CREATE A CLOUD FUNCTION TO PULL A MESSAGE FROM PUB/SUB

A Project Report Submitted in the fulfilment of the requirements for

Final-Project Evaluation

Abhishek Maradi M

2320738

GenC Intern

cognizant

CONTENTS

OBJECTIVES

The objective of this project is to build a real-time stock price monitoring system that alerts users when a stock price crosses a certain threshold.

The system will use Google Cloud Pub/Sub to stream real-time stock price updates and Google Cloud Functions to analyse and send alerts.

INTRODUCTION

In the ever-expanding landscape of cloud computing, efficient data storage management is paramount for organizations to balance performance, accessibility, and cost-effectiveness. Google Cloud Platform (GCP) offers a comprehensive suite of storage solutions tailored to meet diverse needs, including Standard and Archive buckets, each optimized for specific use cases.

Google Cloud Platform (GCP) offered by Google, is a suite of cloud computing services that Provides a series of modular cloud services including computing, data storage, data analytics, and machine learning, artificial intelligence alongside a set of management tools.

GCP run on the same infrastructure that Google uses internally for its own products, such as Google Search and YouTube.

Here are some key points about GCP:

1. **Infrastructure and Services**: GCP offers a wide range of infrastructure and application services that can be accessed on-demand. It enables users to build, deploy, and scale applications seamlessly while taking advantage of Google's powerful and reliable infrastructure.

2. **Global Resources**: GCP consists of physical assets (such as computers and hard disk drives) and virtual resources (such as virtual machines) distributed across data centres worldwide. These resources are organized into regions and zones, providing redundancy and reduced latency for better performance.
3. **Services and Integration**: When you develop applications on GCP, you combine various services to create the infrastructure you need. GCP services include compute, storage, databases, machine learning, and more. Additionally, GCP integrates seamlessly with other Google Cloud products.

Benefits of using GCP

There are many benefits to using GCP, including:

- Scalability: GCP can scale up or down to meet the demands of your application.
- Reliability: GCP is designed to be highly reliable, with a 99.9% uptime guarantee.
- Security: GCP is one of the most secure cloud platforms available, with a variety of security features to protect your data.
- Cost-effectiveness: GCP is a cost-effective cloud platform, with a pay-as-you-go pricing model.

Getting started with GCP is easy. You can create a free account and start using GCP services immediately.

To create a GCP account, visit the GCP website and click on the "Create an account" button. You will need to provide your name, email address, and a password.

Once you have created an account, you can start using GCP services. To learn more about GCP, you can visit the GCP documentation website.

Common use cases for GCP

GCP can be used for a wide variety of applications, including:

- Web and mobile applications: GCP can be used to host web and mobile applications.
- Data storage and analytics: GCP can be used to store and analyse data.
- Machine learning and artificial intelligence: GCP can be used to develop and deploy machine learning and artificial intelligence models.

REAL TIME USECASE:

Real-time Stock Price Monitoring

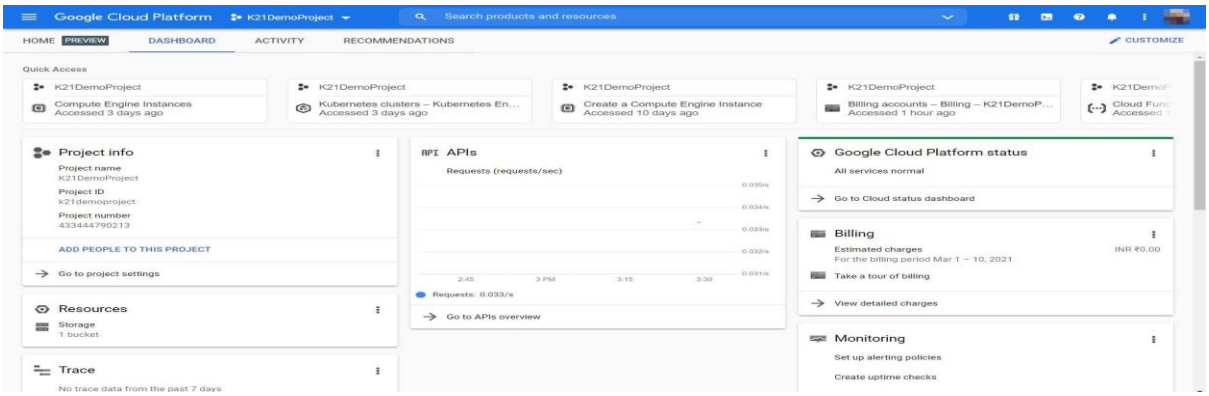
Build a real-time stock price monitoring system that alerts users when a stock price crosses a certain threshold. Use Cloud Pub/Sub to stream real-time stock price updates and Cloud Functions to analyse and send alerts.

Steps:

- Subscribe to a stock price API or feed and publish updates to Cloud Pub/Sub.
- Use Cloud Functions to analyse incoming price updates and check for threshold crossings.
- Send alerts via email, SMS, or push notifications using Cloud Functions.

SOFTWARE REQUIREMENTS :

GCP CONSOLE:



The first one is the Web Console. The moment you sign up with GCP, what you actually have is the web console and it is the front end, it is the gateway to dealing with a variety of services. As a beginner, you might want to spend more time with Web Console just getting yourself familiar with the navigation and understanding all the services offered by GCP.

Google Cloud Platform (GCP) Account: Sign up for a GCP account if you don't have one already. This provides access to GCP's storage services, including Cloud Storage for creating buckets and managing data.

GOOGLE COLAB :

Google Colab (short for Collaboratory) is a cloud-based Jupyter notebook environment that allows you to write and execute Python code. It's particularly useful for data science, machine learning, and collaborative projects.

GOOGLE LOOKER:

Google Looker is a business intelligence and data analytics platform developed by Looker Data Sciences, Inc., which was acquired by Google Cloud in 2020.

Looker provides a comprehensive suite of tools for data exploration, visualization, and reporting, enabling organizations to derive insights from their data.

It offers a semantic modelling layer that allows users to define metrics and dimensions in a business-friendly language, facilitating collaboration between technical and non-technical users.

Google Looker is often used by businesses to analyse data from various sources, create interactive dashboards, and make data-driven decisions.

With the acquisition by Google Cloud, Looker is integrated with Google's cloud infrastructure and services, providing additional capabilities for data processing and analysis.

PROJECT REQUIREMENTS

Pub/Sub :

- Google Cloud Pub/Sub is a managed service to ingest data at scale.
- It is built using the publisher subscriber model where you have a set of publishers that send messages to a topic, and there are a set of subscribers that subscribe to the topic, and Pub/Sub provides the infrastructure for the publishers and subscribers to reliably exchange messages.
- Pub/Sub acts as a global entry point to GCP based analytics services. Whether you are ingesting elementary data logs or any of the data that is ingested into the Cloud. It is typically sent by a Cloud Pub/Sub.
- It acts as a reliable and simple staging location for data though Pub/Sub is not meant to be a durable data store. It can be used for staging data as it enters the Cloud and is waiting to get processed by either cloud dataflow or data proc.
- In fact, Pub/Sub can deliver the data to a variety of destinations depending on how the subscribers are configured. This is tightly integrated with services such as Cloud storage and Cloud Dataflow, where you can use Pub/Sub to store inbound data through - for real time processing through Dataflow or for historical dataset archival on Cloud storage.
- Cloud Pub/Sub supports at least once delivery with synchronous cross zone message replication. What this means is you actually get a highly reliable delivery mechanism based on Pub/Sub, and there is redundancy because of cross zone message replication.

- You don't lose messages when it is sent via Cloud Pub/Sub infrastructure. Unlike most of the services of Google Cloud platform comes with end to end encryption, integration with identity and access management, and audit logging.
- So all these capabilities give you additional mechanism to secure and also monitor the inbound data coming where Cloud Pub/Sub.

Cloud Functions:

