrice"))).alias("value"))
# Sending the data to kafka broker
```

```

query = output_df.writeStream.format("kafka").option("kafka.bootstrap.servers",
"localhost:9092").option("checkpointLocation", "/tmp/checkpoint1").option("topic", "askPriceOutput").start()

# Waits for the termination signal from user.outputMode("complete")
query.awaitTermination()

```

STEP: - 9

1. GOTO SPARK BIN FOLDER

```
cd /spark-3.1.2-bin-hadoop3.2/bin/
```

2. RUN SPARK-SUBMIT JOB WITH usecase.py FILE AND STORE THE OUTPUT ON askPriceOutput TOPIC

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 /home/ubuntu/Desktop/usecase2.py
```

```
b9-f26bb6783ff7-921269974-driver-0-1, groupId=spark-kafka-source-48ae90fa-a237-4499-9db9-f26bb6783ff7-921269974-driver-0] Resetting offset for partition project3-0 to offset 74.
^ Terminal
[1]+ Stopped                  spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.1.2 /home/u
buntu/Desktop/usecase2.py
ubuntu@ubuntu:~/spark-3.1.2-bin-hadoop3.2/bin$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2
.12:3.1.2 /home/ubuntu/Desktop/usecase2.py
```

STEP: - 10

OUTPUT WILL BE SHOWING ON askPriceOutput TOPIC'S CONSUMER

```
ubuntu@ubuntu: ~ cd kafka_2.11-2.0.0/
ubuntu@ubuntu: ~/kafka_2.11-2.0.0$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --topic askPriceOutput --replication-factor 1 --partitions 1
Created topic "askPriceOutput".
ubuntu@ubuntu: ~/kafka_2.11-2.0.0$ bin/kafka-console-consumer.sh --topic askPriceOutput --bootstrap-server localhost:9092
{"quoteTimestamp":"2021-09-27T07:02:15.849000+00:00", "askPrice":0.73064}
{"quoteTimestamp":"2021-09-27T07:02:20.134000+00:00", "askPrice":0.73065}
{"quoteTimestamp":"2021-09-27T07:02:40.815000+00:00", "askPrice":0.73078}
{"quoteTimestamp":"2021-09-27T07:02:51.557000+00:00", "askPrice":0.73084}
{"quoteTimestamp":"2021-09-27T07:02:58.997000+00:00", "askPrice":0.73076}
{"quoteTimestamp":"2021-09-27T07:03:07.266000+00:00", "askPrice":0.73075}
{"quoteTimestamp":"2021-09-27T07:03:21.222000+00:00", "askPrice":0.73076}
{"quoteTimestamp":"2021-09-27T07:03:30.539000+00:00", "askPrice":0.73086}
{"quoteTimestamp":"2021-09-27T07:03:43.799000+00:00", "askPrice":0.73082}
{"quoteTimestamp":"2021-09-27T07:03:55.890000+00:00", "askPrice":0.73094}
{"quoteTimestamp":"2021-09-27T07:04:00.436000+00:00", "askPrice":0.73084}
{"quoteTimestamp":"2021-09-27T07:04:06.772000+00:00", "askPrice":0.73085}
{"quoteTimestamp":"2021-09-27T07:04:16.894000+00:00", "askPrice":0.73087}
{"quoteTimestamp":"2021-09-27T07:04:29.302000+00:00", "askPrice":0.73084}
{"quoteTimestamp":"2021-09-27T07:04:35.078000+00:00", "askPrice":0.73077}
{"quoteTimestamp":"2021-09-27T07:29:39.559000+00:00", "askPrice":0.73129}
{"quoteTimestamp":"2021-09-27T07:29:52.031000+00:00", "askPrice":0.73132}
```

STEP: - 11

OPEN JUPYTER NOTEBOOK AND INSTALL matplotlib library and kafka-python ON IT

```
import findspark
findspark.init()
```

```
from kafka import KafkaConsumer
import matplotlib.pyplot as plt
```

```
import json
import threading

consumer = KafkaConsumer('askPriceOutput', group_id='askPriceOutput',
bootstrap_servers=['localhost:9092'],
)
print("consumer started ...")

x = {}

def plot():
    global x
    for message in consumer:
        x[json.loads((message.value).decode("utf-8"))["quoteTimestamp"]] =
            json.loads((message.value).decode("utf-8"))["askPrice"]

plot_thread = threading.Thread(target=plot)

plot_thread.start()

try:
    fig = plt.figure(figsize = (12, 6))
    x = dict(sorted(x.items(), key=lambda item: item[1], reverse=True))
    plt.plot([*x.keys()][:10], [*x.values()][:10])
    plt.xlabel("time stamp", fontsize=15)
    plt.ylabel("Ask Price", fontsize=15)
    plt.title("Ask price with time", fontsize=15)
    plt.xticks(rotation = 90)
    plt.show()
except:
    print(f"atleast 10 data needed but {len(x)} data is there")
```

```
In [2]: import findspark  
findspark.init()
```

```
In [3]: #pip install kafka-python
```

```
In [4]: from kafka import KafkaConsumer  
import matplotlib.pyplot as plt  
import json  
import threading
```

```
In [5]: consumer = KafkaConsumer('askPriceOutput',  
                                group_id='askPriceOutput',  
                                bootstrap_servers=['localhost:9092'],  
                                )  
print("consumer started ...")  
consumer started ...
```

```
In [6]: x = {}
```

```
In [7]: def plot():  
    global x  
    for message in consumer:  
        x[json.loads((message.value).decode("utf-8"))["quoteTimestamp"]] = json.loads((message
```

```
In [8]: plot_thread = threading.Thread(target=plot)
```

```
In [9]: plot_thread.start()
```

```
In [14]: try:  
    fig = plt.figure(figsize = (12, 6))  
    x = dict(sorted(x.items(), key=lambda item: item[1], reverse=True))  
  
    plt.plot([*x.keys()[:10], [*x.values()[:10]]]  
  
    plt.xlabel("time stamp", fontsize=15)  
    plt.ylabel("Ask Price", fontsize=15)  
    plt.title("Ask price with time", fontsize=15)  
    plt.xticks(rotation = 90)  
    plt.show()
```

