DATAFLOW

Project Title: Running WordCount Pipeline in Google Cloud Dataflow

Abstraction:

In this project, we aim to leverage the capabilities of Google Cloud Dataflow alongside Apache Beam SDK to build an efficient data transformation pipeline in the cloud environment. By enabling the Dataflow API and assigning appropriate IAM roles, we ensure seamless access and management of resources. The installation of Apache Beam SDK in Cloud Shell facilitates the development and execution of data processing tasks. Cloud Storage buckets are utilized for storing input and output data securely. The execution of an example WordCount pipeline showcases the functionality of Dataflow in transforming data at scale. Monitoring tools provided by Dataflow enable real-time tracking of job performance, ensuring efficient resource allocation. Upon job completion, the output data can be easily accessed and analyzed. Through this project, we demonstrate the power and flexibility of cloud-based data processing solutions for modern data-driven applications.

Introduction:

This project aims to harness the capabilities of Google Cloud Dataflow and Apache Beam SDK to build a robust data transformation pipeline in the cloud environment. By seamlessly integrating these technologies, we enable organizations to efficiently process large volumes of data while taking advantage of the scalability and flexibility offered by the cloud.

Tools and Technologies used:

- **Google Cloud Platform:** Google Cloud Platform (GCP) is a comprehensive suite of cloud computing services provided by Google, offering infrastructure, platform, and software services for businesses and developers. GCP provides scalable and reliable resources that can be dynamically provisioned to meet fluctuating demand, ensuring high availability and performance. Security is a top priority for GCP, with robust features such as encryption, identity and access management, and built-in compliance controls to protect customer data. GCP's global network infrastructure ensures low-latency and high-speed connectivity across regions, facilitating

efficient data transfer and application deployment. With a wide range of services for big data analytics, machine learning, and artificial intelligence, GCP empowers organizations to derive valuable insights and drive innovation from their data. GCP offers transparent and competitive pricing models, allowing customers to pay only for the resources they use and providing tools for cost optimization and budget management.

- **Dataflow:** Google Cloud Dataflow is a fully managed service for executing and managing data processing pipelines, offering both batch and stream processing capabilities. Developed by Google and based on Apache Beam SDK, Dataflow provides a unified programming model for building complex data processing workflows that can scale dynamically to handle massive datasets. With Dataflow, developers can focus on writing business logic without worrying about infrastructure management, as the service automatically handles resource provisioning, scaling, and optimization. Dataflow offers advanced features such as windowing, watermarking, and stateful processing, enabling the processing of real-time streaming data with high accuracy and efficiency. The service integrates seamlessly with other Google Cloud Platform services such as BigQuery, Pub/Sub, and Cloud Storage, facilitating data ingestion, transformation, and analysis across the entire data pipeline.

- **Apache Beam:** Apache Beam is an open-source unified programming model for defining both batch and streaming data processing pipelines. It provides a high-level API that abstracts the complexities of distributed computing, allowing developers to write data processing logic in a language-agnostic manner. Apache Beam supports multiple programming languages, including Java, Python, and Go. The core abstraction in Apache Beam is the "PCollection," which represents a distributed dataset that can be processed in parallel across multiple nodes in a cluster. Apache Beam's model is designed to be portable, allowing pipelines to run on various execution engines, including Apache Flink, Apache Spark, Google Cloud Dataflow, and others. Apache Beam provides a rich set of built-in transformations and connectors for common data processing tasks, such as filtering, aggregating, joining, and reading from/writing to different data sources. Apache Beam simplifies the development and deployment of data processing pipelines, accelerating time to insights and enabling organizations to leverage their data effectively for business intelligence and decision-making.

Description:

The project involves the following key steps:

- Enable Dataflow in a Google Cloud project.
- Add the necessary IAM roles to the Compute Engine default service account.
- Install the Apache Beam SDK in Cloud Shell to let Dataflow transform data for the pipelines.
- Set up a Cloud Storage bucket for the output data.
- Run the example WordCount pipeline in Dataflow as a job.
- Monitor your job in Dataflow.
- View the output of your job.
- Clean up to avoid billing.

Text File Description: This is how, the input text file looks like and it is named as "word-count.txt"

```
This is a Japanese doll
The team members were hard to tell apart since they all wore their hair in a ponytail
As the years pass by we all know owners look more and more like their dogs
If you don't like toenails you probably shouldn't look at your feet
He was disappointed when he found the beach to be so sandy and the sun so sunny
When he encountered maize for the first time he thought it incredibly corny
Situps are a terrible way to end your day
Toddlers feeding raccoons surprised even the seasoned park ranger
Edith could decide if she should paint her teeth or brush her nails
Her daily goal was to improve on yesterday
Tomorrow will bring something new so leave today as a memory
His son quipped that power bars were nothing more than adult candy bars
He wondered why at 18 he was old enough to go to war but not old enough to buy cigarettes
If my calculator had a history it would be more embarrassing than my browser history
The hummingbird's wings blurred while it eagerly sipped the sugar water from the feeder
He went on a whiskey diet and immediately lost three days
This is the last random sentence I will be writing and I am going to stop mid-sent
I come from a tribe of head-hunters so I will never need a shrink
The delicious aroma from the kitchen was ruined by cigarette smoke
Weather is not trivial - it's especially important when you're standing in it
She had a difficult time owning up to her own crazy self
Gary didn't understand why Doug went upstairs to get one dollar bills when he invited him to go cow tipping
He loved eating his bananas in hot dog buns
Peanut butter and jelly caused the elderly lady to think about her past
We have never been to Asia nor have we visited Africa
A dead duck doesn't fly backward
Having no hair made him look even hairier
The secret ingredient to his wonderful life was crime
He played the game as if his life depended on it and the truth was that it did
Baby wipes are made of chocolate stardust
He is no James Bond his name is Roger Moore
A quiet house is nice until you are ordered to stay in it for months
His thought process was on so many levels that he gave himself a phobia of heights
```

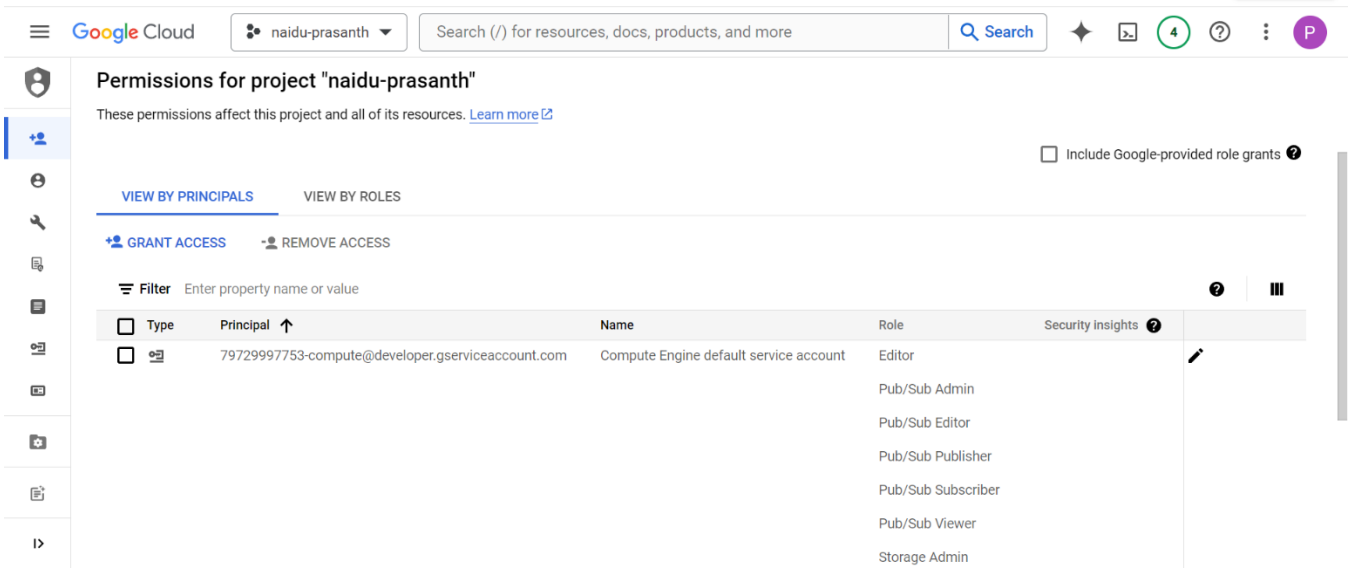
Project Workflow:

Step 1: Enabling Dataflow API

- Access GCP Console: Navigate to the GCP Console (https://console.cloud.google.com/).
- Open APIs & Services: In the navigation bar, select APIs & Services.
- Enable Dataflow API:
  - ➢ Click on Library.
  - ➢ Search for Dataflow API.
  - ➢ Enable the Dataflow API for your project.

Step 2: Granting IAM Permissions

- The Compute Engine default service account requires specific roles to run Dataflow jobs.
- Assign the following roles to the Compute Engine Default service account:
  - ➢ Dataflow Worker
  - ➢ Storage Object Admin



Step 3: Installing Apache Beam SDK (Cloud Shell)

- Open Cloud Shell: In the GCP Console, navigate to the top right corner and click on the "Activate Cloud Shell" button.
- Install Beam SDK: Use the package manager for your chosen language to install the Apache Beam SDK (e.g., Python: pip install apache-beam[gcp]).

Step 4: Creating a Cloud Storage Bucket

A **bucket** is a fundamental concept in **Google Cloud Storage**. Here are the key points:
Purpose of Buckets:

- Buckets serve as the **basic containers** for storing data in Cloud Storage.
- **Everything** you store in Cloud Storage must be contained within a bucket.
- Unlike directories or folders, **buckets cannot be nested** inside one another.
- There is **no limit** to the number of buckets you can have in a project or location.

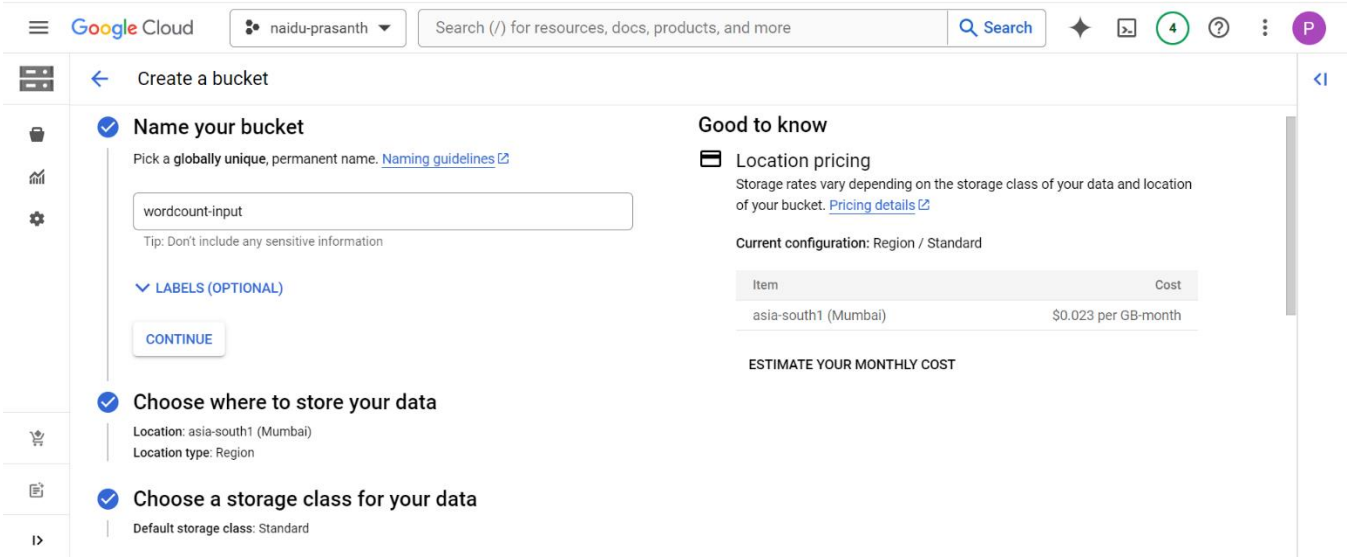
Bucket Properties:

- When you create a bucket, you give it a **globally-unique name** and specify a **geographic location** where the bucket and its contents will be stored.
- You cannot change the name or location of an existing bucket directly. Instead, you can create a new bucket with the desired properties and move the contents from the old bucket to the new one.
- The **pricing** for buckets depends on factors such as location and storage classes of objects within them.

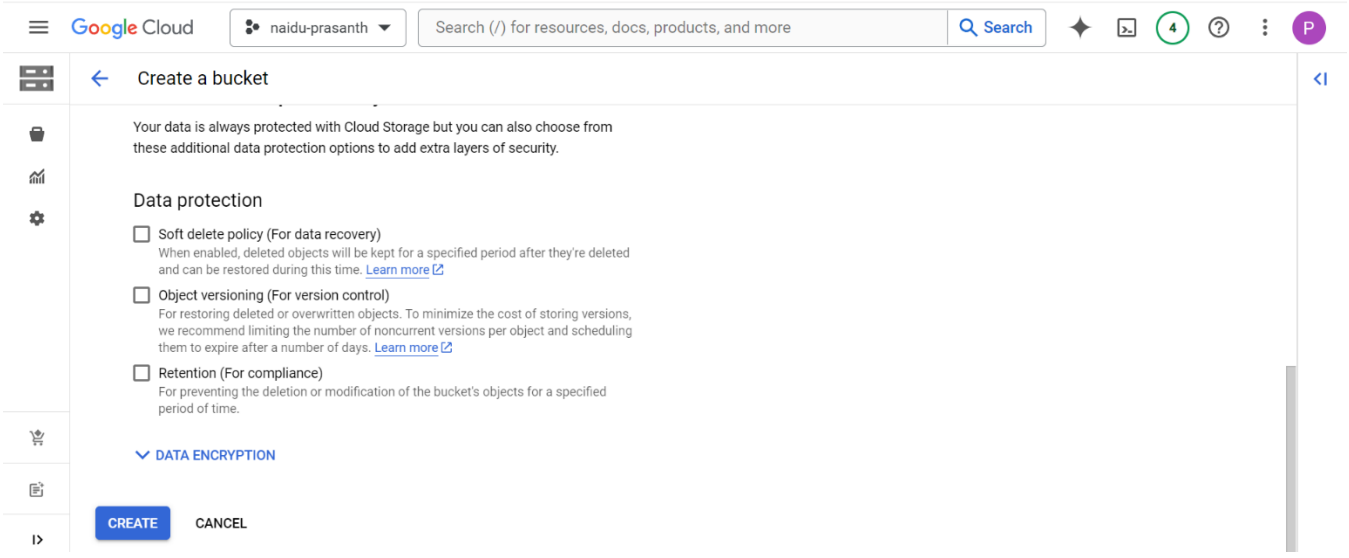
1. **Access Control**:
    - You can use **Identity and Access Management (IAM)** to control access to individual buckets.
    - IAM allows you to manage permissions for users and services accessing the bucket.
2. **Bucket Naming Rules**:
    - Bucket names must meet specific requirements:
        - Only **lowercase letters**, **numeric characters**, **dashes**, **underscores**, and **dots** are allowed.
        - Spaces are **not allowed**.
        - Names containing dots require **verification**.
        - Bucket names must start and end with a **number or letter**.
        - Bucket names must contain **3-63 characters**.
        - Names containing dots can have up to **222 characters**, but each dot-separated component cannot exceed 63 characters.
        - Bucket names cannot be represented as an **IP address**.

