Virtualization and Cloud Computing (CSL7510)

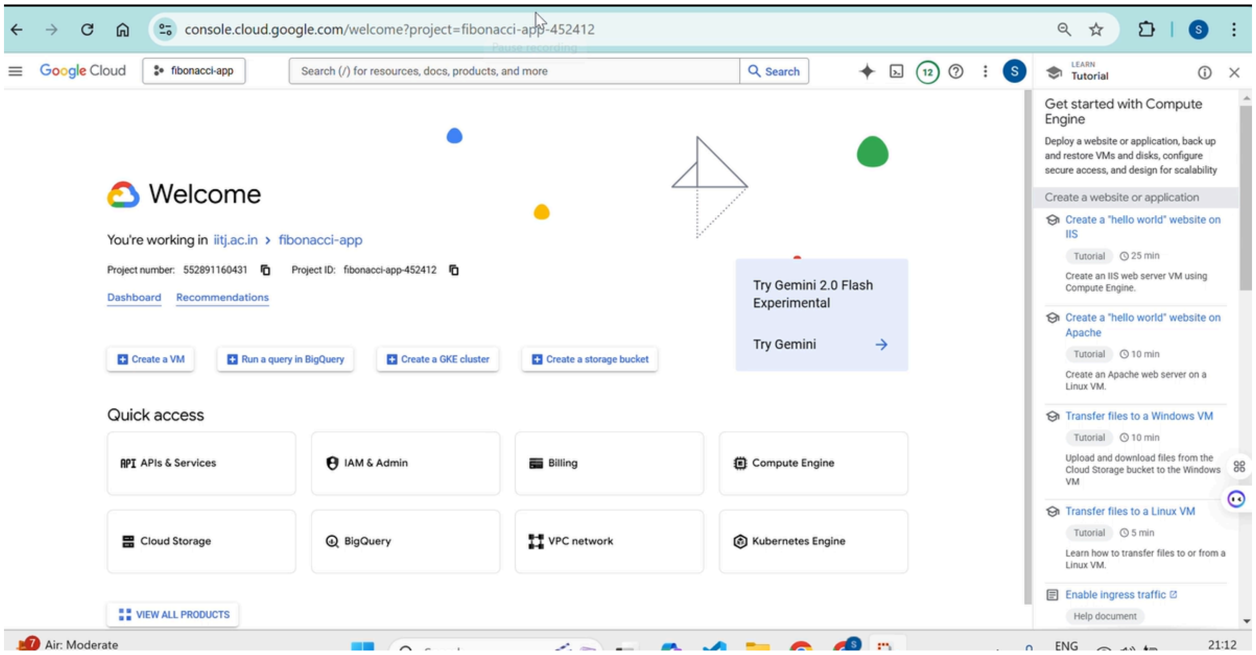
Assignment 2

Sireejaa Uppal (M23CSE023)

2nd March, 2025

Introduction

In the context of modern cloud computing, deploying scalable and secure applications is a critical requirement for ensuring optimal performance, cost-efficiency, and data protection. Google Cloud Platform (GCP) offers a robust suite of tools and services, including Compute Engine for virtual machine (VM) management, Managed Instance Groups (MIGs) for auto-scaling, and Identity and Access Management (IAM) for security. This document provides a step-by-step guide to **set up a virtual machine in GCP, implement auto-scaling policies based on workload, and configure security measures such as firewall rules and IAM roles**.
For this assignment, a **Fibonacci calculator application** has been developed using Python and Flask. This application calculates Fibonacci numbers and serves as a compute-intensive workload to demonstrate the implementation of auto-scaling and security configurations. By following this guide, you will learn how to create a scalable and secure infrastructure on GCP that dynamically adjusts to workload demands while maintaining strict access controls.
I started by logging into the GCP account and accumulating the credits. After that I created a new project as shown in the screenshot below:



console.cloud.google.com/welcome?project=fibonacci-app-452412
Google Cloud
fibonacci-app
Search (/) for resources, docs, products, and more
Search
Welcome
You're working in iitj.ac.in > fibonacci-app
Project number: 552891160431
Project ID: fibonacci-app-452412
Dashboard
Recommendations
Create a VM
Run a query in BigQuery
Create a GKE cluster
Create a storage bucket
Try Gemini 2.0 Flash Experimental
Try Gemini
Quick access
APIs & Services
IAM & Admin
Billing
Compute Engine
Cloud Storage
BigQuery
VPC network
Kubernetes Engine
VIEW ALL PRODUCTS
LEARN
Tutorial
Get started with Compute Engine
Deploy a website or application, back up and restore VMs and disks, configure secure access, and design for scalability
Create a website or application
Create a "hello world" website on IIS
Tutorial
25 min
Create an IIS web server VM using Compute Engine.
Create a "hello world" website on Apache
Tutorial
10 min
Create an Apache web server on a Linux VM.
Transfer files to a Windows VM
Tutorial
10 min
Upload and download files from the Cloud Storage bucket to the Windows VM
Transfer files to a Linux VM
Tutorial
5 min
Learn how to transfer files to or from a Linux VM.
Enable ingress traffic
Help document
Air: Moderate
ENG
21:12

Architecture Design

The architecture of this project on Google Cloud Platform (GCP) is designed to ensure scalability, security, and efficient resource utilization. It consists of four main components: Virtual Machine (VM) setup, Auto-Scaling Configuration, Security Policies, and Identity & Access Management (IAM). These components work together to provide a reliable and adaptable infrastructure for hosting the Fibonacci calculator application developed using Python and Flask.

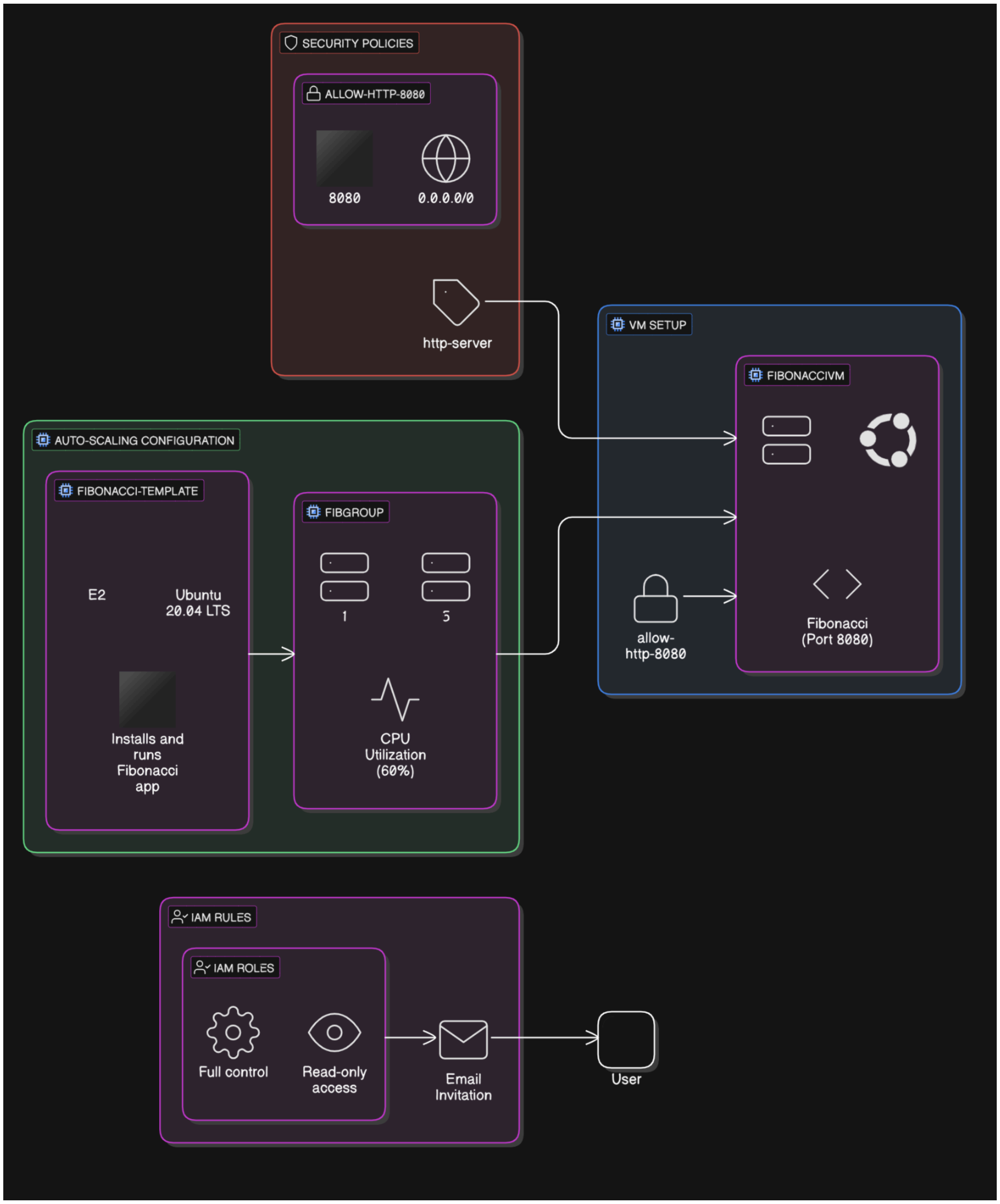
The VM setup begins with creating a Compute Engine instance named fibonaccivm in the us-central1-a zone. The VM is configured with an E2 machine type for cost efficiency and runs Ubuntu 20.04 LTS. The Fibonacci application is deployed on this instance, listening on port 8080. To allow external access, a firewall rule (allow-http-8080) is created to permit inbound HTTP traffic.

The auto-scaling configuration ensures that the system dynamically adjusts based on workload demand. An Instance Template (fibonacci-template) is created with the same VM settings. A Managed Instance Group (MIG) named fibgroup is set up using this template, enabling automatic scaling based on CPU utilization. The system is configured with a minimum of one instance and a maximum of five instances, scaling up when CPU usage exceeds 60% and scaling down when utilization drops. This allows the infrastructure to efficiently manage traffic spikes while optimizing costs.

The security policies are enforced through firewall rules and network tags to regulate inbound traffic. The allow-http-8080 rule ensures that only port 8080 is accessible, restricting unauthorized connections. Network tags (http-server) apply this rule specifically to instances running the Fibonacci application, ensuring controlled access while preventing unnecessary exposure of other services.

The IAM roles and access control settings are configured to manage permissions for different users. The Viewer role provides read-only access, allowing users to monitor resources without making changes. The Compute Admin role grants full control over Compute Engine resources, enabling administrative actions. Access to the project is granted via email invitations, ensuring that only authorized users can interact with the infrastructure.

This architecture ensures a scalable, secure, and cost-effective cloud deployment. The system dynamically adjusts to workload changes, enforces strict access controls, and maintains high availability while optimizing resources.

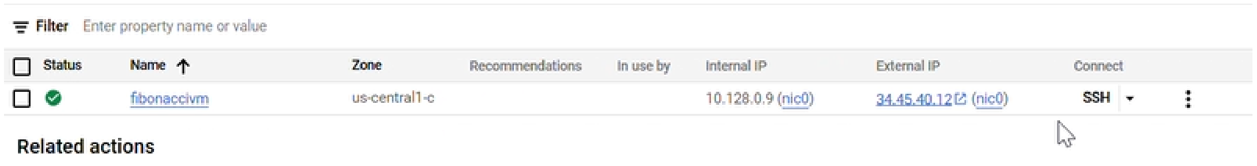


Create a VM Instance on GCP

This section outlines the steps to create a virtual machine instance on GCP. The VM will host the **Fibonacci calculator application**, which calculates Fibonacci numbers and returns the result along with the time taken for computation. Key steps include selecting the appropriate machine type, boot disk, and region/zone.

Step 1: Create the VM Instance

1. Log in to the Google Cloud Console and navigate to **Compute Engine** > **VM Instances**.
2. Click **Create Instance** to configure the VM.
3. Set the name of the VM to `fibonaccivm`.
4. Choose the region as **US Central (us-central1)** and select any available zone (e.g., `us-central1-a`). This region is chosen for its low latency and high availability.
5. Use the default **E2** machine type, which is cost-effective and suitable for general-purpose workloads like the Fibonacci application. The E2 series provides a balance of CPU, memory, and network performance.
6. Change the boot disk operating system from Debian to **Ubuntu 20.04 LTS**. Ubuntu is widely supported and ideal for Python-based applications due to its extensive package repositories and community support.
7. Under **Networking**, enable the option to **Allow HTTP traffic**. This automatically configures a firewall rule to allow incoming HTTP traffic on port 80. However, since the Fibonacci application runs on port 8080, additional firewall rules will be configured later.
8. Click **Create** to launch the VM. Once created, a green tick will appear next to the VM name, indicating it is running.
   Screenshot:



Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	fibonaccivm	us-central1-c			10.128.0.9 (nic0)	34.45.40.12 (nic0)	SSH

Step 2: Set Up the Fibonacci Application

1. **SSH into the VM**:
    - In the **VM Instances** list, click the **SSH** button next to `fibonaccivm` to open a terminal connection.
2. **Update and Install Dependencies**:
    - Run the following commands to update the system and install Python and Flask:

```
sudo apt update
sudo apt upgrade -y
sudo apt install python3 python3-pip -y
pip3 install flask
```

    - These commands ensure the VM has the latest security patches and software updates. Flask is installed as the web framework to run the Fibonacci application.

3. **Create and Run the Application**:
    - Create a file named `app.py` using the `nano` editor:

```
nano app.py
```

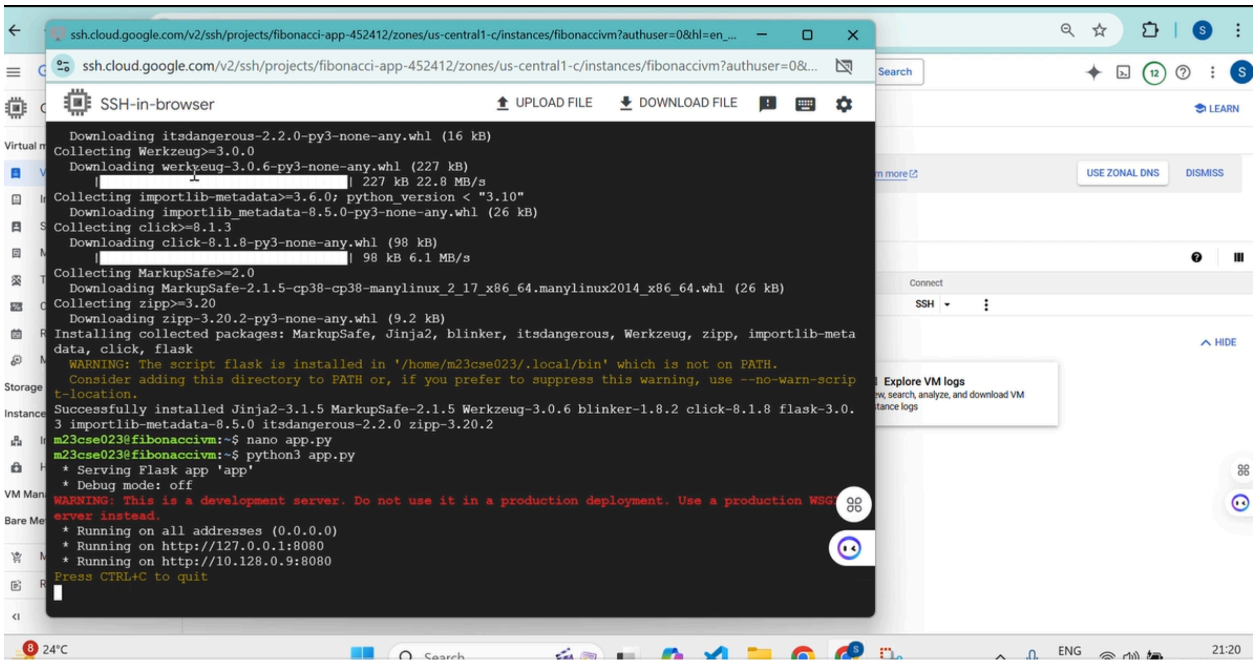
    - Add the following Python code to the file:

```
from flask import Flask, request
import time

app = Flask(__name__)

def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

@app.route("/fib")
def fib():
    n = int(request.args.get("n", 30))  # Default to calculating
fib(30)
    start_time = time.time()
    result = fibonacci(n)
    end_time = time.time()
    return f"Fibonacci({n}) = {result}, Time taken: {end_time -
start_time:.2f} seconds"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080)
```

- Save the file and exit the editor (`Ctrl + O`, `Enter`, `Ctrl + X`).
- Run the application using

```
python3 app.py
```

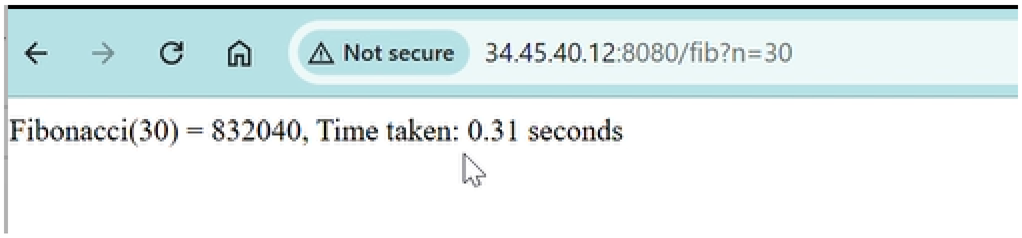
- The application will start and listen on port 8080.


ssh.cloud.google.com/v2/ssh/projects/fibonacci-app-452412/zones/us-central1-c/instances/fibonaccivm?authuser=0&hl=en_...
SSH-in-browser
UPLOAD FILE
DOWNLOAD FILE

```
  Downloading itsdangerous-2.2.0-py3-none-any.whl (16 kB)
Collecting Werkzeug>=3.0.0
  Downloading werkzeug-3.0.6-py3-none-any.whl (227 kB)
     |████████████████████████████████| 227 kB 22.8 MB/s
Collecting importlib-metadata>=3.6.0; python_version < "3.10"
  Downloading importlib_metadata-8.5.0-py3-none-any.whl (26 kB)
Collecting click>=8.1.3
  Downloading click-8.1.8-py3-none-any.whl (98 kB)
     |████████████████████████████████| 98 kB 6.1 MB/s
Collecting MarkupSafe>=2.0
  Downloading MarkupSafe-2.1.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26 kB)
Collecting zipp>=3.20
  Downloading zipp-3.20.2-py3-none-any.whl (9.2 kB)
Installing collected packages: MarkupSafe, Jinja2, blinker, itsdangerous, Werkzeug, zipp, importlib-meta
data, click, flask
  WARNING: The script flask is installed in '/home/m23cse023/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-scrip
t-location.
Successfully installed Jinja2-3.1.5 MarkupSafe-2.1.5 Werkzeug-3.0.6 blinker-1.8.2 click-8.1.8 flask-3.0.
3 importlib-metadata-8.5.0 itsdangerous-2.2.0 zipp-3.20.2
m23cse023@fibonaccivm:~$ nano app.py
m23cse023@fibonaccivm:~$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSG
erver instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://10.128.0.9:8080
Press CTRL+C to quit
```

```
http://34.45.40.12:8080/fib?n=30
```

4. **Test the Application**:


Not secure 34.45.40.12:8080/fib?n=30

Fibonacci(30) = 832040, Time taken: 0.31 seconds

Configure Auto-Scaling Policies

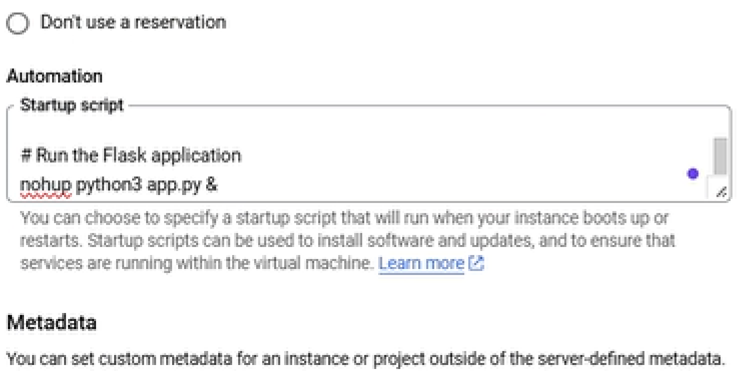
This section explains how to set up an **Instance Template** and **Managed Instance Group (MIG)** to enable auto-scaling based on CPU utilization. The Fibonacci application, being compute-intensive, is an ideal candidate for testing auto-scaling. When the application experiences high traffic, the MIG will automatically scale out by adding more VM instances to handle the increased load.

Step 1: Create an Instance Template

1. **Navigate to Instance Templates**:
   - Go to **Compute Engine** > **Instance Templates**.
   - Click **Create Instance Template**.
2. **Configure the Template**:
   - **Name**: Enter a name for the template (e.g., `fibonacci-template`).
   - **Location**: Choose the same region and zone as the VM created earlier (e.g., `us-central1-a`).
   - **Machine Configuration**: Use the **E2** machine type for cost-efficiency and scalability.
   - **Boot Disk**: Select **Ubuntu 20.04 LTS** as the operating system.
   - **Networking**: Enable **Allow HTTP traffic** to configure a firewall rule for HTTP traffic on port 80.
3. **Add Automation Script**:
   - Under the **Management** tab, scroll down to the **Automation** section.
   - Add the following startup script to automate the installation and deployment of the Fibonacci application:

```
#!/bin/bash
sudo apt update
sudo apt install -y python3 python3-pip
pip3 install flask
mkdir -p /home/myapp
cd /home/myapp
echo '
from flask import Flask, request
import time

app = Flask(__name__)

def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

@app.route("/fib")
def fib():
    n = int(request.args.get("n", 30))
    start_time = time.time()
    result = fibonacci(n)
    end_time = time.time()
    return f"Fibonacci({n}) = {result}, Time taken: {end_time - 
start_time:.2f} seconds"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080)
' > app.py
nohup python3 app.py &
```

- This script ensures that every VM created from this template automatically installs and runs the Fibonacci application.



Don't use a reservation

Automation

Startup script

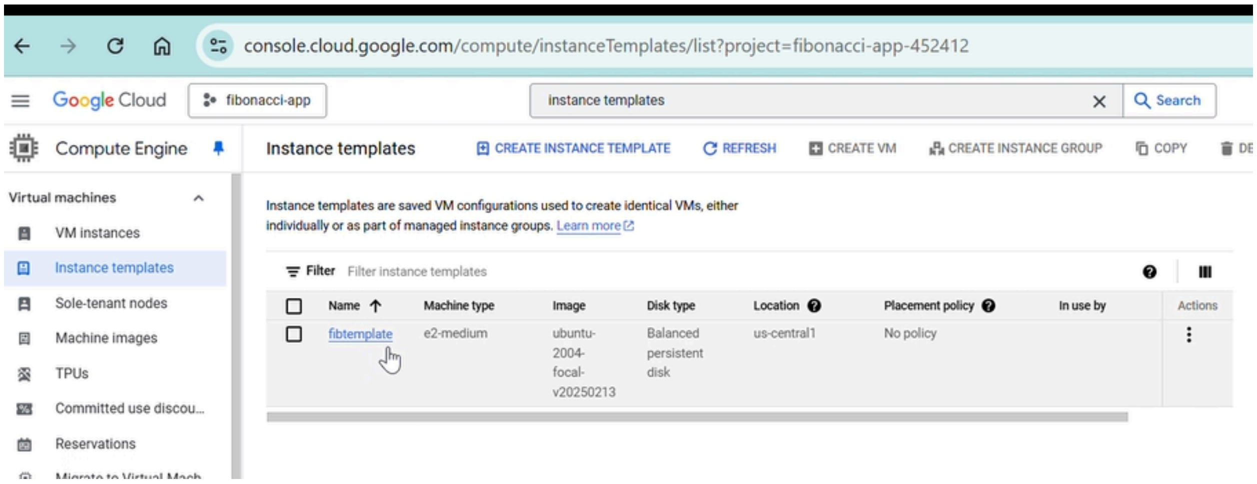
```
# Run the Flask application
nohup python3 app.py &
```

You can choose to specify a startup script that will run when your instance boots up or restarts. Startup scripts can be used to install software and updates, and to ensure that services are running within the virtual machine. Learn more

Metadata

You can set custom metadata for an instance or project outside of the server-defined metadata.

4. **Create the Template**:
   - Review the configuration and click **Create**.
   - The instance template is now ready to be used for creating VM instances.



console.cloud.google.com/compute/instanceTemplates/list?project=fibonacci-app-452412

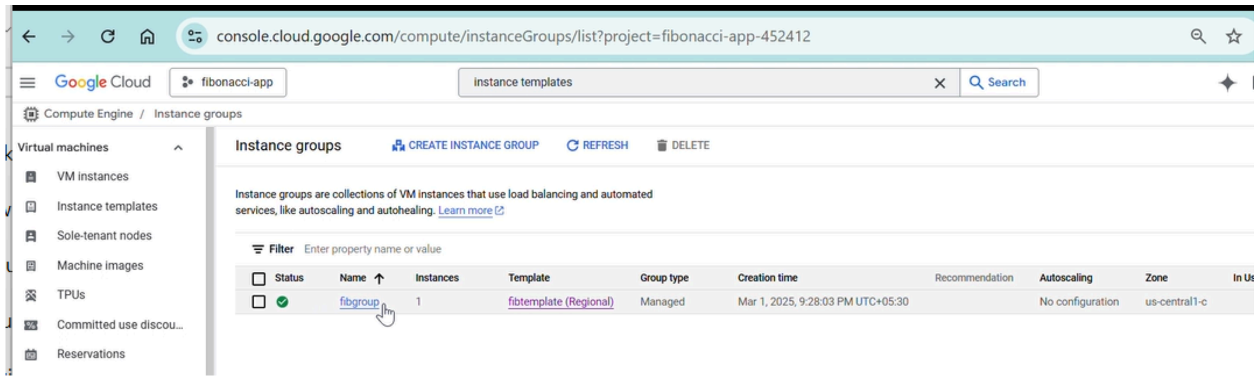
Instance templates are saved VM configurations used to create identical VMs, either individually or as part of managed instance groups. Learn more

Name	Machine type	Image	Disk type	Location	Placement policy	In use by	Actions
fibtemplate	e2-medium	ubuntu-2004-focal-v20250213	Balanced persistent disk	us-central1	No policy		⋮

Step 2: Create a Managed Instance Group (MIG)

1. **Navigate to Instance Groups**:
   - Go to **Compute Engine** > **Instance Groups**.
   - Click **Create Instance Group**.
2. **Configure the MIG**:
   - **Name**: Enter a name for the group (e.g., `fibgroup`).
   - **Location**: Select **Single Zone** and choose the same zone as the instance template (e.g., `us-central1-a`).
   - **Instance Template**: Select the `fibonacci-template` created in the previous step.

3. **Enable Auto-Scaling**:
    - Set the **Autoscaling Mode** to **On: Add or remove instances to the group**.
    - **Minimum Number of Instances**: Set to `1`.
    - **Maximum Number of Instances**: Set to `5`.
    - **Autoscaling Metric**: Select **CPU Utilization**.
    - **Target CPU Utilization**: Set to `60%`. This means the MIG will add more instances when CPU utilization exceeds 60% and remove instances when it falls below this threshold.
4. **Create the MIG**:
    - Review the configuration and click **Create**.
    - The MIG is now created and will automatically manage VM instances based on CPU utilization.

1. 

console.cloud.google.com/compute/instanceGroups/list?project=fibonacci-app-452412

Compute Engine / Instance groups

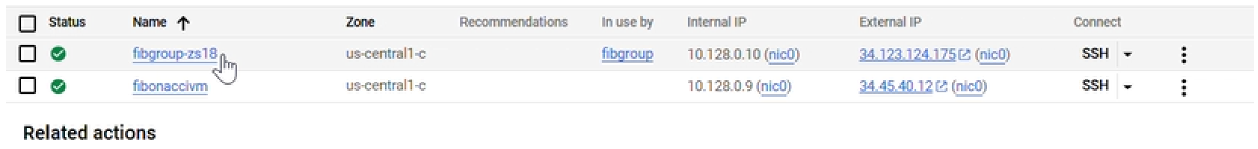
Instance groups

Instance groups are collections of VM instances that use load balancing and automated services, like autoscaling and autohealing. Learn more

Status	Name	Instances	Template	Group type	Creation time	Recommendation	Autoscaling	Zone	In us
✓	fibgroup	1	fibtemplate (Regional)	Managed	Mar 1, 2025, 9:28:03 PM UTC+05:30		No configuration	us-central1-c	

Step 3: Test Auto-Scaling

1. **Initial State**:
    - After creating the MIG, it starts with one VM instance, as shown in the screenshot below.

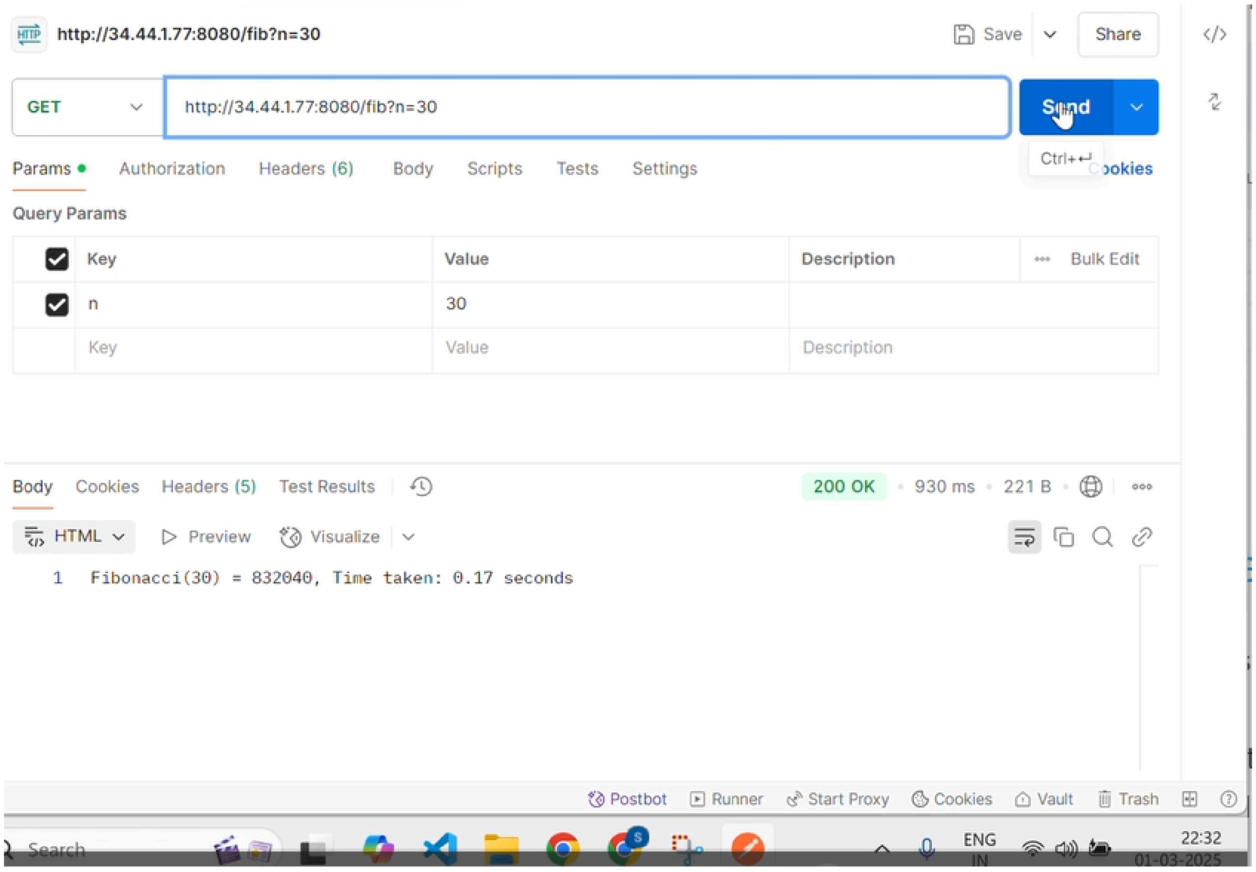


Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	fibgroup-zs18	us-central1-c		fibgroup	10.128.0.10 (nic0)	34.123.124.175 (nic0)	SSH
✓	fibonaccivm	us-central1-c			10.128.0.9 (nic0)	34.45.40.12 (nic0)	SSH

The MIG initially creates one VM instance.

2. **Increase Load**:
    - To simulate high CPU load, fire repeated API requests to the Fibonacci application using **Postman** or a load-testing tool.
    - Example Postman configuration:
        - Set the request method to **GET**.

- Use the URL:http://34.45.40.12:8080/fib?n=30
- Send multiple requests concurrently to increase CPU utilization.



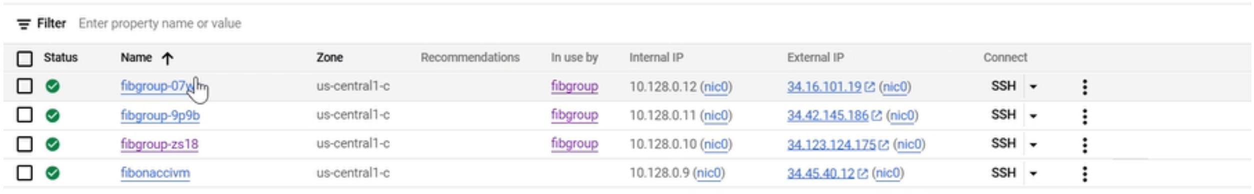
GET http://34.44.1.77:8080/fib?n=30

Key	Value	Description
n	30	

```
1  Fibonacci(30) = 832040, Time taken: 0.17 seconds
```

Postman is used to send repeated API requests to increase CPU load.

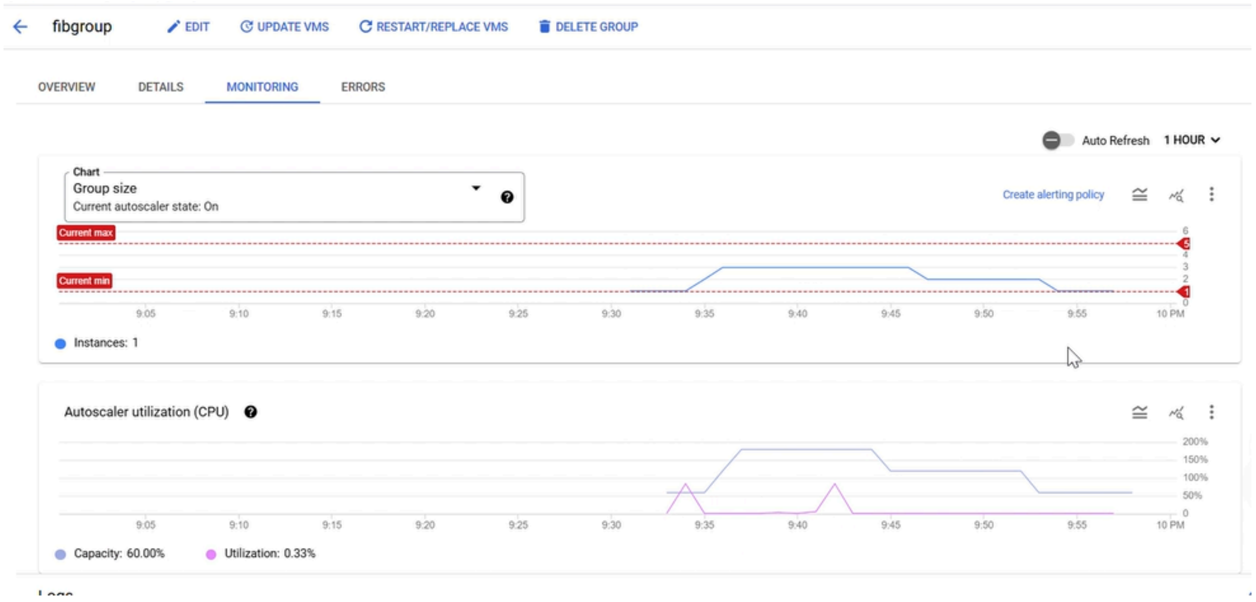
3. **Observe Auto-Scaling**:
   - As the CPU utilization exceeds the 60% threshold, the MIG automatically adds more VM instances to handle the increased load.
   - The **Monitoring** tab in the MIG dashboard shows the CPU utilization curve and the number of instances added over time.



Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
✓	fibgroup-07w	us-central1-c		fibgroup	10.128.0.12 (nic0)	34.16.101.19 (nic0)	SSH
✓	fibgroup-9p9b	us-central1-c		fibgroup	10.128.0.11 (nic0)	34.42.145.186 (nic0)	SSH
✓	fibgroup-zs18	us-central1-c		fibgroup	10.128.0.10 (nic0)	34.123.124.175 (nic0)	SSH
✓	fibonaccivm	us-central1-c			10.128.0.9 (nic0)	34.45.40.12 (nic0)	SSH

The MIG adds more VM instances as CPU utilization increases.

4. **Reduce Load**:
   - Stop the load test and observe the MIG removing instances as CPU utilization drops below the threshold.



Configure Security Measures

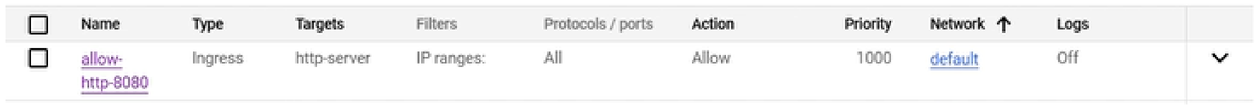
This section covers the implementation of **firewall rules** to control inbound and outbound traffic. For the Fibonacci application, firewall rules are configured to allow HTTP traffic on port 8080 while blocking unauthorized access. This ensures that the application remains secure and accessible only to authorized users.

Step 1: Allow HTTP Traffic

1. **Enable HTTP Traffic During VM Creation**:
   - When creating the VM instance (`fibonaccivm`), the **Allow HTTP traffic** checkbox was selected. This automatically configures a firewall rule to allow incoming HTTP traffic on port 80.
   - However, since the Fibonacci application runs on port 8080, additional firewall rules are required.

Step 2: Create a Custom Firewall Rule for Port 8080

1. **Navigate to Firewall Rules**:
   - Go to **VPC Network** > **Firewall** in the GCP Console.
   - Click **Create Firewall Rule**.
2. **Configure the Firewall Rule**:
   - **Name**: Enter a name for the rule (e.g., `allow-http-8080`).
   - **Target Tags**: Add a tag (e.g., `http-server`) to apply this rule to specific VM instances.
   - **Source IP Ranges**: Set to `0.0.0.0/0` to allow traffic from all IP addresses. For production environments, restrict this to specific IP ranges for enhanced security.
   - **Protocols and Ports**: Select **TCP** and specify port `8080` to allow traffic for the Fibonacci application.
3. **Create the Rule**:
   - Review the configuration and click **Create**.
   - The new firewall rule will appear in the list of firewall rules.

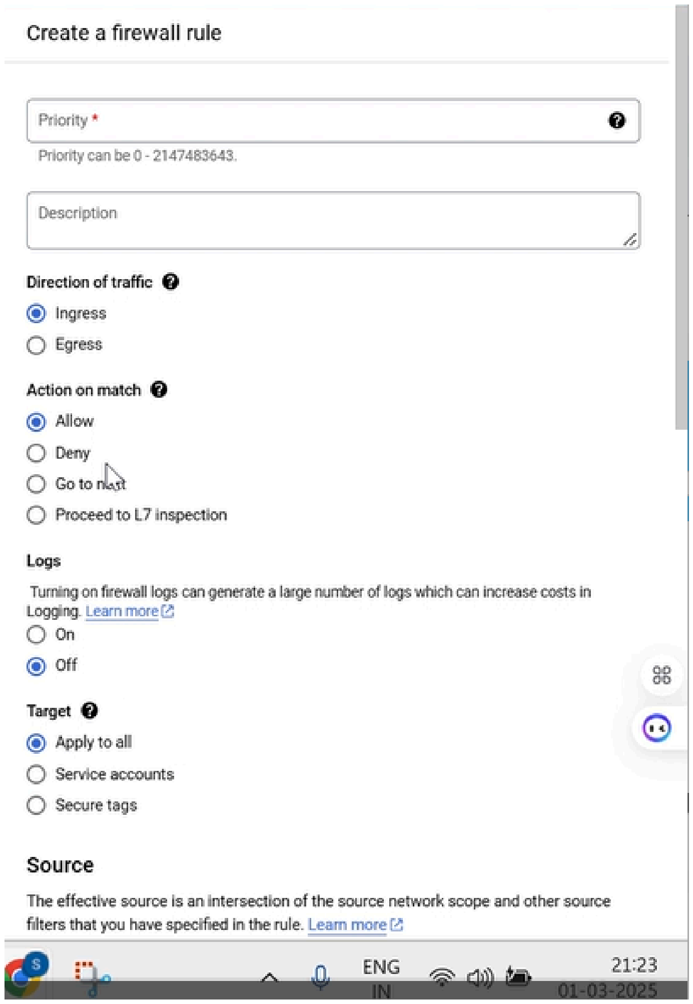


☐	Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network ↑	Logs
☐	allow-http-8080	Ingress	http-server	IP ranges:	All	Allow	1000	default	Off

The firewall rule `allow-http-8080` is successfully created.

Step 3: Apply the Firewall Rule to VM Instances

1. **Tag VM Instances**:
   - Go to **Compute Engine** > **VM Instances**.
   - Select the VM instance (`fibonaccivm`) and click **Edit**.
   - Under **Network Tags**, add the tag `http-server`.
   - Click **Save**.


Create a firewall rule

Priority *

Priority can be 0 - 2147483643.

Description

Direction of traffic

[✓ Ingress] Egress

Action on match

[✓ Allow] Deny Go to next Proceed to L7 inspection

Logs

Turning on firewall logs can generate a large number of logs which can increase costs in Logging. Learn more

On [✓ Off]

Target

[✓ Apply to all] Service accounts Secure tags

Source

The effective source is an intersection of the source network scope and other source filters that you have specified in the rule. Learn more

ENG IN 21:23 01-03-2025

2. **Verify the Firewall Rule**:
   - The firewall rule `allow-http-8080` is now applied to the VM instance, allowing incoming traffic on port 8080.

Step 4: Test the Firewall Rule

1. **Access the Fibonacci Application**:
   - Open a web browser and navigate to:
   - You should see the Fibonacci application running successfully.
2. **Block Unauthorized Traffic**:
   - Attempt to access the application on a port other than 8080 (e.g., port 80). The request should be blocked, demonstrating that the firewall rule is working as intended.

Configure IAM Rules

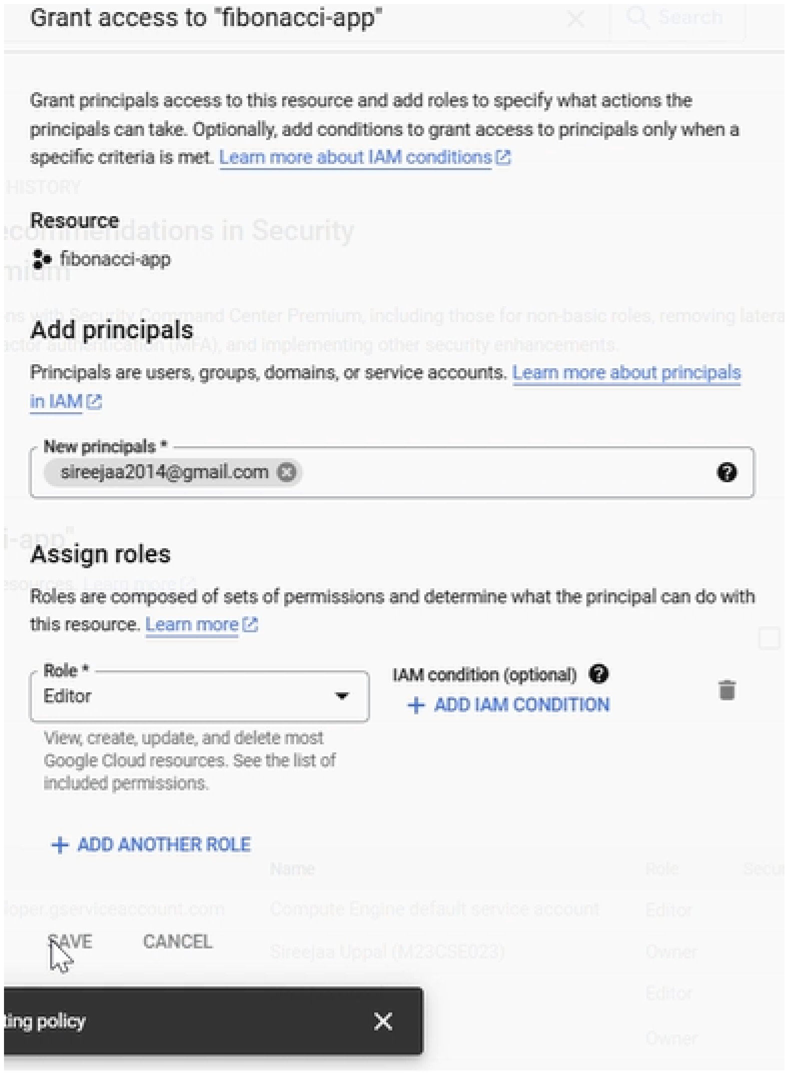
This section details the setup of **IAM roles** to manage access to GCP resources. For example, the **Viewer role** can be assigned to users who only need read-only access, while the **Compute Admin role** can be assigned to users who require full control over VM instances. This ensures that only authorized users can perform specific actions on the resources.
Identity and Access Management (IAM) in GCP allows you to control who has access to your resources and what actions they can perform. In this section, we will configure IAM roles to grant restricted access to users for the Fibonacci application project.

Step 1: Navigate to IAM & Admin

1. **Go to IAM & Admin**:
    - In the GCP Console, navigate to **IAM & Admin** > **IAM**.
2. **Grant Access**:
    - Click the **Grant Access** button to add a new member and assign roles.

Step 2: Add a Member and Assign Roles

1. **Enter Email ID**:
    - In the **New Members** field, enter the email address of the user you want to grant access to.
2. **Assign Roles**:
    - Select the appropriate role from the dropdown menu. For example:
        - **Viewer**: Grants read-only access to the project.
        - **Compute Admin**: Grants full control over Compute Engine resources.
        - **Editor**: Grants edit access to the project but not administrative privileges.
    - For this assignment, assign the **Viewer** role to provide read-only access.
3. **Save the Configuration**:
    - Click **Save** to apply the changes.


Grant access to "fibonacci-app"

Grant principals access to this resource and add roles to specify what actions the principals can take. Optionally, add conditions to grant access to principals only when a specific criteria is met. Learn more about IAM conditions

Resource

fibonacci-app

Add principals

Principals are users, groups, domains, or service accounts. Learn more about principals in IAM

New principals *: sireejaa2014@gmail.com

Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. Learn more

Role *: Editor

View, create, update, and delete most Google Cloud resources. See the list of included permissions.

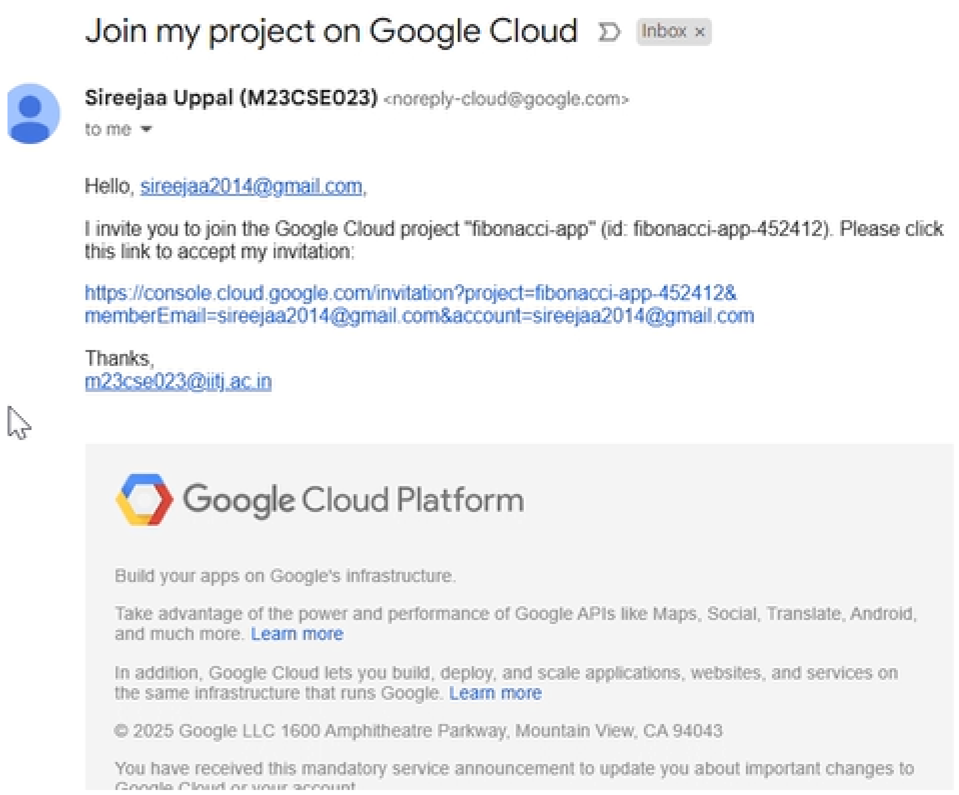
IAM condition (optional)

Assigning the Viewer role to a user.

Step 3: Notification and Acceptance

1. **Notification Email**:
   - The user will receive a notification email with an invitation to access the project.
   - The email will include details about the assigned role and a link to accept the invitation.
2. **Accept the Invitation**:
   - The user must click the link in the email and accept the invitation.

- Once accepted, the user will have access to the project with the assigned role.



Join my project on Google Cloud Inbox ×

Sireejaa Uppal (M23CSE023) <noreply-cloud@google.com>
to me

Hello, sireejaa2014@gmail.com,

I invite you to join the Google Cloud project "fibonacci-app" (id: fibonacci-app-452412). Please click this link to accept my invitation:

https://console.cloud.google.com/invitation?project=fibonacci-app-452412&memberEmail=sireejaa2014@gmail.com&account=sireejaa2014@gmail.com

Thanks,
m23cse023@iitj.ac.in

Google Cloud Platform

Build your apps on Google's infrastructure.

Take advantage of the power and performance of Google APIs like Maps, Social, Translate, Android, and much more. Learn more

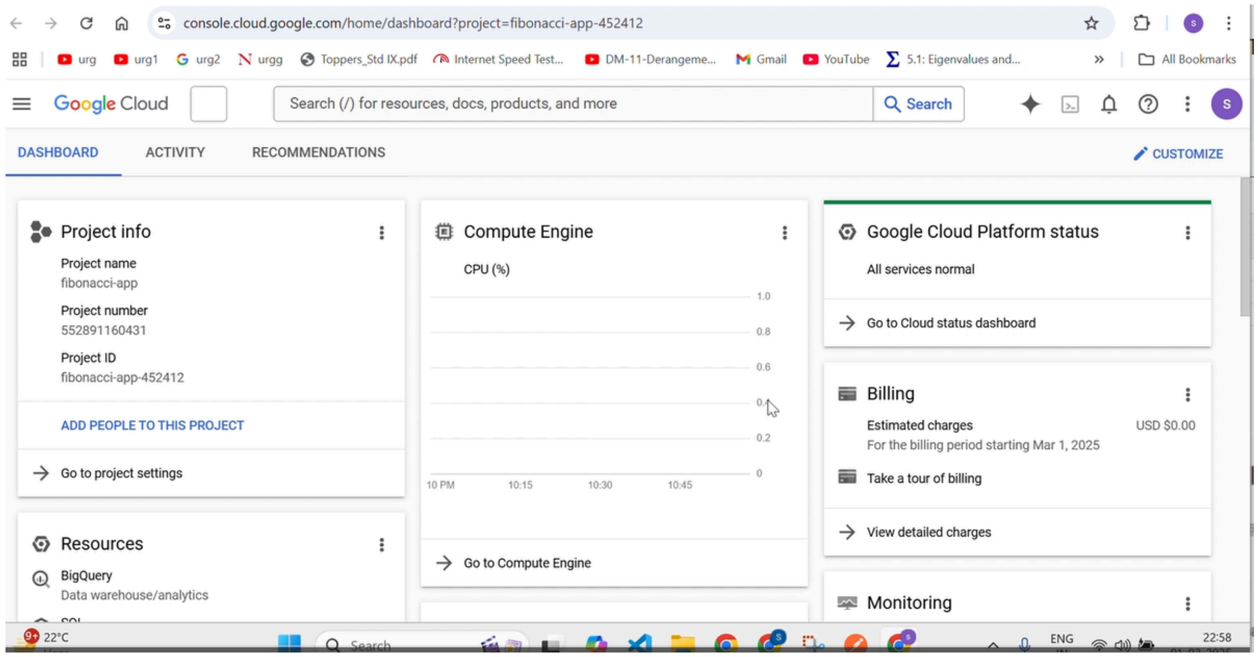
In addition, Google Cloud lets you build, deploy, and scale applications, websites, and services on the same infrastructure that runs Google. Learn more

© 2025 Google LLC 1600 Amphitheatre Parkway, Mountain View, CA 94043

You have received this mandatory service announcement to update you about important changes to Google Cloud or your account.

The notification email sent to the user with the invitation to access the project.

3. **Access the Project**:
   - After accepting the invitation, the user can log in to the GCP Console and access the project with the assigned role (e.g., Viewer).



Project info

Project name
fibonacci-app

Project number
552891160431

Project ID
fibonacci-app-452412

ADD PEOPLE TO THIS PROJECT

Go to project settings

Resources

BigQuery
Data warehouse/analytics

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Billing

Estimated charges USD $0.00
For the billing period starting Mar 1, 2025

Take a tour of billing

View detailed charges

The user accesses the project with the Viewer role.

Technical Terms and Details

- **Port 8080**: The Fibonacci application runs on port 8080, so the firewall rule is configured to allow traffic on this port.
- **Source IP Ranges**: Setting the source IP range to `0.0.0.0/0` allows traffic from any IP address. In production, this should be restricted to specific IP ranges for security.
- **Network Tags**: Tags are used to apply firewall rules to specific VM instances. The `http-server` tag ensures that only instances with this tag are affected by the rule.
- **Instance Template**: The template defines the configuration for VM instances, including the machine type, boot disk, and startup script. It ensures consistency across all instances in the MIG.
- **Managed Instance Group (MIG)**: The MIG automatically manages VM instances based on workload, ensuring high availability and scalability.

- **Auto-Scaling Metrics**: CPU utilization is used as the primary metric for scaling. Other metrics, such as network traffic or custom metrics, can also be configured.
- **Postman Load Testing**: Postman is used to simulate high traffic and test the auto-scaling behavior of the MIG.
- **E2 Machine Type**: The E2 series is designed for cost-effective general-purpose workloads. It provides a balance of CPU, memory, and network performance, making it ideal for hosting the Fibonacci application.
- **Ubuntu OS**: Ubuntu 20.04 LTS is chosen for its stability, long-term support, and compatibility with Python and Flask.
- **Firewall Rules**: Enabling **Allow HTTP traffic** during VM creation configures a firewall rule for port 80. However, since the Fibonacci application runs on port 8080, additional firewall rules will be configured in the security section.
- **IAM Roles**: IAM roles define what actions a user can perform on GCP resources. Roles can be predefined (e.g., Viewer, Editor, Compute Admin) or custom.
- **Viewer Role**: The Viewer role provides read-only access, allowing users to view resources but not modify them.
- **Compute Admin Role**: The Compute Admin role grants full control over Compute Engine resources, including creating, modifying, and deleting VM instances.
- **Email Invitation**: When a user is granted access, they receive an email invitation. They must accept the invitation to access the project.

Links

Github : https://github.com/Siree16/VCC_Assignment2/

Video : vcc2.mp4

Report : Assignment2_VCC

Plagiarism Clause:: I declare that this project is my own work, and any external resources have been properly credited. I understand that plagiarism will lead to disqualification from the assignment and possible further disciplinary actions.