Auto-Scaling of a Local VM to GCP When Resource Usage Exceeds 75%

Objective

This project focuses on creating a local Virtual Machine (VM) and implementing a resource usage monitoring system. When resource usage surpasses 75%, the system automatically scales to the Google Cloud Platform (GCP) through the GCP CLI. It includes setting up a local VM, configuring monitoring tools, applying auto-scaling policies, and deploying a sample application.

- Set up a local VM using VirtualBox or VMware.
- Implement resource monitoring using Prometheus, Grafana, or a custom script.
- Configure auto-scaling using the GCP CLI when CPU or memory usage exceeds 75%.
- Deploy a sample application to validate the scalability process.

Introduction

As the demand for computing resources grows, businesses and developers require flexible solutions that can scale according to real-time needs. This project tackles this issue by introducing an auto-scaling mechanism that automatically moves workloads from a local VM to the cloud once resource usage surpasses a set threshold.

System Requirements:

- Host System: Windows with sufficient RAM (8GB+ recommended) with a minimum hard disk space of 40 GB.

- Virtual Machines: Windows 10 (two instances) iso file(any OS can be used, e.g – Ubuntu-Linux).
- Software Requirements:
- Oracle VirtualBox (for VM management)
- VirtualBox Extension Pack (for enhanced networking)
- Node.js (for building the API)
- MongoDB (for database storage)
- Git (optional for version control)
- Postman (for API testing)
- Windows PowerShell/Command Prompt (for command execution)

System Architecture

Workflow

1. **Create a Local VM**: Using VirtualBox or VMware.
2. **Resource Monitoring**: Tools track CPU and memory usage.
3. **Auto-Scaling Trigger**: When resource usage exceeds 75%, a custom script triggers the creation of a GCP instance using the GCP CLI.
4. **Cloud Deployment**: The sample application is deployed to the cloud instance.

Implementation

1. **Local VM Setup**

   Tools Required:

   - VirtualBox or VMware

- Windows-based OS image
- Python and shell scripting capabilities

Steps:

- **Install VirtualBox/VMware**: Download and install the preferred hypervisor.
- **Install Ubuntu OS Image**: Create a virtual machine and install an Ubuntu-based OS.
- **Networking Configuration**: Ensure that the VM has internet access via NAT or Bridged networking.
- **Install Required Packages**:

Virtual Machine Setup

Step 1: Install VirtualBox

1. Download VirtualBox from VirtualBox Official Website and install it.
2. Download and install the VirtualBox Extension Pack to enable additional networking features.
3. Restart your computer if prompted after installation.

Step 2: Create Virtual Machines

1. Open VirtualBox and click New to create a new VM.
2. Enter a name for the VM (e.g., VM1-API for the API server, VM2-DB for the database server).
3. Select Windows 10 (64-bit) as the operating system.
4. Allocate at least 2GB RAM per VM (4GB recommended for better performance).
5. Create a Virtual Hard Disk (at least 10GB, dynamically allocated).
6. Install Windows 10 on the VM using an ISO file.

7. Repeat the process for the second VM.

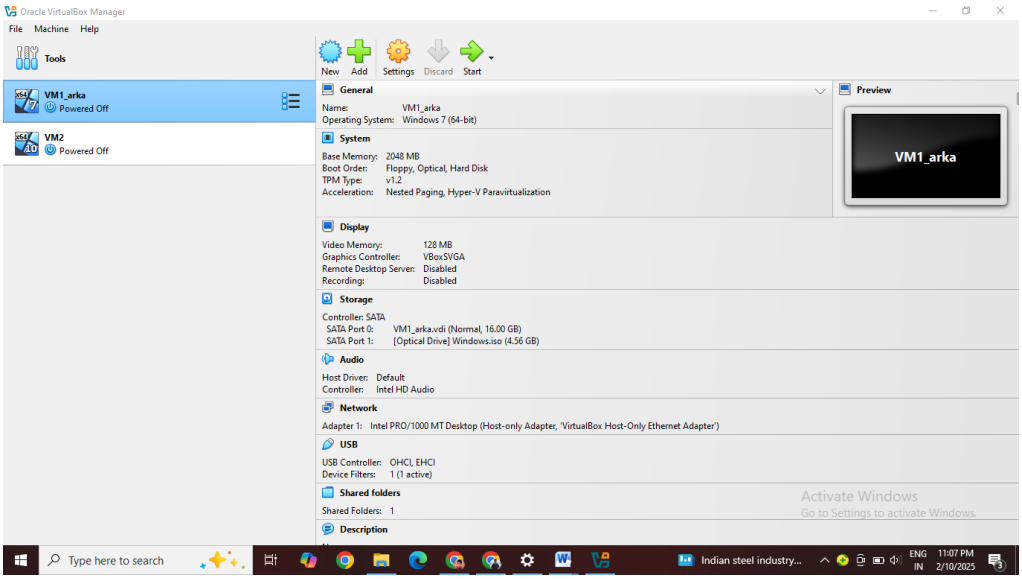


Figure 1: VM1 creation

Resource Monitoring Implementation

Using Prometheus and Grafana:

- Install and configure Prometheus to monitor CPU and memory usage by editing the `prometheus.yml` file.
- Start Prometheus with the command:
- `prometheus --config.file=prometheus.yml`
- Install and configure Grafana to visualize the metrics.

