

Multi-Cloud Interoperability Report: AWS & GCP

Executive Summary

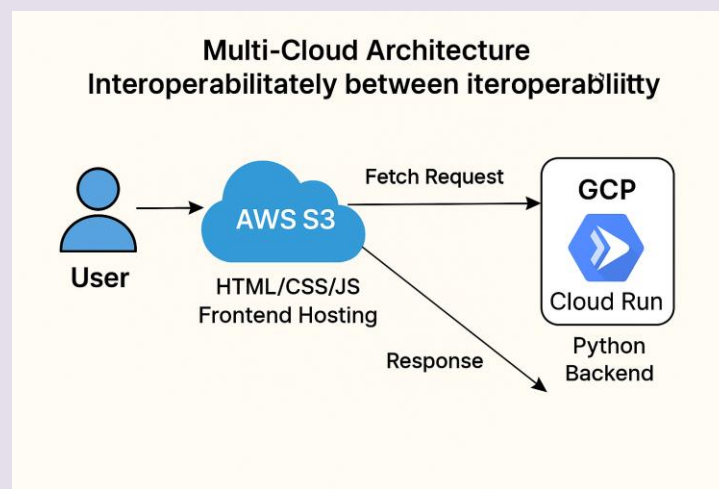
This report documents and demonstrates a practical implementation of multi-cloud interoperability using Amazon Web Services (AWS) and Google Cloud Platform (GCP). The project showcases a seamless interaction between a static frontend hosted on AWS S3 and a backend API deployed using GCP Cloud Run. This integration highlights the ability to leverage the strengths of different cloud platforms within a single application architecture.

Project Objective

The primary objective of this project is to showcase cross-platform cloud integration, allowing components of a web application to run on different cloud service providers. Specifically, the frontend is hosted on AWS S3, while the backend API is powered by GCP Cloud Run. The goal is to demonstrate how to achieve secure and functional communication between the two services.

Architecture Overview

The following architecture represents the layout of the project. A static frontend is served from an AWS S3 bucket with static website hosting enabled. This frontend includes JavaScript that makes a network request to an API endpoint hosted on GCP Cloud Run. The backend processes the request and returns a response, which is displayed on the frontend.



Technologies Used

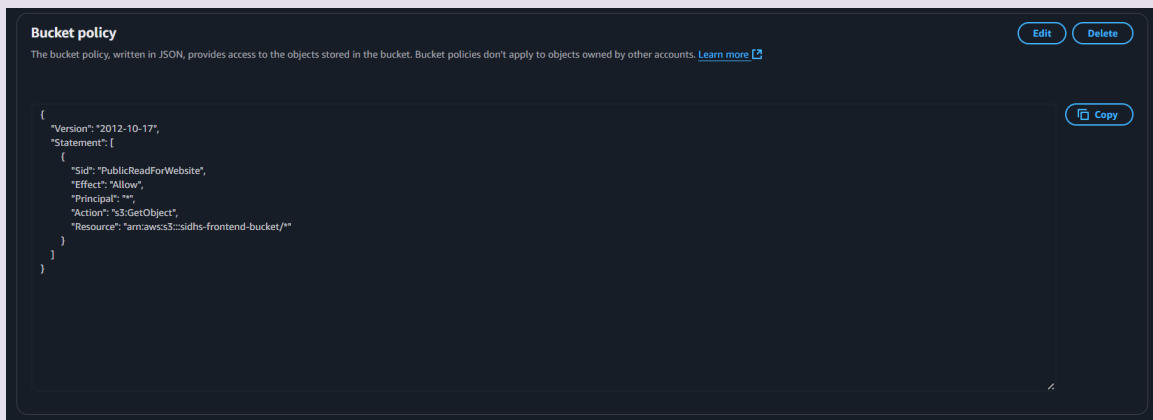
- AWS S3 - Static website hosting
- GCP Cloud Run - Python backend API
- HTML, CSS, JS - Frontend technologies
- CORS - For cross-domain communication
- Python Functions Framework

Implementation Details

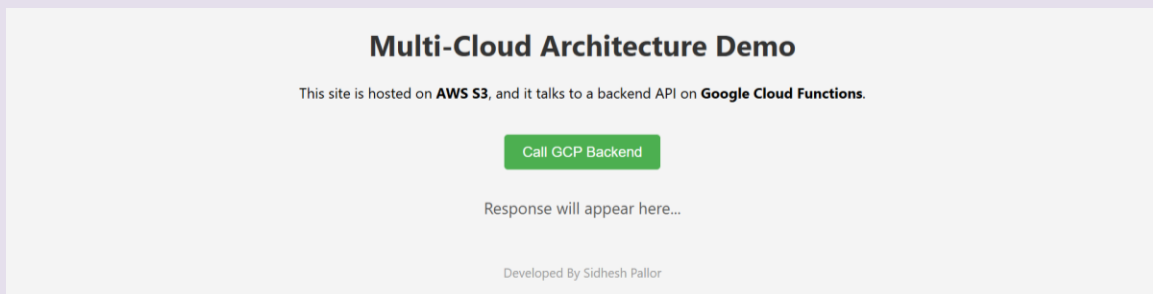
1. Frontend (AWS S3)

