CLOUD COMPUTING LAB EXAM

Name: Seerat Fatima

Reg. No: 2023-BSE-060

Submitted To: Engr. Shoaib

Q1 – AWS IAM Setup Using AWS CLI and Console Verification (10 marks)

Create IAM group `SoftwareEngineering` using AWS CLI

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam create-group --group-name SoftwareEngineering
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3QUYFZVYKHETOBLPO",
        "Arn": "arn:aws:iam::791666871664:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:32:14+00:00"
    }
}
```

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam get-group --group-name SoftwareEngineering
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3QUYFZVYKHETOBLPO",
        "Arn": "arn:aws:iam::791666871664:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:32:14+00:00"
    }
}
```

Create IAM user (your name) and view details

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam create-user --user-name SeeratFatima
{
    "User": {
        "Path": "/",
        "UserName": "SeeratFatima",
        "UserId": "AIDA3QUYFZVYNLYLYTHR3",
        "Arn": "arn:aws:iam::791666871664:user/SeeratFatima",
        "CreateDate": "2026-01-19T07:35:30+00:00"
    }
}
```

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam get-user --user-name SeeratFatima
{
    "User": {
        "Path": "/",
        "UserName": "SeeratFatima",
        "UserId": "AIDA3QUYFZVYNLYLYTHR3",
        "Arn": "arn:aws:iam::791666871664:user/SeeratFatima",
        "CreateDate": "2026-01-19T07:35:30+00:00"
    }
}
```

Add the IAM user to the `SoftwareEngineering` group

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam add-user-to-group \
  --user-name SeeratFatima \
  --group-name SoftwareEngineering
```

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam get-group --group-name SoftwareEngineering
{
    "Users": [
        {
            "Path": "/",
            "UserName": "SeeratFatima",
            "UserId": "AIDA3QUYFZVYNLYLYTHR3",
            "Arn": "arn:aws:iam::791666871664:user/SeeratFatima",
            "CreateDate": "2026-01-19T07:35:30+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3QUYFZVYKHETOBLPO",
        "Arn": "arn:aws:iam::791666871664:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:32:14+00:00"
    }
}
```

Attach `AdministratorAccess` managed policy to the `SoftwareEngineering` group

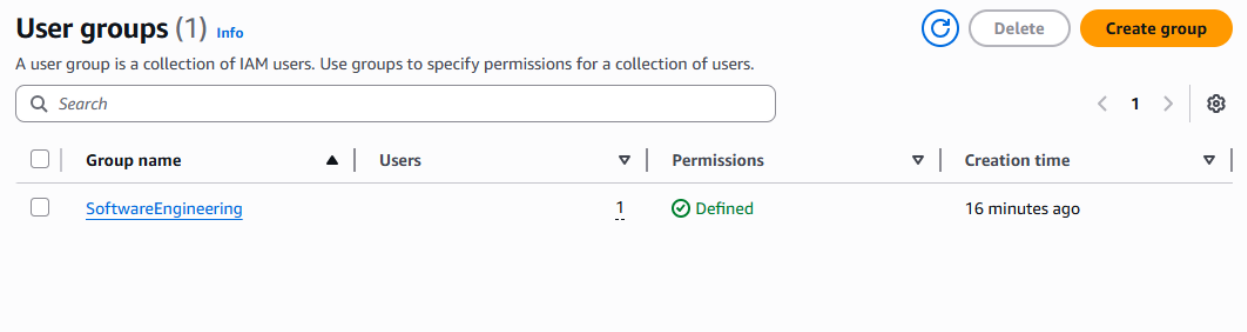
```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam list-policies --scope AWS --query "Policies[?PolicyName=='AdministratorAccess']
"
[
    {
        "PolicyName": "AdministratorAccess",
        "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
        "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2015-02-06T18:39:46+00:00",
        "UpdateDate": "2015-02-06T18:39:46+00:00"
    }
]
```

