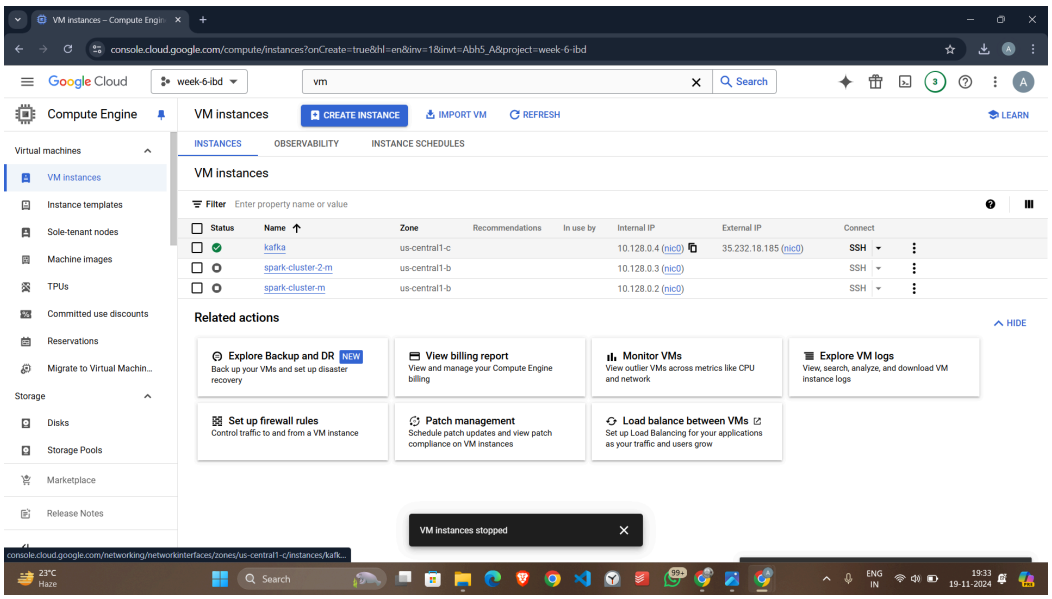
Introduction to Big Data
Graded Assignment 7

Name - Avijeet Palit
Roll - 21f1005675
Date - 15/11/2024

Question: Create an input data file that has at least 1000 rows and place it in a GCS bucket. You are welcome to reuse from previous assignments if the file has the minimum number of rows.
Write a producer that reads from that file, breaks the data into batches of 10 records, and writes to kafka, such that each batch is separated by a sleep time of 10 seconds from the previous batch. The producer can stop emitting once 1000 records are Written. Write a spark Streaming consumer that reads from the same Kafka topic every 5 seconds and emits the count of rows seen in the last 10 seconds.Launch both the producer and spark streaming in 2 separate windows such that both
are running simultaneously. The runs can stop when all 1000 rows are consumed.