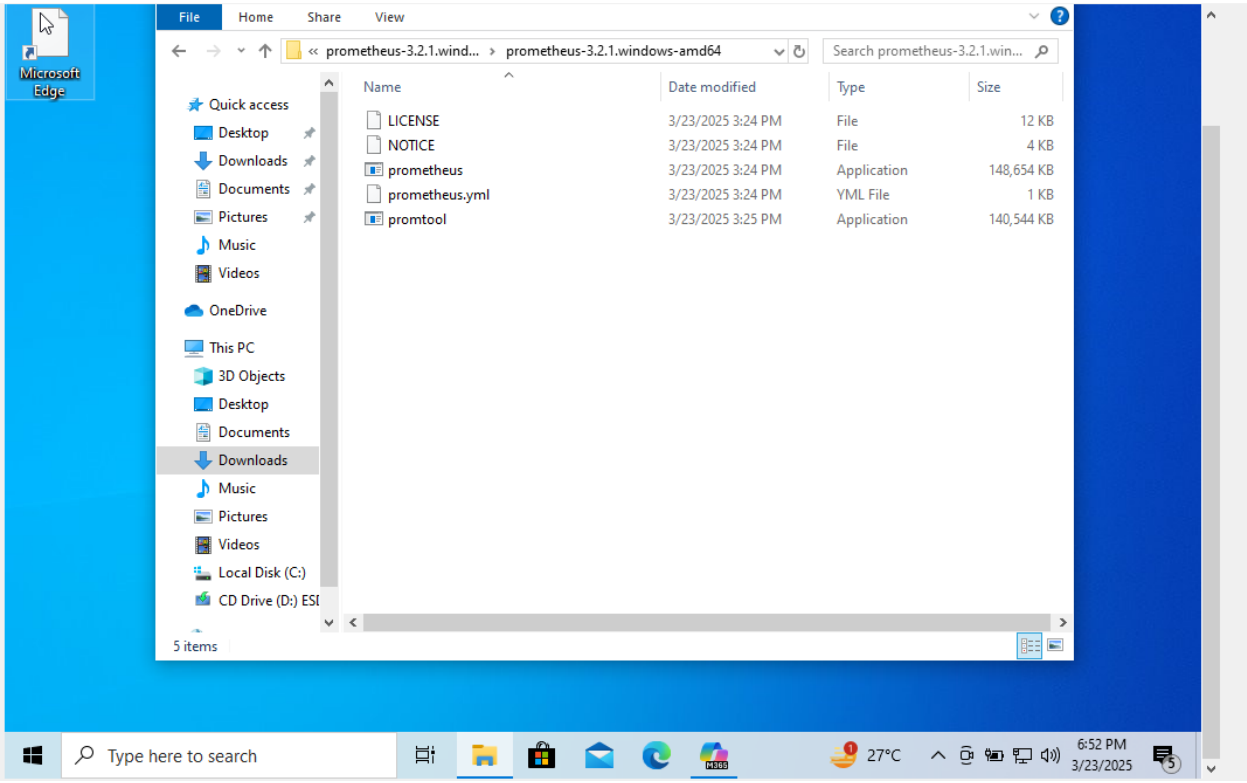


Figure 2: Configuration of Prometheus

Using a Custom Python Script: The Python script monitors CPU and memory usage and triggers the auto-scaling mechanism when thresholds are exceeded.

Auto-Scaling Configuration using GCP CLI

Setting Up Cloud Environment:

- Authenticate the GCP CLI:
- `gcloud auth login`
- Set the default project:
- `gcloud config set project [PROJECT_ID]`
- Enable required services:
- `gcloud services enable compute.googleapis.com`

