Flight Data Scraping and Cloud Storage

Table of Contents

1. 1. Introduction
2. 2. Prerequisites
3. 3. Script Explanation
4. 3.1 Imports and Setup
5. 3.2 Airport Code Mapping
6. 3.3 Input Data
7. 3.4 Main Cloud Function
8. 3.5 URL Generation
9. 3.6 Show More Button Click
10. 3.7 Data Scraping
11. 3.8 Uploading to Google Cloud Storage
12. 4. Instructions for Using the Script
13. 4.1 Setting Up Google Cloud Environment
14. 4.2 Creating Google Cloud Storage Bucket
15. 4.3 Preparing and Deploying the Script
16. 4.4 Triggering and Monitoring the Cloud Function
17. 4.5 Checking Uploaded Data
18. 4.6 Troubleshooting
19. 5. Conclusion
20. 6. Screenshots

# 1. Introduction

This project is aimed at scraping flight data from the Kayak website and uploading the scraped data to Google Cloud Storage using a Google Cloud Function. The script automates the process of gathering flight information such as airlines, departure times, arrival times, and prices, making it easier to perform data analysis and tracking over time.

# 2. Prerequisites

Before running the script, the following prerequisites must be met:

* A Google Cloud account with permissions to create and manage projects, storage buckets, and deploy Cloud Functions.
* Google Cloud SDK installed on your local machine.
* WebDriver for Selenium installed and the path to the WebDriver is correctly specified in the script.
* Python installed with required packages (pandas, selenium, beautifulsoup4, google-cloud-storage, functions-framework).

# Script Explanation

## Imports and Setup

The script begins by importing necessary Python libraries. These include libraries for web scraping (Selenium, BeautifulSoup), data manipulation (Pandas), and interaction with Google Cloud Storage. The script also sets up necessary configurations such as the WebDriver path and options for running Chrome in headless mode.

## Airport Code Mapping

A dictionary is used to map full airport names to their respective IATA codes. This mapping is crucial for generating the correct URLs for flight searches on the Kayak website.

## Input Data

The input data is a list of dictionaries, where each dictionary contains a source and destination airport. These pairs define the routes for which the flight data will be scraped.

## Main Cloud Function

The main function, `scrape\_flights`, is designed to be triggered by a cloud event. It orchestrates the entire process of URL generation, data scraping, and uploading the results to Google Cloud Storage.

## URL Generation

URLs for flight searches are generated based on the source, destination, date, and travel class (economy, premium economy, or business). The function `generate\_flight\_url` constructs these URLs dynamically for each route and date combination.

## Show More Button Click

The `click\_show\_more\_button` function handles the clicking of the 'Show More' button on the Kayak page to load additional flight results. This is done using Selenium's WebDriver, ensuring more data is available for scraping.

## Data Scraping

The `scrape` function extracts flight details such as airline, departure time, arrival time, number of stops, stopover details, and price from the webpage using BeautifulSoup. The extracted data is then stored in a Pandas DataFrame.

## Uploading to Google Cloud Storage

Finally, the `upload\_to\_gcs` function uploads the DataFrame as a CSV file to a specified Google Cloud Storage bucket. This allows the scraped data to be easily accessed and analyzed later.

# Instructions for Using the Script

## 1 Setting Up Google Cloud Environment

1. Authenticate your Google Cloud account using the following command:

gcloud auth login

2. Ensure you have the necessary permissions to create and manage projects, storage buckets, and deploy Cloud Functions.

## 2 Creating Google Cloud Storage Bucket

1. Navigate to the Google Cloud Console and create a new storage bucket under the 'Storage' section.

2. Name the bucket `1flight\_data\_analysis` or update the script if you use a different name.

## 3 Preparing and Deploying the Script

1. Saved the script as `one.py` on my local machine.

2. Installed the required Python packages using pip or a `requirements.txt` file.

3. Deployed the Cloud Function using the following command:

gcloud functions deploy scrape\_flights \   
 --runtime python39 \   
 --trigger-http \   
 --allow-unauthenticated \   
 --entry-point scrape\_flights \   
 --timeout 540s

## 4 Triggering and Monitoring the Cloud Function

1. Trigger the Cloud Function via an HTTP request or directly through the Google Cloud Console.

2. Monitor the execution using logs available in the Cloud Function section of the Google Cloud Console.

## 5 Checking Uploaded Data

1. Navigate to your Google Cloud Storage bucket to verify that the CSV files containing the scraped data have been uploaded successfully.

## 6 Troubleshooting

1. Check the logs in the Google Cloud Console if the script fails.

2. Ensure the WebDriver path is correct and all necessary Python packages are installed.

3. Adjust timeouts or other settings in the script if needed.

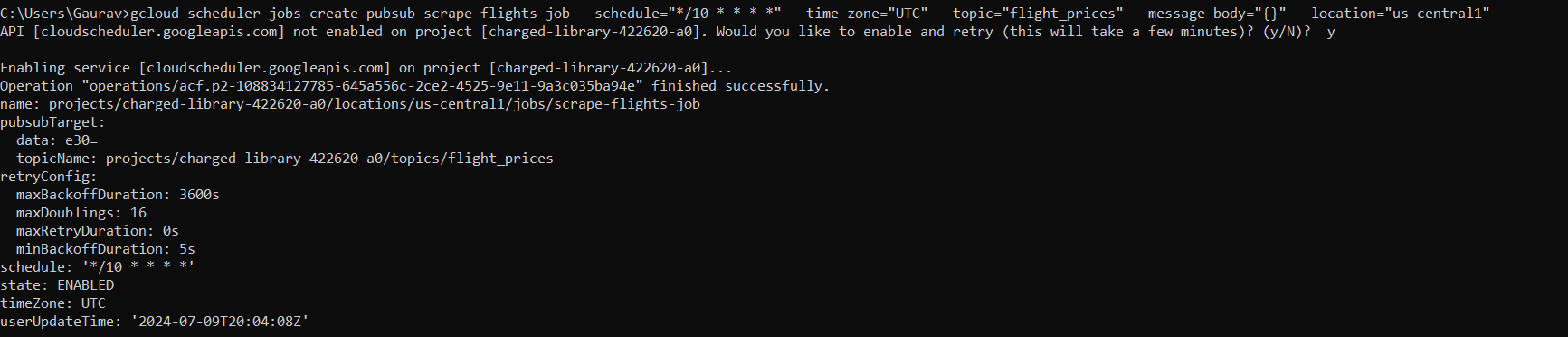
# 5. Conclusion

This project demonstrates the integration of web scraping with cloud storage using Google Cloud Functions. By automating the process of collecting flight data, this solution enables continuous monitoring and analysis of flight prices and availability, offering valuable insights for various use cases.

# 6. Screenshots

# A computer screen shot of a computer screen Description automatically generated

# Uploaded image



A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a graph

Description automatically generated