The frontend is a static HTML page hosted in an S3 bucket with static website hosting enabled. It contains a button that, when clicked, makes a fetch call to the backend hosted on GCP.

AWS S3 Bucket Settings:



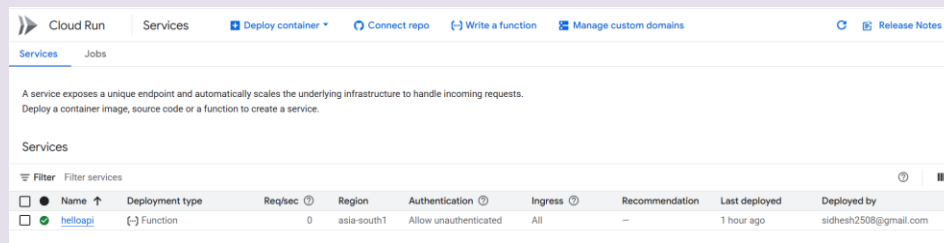
index.html Website in Browser:



2. Backend (GCP Cloud Run)

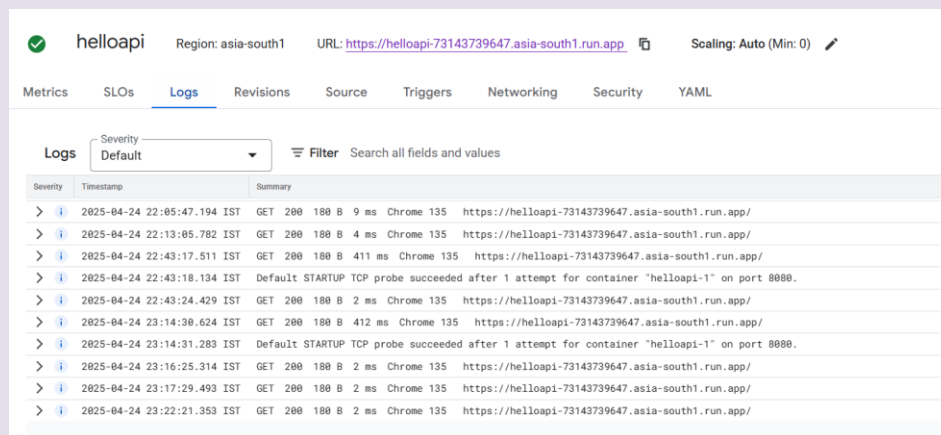
The backend is a Python function deployed on GCP Cloud Run. It includes CORS headers to handle requests from other origins. The function returns a simple message which the frontend displays to the user.

GCP Cloud Run Service Page:



Name	Deployment type	Req/sec	Region	Authentication	Ingress	Recommendation	Last deployed	Deployed by
helloapi	(-) Function	0	asia-south1	Allow unauthenticated	All	—	1 hour ago	sidhesh2508@gmail.com

Logs Showing API Trigger:



Severity	Timestamp	Summary
INFO	2025-04-24 22:05:47.194 IST	GET 200 180 B 9 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 22:13:05.782 IST	GET 200 180 B 4 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 22:43:17.511 IST	GET 200 180 B 411 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 22:43:18.134 IST	Default STARTUP TCP probe succeeded after 1 attempt for container 'helloapi-1' on port 8080.
INFO	2025-04-24 22:43:24.429 IST	GET 200 180 B 2 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 23:14:30.624 IST	GET 200 180 B 412 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 23:14:31.283 IST	Default STARTUP TCP probe succeeded after 1 attempt for container 'helloapi-1' on port 8080.
INFO	2025-04-24 23:16:25.314 IST	GET 200 180 B 2 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 23:17:29.493 IST	GET 200 180 B 2 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/
INFO	2025-04-24 23:22:21.353 IST	GET 200 180 B 2 ms Chrome 135 https://helloapi-73143739647.asia-south1.run.app/

