leaplogic
an IMPETUS product

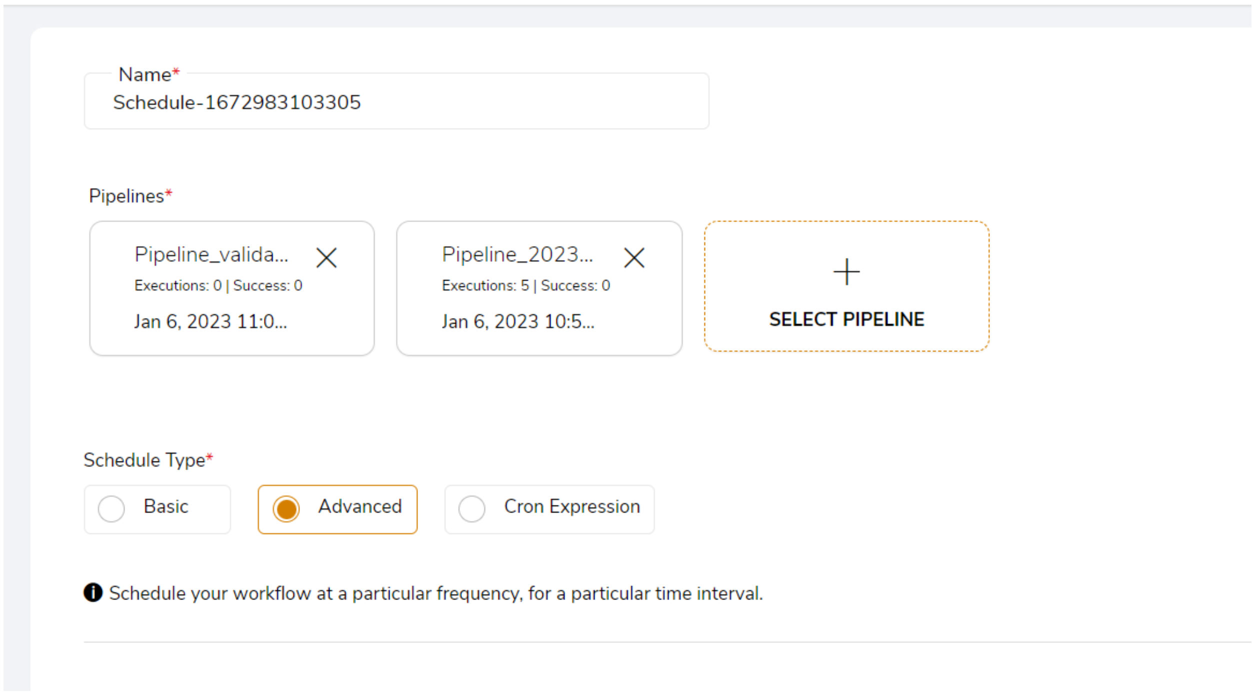
Pipeline Scheduling using GCP Function

Contents

1. Automatic Push

This option allows LeapLogic to generate GCP Function code dynamically and push that to your selected Google cloud environment to generate GCP Function. It also allows to trigger the pipeline execution or schedule pipelines. You can provide the credentials of the respective cloud environment in the given format.

1. Go to Operationalization > Parallel Run
2. Select the pipelines that need to be scheduled.



Name*
Schedule-1672983103305

Pipelines*

Pipeline_valida...
Executions: 0 | Success: 0
Jan 6, 2023 11:0...

Pipeline_2023...
Executions: 5 | Success: 0
Jan 6, 2023 10:5...