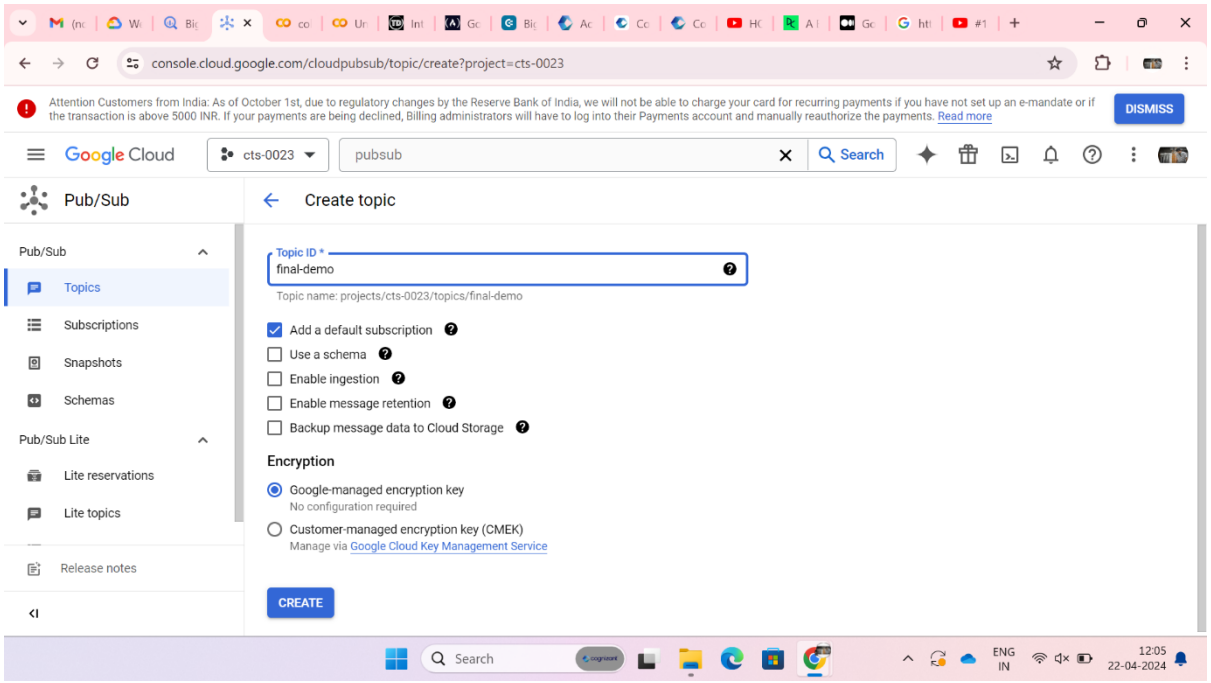
- It's a serverless execution environment for building and connecting cloud services. So serverless computer environment is a mechanism where you don't have to provision and configure resources beforehand.
- This is fundamentally different from the way you deal with app engine, compute engine or even Kubernetes engine. For each of those you got to provision resources beforehand.
- We need to create app engine instances you need to launch VMs. We need to create a cluster, even before you can run your first line of code. But that's very different when it comes to cloud functions.
- In cloud functions, you write code as a function, which has a well-defined entry point and exit point, and you deploy that with no changes.
- That's the reason why it is called as a serverless compute environment where you don't have to provision a virtual machine or a container, to run your code. It typically, serverless compute environments respond to events.
- So instead of running this code forever, they get executed only when there is an external event. For example, adding a new object to a storage bucket or dropping a new message to the pub Sub queue, or for that matter, invoking a hit HTTP endpoint which will result in executing the serverless code.

- So any of those can be considered as the external event responsible for triggering the code. You can write cloud functions in JavaScript, Python 3, and Go. You don't have to package them in a specific format.
- At the most it's a zip file that gets into GCP environment and starts executing against events. GCP events fire a cloud function through a trigger. So trigger is what connects the external resource to a cloud function.

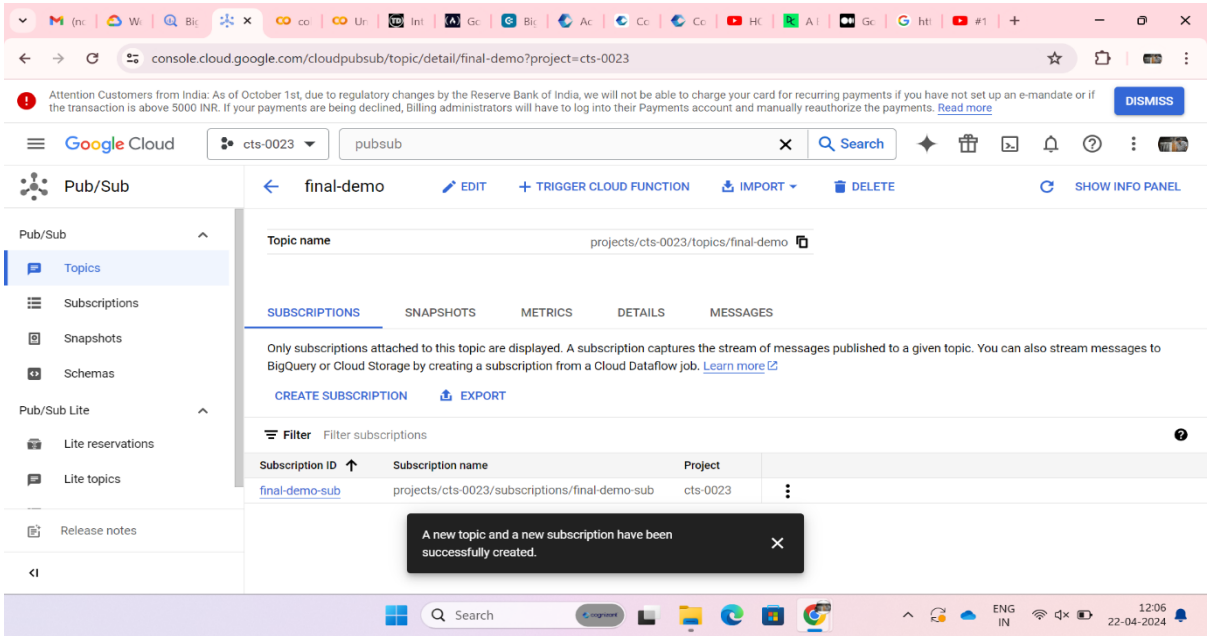
An example event could be adding a new object to a storage bucket, a classic use case in this scenario is converting high resolution images uploaded to Google Cloud storage to thumbnails. So every time a new high resolution image is added to a storage bucket, it triggers a cloud function and using an image manipulation library, it gets resized into a thumbnail and put in a different storage bucket. It this happens every time a new high risk image is added to the storage bucket. This is one of the most efficient and economical way of running code in the cloud. Triggers connect events to the function. So there is a trigger and there is an event and a function. So you define an event and you connect it to the function, and every time the trigger takes place it invokes the function via the event. So this is the environment that is very useful for executing code, which is written as code snippets or functions, and that's one of the reasons it's called as functions as a service or FAAS.

IMPLEMENTATION

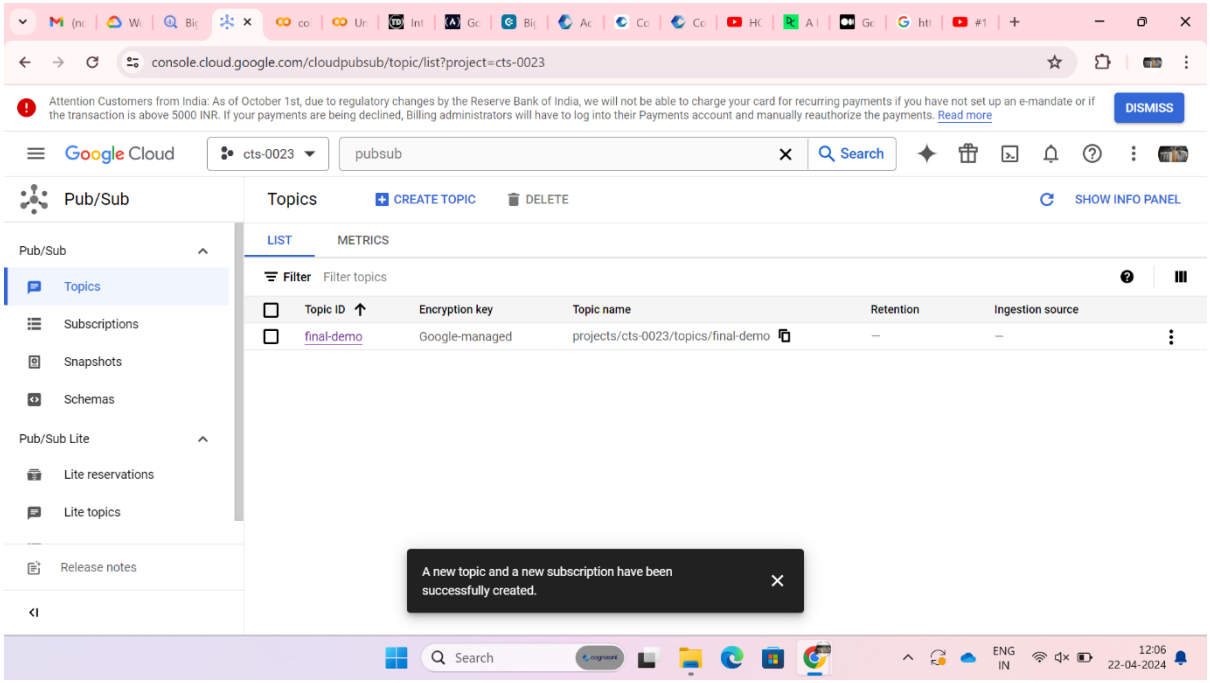
Firstly, we created a topic named as “final-demo” with default subscription.



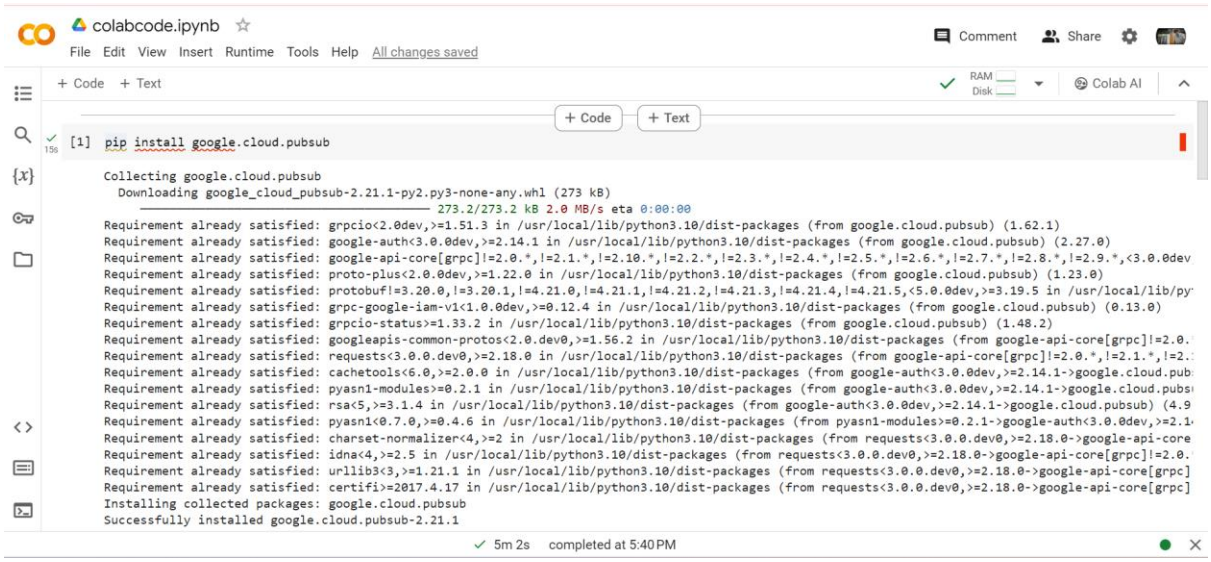
Here, default subscription is named as “final-demo-sub”.