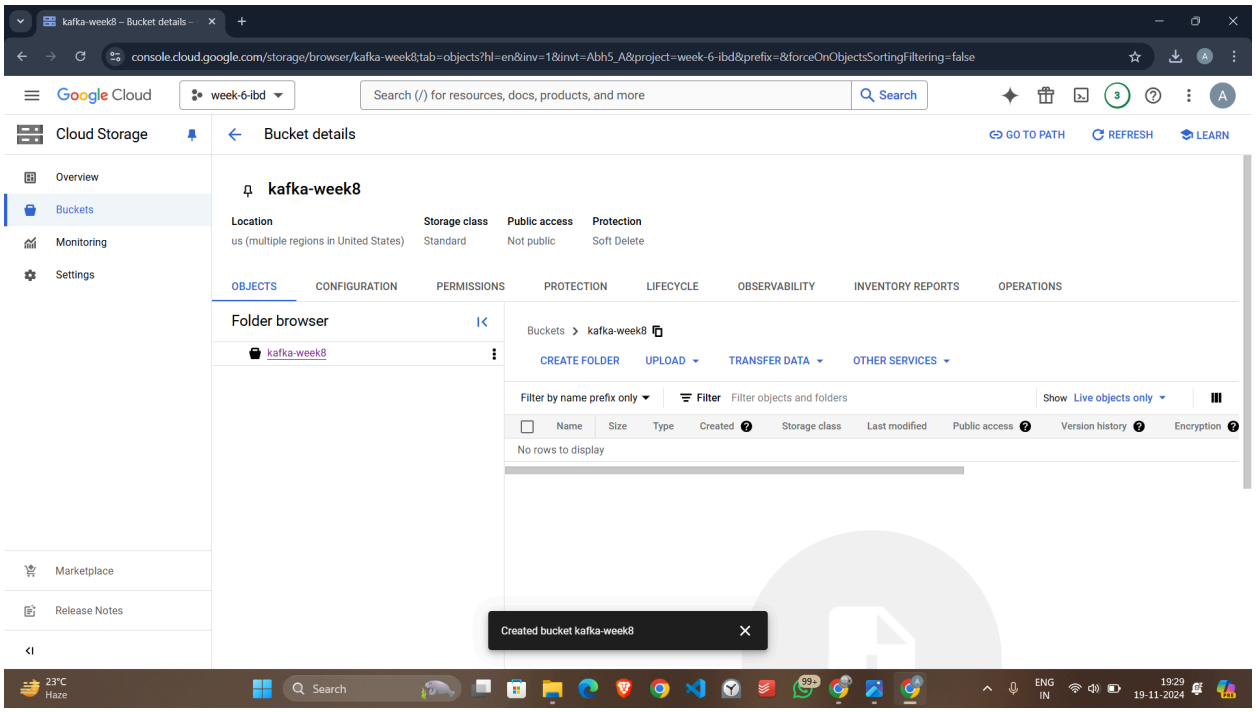
Step 1: Virtual Machine (VM) Setup Set up a VM instance to host Kafka, PySpark, and other required components.

1. Logged into the **Google Cloud Console**.
2. Created a VM instance named **kafka**.
3. Connected to the VM using SSH for configuration.



Step 2: Cloud Bucket Setup Create a cloud storage bucket for storing the CSV file for ease of access and processing.

1. Created a bucket named **kafka-week8**.



Step 3: CSV File Preparation Generate the `input_data.csv` file for processing.

1. Used a Python script (`data.py`) to create and preprocess the `input_data.csv` file locally.

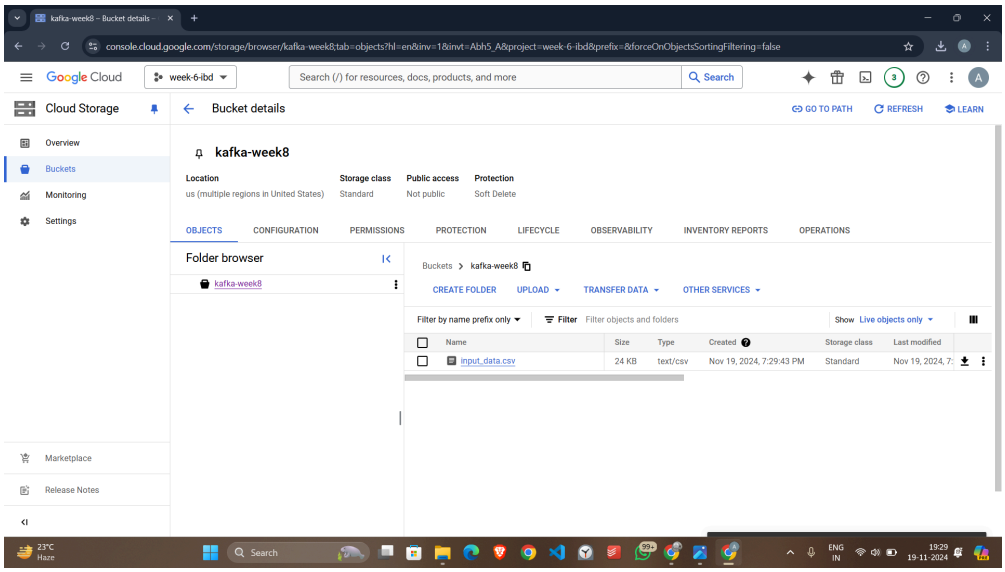
```python
import pandas as pd
import numpy as np

# Generate sample data
rows = 1000
data = {
    'id': range(1, rows + 1),
    'name': [f'User_{i}' for i in range(1, rows + 1)],
    'age': np.random.randint(18, 60, size=rows),
    'city': np.random.choice(['Kolkata', 'Delhi', 'Mumbai', 'Chennai', 'Bangalore'], size=rows)
}

df = pd.DataFrame(data)
# Save to CSV
df.to_csv('input_data.csv', index=False)
print("CSV file generated successfully.")
```