Schedule Type*
Basic [✓ Advanced] Cron Expression

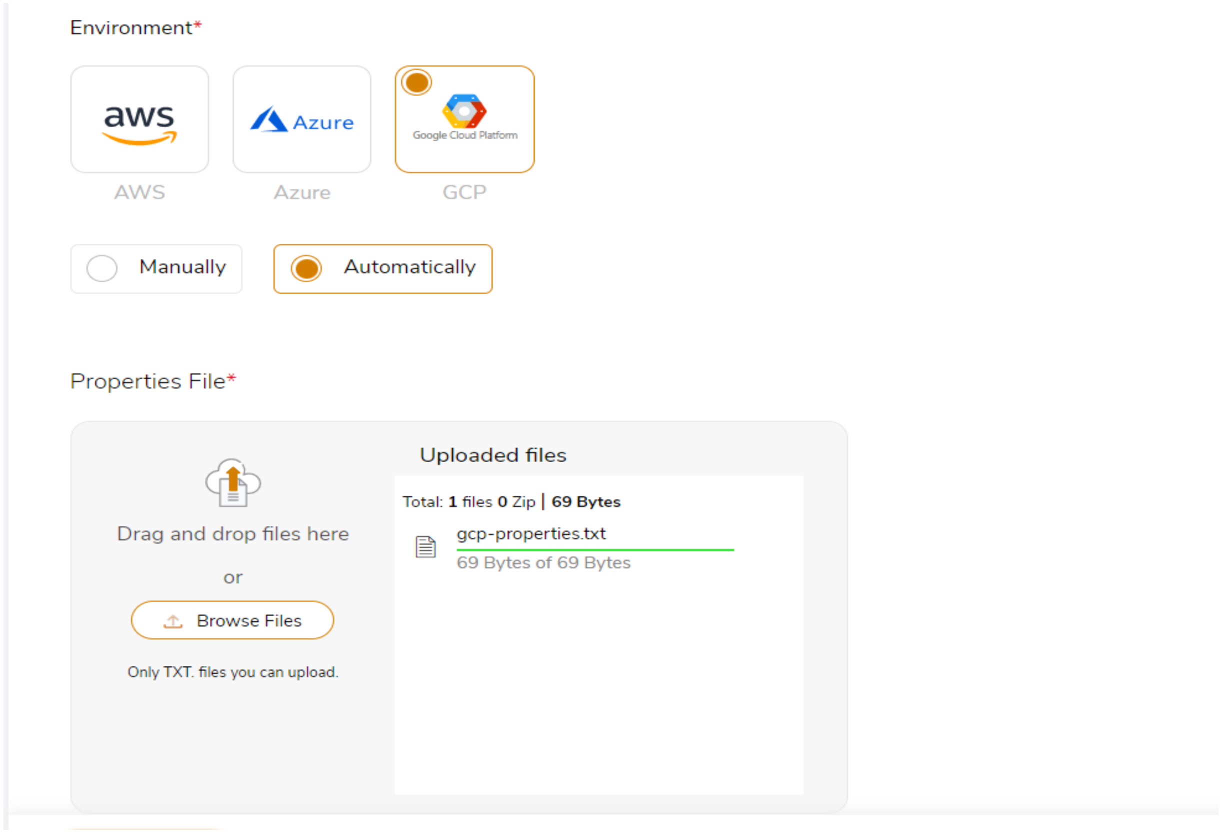
Schedule your workflow at a particular frequency, for a particular time interval.

3. Click **Advanced** as schedule type.
4. Select Environment for advance trigger. Select GCP
5. Click **Automatically**. You can upload the properties file as per the below format

```
projectId=<Project Id>
bucketName=<bucket name>
region=<Region>
vpcConnectorName=<Optional Serverless VPC Connector Name>
```

Note

vpcConnectorName is optional. These fields are required if LeapLogic is deployed on Google cloud and available only in a closed network. You need to create VPC connector with same network as LeapLogic is deployed. To create VPC Connector, pass the connector name in the input properties file. If LeapLogic is available in an open network, then do not create/provide VPC Connector details.


Environment*

Manually [✓ Automatically]

Properties File*

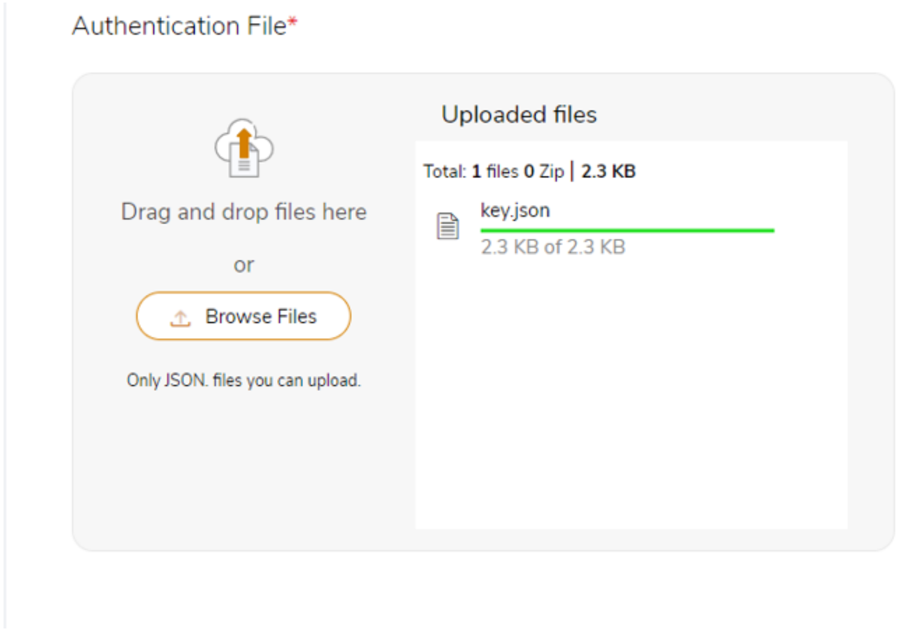
Only TXT. files you can upload.

