Placement Empowerment Program

Cloud Computing and DevOps Centre

Write a Shell Script to Manage Cloud Resources: Create a script to launch, stop, and terminate cloud VMs using the CLI.

Name: Renita A Department: ECE

Introduction

Managing cloud resources efficiently is critical in today's clouddriven IT landscape. AWS Command Line Interface (CLI) provides a powerful tool for interacting with AWS services programmatically. By leveraging shell scripting, we can automate repetitive tasks like launching, stopping, and terminating virtual machines (VMs). This Proof of Concept (POC) demonstrates the use of AWS CLI integrated with a shell script to simplify VM management, showcasing automation's role in reducing manual effort and increasing productivity.

Overview

This POC focuses on creating a shell script to manage AWS EC2 instances using the AWS CLI. The script allows users to:

1. Launch new EC2 instances with pre-configured settings.
2. Stop running EC2 instances to optimize costs.
3. Terminate EC2 instances when no longer needed.
4. List currently running EC2 instances for better resource tracking.

The script uses a menu-driven approach, where users can choose specific actions, making it user-friendly and flexible. It is tested using Git Bash on Windows and adheres to AWS Free Tier limitations to ensure cost-effective implementation.

Objective

The primary objective of this POC is to:

1. Automate the management of AWS EC2 instances through shell scripting.

2. Provide an easy-to-use interface for launching, stopping, terminating, and listing instances.

3. Demonstrate the capabilities of AWS CLI and shell scripting for cloud resource management.

4. Build a foundational understanding of automation practices in cloud computing.

Importance

1. **Efficiency**: Automating cloud resource management reduces time and effort spent on manual tasks.

2. **Cost Optimization**: The ability to stop or terminate unused VMs prevents unnecessary expenses, adhering to best practices in cloud cost management.

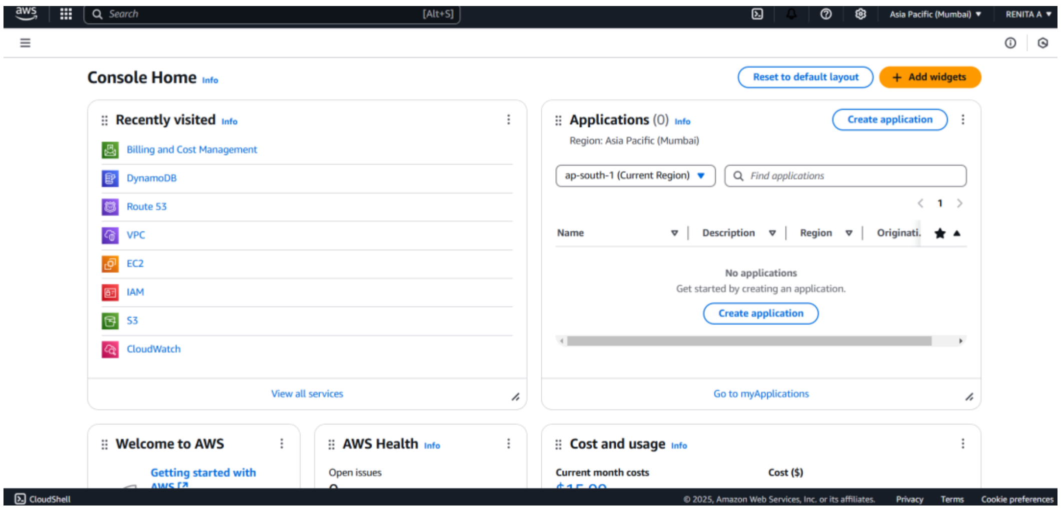
3. **Scalability**: Scripting provides a scalable solution for managing multiple resources simultaneously.

4. **Skill Development**: Enhances your technical expertise in AWS CLI, scripting, and cloud automation, which are in high demand in the IT industry.

5. Foundation for Advanced Automation: Serves as a stepping stone to more complex automation tasks, such as infrastructure as code (e.g., using tools like Terraform or CloudFormation).

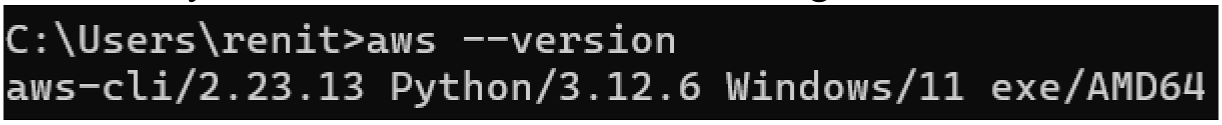
Step-by-Step Overview Step 1:

1. Go to AWS Management Console.
2. Enter your username and password to log in.



Step 2:

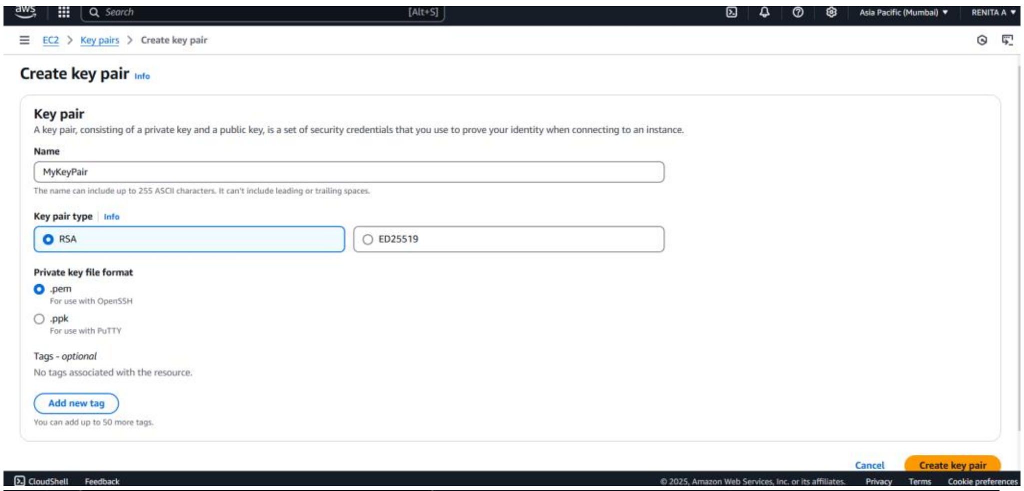
Make sure your AWS CLI is installed and configured.



```
C:\Users\renit>aws --version
aws-cli/2.23.13 Python/3.12.6 Windows/11 exe/AMD64
```

Step 3:

1. Go to the **EC2 Dashboard**.
2. In the left sidebar, click **Key Pairs** under **Network & Security**.
3. Click **Create Key Pair**.
4. Enter a name (e.g., MyKeyPair) and choose **.pem** format.
5. Download the .pem file and keep it safe—you'll need it to SSH into your instance.



Create key pair

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name: MyKeyPair

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

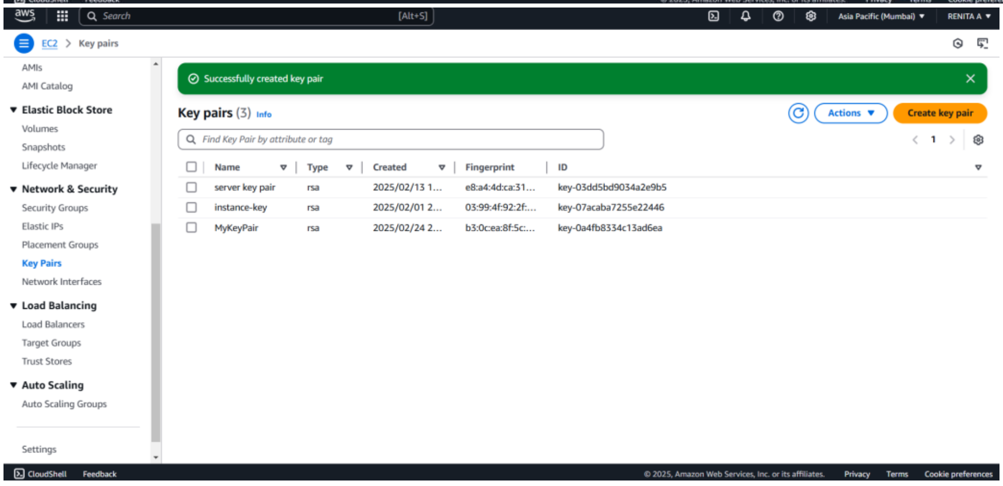
Key pair type: [✓ RSA] ED25519

Private key file format:
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

Tags - *optional*

No tags associated with the resource.

You can add up to 50 more tags.



Successfully created key pair

Key pairs (3)

Name	Type	Created	Fingerprint	ID
server key pair	rsa	2025/02/13 1...	e8:a4:4d:ca:31...	key-03dd5bd9034a2e9b5
instance-key	rsa	2025/02/01 2...	03:99:4f:92:2f:...	key-07acaba7255e22446
MyKeyPair	rsa	2025/02/24 2...	b3:0c:ea:8f:5c:...	key-0a4fb8334c13ad6ea

Step 4:

1. Go to the **AWS EC2 Dashboard**.

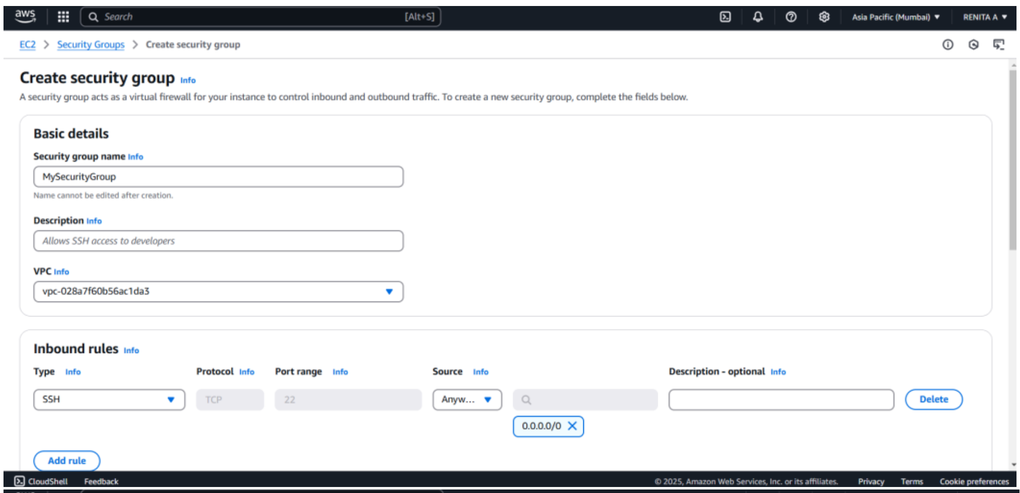
2. In the left sidebar, click **Security Groups**.

3. Click **Create Security Group**.

4. Enter a name (e.g., MySecurityGroup) and a description.

5. Add the following inbound rule:

   - **Type**: SSH
   - **Protocol**: TCP
   - **Port Range**: 22
   - **Source**: Anywhere (0.0.0.0/0) (Note the Id after created)



EC2 > Security Groups > Create security group

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name Info: MySecurityGroup

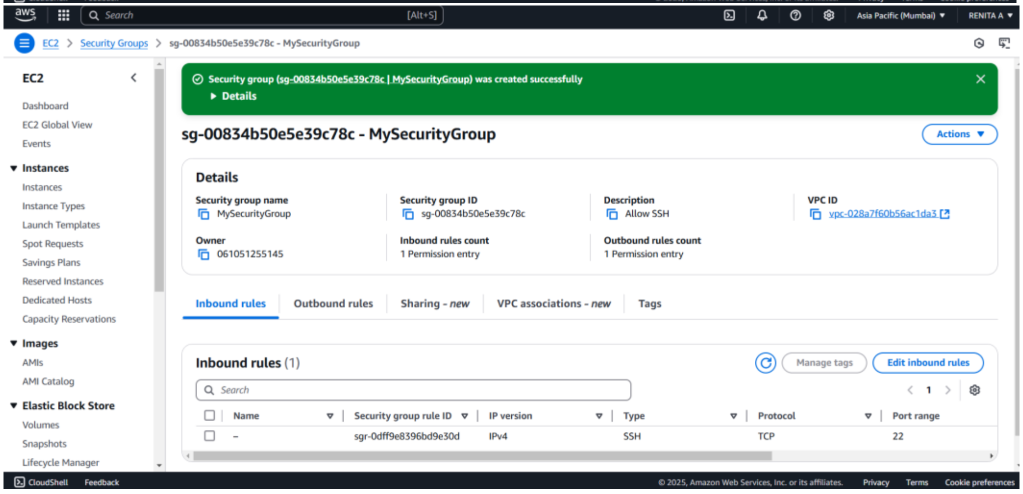
Name cannot be edited after creation.

Description Info: Allows SSH access to developers

VPC Info: vpc-028a7f60b56ac1da3

Inbound rules Info

Type	Protocol	Port range	Source		Description - optional
SSH	TCP	22	Anyw...	0.0.0.0/0	



EC2 > Security Groups > sg-00834b50e5e39c78c - MySecurityGroup

Security group (sg-00834b50e5e39c78c | MySecurityGroup) was created successfully

sg-00834b50e5e39c78c - MySecurityGroup

Details

Security group name	Security group ID	Description	VPC ID
MySecurityGroup	sg-00834b50e5e39c78c	Allow SSH	vpc-028a7f60b56ac1da3
Owner	**Inbound rules count**	**Outbound rules count**	
061051255145	1 Permission entry	1 Permission entry	

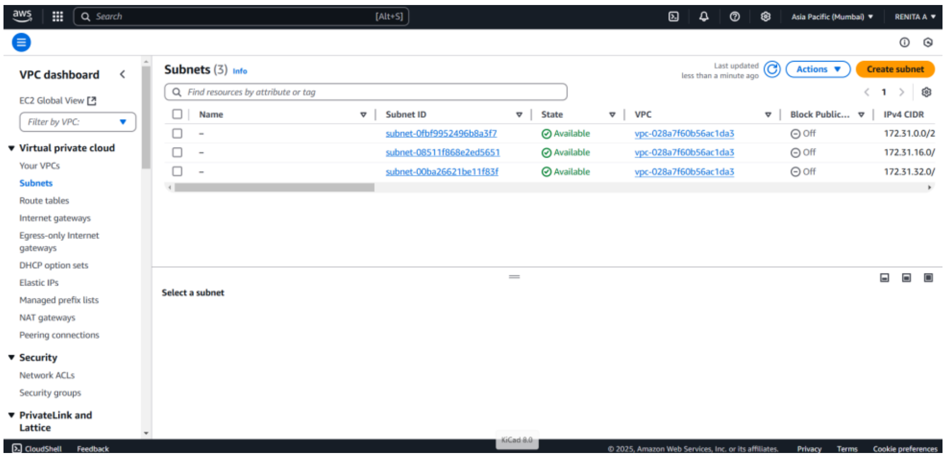
Inbound rules (1)

Name	Security group rule ID	IP version	Type	Protocol	Port range
–	sgr-0dff9e8396bd9e30d	IPv4	SSH	TCP	22

Step 5:

1. In the **VPC Dashboard**, click **Subnets** in the left sidebar.

2. Note the **Subnet ID** of one of your subnets. Example: subnet0abcd1234.



Subnets (3) Info

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
–	subnet-0fbf9952496b8a3f7	Available	vpc-028a7f60b56ac1da3	Off	172.31.0.0/2
–	subnet-08511f868e2ed5651	Available	vpc-028a7f60b56ac1da3	Off	172.31.16.0/
–	subnet-00ba26621be11f83f	Available	vpc-028a7f60b56ac1da3	Off	172.31.32.0/

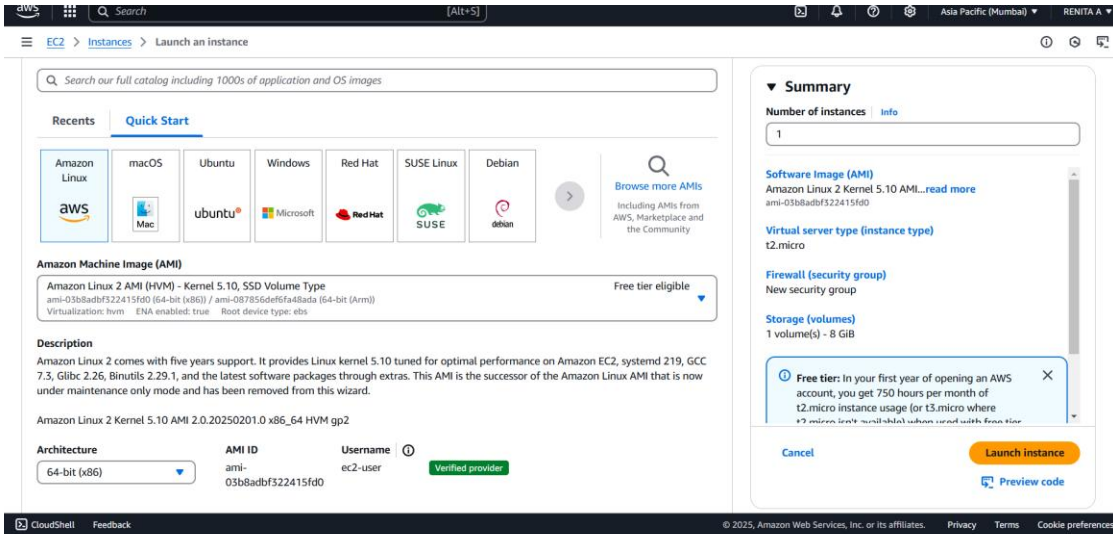
Select a subnet

Step 6:

1. In the **AWS EC2 Dashboard**, click **Launch Instance**.

2. Search for "Amazon Linux 2" and select it.

3. Note the **AMI ID** (e.g., ami-0c02fb55956c7d316).



Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type — Free tier eligible
ami-03b8adbf322415fd0 (64-bit (x86)) / ami-087856def6fa48ada (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2 comes with five years support. It provides Linux kernel 5.10 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is now under maintenance only mode and has been removed from this wizard.

Amazon Linux 2 Kernel 5.10 AMI 2.0.20250201.0 x86_64 HVM gp2

Architecture: 64-bit (x86)
AMI ID: ami-03b8adbf322415fd0
Username: ec2-user — Verified provider

Summary

Number of instances: 1

Software Image (AMI): Amazon Linux 2 Kernel 5.10 AMI...read more
ami-03b8adbf322415fd0

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Free tier: In your first year of opening an AWS account, you get 750 hours per month of t2.micro instance usage (or t3.micro where

Step 7:

Here's a simple shell script to manage cloud resources (launch, stop, and terminate VMs) using the AWS CLI.

Open **Notepad.**

Paste the script into the Notepad.

Replace the placeholders (YourKeyPairName, YourSecurityGroupID, etc.) with your actual values:

- **Key Pair Name**: Replace with the name of your key pair.
- **Security Group ID**: Replace with your security group ID.
- **Subnet ID**: Replace with your subnet ID. ▯ **AMI ID**: Replace with the AMI ID.

```bash
#!/bin/bash

# A simple shell script to manage AWS EC2 instances.
# Prerequisites: AWS CLI must be installed and configured.

function launch_instance() {
    echo "Launching a new EC2 instance..."
    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id ami-03b8adbf322415fd0 \
        --instance-type t2.micro \
        --key-name MyKeyPair \
        --security-group-ids sg-00834b50e5e39c78c \
        --subnet-id subnet-00ba26621be11f83f \
        --query 'Instances[0].InstanceId' \
        --output text)

    echo "Instance launched successfully! Instance ID: $INSTANCE_ID"
}

function stop_instance() {
    echo "Enter the Instance ID to stop:"
    read INSTANCE_ID
    aws ec2 stop-instances --instance-ids $INSTANCE_ID
    echo "Instance $INSTANCE_ID has been stopped."
}

function terminate_instance() {
    echo "Enter the Instance ID to terminate:"
    read INSTANCE_ID
    aws ec2 terminate-instances --instance-ids $INSTANCE_ID
    echo "Instance $INSTANCE_ID has been terminated."
}
```

```
function display_menu() {
  echo "\nCloud Resource Management Script"
  echo "1. Launch a new EC2 instance"
  echo "2. Stop an existing EC2 instance"
  echo "3. Terminate an EC2 instance"
  echo "4. Exit"
  echo "Choose an option:"
}

while true; do
  display_menu
  read OPTION

  case $OPTION in
    1)
      launch_instance
      ;;
    2)
      stop_instance
      ;;
    3)
      terminate_instance
      ;;
    4)
      echo "Exiting the script. Goodbye!"
      exit 0
      ;;
    *)
      echo "Invalid option. Please try again."
      ;;
  esac
done
```

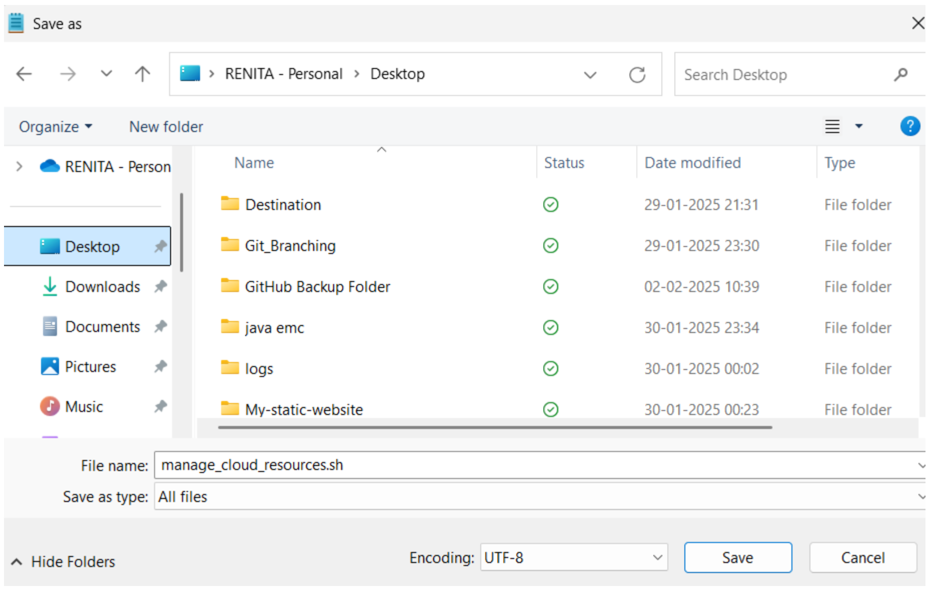
Step 8:

1. Click **File** → **Save As**.

2. In the **Save As** window:

- **File Name**: Enter manage_cloud_resources.sh.
- **Save as type**: Select **All Files** from the dropdown.
- **Encoding**: Select **UTF-8** (if available). ▯ **Location**: Save it in Desktop.

Important: Make sure the file has the .sh



Name	Status	Date modified	Type
Destination	✓	29-01-2025 21:31	File folder
Git_Branching	✓	29-01-2025 23:30	File folder
GitHub Backup Folder	✓	02-02-2025 10:39	File folder
java emc	✓	30-01-2025 23:34	File folder
logs	✓	30-01-2025 00:02	File folder
My-static-website	✓	30-01-2025 00:23	File folder

File name: manage_cloud_resources.sh

Save as type: All files

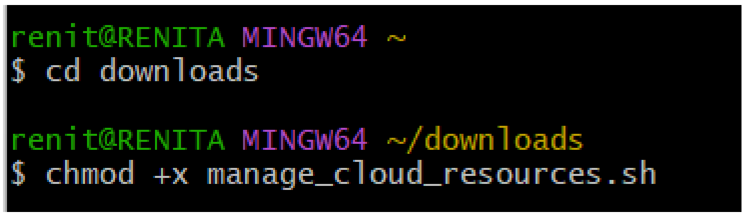
Encoding: UTF-8

ch

Step 9:

1. Open Git Bash
2. Run the following command in Git Bash:

chmod +x manage_cloud_resources.sh

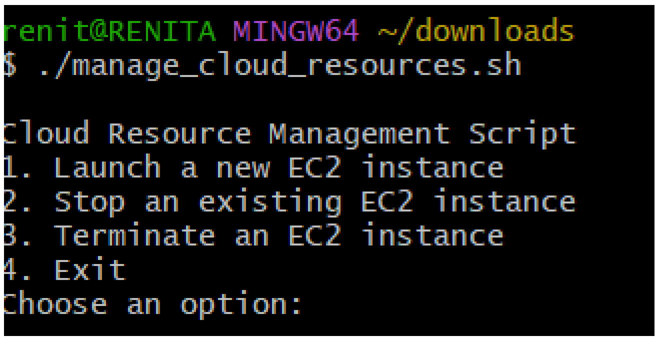


```
renit@RENITA MINGW64 ~
$ cd downloads

renit@RENITA MINGW64 ~/downloads
$ chmod +x manage_cloud_resources.sh
```

Step 10:

Run the script using:

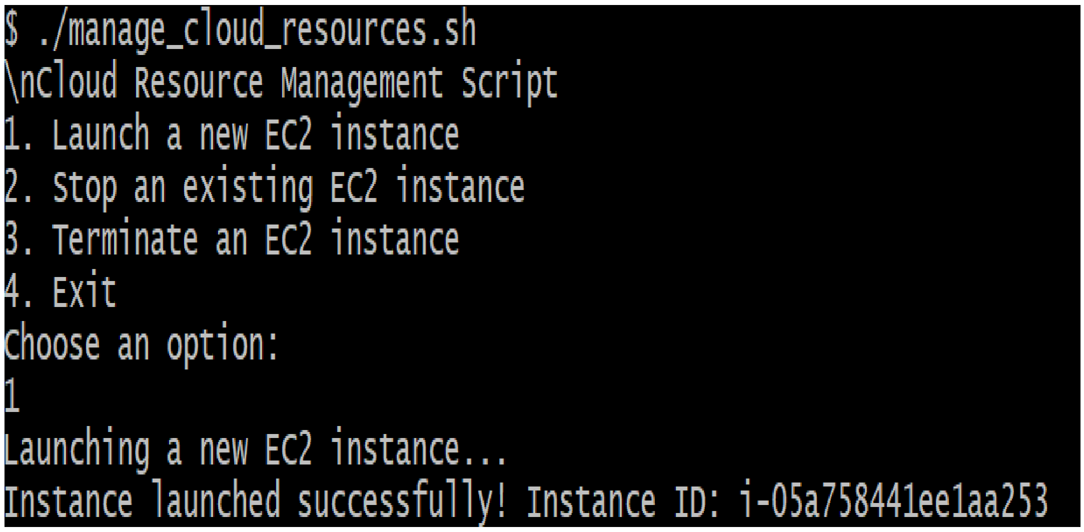
./manage_cloud_resources.sh



```
renit@RENITA MINGW64 ~/downloads
$ ./manage_cloud_resources.sh

Cloud Resource Management Script
1. Launch a new EC2 instance
2. Stop an existing EC2 instance
3. Terminate an EC2 instance
4. Exit
Choose an option:
```

Step 10:

1. Select 1 to launch an instance.
2. The script will create an EC2 instance and display its **Instance ID**. Make a note of this ID for the next steps.



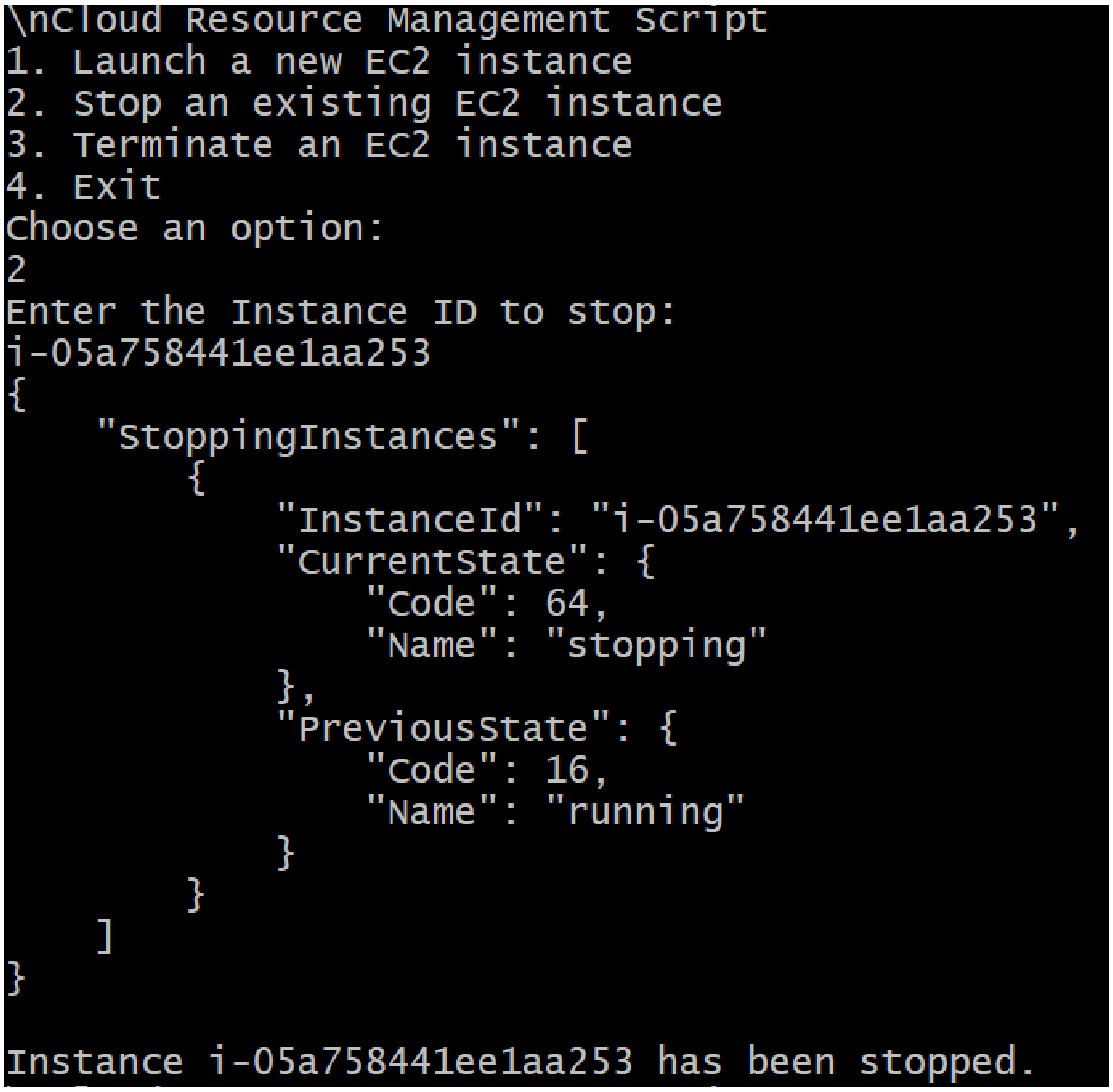
```
$ ./manage_cloud_resources.sh
\nCloud Resource Management Script
1. Launch a new EC2 instance
2. Stop an existing EC2 instance
3. Terminate an EC2 instance
4. Exit
Choose an option:
1
Launching a new EC2 instance...
Instance launched successfully! Instance ID: i-05a758441ee1aa253
```

Step 11:

1. Select 2 to stop an instance.

2. Enter the **Instance ID** of the instance you launched earlier.

3. The script will stop the instance.



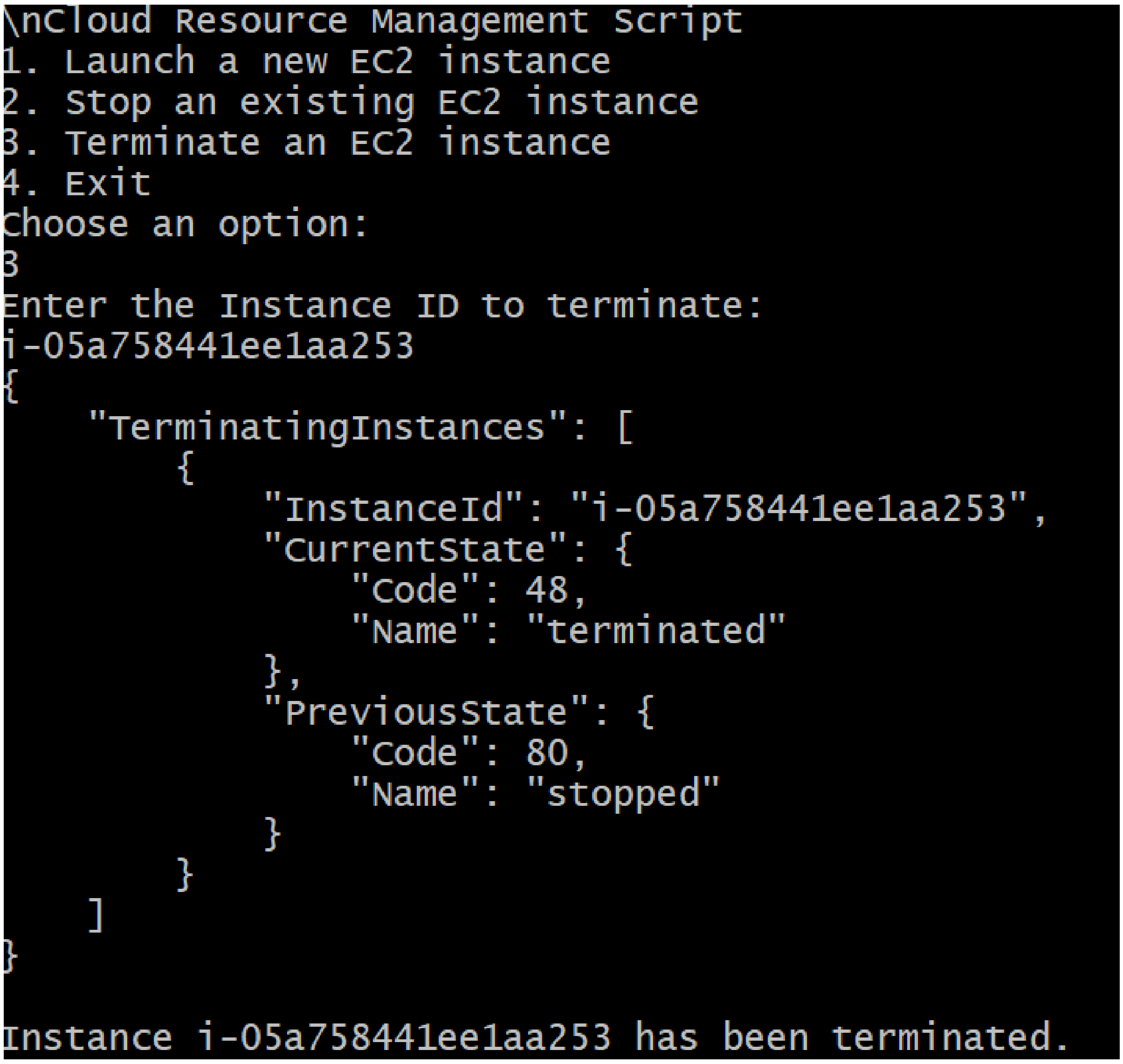
```
\nCloud Resource Management Script
1. Launch a new EC2 instance
2. Stop an existing EC2 instance
3. Terminate an EC2 instance
4. Exit
Choose an option:
2
Enter the Instance ID to stop:
i-05a758441ee1aa253
{
    "StoppingInstances": [
        {
            "InstanceId": "i-05a758441ee1aa253",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

Instance i-05a758441ee1aa253 has been stopped.
```

1. Select 3 to terminate an instance.

Step 12:

2. Enter the **Instance ID** of the instance you launched earlier.

3. The script will terminate the instance.



```
\nCloud Resource Management Script
1. Launch a new EC2 instance
2. Stop an existing EC2 instance
3. Terminate an EC2 instance
4. Exit
Choose an option:
3
Enter the Instance ID to terminate:
i-05a758441ee1aa253
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-05a758441ee1aa253",
            "CurrentState": {
                "Code": 48,
                "Name": "terminated"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}

Instance i-05a758441ee1aa253 has been terminated.
```

Successfully completed the PoC!

Outcome

By completing this POC on managing AWS cloud resources using the CLI and a shell script, you will:

1. Automate essential EC2 instance management tasks, including launching, stopping, and terminating VMs, through a menu-driven shell script.

2. Efficiently manage multiple EC2 instances using AWS CLI commands integrated with shell scripting, ensuring scalability and consistency.

3. Gain hands-on experience with AWS CLI for interacting with cloud resources programmatically, building your foundation for advanced automation.

4. Enhance your skills in shell scripting and cloud resource management, critical for DevOps and cloud engineering roles.

5. Understand key AWS services like EC2, IAM (for key pairs), and security groups, along with best practices in cloud cost optimization.

6. Validate the practical implementation of a script by successfully launching, stopping, and terminating multiple EC2 instances.