Question : Write a Python program to count the lines of a file that is placed in GCS using Google Cloud Functions.

Submitted By: Shramana Sinha, 23F1002703

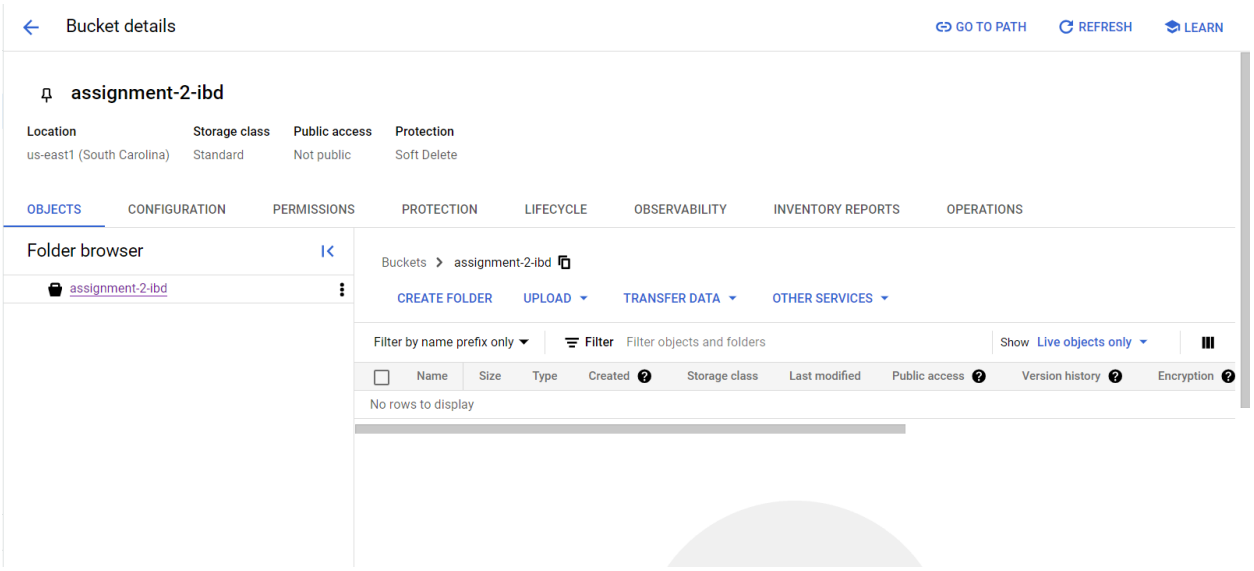
1. GCS Bucket Creation

A Google Cloud Storage (GCS) bucket was created to store input files and the output generated by the Cloud Function.

Bucket Name: `assignment-2-ibd`

Storage Settings: Default settings applied

Data Storage Location: `us-east1`



Screenshot 1: GCS bucket

2. Cloud Function Setup

A Cloud Function was created for this assignment.

Region: `us-east1`

Trigger Type: Cloud Storage (`google.cloud.storage.object.v1.finalized`)