Uploaded files

Total: **1** files **0** Zip | **69 Bytes**

gcp-properties.txt
69 Bytes of 69 Bytes

6. Upload the Authentication File.


Authentication File*

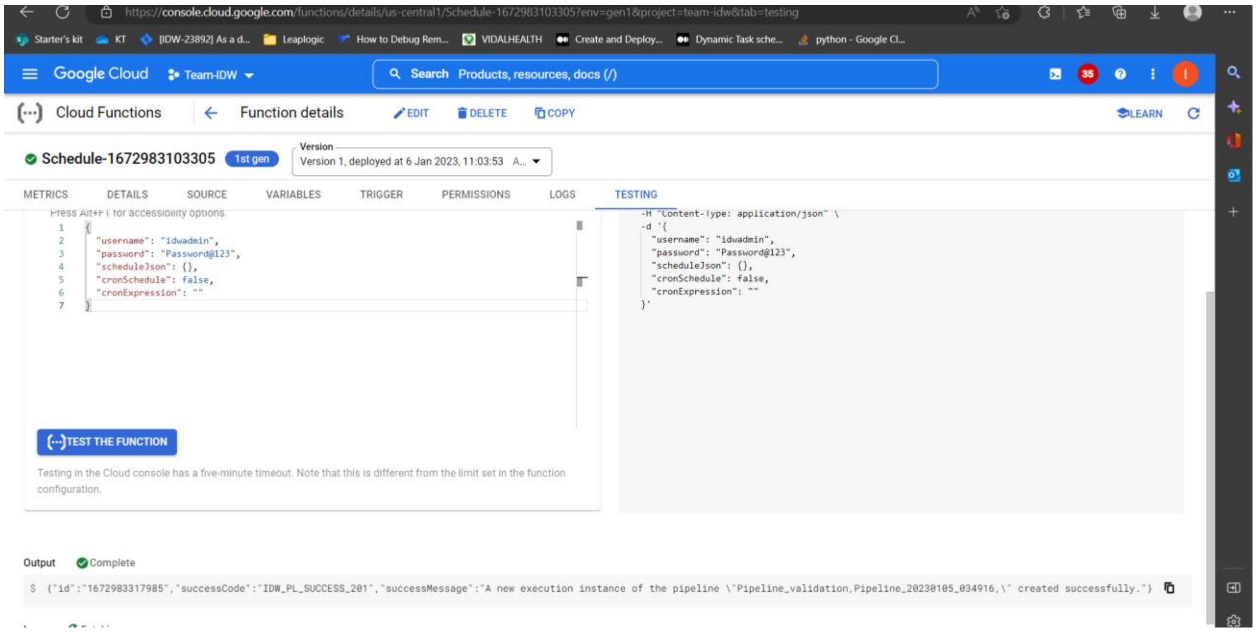
Only JSON. files you can upload.

Uploaded files

Total: **1** files **0** Zip | **2.3 KB**

key.json
2.3 KB of 2.3 KB

7. Click **Schedule**. This generates GCP Function on your cloud environment with the provided GCP environment details.



Press Alt+F1 for accessibility options.

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": false,
  "cronExpression": ""
}
```

```
-H "Content-Type: application/json" \
-d '{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": false,
  "cronExpression": ""
}'
```

Testing in the Cloud console has a five-minute timeout. Note that this is different from the limit set in the function configuration.

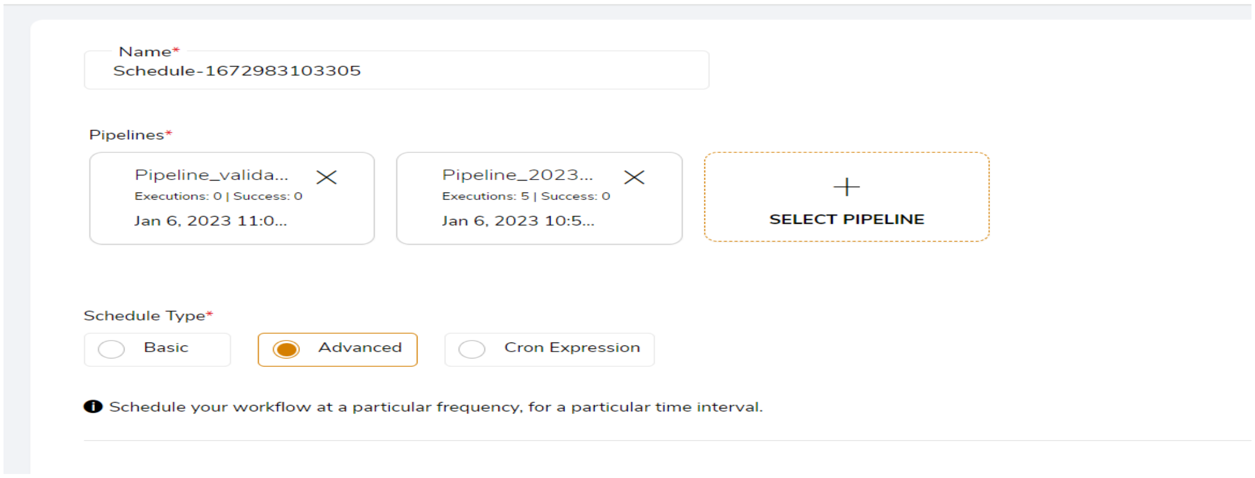
Output Complete

