National College of Ireland

Configuration Manual

MSc Research Project

MSc. in Cloud Computing

Gurnam Sunil Arora

Student ID: 22142525

School of Computing

National College of Ireland

Supervisor: Dr. Punit Gupta

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Gurnam Sunil Arora		
Student ID:	22142525		
Programme:	Cloud Computing	**Year:**	2023-2024
Module:	MSc Research Project		
Lecturer:	Dr. Punit Gupta		
Submission Due Date:	14/12/2023		
Project Title:	Optimising Inter-Function Communication in Serverless Computing through Network-Aware Adaptive Data Compression		
Word Count:	1688		
Page Count:	9		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Gurnam Sunil Arora

Date: 13/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Gurnam Sunil Arora
Student ID: 22142525

1 Introduction

This configuration manual provides comprehensive guidelines and instructions for setting up and running the experiments discussed in the research paper "Optimising Inter-Function Communication in Serverless Computing through Network-Aware Adaptive Data Compression".The research employs a two-pronged methodology through simulations and a real-world application deployment to evaluate the efficacy of an Adaptive Data Compression (ADC) technique in enhancing performance of communications between distributed serverless functions.

In order to enable reproducibility and repeatability of the experiments, this manual covers end-to-end configuration and execution across three key areas:

1. Preliminary Experiments:

- Deploying a cloud function to run compression/decompression using the ADC library and gather vital metrics on compression factor, compression times and decompression times across diverse workloads.

2. Simulation Setup:

- Configuring and executing WorkflowSim simulations of representative workflows (Montage, Inspiral).
- Imputing vital metrics from preliminary experiments to enable a comparative 'compression on' vs 'compression off' analysis.

3. Real-World Application Deployment:

- Deploying a serverless data analytics application integrated with the ADC library on a cloud platform (GCP).
- Instrumenting performance monitoring and running load tests to evaluate efficacy of ADC under operational conditions.

The manual is structured to provide intuitive, step-by-step directions to re-run key experiments that validate the potential of adaptive data compression in enhancing inter-function communications and overall performance of serverless applications.

Detailed configuration specifications related to cloud platforms, simulation software, and sample workloads are presented in their respective sections. The document also covers best practices in results analysis and visualisation to effectively compare compression enabled vs disabled scenarios.

.

2 Preliminary Experiment Setup

The preliminary experiments are focused on using Google Cloud Functions to perform compression/decompression on synthetic workloads and record key metrics. This section provides step-by-step guidelines on recreating this experimental setup on Google Cloud Platform (GCP).

2.1 Prerequisites

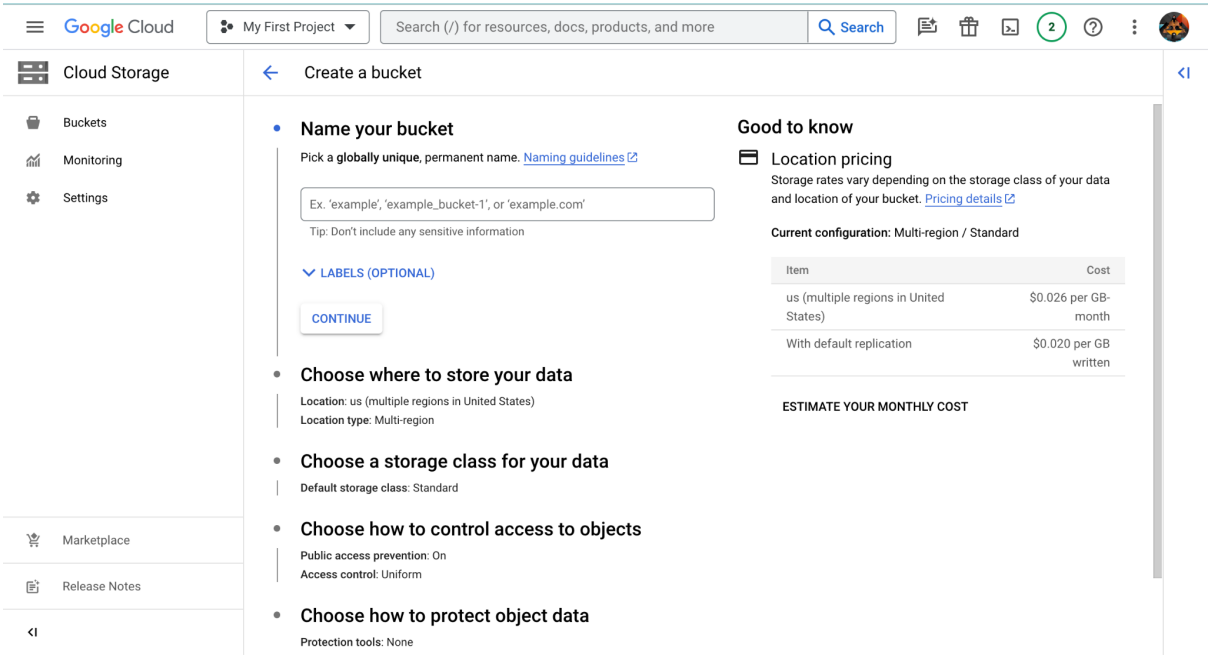
- Google Cloud account with billing enabled
- Google Cloud SDK installed on your machine (optional)
- Python 3.9
- Codebase and synthetic payloads archive (available in ICT Solution artefact zip provided with this manual)