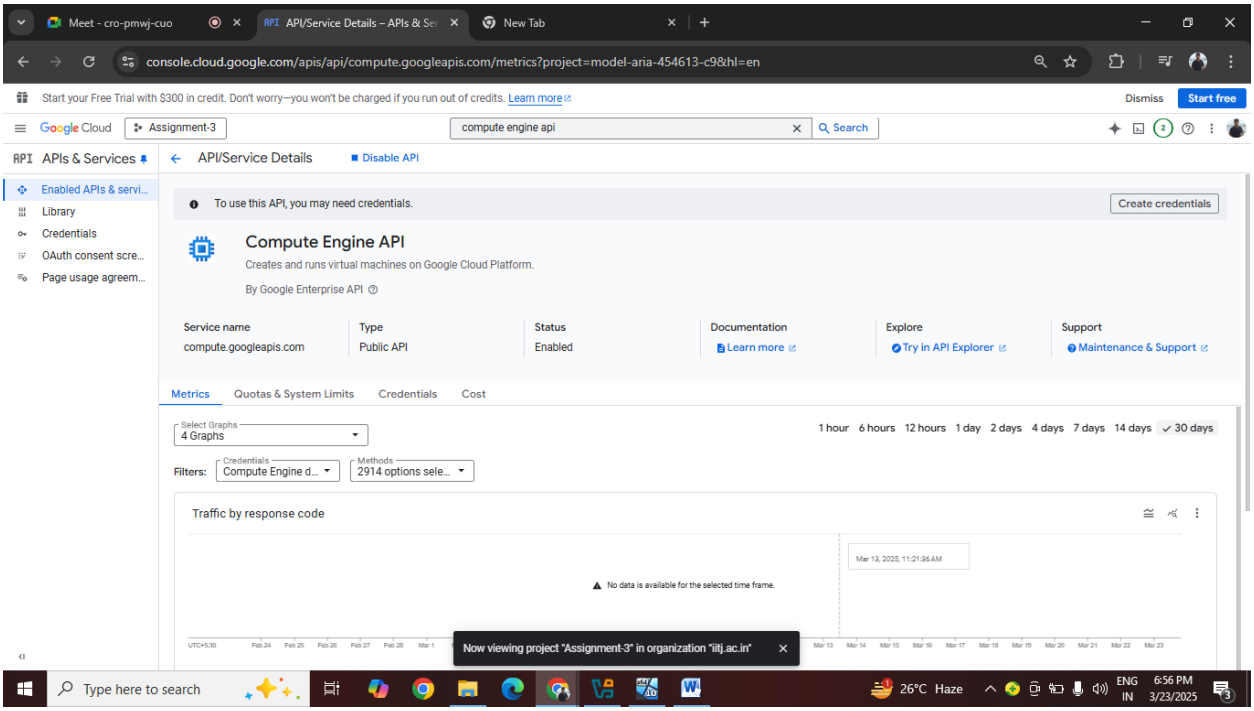


Figure 3 : API configuration

Creation of Service Account:

We have to create of service accounts for a Google Cloud Platform (GCP) project. The accounts are linked to specific roles and keys for managing Compute Engine VM instances. These service accounts are essential for automating processes like VM creation and resource management within the GCP environment.

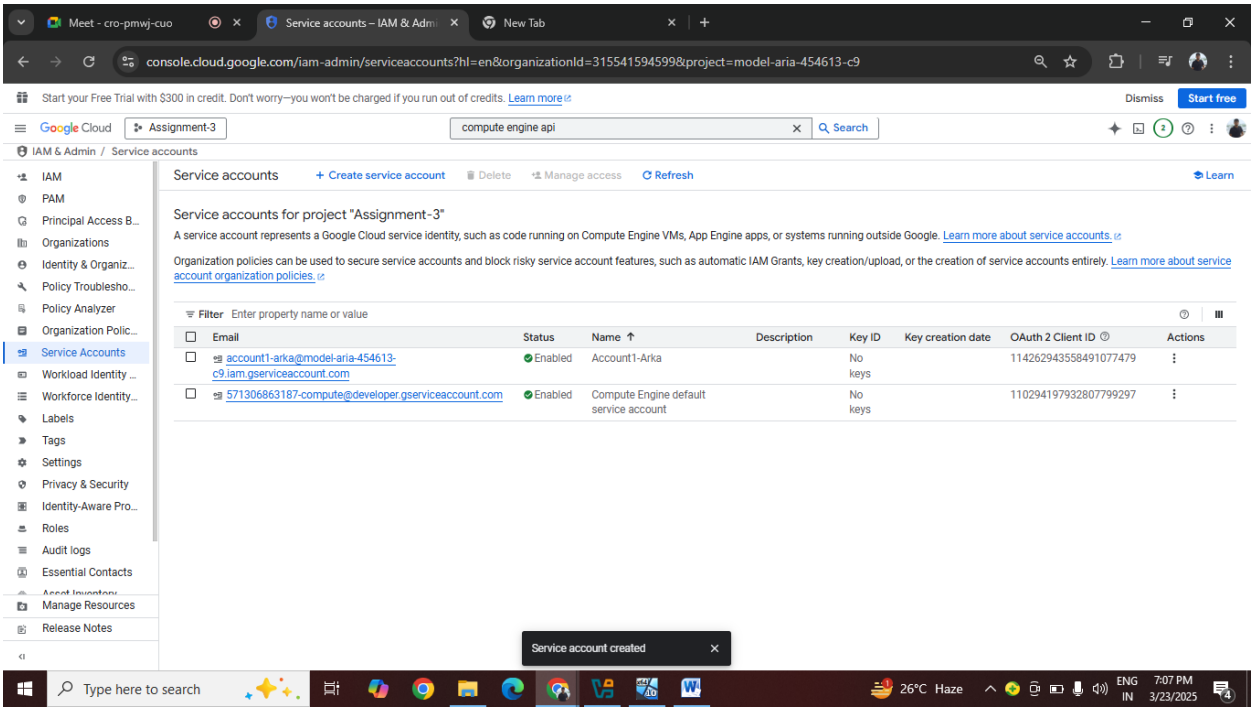


Figure 4: Creation of Service Account

Key Setup:

It highlights the creation of an active key for the service account, used for authenticating and automating tasks like VM management. The expiration date and key details are also visible, ensuring secure and controlled access to GCP resources.

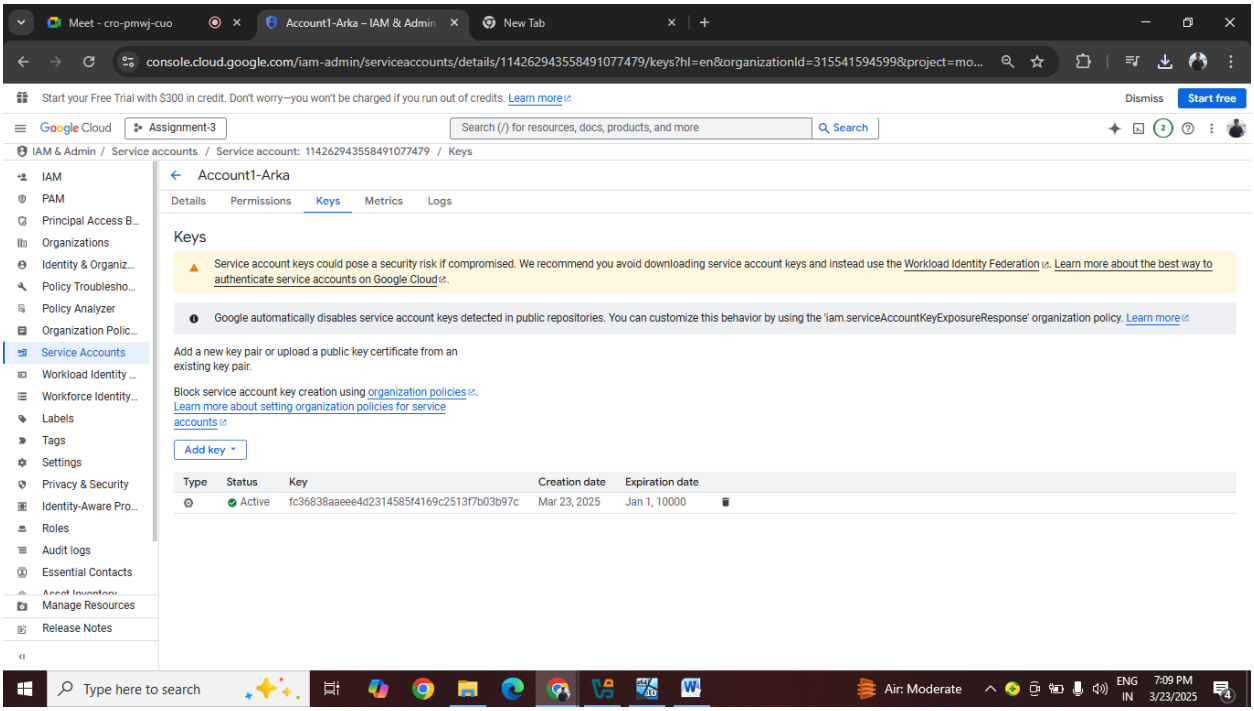


Figure 5: Key Setup

Google Cloud CLI Initialization:

This shows the initialization of the `gcloud` command-line interface. The user is guided through setting up the CLI, including running diagnostic checks to resolve any network connection issues. The system prompts the user to sign in, completing the configuration before using `gcloud` commands.

