

Introduction to Big Data

Graded Assignment : 9

Name - Avijeet Palit

Roll - 21f1005675

Date - 04/12/2024

Objective.....	3
Detailed Steps.....	3
Step 1: Create a GCP Project.....	3
Step 2: Download the Image data set from tf flower dataset.....	3
Step 3: Set Up Google Cloud Storage (GCS).....	4
Step 4: Create a Virtual Machine (VM):.....	5
Step 3: Install Kafka and Dependencies.....	5
Step 4: Set Up and Start Zookeeper.....	6
Step 5: Create a New Firewall Rule.....	6
Step 6: Set Up Kafka.....	7
Step 7: Kafka Producer for Image Metadata.....	7
Step 8: Create a Dataproc Cluster.....	7
Step 9: Result and Testing the Pipeline.....	8
Conclusion.....	8

Objective

Convert the batch image classification use case walked through in the class to a real-time execution model using Spark Streaming. The notebook is attached as a separate link in week 10. Run it and record your screen to capture the outputs so as to show proof of the streaming execution.

Detailed Steps

Step 1: Create a GCP Project

- Log in to the Google Cloud Console.
- Create a new project or use an existing one.
- Enable the required APIs:
 - **Cloud Storage API**
 - **Dataproc API**
 - **Compute Engine API**

Step 2: Download the Image data set from tf flower dataset

- Setup the colab
- Download the images
- Save it in a zip folder
- Select some images and store it in a folder “images” in local machine

```
[ ] 1 import tensorflow as tf
    2
    3 # Download and extract the dataset
    4 data_dir = tf.keras.utils.get_file(
    5     origin='https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz',
    6     fname='flower_photos', untar=True
    7 )
    8
    9 print("Dataset downloaded and extracted at:", data_dir)
```

Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 — 11s 0us/step
Dataset downloaded and extracted at: /root/.keras/datasets/flower_photos

```
▶ 1 import os
    2
    3 # Check extracted files
    4 print("Extracted files:")
    5 for root, dirs, files in os.walk(data_dir):
    6     print(root, "contains", len(files), "files.")
    7
```

Extracted files:
/root/.keras/datasets/flower_photos contains 1 files.
/root/.keras/datasets/flower_photos/roses contains 641 files.
/root/.keras/datasets/flower_photos/tulips contains 799 files.
/root/.keras/datasets/flower_photos/daisy contains 633 files.
/root/.keras/datasets/flower_photos/dandelion contains 898 files.
/root/.keras/datasets/flower_photos/sunflowers contains 699 files.

```
[ ] 1 !zip -r flower_photos.zip /root/.keras/datasets/flower_photos
    2
```

Step 3: Set Up Google Cloud Storage (GCS)

- Create a Cloud Storage bucket for storing images:
 - Go to **Cloud Storage > Buckets**.
 - Create a bucket (e.g., **image-classification-project**) with the appropriate region and permissions.
 - Upload some test images to the bucket (e.g., under **images/**).
 - Upload **install_packages.sh**

←

Bucket details

GO TO PATH

REFRESH

LEARN

📁 image-classification-project

Location

us (multiple regions in United States)

Storage class

Standard

Public access

Not public

Protection

Soft Delete

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

INVENTORY REPORTS

OPERATIONS

Folder browser

image-classification-project

google-cloud-dataproc-metainfo/

images/

notebooks/

Buckets > image-classification-project

CREATE FOLDER

UPLOAD

TRANSFER DATA

OTHER SERVICES

Filter by name prefix only

Filter

Filter objects and folders

Show Live objects only

<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	
<input type="checkbox"/>	google-cloud-dataproc-metainfo/	—	Folder	—	—	—	
<input type="checkbox"/>	images/	—	Folder	—	—	—	
<input type="checkbox"/>	install_packages.sh	265 B	text/x-sh	Dec 7, 2024, 2:41:48 PM	Standard	Dec 7, 2024	
<input type="checkbox"/>	notebooks/	—	Folder	—	—	—	
<input type="checkbox"/>	stream.py	2.5 KB	text/x-python	Dec 7, 2024, 11:18:04 PM	Standard	Dec 7, 2024	

Step 4: Create a Virtual Machine (VM):

- Navigate to **Compute Engine > VM Instances**.
- Click **Create Instance** “kafka-external”
- Choose:
 - Machine type (e.g., e2-medium for basic setups).
 - Operating system (e.g., Ubuntu 22.04 LTS).
- Configure networking to allow external access:
 - Under **Firewall**, check "Allow HTTP traffic" and "Allow HTTPS traffic".

