**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**

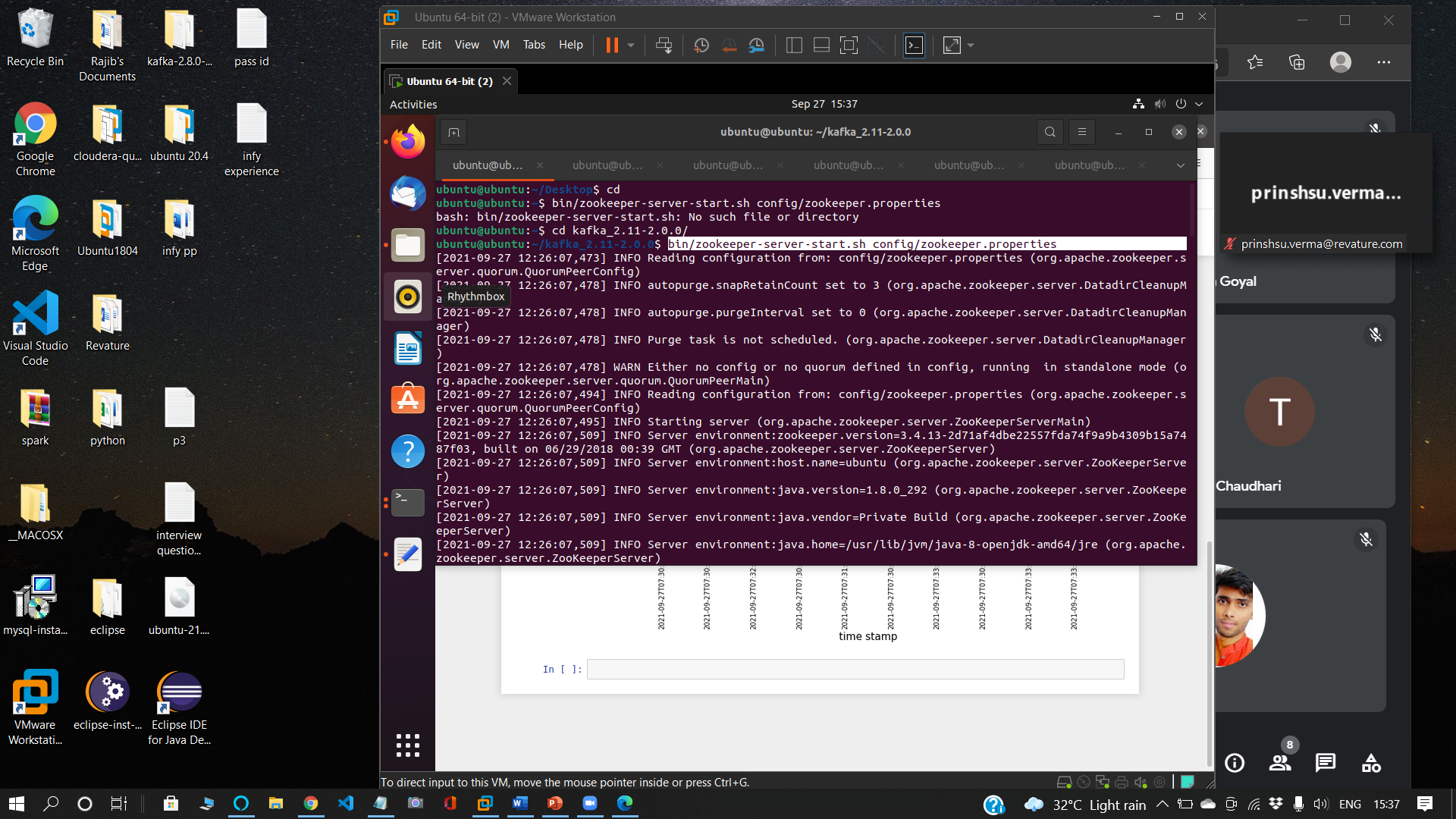
<https://www.digitalocean.com/community/tutorials/how-to-set-up-jupyter-notebook-with-python-3-on-ubuntu-20-04-and-connect-via-ssh-tunneling>