2.2 Setting Up Cloud Storage

The synthetic payload files used in the experiments have been made available in the 'payloads' folder within the code artefacts.

To set up cloud storage:

- Open GCP Console and navigate to Cloud Storage service to create a Cloud Storage bucket
- Click on the ‘Create Bucket’ button to create a new bucket



- Give a name to your bucket and select a region, preferably closest to your location.
- Set the storage class as ‘Standard’ which is the default option.
- Disable public access prevention for this bucket (optional)

- **Choose how to control access to objects**

  Prevent public access

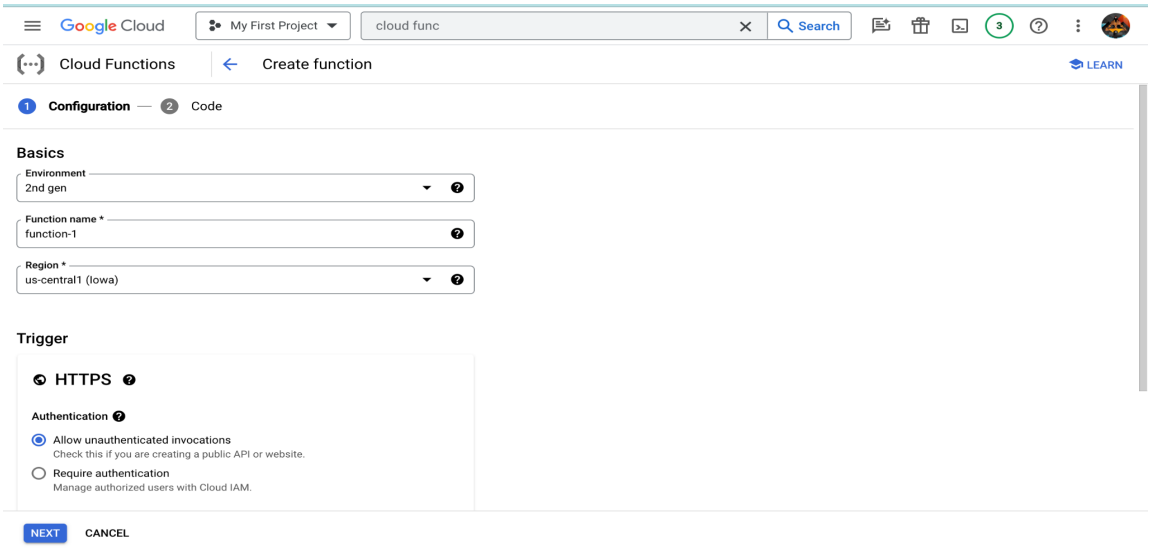
  Restrict data from being publicly accessible via the internet. Will prevent this bucket from being used for web hosting. Learn more