Runtime: Python 3.12

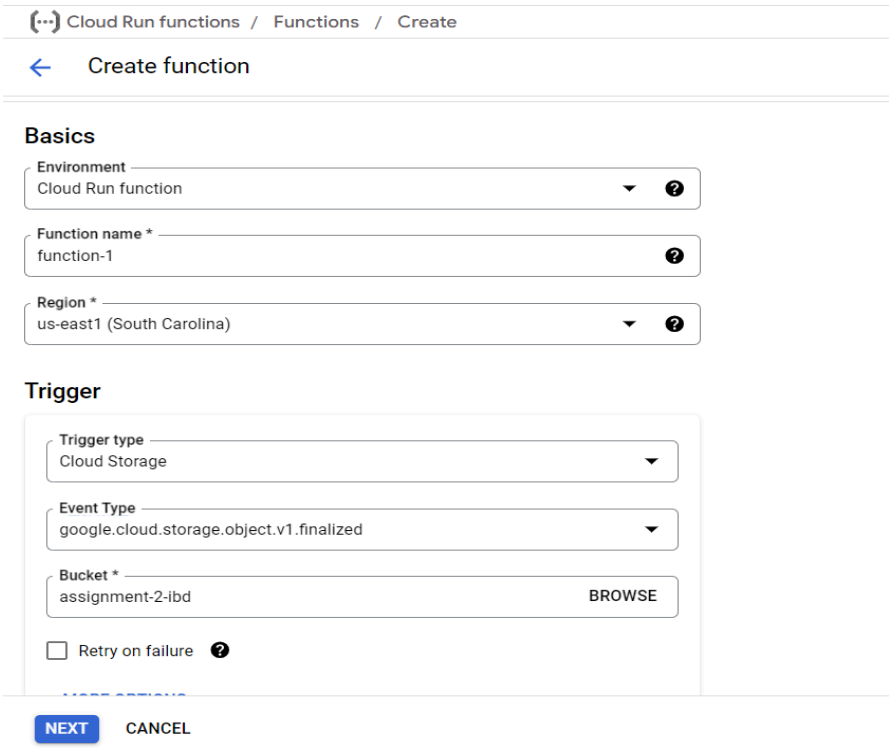
Entry Point: `count_lines`

Source Code: Inline Editor

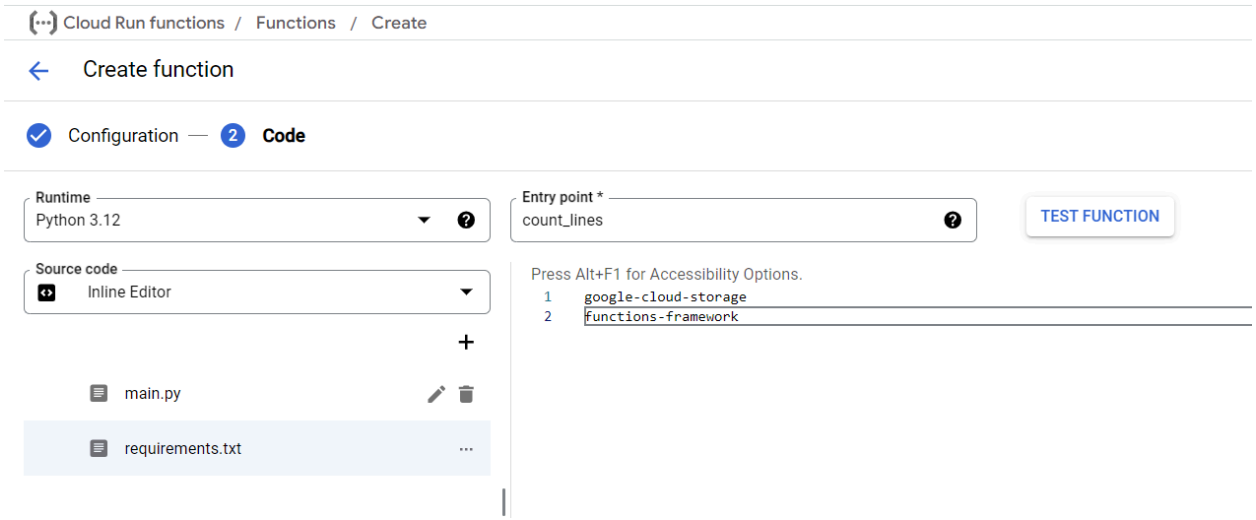
Dependencies:

```
Unset
google-cloud-storage
functions-framework
```

These dependencies were specified in the `requirements.txt` file.



Screenshot 2: Cloud Function basic settings



Screenshot 3: Cloud Function code settings

3. Code Explanation

A. Import the necessary libraries:

```python
import functions_framework
from google.cloud import storage

@functions_framework.cloud_event
def count_lines(cloud_event):
    # Function implementation
```

a. The `@functions_framework.cloud_event` decorator indicates that this function handles Cloud Events.

b. The function receives a `cloud_event` parameter containing information about the GCS event.

B. Extract Event Data: Extracts bucket name and file name from the extracted event data.

```python
data = cloud_event.data
```

```
bucket_name = data["bucket"]
file_name = data["name"]
```

C. **Skip Processing for output.txt**: Prevents recursive processing by skipping the output file.

```
Python
if file_name == "output.txt":
    print(f"Skipping {file_name} to avoid recursion")
    return "Skipped output file"
```

D. **File Processing**:

```
Python
storage_client = storage.Client()
bucket = storage_client.bucket(bucket_name)
blob = bucket.blob(file_name)
content = blob.download_as_text()
line_count = len(content.splitlines())
```

a. Initialize the GCS client using `storage.Client()`.
b. Get the specified bucket and file using the extracted names.
c. Download the file content as text using `blob.download_as_text()`.
d. Split the text content into a list of lines using `splitlines()`.
e. Count the number of lines using `len()`.