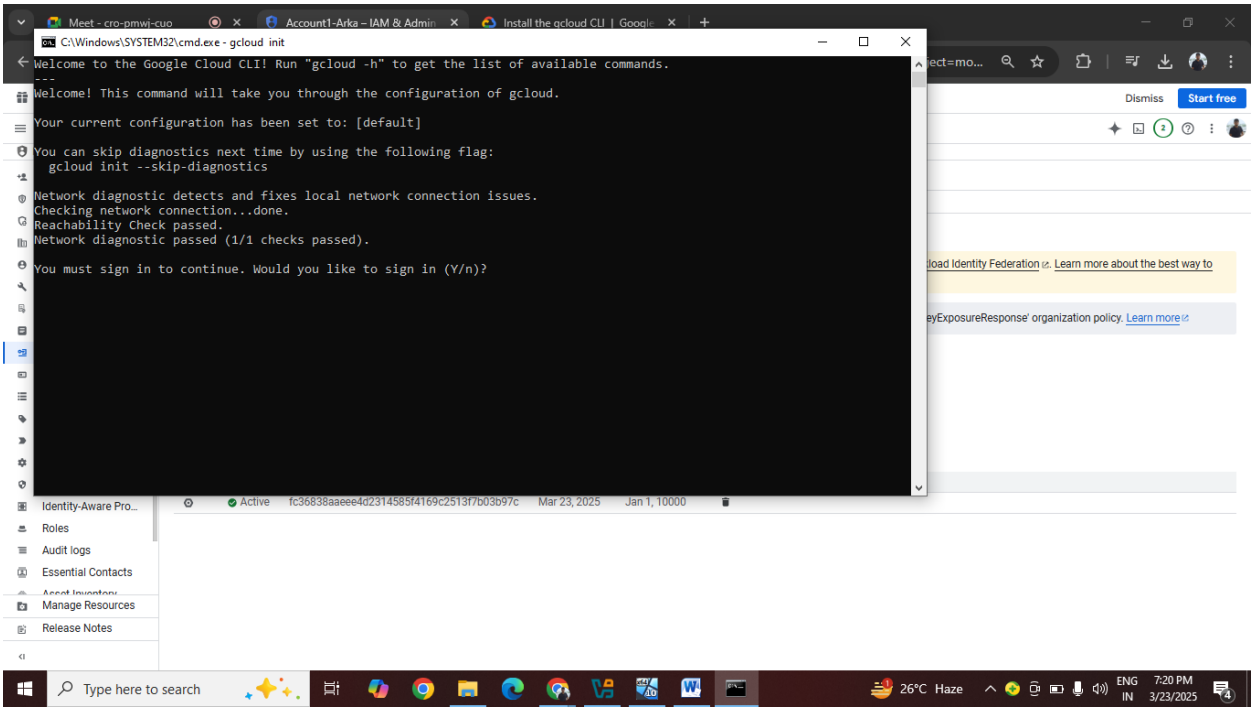


Figure 6: Google Cloud CLI Initialization

Authentication Confirmation:

After successful login, the user is shown the message indicating they have been authenticated with the Google Cloud CLI. The page confirms the authentication flow completion, allowing the user to interact with Google Cloud resources and execute commands through the CLI, such as managing VM instances or setting configurations.