```
$ {"id":"1672983317985","successCode":"IDW_PL_SUCCESS_201","successMessage":"A new execution instance of the pipeline \"Pipeline_validation,Pipeline_20230105_034916,\" created successfully."}
```

2. Manual

This option allows LeapLogic to generate GCP Function code dynamically. You can download the generated code in zip format and generate GCP Function manually.

1. Go to Operationalization > Parallel Run
2. Select the pipelines that need to be scheduled.



Name*
Schedule-1672983103305

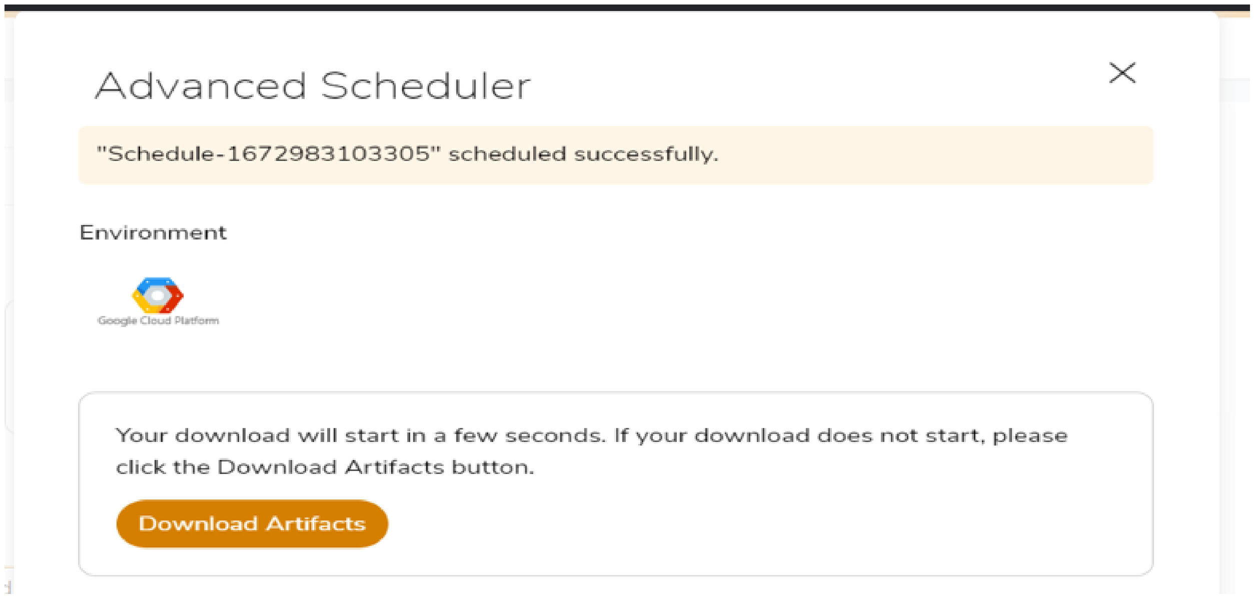
Pipelines*

Pipeline	Executions	Date
Pipeline_valida...	Executions: 0 | Success: 0	Jan 6, 2023 11:0...
Pipeline_2023...	Executions: 5 | Success: 0	Jan 6, 2023 10:5...

Schedule Type*
Basic [✓ Advanced] Cron Expression

Schedule your workflow at a particular frequency, for a particular time interval.

3. Click **Advanced** as schedule type.
4. Select Environment for advance trigger and then select **GCP.**
5. Click **Manually**
6. Click **Schedule**. The download artifact option appears.



Advanced Scheduler

"Schedule-1672983103305" scheduled successfully.

Environment

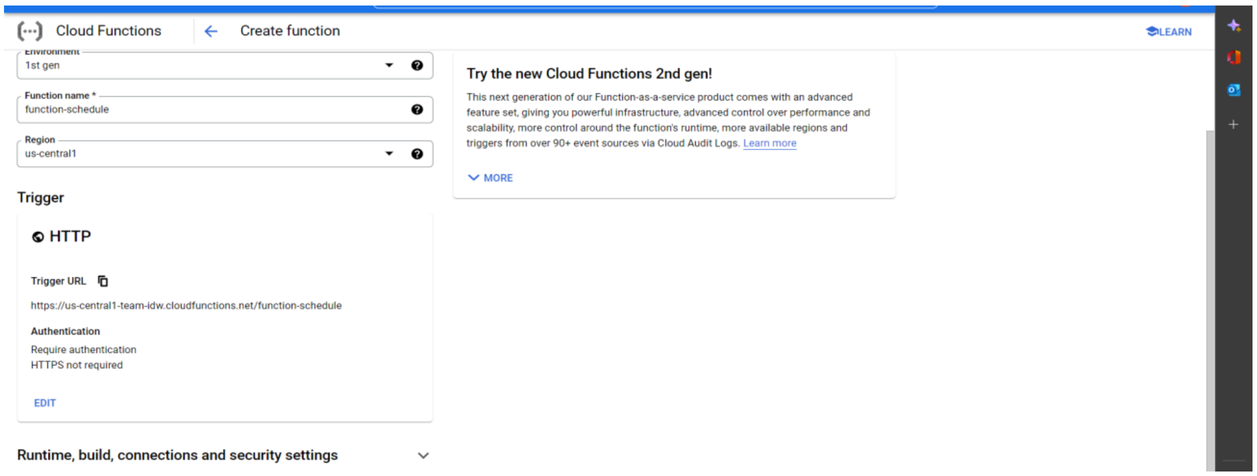
Your download will start in a few seconds. If your download does not start, please click the Download Artifacts button.

7. Download the zip file if not automatically downloaded.

2.1 Creating GCP Function Manually

1. Go to Cloud Function from Google Cloud Console.

2. Click **Create Function**



Environment
1st gen

Function name *
function-schedule

Region
us-central1

Trigger

HTTP

Trigger URL

https://us-central1-team-idw.cloudfunctions.net/function-schedule

Authentication

Require authentication
HTTPS not required

EDIT

Runtime, build, connections and security settings

Try the new Cloud Functions 2nd gen!

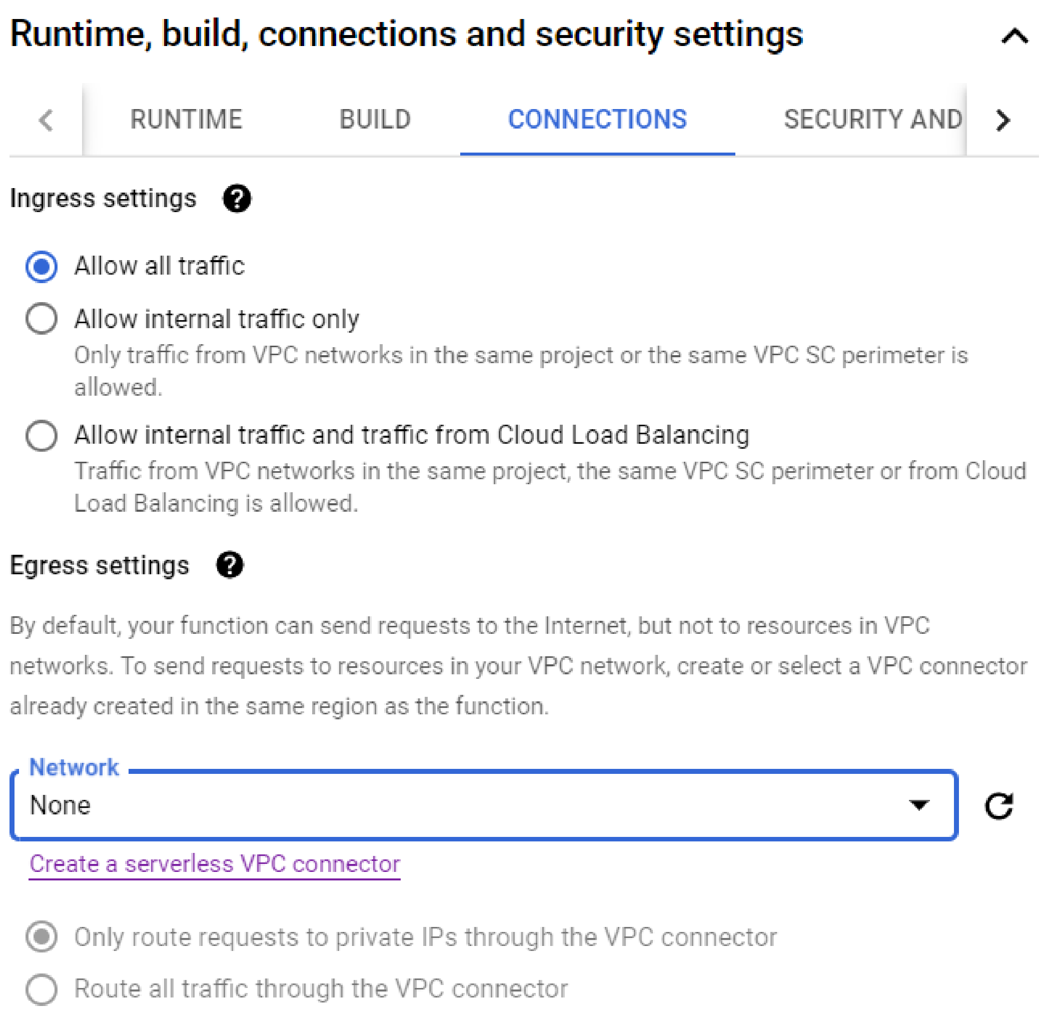
This next generation of our Function-as-a-service product comes with an advanced feature set, giving you powerful infrastructure, advanced control over performance and scalability, more control around the function's runtime, more available regions and triggers from over 90+ event sources via Cloud Audit Logs. Learn more

MORE

3. Provide the appropriate function name.
4. Choose VPC Connector if required.

Note

VPC Connector is optional. These fields are required if LeapLogic is deployed on Google Cloud and available only in a closed network. You need to create the VPC connector on the same network as LeapLogic is deployed. Pass the connector name in the properties file. If LeapLogic is available in an open network, then do not provide any input in this field.



Runtime, build, connections and security settings

RUNTIME | BUILD | CONNECTIONS | SECURITY AND

Ingress settings

[✓ Allow all traffic]

Allow internal traffic only
Only traffic from VPC networks in the same project or the same VPC SC perimeter is allowed.

Allow internal traffic and traffic from Cloud Load Balancing
Traffic from VPC networks in the same project, the same VPC SC perimeter or from Cloud Load Balancing is allowed.

Egress settings

By default, your function can send requests to the Internet, but not to resources in VPC networks. To send requests to resources in your VPC network, create or select a VPC connector already created in the same region as the function.

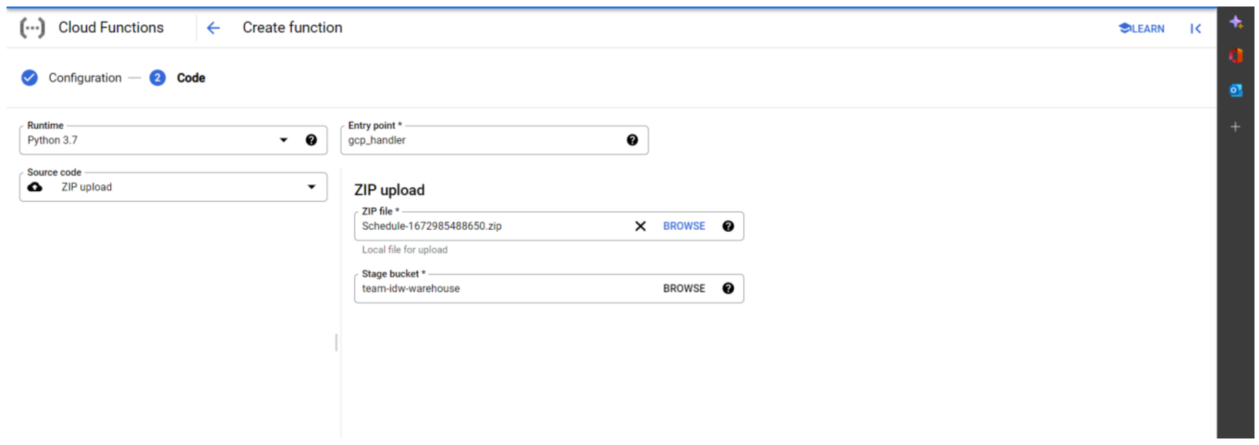
Network: None

Create a serverless VPC connector

[✓ Only route requests to private IPs through the VPC connector]

Route all traffic through the VPC connector

5. Click **Next**
6. Select Python 3.7 as runtime and provide the entry point as **schedule_handler.**