**STEP: - 4**

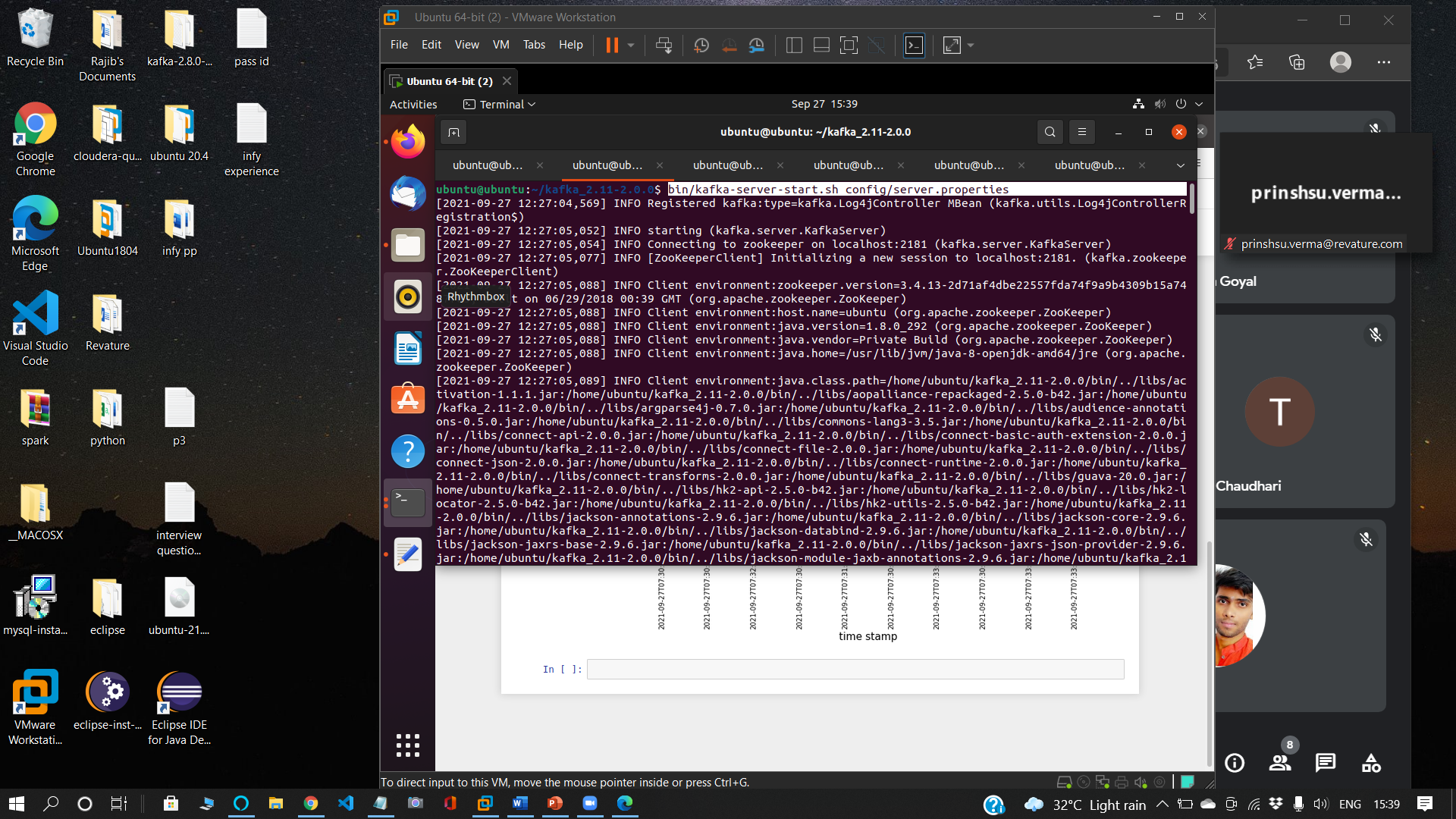
**GOTO KAFKA DIRECTORY AND RUN ZOOKEEPER AND KAFKA SERVER**

cd kafka\_2.11-2.0.0/

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



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