Step 4: Upload CSV to Cloud Bucket Upload the prepared CSV file to the cloud bucket.

1. Uploaded the `input_data.csv` file to the **kafka-week8** bucket for further use.



Step 5: Kafka Setup for CSV Data Configure Kafka to process data from the uploaded CSV file in real-time.

1. **Kafka Producer:**
    - Created a producer script (`producer.py`) to read data from `input_data.csv` and send it to a Kafka topic named `my_topic`.

consumer.py 3 producer.py 1

C: > Users > HP > OneDrive > Desktop > IBD > 21f1005675assignment_week8 > producer.py > ...

```python
from kafka import KafkaProducer
import pandas as pd
import time
import json

# Set Kafka Producer
producer = KafkaProducer(bootstrap_servers='localhost:9092', value_serializer=lambda v: json.dumps(v).encode('utf-8'))

# Read data from the downloaded CSV file
df = pd.read_csv('input_data.csv')

# Send data in batches of 10
batch_size = 10
for i in range(0, len(df), batch_size):
    batch = df.iloc[i:i + batch_size].to_dict(orient='records')
    producer.send('my_topic', value=batch)
    print(f'Sent batch {i // batch_size + 1}')
    time.sleep(10)  # Sleep for 10 seconds before sending the next batch

producer.flush()
producer.close()
```

2. **Kafka Consumer:**
    - Developed a Spark consumer script (`consumer.py`) to consume messages from `my_topic` and display the data on the console.

C: > Users > HP > OneDrive > Desktop > IBD > 21f1005675assignment_week8 > consumer.py > ...

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.sql.types import StringType

# Initialize Spark Session
spark = SparkSession.builder \
    .appName("KafkaSparkConsumer") \
    .getOrCreate()

# Read from Kafka topic
df = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "my_topic") \
    .load()

# Convert Kafka value to String
df = df.selectExpr("CAST(value AS STRING)")

# Count rows in each batch
count_df = df.groupBy().count()

# Output the count of rows to the console every 10 seconds
query = count_df.writeStream \
    .outputMode("complete") \
    .format("console") \
    .trigger(processingTime='10 seconds') \
    .start()

