\*\*Lab 3: Automating EC2 Instance Management with Python\*\*

\*\*Objective\*\*:

Create Python scripts to start, stop, and monitor EC2 instances.

\*\*Tasks\*\*:

1. Write a script to start/stop an EC2 instance.

2. Extend the script to monitor CPU utilization.

import boto3

import time

# Initialize the EC2 and CloudWatch clients

ec2 = boto3.client('ec2', region\_name='your\_region')

cw = boto3.client('cloudwatch', region\_name='your\_region')

# Define the instance ID of the EC2 instance to manage

instance\_id = 'your\_instance\_id'

# Function to start the EC2 instance

def start\_instance():

ec2.start\_instances(InstanceIds=[instance\_id])

print(f"Starting EC2 instance with ID: {instance\_id}")

# Function to stop the EC2 instance

def stop\_instance():

ec2.stop\_instances(InstanceIds=[instance\_id])

print(f"Stopping EC2 instance with ID: {instance\_id}")

# Function to get CPU utilization for the instance

def get\_cpu\_utilization():

# Specify the metric and dimensions for the instance

metric\_name = 'CPUUtilization'

namespace = 'AWS/EC2'

period = 300 # 5 minutes

dimensions = [{'Name': 'InstanceId', 'Value': instance\_id}]

# Get the metric data

response = cw.get\_metric\_data(

MetricDataQueries=[

{

'Id': 'm1',

'MetricStat': {

'Metric': {

'Namespace': namespace,

'MetricName': metric\_name,

'Dimensions': dimensions

},

'Period': period,

'Stat': 'Average'

},

'ReturnData': True,

},

],

StartTime=time.time() - 3600, # 1 hour ago

EndTime=time.time(),

)

if 'MetricDataResults' in response:

datapoints = response['MetricDataResults'][0]['Values']

if datapoints:

print(f"Average CPU Utilization: {datapoints[-1]}%")

else:

print("No CPU utilization data available.")

else:

print("Unable to fetch CPU utilization data.")

# Main function

def main():

while True:

action = input("Enter 'start' to start the instance, 'stop' to stop it, 'monitor' to check CPU utilization, or 'exit' to quit: ")

if action == 'start':

start\_instance()

elif action == 'stop':

stop\_instance()

elif action == 'monitor':

get\_cpu\_utilization()

elif action == 'exit':

break

else:

print("Invalid action. Please enter 'start', 'stop', 'monitor', or 'exit'.")

if \_\_name\_\_ == "\_\_main\_\_":

main()