**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

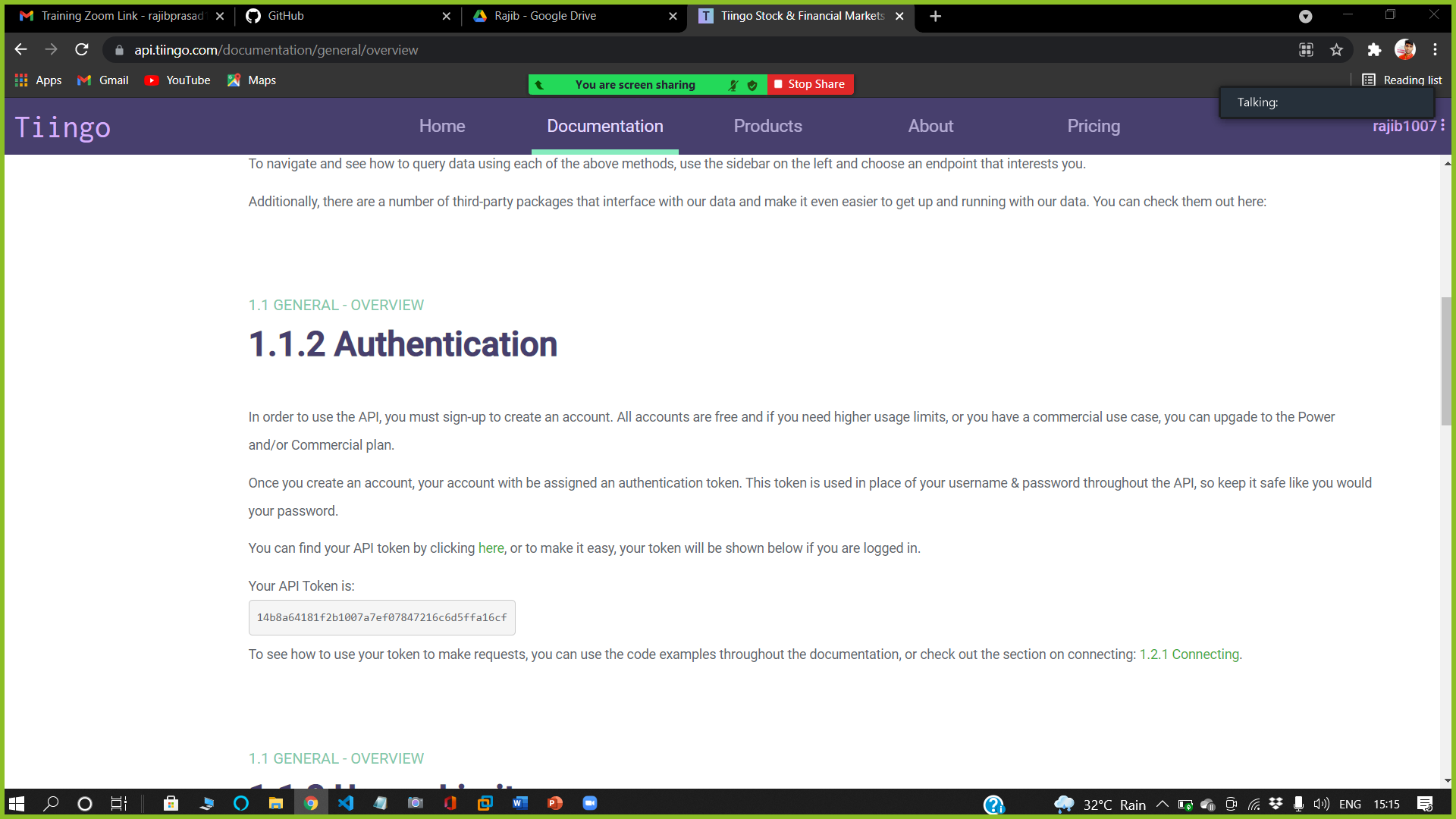
**STEP: - 6**

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

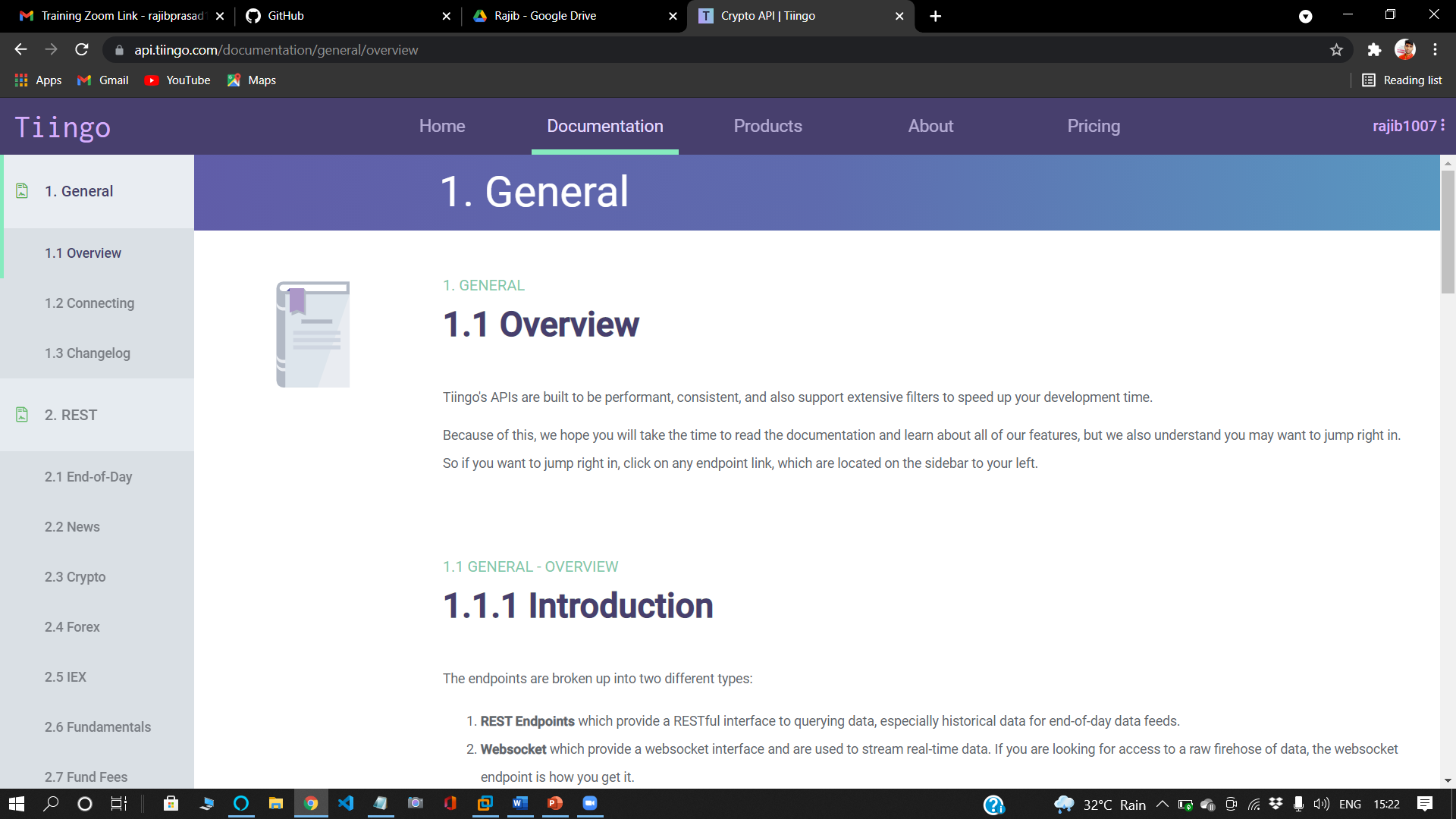
1. **CREATE ACCOUNT ON api.tiingo.com**

[**https://api.tiingo.com/**](https://api.tiingo.com/)