Key Code Snippets

Backend: main.py

```
def hello_world(request):
    if request.method == 'OPTIONS':
        headers = {
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'GET, OPTIONS',
            'Access-Control-Allow-Headers': 'Content-Type',
        }
        return ('', 204, headers)

    headers = {
        'Access-Control-Allow-Origin': '*',
    }
    return ('Hello from GCP backend!', 200, headers)
```

Frontend: index.html (JavaScript Snippet)

```
function callBackend() {
  const output = document.getElementById("output");
  output.innerText = "Calling backend...";
  fetch("https://helloapi-73143739647.asia-south1.run.app")
    .then(response => {
      if (!response.ok) throw new Error("Network response was not ok");
      return response.text();
    })
    .then(data => {
      output.innerText = "Response from GCP: " + data;
    })
    .catch(error => {
      output.innerText = "Error: " + error.message;
    });
}
```

Live Demo Instructions

1. Open the hosted frontend URL from AWS S3 in a browser.
2. Click the 'Call GCP Backend' button.
3. The browser sends a fetch request to the GCP Cloud Run backend.
4. The backend responds with a message which is shown on the webpage.

Conclusion

This project successfully demonstrates how two major cloud platforms can interoperate to build a functioning web application. By combining AWS for frontend hosting and GCP for backend services.

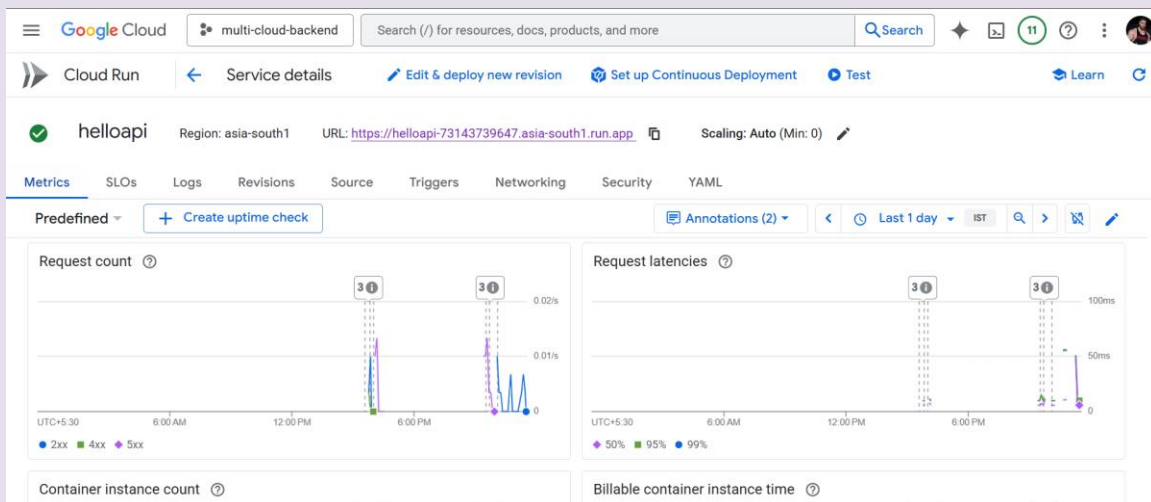
Appendix: Screenshots

AWS S3 BUCKET:

The screenshot shows the AWS S3 console interface for a bucket named 'sidhs-frontend-bucket'. The 'Objects' tab is selected, displaying a list of objects. There is one object named 'sindex.html' with a size of 1.8 KB and a storage class of 'Standard'. The object was last modified on April 24, 2025, at 23:16:08 (UTC+05:30). The console includes navigation tabs for Objects, Properties, Permissions, Metrics, Management, and Access Points. Action buttons like 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload' are visible. A search bar and a 'Show versions' toggle are also present.

Name	Type	Last modified	Size	Storage class
sindex.html	html	April 24, 2025, 23:16:08 (UTC+05:30)	1.8 KB	Standard

GCP Cloud Run Deployment View:



BROWSER SHOWING FRONTEND CALL RESULT:

The screenshot shows a web browser displaying the 'Multi-Cloud Architecture Demo'. The page features a rocket icon and the title 'Multi-Cloud Architecture Demo'. Below the title, it states 'This site is hosted on AWS S3, and it talks to a backend API on Google Cloud Functions.' A green button labeled 'Call GCP Backend' is present. Below the button, the response from the GCP backend is displayed: 'Response from GCP: Hello from GCP backend!'. At the bottom, it says 'Built with ❤️ using AWS & GCP'.

Multi-Cloud Architecture Demo

This site is hosted on **AWS S3**, and it talks to a backend API on **Google Cloud Functions**.

[Call GCP Backend](#)

Response from GCP: Hello from GCP backend!

Built with ❤️ using AWS & GCP