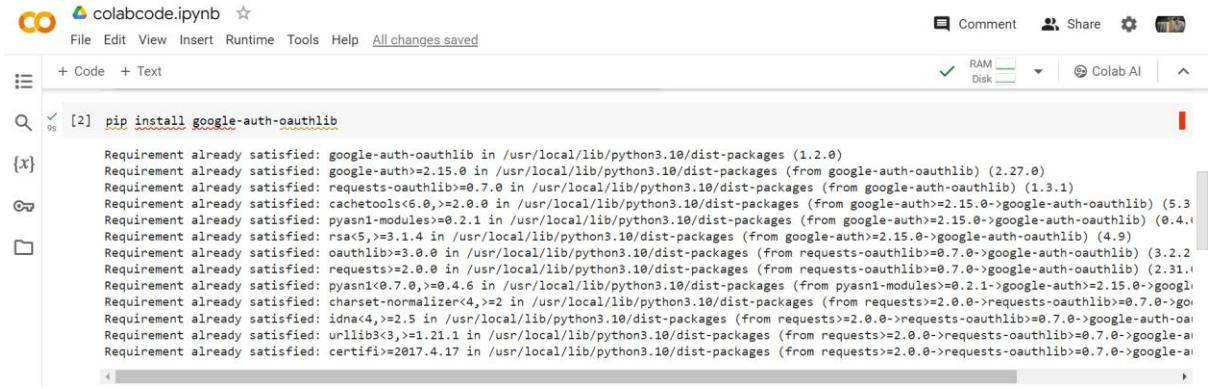
The topic named “final-demo” is created successfully.



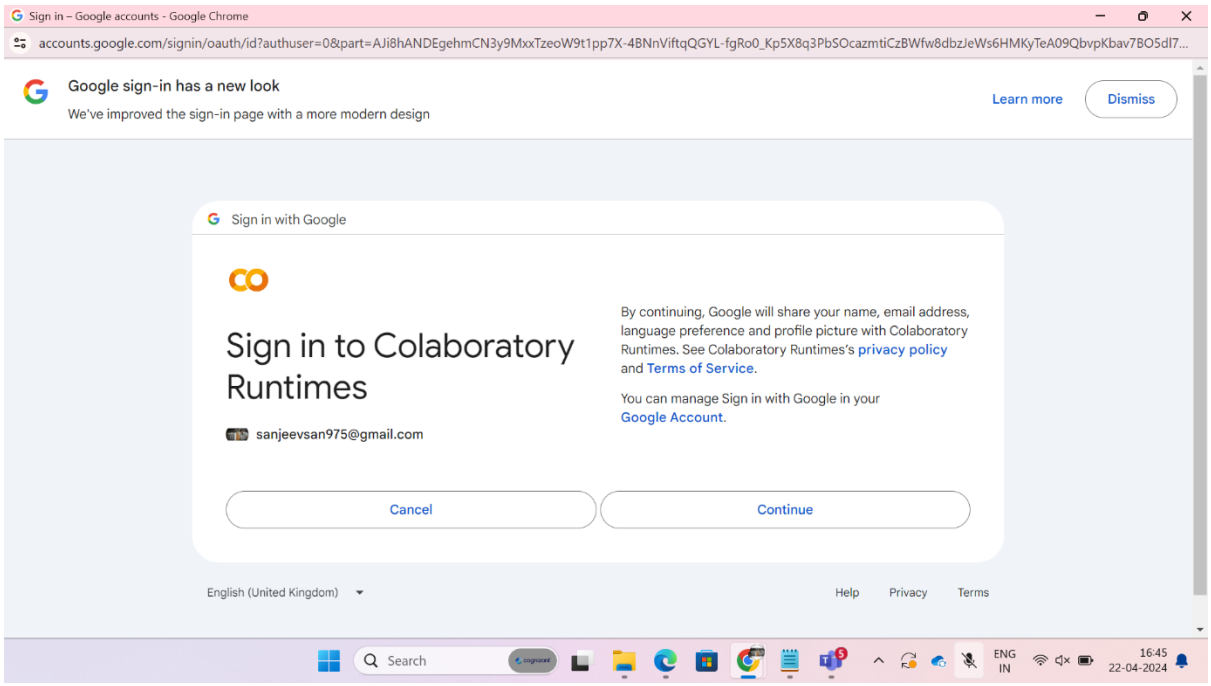
This line installs the `google.cloud.pubsub` package, which is required for interacting with Google Cloud Pub/Sub.



This line installs the `google-auth-oauthlib` package, which is required for handling authentication with Google services.



Here, Sign in page of colab will pop up as a part of Authentication from the OAuth package.



Here, it initiates the authentication process for the user, which is necessary for accessing Google Cloud services securely.

Next line imports the `requests` module, which is used for making HTTP requests to fetch web pages or interact with web APIs.

Then next line imports the `BeautifulSoup` class from the `bs4` (Beautiful Soup 4) package, which is used for parsing HTML and XML documents.

Next line imports the `time` module, which is used for adding time delays or getting current time information.

Then next line imports the `pubsub_v1` module from the `google.cloud` package, which provides functionality for interacting with Google Cloud Pub/Sub.

Then next line defines a function named `scrape_price` that takes two parameters: `url` and `class_name`. This function is used to scrape price data from a webpage.

Next line extracts the price data from the HTML content, converts it to a floating-point number, and removes any extraneous characters or formatting.

```
[3] from google.colab import auth
    from google.auth.transport.requests import Request
    from google.oauth2 import id_token

[5] auth.authenticate_user()

import requests
from bs4 import BeautifulSoup
import time
import pandas as pd
from google.cloud import pubsub_v1

# Function to scrape price data
def scrape_price(url, class_name):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Check if the element with the specified class name exists
    if soup.find(class_=class_name) is not None:
        price = float(soup.find(class_=class_name).text.strip()[1:].replace(",",""))
        return price
```

Then it assigns the value `"INFY"` to the variable `ticker`, representing the stock ticker symbol.

Then it constructs the URL for fetching the stock price data from Google Finance, using the ticker symbol `"INFY"`.

Then it assigns the class name of the HTML element containing the stock price data to the variable `class_name`.

Then next it creates a publisher client object for interacting with Cloud Pub/Sub.

Then it calls the `scrape_price` function to fetch the stock price data from the specified URL using the given class name.

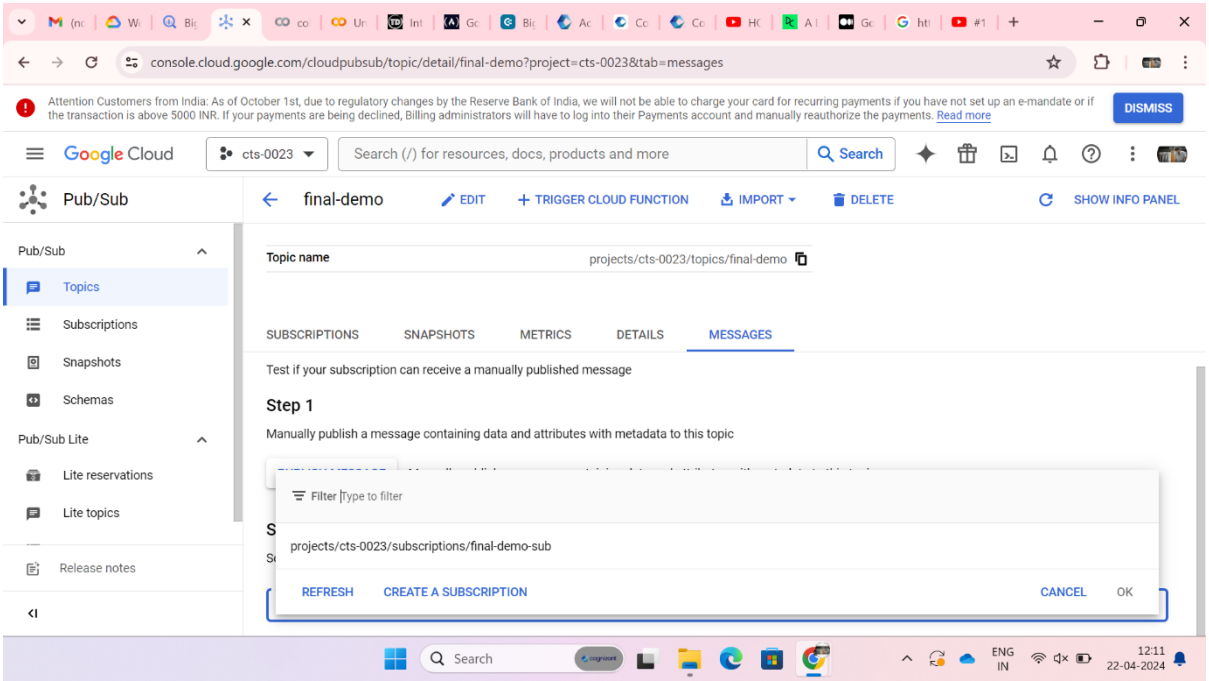
The line encodes the message data as bytes before publishing it to Cloud Pub/Sub.