  Enforce public access prevention on this bucket

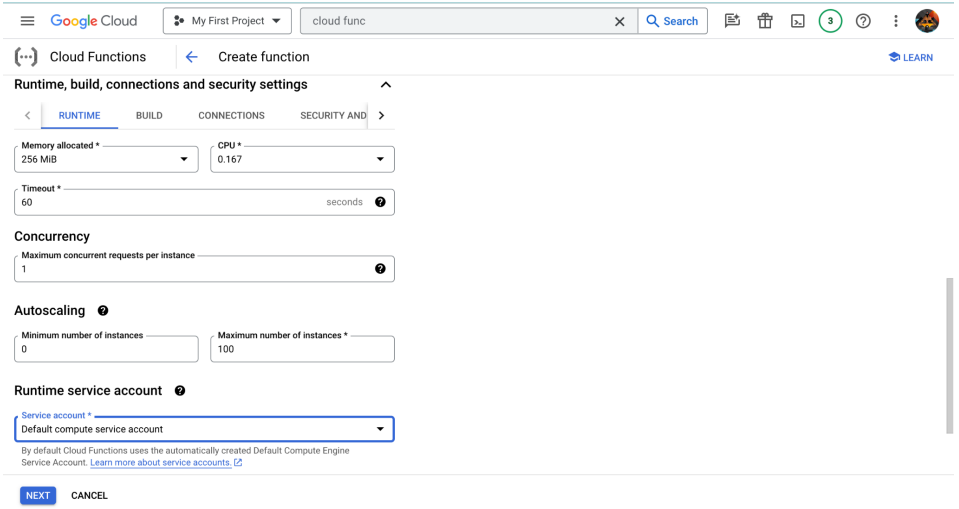
- Upload all files from the 'payloads' folder provides in the ICT Solution artefact zip to the newly created bucket
- Note down the name and for this storage bucket as this will be required later.

2.3 Deploying the Cloud Function

- Navigate to Cloud Functions in GCP Console
- Click on the ‘Create Function’ Button



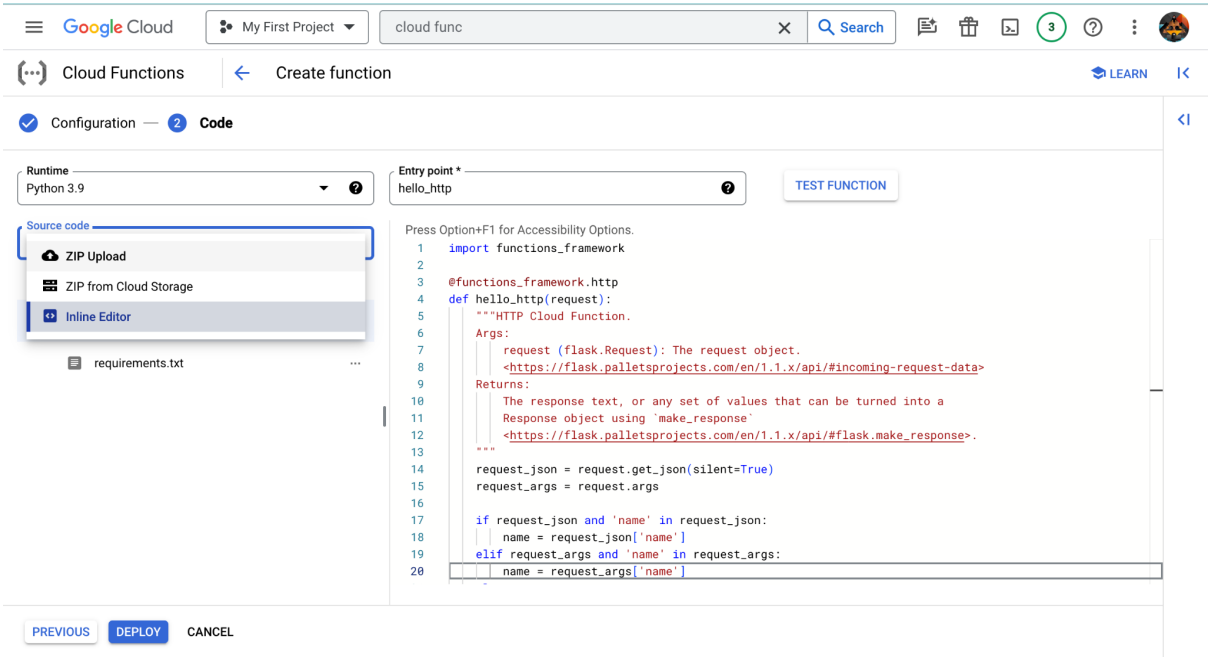
- Select 2nd Gen as Environment, give a name to this function and select ‘allow unauthenticated invocations’ in the Trigger.



