Automated Scaling and Resource Migration: From Local VM to Google Cloud Platform (GCP)

Step-by-Step

Imagine a local virtual machine (VM) chugging along, doing its job, but suddenly it’s working too hard—CPU usage spikes above 75%. You don’t want it to crash or slow down, how to set up an automated system that notices the overload, spins up a new VM in Google Cloud Platform (GCP), and shifts some of that workload over seamlessly.

1. Setting Up Your Local VM

First things first, you need a local VM to work with. This is like your on-premises server:

Pick a Virtualization Tool: Download and install VirtualBox

- \*\*Create the VM\*\*: seting up a new VM. used Ubuntu 22.04 LTS,Linux operating system. Give it a decent amount of resources: 2 CPUs, 4GB of RAM, and 10GB of disk space.

2. Keeping an Eye on CPU Usage

we need to watch how hard your VM is working. If its CPU usage climbs too high (above 75%). Here’s how we do it with a simple script:

We’ll use a Bash script to check CPU usage. The `top` command gives us a live look at system performance, and we’ll grab the “idle” percentage (time the CPU isn’t busy). Subtract that from 100, and you’ve got the actual usage.

```bash

CPU\_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.\*, \*\([0-9.]\*\)%\* id.\*/\1/" | awk '{print 100 - $1}')

echo "CPU Usage: $CPU\_USAGE%"

This will work when the CPU gets too busy.

3. Bringing GCP

First we need to install goggle SDK

Then we will create a project and give proper permissions.

Then we will take the servise key and activate it.

When local VM’s CPU hits over 75%We’ll automate this with the `gcloud` command-line tool (have to install it and logged in with `gcloud auth login` first).

```bash

if (( $(echo "$CPU\_USAGE > 75" | bc -l) )); then

echo "CPU usage exceeded 75%. Creating a new VM in GCP..."

gcloud compute instances create scaled-vm-$(date +%s) \

--zone=us-central1-a \

--machine-type=n1-standard-1 \

--image-family=ubuntu-2204-lts \

--image-project=ubuntu-os-cloud

fi

```

---

4. Moving the Workload to the Cloud

Now that your GCP VM is created, We’ll transfer a script and run it remotely.

- \*\*Get the IP\*\*: Find the new VM’s external IP address:

```bash

EXTERNAL\_IP=$(gcloud compute instances list --format='get(EXTERNAL\_IP)' --filter="name=(scaled-vm-\*)")

echo "New GCP VM External IP: $EXTERNAL\_IP"

```

Transfer and Run:

```bash

if [ -n "$EXTERNAL\_IP" ]; then

echo "Transferring workload to GCP VM..."

gcloud compute scp /tmp/workload\_script.sh ubuntu@$EXTERNAL\_IP:/home/ubuntu/

gcloud compute ssh ubuntu@$EXTERNAL\_IP --command "chmod +x /home/ubuntu/workload\_script.sh && /home/ubuntu/workload\_script.sh &"

fi

```

5. Testing It Out with a Sample App

`workload\_script.sh` is a simple microservice that generates CPU load or serves a webpage. Here’s how to test it:

- \*\*The Workload Script\*\*: Create `/tmp/workload\_script.sh` on local machine:

```bash

#!/bin/bash

echo "Starting workload on GCP VM..."

while true; do echo "Working hard!" && sleep 1; done # Simulates CPU activity

```

Check It’s Running:

```bash

gcloud compute ssh ubuntu@$EXTERNAL\_IP --command "ps aux | grep workload\_script.sh"

```

it’s working, spot script in the process list. also add logging (`echo "Log entry" >> /home/ubuntu/workload.log`) to track progress.

---

Conclusion

With this setup, local VM When it gets overwhelmed (CPU > 75%), a Bash script steps in, fires up a GCP VM, and shifts some of the load—all without you lifting a finger. It’s like having a smart assistant who knows exactly when to call for backup.