

Virtual Cloud Computing  
(CSL7510)

*Submitted by*  
Sachin Kumar Gupta

**Assignment 3:**

*Create a Local VM and Auto-Scale It to GCP or Any Other Public Cloud When Resource Usage Exceeds  
75%*



Indian Institute of Technology Jodhpur  
Department of Artificial Intelligence and Data Science  
*March 2025*

## **OBJECTIVE**

Create and configure multiple Virtual Machines (VMs) using VirtualBox, establish a network between them, and deploy a microservice-based application across the connected VMs.

# 1. Introduction

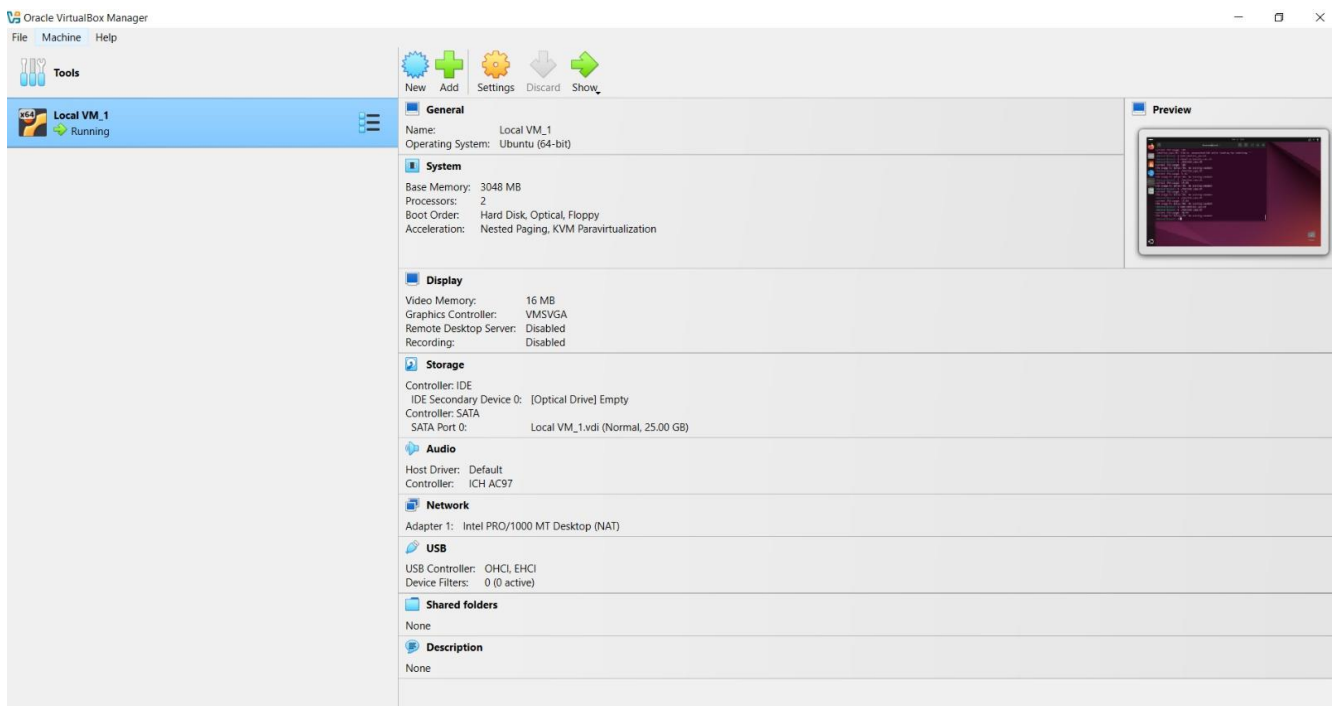
The objective of this project is to create a local virtual machine (VM), implement resource monitoring, and configure auto-scaling to a public cloud (GCP, AWS, or Azure) when resource usage exceeds 75%. This implementation demonstrates the process of monitoring system resources, triggering auto-scaling, and migrating to the cloud when needed.

## 2. VM Creation and Configuration

### 1. Local VM Creation

#### VM Setup:

1. **Install Virtual Box or VMware:**
  - Download and install Virtual Box or VMware Workstation Player on your system.
2. **Create a New Virtual Machine:**
  - Set up a new virtual machine (VM) by selecting a suitable operating system (e.g., Ubuntu or another Linux distribution).
3. **Allocate Resources:**
  - Assign sufficient resources (such as CPU, RAM, and Storage) to your VM based on your specific requirements.



1.Initial configuration and setup of the VM in VirtualBox.

## 2. Install Google Cloud SDK on the VM

Steps to install Google Cloud SDK:

### Update Package Lists:

- Begin by updating the package lists for upgrades and new package installations:

```
sudo apt-get update
```

### Install Required Dependencies:

- Install the necessary dependencies for the Google Cloud SDK:

```
sudo apt-get install apt-transport-https ca-certificates gnupg
```

### Verify the Project ID:

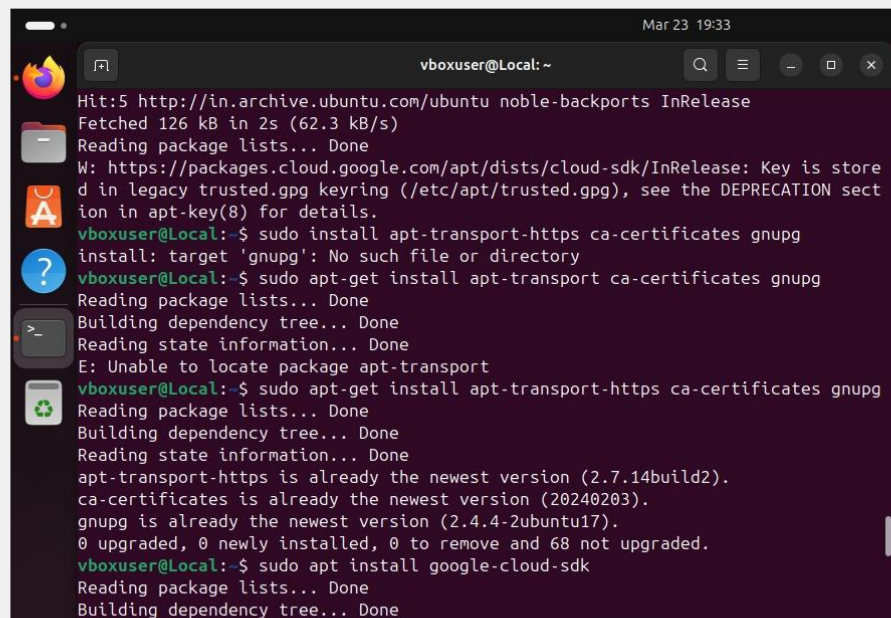
- After installing the SDK, use the following command to check the available projects:

```
gcloud project list
```

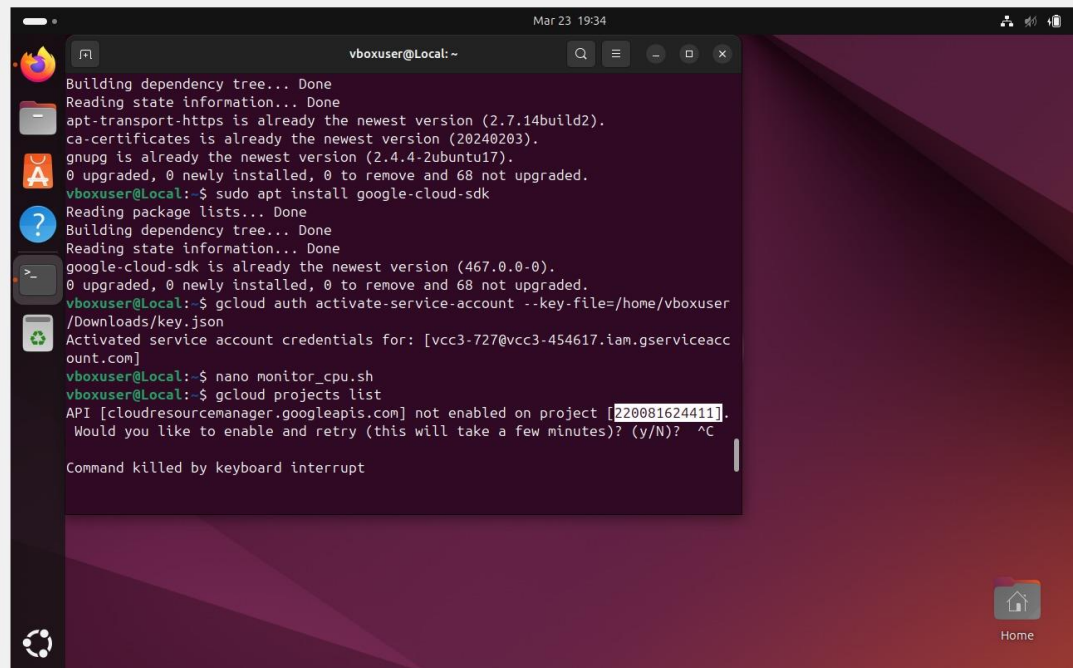
- This step confirms that the SDK installation was successful and allows you to view the project ID.

Local VM\_1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help



```
Mar 23 19:33
vboxuser@Local: ~
Hit:5 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease
Fetched 126 kB in 2s (62.3 kB/s)
Reading package lists... Done
W: https://packages.cloud.google.com/apt/dists/cloud-sdk/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the DEPRECATION section in apt-key(8) for details.
vboxuser@Local:~$ sudo install apt-transport-https ca-certificates gnupg
install: target 'gnupg': No such file or directory
vboxuser@Local:~$ sudo apt-get install apt-transport ca-certificates gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package apt-transport
vboxuser@Local:~$ sudo apt-get install apt-transport-https ca-certificates gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.7.14build2).
ca-certificates is already the newest version (20240203).
gnupg is already the newest version (2.4.4-2ubuntu17).
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.
vboxuser@Local:~$ sudo apt install google-cloud-sdk
Reading package lists... Done
Building dependency tree... Done
```



```
vboxuser@Local: ~  
Building dependency tree... Done  
Reading state information... Done  
apt-transport-https is already the newest version (2.7.14build2).  
ca-certificates is already the newest version (20240203).  
gnupg is already the newest version (2.4.4-2ubuntu17).  
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.  
vboxuser@Local:~$ sudo apt install google-cloud-sdk  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
google-cloud-sdk is already the newest version (467.0.0-0).  
0 upgraded, 0 newly installed, 0 to remove and 68 not upgraded.  
vboxuser@Local:~$ gcloud auth activate-service-account --key-file=/home/vboxuser/Downloads/key.json  
Activated service account credentials for: [vcc3-727@vcc3-454617.iam.gserviceaccount.com]  
vboxuser@Local:~$ nano monitor_cpu.sh  
vboxuser@Local:~$ gcloud projects list  
API [cloudresourcemanager.googleapis.com] not enabled on project [220081624411].  
Would you like to enable and retry (this will take a few minutes)? (y/N)? ^C  
Command killed by keyboard interrupt
```

For checking the project id (it will be applicable if only success)

### 3. Create a Service Account on Google Cloud Platform (GCP)

Steps to create a Service Account:

#### Access Google Cloud Console:

- Navigate to the Google Cloud Console ([console.cloud.google.com](https://console.cloud.google.com)).

#### Go to IAM & Admin Section:

- In the left-hand menu, find and select IAM & Admin.

#### Select Service Accounts:

- Under IAM & Admin, choose Service Accounts.

#### Create a New Service Account:

- Click on the Create Service Account button.

#### Provide a Name:

- Enter a name for the service account, then click Create.

#### Assign Roles to the Service Account:

- Select roles such as Editor, Viewer, or custom roles based on the permissions you need.

## Finish the Creation Process:

- Once roles are assigned, click Done to complete the creation of the service account.

The screenshot shows the Google Cloud IAM & Admin console. The left sidebar contains navigation links: IAM, PAM, Principal Access Boundary, Organizations, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (highlighted), Workload Identity Federat..., Manage Resources, and Release Notes. The main content area is titled 'Service accounts' and includes a '+ Create service account' button, 'Delete', 'Manage access', and 'Refresh' options. Below this, there is a section for 'Service accounts for project "VCC3"' with explanatory text and links. A table lists the service accounts:

Filter	Email	Status	Name	Description	Key ID	Key creation date	Actions
	<a href="mailto:220081624411-compute@developer.gserviceaccount.com">220081624411-compute@developer.gserviceaccount.com</a>	Enabled	Compute Engine default service account		No keys		
	<a href="mailto:vcc3-727@vcc3-454617.iam.gserviceaccount.com">vcc3-727@vcc3-454617.iam.gserviceaccount.com</a>	Enabled	VCC3		fb34b970749c33121f38c52587ff000fefa17506	Mar 23, 2025	

## 4. Download the Service Account Key

### Select the Service Account:

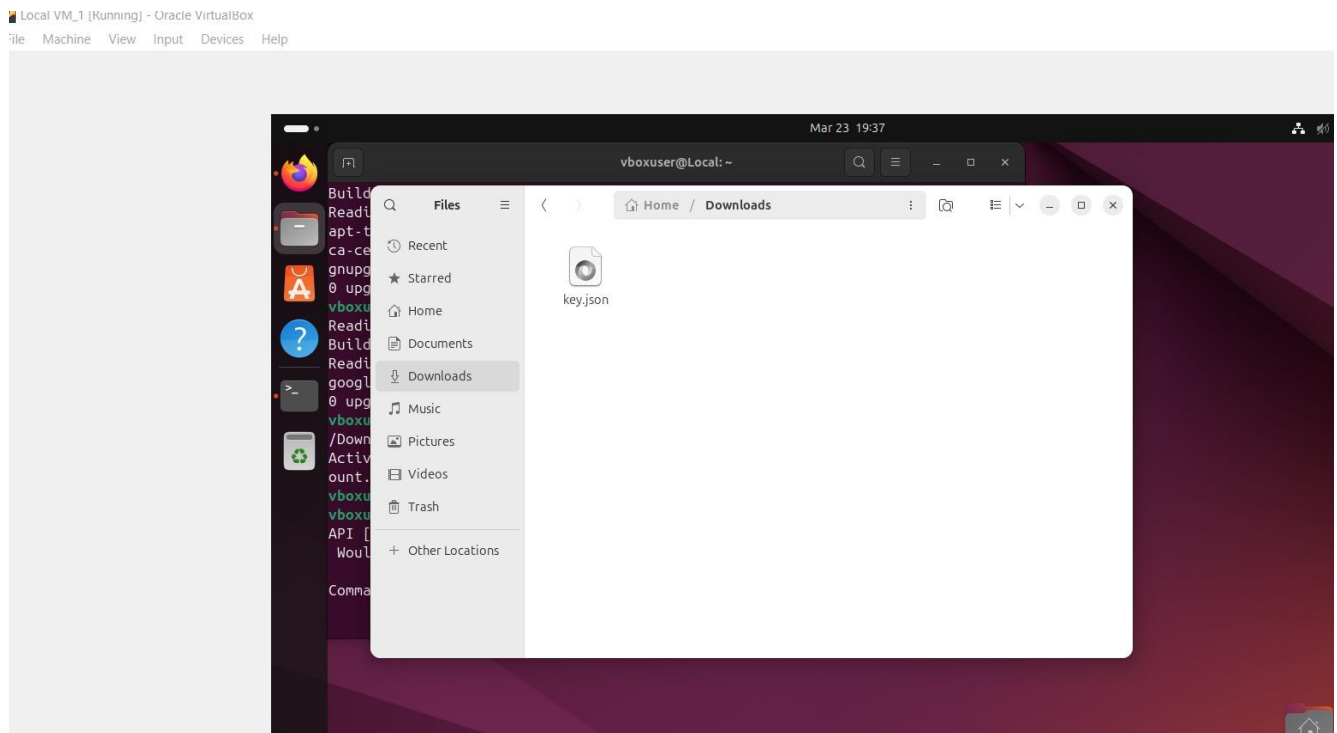
- In the Service Accounts section, find and select the newly created service account.

### Add a Key:

- Click on Add Key and choose JSON from the options.

### Download the Key:

- The key file in JSON format will be downloaded to your local machine.



## 5. Activate the Service Account on VM

### Open the Terminal:

- On your VM, open the terminal.

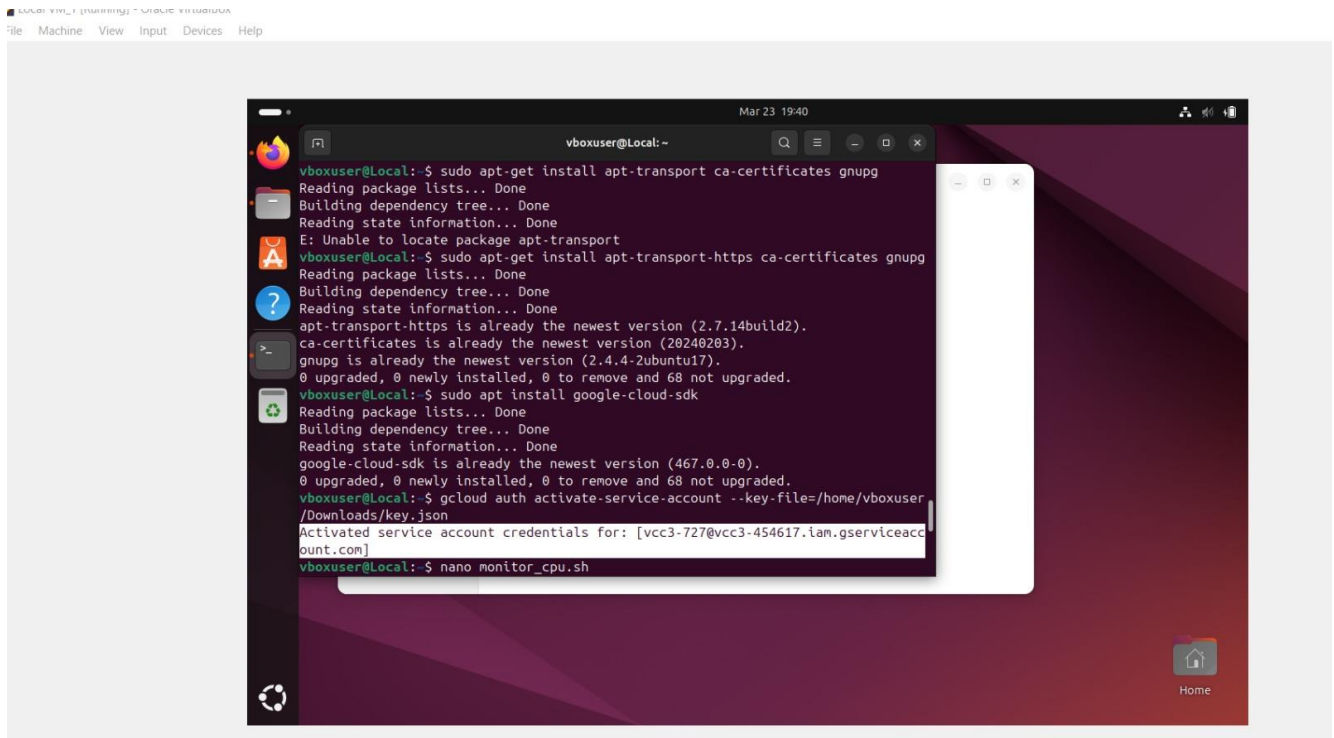
### Navigate to the Directory:

- Change to the directory where the service account key file (JSON) is stored.

### Authenticate with the Service Account Key:

- Run the following command to authenticate using the service account key:

```
gcloud auth activate-service-account --key-file=/path/to/your/key.json
```



## 6. Run the Script for Auto-scaling and Monitoring

### 1. Create or Find a Script:

- You can either create or find an existing script that monitors system resources, such as CPU and memory usage, on your VM. Tools like Prometheus and Grafana can be used for advanced monitoring, or you can write a custom shell script for simpler needs.

### 2. Monitor Resource Usage:

- The script should continuously check the system's resource usage (e.g., CPU, memory).

### 3. Trigger Auto-scaling:

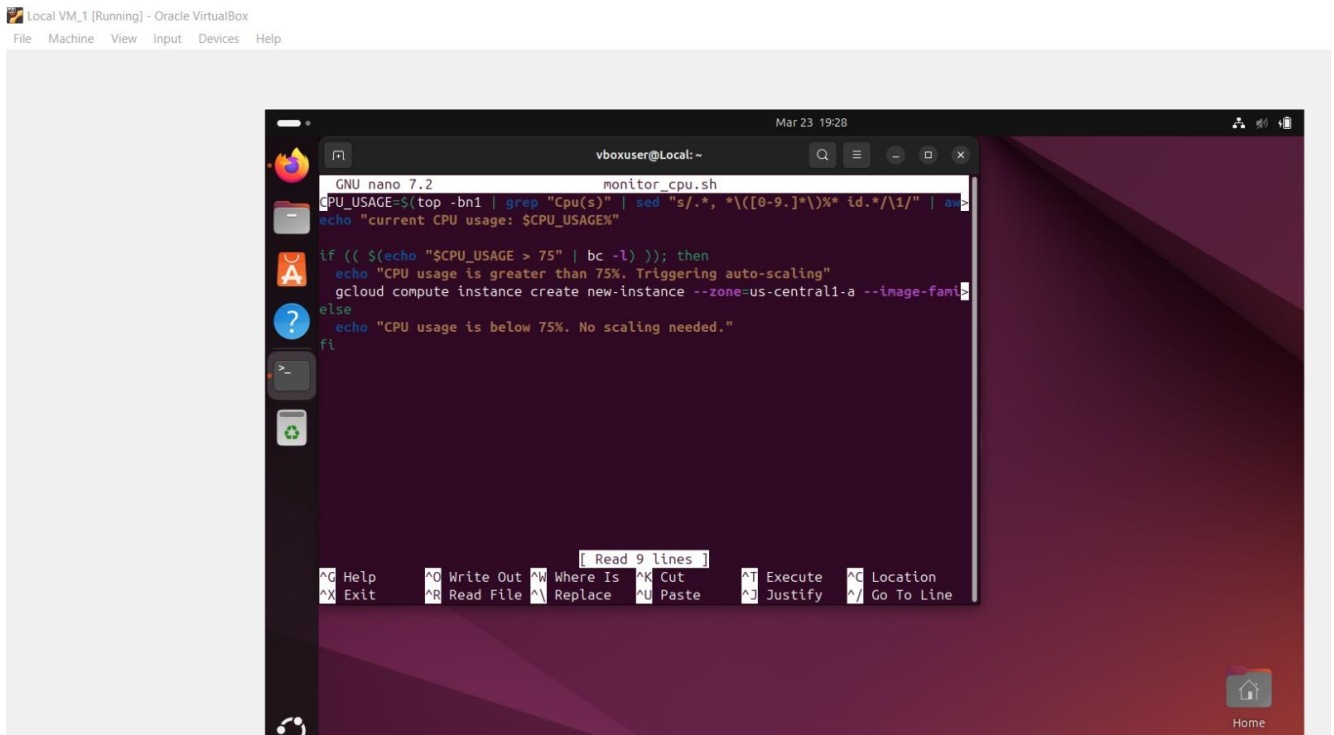
- If the script detects that resource usage exceeds 75%, it should trigger the auto-scaling process to a public cloud (e.g., Google Cloud Platform). This could involve actions like spinning up additional VM instances on the cloud to handle the increased load.

**Write below script:**

```
# Check CPU usage and scale if > 75%
CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/., *([0-9.])%* id.*\1/" | awk '{print 100 - $1}')
echo "Current CPU usage: $CPU_USAGE%"

if (( $(echo "$CPU_USAGE > 75" | bc -l) )); then
    echo "CPU usage is greater than 75%. Triggering auto-scaling."
    # Trigger auto-scaling (e.g., create a new instance in GCP)
    gcloud compute instances create new-instance --zone=us-central1-a --image-family=debian-9 --image-project=debian-cloud
else
    echo "CPU usage is below 75%. No scaling needed."
fi
```





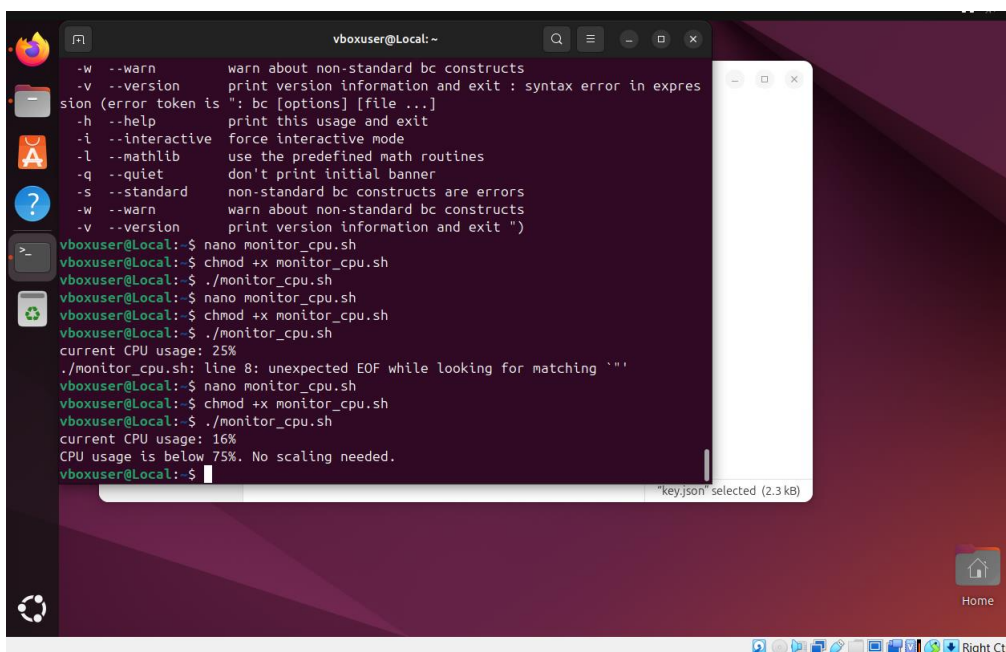
Save and close the file (CTRL + X, Y, Enter).

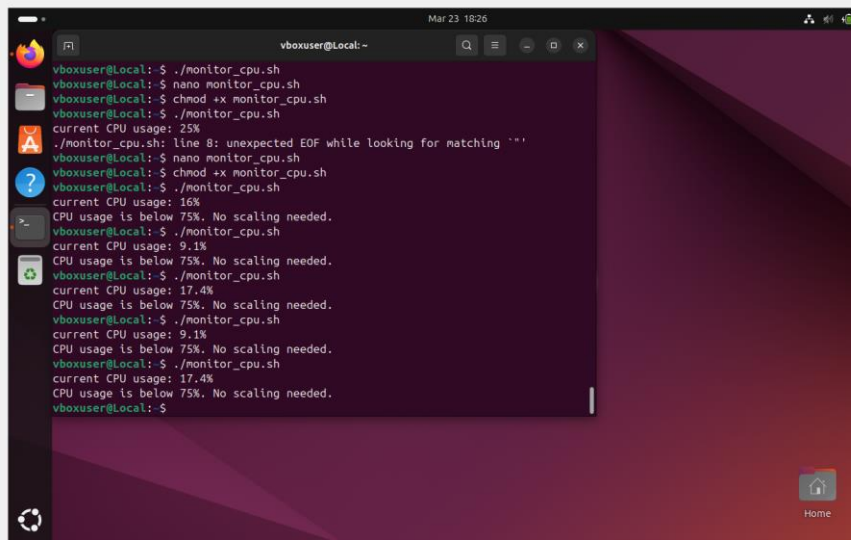
## Running the Script in GCP (Auto-scaling to GCP)

We will run the following commands:

`chmod +x monitor_cpu.sh`

`./monitor_cpu.sh` --This command will show CPU utilization.

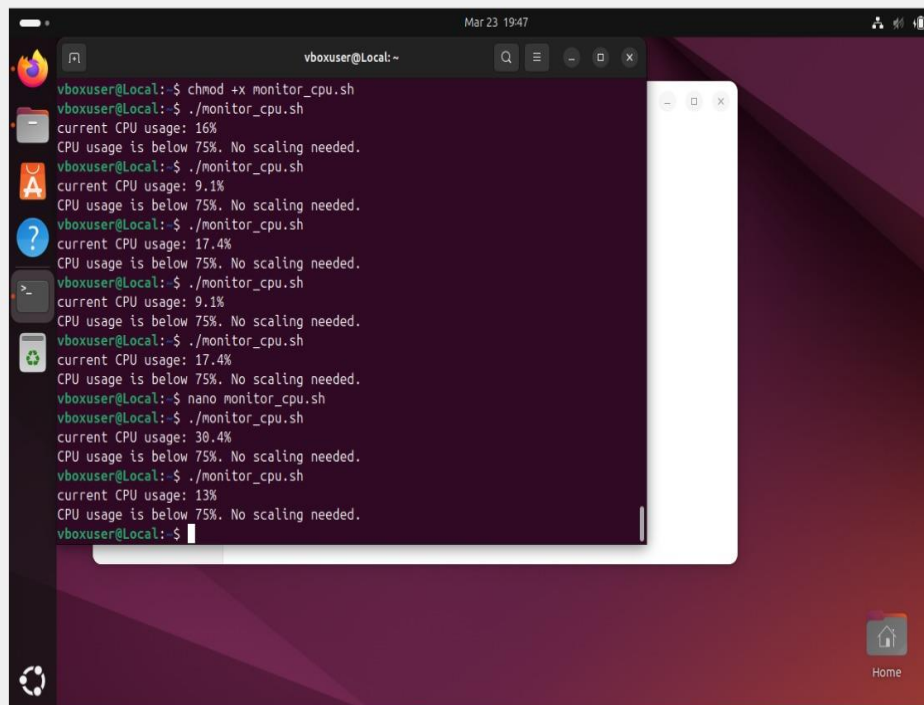




A terminal window titled 'vboxuser@Local:~' with a search bar and window controls. The terminal shows the following commands and output:

```
vboxuser@Local:~$ ./monitor_cpu.sh
vboxuser@Local:~$ nano monitor_cpu.sh
vboxuser@Local:~$ chmod +x monitor_cpu.sh
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 25%
./monitor_cpu.sh: line 8: unexpected EOF while looking for matching `''
vboxuser@Local:~$ nano monitor_cpu.sh
vboxuser@Local:~$ chmod +x monitor_cpu.sh
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 16%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 9.1%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 17.4%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 9.1%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 17.4%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$
```

The background shows a desktop environment with a purple and red gradient, a 'Home' icon, and a system clock showing 'Mar 23 18:26'.

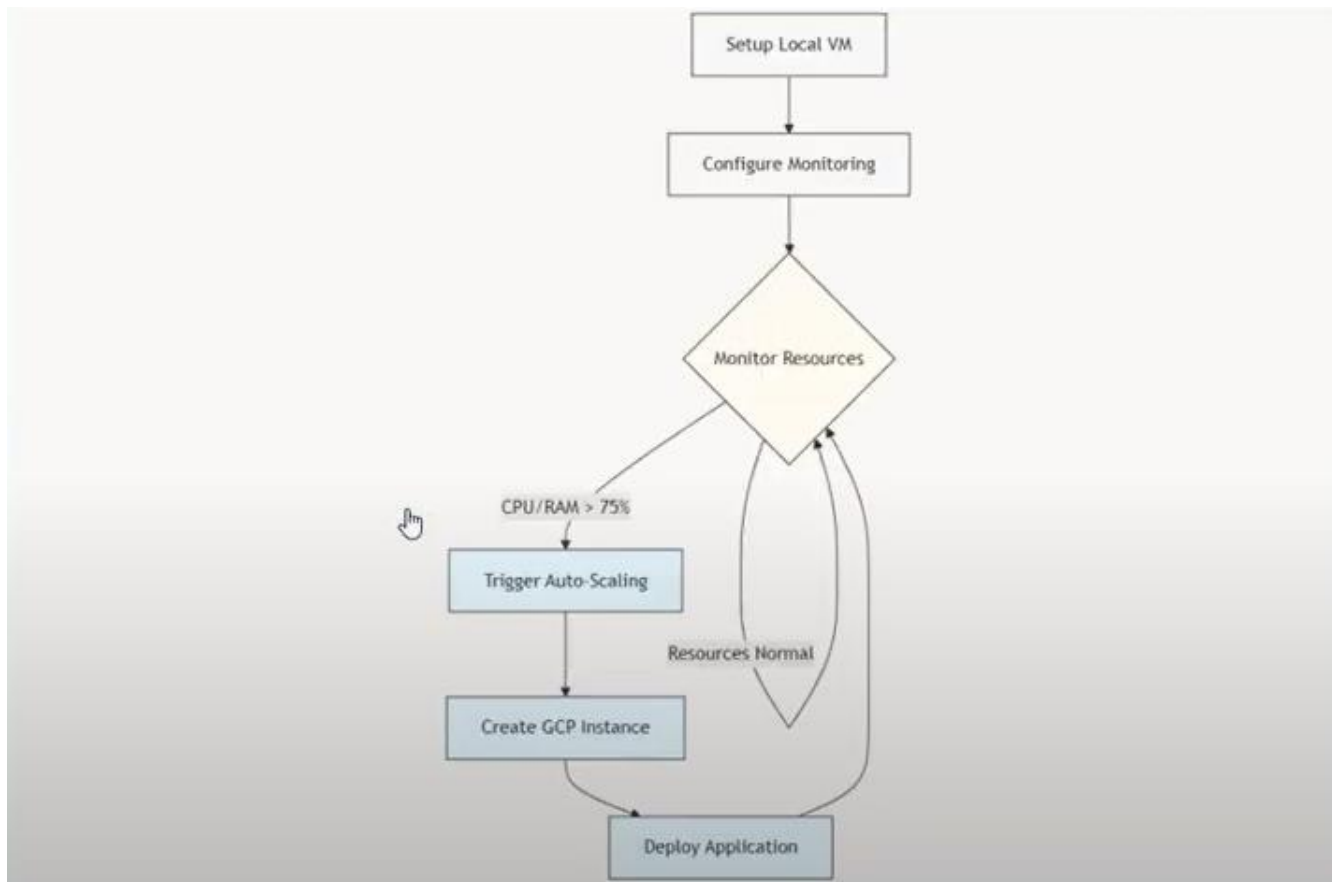


A terminal window titled 'vboxuser@Local:~' with a search bar and window controls. The terminal shows the following commands and output:

```
vboxuser@Local:~$ chmod +x monitor_cpu.sh
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 16%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 9.1%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 17.4%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 9.1%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 17.4%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ nano monitor_cpu.sh
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 30.4%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$ ./monitor_cpu.sh
current CPU usage: 13%
CPU usage is below 75%. No scaling needed.
vboxuser@Local:~$
```

The background shows a desktop environment with a purple and red gradient, a 'Home' icon, and a system clock showing 'Mar 23 19:47'.

## Architectural Diagram



## Conclusion

You have now successfully set up a local Ubuntu VM to monitor its CPU usage and trigger auto-scaling on GCP when necessary. This script can be expanded to monitor additional resources like memory, disk usage, etc., and scale other cloud resources as needed.

## Resources

Link to Source Code Repo :

[https://github.com/Sg714274/Assignment3\\_VCC](https://github.com/Sg714274/Assignment3_VCC)

Link to Recorded Video Demo :

<https://drive.google.com/file/d/1itYBLzTO5P2A6dYerQTD7VoaPTRpr0aG/view>