- Next change the Runtime configuration for this function to have 1 CPU and 1 GB Ram and set timeout to 3600 seconds. These changes are necessary to allow the

function to have enough compute, memory and time to run and collect the critical metrics

- Next set the Service Account as ‘Default Compute Service account’ and click next.



- Select python 3.9 as runtime and choose Zip Upload for Source Code. The zip file containing the source code for this function is available within the ICT Solutions Artefact zip.
- Change the Entry point from hello_http to ‘process_files’ (this is the name of the function within main.py)
- Update the bucket name variable within main.py to the name you chose for creating a bucket in section 2.2 and Click Deploy
- We must now set permissions to allow this function to be invoked by anybody using its Http Triger URL. To Do this open the cloud shell terminal by clicking its icon on the top right and paste the following code

```
gcloud functions add-invoker-policy-binding function-1 \
      --region="us-central1" \
      --member="allUsers"
```

2.4 Invoking the Function

- Click on the Http trigger to trigger the function. When the function is finished executing, The function prints out Compression Factor, Compression Time and Decompression Time on the browser screen for all payloads within the cloud storage bucket. These values allow us to determine the critical metrics which are then utilised in simulation setup

3. Simulation Setup

This section provides step-by-step configuration of WorkflowSim simulations to evaluate efficacy of adaptive data compression.

3.1 Installing WorkflowSim

- Download the latest WorkflowSim JAR file (version 1.0 as of writing) from: https://github.com/WorkflowSim/WorkflowSim-1.0
- More information about installation can be accessed on the same github link
- Create a Java project in your preferred IDE (Eclipse/IntelliJ/NetBeans)
- Add the downloaded workflowsim JAR file to project build path
  - In Eclipse: Build Path > Configure Build Path > Add external JARs
- Import WorkflowSim Java packages in your code: - import org.workflowsim.*

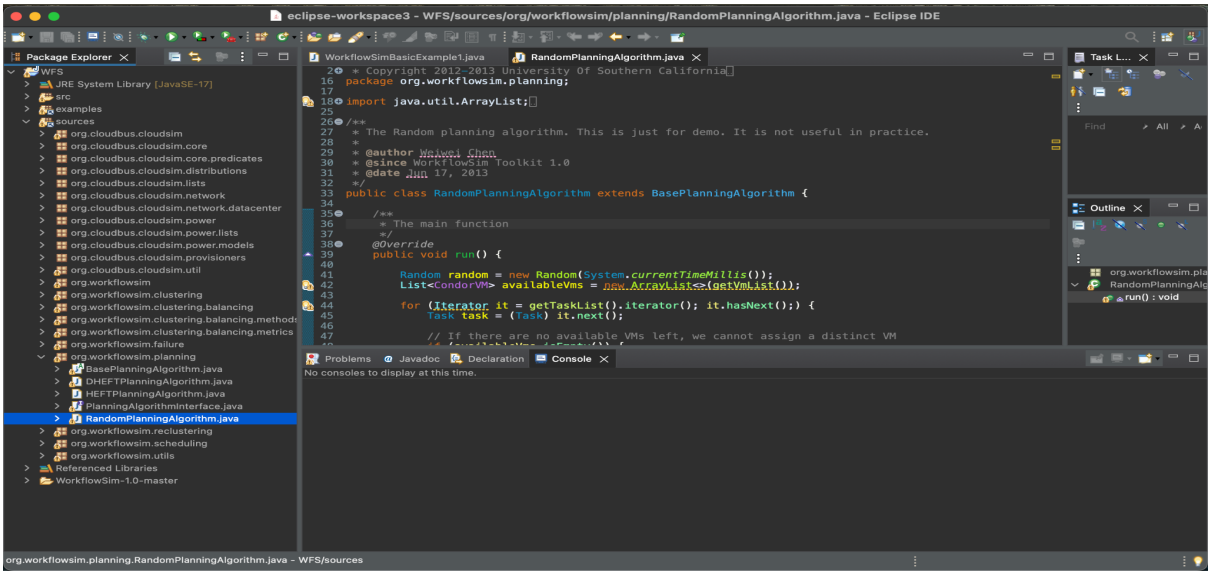
3.2 Implementing the Distinct Planning algorithm