Then next it publishes the message data to the Cloud Pub/Sub topic.

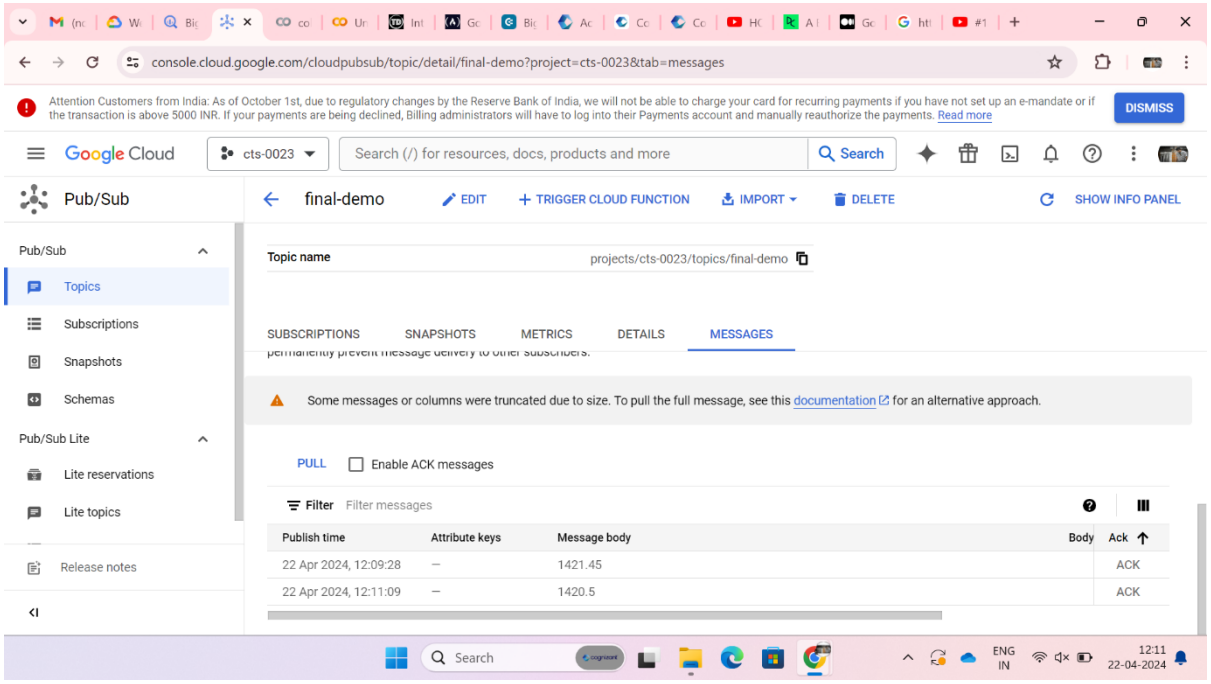
Then next it waits for the message to be published before proceeding.

Then it pauses the execution of the program for 100 seconds before the next iteration of the loop. This is used to limit the frequency of price updates to once every 100 seconds.

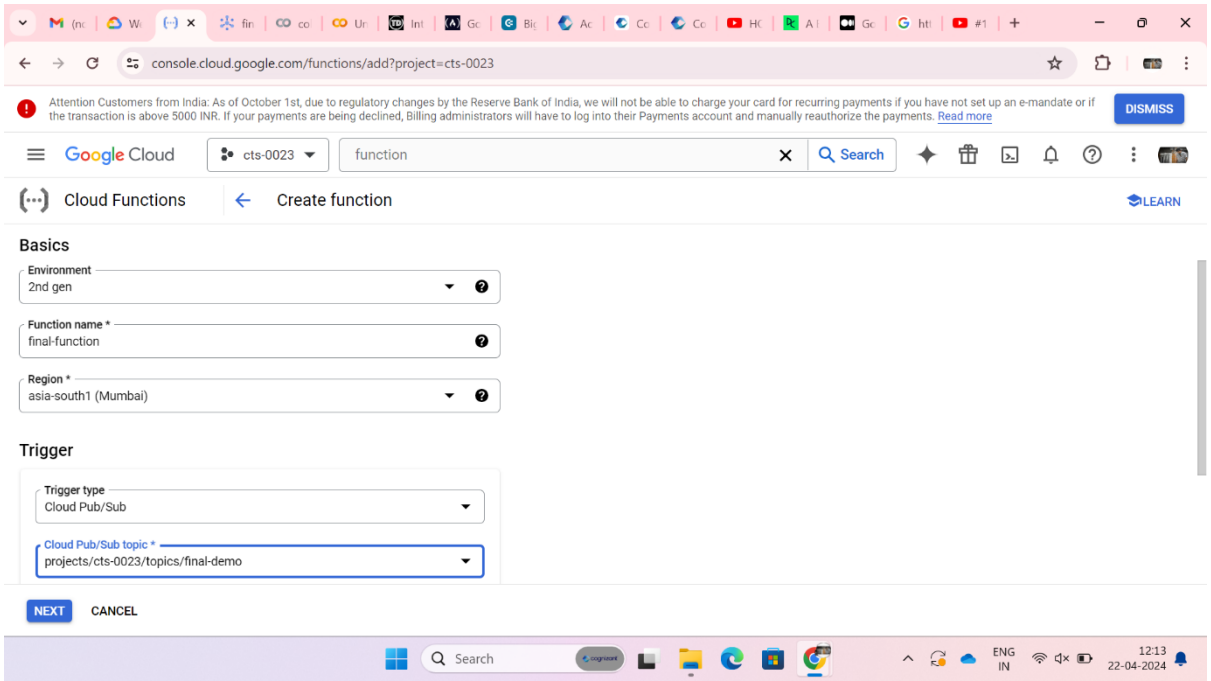
After fetching the stock price, we are selecting the subscription ID to where the message has to be sent.



The INFY stock price is published in topic messages.



Then, we built a cloud function named “final-function” with trigger type “cloud pub/sub” with the respective topic.



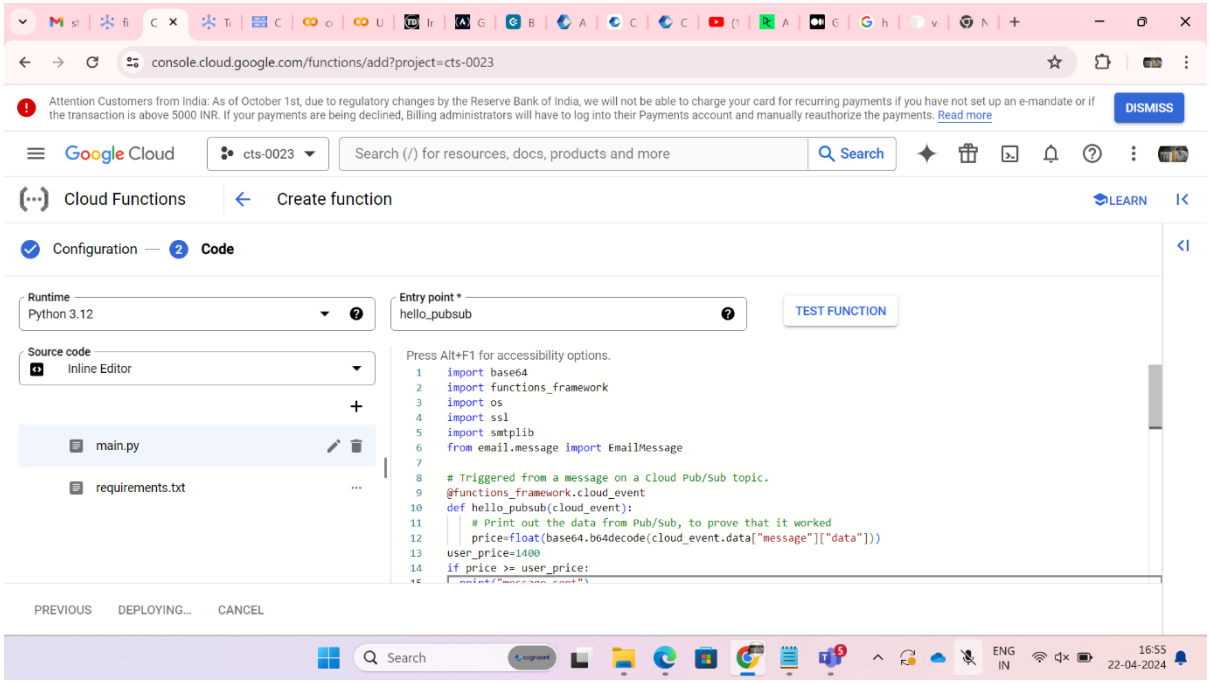
Here, we defined a cloud function .

Initially, Base64 is a binary-to-text encoding scheme used to encode data.

