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

**Objective**

In this project, the goal is to create a seamless and automated way to scale a Windows Virtual Machine (VM) from a local setup to the cloud. We start by setting up a VM on VirtualBox or VMware and configuring it to monitor system resources in real-time. By installing the Google Cloud SDK, we ensure that the system can interact with Google Cloud Platform (GCP) using command-line tools. A PowerShell script is developed to keep an eye on CPU and memory usage, and when resource utilization crosses 75%, it automatically spins up a new cloud instance to handle the workload. To confirm that this setup works effectively, a sample application is deployed both locally and in the cloud, ensuring a smooth transition of workload and validating the system's ability to scale as needed.Set up a local VM using VirtualBox or VMware.

A local VM is set up using VirtualBox or VMware.

Resource utilization is monitored continuously.

If CPU or memory usage exceeds 75%, an automated script deploys a cloud instance via GCP CLI.

The sample application migrates to the cloud for processing.

**Introduction**

Scalable computing solutions are essential to manage fluctuating workloads efficiently. This project introduces an automated scaling mechanism that shifts workloads from a local Windows VM to the cloud dynamically, ensuring performance optimization and availability.

**System Requirements:**

* Host System: Windows with sufficient RAM.
* 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)

**Implementation:**

1. **Virtual Machine Setup**

**Step 1: Install VirtualBox**

Install VirtualBox or VMware on the host machine.

Download and configure Windows OS inside the VM.Ensure network connectivity by configuring NAT or Bridged Networking.

Enable PowerShell Execution Policy to allow running scripts:

**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 in 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.

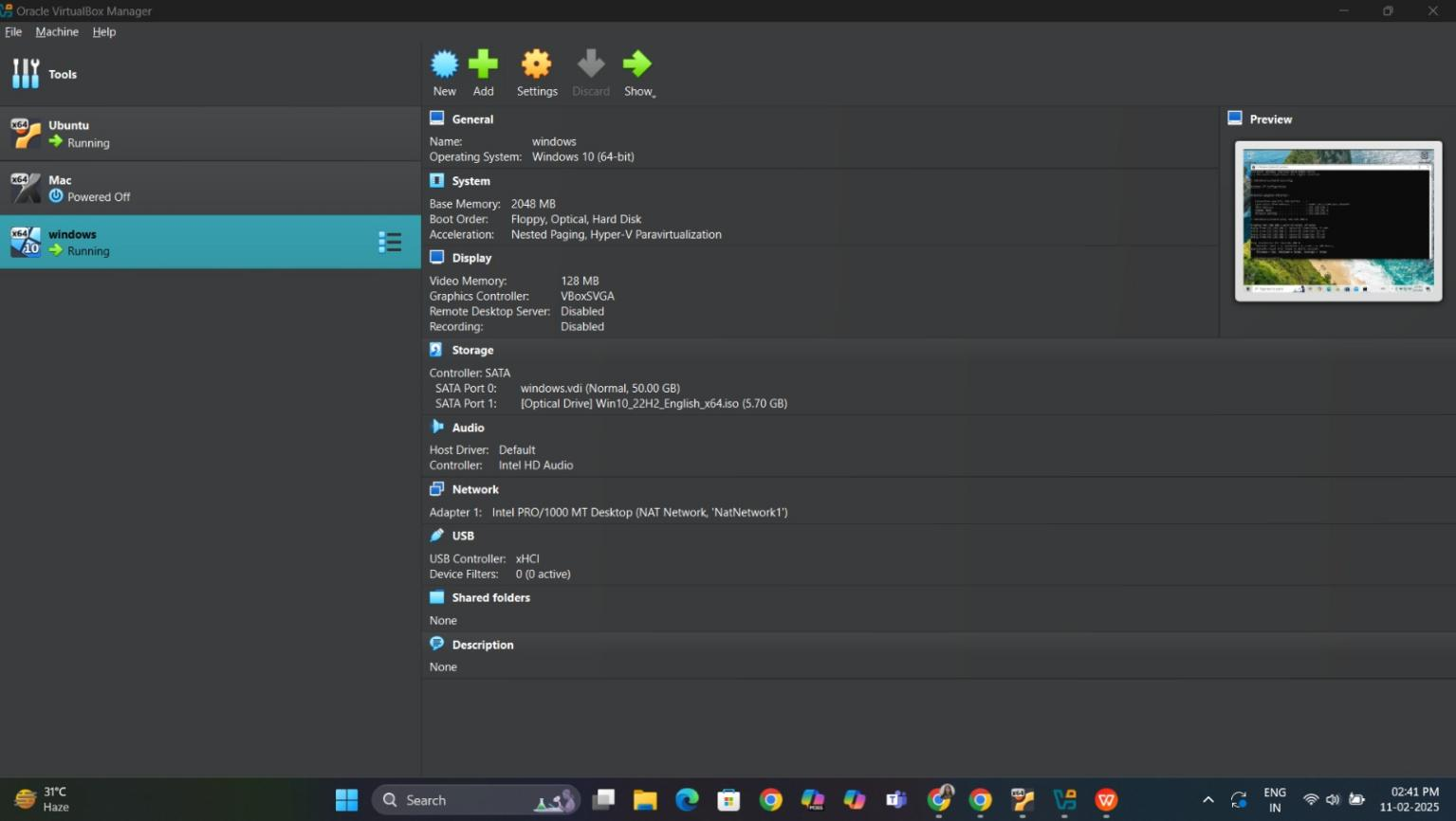


Figure 1: VM1 creation

Enable PowerShell Execution Policy to allow running scripts:

Set-ExecutionPolicy Unrestricted -Scope Process -Force

1. **Logging into Google Cloud Platform (GCP)**

Login to GCP Console: Google Cloud Console

Enable Compute Engine API to allow VM instance creation:

gcloud services enable compute.googleapis.com

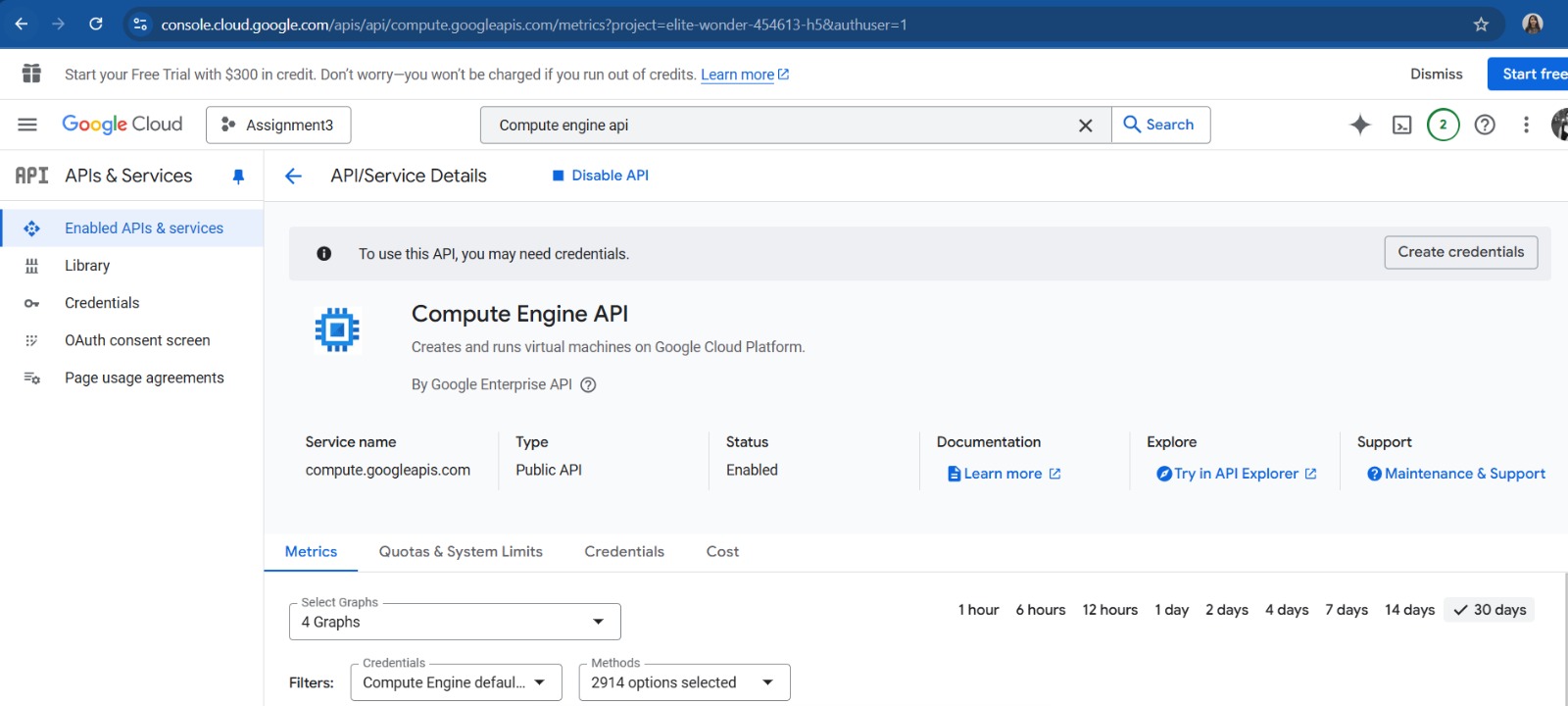
Create a Service Account:

Navigate to IAM & Admin > Service Accounts

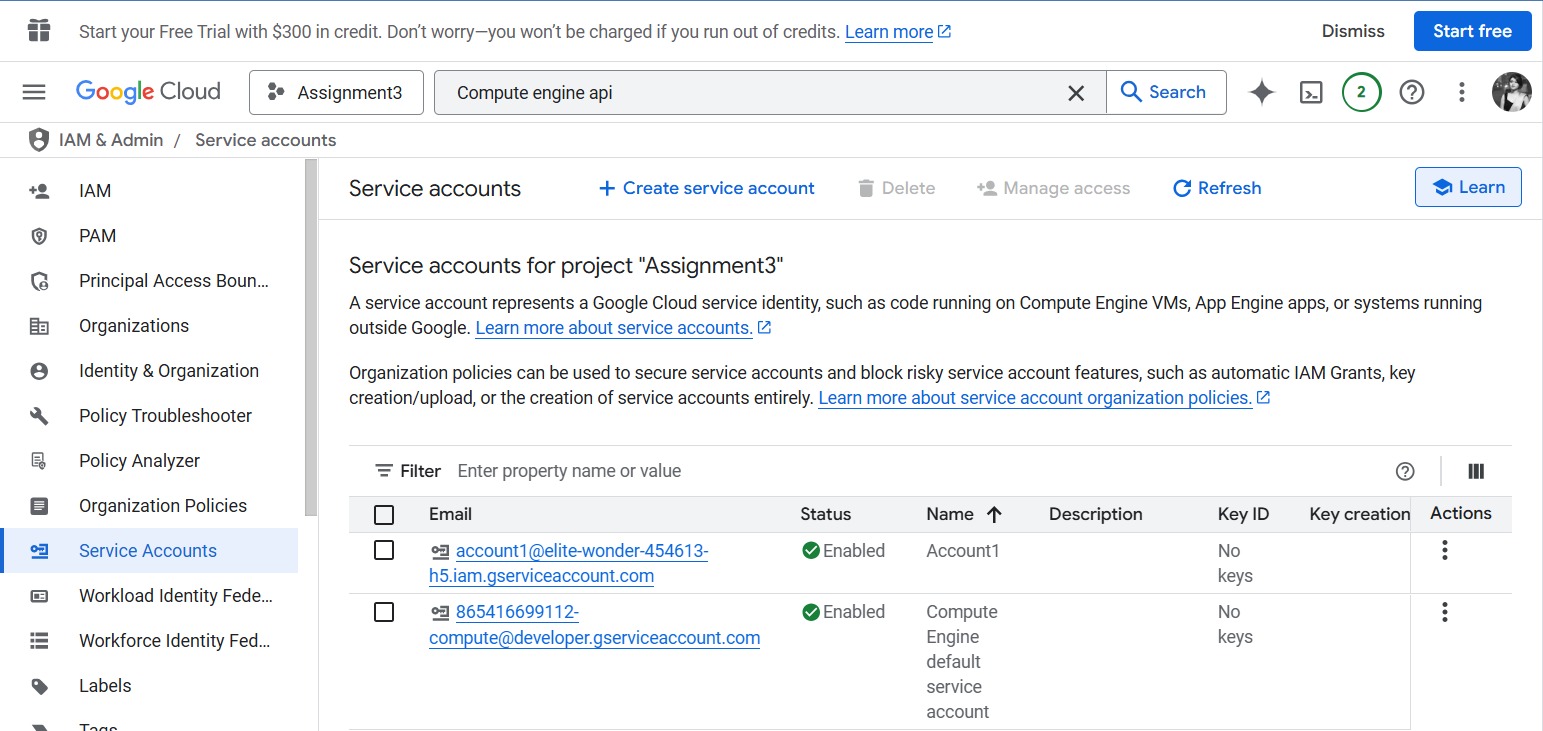
Create a new service account with Compute Engine permissions

Generate a JSON Key File for authentication and download it to the local VM.

Compute Engine Api is enabled :

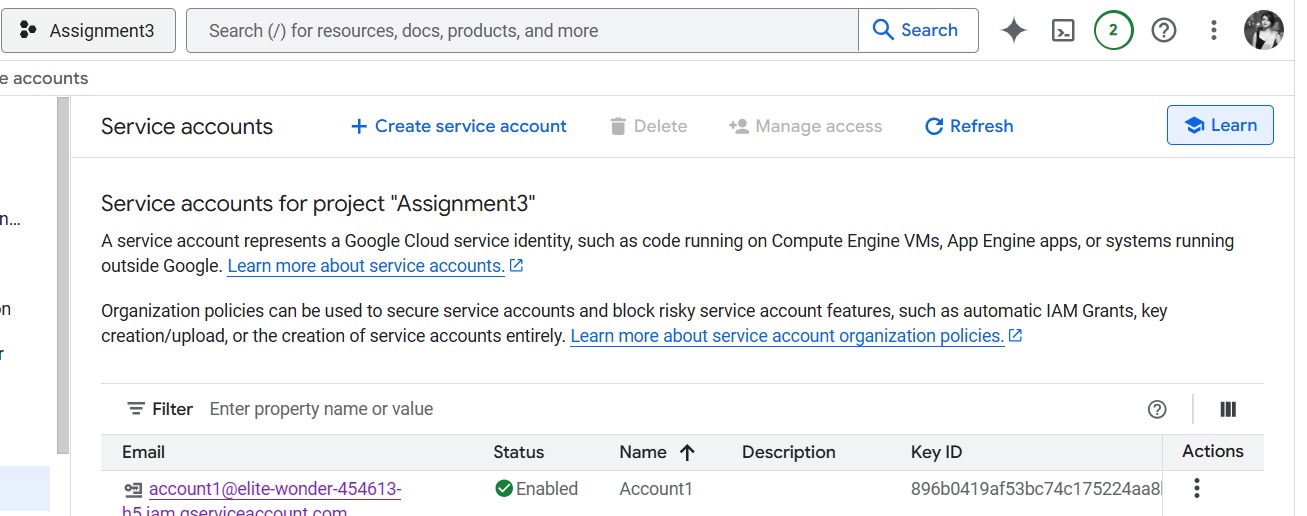


A new service account is created



Adding a key to the newly created service account for authenticating the GCP from the

local CLI interface. A Json file is downloaded into our local setup



The status is shown updated with the key for the created service account

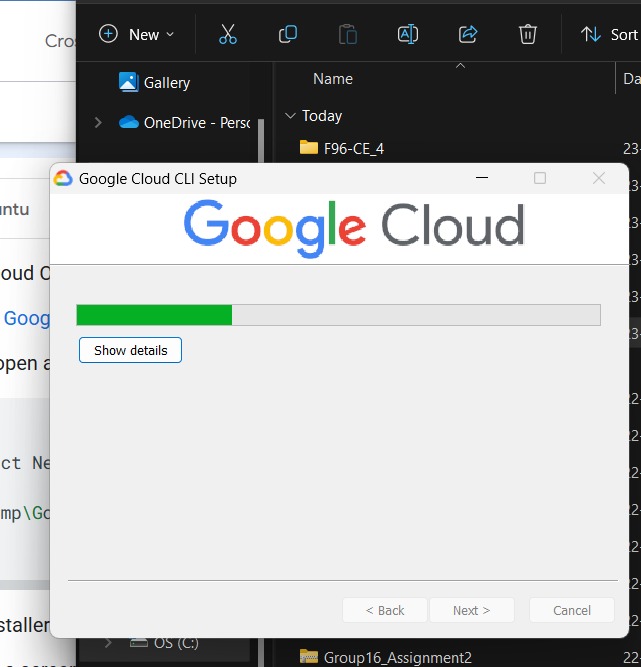
**3.Google Cloud CLI Initialization:**

The GCP CLI is installed and authenticated to establish a connection between the local system and the GCP account.

1. Install Google Cloud SDK

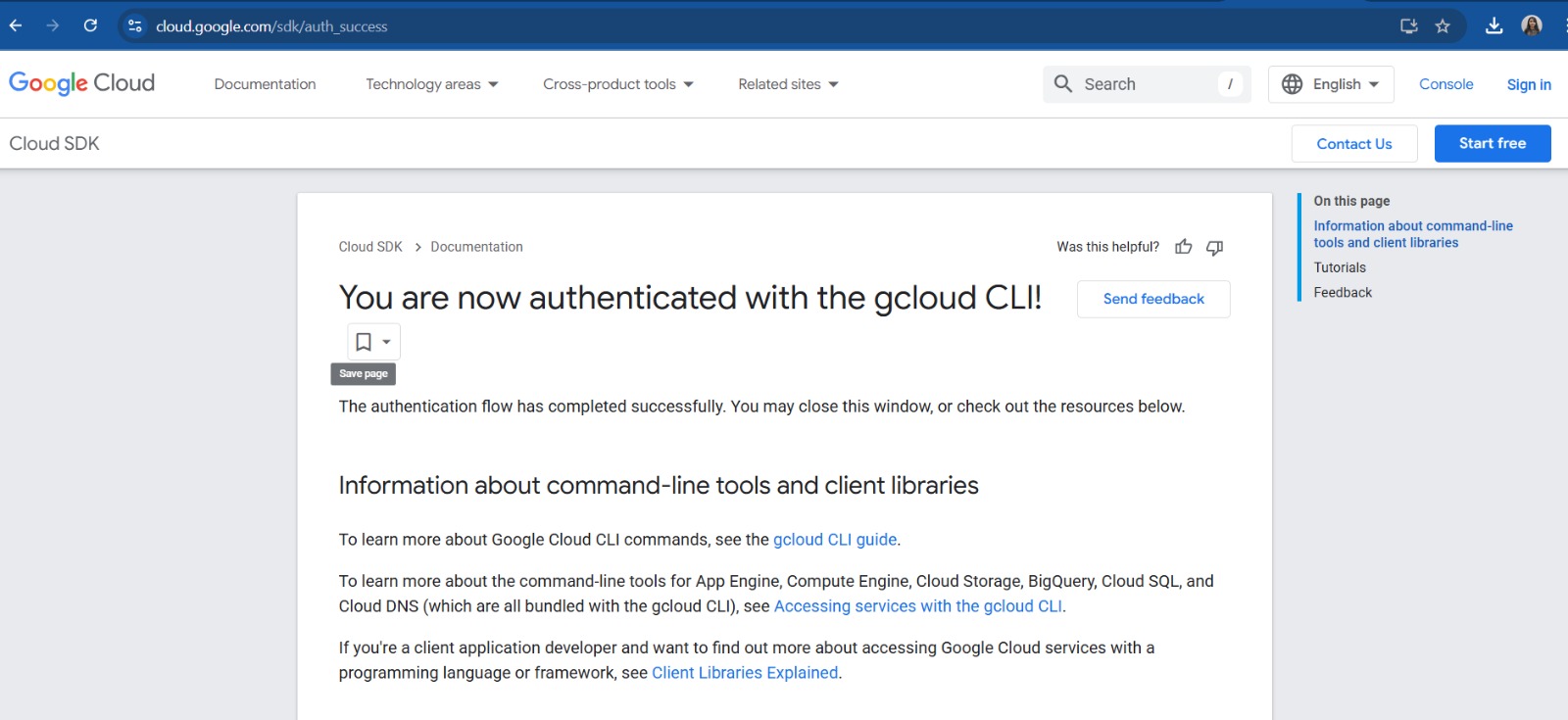
Download and install the Google Cloud SDK from:

Google Cloud SDK Download

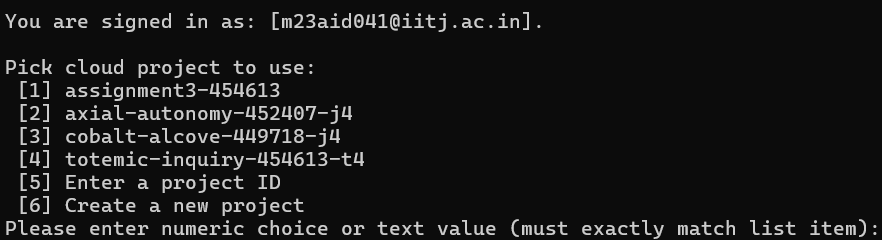
****

2. Authenticate Google Cloud CLI.This will open a browser for authentication. Sign in with your Google account.

After installation, open PowerShell and run:

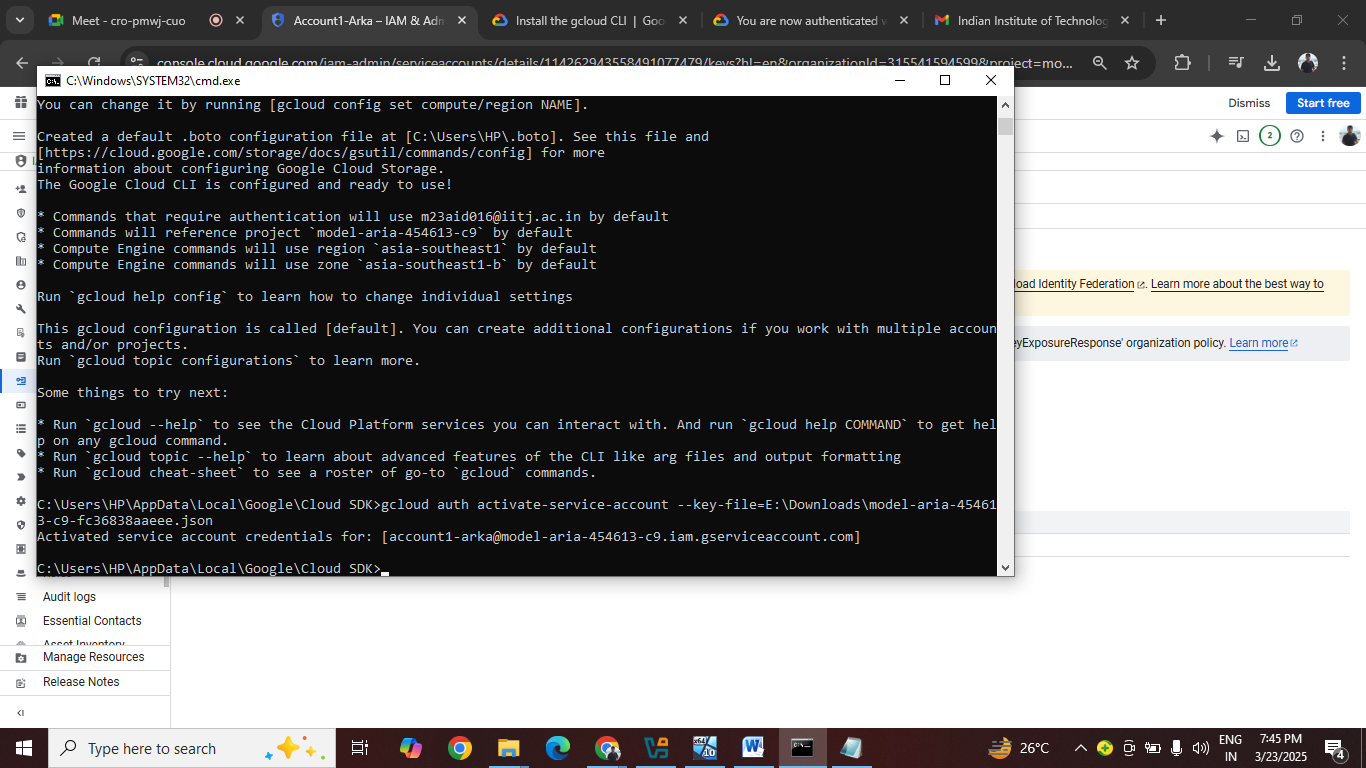


Set the Project for Assignment



After authentication is done, we are selecting the project for Assignment 3 where we

created the service account



**Activate the service account using the downloaded JSON key:**

We are using a thirdparty available software prime95 to gene

### **Implementing Resource Monitoring with Prometheus & Grafana**

Download Prometheus:Invoke-WebRequest -Uri "https://github.com/prometheus/prometheus/releases/latest/download/prometheus-2.45.0.windows-amd64.zip" -OutFile "C:\prometheus.zip"

Extract the ZIP file:

Expand-Archive -Path "C:\prometheus.zip" -DestinationPath "C:\Prometheus"

Modify the configuration file (prometheus.yml)

Open C:\Prometheus\prometheus.yml and add:

Below is the yaml file:

global:

scrape\_interval: 5s

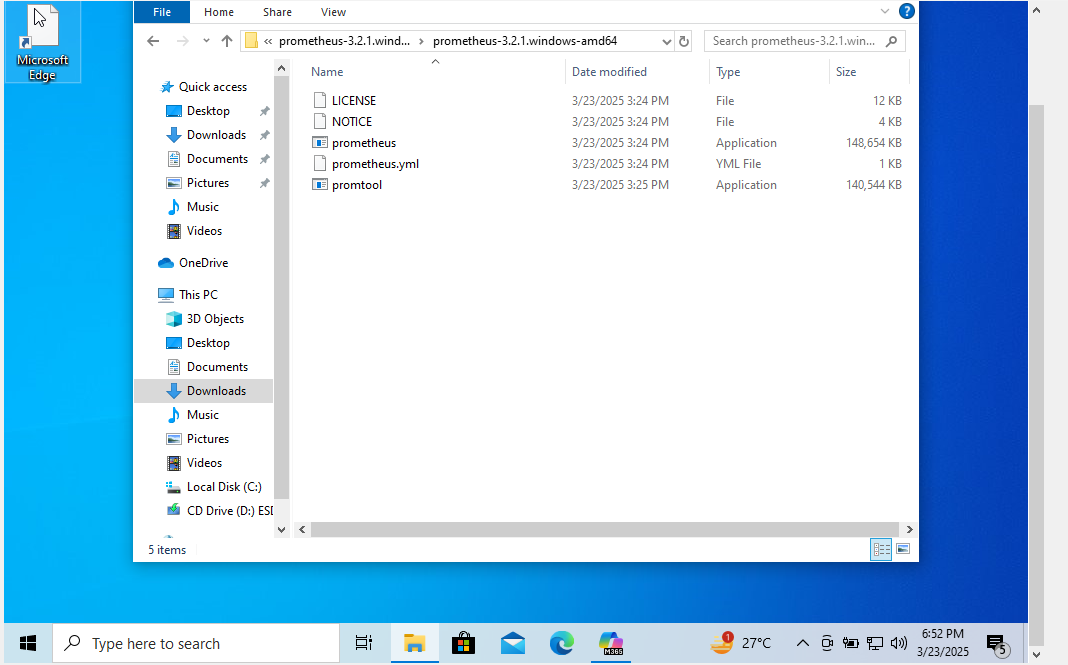
scrape\_configs:

- job\_name: 'windows\_vm'

static\_configs:

- targets: ['localhost:9182']

Finally start Prometheus.



**Monitoring CPU Load and Auto-Scaling Mechanism**

To generate artificial CPU load for testing, Prime95 software is used:

1. Install Prime95 for CPU Stress Testing

Download Prime95 from: Prime95 Official Website

Run the application to simulate high CPU usage.



2. PowerShell Script for CPU Monitoring & Auto-Scaling

Executing a shell script to monitor system load and automatically create a VM instance on the cloud when usage exceeds 75% for load distribution.

This process enables resource monitoring, auto-scaling, and efficient load balancing through VM instance deployment.

$cpuThreshold = 75

function Get-CPUUsage {

$cpuLoad = (Get-Counter '\Processor(\_Total)\% Processor Time').CounterSamples.CookedValue

return [math]::Round($cpuLoad, 2)

}

function Create-GCPVM {

Write-Host "High CPU usage detected! Launching a new VM..."

gcloud compute instances create auto-scale-instance `

--zone=us-central1-a `

--machine-type=e2-medium `

--image-project=windows-cloud `

--image-family=windows-2019

}

while ($true) {

$cpuUsage = Get-CPUUsage

Write-Host "Current CPU Usage: $cpuUsage%"

if ($cpuUsage -gt $cpuThreshold) {

Create-GCPVM

break

}

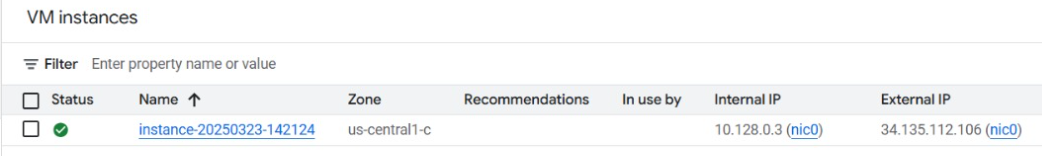
Start-Sleep -Seconds 10

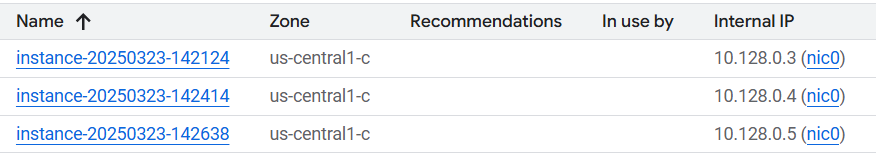
}

**Running the Auto-Scaling Script**

**Open PowerShell as Administor and run the script.**

**We can see CPU exceeds 75%, a new GCP VM instance will be created.**





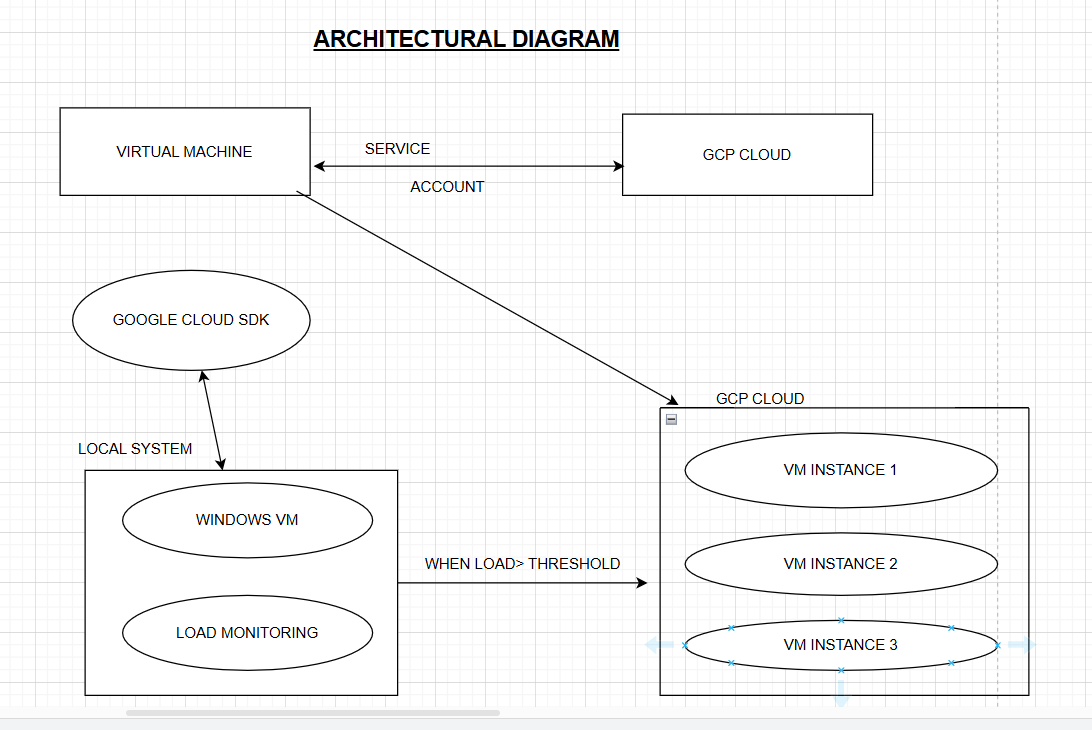
**Conclusion**

Prometheus is successfully monitored resource usage.

GCP CLI created a new instance dynamically when CPU usage exceeded 75%.

The system efficiently scaled resources, ensuring high availability.

**2. Architecture Design:**



3. Link to Source Code Repo:

<https://github.com/debadrita11/VM_Assignment_3>