query.awaitTermination()
```

Step 6: VM Terminals for Kafka Setup

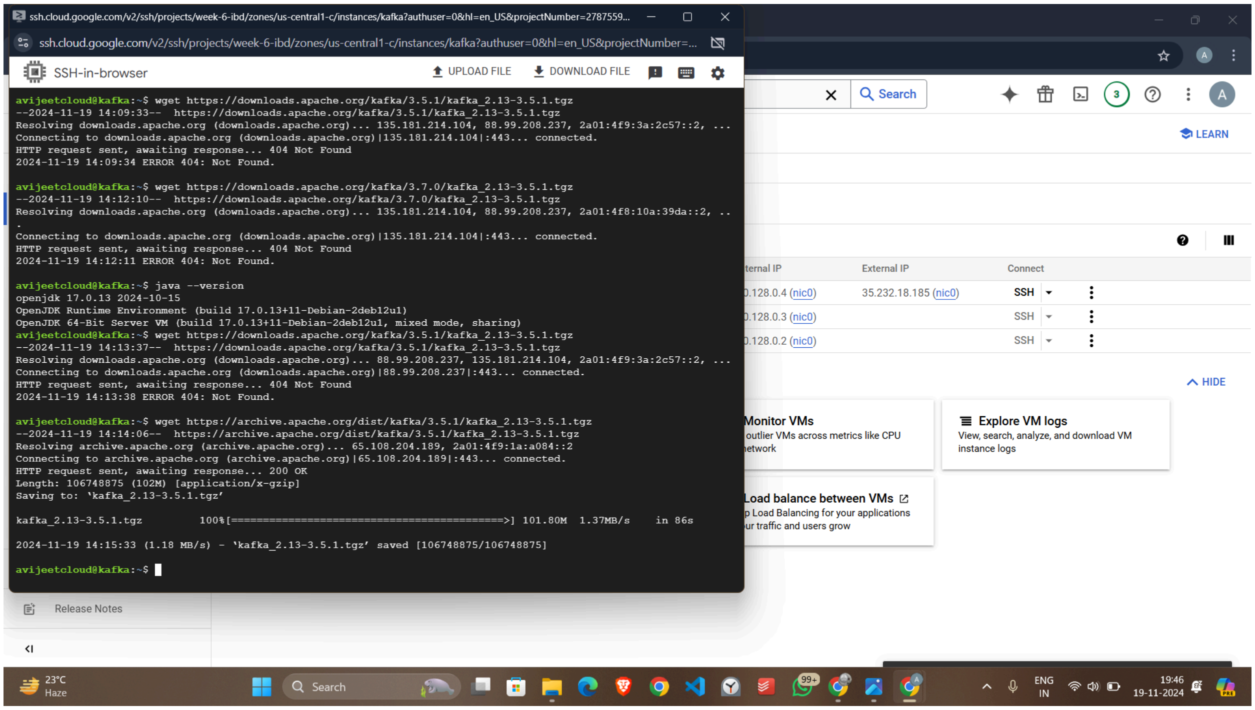
VM 1: Zookeeper Setup

1. Connected to the VM via SSH. Installed Java (required for Kafka):

   ```
   sudo apt-get update
   sudo apt-get install kafka
   ```

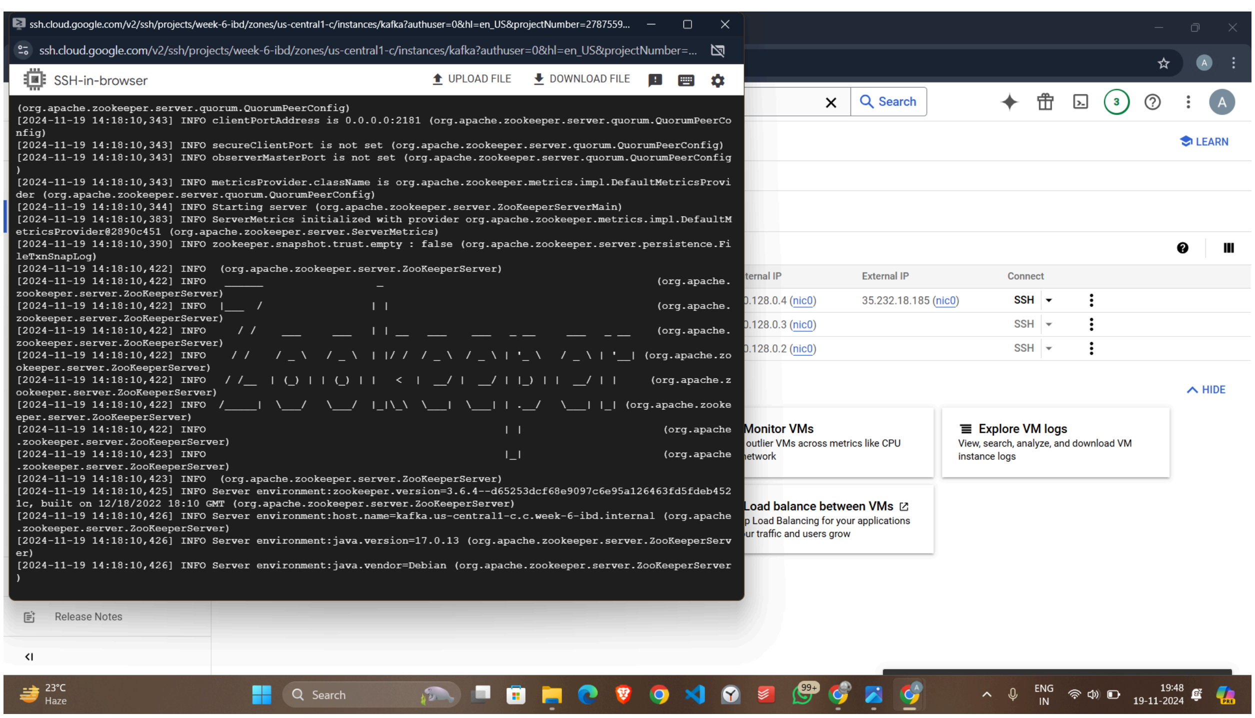
2. Downloaded and extracted Kafka:
   `wget https://archive.apache.org/dist/kafka/3.5.1/kafka_2.13-3.5.1.tgz`