The code for the DistinctPlanningAlgorithm is provided in the WorkflowSim code artefacts. To implement:

- Navigate to the RandomPlanningAlgorithm Java file in WorkflowSim package:

  Sources>org.workflosim.planning>RandomPlaningAlgorithm.java
- Replace the code in this file with the code from DistinctPlanningAlgorithm provided in ICT Solution Artefact Zip
- Build project

This will override the default random allocation with the new algorithm that allocates each task to a separate VM to enable data transfer modelling.



3.3 Configuring Simulation Scenarios

The main simulation code along with pre-configured DAX files for Compression On and Off scenarios are provided in the ICT Solution Artefact zip

To setup simulations:

- The code for the simulation is contained in WorkflowSimBasicExamlpe1.java provided within the ICT Solution Artefact ZIP
- Unzip the Montage and Inspiral DAX files for each scenario in a directory.
- Update the daxPath variable to point to correct DAX file path

The DAX files have pre-configured 'size' and 'runtime' attributes based on metrics gathered during preliminary experiments. This enables modelling compression enabled vs disabled scenarios. For example,
to setup simulation with Montage_25 workflow in ‘Compression Off’ setting the dax path variable will look like

```
*/
String daxPath = "/Users/apple/Desktop/SEMESTER 3/SEM 3 WORK/Workflows/10mb_off/10mb_off_Montage_25.xml";
```

and to setup simulation with the same workflow in ‘Compression On’ setting the dax path variable will be updated to point to the correct DAX XML

```
*/
String daxPath = "/Users/apple/Desktop/SEMESTER 3/SEM 3 WORK/Workflows/10mb_on/10mb_on_Montage_25.xml";
```

Only the daxPath variable needs to be updated to run the simulations. Rest of the code can be reused as is.
All the workflow DAX XML’s have been provided in the ICT Solutions Artefact zip for both ‘Compression On’ and ‘Compression Off’ settings.

- Define number of VMs and Host’s equals to number of tasks by making changes to the lines 98 and 194 within the WorkflowSimBasicExample1.java
- Run Simulations and collect metrics from simulation output to calculate Total Execution Time, Average Task Execution Time, Average task Start Time and Average Task Finish Time

```
159             List<CondorVM> vmlist0 = createVM(wfEngine.getSchedulerId(0), Parameters.getVmNum());
160
161             /**
162              * Submits this list of vms to this WorkflowEngine.
163              */
164             //Collections.shuffle(vmlist0);
```

Problems | Javadoc | Declaration | Console

<terminated> WorkflowSimBasicExample1 (2) [Java Application] /Users/apple/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_17.0.7.v20230425-1502/jre/bin/ja

11	12,	SUCCESS	2	46	22.08	18.74	40.82	2
6	7,	SUCCESS	2	2	22.16	18.74	40.9	2
5	6,	SUCCESS	2	27	22.16	18.74	40.9	2
12	13,	SUCCESS	2	47	22.19	18.74	40.93	2
7	8,	SUCCESS	2	23	22.45	18.74	41.19	2
14	15,	SUCCESS	2	32	7.65	41.19	48.84	3
15	16,	SUCCESS	2	48	6.06	48.84	54.9	4
16	17,	SUCCESS	2	17	17.38	54.9	72.28	5
17	18,	SUCCESS	2	40	17.63	54.9	72.53	5
20	21,	SUCCESS	2	34	17.75	54.9	72.65	5
18	19,	SUCCESS	2	44	17.82	54.9	72.72	5
19	20,	SUCCESS	2	0	17.92	54.9	72.82	5
21	22,	SUCCESS	2	38	26.64	72.82	99.46	6
22	23,	SUCCESS	2	11	7.73	99.46	107.19	7
23	24,	SUCCESS	2	39	8.5	107.19	115.69	8
24	25,	SUCCESS	2	28	2.8	115.69	118.49	9