Basic information	
Name	kafka-external
Instance Id	3029789174895597856
Description	None
Type	Instance
Status	Running
Creation time	Dec 7, 2024, 8:50:48 PM UTC+05:30
Location	us-central1-c
Instance template	None
In use by	None
Reservations	Automatically choose

Step 3: Install Kafka and Dependencies

- **Connect to the VM: SSH into the VM**
 - Install Java: Kafka requires Java. Install OpenJDK:


```
sudo apt update
sudo apt install -y default-jdk
java -version
```

- **Download and Install Kafka:**

- Navigate to [Apache Kafka Downloads](#).
- Download Kafka:

```
Wget
```

```
https://downloads.apache.org/kafka/3.5.1/kafka_2.13-3.5.1.tgz
```

- Extract the Kafka archive:

```
tar -xvzf kafka_2.13-3.5.1.tgz
```

```
mv kafka_2.13-3.5.1 kafka
```

- **Configure Kafka:**

- Open Kafka configuration file:

```
nano kafka/config/server.properties
```

- Set the following:

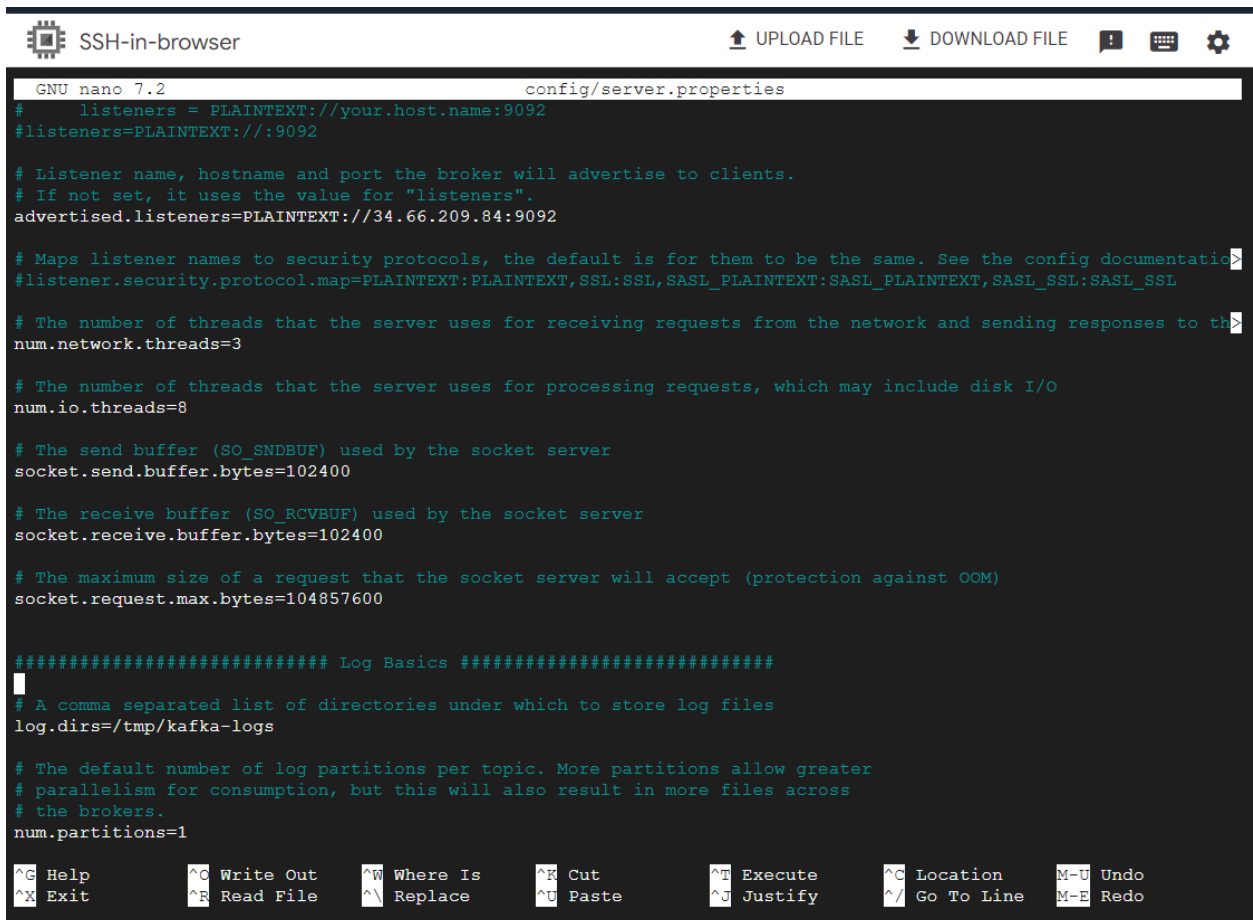
- Advertised Hostname:

```
advertised.listeners=PLAINTEXT://34.66.209.84:9092
```