        - Bucket names cannot begin with the prefix "**goog**".
        - Avoid using "**google**" or similar misspellings.
    - Bucket names are **publicly visible**, so avoid using personally identifiable information (PII) or sensitive data.

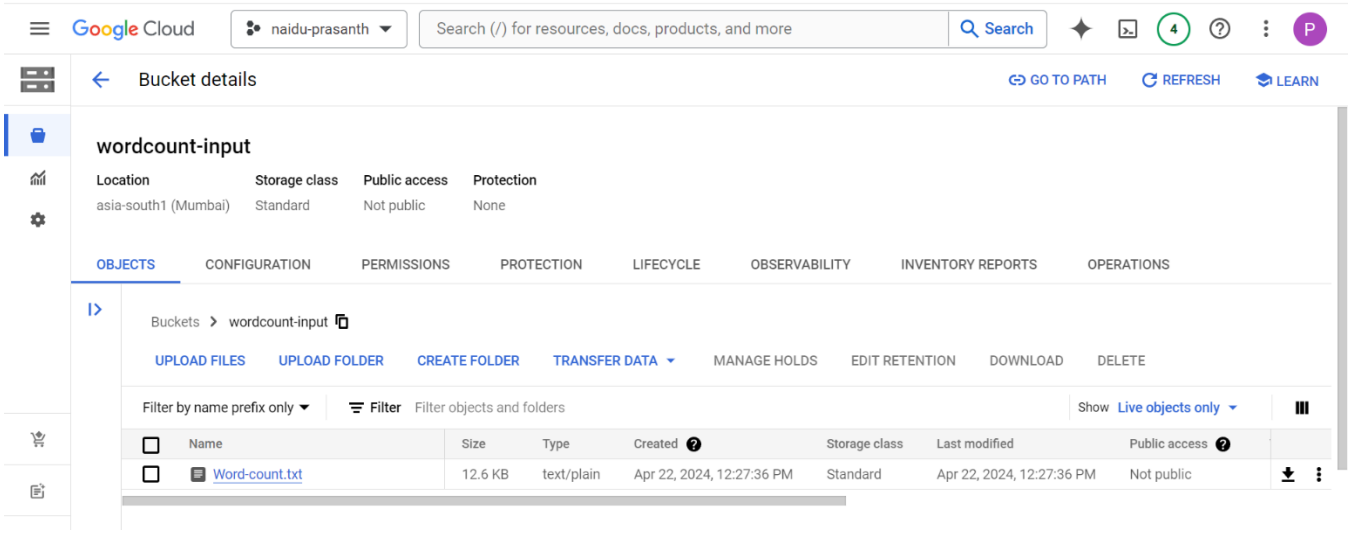
- Access Storage Section: In the navigation bar, select Storage.
- Create Bucket: Click on "Create bucket".
- Configure Bucket: Choose a unique bucket name and region, then click "Create".



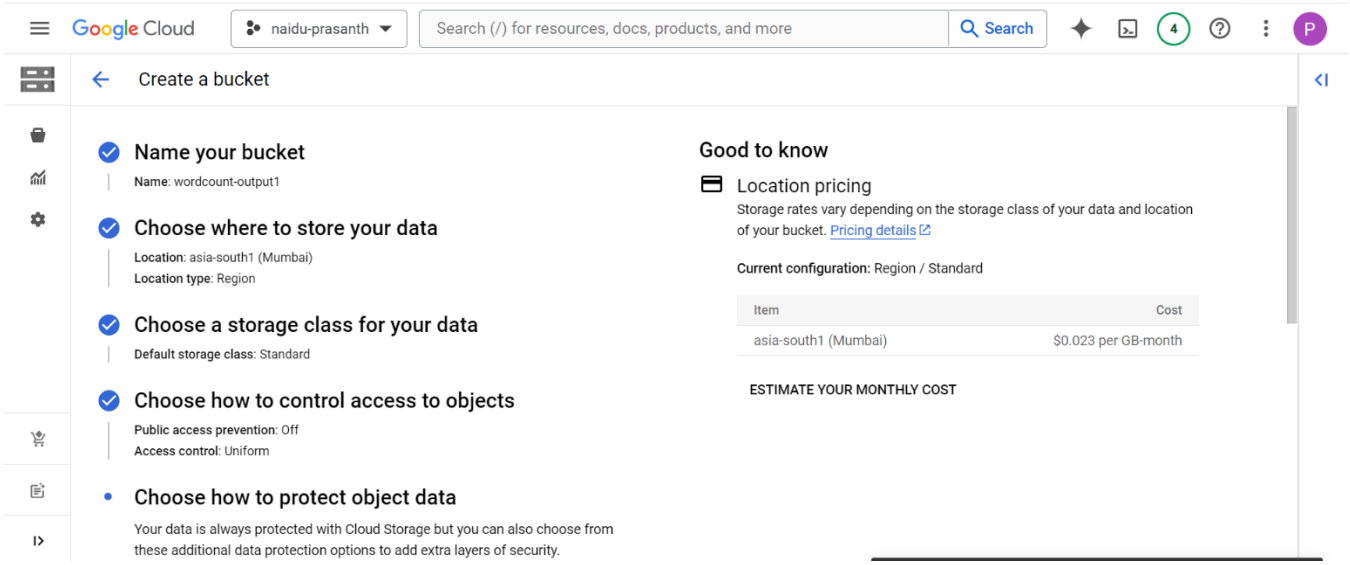
Here we are creating input bucket named “wordcount-input” to store the text file on which we are going to perform wordcount.



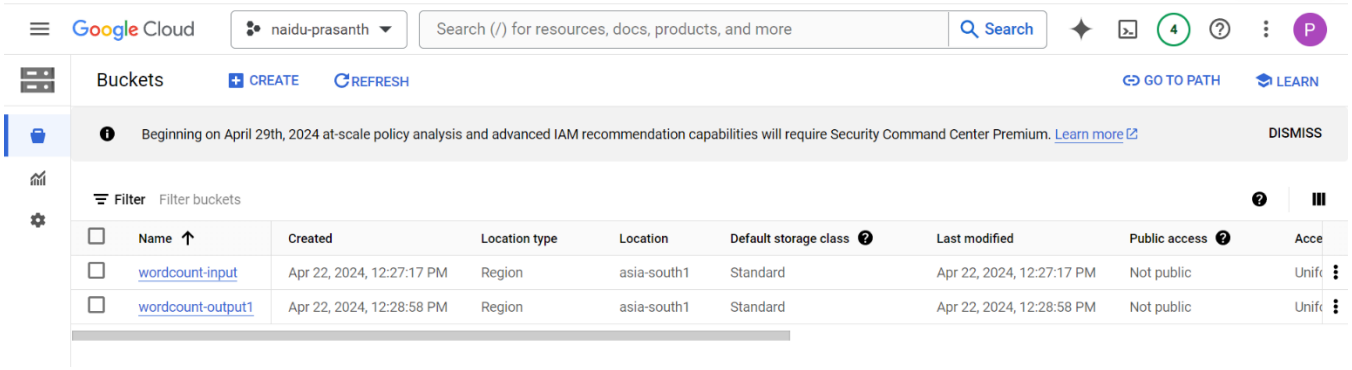
Now we are going to upload text file named “word-count.txt” to the “wordcount-input” bucket.



Now we are going to create output bucket named “wordcount-output1” to store the result.



Now we can verify both the input and output buckets which we created earlier.

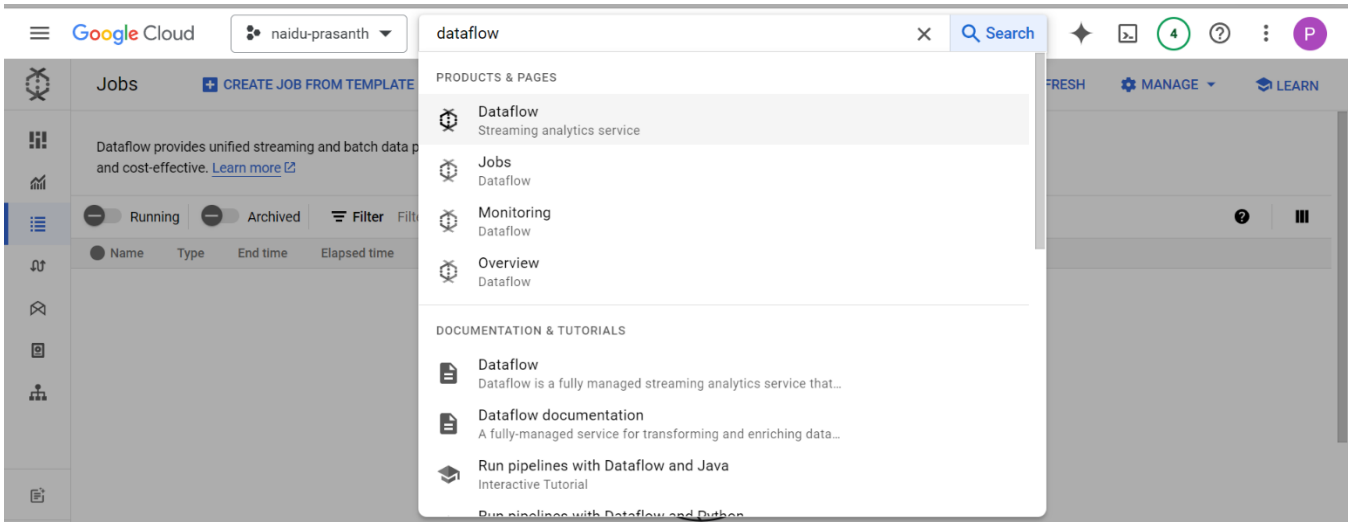


Step 5: Running the WordCount Pipeline

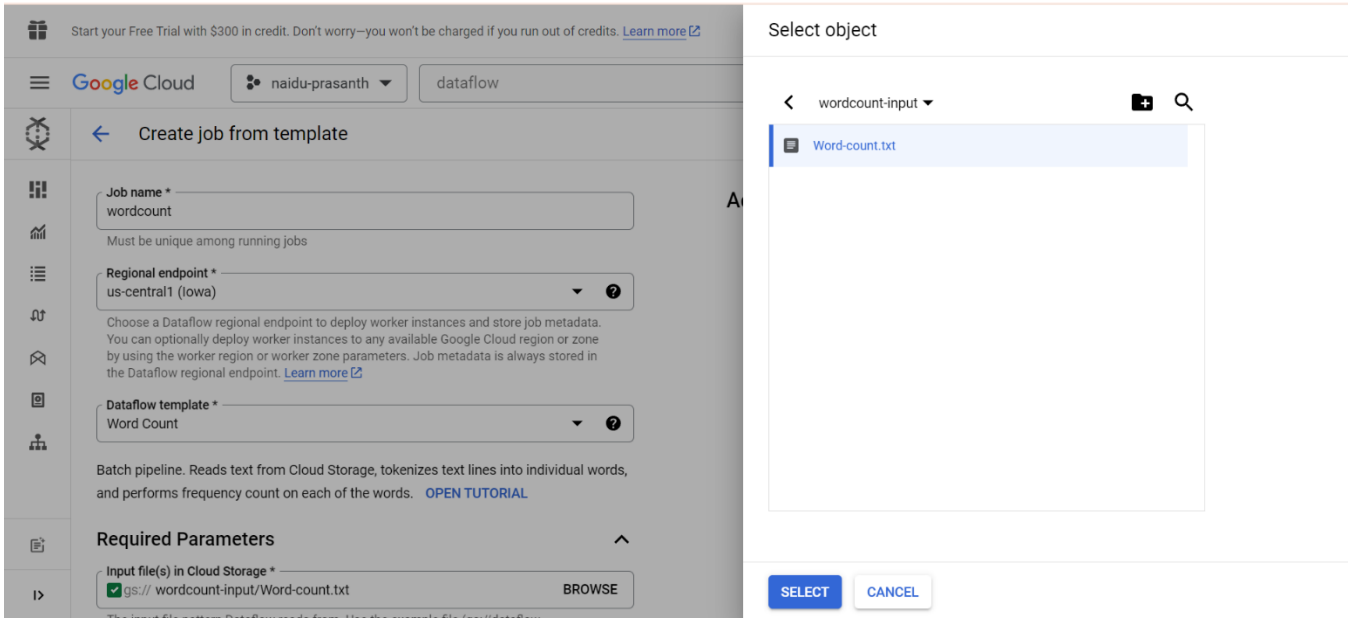
There are two options to run the WordCount pipeline:

Option A: Using a Dataflow Template

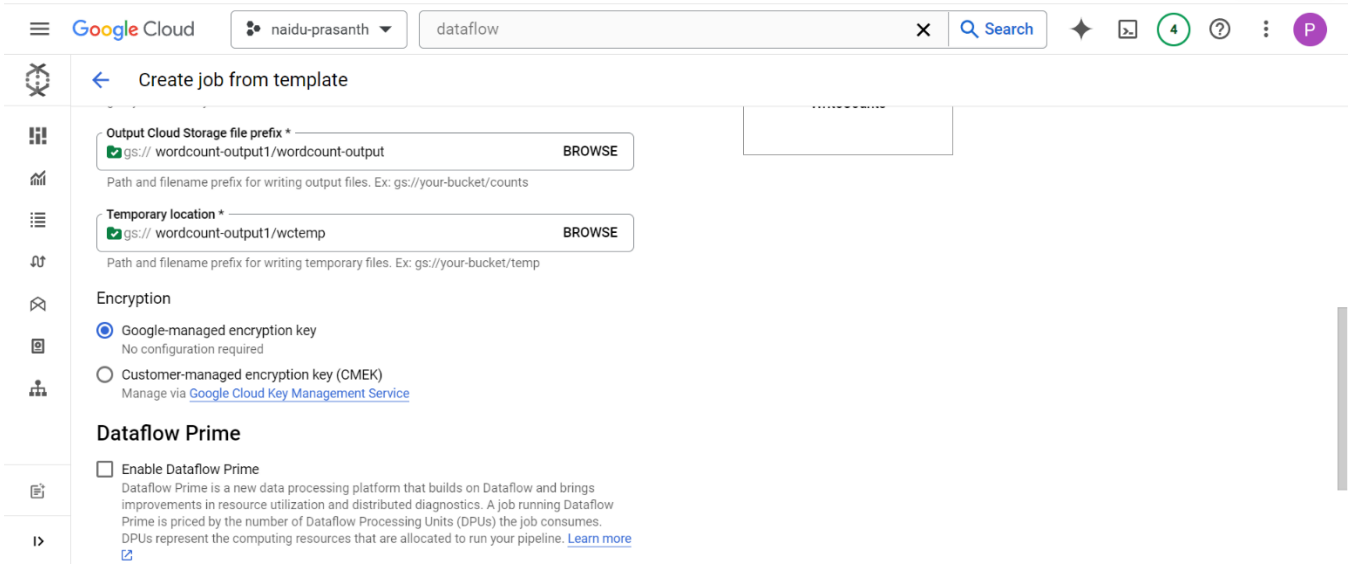
- Access Dataflow Section: In the navigation bar, select Dataflow



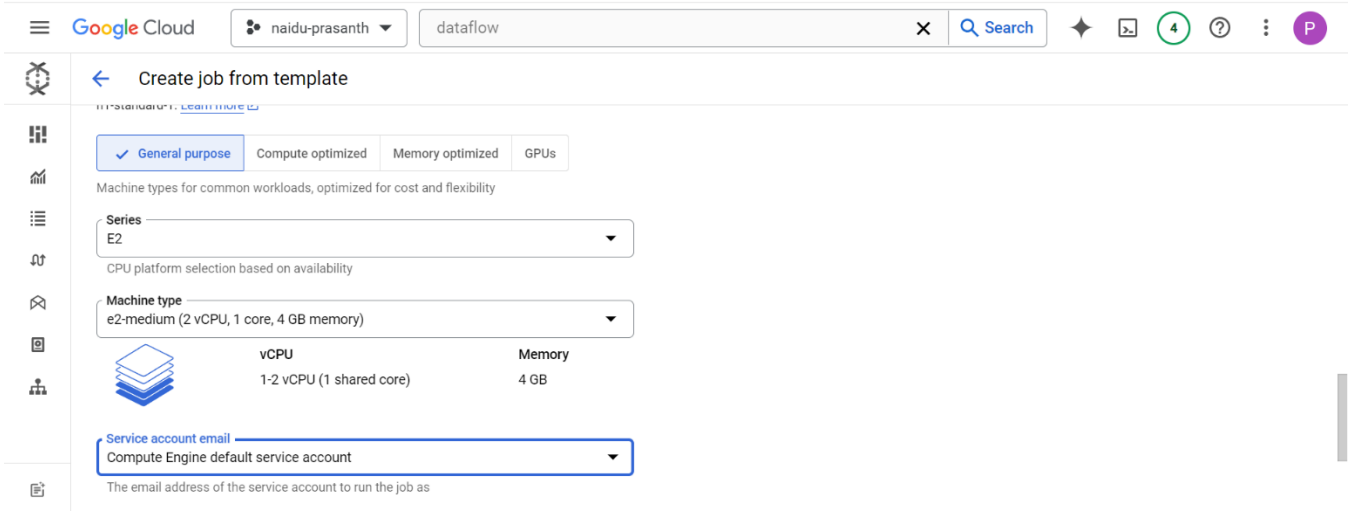
- Create Job: Click on "Create job".
- Select Template: Choose "Create job from template".
- Choose WordCount Template: Select "WordCount" from the template dropdown menu.
- Choose the input text file to perform WordCount.



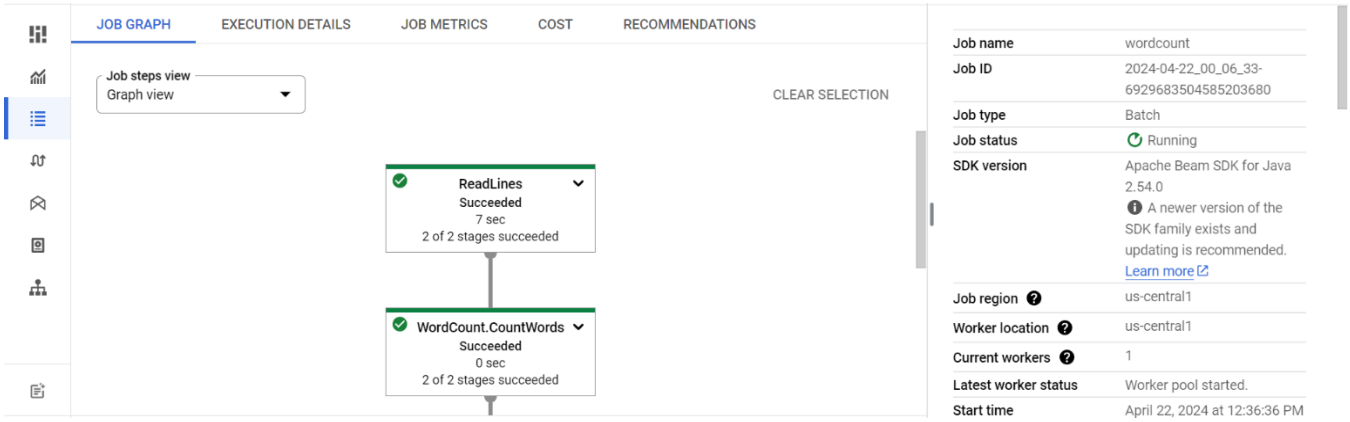
- Configure Parameters: Configure the template parameters (e.g., input and output locations).



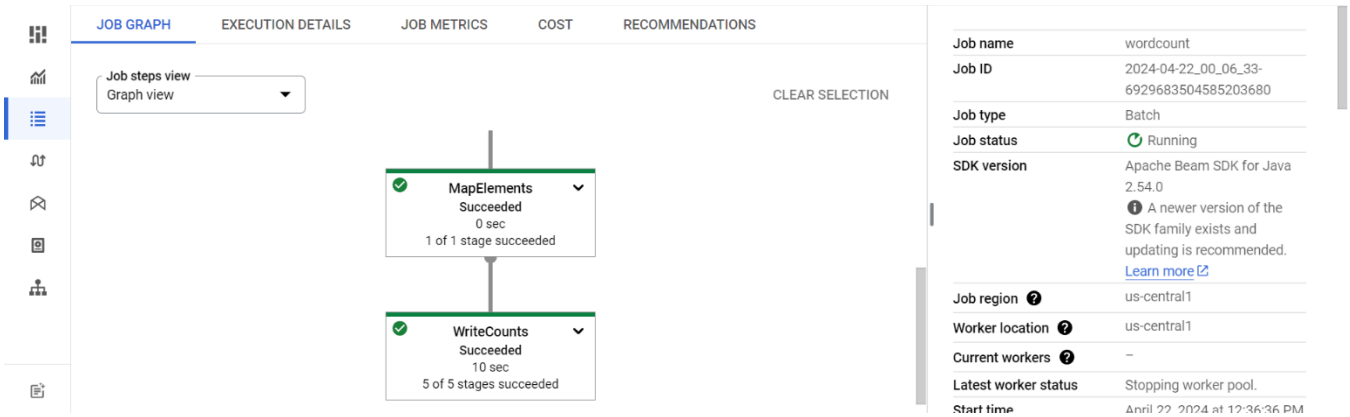
- Here we customized the optional parameters.



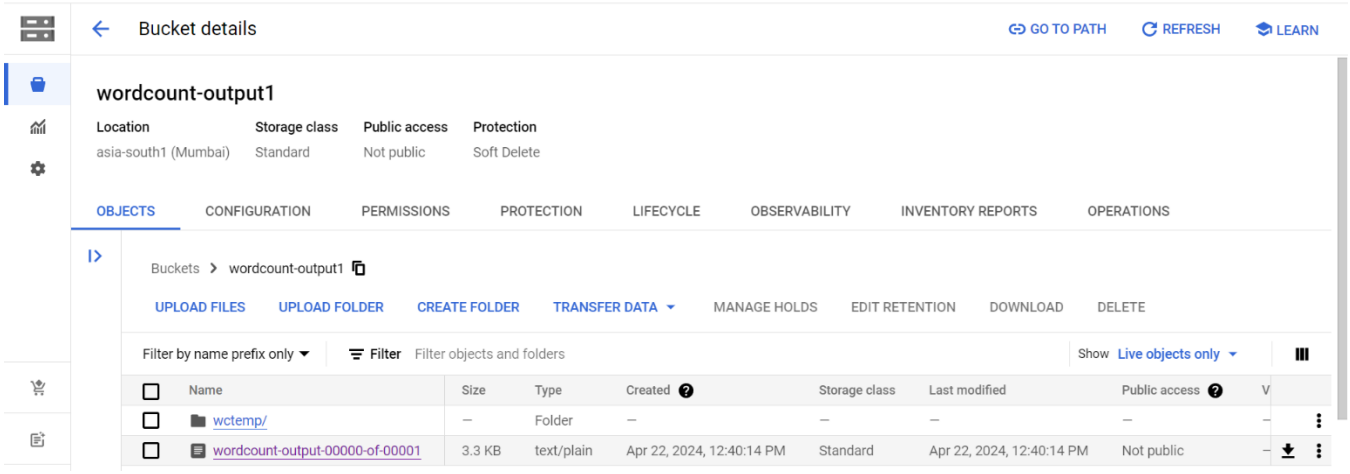
- Run the Job: Click "Run job".

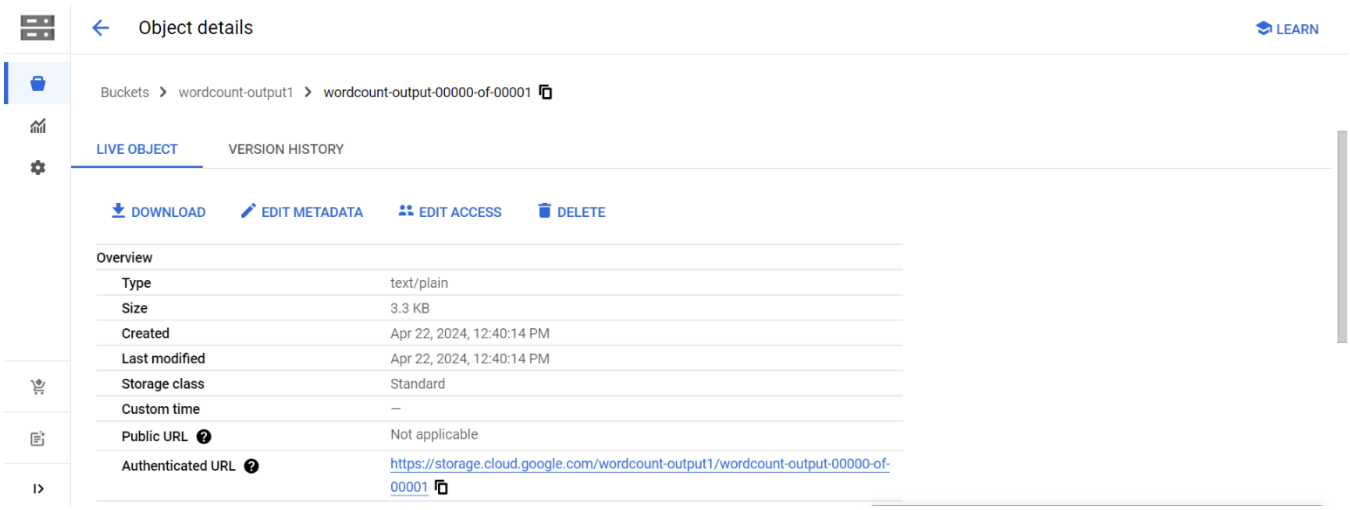


- Here we can see that, the Dataflow job get succeeded.



- We can see the output of the above Dataflow job in “wordcount-output1” bucket.





