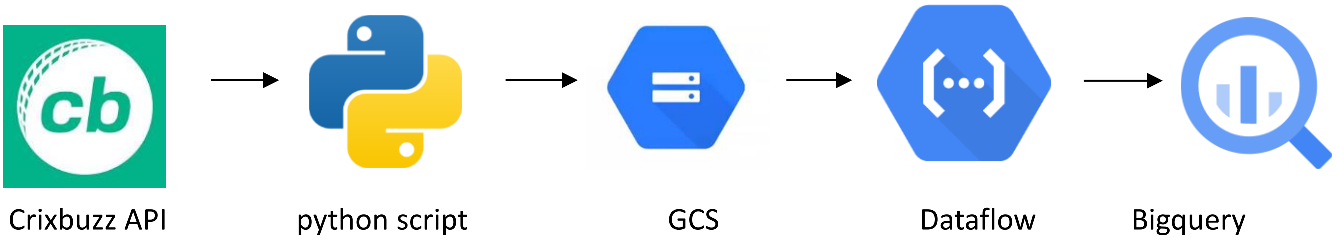
# Building a Cricket Statistics Pipeline with Google Cloud Services and stream it using Dataflow

Arhitecture:



This guide outlines the process of fetching data from RapidAPI, storing it in Google Cloud Storage (GCS), and then streaming it into BigQuery using Dataflow. The workflow consists of the following steps:

1. **Fetching data from RapidAPI using Python**
2. **Uploading the data to GCS**
3. **Streaming the data from GCS to BigQuery using Dataflow**

---

### Prerequisites

Before starting, ensure you have the following:

- A Google Cloud Project
- A BigQuery dataset
- A Cloud Storage bucket
- Dataflow API enabled

### Step 1: Storing Data in Google Cloud Storage (GCS)

Once the data is obtained, our next step involves preserving it securely in the cloud. We'll explore how to store this data in a CSV format within Google Cloud Storage (GCS), ensuring accessibility and scalability for future processing.

1. Create a bucket that will store the file and name the bucket.

2. Create one more bucket named “temp” that will store the temporary files by Dataflow during the processing.

Step2: **Go to Google > Search for Rapidapi**

-Click on this > [API Hub - Free Public & Open Rest APIs | Rapid (rapidapi.com)](https://rapidapi.com)

-Sign up with your Gmail account, Fill the Details & sign up

-Search Cricbuzz > select Cricbuzz Cricket

-Click on subscribe to test, Click on basic > Subscribe

-**Click on Endpoint**

**-Scroll Down to the end & select Stats**

**-Expand the stats > Select get-icc-Ranking**

**-In Code Snippets select target as “Python” & Client as “Requests”**

**-Copy this code**

```
import requests

url = "https://cricbuzz-cricket.p.rapidapi.com/stats/v1/rankings/batsmen"

querystring = {"formatType":"test"}

headers = {
	"x-rapidapi-key": "Sign Up for Key",
	"x-rapidapi-host": "cricbuzz-cricket.p.rapidapi.com"
}

response = requests.get(url, headers=headers, params=querystring)

print(response.json())
```

**-Create one file with .py extension**

1. **This Python script sends a GET request to the "https://cricbuzz-cricket.p.rapidapi.com/stats/v1/rankings/batsmen" API, requesting data in the "test" format for batsmen rankings.**

**2. It extracts the relevant data (rank, name, and country) from the API response, writes it to a CSV file named 'batsmen_rankings.csv', and then uploads this CSV file to a Google Cloud Storage (GCS) bucket named 'bkt-ranking-data'.**

**In the below Code**

- **Remove the url, query string, headers, response**
- **Add your Rapid code**

```
import requests
import csv
from google.cloud import storage

# Cricbuzz API details
url = "https://cricbuzz-cricket.p.rapidapi.com/stats/v1/rankings/batsmen"
querystring = {"formatType": "test"}
headers = {
    "X-RapidAPI-Key": "9a243e9716msh0aee7a2c1217623p18ec49jsn6fc9a0d217b4",
    "X-RapidAPI-Host": "cricbuzz-cricket.p.rapidapi.com"
}

# Fetch data from Cricbuzz API
response = requests.get(url, headers=headers, params=querystring)

if response.status_code == 200:
    data = response.json().get('rank', [])  # Extracting the 'rank' data
    csv_filename = "batsmen_rankings.csv"

    if data:
        field_names = ["rank", "name", "country"]  # Specify required field
names

        # Write data to CSV file with headers
        with open(csv_filename, "w", newline="", encoding="utf-8") as
csvfile:
            writer = csv.DictWriter(csvfile, fieldnames=field_names)
            writer.writeheader()  # Fix: Write column headers
            for entry in data:
                writer.writerow({field: entry.get(field) for field in
field_names})

        print(f" Data written to '{csv_filename}' successfully!")

        # Upload to GCS
        def upload_to_gcs(bucket_name, source_file_name,
destination_blob_name):
            """Uploads a file to Google Cloud Storage."""
            storage_client = storage.Client()
            bucket = storage_client.bucket(bucket_name)
            blob = bucket.blob(destination_blob_name)
            blob.upload_from_filename(source_file_name)

            print(f"File '{source_file_name}' uploaded to
gs://{bucket_name}/{destination_blob_name}")

        # Fix: Correct GCS path
        bucket_name = "landing_data_for_cricbuzz"
        destination_blob_name = "batsmen_rankings.csv"  # Just the filename,
NO 'gs://'

        upload_to_gcs(bucket_name, csv_filename, destination_blob_name)

    else:
        print("No data available from the API.")
else:
    print(f"Failed to fetch data: {response.status_code}")
```

**Note- Change the Bucket Name & give the your bucket name accordingly**

3. **To check above is working or not**

**-Open cloud shell**

**-Click on three dots > upload**

**-Choose file & upload, Now click on open editor**

**-Click on open file,  Select the file, which u uploaded**

**- click run**

**-It will automatically create file & uploads to the bucket**

4. **Go to Bucket & check if file is present or not**

**Step 3: Set Up BigQuery**

1. **Create a Dataset**
   - **Navigate to BigQuery in GCP Console.**
   - **Click Create Dataset and configure settings.**
2. **Create a Table**
   - **Inside the dataset, click Create Table.**
   - **Select Google Cloud Storage as the source.**
   - **Set schema to auto-detection.**

**Step 4: Use Dataflow to Stream Data into BigQuery**

1. **Enable Dataflow API**
   - Navigate to **APIs & Services**.
   - Enable **Dataflow API**.
2. **Create a Dataflow Job Using GCP Console**
   - Go to the **Dataflow** service in the GCP Console.
   - Click **Create Job from Template**.
   - In the **Template** dropdown, select **"CSV files on cloud storage to BigQuery"**.
   - Provide the following inputs:
     - **Job Name**: Enter a unique job name.
     - **Cloud Storage Input File**: The Cloud Storage path to the CSV file that contains the text to process. For example, `gs://your-bucket/path/*.csv`

**{**

**Select your bucket from the list in GCS.**

**Navigate to the file whose path you need.**

**Click on the file to open its details.**

**Copy the "gsutil URI" (e.g., gs://your-bucket-name/path/to/your-file.csv).**

**}**

- **BigQuery Output Table**: your-project-id:your_dataset.your_table
- **Temporary Location**: gs://your-bucket-name/temp/

- Click **Run Job** to start the pipeline.
- Monitor progress in **Cloud Dataflow**.