Functions_framework is a framework for writing lightweight functions as part of Google Cloud Functions

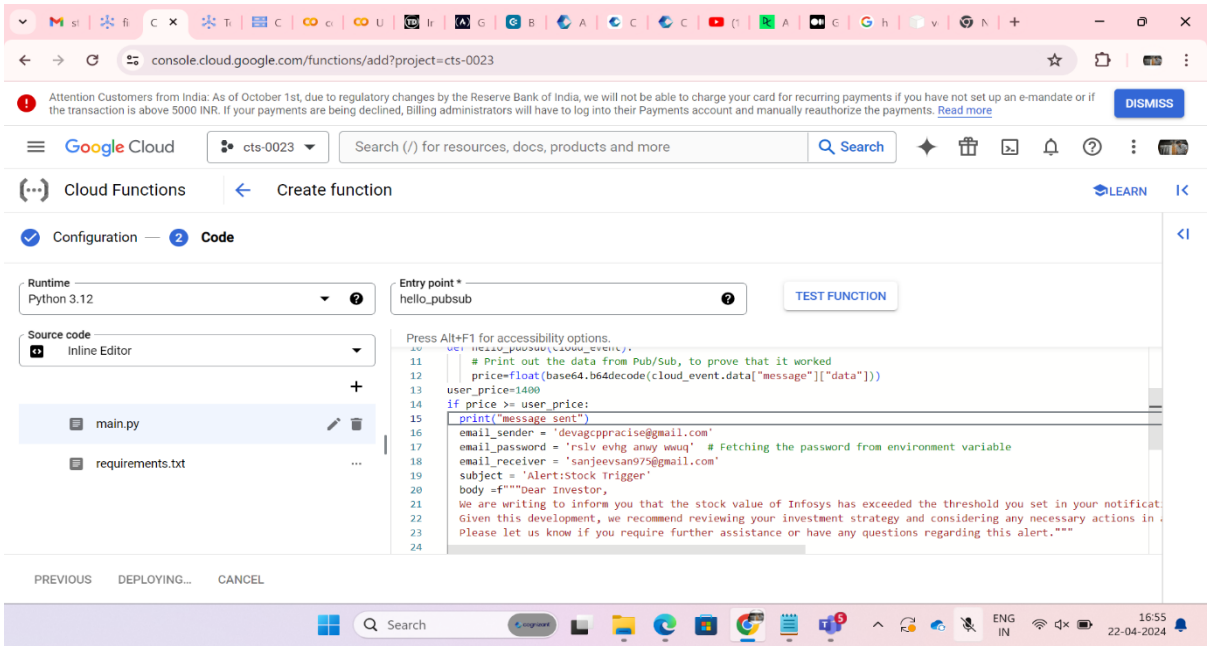
Then we imported the EmailMessage class from the email.message module, which is used to represent an email message.

Then it defines a function named `hello_pubsub` which takes a `cloud_event` as its parameter. This function will be triggered when a message is published to a Cloud Pub/Sub topic.

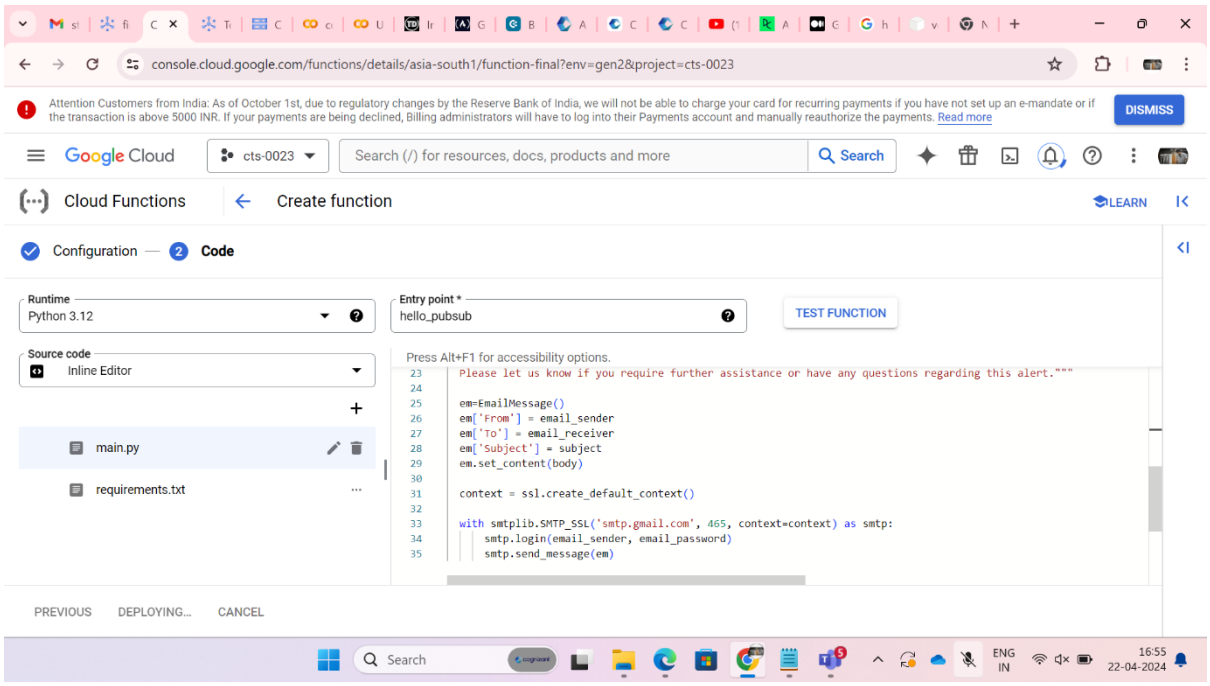


The line decodes the data from the Cloud Pub/Sub message, which is encoded in base64 format. It then converts it to a float, assuming the data represents a numeric value like a stock price.

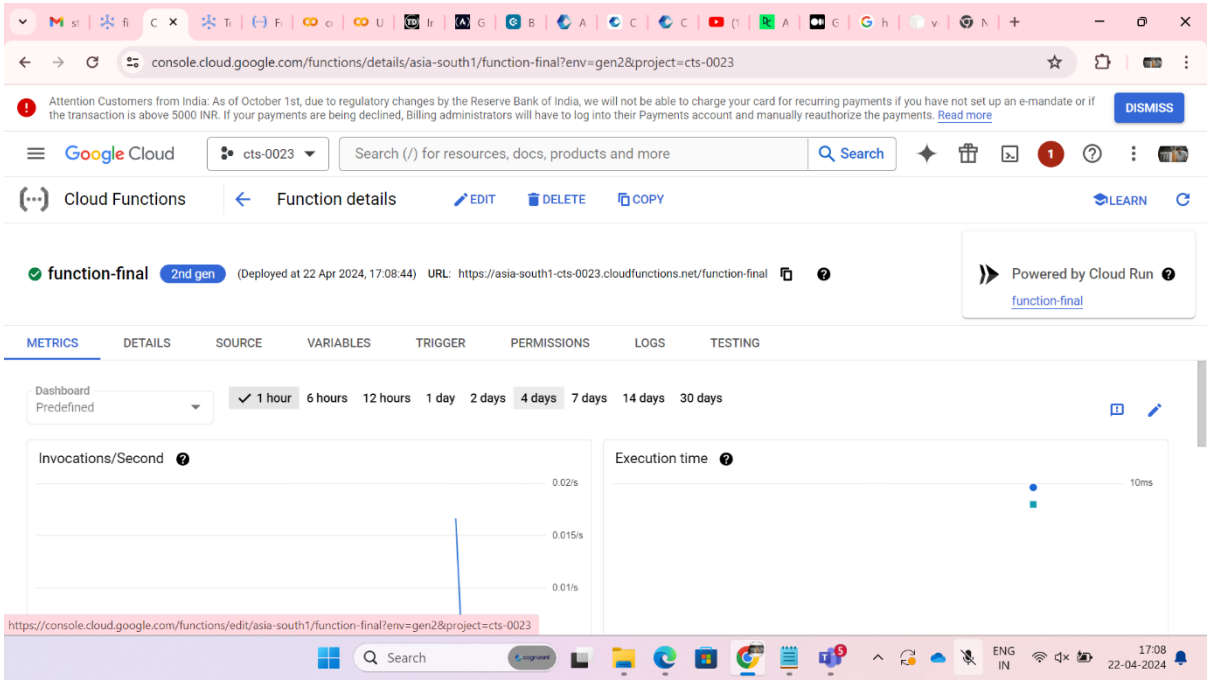
Next it checks if the price extracted from the Pub/Sub message is greater than or equal to a threshold price (`user_price`). If the condition is true, it proceeds to send an alert.