- This is how the output looks like.

```
she: 8
yesterday: 4
mean: 4
didn't: 4
feeder: 4
Japanese: 4
nor: 4
from: 16
said: 4
been: 4
it's: 4
tipping: 4
The: 32
Africa: 4
created: 4
shrink: 4
outside: 4
lost: 4
book: 4
wipes: 4
crackers: 4
Please: 4
could: 8
whether: 4
Weather: 4
depended: 4
find: 4
in: 24
consider: 4
into: 4
preoccupied: 4
enough: 8
phone: 4
doll: 4
hopes: 4
feet: 4
stop: 8
```

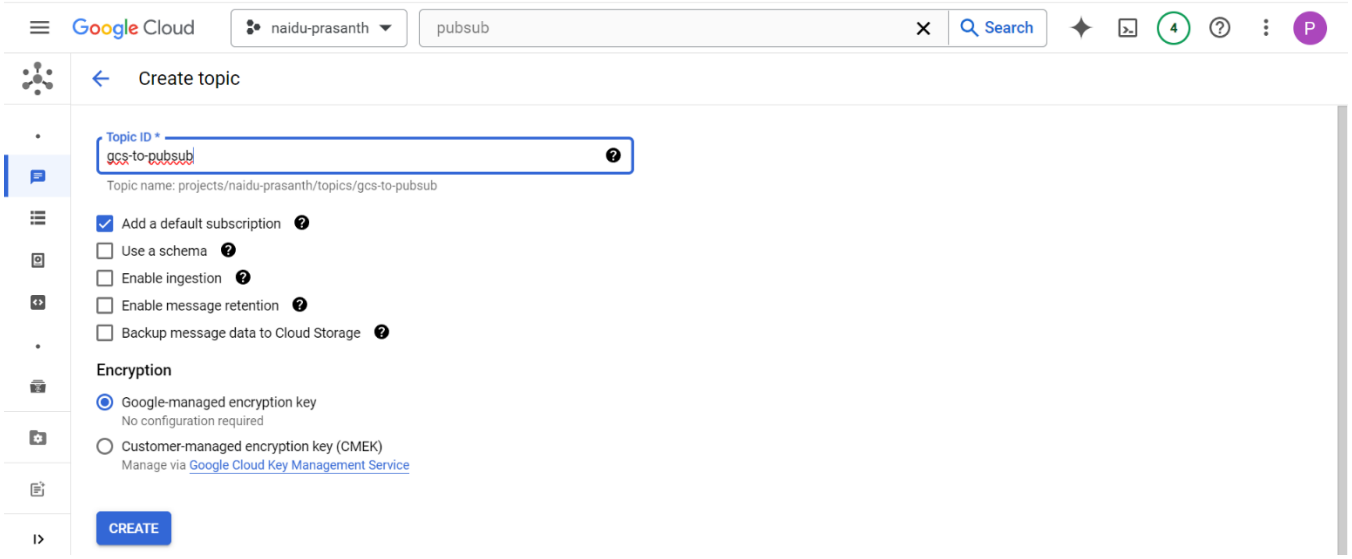
Step 6: Going to run the Pub/Sub Pipeline (Additional work)

Now we are going to connect the wordcount output to Pub/Sub.

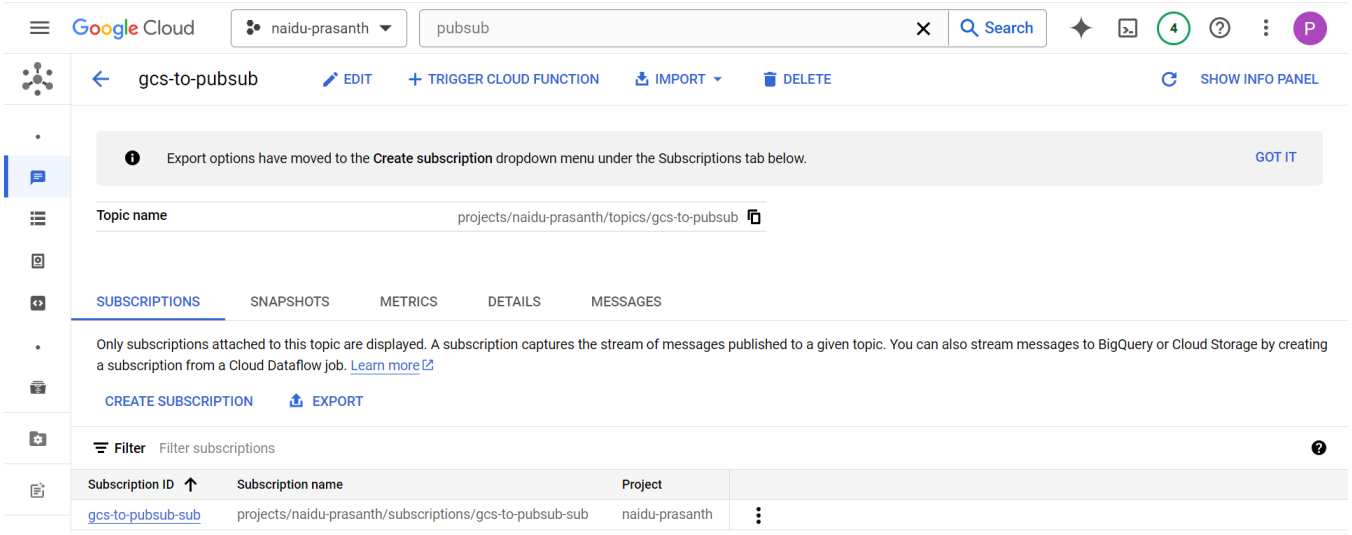
Pub/Sub, short for publish-subscribe, is an asynchronous messaging system used for communication between different parts of an application.

Pub/Sub provides a flexible and reliable way for different parts of a distributed system to communicate asynchronously, without tight coupling or waiting times.

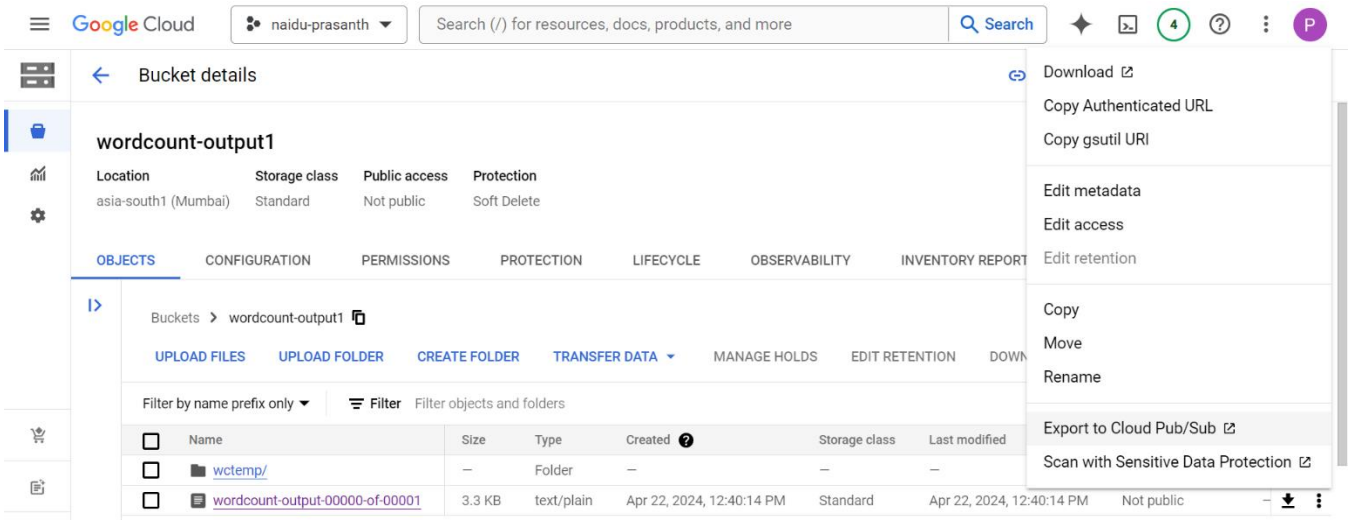
- For that, first we need to create a “Topic” and “Subscription” in Pub/Sub



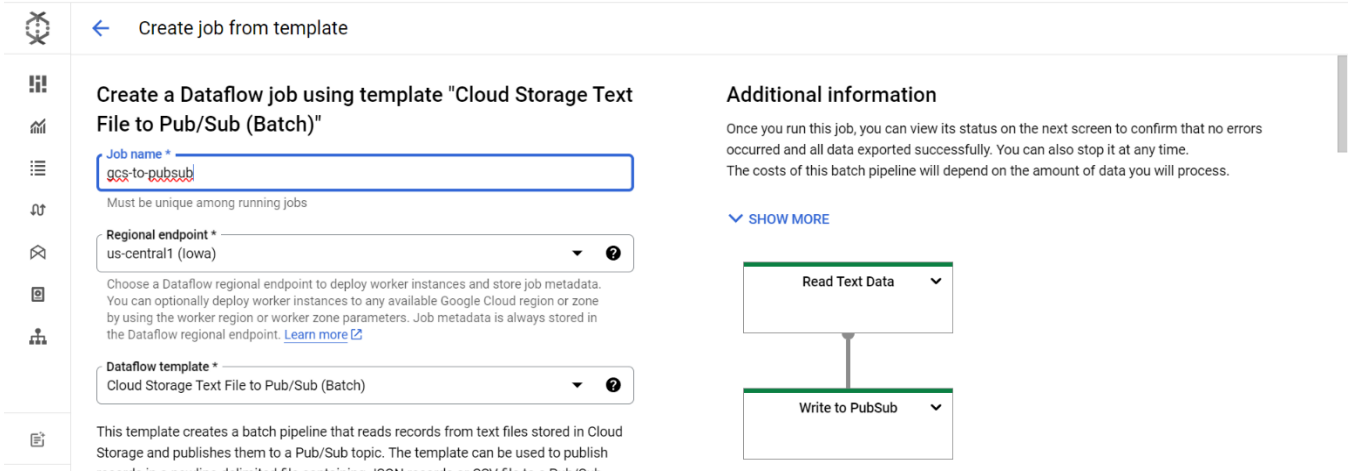
- Here our Topic ID is “gcs-to-pubsub” and Subscription ID is “gcs-to-pubsub-sub” and we can verify that below.



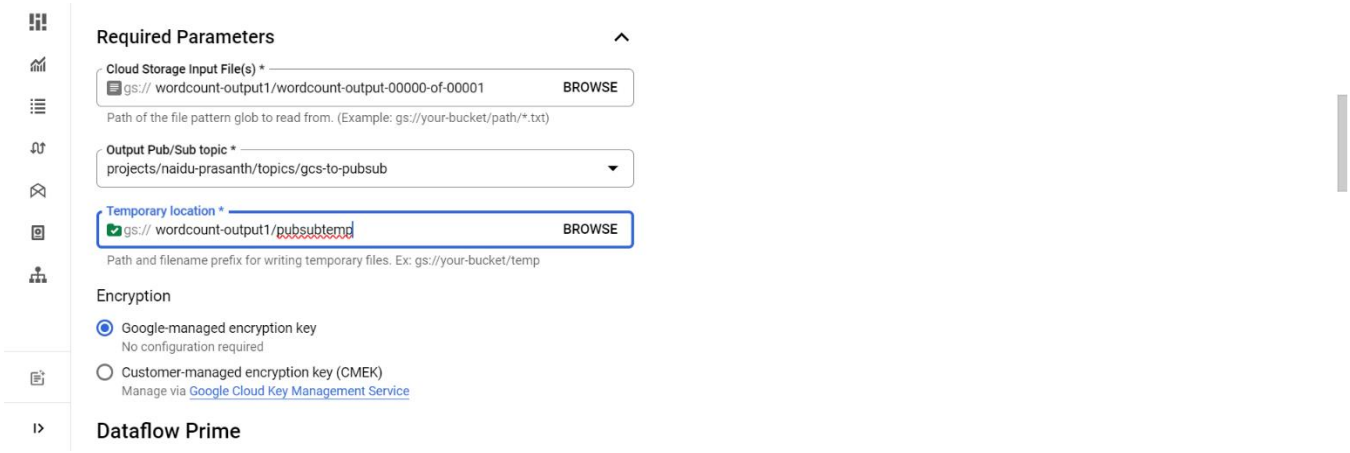
- Now we have to go to the output file of the wordcount and click on “Export to cloud Pub/Sub”



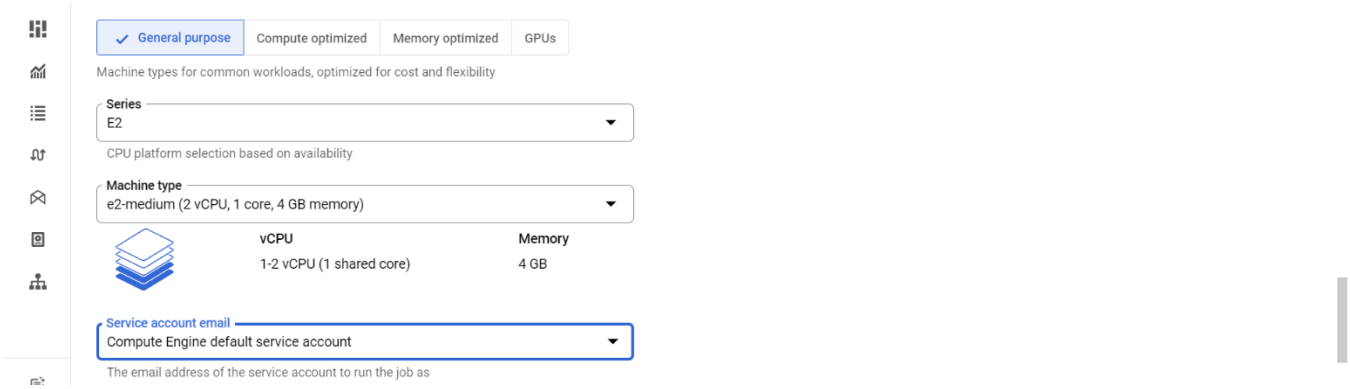
- In next step it will automatically redirect to Dataflow service and it will select the Dataflow template by default.



