



DATA INGESTION PIPELINE

Note : The screen shots indicated in the documentation may/may not reveal the actual GCP object names used in the project and are for guidance and reference purposes for process followed in creating the data pipeline.

The data pipeline part consists of pulling historical and daily data from API and ingest into data lake (GCS bucket) . A Google Cloud Function and Google Scheduler has been used to ingest data everyday at 1600 hours .

Step 1

Create a project in GCP .

 **Project info** 

Project name
DTC-DE-course

Project number
205136287098

Project ID
dtc-de-course-338717

[ADD PEOPLE TO THIS PROJECT](#)

[→ Go to project settings](#)

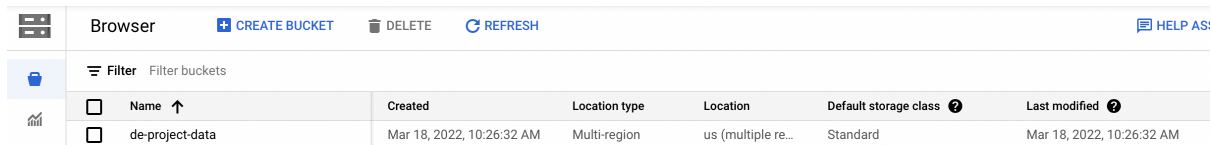
Step 2

Create a Storage bucket within this bucket .

(a) click on Storage on project page or search for storage in search box on top

(b) This shows list of storage buckets in our project

(C) Use create bucket to create a storage bucket . I have chosen US-multi region as bucket location . The name of the bucket is de-project-data. Use standard config and fine grained .



The screenshot shows the Google Cloud Storage browser interface. At the top, there's a 'Browser' tab with buttons for 'CREATE BUCKET', 'DELETE', and 'REFRESH'. Below this is a 'Filter' section with 'Filter buckets' text. The main area is a table with columns: Name, Created, Location type, Location, Default storage class, and Last modified. A single bucket named 'de-project-data' is listed with a creation time of 'Mar 18, 2022, 10:26:32 AM', 'Multi-region' location type, 'us (multiple re...' location, 'Standard' storage class, and the same creation time for 'Last modified'.

Name	Created	Location type	Location	Default storage class	Last modified
de-project-data	Mar 18, 2022, 10:26:32 AM	Multi-region	us (multiple re...	Standard	Mar 18, 2022, 10:26:32 AM

Step 3

Now that we have created our bucket in GCS , we will head to creating a Google Cloud Function to write a python function to ingest data into our above bucket. This GCF will be triggered at a particular time everyday using a Google Cloud Scheduler.

[Google Cloud Functions](#) is a Function as a Service (FaaS) that allows engineers and developers to run code without worrying about server management. Cloud Functions scales as needed and integrates with Google Cloud's operations suite (such as Cloud Logging) out of the box. Functions are useful when you have a task or series of tasks that need to happen in response to an event. Cloud Functions can be invoked from several events, such as HTTP, Cloud Storage, Cloud Pub/Sub, Cloud Firestore, Firebase, and in response to Google Cloud Logging events. With Cloud Functions, monitoring, logging, and debugging are all integrated, and functions scale up and down as required.

For ingesting COVID data from API , we will create a Cloud Function (trigger -http) with a python script . The Cloud Function will be run on a schedule by using Cloud Scheduler

At the beginning , Enable Cloud Functions API (API and Services .>> Library >> API Enabled

Generate a Service Account Key

Following steps will generate a service account key.

Select the hamburger menu from the upper left-hand corner of the Google Cloud

Platform console.

Select APIs & Services.

Select Credentials. (image 9)

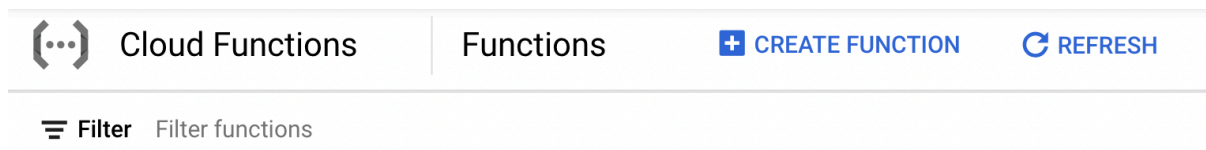
Press the blue CREATE CREDENTIALS button and select Service account.