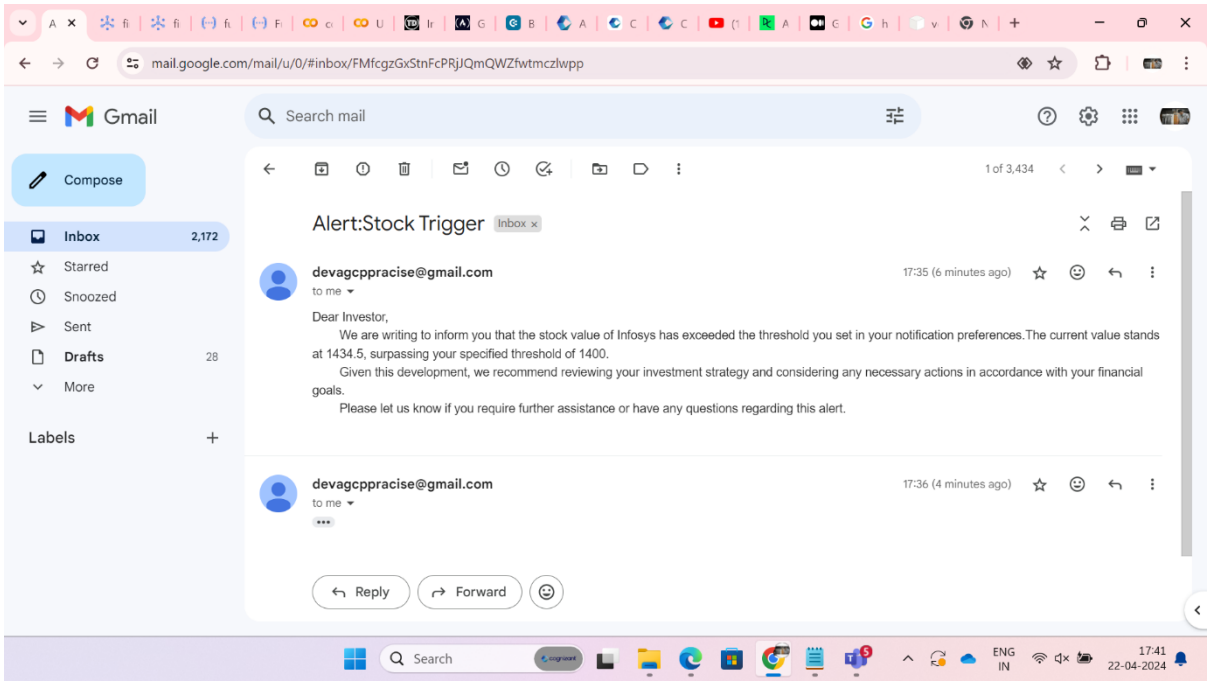
An instance of EMAILMESSAGE class is created and that instance is used to create from, to, subject and body fields of email.



The cloud function got deployed.

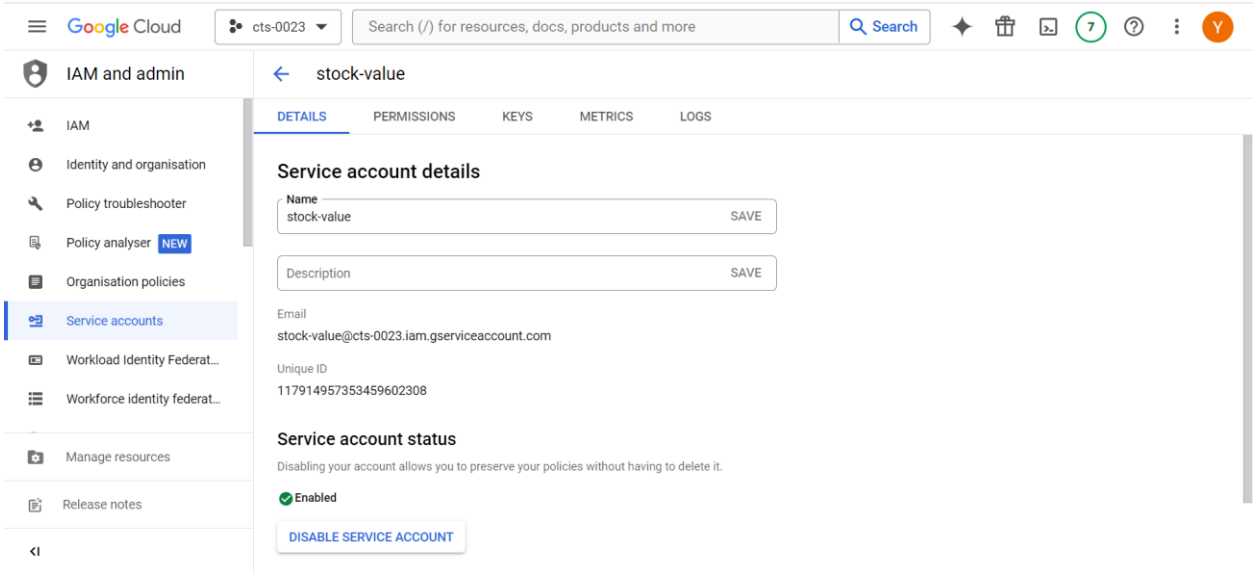


When the stock price exceeded the threshold value then alert mail got triggered with a message.

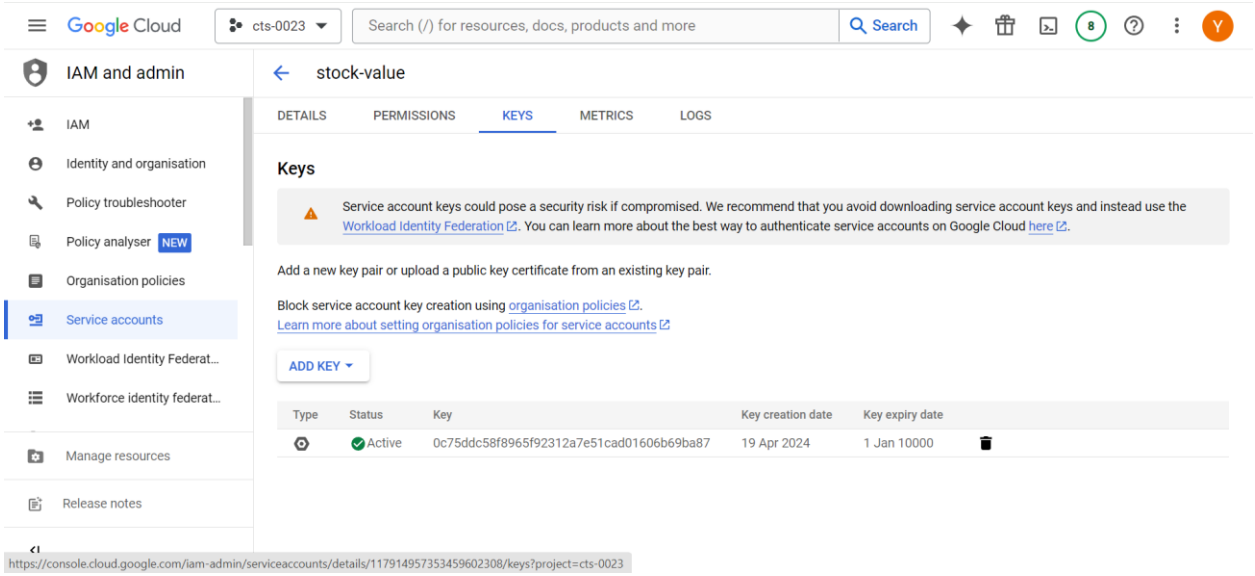


VISUALISATION:

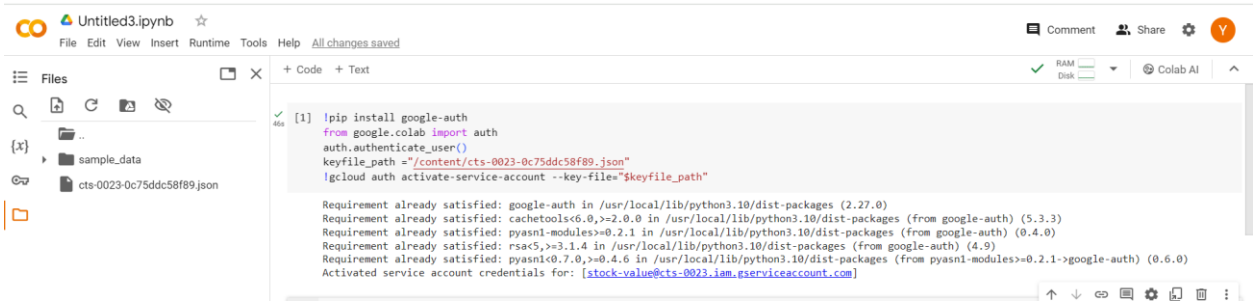
Here, we are creating a service account for authentication to access the cloud storage bucket.



We created a json key file to service account.

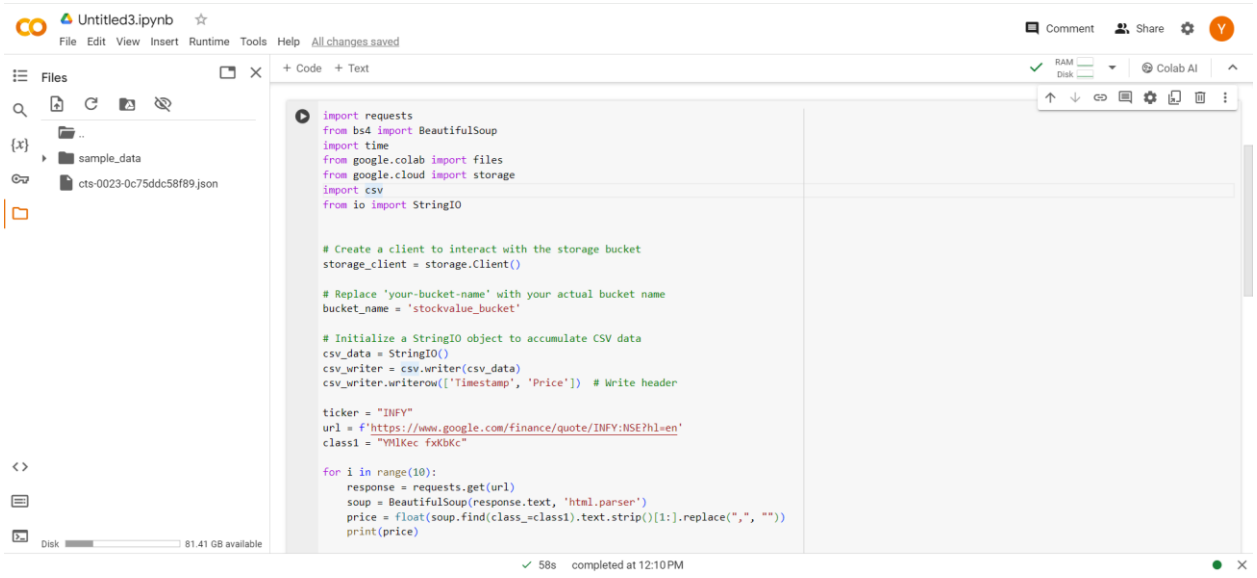


Then, we stored the json keyfile path in a variable for authentication.