Writable | Smart Insert | 102 : 99 : 3948

4. Real World Application Setup

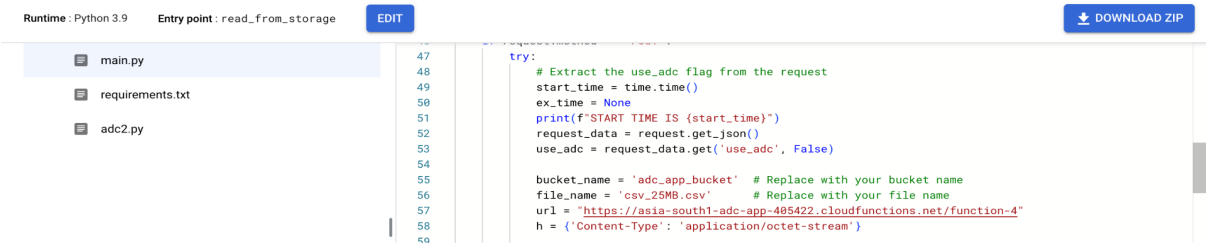
This section provides steps for deploying the 3-tier serverless application with integrated adaptive data compression library compression on GCP. The adaptive data compression library is included in the source zip for each function.

4.1 Setting up GCP Environment

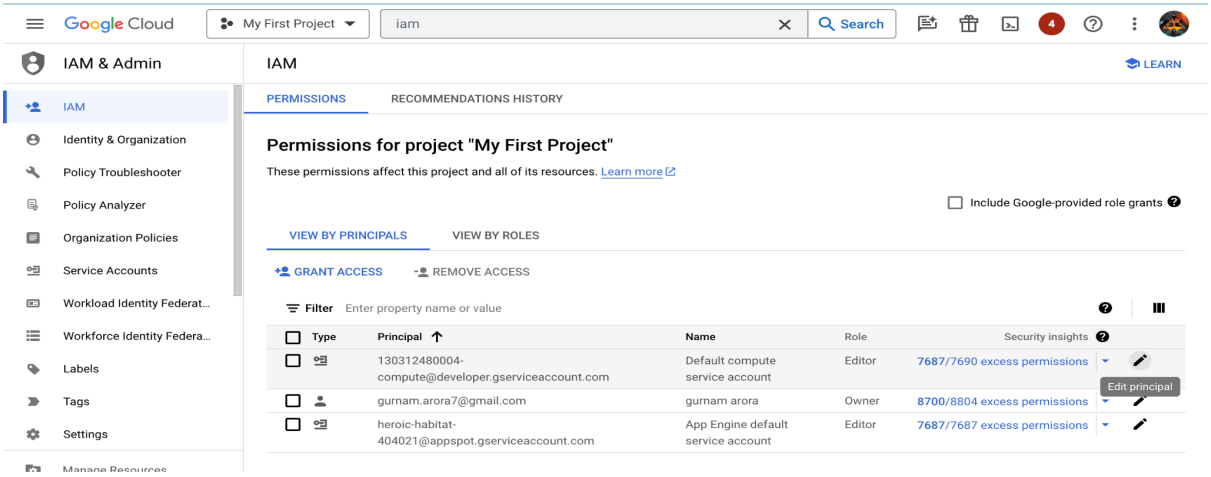
- Create a new project in GCP
- Repeat steps from Section 2 to setup cloud storage bucket, note down the name for this bucket.
- Upload the test payload provided in the ICT Solution Artefact zip into this storage bucket