7. In Source Code, select ZIP upload.
8. Upload the zip file downloaded at the time of scheduling.



Cloud Functions ← Create function

Configuration — 2 Code

Runtime: Python 3.7

Entry point *: gcp_handler

Source code: ZIP upload

ZIP upload

ZIP file *: Schedule-1672985488650.zip BROWSE
Local file for upload

Stage bucket *: team-idw-warehouse BROWSE

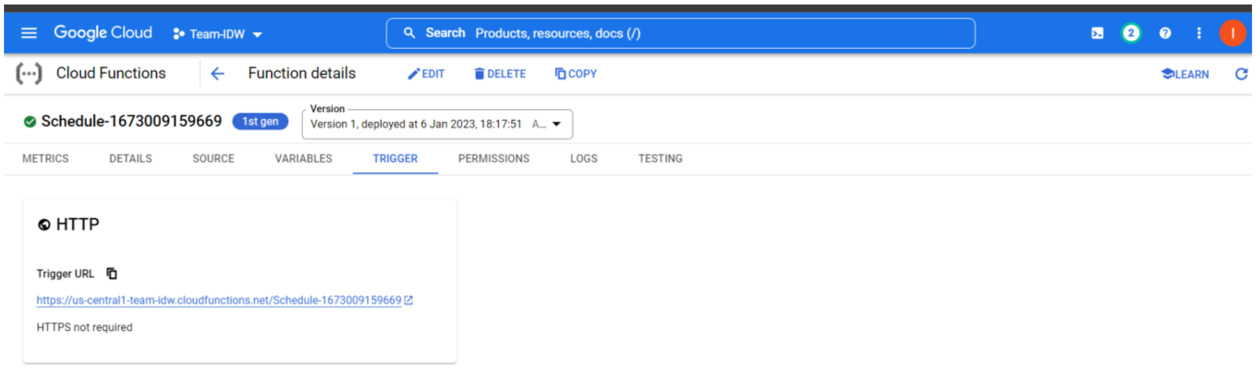
9. This generates and deploys the GCP Function.

2.2 Using GCP Function to Schedule/Execute Pipeline

You can now execute or schedule the pipelines as per your requirement by triggering GCP Function with appropriate JSON. You can also provide its credentials in JSON to authorize/authenticate beforehand.

Example: Execute GCP Function by triggering REST endpoint generated.

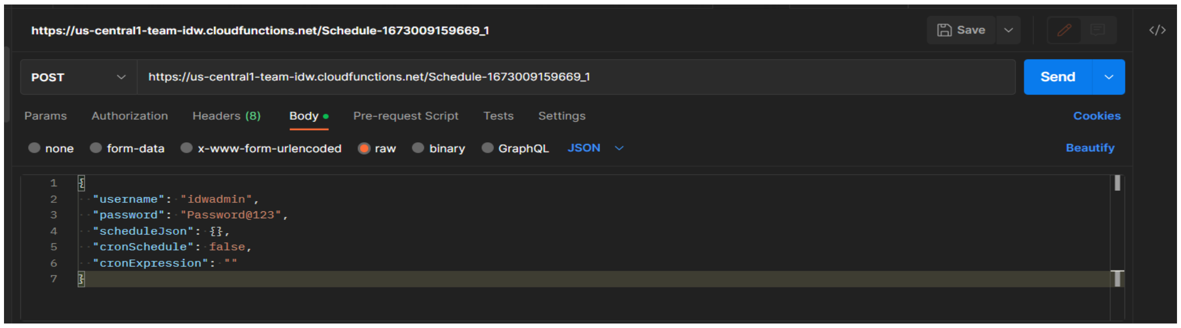
1. Go to User to find the Trigger URL as shown below



https://us-central1-team-idw.cloudfunctions.net/Schedule-1673009159669

HTTPS not required

2. Use this Trigger URL to trigger the pipeline with appropriate JSON.



POST https://us-central1-team-idw.cloudfunctions.net/Schedule-1673009159669_1

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": false,
  "cronExpression": ""
}
```