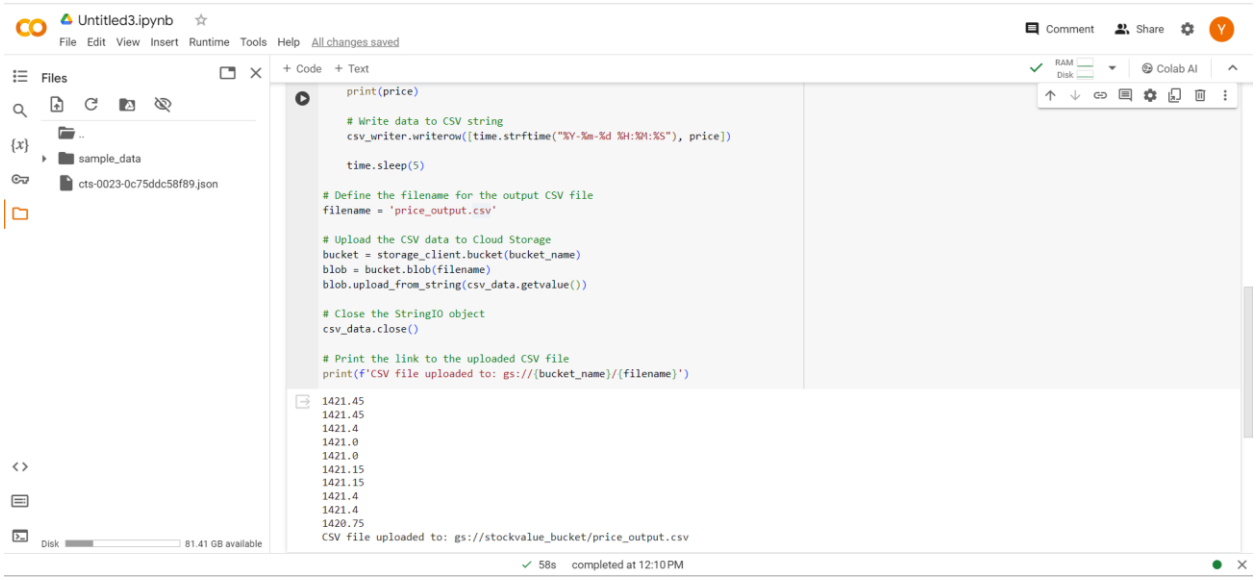


We imported csv module to store the stock price and its timestamp in a csv file.

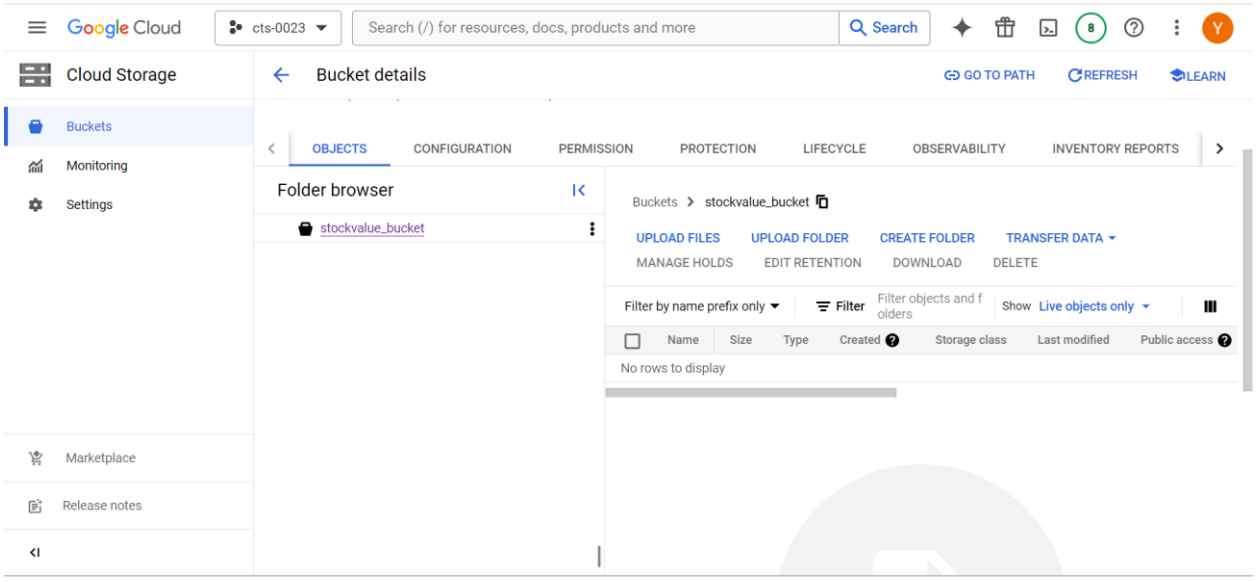
Then, we created a storage client to create a bucket.



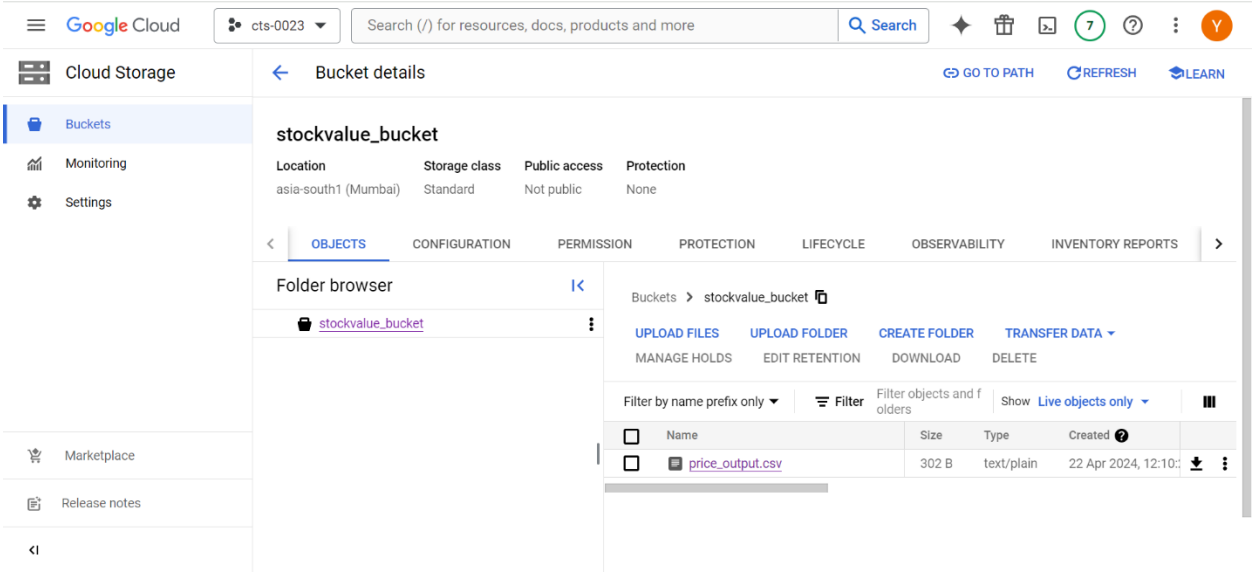
We created a csv file with first 10 stock price values of INFY and stored in price_output.csv



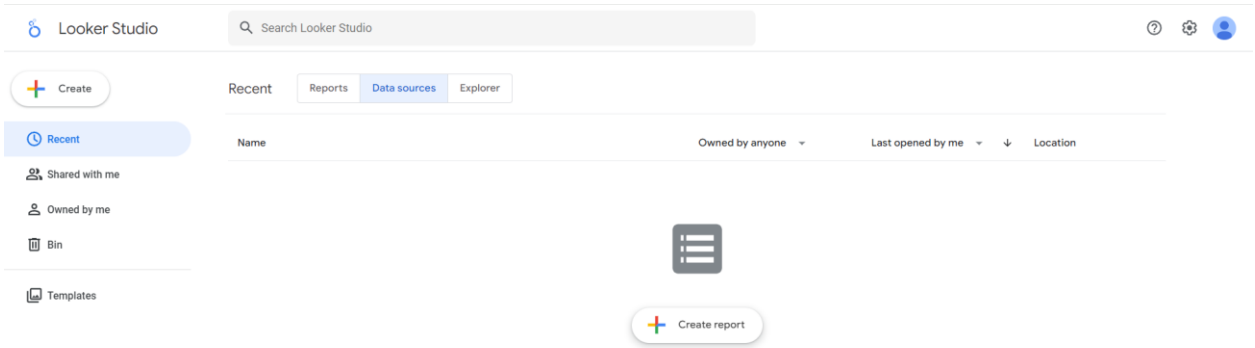
The bucket created before adding object.