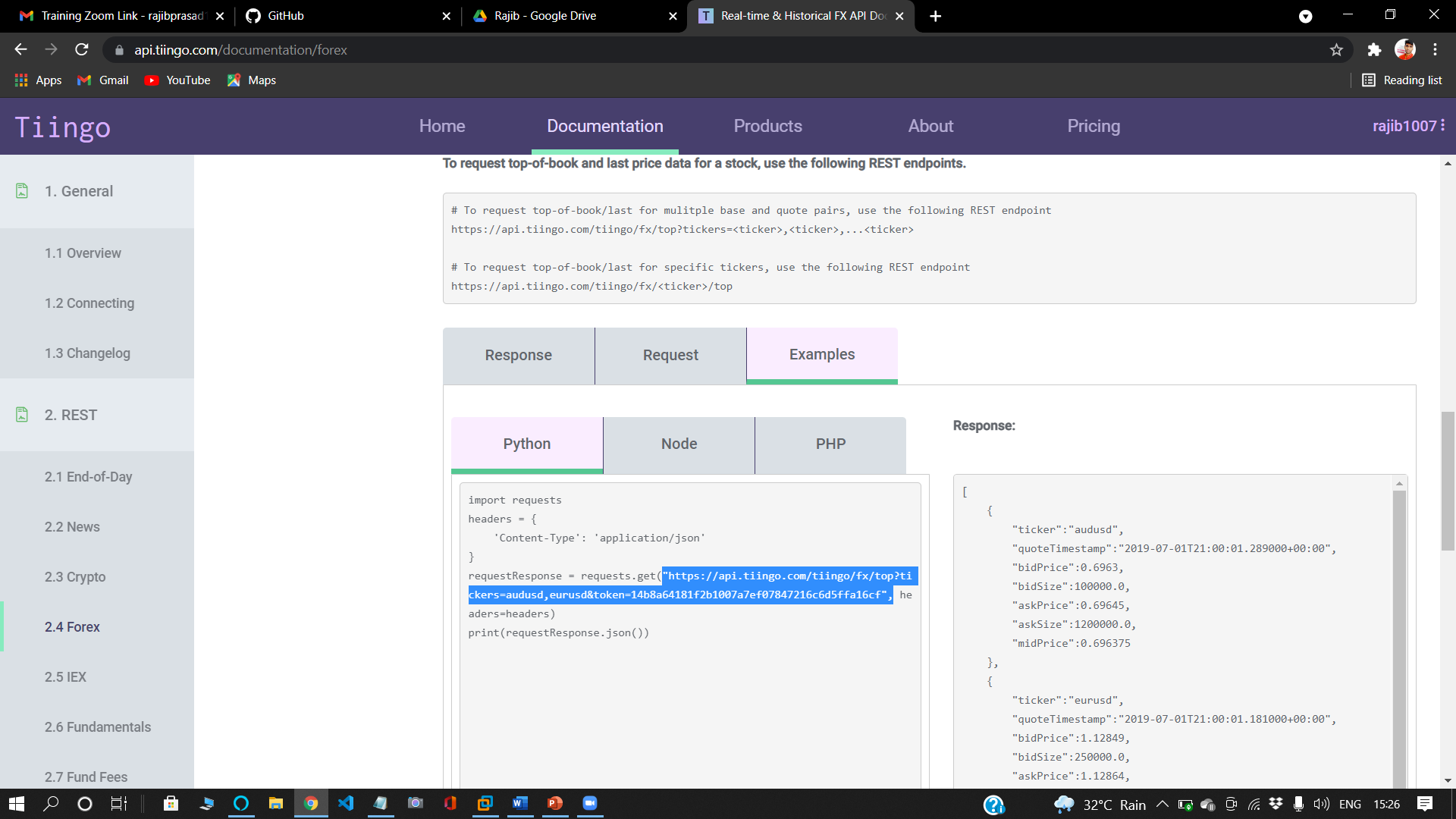
1. **GOTO DOCUMENTATION SECTION AND COLLECT YOUR API KEY**



1. **COLLECT FOREX DATA API FROM REST SECTION**



1. **SCROLL DOWN AND COLLECT YOUR