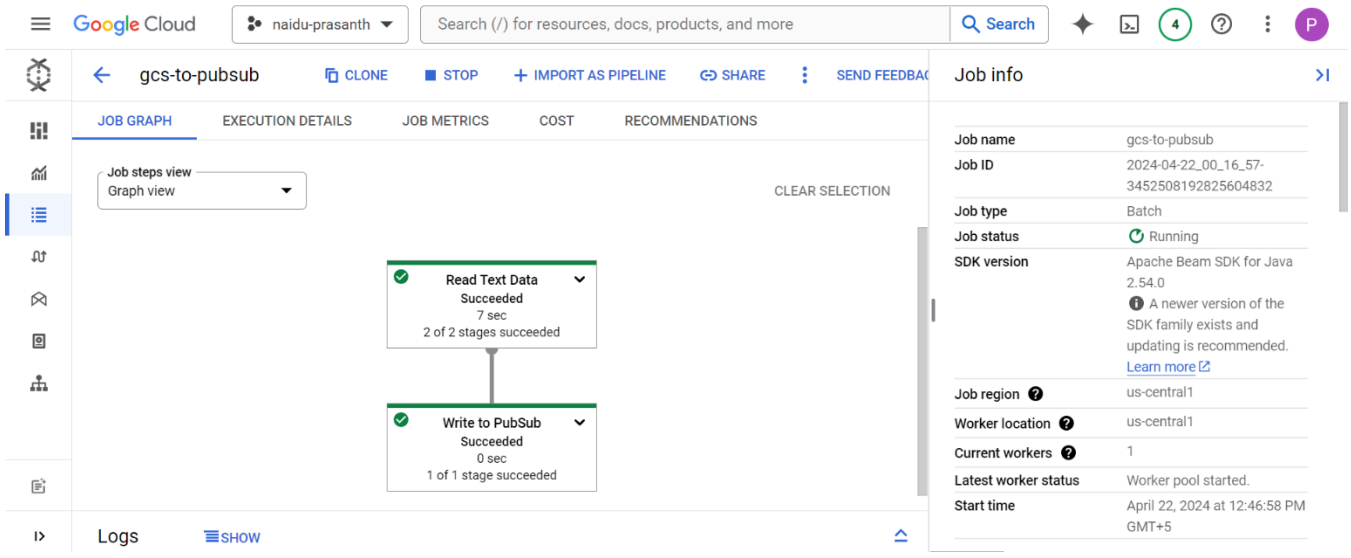
- Now we have to give the Pub/Sub topic which we created earlier and give the temporary location.



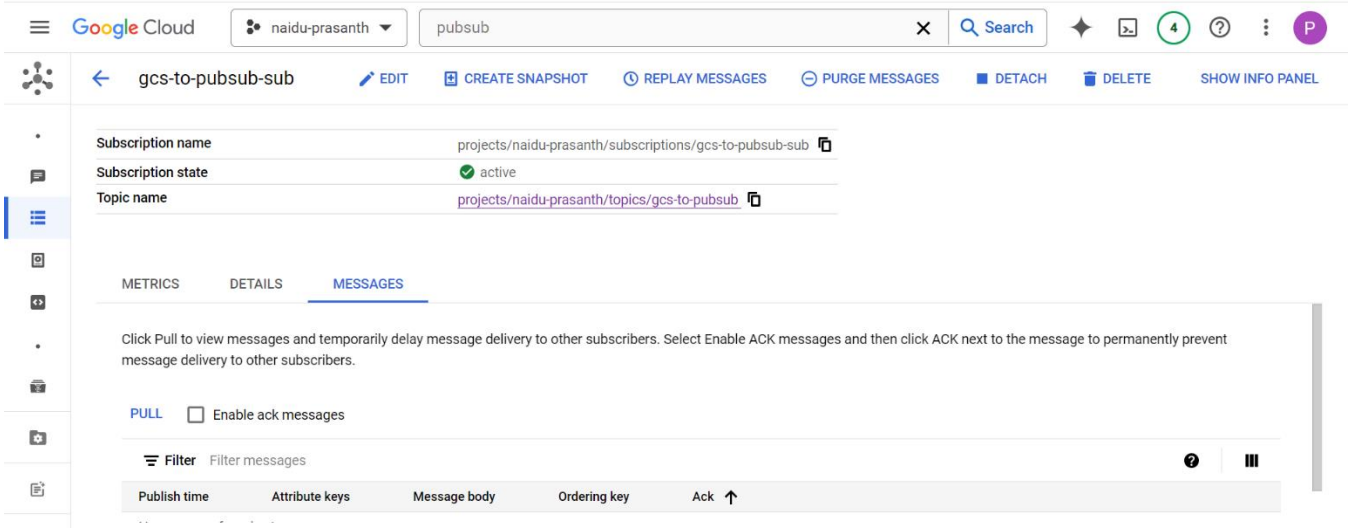
- Now customize the optional parameters.



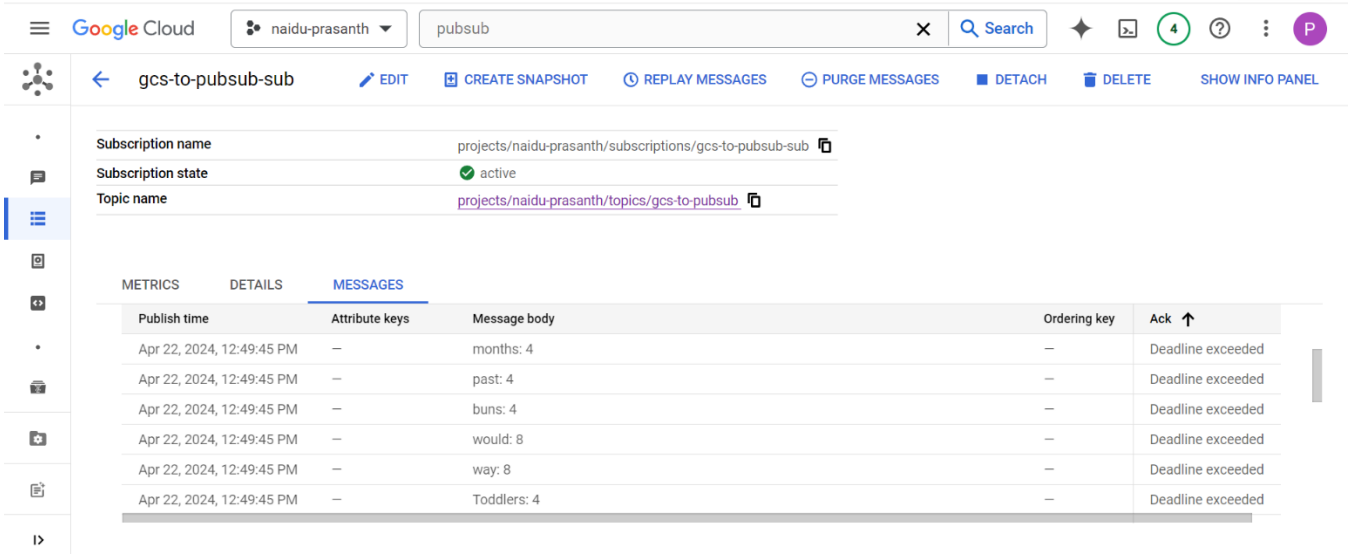
- Click on run job and verify whether Dataflow job get succeeded or not.



- We can verify it by navigating to Pub/Sub subscription and Pull the message.

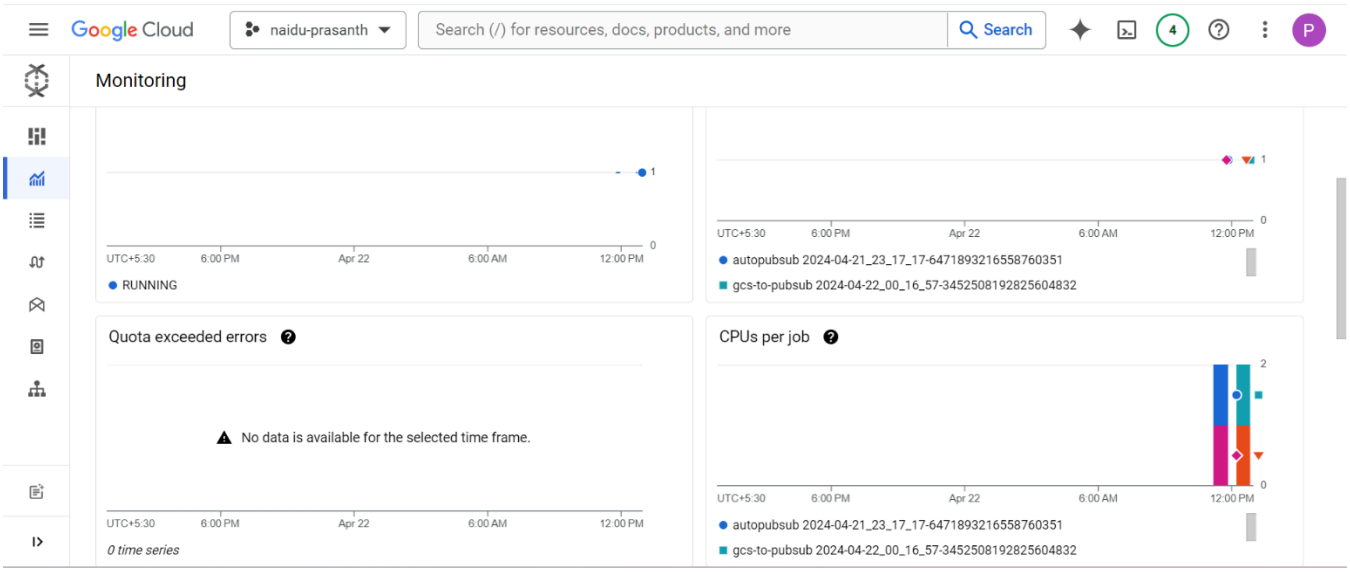


- Finally, we can verify the output.



Step 7: Monitoring Your Job

- Access Dataflow Section: In the navigation bar, select Dataflow.
- View Jobs List: You'll see a list of your Dataflow jobs.
- View Job Details: Click on the specific WordCount job to view details like status, logs, and metrics.



Deleting buckets and Pub/Sub topics after use serves several important purposes:

Resource Management and Cost Optimization:

- **Buckets**: When you delete a **Cloud Storage bucket**, it frees up the resources associated with it. This helps manage your cloud resources efficiently and reduces costs.
- **Pub/Sub Topics**: Deleting unused Pub/Sub topics ensures that you are not paying for unnecessary message handling. It's a way to optimize costs by cleaning up resources that are no longer needed.

Security and Data Privacy:

- **Buckets**: Deleting buckets prevents unauthorized access to data. Even if a bucket is no longer actively used, it may still contain sensitive information. Deleting it ensures that no one can accidentally or intentionally access the data.
- **Pub/Sub Topics**: Removing unused topics reduces the attack surface. If a topic is no longer in use, it's best to delete it to minimize potential security risks.

Maintaining a Clean Environment:

- **Buckets**: Regularly deleting unused buckets keeps your cloud environment tidy. It avoids clutter and makes it easier to manage and navigate your resources.
- **Pub/Sub Topics**: Similar to buckets, cleaning up unused topics ensures a streamlined Pub/Sub setup.

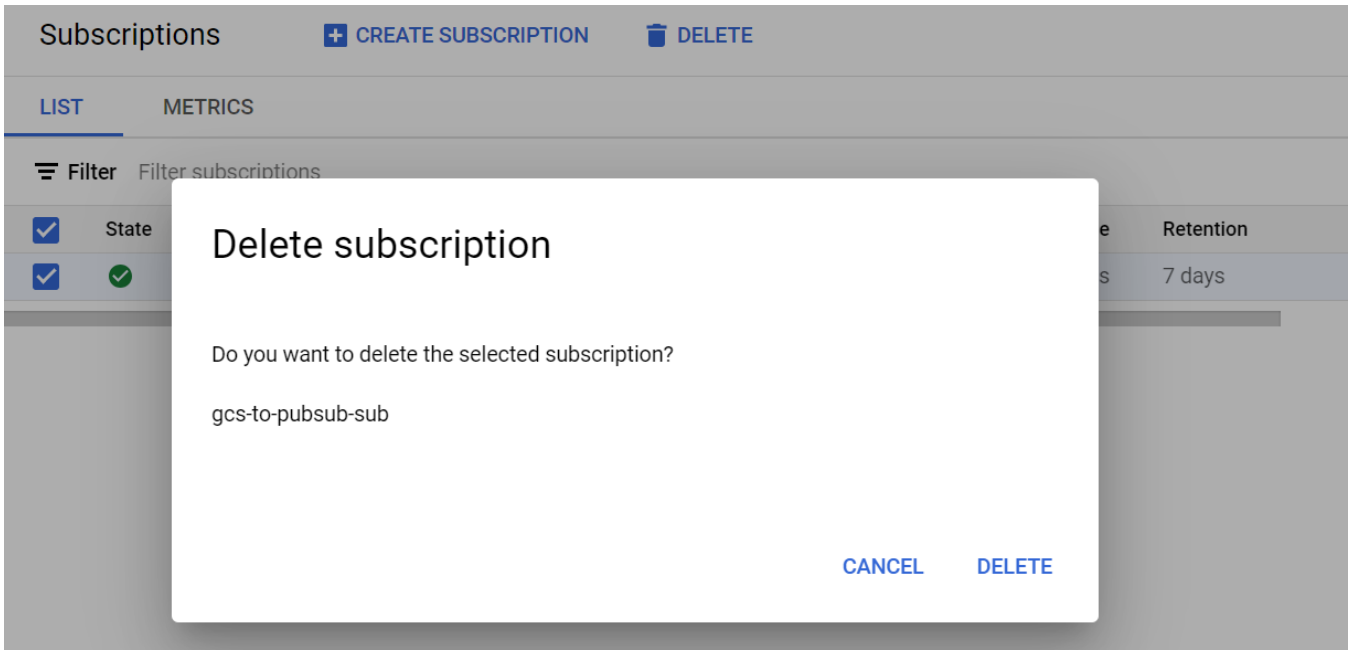
Avoiding Confusion and Accidental Usage:

- **Buckets**: If you have multiple buckets, leaving unused ones can lead to confusion. Developers might accidentally use the wrong bucket, causing unexpected behavior.
- **Pub/Sub Topics**: Unused topics can also confuse developers or result in unintended message delivery.