```
tar -xzf kafka_2.13-3.5.1.tgz
cd kafka_2.13-3.5.1
```



3. Started Zookeeper:

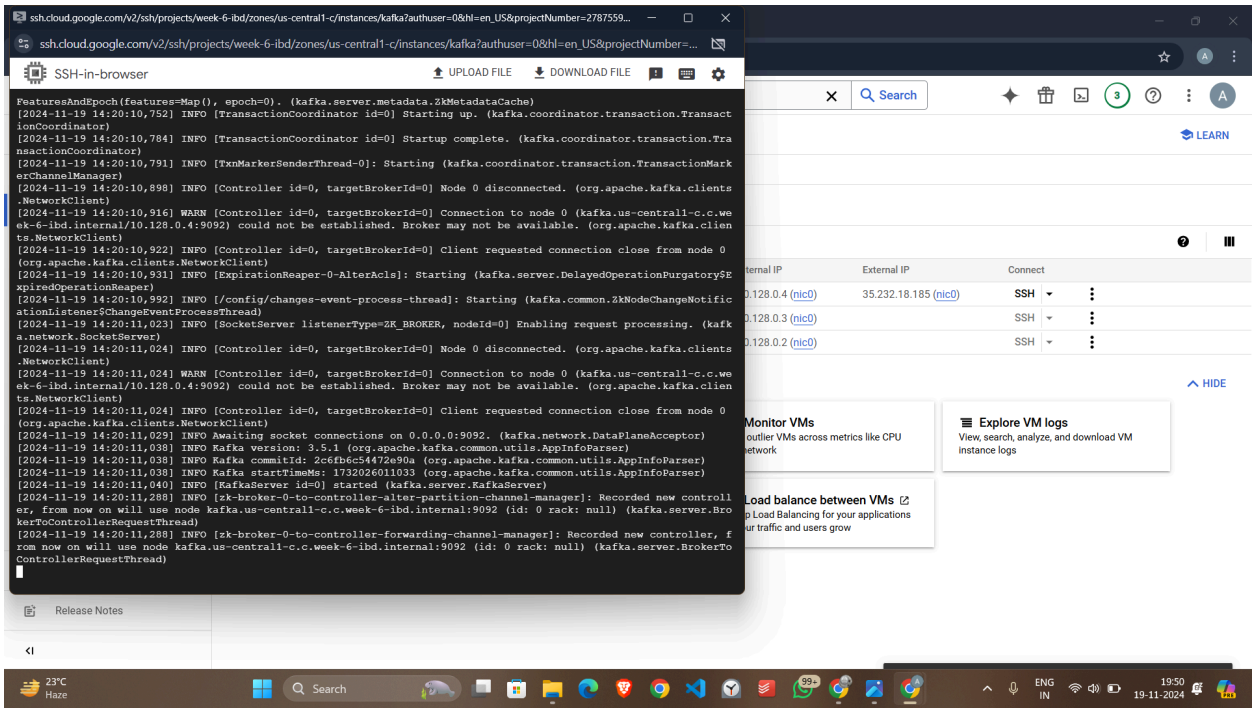
```
bin/zookeeper-server-start.sh config/zookeeper.properties
```



VM 2: Kafka Server Setup Opened another terminal and navigated to the Kafka directory:

```
cd kafka_2.13-3.5.1
```

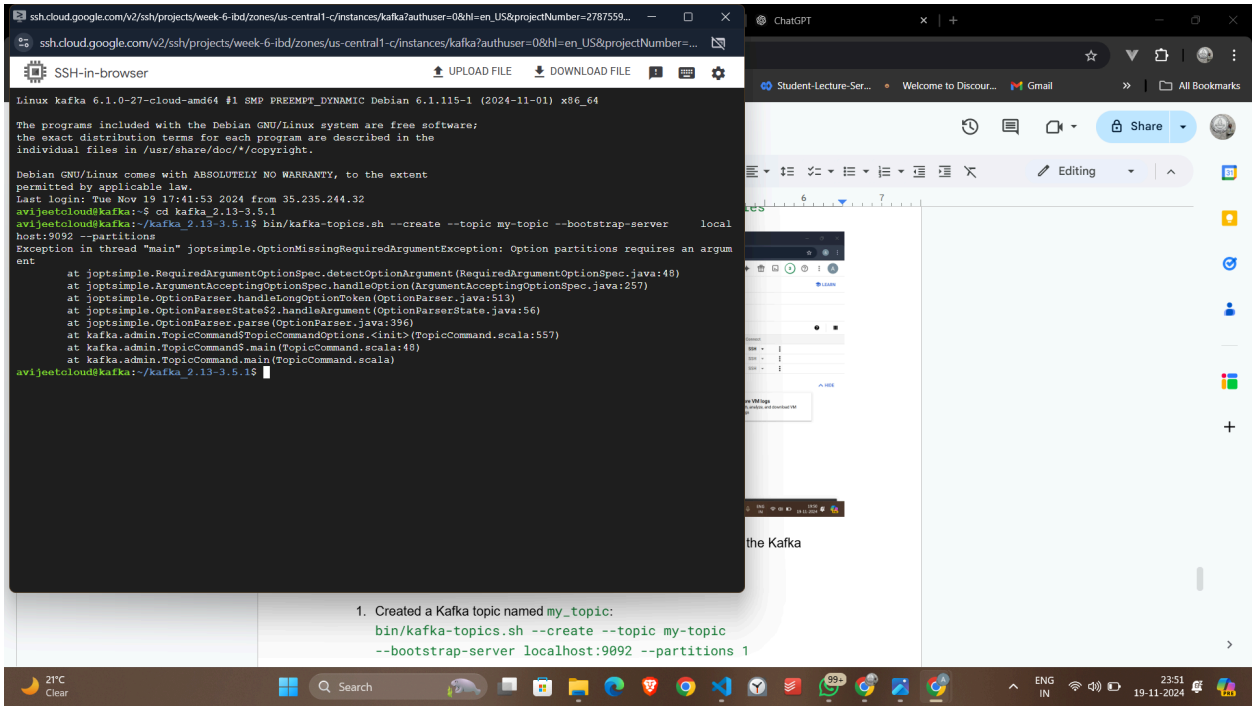
1. Started the Kafka server:
   ```
   bin/kafka-server-start.sh config/server.properties
   ```



VM 3: Kafka Topic Creation Opened a third terminal and navigated to the Kafka directory:

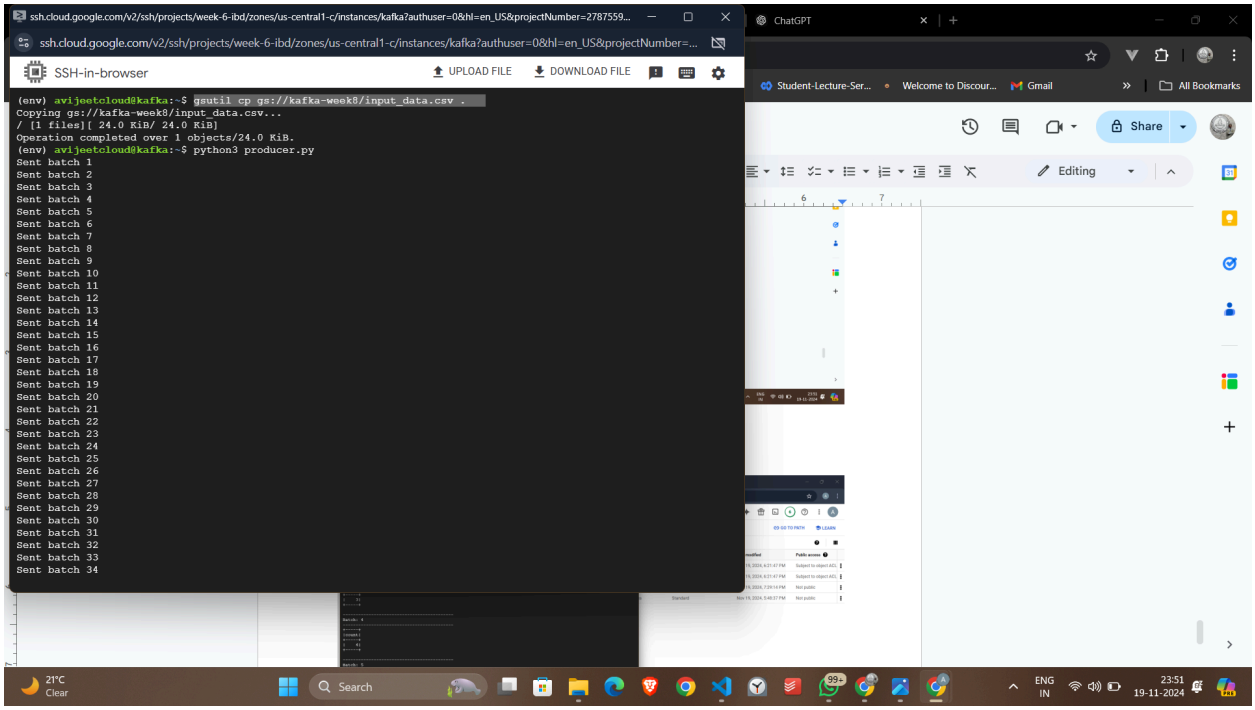
```
cd kafka_2.13-3.5.1
```

1. Created a Kafka topic named `my_topic`:
   ```
   bin/kafka-topics.sh --create --topic my-topic
   --bootstrap-server localhost:9092 --partitions 1
   ```



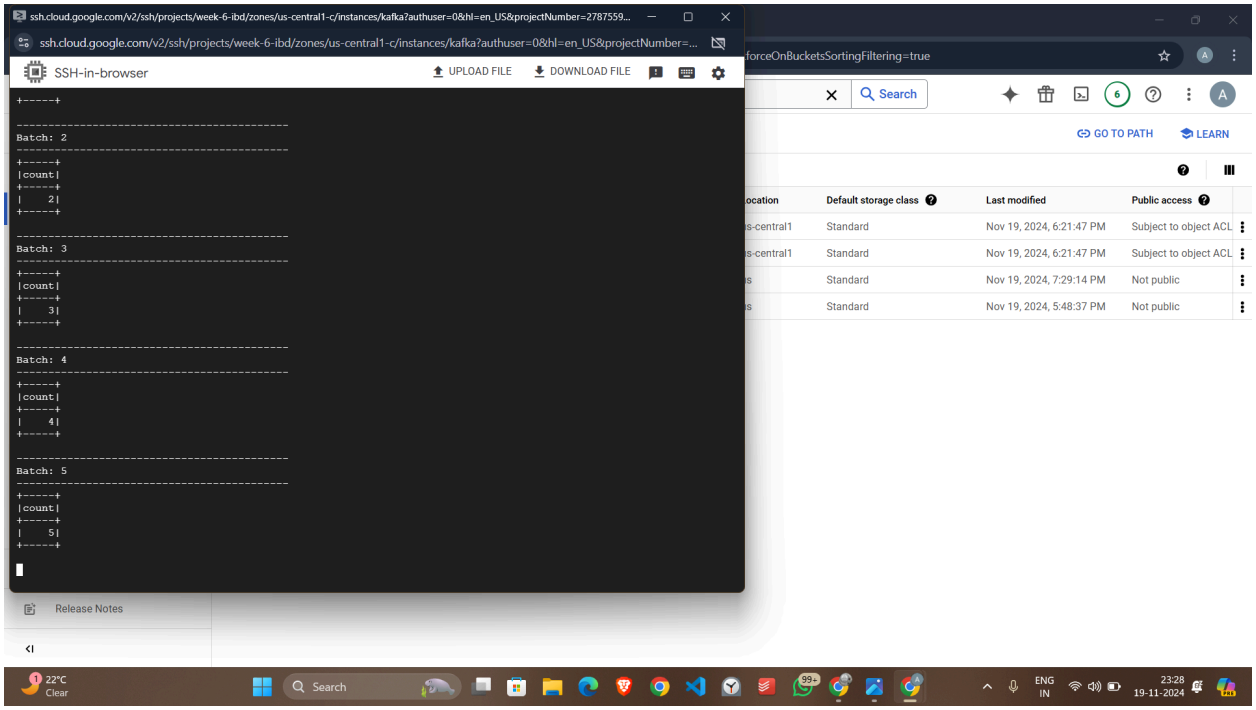
VM 4: Kafka Producer Execution

1. Uploaded the producer.py file to the VM.
   Navigated to the Kafka directory:
   `cd kafka_2.13-3.5.1`
2. Created a Python virtual environment and installed required libraries:
   `python -m venv ven`
   `source ven/bin/activate`
   `sudo apt install python-pip`
   `pip install kafka-python pandas`
3. Downloaded the CSV file from the cloud bucket:
   `gsutil cp gs://kafka-week8/input_data.csv .`
4. Ran the producer script to send data to Kafka:
   `python producer.py`



VM 5: Spark Consumer Execution

1. Uploaded the `consumer.py` file to the VM.
   Navigated to the Kafka directory:
   `cd kafka_2.13-3.5.1`
2. Created a Python virtual environment and installed required libraries:
   `python -m venv ven`
   `source ven/bin/activate`
   `pip install pyspark`
3. Ran the Spark consumer script with Kafka integration:
   `spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.1 consumer.py`
4. `python consumer.py`


ssh.cloud.google.com/v2/ssh/projects/week-6-ibd/zones/us-central1-c/instances/kafka?authuser=0&hl=en_US&projectNumber=2787559...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE
+-----+
Batch: 2
|count|
| 2|
Batch: 3
|count|
| 3|
Batch: 4
|count|
| 4|
Batch: 5
|count|
| 5|
Search
GO TO PATH
LEARN
Default storage class
Last modified
Public access
Standard
Nov 19, 2024, 6:21:47 PM
Subject to object ACL
Nov 19, 2024, 7:29:14 PM
Not public
Nov 19, 2024, 5:48:37 PM
Release Notes
22°C
Clear
ENG
IN
23:28
19-11-2024