Give the new service account a name and description then press Create.


Set the Owner role on the service account and leave the other fields blank.


Creating a Cloud Function


Now head to Google Cloud Functions console (type Google Cloud Function) in search box





Hit on CREATE FUNCTION. The following screen takes you to basic configuration of the CF . Provide a name for the function and select http trigger.

 Cloud Functions

 Create function

Name *
function-2 

Memory allocated *
256 MiB 

Trigger
HTTP 

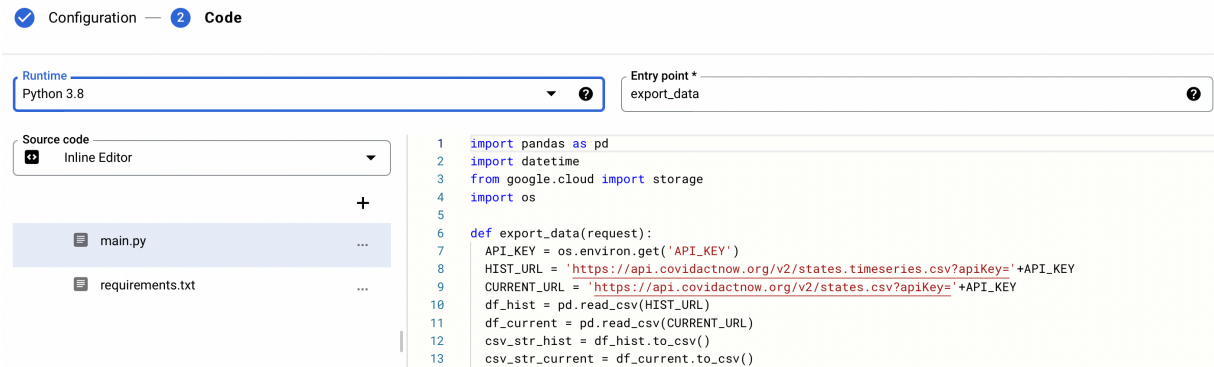
URL

you will get a Trigger URL. Copy this URL somewhere as this is the URL we will use to invoke our function. The Authentication box is the tricky bit. If selected then **anyone** can make a call to the function.

This is not desired , and the URL has to be kept without sharing with anybody. We can choose restricted access ."SAVE". Then hit "NEXT".

We will not provide API_KEY in the code . In the add env variables , enter variable as API_KEY and value as actual key value. , Next button will take you to run time environment where we write our script .

Use ingest_data.py in GitHub repo as the our python function . Use the function name in the entry point . We have selected "Python 3.8". The "Entry point" beside the "Runtime" shows the method which will be invoked when a request has been received. Under the source code section, you will also see the requirements.txt file. This file is used to specify additional dependencies for your function Now let's hit "DEPLOY" to deploy the function. Now you will again be redirected to the functions manager site. It should take about 1-2 minutes for the function to be deployed. We can Test our function once it is deployed . The testing part is carried out by clicking three dots at the right corner CF console specific to our function and hit TEST THE FUNCTION .



The testing should give us a satisfactory result and files in the GCS bucket. The python code applies following logic to pull the data and put into GCS bucket

(a) Use pandas read_csv(URL) function to read the CSV files (with historical and current data) into a data frame

(B) the df.tocsv() converts this back to a csv file and saves it in a variable

(C) This string object is uploaded to GCS bucket using blob.upload_from_string() method from google.cloud.storage .

(d) we upoad files by calling a internal function upload_blob() within main function

(e) We have also merged historic data and current data to create a merged data frame . The function called merge_data() with two data frames as arguments is used for this purpose. We have uploaded the combined data frame to GCS bucket in similar manner.