- Zookeeper (default):

```
zookeeper.connect=localhost:2181
```

- Save and exit.



The screenshot shows a web browser window titled "SSH-in-browser" with a dark theme. The main content is a nano 7.2 editor window displaying the file "config/server.properties". The editor shows the following configuration:

```
GNU nano 7.2 config/server.properties
# listeners = PLAINTEXT://your.host.name:9092
#listeners=PLAINTEXT://:9092

# Listener name, hostname and port the broker will advertise to clients.
# If not set, it uses the value for "listeners".
advertised.listeners=PLAINTEXT://34.66.209.84:9092

# Maps listener names to security protocols, the default is for them to be the same. See the config documentation
#listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL

# The number of threads that the server uses for receiving requests from the network and sending responses to the network
num.network.threads=3

# The number of threads that the server uses for processing requests, which may include disk I/O
num.io.threads=8

# The send buffer (SO_SNDBUF) used by the socket server
socket.send.buffer.bytes=102400

# The receive buffer (SO_RCVBUF) used by the socket server
socket.receive.buffer.bytes=102400

# The maximum size of a request that the socket server will accept (protection against OOM)
socket.request.max.bytes=104857600

##### Log Basics #####
# A comma separated list of directories under which to store log files
log.dirs=/tmp/kafka-logs

# The default number of log partitions per topic. More partitions allow greater
# parallelism for consumption, but this will also result in more files across
# the brokers.
num.partitions=1
```

At the bottom of the editor, there is a status bar with various keyboard shortcuts for nano editor functions:

^G Help	^O Write Out	^W Where Is	^R Cut	^T Execute	^C Location	M-U Undo
^X Exit	^R Read File	^N Replace	^U Paste	^J Justify	^_ Go To Line	M-E Redo

Step 4: Set Up and Start Zookeeper

- Kafka requires Zookeeper for coordination.
- Start Zookeeper:

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

SSH-in-browser

UPLOAD FILE

DOWNLOAD FILE

! 📄 ⚙️

```
nfig)
[2024-12-07 19:30:03,310] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-07 19:30:03,310] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-07 19:30:03,310] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-12-07 19:30:03,310] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)
[2024-12-07 19:30:03,341] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@2890c451 (org.apache.zookeeper.server.ServerMetrics)
[2024-12-07 19:30:03,346] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2024-12-07 19:30:03,365] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,365] INFO _____ (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO |__ / | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO // __ __ | | __ __ __ __ (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO // / _ \ / _ \ | | / / / _ \ / _ \ | ' \ / _ \ | ' _| (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO //__ | () | | () | | < | _/ | _/ | | | _/ | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO /___| \_/ \_/ |_|\_ \_/ \_/ | | _/ \_/ | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,366] INFO | | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,367] INFO |_ | (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,367] INFO (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,369] INFO Server environment:zookeeper.version=3.6.4--d65253dcf68e9097c6e95a126463fd5fdeb4521c, built on 12/18/2022 18:10 GMT (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,369] INFO Server environment:host.name=kafka-external.us-central1-c.c.week-6-ibd.internal (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,369] INFO Server environment:java.version=17.0.13 (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,370] INFO Server environment:java.vendor=Debian (org.apache.zookeeper.server.ZooKeeperServer)
[2024-12-07 19:30:03,370] INFO Server environment:java.home=/usr/lib/jvm/java-17-openjdk-amd64 (org.apache.zookeeper.server.ZooKeeperServer)
```

Step 5: Create a New Firewall Rule

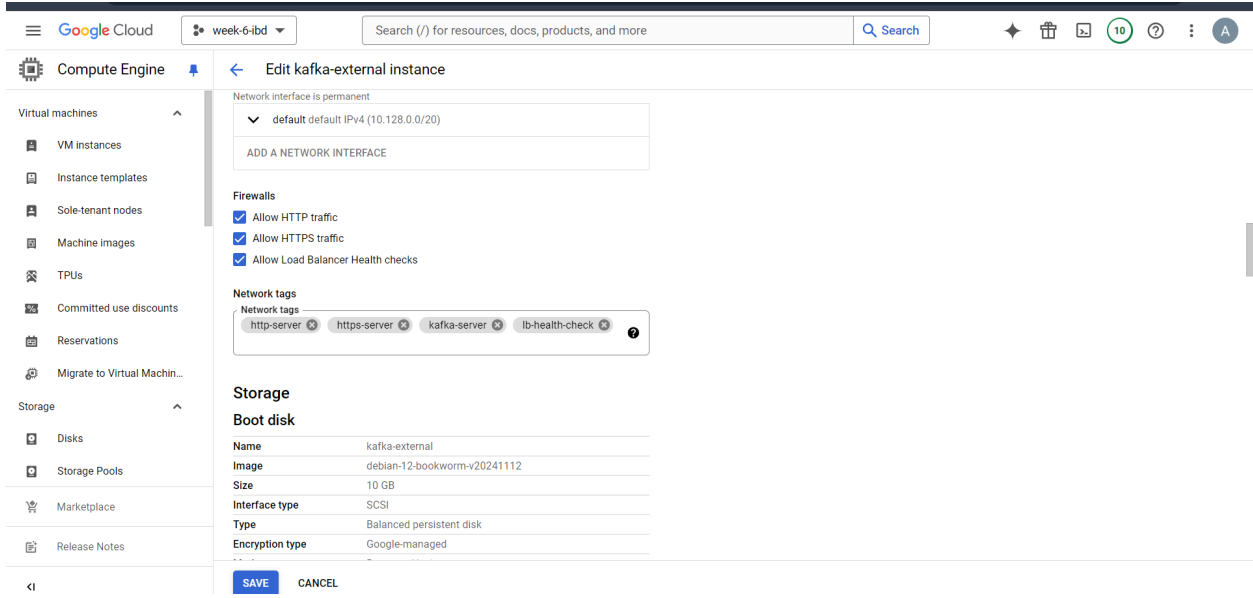
- Click the **"Create Firewall Rule"** button at the top.
- Fill in the details for the new rule:
 - **Name:** Enter a descriptive name, such as `allow-kafka-9092`.
 - **Network:** Select the network your VM instance is part of (usually `default` unless you've set up a custom network).
 - **Priority:** Leave the default value (1000) unless you have specific needs.
 - **Direction of traffic:** Select **Ingress** (incoming traffic).
 - **Action on match:** Select **Allow**.
 - **Targets:**
 - Choose **Specified target tags**.
 - Add a **tag** (e.g., `kafka-server`) that you will assign to your VM later.