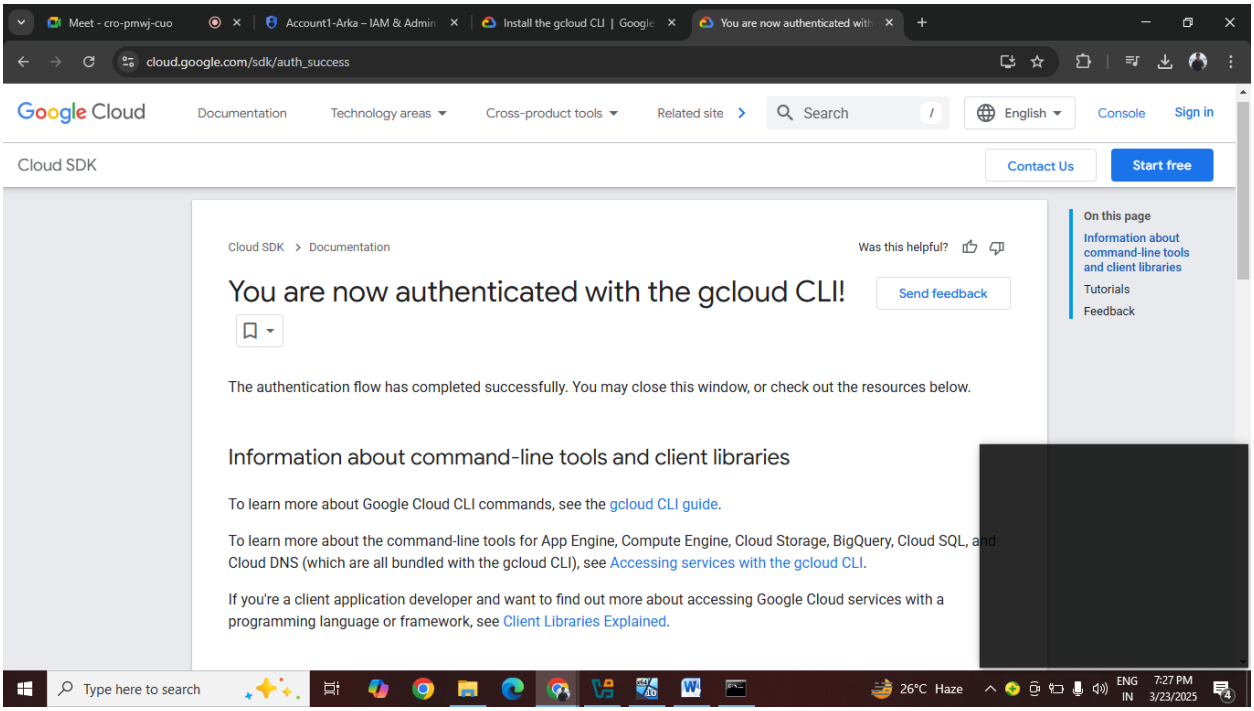


Figure 7: Authentication Confirmation

Authentication Success in the Cloud SDK:

This displays the confirmation message stating the user is now authenticated with the Google Cloud SDK. It provides a link to further documentation on using `gcloud` commands and libraries for managing cloud services, including Compute Engine, Cloud Storage, and BigQuery, which are all integrated with the `gcloud` CLI.

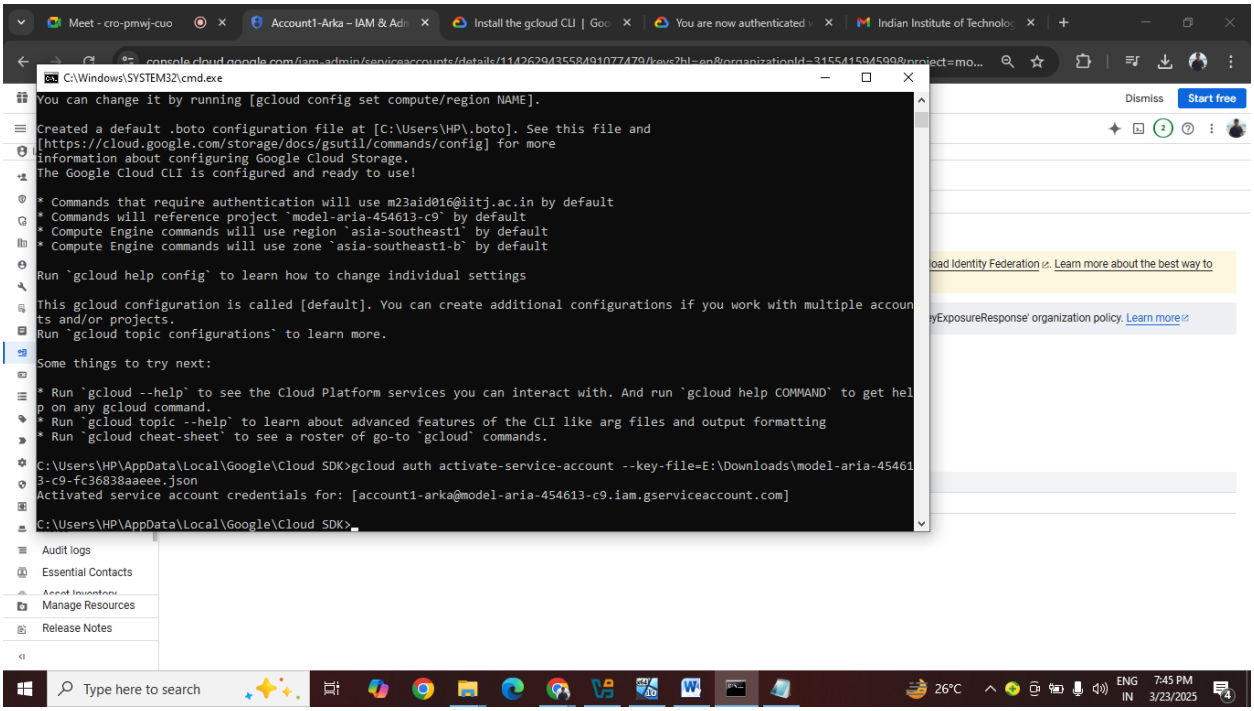


Figure 8: Authentication Success in the Cloud SDK

Sample Application Deployment

A simple Flask or Node.js application can be deployed on both the local and cloud VMs to validate the scaling process.

Shell Script for Resource Monitoring and Auto-Scaling

```
@echo off
REM Check the system load and create a VM in GCP if it exceeds the threshold

REM Set the threshold value
SET THRESHOLD=75

REM Get the current CPU load (as a percentage)
FOR /F "tokens=2 delims==" %%I IN ('wmic cpu get loadpercentage /value') DO
SET CPULOAD=%%I
```

```
REM Print the current CPU load
echo Current CPU Load: %CPULOAD%

REM Check if the load exceeds the threshold
IF %CPULOAD% GEQ %THRESHOLD% (
    echo Load exceeded %THRESHOLD%!. Creating a new VM instance in GCP...

    REM Make sure you are logged in to GCP
    gcloud auth login

    REM Set your GCP project
    gcloud config set project your-project-id

    REM Create a new VM instance (customize the command as per your
requirements)
    gcloud compute instances create "instance-name" ^
        --zone=us-central1-a ^
        --image-family=debian-9 ^
        --image-project=debian-cloud ^
        --machine-type=n1-standard-1

    echo VM instance created successfully!
) ELSE (
    echo System load is within safe limits.
)

pause
```

Results and Discussion

The system successfully monitors resource usage on the local VM and triggers auto-scaling when the threshold (75%) is exceeded. The GCP CLI automation effectively provisions cloud instances based on demand, demonstrating the feasibility of dynamic resource scaling in a cloud environment.

It shows the successful creation of three VM instances in Google Cloud Platform (GCP) after the system's resource utilization exceeded 75%. These VMs, located in the us-central1-c zone, are automatically provisioned to handle the increased load. The internal and external IP addresses are displayed, with the option to connect via SSH. This setup is part of an auto-scaling solution, where the GCP CLI is triggered by resource monitoring. The process ensures that the system can dynamically scale its infrastructure to manage resource demands effectively, providing seamless service even under heavy load conditions.

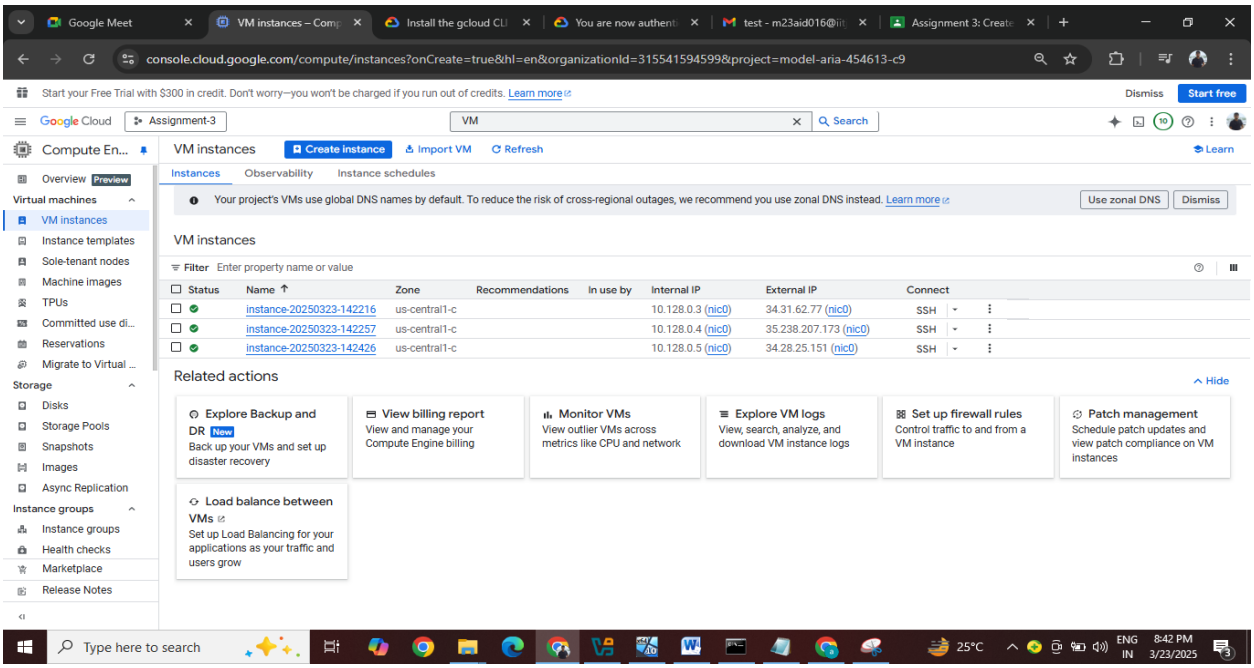
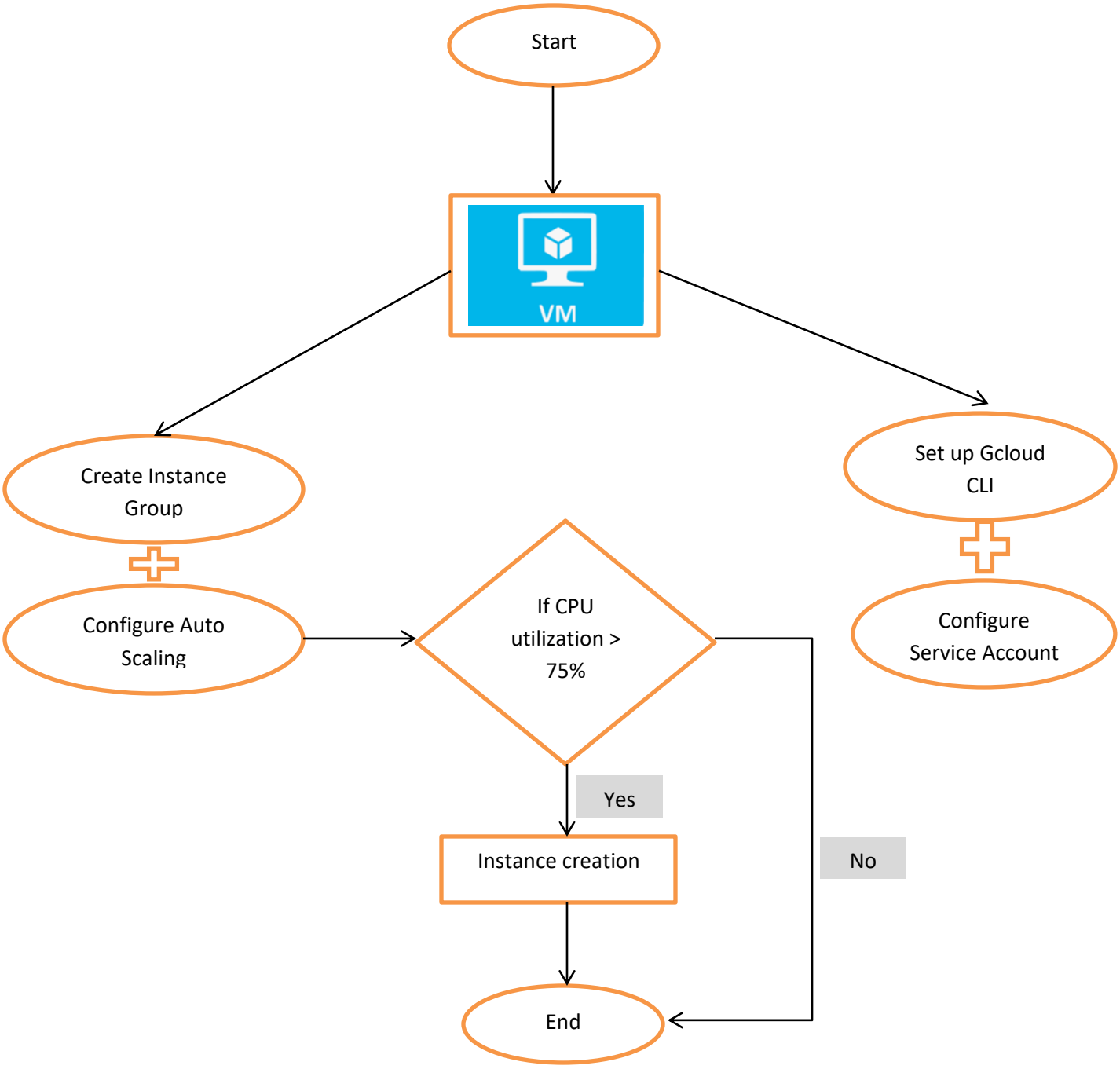


Figure 9: Instance Creation

Fig: Application to create autoscaling

This report provides a step-by-step guide for setting up a local VM, monitoring its resources, and triggering auto-scaling to GCP when needed. The included shell script automates the process of checking system load and scaling the resources using the GCP CLI.

Architechture Diagram



Source Code Repository

Github Repository Link

https://github.com/ArkaGayen16/VccAssignment3.git

---

8. Recorded Video Demo

https://drive.google.com/file/d/1sS5lO6U7B6xcDitJqn3bWmSv9oTXdh5N/view?usp=drive_web

---

9. Deliverables

Included:

1. **Project Report** (This Document)
2. **Architecture Diagram**
3. **Source Code Repository**
4. **Recorded Video Demo**

Conclusion

In conclusion, this project successfully demonstrates the implementation of an auto-scaling mechanism that transitions workloads from a local Virtual Machine (VM) to the cloud when resource usage exceeds a specified threshold. By leveraging Google Cloud Platform (GCP) and the GCP CLI, the system ensures dynamic scaling based on real-time resource demands. This approach enhances the system's efficiency, allowing businesses and developers to maintain optimal performance without manual intervention, thus providing a scalable and cost-effective solution to resource management. The project highlights the importance of automation in managing cloud infrastructure and meeting growing computational demands.