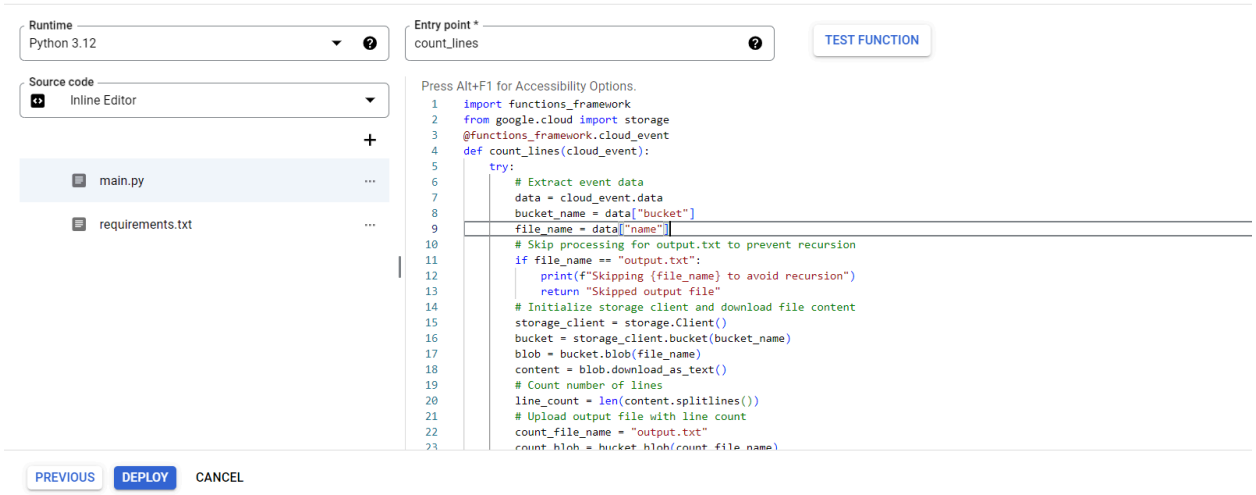
E. **Create Output File**:

```
Python
count_file_name = "output.txt"
count_blob = bucket.blob(count_file_name)
```

```
count_blob.upload_from_string(
    f"Number of lines in gs://{bucket_name}/{file_name}: {line_count}"
)
```

a. Creates a new blob named `output.txt`.

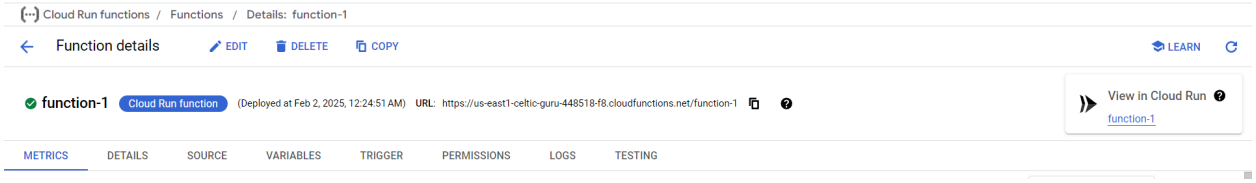
b. Uploads the line count information.



Screenshot 4: Cloud Function code editor

4. Deployment and Testing

A. Deployed the function using the Google Cloud Console.

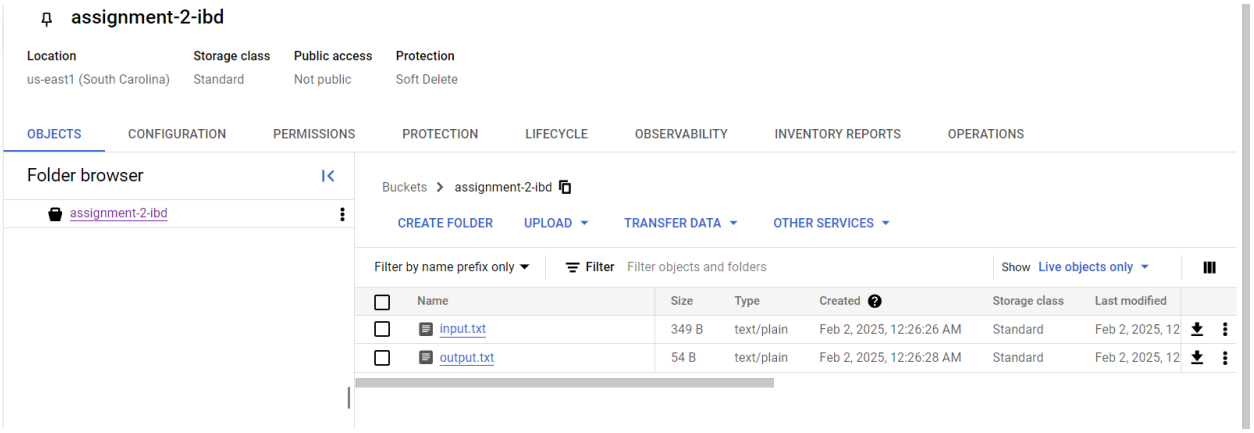


Screenshot 4: Successful deployment message

B. Uploaded a test file to the `assignment-2-ibd` bucket.

C. Verified that the Cloud Function executed successfully.

D. Confirmed that `output.txt` was created and contained the correct line count.



Screenshot 5: GCS bucket showing uploaded file and output.txt

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
This is line 11.
This is line 12.
This is line 13.
This is line 14.
This is line 15.
This is line 16.
This is line 17.
This is line 18.
This is line 19.
This is line 20.
```

```
Number of lines in gs://assignment-2-ibd/input.txt: 20
```

Screenshot 6: Comparison between the actual number of lines in the input file (top) and the line count output provided by the function (bottom).