

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: Hema S

Department: ECE

Introduction

Managing cloud resources efficiently is critical in today's cloud-driven 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.

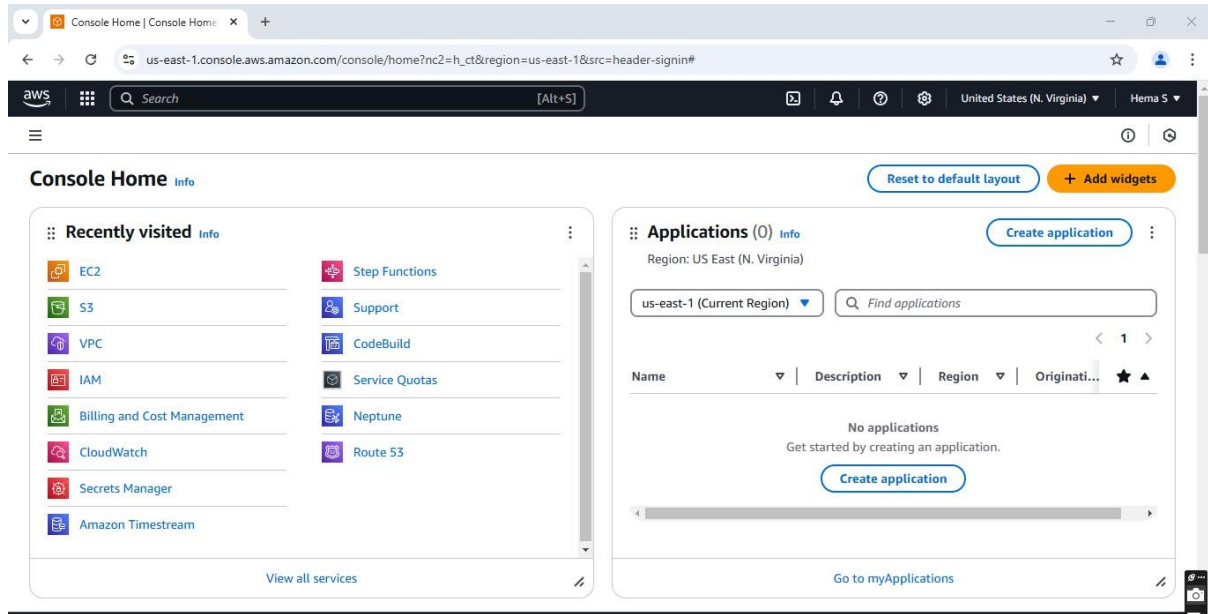
Importance

- 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.



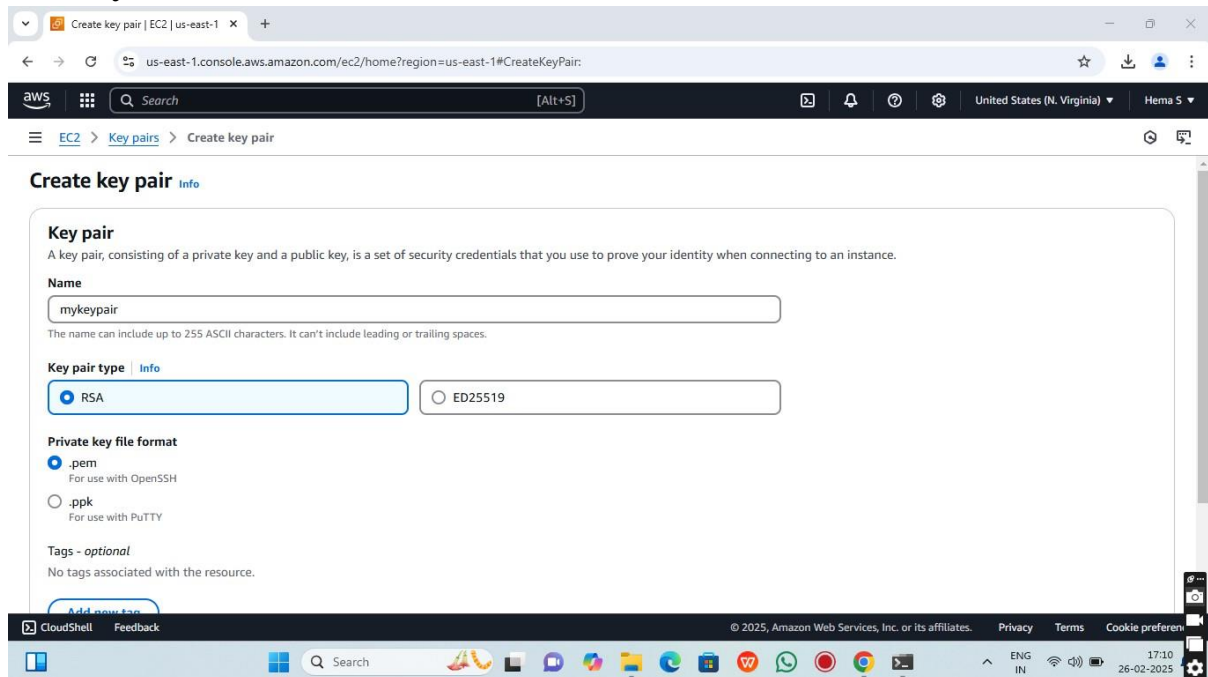
Step 2:

Make sure your AWS CLI is installed and configured.

```
C:\Users\sppra>aws --version
aws-cli/2.24.10 Python/3.12.9 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.



Key pairs | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#KeyPairs:

Search

[Alt+S]

United States (N. Virginia)

Hema S

EC2 > Key pairs

Snapshots

Lifecycle Manager

▼ Network & Security

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

▼ Load Balancing

Load Balancers

Target Groups

Trust Stores

▼ Auto Scaling

Auto Scaling Groups

Settings

Successfully created key pair

Key pairs (7/1) Info

Find Key Pair by attribute or tag

< 1 >

	Name	Type	Created	Fingerprint	ID
<input type="checkbox"/>	mykeypair	rsa	2025/02/26 17:09 GMT+5:30	e4:77:55:2b:aa:11:a7:89:75:ba:39:e0:b...	key-0

Create key pair

CloudShell

Feedback

© 2025 Amazon Web Services, Inc. or its affiliates

Privacy

Terms

Cookie preferences

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)

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](#)

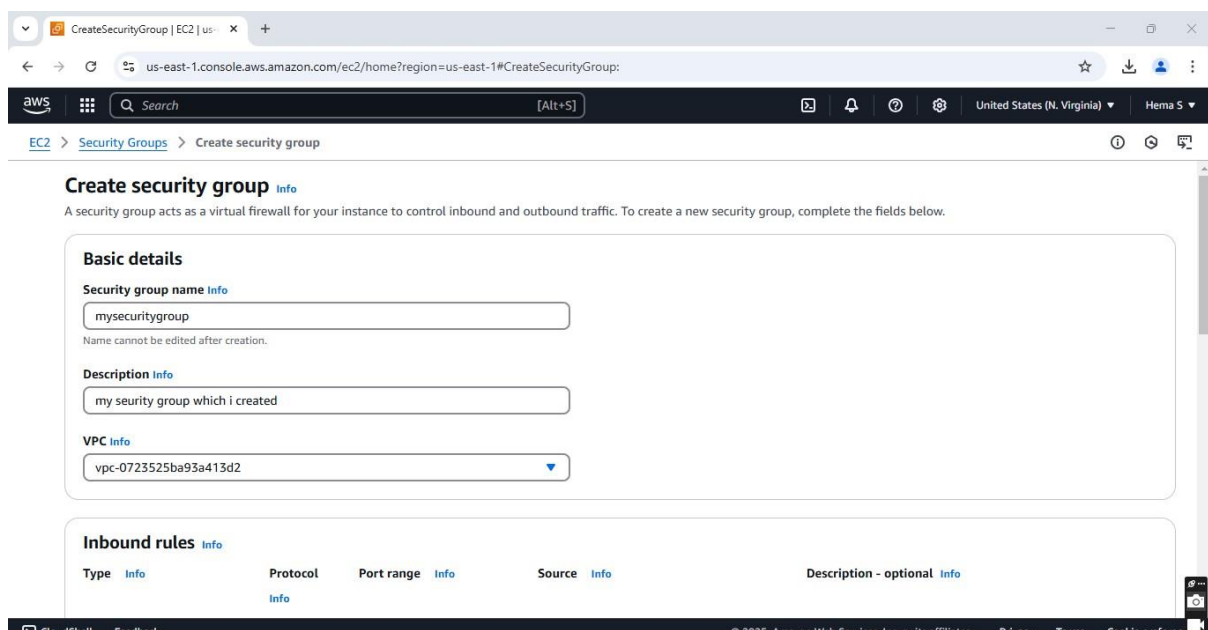
Name cannot be edited after creation.

Description [Info](#)

VPC [Info](#)

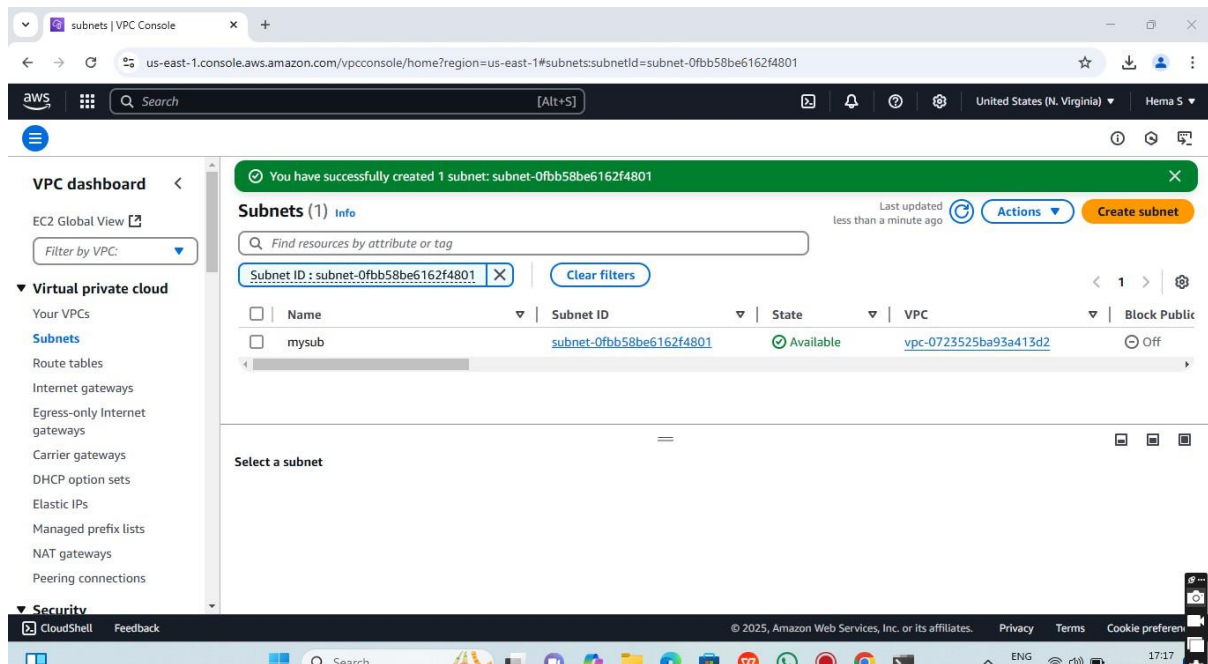
Inbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
---------------------------	-------------------------------	---------------------------------	-----------------------------	---



Step 5:

1. In the **AWS VPC Dashboard**, click **Subnets** in the left sidebar.
2. Note the **Subnet ID** of one of your subnets. Example: subnet-0abcd1234.



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).

Launch an instance | EC2 | us-east-1

us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:

Search [Alt+S]

United States (N. Virginia) Hema S

EC2 > Instances > Launch an instance

Recents Quick Start

Amazon Linux

macOS

Ubuntu

Windows

Red Hat

SUSE Linux