API FROM EXAMPLES SECTION**



1. **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]

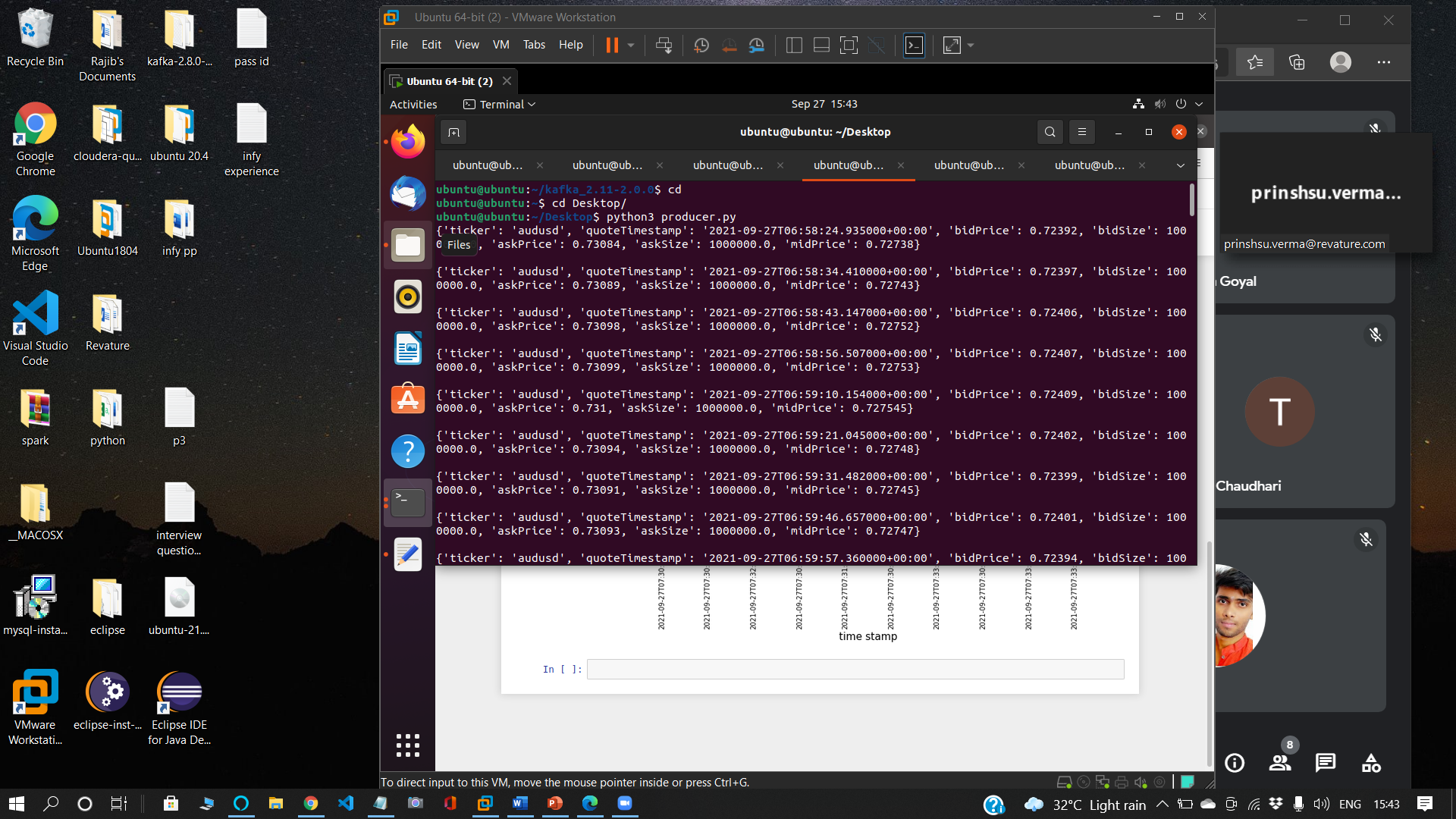
kafka\_data\_producers.send('project3', value=data)

print(data)

print()

time.sleep(10)

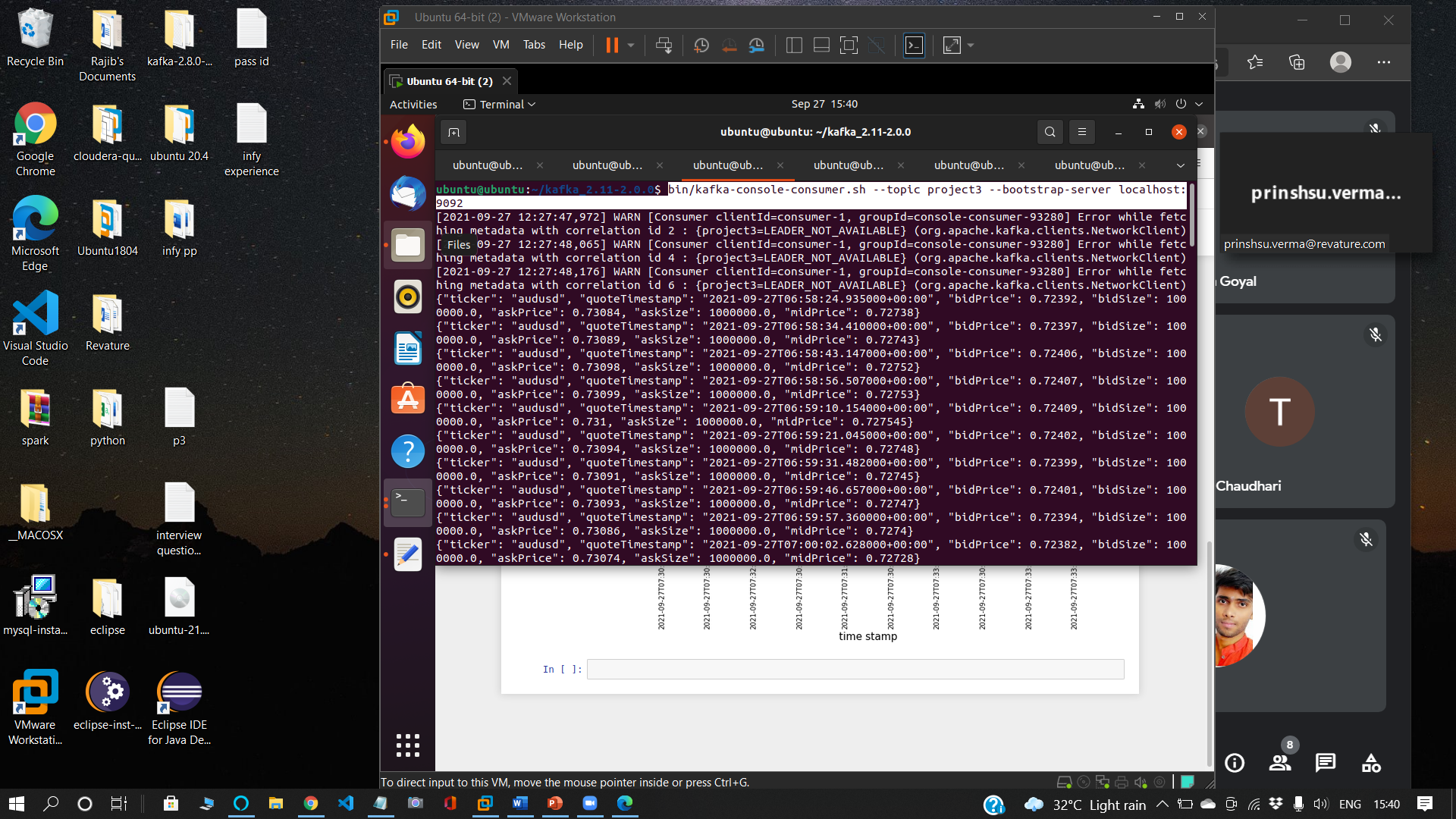
1. **RUN producer.py file**



**STEP: - 7**

**START CONSUMER API IN NEW TERMINAL**

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



**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

1. **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 brocker

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()

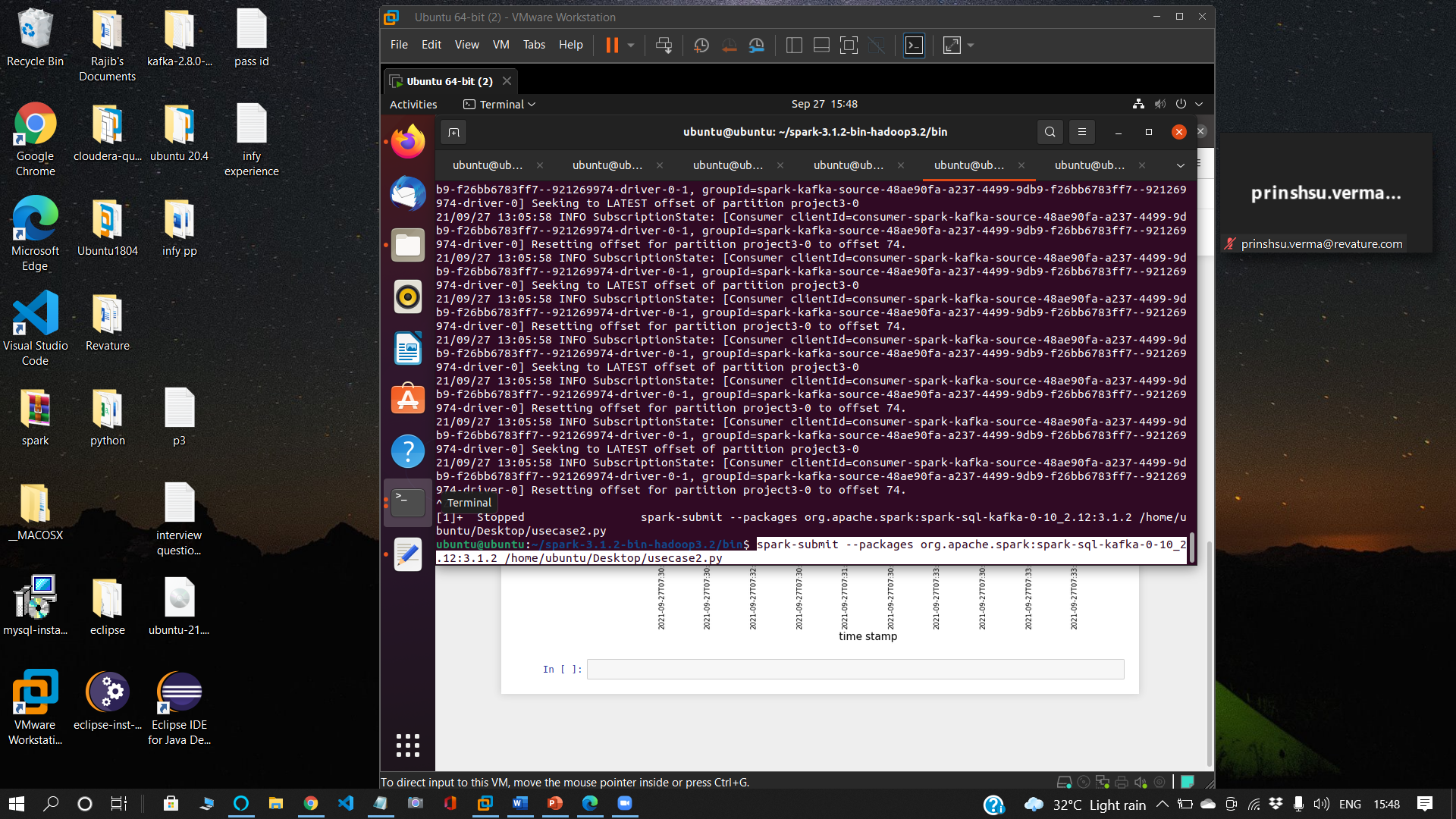
**STEP: - 9**

1. **GOTO SPARK BIN FOLDER**

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

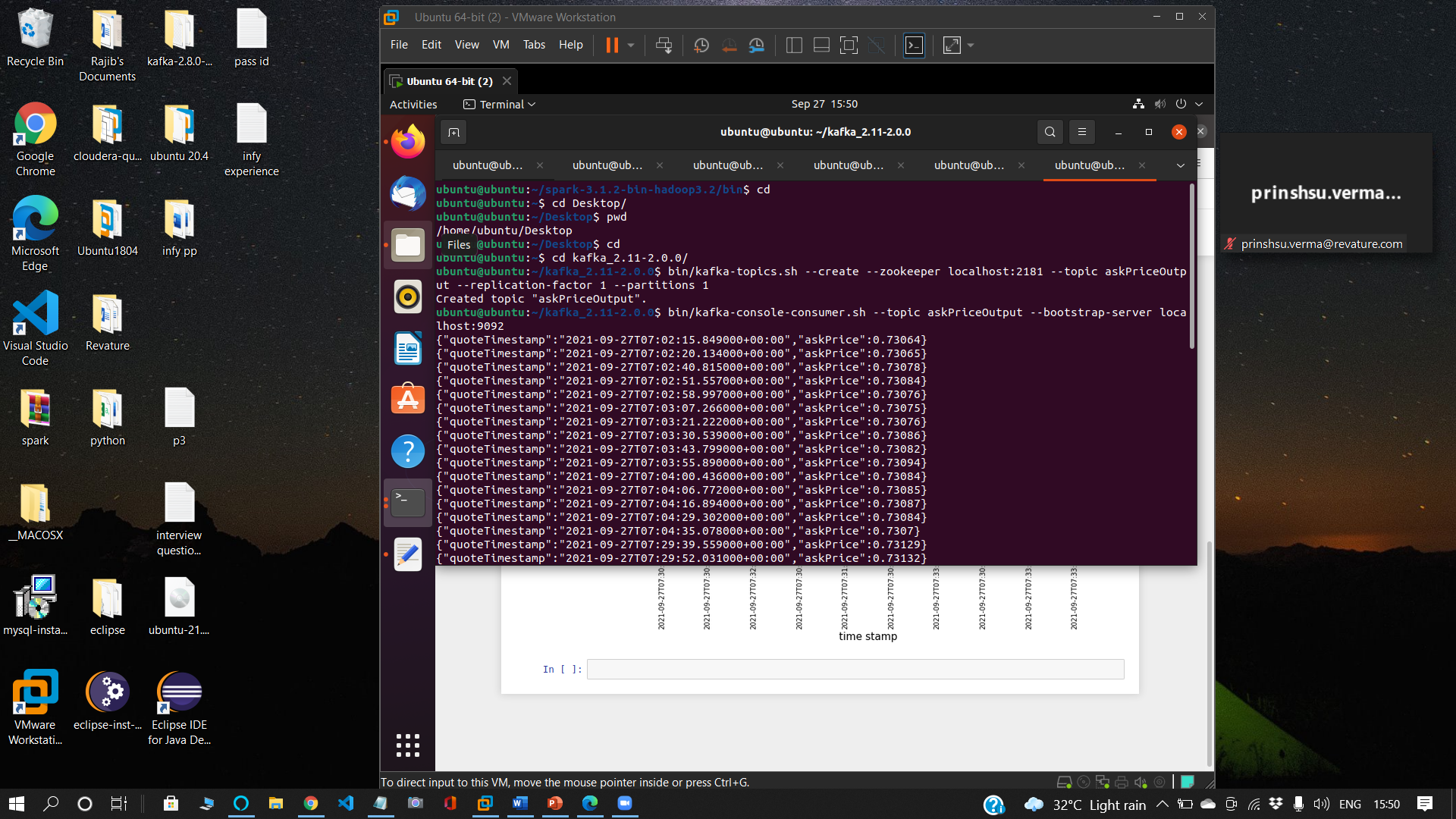
1. **RUN SPARK-SUBMIT JOB WITH usecase.py FILE AND STORE THE OUTPUT ON askPriceOutput TOPIC**

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



**STEP: - 10**

**OUTPUT WILL BE SHOWING ON askPriceOutput TOPIC’S CONSUMER**



**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 isthere")