- **Source filter:**
 - Choose **IP ranges**.
 - Enter the IP range allowed to access your Kafka server:
 - For open access: **0.0.0.0/0** (not recommended for production environments due to security risks).
 - For restricted access: Use a specific IP or range, e.g., **192.168.1.0/24**.
- **Protocols and ports:**
 - Select **Specified protocols and ports**.
 - Check **tcp** and specify port **9092**.
- **Save the Firewall Rule**
 - Click the **Create** button to save the rule.

The screenshot shows the Google Cloud Network Security console. The left sidebar lists various security services, with 'Firewall policies' selected. The main panel displays the configuration for a firewall rule. The 'Targets' section shows 'kafka-server' as the target tag. The 'Source filters' section shows 'IP ranges' set to '0.0.0.0/0'. The 'Protocols and ports' section shows 'tcp:9092'. The 'Enforcement' section shows 'Enabled'. The 'Insights' section shows 'None'. The 'Hit count monitoring' section shows a question mark icon. Below these sections, a message states: 'The following table does not show any App Engine flexible environment instances'. A table with columns: Name, Subnetwork, Internal IP ranges, External IP ranges, Tags, Service accounts, Project, Labels, and Network details is shown. The table contains one entry: 'kafka-external' with subnetwork 'default', internal IP range '10.128.0.16', external IP range '34.66.209.84', tag 'http-', service account '278755983826-', project 'week-6-ibd', and network details 'VIEW DE'.

Name	Subnetwork	Internal IP ranges	External IP ranges	Tags	Service accounts	Project	Labels	Network details
kafka-external	default	10.128.0.16	34.66.209.84	http-	278755983826-	week-6-ibd		VIEW DE

- **Tag Your VM with the Firewall Rule**
 - Go to **Compute Engine > VM Instances**.
 - Find the VM running Kafka.
 - Click the **Edit** button at the top of the VM details page.
 - In the **Network tags** section, add the same tag you used in the firewall rule (e.g., **kafka-server**).
 - Click **Save**.



Step 6: Set Up Kafka

- Start Kafka Broker:
`kafka/bin/kafka-server-start.sh kafka/config/server.properties &`

```
[2024-12-07 19:31:03,066] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.group.GroupCoordinator)
[2024-12-07 19:31:03,127] INFO [TransactionCoordinator id=0] Starting up. (kafka.coordinator.transaction.TransactionCoordinator)
[2024-12-07 19:31:03,145] INFO [TransactionCoordinator id=0] Startup complete. (kafka.coordinator.transaction.TransactionCoordinator)
[2024-12-07 19:31:03,154] INFO [TxnMarkerSenderThread-0]: Starting (kafka.coordinator.transaction.TransactionMarkerChannelManager)
[2024-12-07 19:31:03,281] INFO [Controller id=0, targetBrokerId=0] Node 0 disconnected. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,284] WARN [Controller id=0, targetBrokerId=0] Connection to node 0 (/34.66.209.84:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,287] INFO [ExpirationReaper-0-AlterAcls]: Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2024-12-07 19:31:03,292] INFO [Controller id=0, targetBrokerId=0] Client requested connection close from node 0 (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,388] INFO [/config/changes-event-process-thread]: Starting (kafka.common.ZkNodeChangeNotificationListener$ChangeEventProcessThread)
[2024-12-07 19:31:03,399] INFO [Controller id=0, targetBrokerId=0] Node 0 disconnected. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,401] WARN [Controller id=0, targetBrokerId=0] Connection to node 0 (/34.66.209.84:9092) could not be established. Broker may not be available. (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,402] INFO [Controller id=0, targetBrokerId=0] Client requested connection close from node 0 (org.apache.kafka.clients.NetworkClient)
[2024-12-07 19:31:03,421] INFO [SocketServer listenerType=ZK_BROKER, nodeId=0] Enabling request processing. (kafka.network.SocketServer)
[2024-12-07 19:31:03,426] INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.DataPlaneAcceptor)
[2024-12-07 19:31:03,439] INFO Kafka version: 3.5.1 (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,440] INFO Kafka commitId: 2c6fb6c54472e90a (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,440] INFO Kafka startTimeMs: 1733599863433 (org.apache.kafka.common.utils.AppInfoParser)
[2024-12-07 19:31:03,442] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
[2024-12-07 19:31:03,598] INFO [zk-broker-0-to-controller-forwarding-channel-manager]: Recorded new controller, from now on will use node 34.66.209.84:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2024-12-07 19:31:03,639] INFO [zk-broker-0-to-controller-alter-partition-channel-manager]: Recorded new controller, from now on will use node 34.66.209.84:9092 (id: 0 rack: null) (kafka.server.BrokerToControllerRequestThread)
[2024-12-07 19:31:03,708] INFO [ReplicaFetcherManager on broker 0] Removed fetcher for partitions Set(image-topic-0) (kafka.server.ReplicaFetcherManager)
[2024-12-07 19:31:03,752] INFO [Partition image-topic-0 broker=0] Log loaded for partition image-topic-0 with initial high watermark 815 (kafka.cluster.Partition)
```

- Create a topic:
`kafka/bin/kafka-topics.sh --create --topic test-topic
--bootstrap-server 34.66.209.84:9092`
- List topics:
`kafka/bin/kafka-topics.sh --list --bootstrap-server
34.66.209.84:9092`

```
avijeetcloud@kafka-external:~/kafka_2.13-3.5.1$ bin/kafka-topics.sh --list --bootstrap-server 34.66.209.84:9092
image-topic
avijeetcloud@kafka-external:~/kafka_2.13-3.5.1$
```

Step 7: Kafka Producer for Image Metadata

- Write a **Kafka producer** script and upload it to the VM ssh.

```
1 from kafka import KafkaProducer
2 from google.cloud import storage
3 import time
4
5 # Initialize Kafka Producer
6 producer = KafkaProducer(
7     bootstrap_servers='34.66.209.84:9092',
8     acks='all', # Wait for acknowledgment from all replicas
9     batch_size=1 # Send each record as a separate batch
10 )
11
12 # Function to list and send files from Google Cloud Storage
13 def list_files(bucket_name):
14     client = storage.Client()
15     bucket = client.get_bucket(bucket_name)
16     blobs = bucket.list_blobs(prefix="images/") # Adjust prefix if ne
17     for blob in blobs:
18         # Send each image name as a message to the Kafka topic
19         producer.send('image-topic', value=blob.name.encode())
20         print(f"Image sent to topic: {blob.name}")
21         time.sleep(7) # Pause between sending images
22
23 # Specify GCS Bucket Name
24 list_files('image-classification-project')
25 print("Done!")
26
```

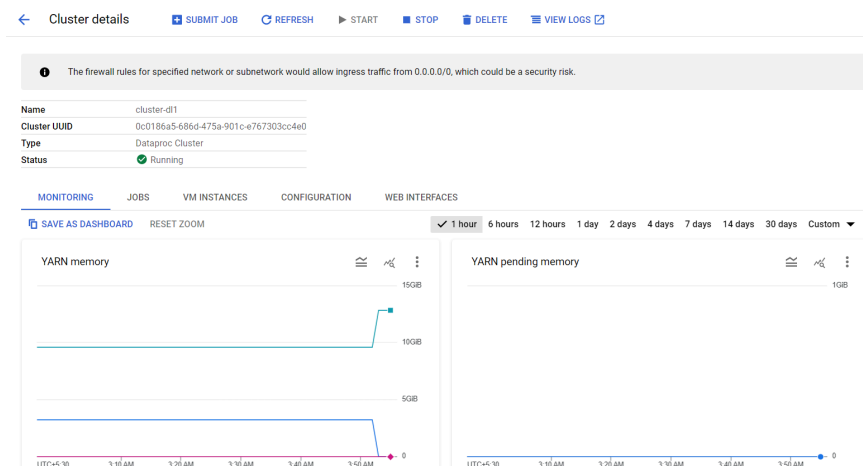
- Install the python virtual environment and activate it
 - `sudo apt update`
 - `sudo apt install python3-venv`

- `python3 -m venv env`
 - `source env/bin/activate`
- Install the Kafka Python client:
 - `pip install kafka-python google-cloud-storage`
- Run the `producer.py`
 - `Python3 producer.py`

Step 8: Create a Dataproc Cluster

- Go to **Dataproc > Clusters > Create Cluster**.
- Choose:
 - Cluster type: **Standard (1 master, N workers)**.
 - Machine type: **e2-standard-4** (adjust based on workload).
 - Enable components like **Jupyter Notebook** if needed.
 - Add **Initialization actions** with
 - `gs://image-classification-project/install_packages.sh`

```
$ install_packages.sh
1  #!/bin/bash
2  # install_packages.sh
3
4  sudo apt-get update
5  # Install pip (if not already installed)
6  sudo apt-get install python3-pip -y
7
8  # Install required Python packages
9  pip3 install kafka-python
10 pip3 install torch torchvision tensorflow pillow google-cloud-storage
11
```



- Check the connection
 - telnet 34.66.209.84 9092

```
avijeetcloud@cluster-d11-m:~$ telnet 34.66.209.84 9092
Trying 34.66.209.84...
Connected to 34.66.209.84.
Escape character is '^]'.
█
```

- Write a Kafka consumer “Stream.py” script with the modified version on **Notebook** given in **Week 10** which leverages **TensorFlow** and uploads it to the VM ssh.
- Run the stream.py
 - Python3 stream.py

```
stream.py > ...
1  from pyspark.sql import SparkSession
2  from pyspark.sql.functions import col, udf
3  from pyspark.sql.types import StringType
4  from kafka import KafkaConsumer
5  import io
6  import json
7  from PIL import Image
8  import torch
9  import requests
10 from torchvision import models, transforms
11 from google.cloud import storage
12
13 # Initialize Spark Session
14 spark = SparkSession.builder \
15     .appName("ImageClassificationStreaming") \
16     .config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.1") \
17     .config("spark.executorEnv.TORCH_HOME", "/tmp/torch_cache") \
18     .getOrCreate()
19
20 # Kafka Source
21 df = spark.readStream \
22     .format("kafka") \
23     .option("kafka.bootstrap.servers", "34.66.209.84:9092") \
24     .option("subscribe", "image-topic") \
25     .option("failOnDataLoss", "false") \
26     .option("maxOffsetsPerTrigger", "10") \
27     .load()
28
29 df = df.selectExpr("CAST(value AS STRING) as path")
30
31 # ImageNet Labels Loader
32 def load_imagenet_labels():
33     url = "https://raw.githubusercontent.com/anishathalye/imagenet-simple-labels/master/imagenet-simple-labels.json"
34     response = requests.get(url)
35     return json.loads(response.text)
36
37 imagenet_labels = load_imagenet_labels()
38
```

```

# UDF for Image Classification
Tabnine | Edit | Test | Explain | Document | Ask
def classify_udf(path):
    try:
        client = storage.Client()
        bucket = client.get_bucket('image-classification-project')
        blob = bucket.get_blob(path)
        content = blob.download_as_bytes()

        transform = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
        image = Image.open(io.BytesIO(content)).convert("RGB")
        image_tensor = transform(image)

        model = models.mobilenet_v2(pretrained=True)
        model.eval()
        with torch.no_grad():
            predictions = model(image_tensor.unsqueeze(0))
            class_index = predictions.argmax().item()
            return json.dumps({"index": class_index, "label": imagenet_labels[class_index]})
    except Exception as e:
        return json.dumps({"error": str(e)})

# Register UDF
classify_spark_udf = udf(classify_udf, StringType())

# Apply UDF and Write Stream
predictions = df.withColumn("prediction", classify_spark_udf(col("path")))

query = predictions.writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .trigger(processingTime='10 seconds') \
    .start()

query.awaitTermination()

```

Step 9: Result and Testing the Pipeline

- **Kafka Testing:**
 - Use the Kafka producer script to push test messages containing image paths to the Kafka topic.
- **Upload Test Images:**
 - Add images to the GCS bucket under the appropriate path (e.g., [images/](#)).
- **Monitor Outputs:**
 - Check the cluster ssh terminal for real-time predictions.
 - Verify the console output for predictions.



SSH-in-browser

↑ UPLOAD FILE

↓ DOWNLOAD FILE



```
avijeetcloud@kafka-external:~$ source env/bin/activate
(env) avijeetcloud@kafka-external:~$ python3 producer.py
Image sent to topic: images/10172379554_b296050f82_n.jpg
Image sent to topic: images/10486992895_20b344ce2d_n.jpg
Image sent to topic: images/12240577184_b0de0e53ea_n.jpg
Image sent to topic: images/12471290635_1f9e3aae16_n.jpg
Image sent to topic: images/1306119996_ab8ae14d72_n.jpg
Image sent to topic: images/134409839_71069a95d1_m.jpg
Image sent to topic: images/13807932364_673b7f1c1c_n.jpg
Image sent to topic: images/14368895004_c486a29c1e_n.jpg
Image sent to topic: images/1441939151_b271408c8d_n.jpg
Image sent to topic: images/1469726748_f359f4a8c5.jpg
Image sent to topic: images/14698531521_0c2f0c6539.jpg
Image sent to topic: images/14741866338_bdc8bfc8d5_n.jpg
Image sent to topic: images/15319767030_e6c5602a77_m.jpg
Image sent to topic: images/15424480096_45bb574b33.jpg
Image sent to topic: images/16484100863_979beacb08.jpg
Image sent to topic: images/17280886635_e384d91300_n.jpg
Image sent to topic: images/18684594849_7dd3634f5e_n.jpg
Image sent to topic: images/1879567877_8ed2a5faa7_n.jpg
Image sent to topic: images/19813618946_93818db7aa_m.jpg
```

```
+-----+-----+
|path                               |prediction                               |
+-----+-----+
|images/16484100863_979beacb08.jpg|{"index": 562, "label": "fountain"}|
+-----+-----+
```

Batch: 9

```
+-----+-----+
|path                               |prediction                               |
+-----+-----+
|images/17280886635_e384d91300_n.jpg|{"index": 228, "label": "Komondor"}|
|images/18684594849_7dd3634f5e_n.jpg|{"index": 985, "label": "daisy"}|
+-----+-----+
```

Batch: 10

```
+-----+-----+
|path                               |prediction                               |
+-----+-----+
|images/1879567877_8ed2a5faa7_n.jpg|{"index": 985, "label": "daisy"}|
+-----+-----+
```

Batch: 11

```
+-----+-----+
|path                               |prediction                               |
+-----+-----+
|images/19813618946_93818db7aa_m.jpg|{"index": 985, "label": "daisy"}|
|images/2019064575_7656b9340f_m.jpg |{"index": 985, "label": "daisy"}|
+-----+-----+
```

Batch: 12

```
+-----+-----+
|path                               |prediction                               |
+-----+-----+
|images/2045022175_ad087f5f60_n.jpg|{"index": 89, "label": "sulphur-crested cockatoo"}|
+-----+-----+
```

Conclusion

This project successfully transformed a batch image classification use case into a real-time streaming model using Spark Streaming and Kafka on GCP. By setting up GCS, VMs, and Dataproc clusters, we created a real-time image processing pipeline that streams image metadata and provides predictions using TensorFlow. Despite challenges in configuring Kafka and firewall rules, the pipeline was successfully tested, producing real-time predictions for uploaded images. This solution demonstrates the potential of cloud computing and streaming technologies for scalable real-time analytics.