Debian

Browse more AMIs

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI
ami-05b10e08d247fb927 (64-bit (x86), uefi-preferred) / ami-0f37c4a1ba152af46 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.6.20250218.2 x86_64 HVM kernel-6.1

Architecture

64-bit (x86)

Boot mode

uefi-preferred

AMI ID

ami-05b10e08d247fb9

Username

ec2-user

Verified provider

Summary

Number of instances [Info](#)

1

Software Image (AMI)

Amazon Linux 2023 AMI 2023.6.2...[read more](#)
ami-05b10e08d247fb927

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

Storage (volumes)

1 volume(s) - 8 GiB

Cancel Launch instance

Preview code

CloudShell Feedback

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Step 7:

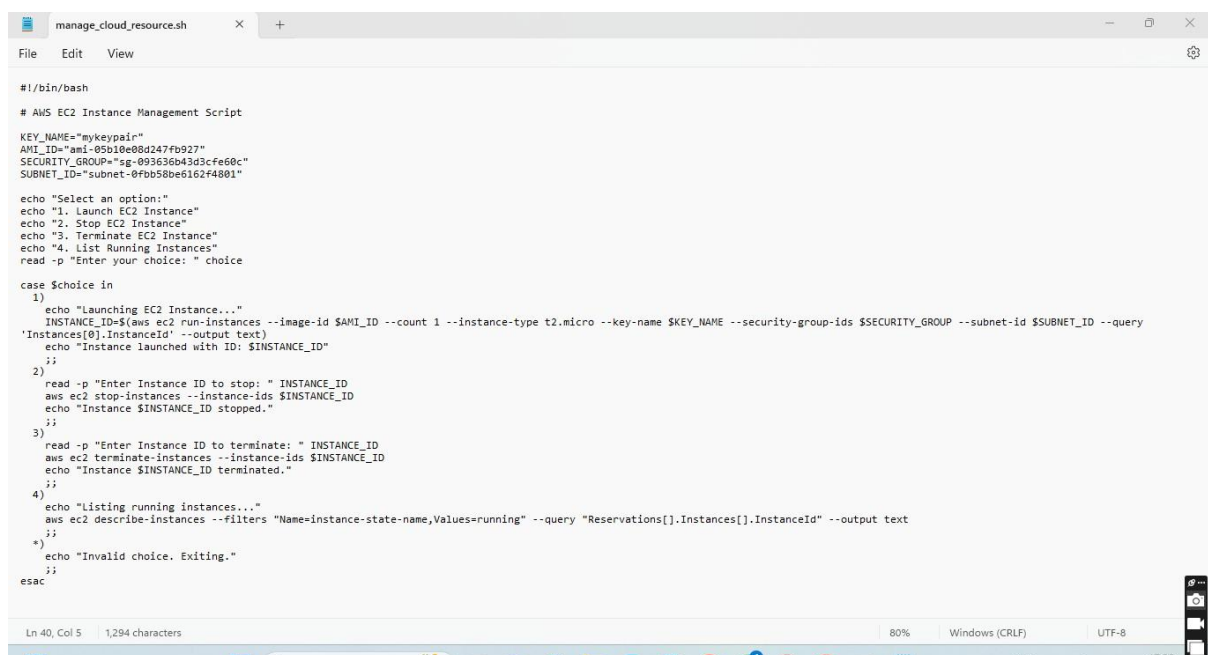
Here's a simple shell script to manage cloud resources (launch, stop, and terminate VMs) using the AWS CLI.

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.



```
manage_cloud_resource.sh

#!/bin/bash

# AWS EC2 Instance Management Script

KEY_NAME="mykeypair"
AMI_ID="ami-05b10e08d247fb927"
SECURITY_GROUP="sg-093636b43d3cfe60c"
SUBNET_ID="subnet-0f0b58be6162f4801"

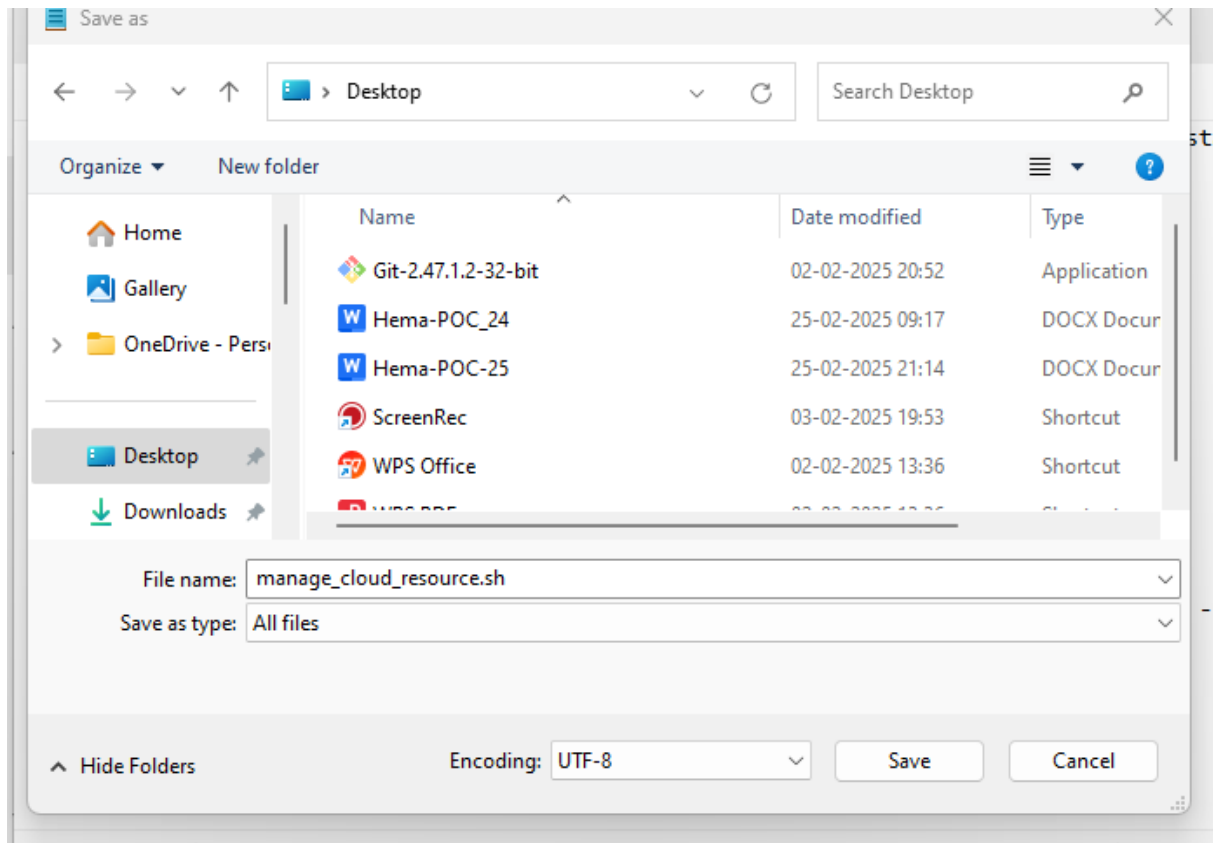
echo "Select an option:"
echo "1. Launch EC2 Instance"
echo "2. Stop EC2 Instance"
echo "3. Terminate EC2 Instance"
echo "4. List Running Instances"
read -p "Enter your choice: " choice

case $choice in
    1)
        echo "Launching EC2 Instance..."
        INSTANCE_ID=$(aws ec2 run-instances --image-id $AMI_ID --count 1 --instance-type t2.micro --key-name $KEY_NAME --security-group-ids $SECURITY_GROUP --subnet-id $SUBNET_ID --query 'Instances[0].InstanceId' --output text)
        echo "Instance launched with ID: $INSTANCE_ID"
    ;;
    2)
        read -p "Enter Instance ID to stop: " INSTANCE_ID
        aws ec2 stop-instances --instance-ids $INSTANCE_ID
        echo "Instance $INSTANCE_ID stopped."
    ;;
    3)
        read -p "Enter Instance ID to terminate: " INSTANCE_ID
        aws ec2 terminate-instances --instance-ids $INSTANCE_ID
        echo "Instance $INSTANCE_ID terminated."
    ;;
    4)
        echo "Listing running instances..."
        aws ec2 describe-instances --filters "Name=instance-state-name,Values=running" --query "Reservations[].Instances[].InstanceId" --output text
    ;;
    *)
        echo "Invalid choice. Exiting."
    ;;
esac
```

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`



Step 9:

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

chmod +x manage_cloud_resources.sh

```
MINGW32/c/Users/sppra/desktop
sppra@DESKTOP-S8GOFLP MINGW32 ~ (master)
$ cd desktop

sppra@DESKTOP-S8GOFLP MINGW32 ~/desktop (master)
$ chmod +x manage_cloud_resource.sh
```

Step 10:

Run the script using:

./manage_cloud_resources.sh

```
sppra@DESKTOP-S8G0FLP MINGW32 ~/desktop (master)
$ ./manage_cloud_resource.sh
Select an option:
1. Launch EC2 Instance
2. Stop EC2 Instance
3. Terminate EC2 Instance
4. List Running Instances
Enter your choice: |
```

Step 11:

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.