4.2 Deploying Serverless Functions

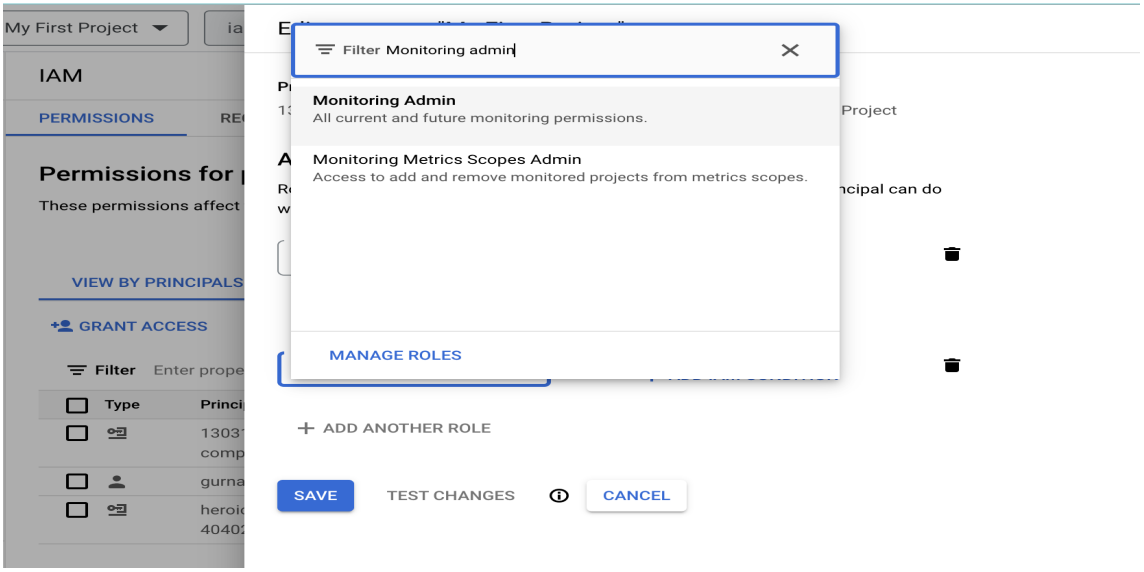
- Navigate to Cloud Functions in GCP console
- Create 3 separate HTTP triggered functions in a similar manner to Section 2.3
- Upload the function source zip (provided in ICT Solution Artefact Zip) for each function and assign new invoker binder policy (see section 2.3)
- Edit the Source Code for Function-1 (in main.py) to update the bucket name with test payload.



- To enable access to google cloud monitoring API (Stackdriver) we must assign the necessary permissions to our function's service account. To do this follow these steps
    - Go to IAM in google cloud console you will see the list of service accounts for your project



- Select the Default Compute Service account and click the pencil icon to edit roles for this account
- Click on Add Role; then Search for ‘Monitoring Admin’ and click save. This role gives the service account for our functions the necessary permission to access Google Cloud Monitoring API.



- Deploy functions

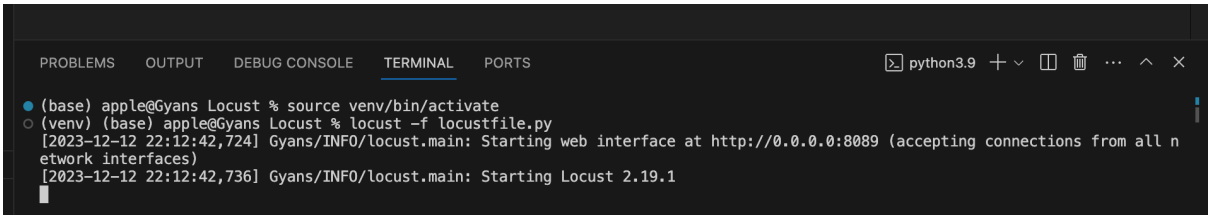
4.3 Configuring Load Generator

- Install Locust on your machine using command. It is recommended to create a virtual environment first.