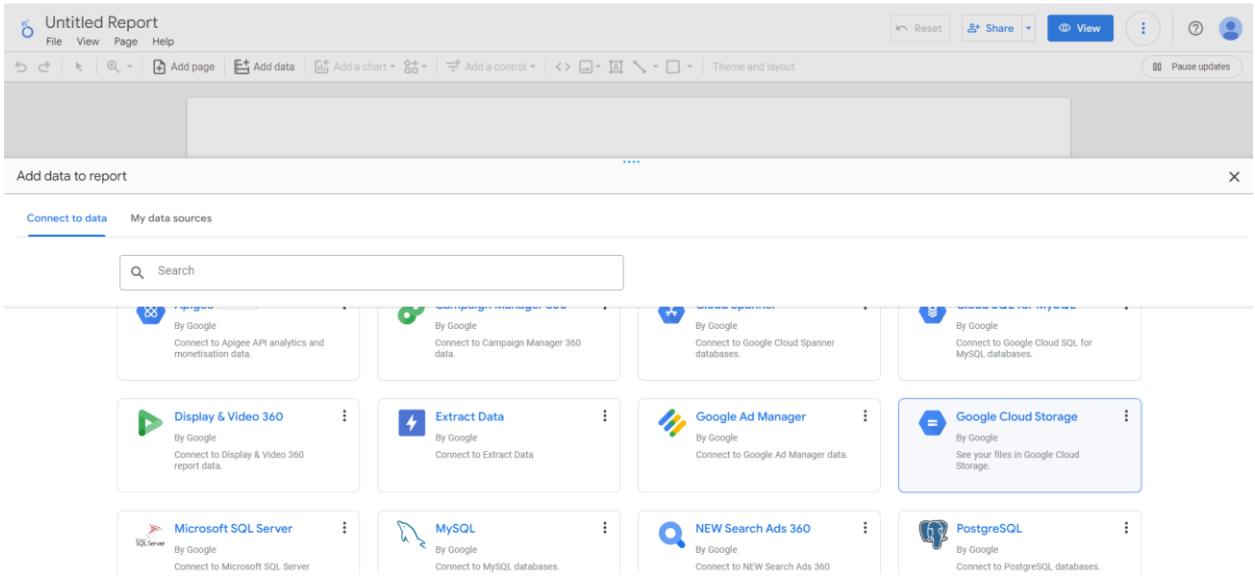
The object “price_output.csv” is reflected in the bucket stockvalue_bucket



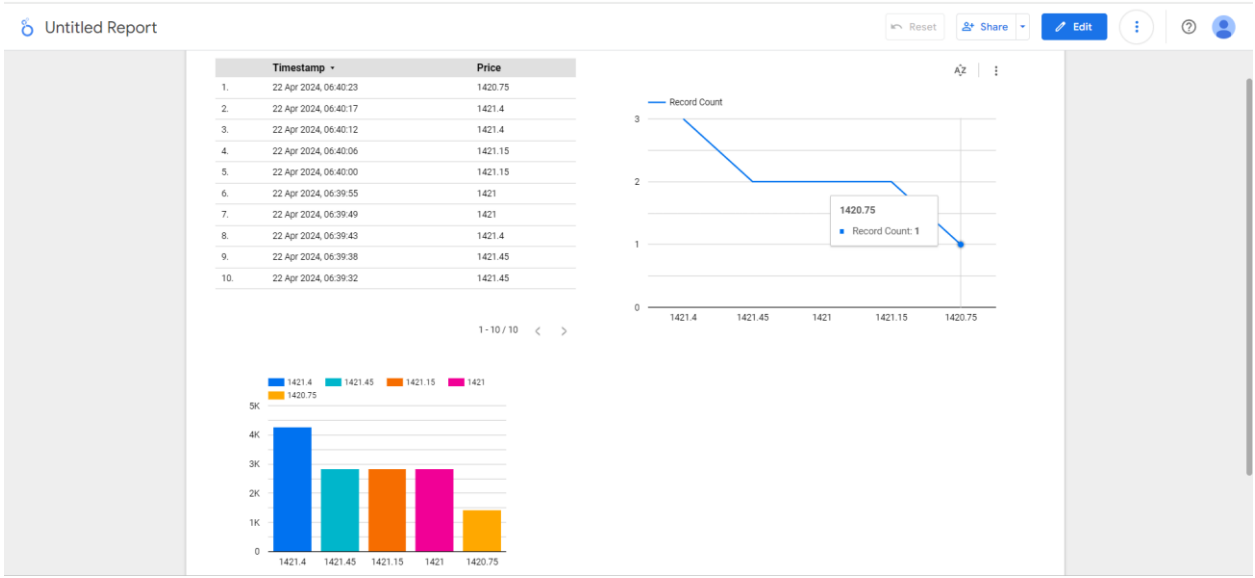
We visualized data using Google Looker Studio by creating report.

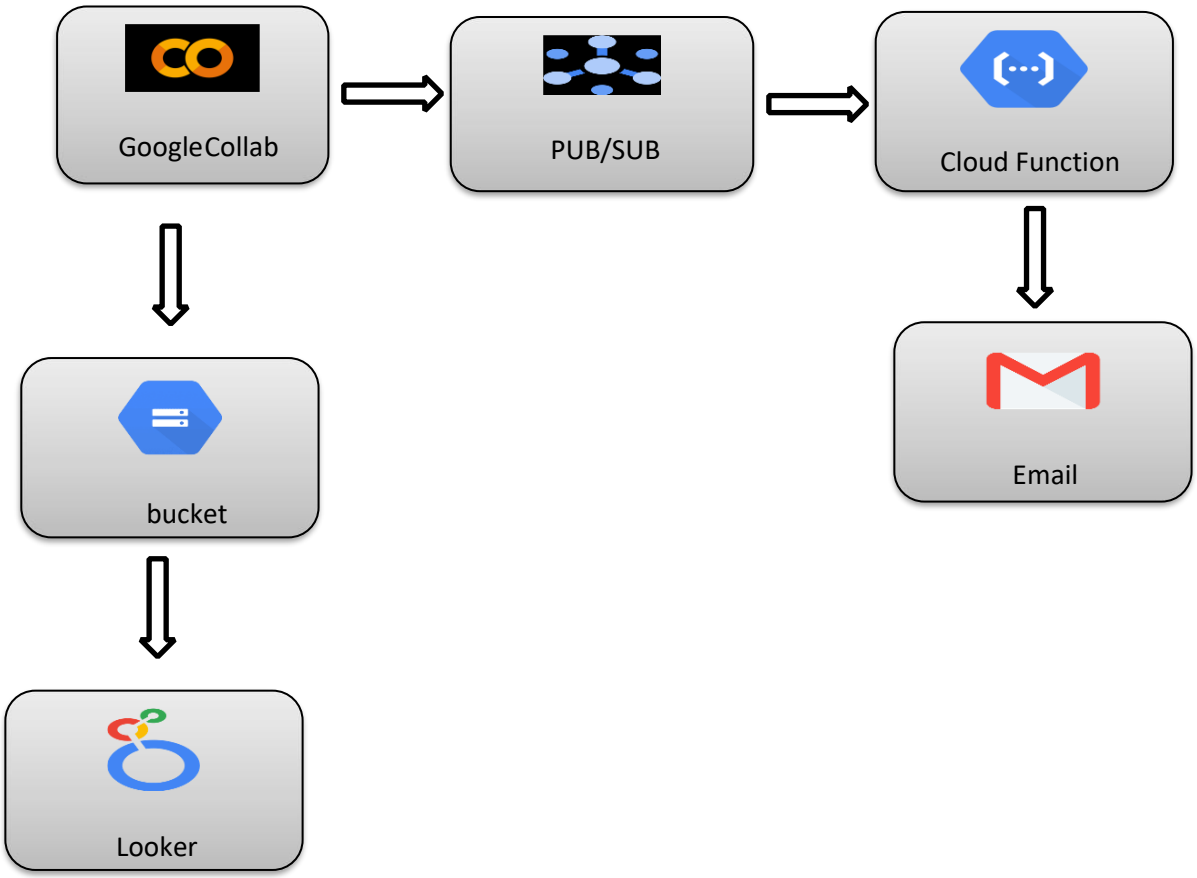


We extracted object from the google cloud storage bucket.



Finally the report is created where y-axis represents Timestamp and x-axis represents Stock price.





CONCLUSION

- The real-time stock price monitoring system successfully scrapes stock prices, publishes them to Pub/Sub, and triggers alerts. With the suggested enhancements, you can make it even more powerful and user-friendly.
- We built a real-time stock price monitoring system that uses Cloud Pub/Sub to stream stock price updates and Cloud Functions to analyze and send alerts.
- The system scrapes stock price data from a website, publishes it to a Pub/Sub topic, and triggers a Cloud Function.
- Scraping component scrapes stock price data from a specified URL using BeautifulSoup. We extract the stock price from the HTML page.
- Pub/Sub publishes the scraped stock price data to a Pub/Sub topic.

- Cloud Function component subscribes to the Pub/Sub topic. It analyzes the stock price data and sends alerts (email and SMS) if the stock price crosses a certain threshold.

FUTURE ENHANCEMENTS

Customizable Alerts:

- Allow users to set personalized alert thresholds for each stock.
- Implement different notification channels (e.g., email, SMS, push notifications) based on user preferences.

Machine Learning Predictions:

- Train ML models to predict stock price movements.
- Use historical data to provide insights and recommendations.

Interactive Dashboard:

- Create a web-based dashboard to visualize real-time stock prices.
- Include charts, graphs, and historical data.

Mobile App Integration:

- Develop a mobile app for users to receive alerts on their smartphones.
- Provide a seamless user experience.

Market News Integration:

- Fetch relevant news articles related to monitored stocks.
- Display news alongside stock prices.