```
sppra@DESKTOP-S8G0FLP MINGW32 ~/desktop (master)
$ ./manage_cloud_resource.sh
Select an option:
1. Launch EC2 Instance
2. Stop EC2 Instance
3. Terminate EC2 Instance
4. List Running Instances
Enter your choice: 1
Launching EC2 Instance...
Instance launched with ID: i-0fda7a05e48314814
```

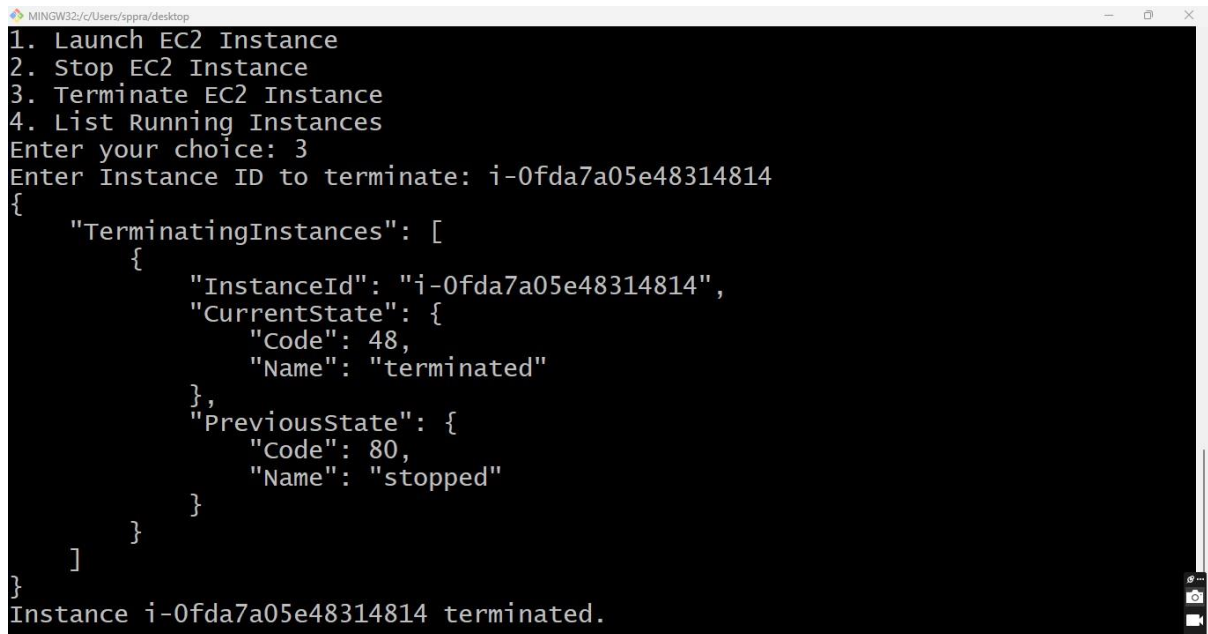
Step 12:

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.

```
MINGW32/c/Users/sppra/desktop
1. Launch EC2 Instance
2. Stop EC2 Instance
3. Terminate EC2 Instance
4. List Running Instances
Enter your choice: 2
Enter Instance ID to stop: i-0fda7a05e48314814
{
  "StoppingInstances": [
    {
      "InstanceId": "i-0fda7a05e48314814",
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
Instance i-0fda7a05e48314814 stopped.
```

Step 13:

1. Select 3 to terminate an instance.
2. Enter the **Instance ID** of the instance you launched earlier.
3. The script will terminate the instance.

A screenshot of a terminal window titled 'MINGW32/c:/Users/sppra/desktop'. The terminal displays a script with four numbered options: 1. Launch EC2 Instance, 2. Stop EC2 Instance, 3. Terminate EC2 Instance, and 4. List Running Instances. The user has entered '3' as their choice. They then enter the instance ID 'i-0fda7a05e48314814'. The script outputs a JSON object showing the instance's state transition from 'stopped' to 'terminated'. The JSON includes the instance ID, current state (code 48, name 'terminated'), and previous state (code 80, name 'stopped'). Finally, the script prints 'Instance i-0fda7a05e48314814 terminated.'

```
1. Launch EC2 Instance
2. Stop EC2 Instance
3. Terminate EC2 Instance
4. List Running Instances
Enter your choice: 3
Enter Instance ID to terminate: i-0fda7a05e48314814
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-0fda7a05e48314814",
      "CurrentState": {
        "Code": 48,
        "Name": "terminated"
      },
      "PreviousState": {
        "Code": 80,
        "Name": "stopped"
      }
    }
  ]
}
Instance i-0fda7a05e48314814 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.