i. **JSON to execute pipelines**

```
{
 "username": "idwadmin",
 "password": "Password@123",
 "scheduleJson": {},
 "cronSchedule": false,
 "cronExpression": ""
}
```

ii. **JSON to schedule with basic details**

```
{
 "username": "idwadmin",
 "password": "Password@123",
 "scheduleJson": {
  "startDate": "2022-12-27",
  "endDate": "2022-12-27",
  "startTime": "18:05",
  "endTime": "18:06",
  "minutes": 0,
  "timezone": "Asia/Calcutta",
  "frequency": "NONE"
 },
 "cronSchedule": false,
 "cronExpression": ""
}
```

Note

Frequency can be NONE(Once), DAILY, WEEKLY, MONTHLY, YEARLY, CUSTOM. With CUSTOM, you can provide minutes to indicate interval of minutes for schedule.

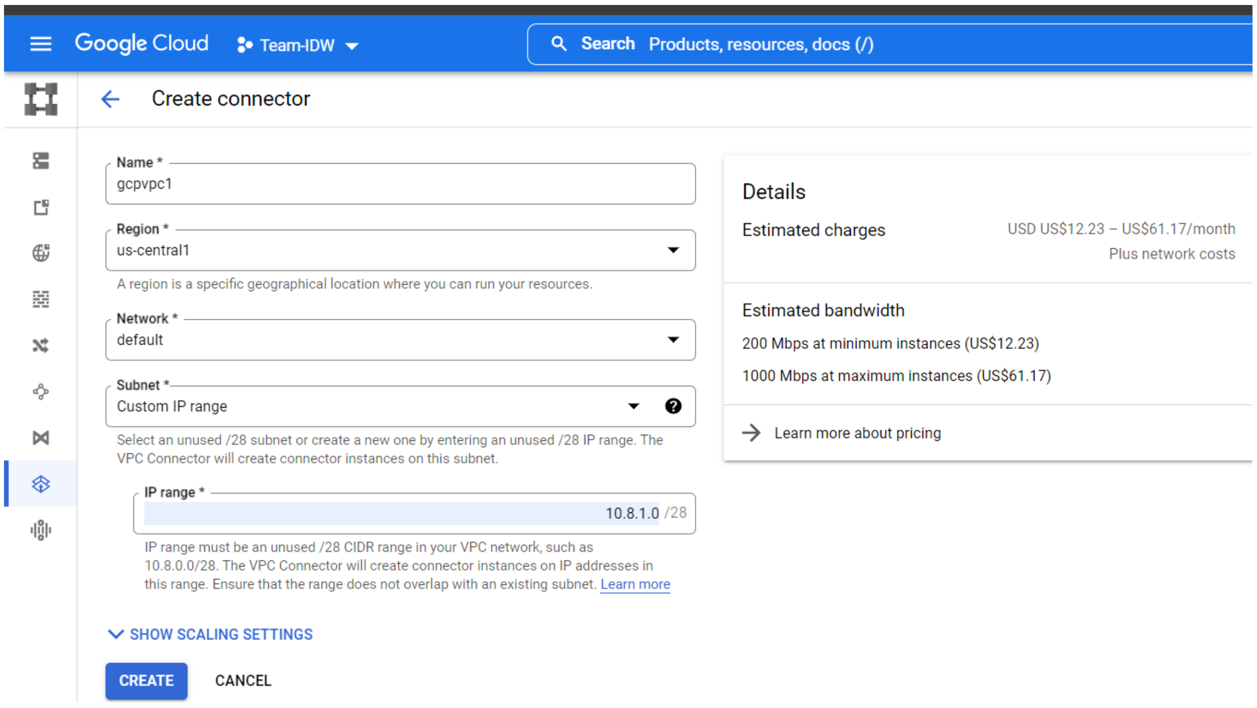
iii. **JSON to schedule with cron expression**

```
{
  "username": "idwadmin",
  "password": "Password@123",
  "scheduleJson": {},
  "cronSchedule": true,
  "cronExpression": "0 30 18 27 12 ? 2022"
}
```

3. You can curl the API URL as well.
4. You can integrate the API URL in this application as well.

2.3 Creating VPC Connector

1. Go to Serverless VPC Access in the console.
2. Choose the network and region same as compute engine instance.



Name *
gcpvpc1

Region *
us-central1

A region is a specific geographical location where you can run your resources.

Network *
default

Subnet *
Custom IP range

Select an unused /28 subnet or create a new one by entering an unused /28 IP range. The VPC Connector will create connector instances on this subnet.

IP range *
10.8.1.0 /28

IP range must be an unused /28 CIDR range in your VPC network, such as 10.8.0.0/28. The VPC Connector will create connector instances on IP addresses in this range. Ensure that the range does not overlap with an existing subnet. Learn more

Details

Estimated charges	USD US$12.23 – US$61.17/month Plus network costs

Estimated bandwidth

200 Mbps at minimum instances (US$12.23)

1000 Mbps at maximum instances (US$61.17)

Note

When VPC connector is used, GCP Function can communicate with LeapLogic through internal IP in closed network. For that, ensure flag gcp.connector.private.ip.enabled is set to Y in turin-framework.properties. In case of open network (https deployment), flag gcp.connector.private.ip.enabled should be set to N.

3. Getting Help

Contact LeapLogic technical support at info@leaplogic.io