For merging the historic and current data following info are pertinent :-

(a) The historic and current df have columns names **unused** which are basically not relevant columns of data


(b) There are 39 columns in historic data and 46 columns in current data and 37 common columns between them


(c) removed non matching columns from both data frames and also renamed date column in current data for matching

(d) Used pd.append function for merging the two data frames one below another

Output  Complete

```
$ Job Completed
```

Logs  Fetched (up to 100 entries). [View all logs](#)

 Scanned up to 12/27/21, 11:00 AM. Scanned 2.2 KB.

Cloud Functions

Functions

[CREATE FUNCTION](#)









[REFRESH](#)

[LEARN](#)

[RELEASE NOTES](#)

Filter

Filter functions

<input type="checkbox"/>		Environment	Name 	Region	Trigger	Runtime	Memory allocated	Executed function	Last deployed	Authentication 	Actions
<input type="checkbox"/>		1st gen	de-project-data-upload	us-east1	HTTP	Python 3.8	512 MB	export_data	Mar 19, 2022, 10:17:06 PM		<div><div> Copy function</div><div> Test function</div><div> View logs</div><div> Delete</div></div>

Permissions for de-project-test

Edit or delete permissions below or "Add Principal" to grant new

 **ADD PRINCIPAL**



Show inherited permissions

Add principals to "de-project-test"

Add principals and roles for "de-project-test" resource

Enter one or more principals below. Then select a role for these principals to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New principals *



Select a role *



 ADD ANOTHER ROLE

SAVE

CANCEL

Now head over to Cloud Scheduler console (type Cloud Scheduler in search box) and click on CREATE JOB



Cloud Scheduler

Jobs

 **CREATE JOB**



SCHEDULER JOBS

APP ENGINE CRON JOBS 

]

- **Define the schedule**

Name *

Must be unique across the jobs in the same region

Region *

Description

Frequency *

Schedules are specified using unix-cron format. E.g. every minute: "* * * * *", every 3 hours: "0 */3 * * *", every monday at 9:00: "0 9 * * 1". [Learn more](#)

Timezone *

Jobs in set in timezones affected by Daylight Saving Time can run outside of cadence during DST change. Using a UTC timezone can avoid the problem. [Learn more](#)

CONTINUE

Provide a name to the Scheduler. The Region is as per your location . The description is short description of what does this scheduler is doing. The Frequency is entered in cronjob format. As I am running the CF every dat at 1600 hours , the format is 0 16 * * *

Please refer to website **crontab guru** for finalizing the scheduler job frequency

- **Configure the execution**

Target type *
HTTP

URL *

HTTP method
GET

HTTP headers
Some headers are set to default values or removed by Cloud Scheduler. [Learn more](#)

+ ADD A HEADER

Auth header
Add OIDC token

Service account *

This service account must have permission to invoke the target. For example, the Cloud Functions Invoker role is required to schedule a Cloud Function. [Learn more](#)

Audience

The URL is the http URL from the CF .

Use GET http method

The Service Account is the CF invoker service account we just created . . HIT CREATE to create our scheduler.

Cloud Scheduler										
Jobs										
<div>CREATE JOB REFRESH EDIT COPY PAUSE RESUME DELETE</div>										
SCHEDULER JOBS APP ENGINE CRON JOBS										
Filter Filter jobs										
<input type="checkbox"/>	Name ↑	Region	State	Description	Frequency	Target	Last run	Last run result	Next run	Logs Run
<input type="checkbox"/>	test-scheduler	us-east1	Enabled		0 14 * * * (America/Detroit)	URL : https://us-east1-dtc-de-course-338717.cloudfunctions.net/de-project-test	Mar 18, 2022, 2:00:00 PM	Success	Mar 19, 2022, 2:00:00 PM	View RUN NOW

The scheduler we created is reflected on the console . This will also tell you when the schedule was last run and when is next slotted for execution. You may test the Cloud Scheduler by RUN NOW.

DATA INGESTION

As we have created the required objects in GCP for data ingestion part , we can monitor the execution part . At the scheduled time , the GCS bucket will have two CSV files , one historical data and one current data.

A combined data is also created . I have used logic of overwriting the data everyday (we can create data for each day pull and GCS bucket) .

I have tested for automatic data ingestion by monitoring the bucket (we can also include a email script in CF to get an email on loading of data) for two days in sequence and found satisfactory.