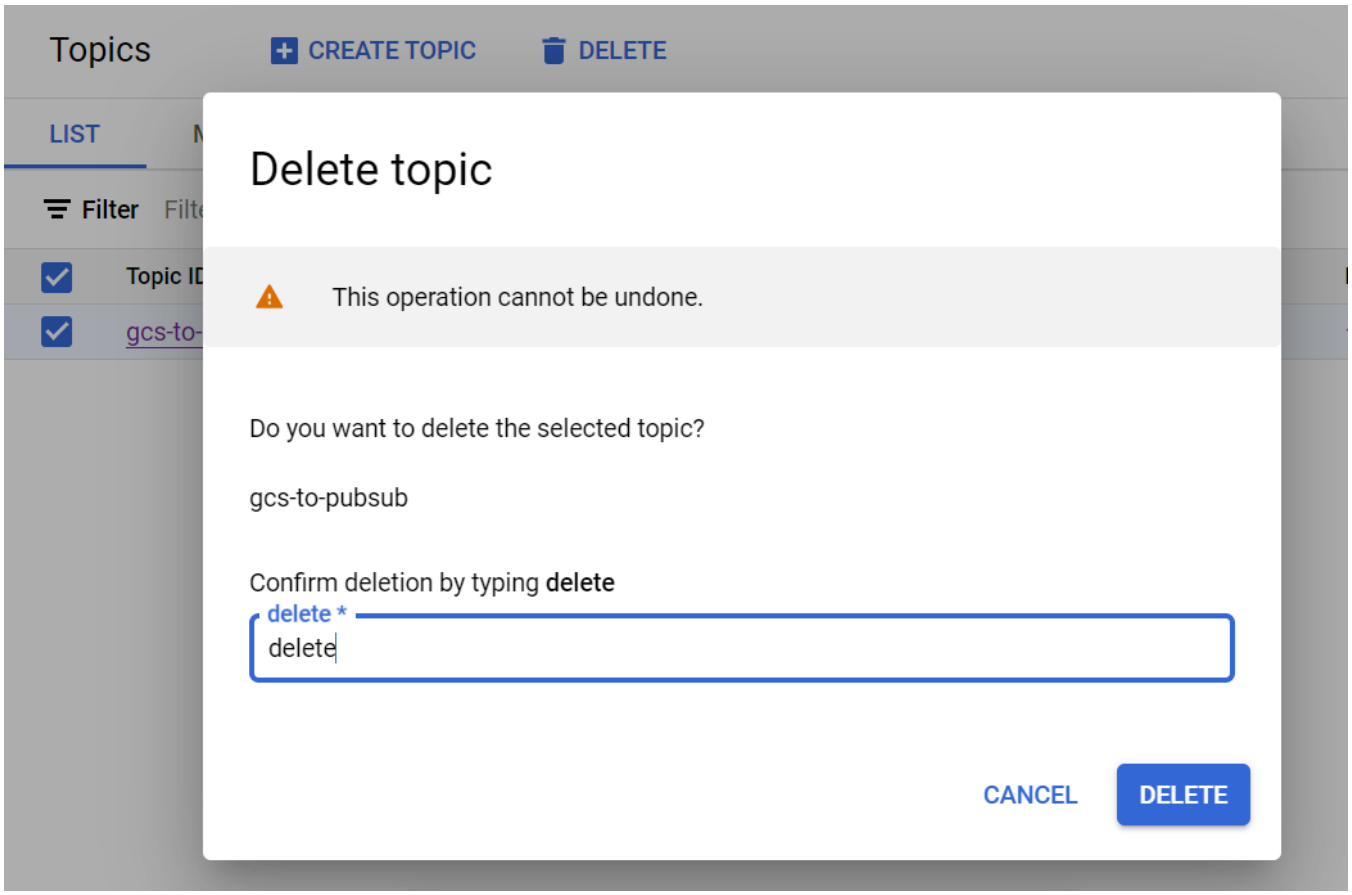
Best Practices:

- **Buckets**: Following best practices, such as deleting buckets after their lifecycle ends, ensures a well-organized and secure cloud storage environment.
- **Pub/Sub Topics**: Regularly reviewing and deleting topics aligns with good resource management practices.

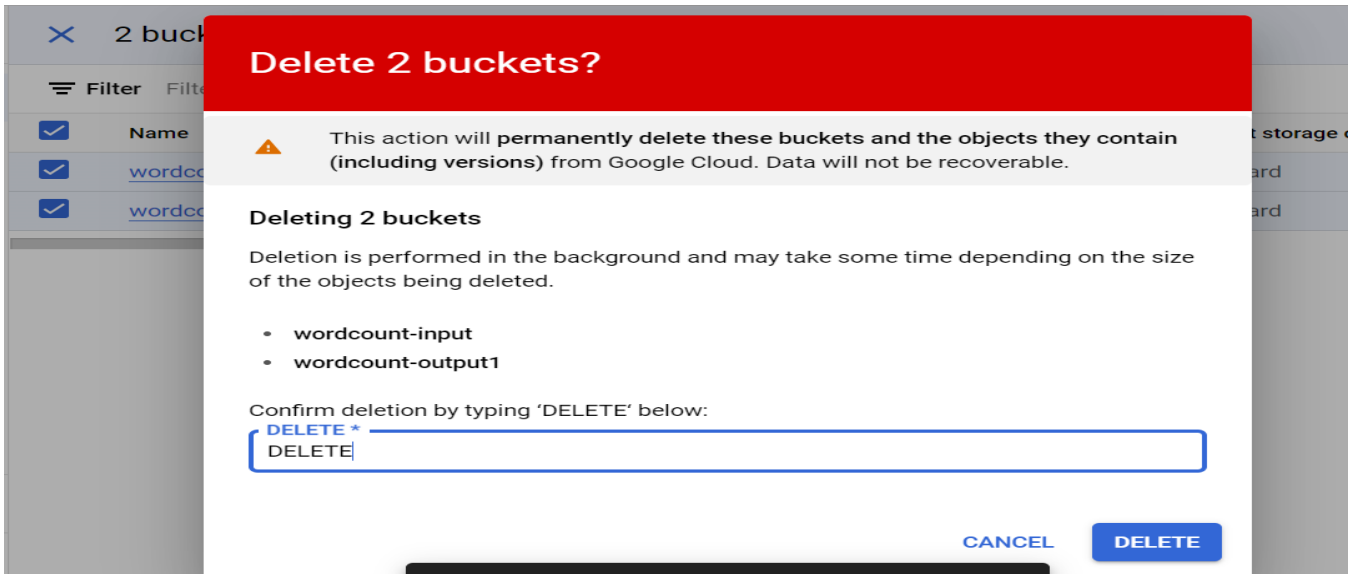
Deletion of subscription in pub/sub:-



Deletion of topic in pub/sub:-



Deletion of buckets:-

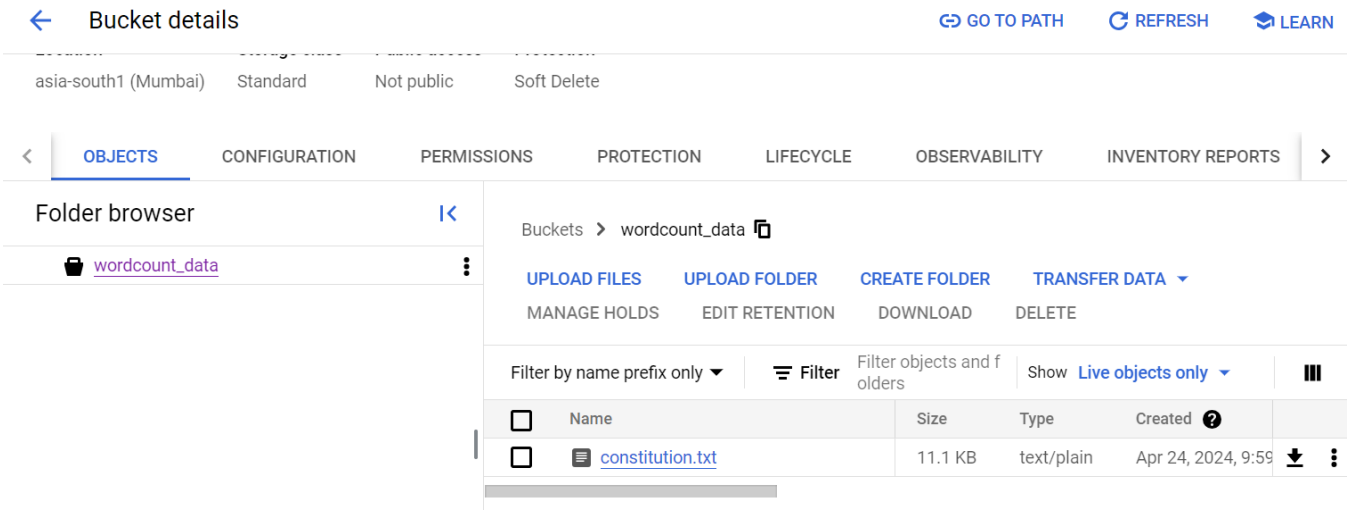


Using command shell

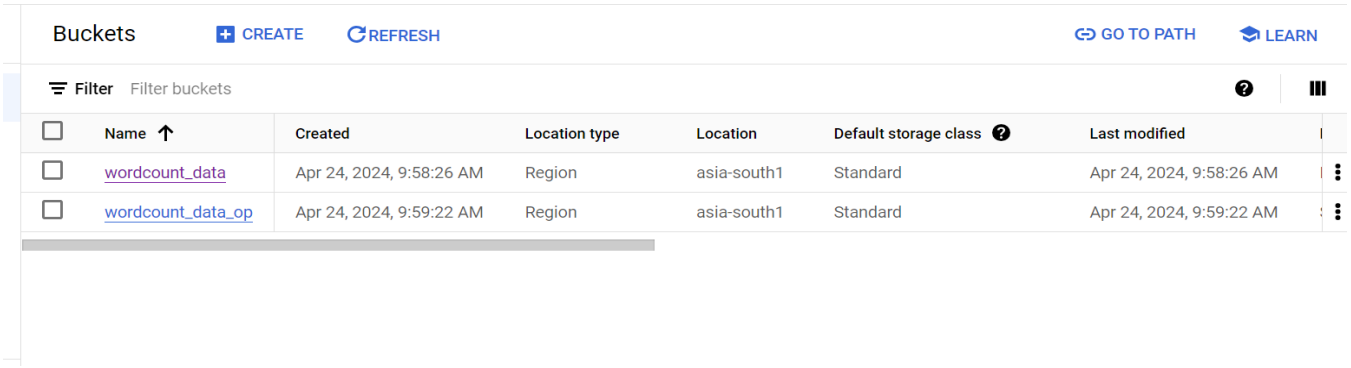
A **bucket** is a fundamental concept in **Google Cloud Storage**. Here are the key points:
Purpose of Buckets:

- Buckets serve as the **basic containers** for storing data in Cloud Storage.
- **Everything** you store in Cloud Storage must be contained within a bucket.
- Unlike directories or folders, **buckets cannot be nested** inside one another.
- There is **no limit** to the number of buckets you can have in a project or location.

Inserting file into bucket wordcount_data



Created output bucket wordcount_data_op for storing output data.



`pip3 install virtualenv`:

- This part of the command uses `pip3`, which is a package manager for Python.
- `pip3` is used to install Python packages and libraries.
- `virtualenv` is a tool that creates isolated Python environments.

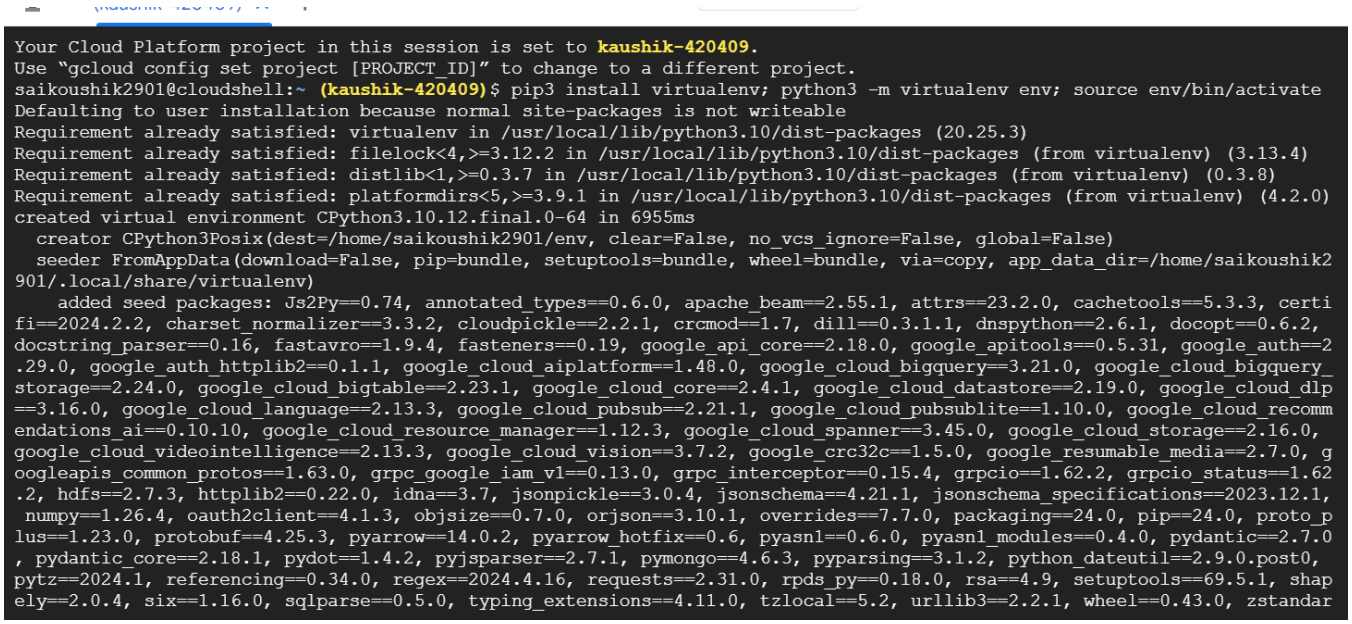
- When you run this command, it installs the `virtualenv` package globally on your system.

`python3 -m virtualenv env`:

- This part of the command creates a new **virtual environment** named `env`.
- A virtual environment is a self-contained directory where you can install Python packages independently of the system-wide Python installation.
- The `-m` flag specifies that we want to run the `virtualenv` module as a script.
- After executing this command, you'll have a new directory called `nv` containing a separate Python environment.