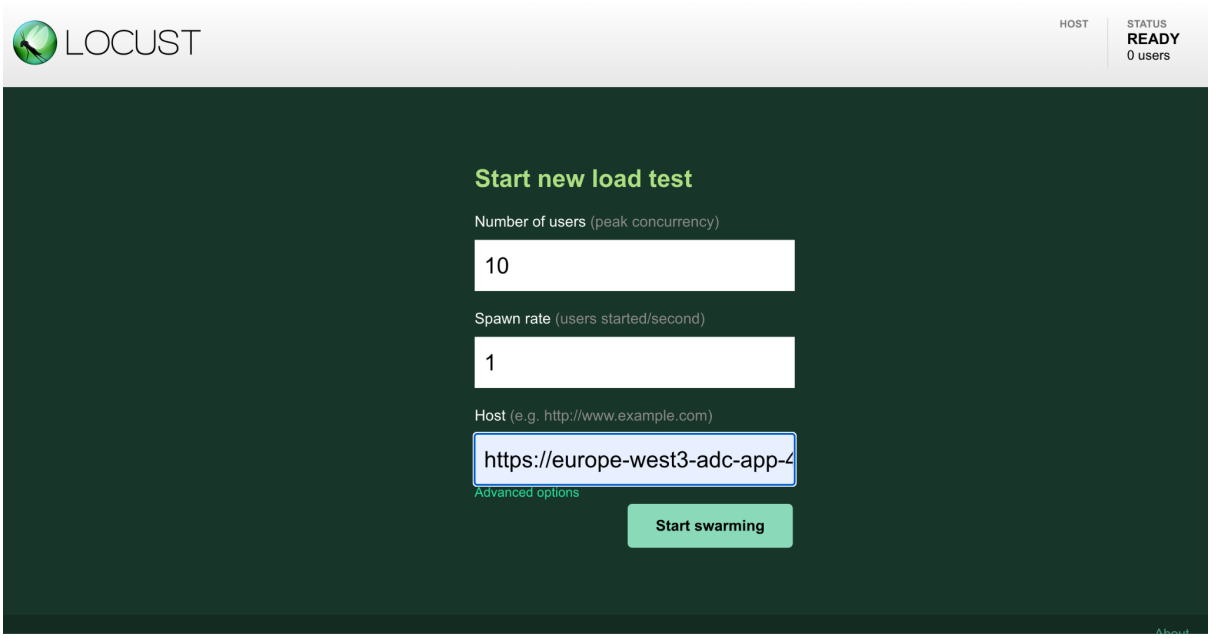
```
pip install locust
```

- Copy load test script 'locustfile.py' provided in code artefacts. To enable the ‘Compression On’ scenario, change the value of ‘use_adc’ parameter to ‘True’ at line:9. Similarly to toggle it off you can change the value of ‘use_adc’ to ‘False
- Start locust load test from terminal/command prompt

```
locust -f locustfile.py
```



- Access the Locust web UI to run load tests. A URL will be provided to access the locust web UI which can be used to change test configuration - users, spawn rate, duration etc.
- Enter the number of users, spawn rate and host (Function-1 Http trigger Url) to start the test. The ‘Compression On’ and ‘Compression Off’ scenarios can be set using the ‘use_adc’ variable in ‘locustfile.py’.



5. Additional Resources

This section contains useful libraries, tools and references utilised in the research experiments for further reading and troubleshooting.

Cloud Platform:

- Google Cloud Platform: https://cloud.google.com/docs
- Cloud Functions Documentation: https://cloud.google.com/functions/docs

Load Testing:

- Locust: https://locust.io/
- Locust documentation: https://docs.locust.io

Synthetic Payload Generation:

- Faker: https://faker.readthedocs.io/en/master/

References

De Prado, R. P. *et al.* (2014) "Processing astronomical image mosaic workflows with an expert broker in cloud computing," *Image Processing & Communications*, 19(4), pp. 5–20. doi: 10.1515/ipc-2015-0020.

Jain, A. and Kumari, R. (2017) "A review on comparison of workflow scheduling algorithms with scientific workflows," in *Advances in Intelligent Systems and Computing*. Singapore: Springer Singapore, pp. 613–622.

Jeong, G. *et al.* (2023) "Characterization of Data Compression in Datacenters," in *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE.