Create a local VM and implement a mechanism to monitor resource usage

VIRTUAL CLOUD COMPUTING

pranav sri vasthav tENALI GNANA

2025

**Subject:** Virtualization and Cloud Computing

**Assignment No:** III

**Submitted By:** Pranav Sri Vasthav Tenali Gnana

**Student Id:** G24AI1114

**e-mail:** [g24ai1114@iitj.ac.in](mailto:g24ai1114@iitj.ac.in)

**Institute:** Indian Institute of Technology Jodhpur

**Submitted To:** Assistant Professor, Dr. Sumit Karla

**Table of Contents:**

Introduction

Creating VM locally

Resource monitoring implementation

Configuration of cloud auto-scaling policies

Increase CPU utilisation

Architecture Design

Conclusion

Reference

**Introduction:**

This document provides a comprehensive guide on deploying a virtual machine (VM) locally, implementing resource monitoring using a custom script, and configuring auto-scaling policies on AWS. The guide also includes methods to simulate high CPU utilization and an overview of the infrastructure design for effective cloud scaling.

**Creating a VM Locally:**

**1. Install Virtualization Software:**

* Download and install [VirtualBox](https://www.virtualbox.org/).
* Install an operating system Ubuntu.

**2. Configure the Virtual Machine:**

* Open VirtualBox and create a new virtual machine.
* Assign the following specifications:
  + **Name**.
  + **CPU & RAM**.
  + **Storage**.
  + Install the OS and perform initial setup.

**3. Resource Monitoring Implementation:**

**Step 1: Install Required Packages**

sudo apt update && sudo apt install python3 python3-pip -y

pip install psutil boto3

**Step 2: Create a Monitoring Script**

import psutil

import time

def monitor\_resources():

while True:

cpu\_usage = psutil.cpu\_percent(interval=5)

memory\_usage = psutil.virtual\_memory().percent

print(f"CPU Usage: {cpu\_usage}% | Memory Usage: {memory\_usage}%")

time.sleep(30)

monitor\_resources()

* This script monitors **CPU and memory usage** every 30 seconds.
* To run it: python3 monitor.py

**4. Configuration of Cloud Auto-Scaling Policies:**

Set Up AWS CLI:

Install AWS CLI:

sudo snap install aws-cli

Configure AWS credentials:

aws configure

Enter:

* AWS Access Key ID
* AWS Secret Access Key
* Default region name (e.g., us-east-1)

Create an Auto-Scaling Group:

Define a launch template in AWS EC2:

aws ec2 create-launch-template --launch-template-name G24Ai1114"Auto-Scaling" --launch-template-data '{"ImageId":"ami-15816684","InstanceType":"t3.micro"}'

Create an auto-scaling group:

aws autoscaling create-auto-scaling-group --auto-scaling-group-name MyAutoScalingGroup --launch-template LaunchTemplateName= G24Ai1114 --min-size 1 --max-size 5 --desired-capacity 1 --availability-zones "us-east-1a" "us-east-1b"

Set up a scaling policy:

aws autoscaling put-scaling-policy --policy-name MyScalingPolicy --auto-scaling-group-name MyAutoScalingGroup --adjustment-type ChangeInCapacity --scaling-adjustment 1 --cooldown 60

**5. Increasing CPU Utilization:**

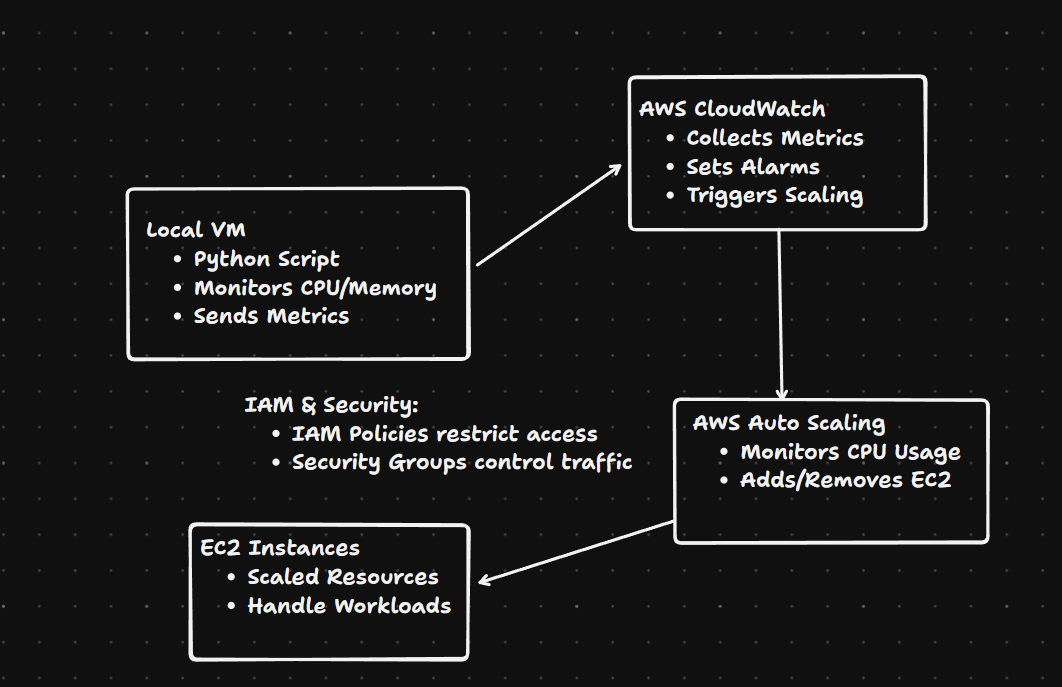
To simulate high CPU usage and trigger auto-scaling:

sudo apt install stress -y

stress --cpu 4 --timeout 60

This will overload the CPU, causing the AWS Auto Scaling Group to spin up additional instances.

**5. Architecture Design:**

****

**6. Conclusion:**

By adhering to this guide, you can monitor resource utilization using a custom script, and configure AWS auto-scaling policies for dynamic workload management. This setup ensures optimal resource allocation, preventing over-provisioning and underutilization.

**7. REFERENCES:**

* [AWS Auto Scaling Documentation](https://docs.aws.amazon.com/autoscaling/)
* [AWS EC2 User Guide](https://docs.aws.amazon.com/ec2/)
* [Python psutil Documentation](https://psutil.readthedocs.io/en/latest/)