`source env/bin/activate`:

- This part of the command **activates** the virtual environment.
- When you activate the environment, it modifies your shell's environment variables to use the Python interpreter and packages from the virtual environment.
- You'll notice that your shell prompt changes to indicate that you are now working within the `env` virtual environment.
- This step is essential because it ensures that any Python packages you install or scripts you run will use the isolated environment rather than the system-wide Python installation.



`pip3`:

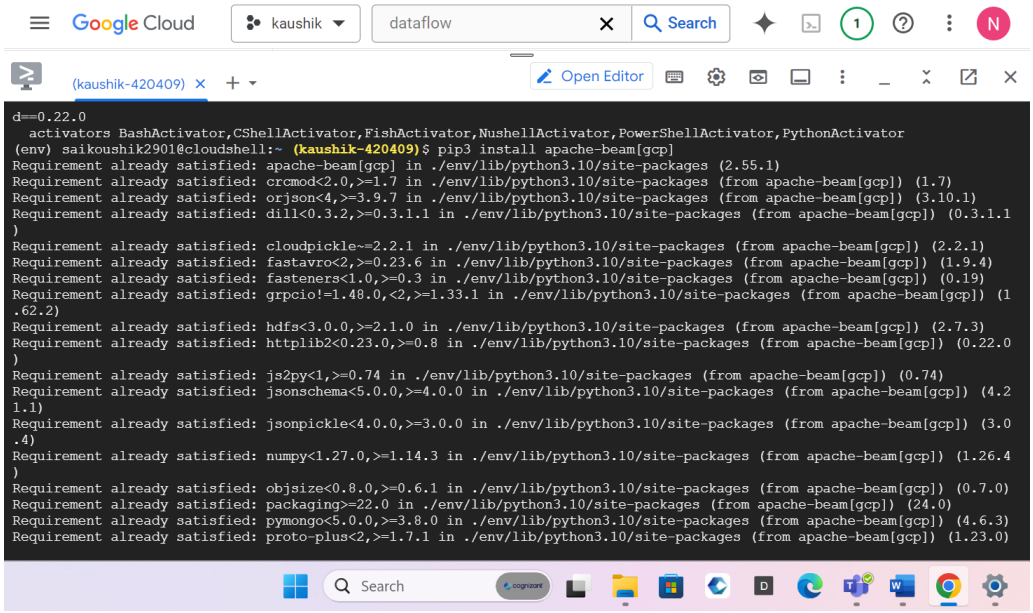
- `pip3` is a package manager for Python.
- It allows you to **install**, **upgrade**, and **manage** Python packages and libraries.

`install`:

- The `install` command is used to **install** Python packages.
- When you run `pip3 install <package-name>`, it fetches the specified package from the Python Package Index (PyPI) and installs it on your system.

`apache-beam`:

- Apache Beam is an open-source project that provides a unified programming model for both batch and streaming data processing.
- It enables efficient execution across diverse distributed execution engines (such as Apache Flink, Apache Spark, and Google Cloud Dataflow) and provides extensibility points for connecting to different technologies and user communities.
- In the context of Python, **`apache-beam`** refers to the Python SDK for Apache Beam.



`python3 -m apache_beam.examples.wordcount`:

- This part of the command runs a Python module using the `-m` flag.
- The module being executed is `apache_beam.examples.wordcount`.

The `wordcount` example is a common introductory example in Apache Beam, demonstrating how to count words in a text document.

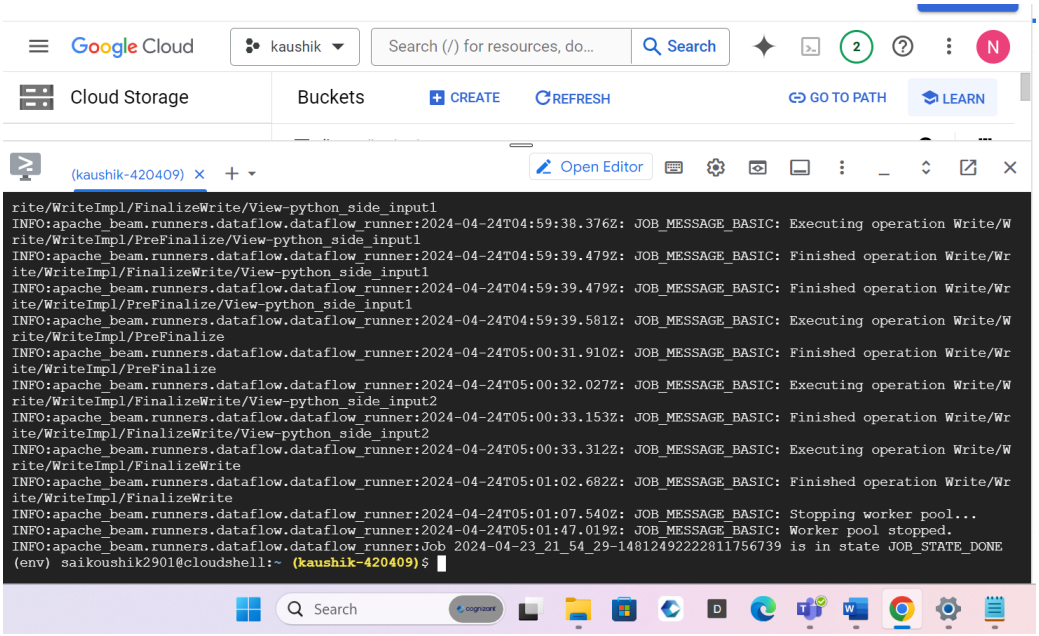
1. **`--region us-central1`:**
    - Specifies the **region** where the Dataflow job will run.
    - In this case, it's set to `us-central1`, which corresponds to the central United States.

2. **`--input gs://wordcount_data/constitution.txt`:**
    - Specifies the **input file** for the word count job.

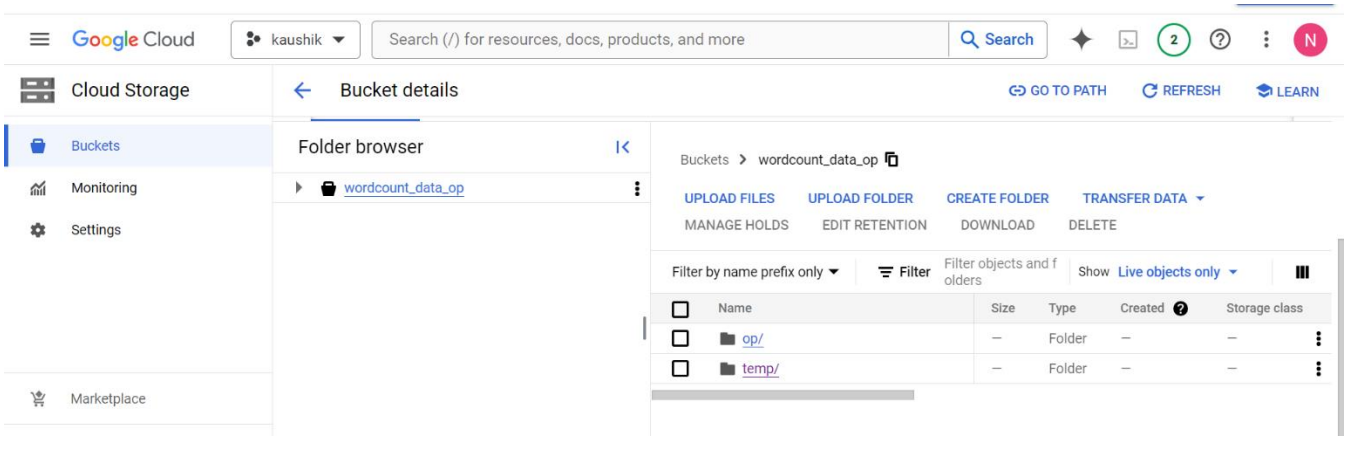
- The input file is located at `gs://wordcount_data/constitution.txt`, which is a Google Cloud Storage (GCS) path.
- The job will read the content of this file to perform word counting.

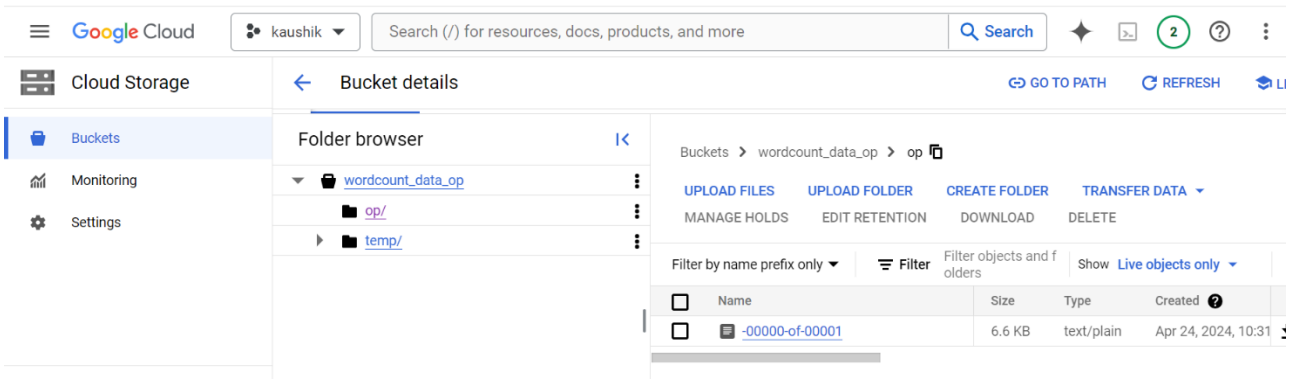
3. **`--output gs://wordcount_data_op`:**
   - Specifies the **output location** for the word count results.
   - The output will be written to a GCS bucket at `gs://wordcount_data_op`.
4. **`--runner DataflowRunner`:**
   - Specifies the **execution runner** for the job.
   - In this case, it's set to `DataflowRunner`, which means the job will run on Google Cloud Dataflow.
   - Dataflow is a managed service for executing Apache Beam pipelines.
5. **`--project kaushik-420409`:**
   - Specifies the **Google Cloud project ID** where the Dataflow job will run.
   - Replace `kaushik-420409` with your actual project ID.
6. **`--temp_location gs://wordcount_data_op/temp`:**
   - Specifies the **temporary location** for storing intermediate data during the job execution.
   - The temporary files generated by the job will be stored in the GCS bucket at `gs://wordcount_data_op/temp`.

```
(env) saikoushik2901@cloudshell:~ (kaushik-420409)$ python3 -m     apache_beam.examples.wordcount     --region us-central1
 --input     gs://wordcount_data/constitution.txt     --output     gs://wordcount_data_op     --runner DataflowRunner
--project kaushik-420409     --temp_location     gs://wordcount_data_op/temp/
INFO:apache_beam.internal.gcp.auth:Setting socket default timeout to 60 seconds.
INFO:apache_beam.internal.gcp.auth:socket default timeout is 60.0 seconds.
INFO:apache_beam.runners.dataflow.dataflow_runner:Pipeline has additional dependencies to be installed in SDK worker conta
iner, consider using the SDK container image pre-building workflow to avoid repetitive installations. Learn more on https:
//cloud.google.com/dataflow/docs/guides/using-custom-containers#prebuild
INFO:root:Using provided Python SDK container image: gcr.io/cloud-dataflow/v1beta3/beam_python3.10_sdk:2.55.1
INFO:root:Python SDK container image set to "gcr.io/cloud-dataflow/v1beta3/beam_python3.10_sdk:2.55.1" for Docker environm
ent
INFO:apache_beam.runners.portability.fn_api_runner.translations:==================== <function pack_combiners at 0x7a83c33
a3b50> ====================
INFO:apache_beam.runners.portability.fn_api_runner.translations:==================== <function sort_stages at 0x7a83c33ac3
a0> ====================
INFO:apache_beam.runners.dataflow.internal.apiclient:Starting GCS upload to gs://wordcount_data_op/temp/beamapp-saikoushik
2901-0424044656-754472-juju4d3x.1713934016.754827/pickled_main_session...
INFO:apache_beam.runners.dataflow.internal.apiclient:Completed GCS upload to gs://wordcount_data_op/temp/beamapp-saikoushi
k2901-0424044656-754472-juju4d3x.1713934016.754827/pickled_main_session in 1 seconds.
INFO:apache_beam.runners.dataflow.internal.apiclient:Starting GCS upload to gs://wordcount_data_op/temp/beamapp-saikoushik
2901-0424044656-754472-juju4d3x.1713934016.754827/pipeline.pb...
INFO:apache_beam.runners.dataflow.internal.apiclient:Completed GCS upload to gs://wordcount_data_op/temp/beamapp-saikoushi
```

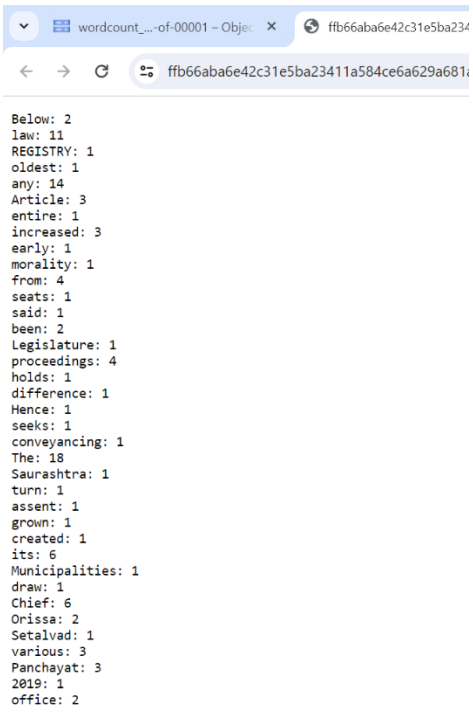


Now we can observe the result in output bucket.

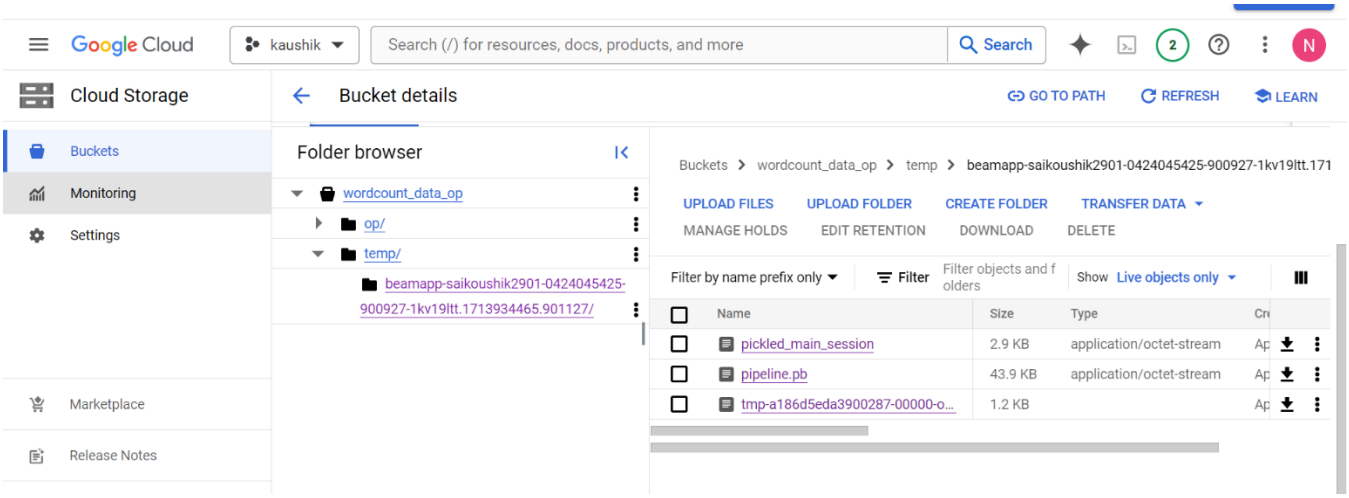


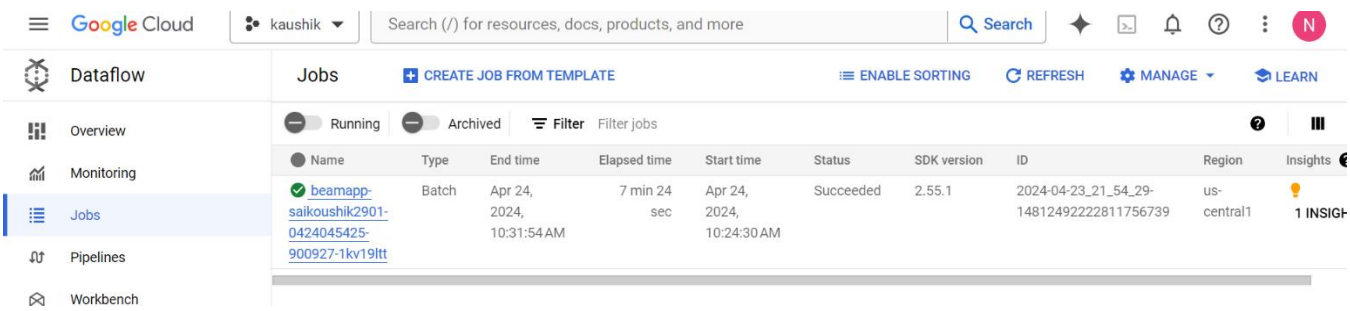


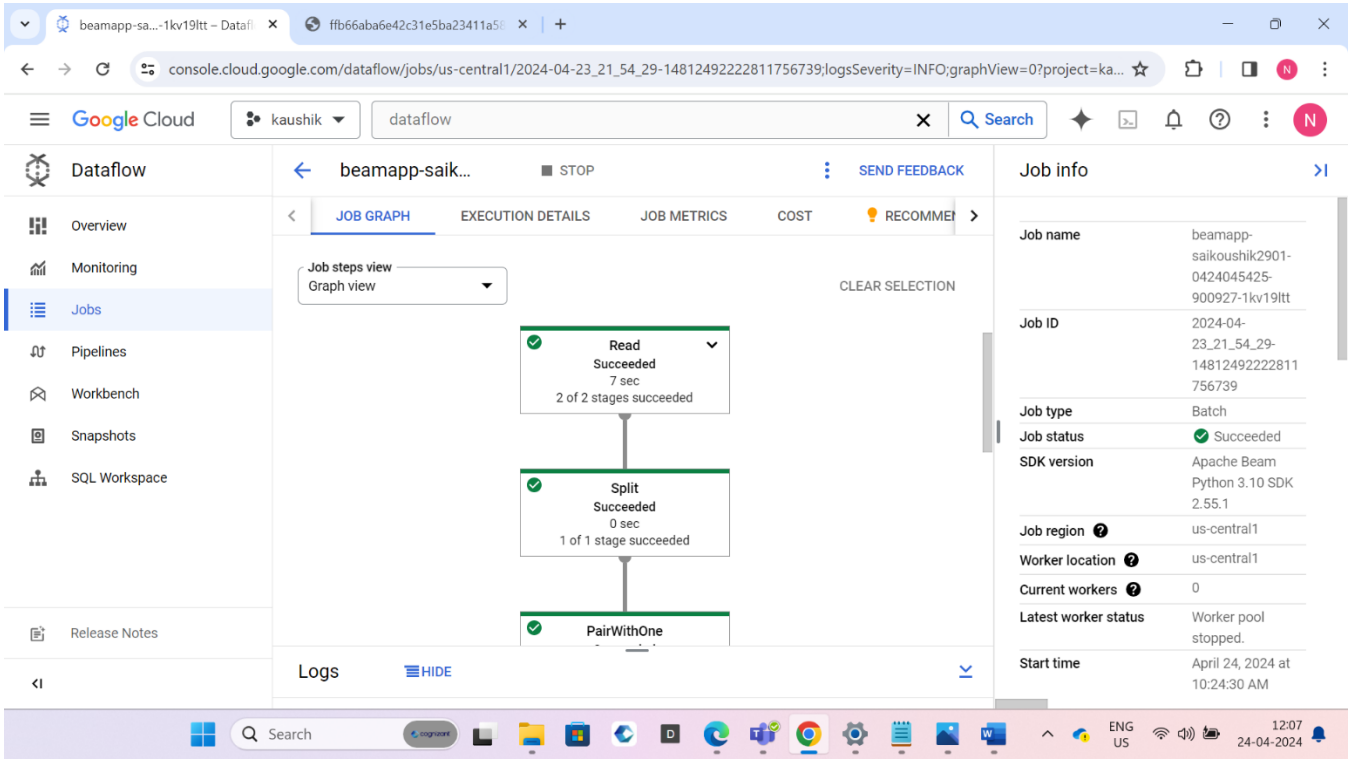
It’s the output bucket



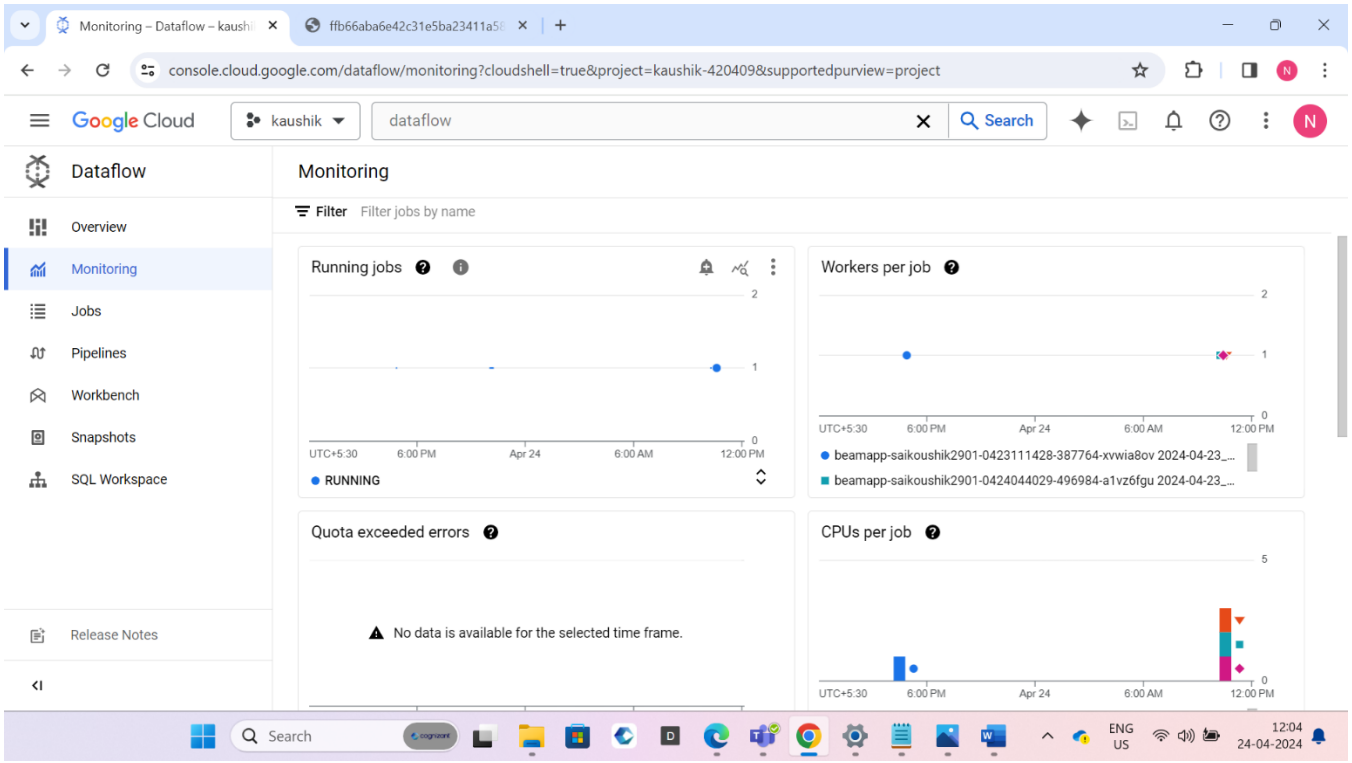
```
Below: 2
law: 11
REGISTRY: 1
oldest: 1
any: 14
Article: 3
entire: 1
increased: 3
early: 1
morality: 1
from: 4
seats: 1
said: 1
been: 2
Legislature: 1
proceedings: 4
holds: 1
difference: 1
Hence: 1
seeks: 1
conveyancing: 1
The: 18
Saurashtra: 1
turn: 1
assent: 1
grown: 1
created: 1
its: 6
Municipalities: 1
draw: 1
Chief: 6
Orissa: 2
Setalvad: 1
various: 3
Panchayat: 3
2019: 1
office: 2
```

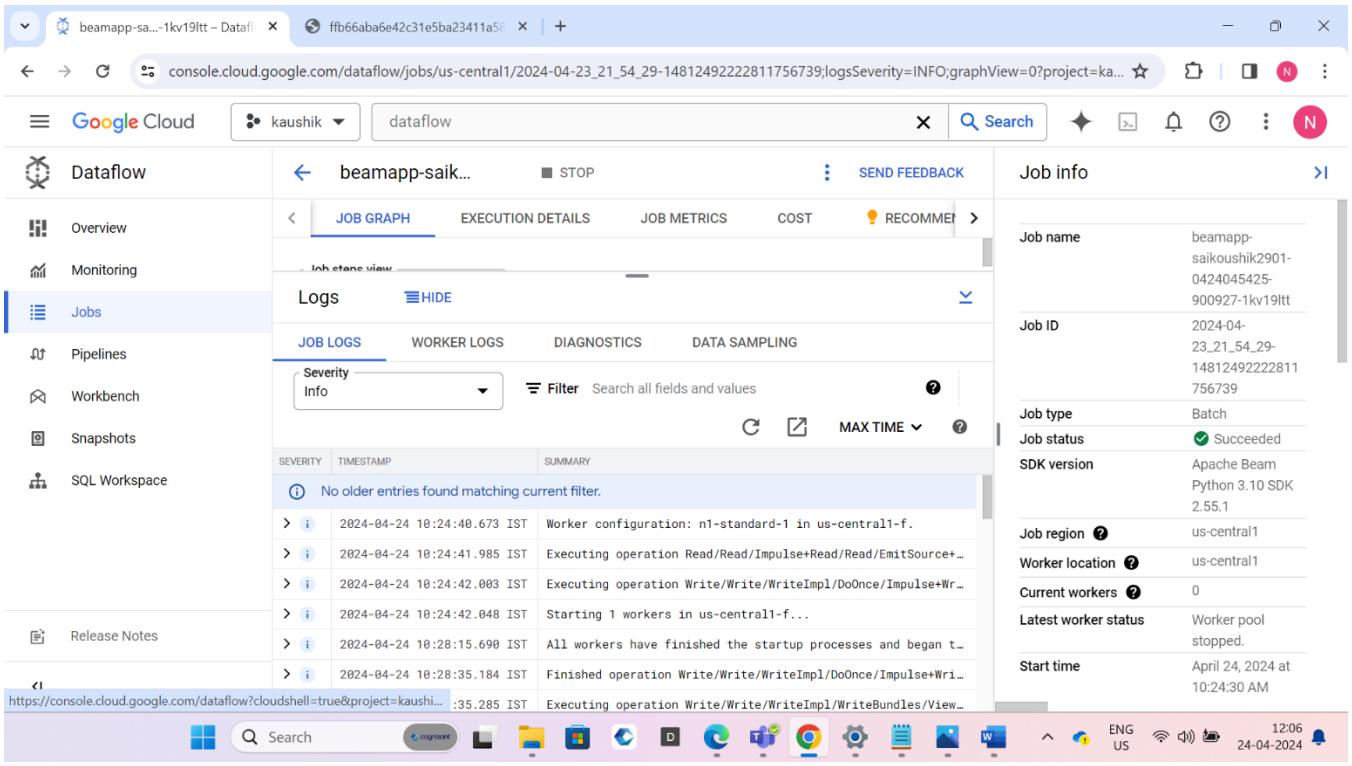






Here we can monitor the job in dataflow





Conclusion:- by using dataflow we successfully executed wordcount in google cloud platform.