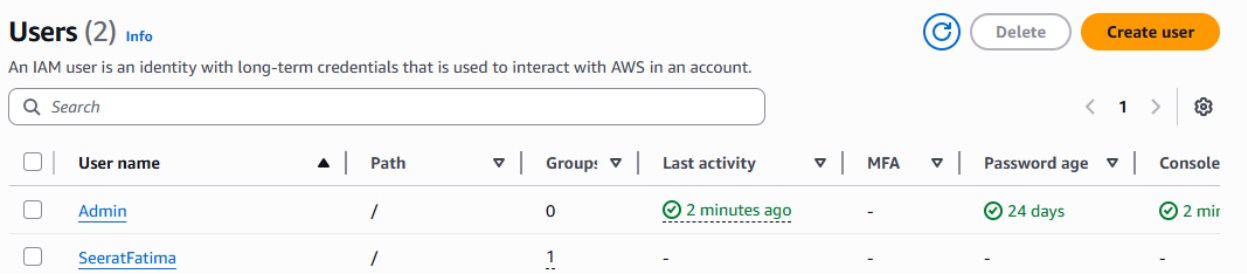
```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam attach-group-policy \
    --group-name SoftwareEngineering \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

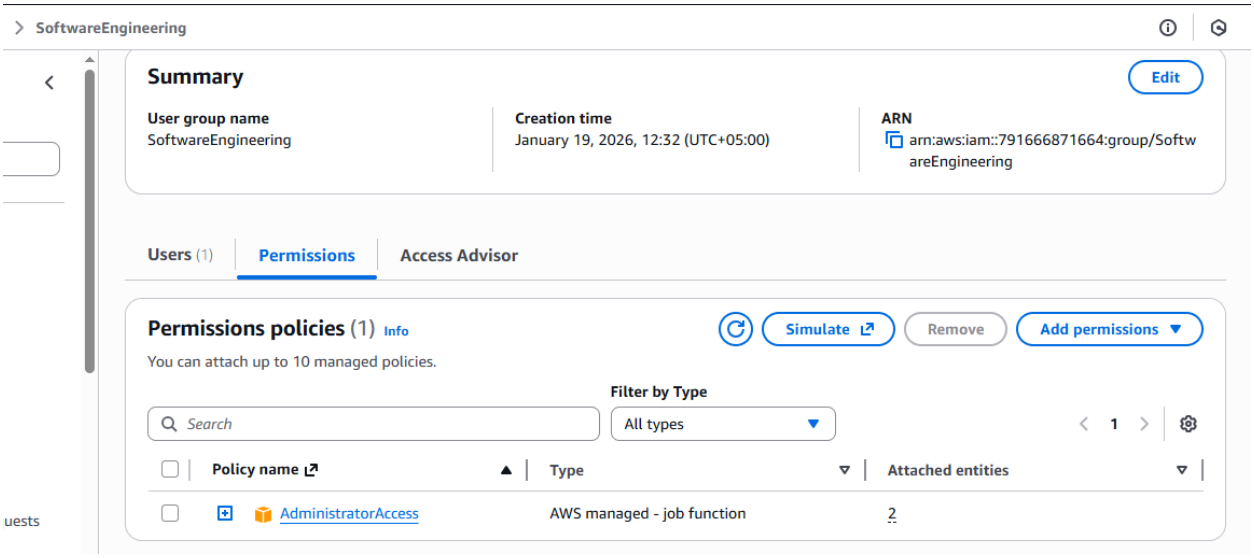
List attached policies of the `SoftwareEngineering` group

```
@SeratFatima00 ➜/workspaces/Lab_Exam (main) $ aws iam list-attached-group-policies --group-name SoftwareEngineering
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
```

Verify IAM configuration in AWS Management Console







Q2 – Terraform Lab: Simple AWS Environment with Nginx over HTTPS (30 marks)

Configure the AWS provider

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region  = "us-east-1"
  profile = "default"
}
```

Define input variables

Create VPC and subnet

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block

  tags = {
    Name = "${var.env_prefix}-vpc"
  }
}

resource "aws_subnet" "myapp_subnet" {
  vpc_id            = aws_vpc.myapp_vpc.id
  cidr_block        = var.subnet_cidr_block
  availability_zone = var.availability_zone

  tags = {
    Name = "${var.env_prefix}-subnet-1"
  }
}
```

Create Internet Gateway and configure default route table

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = aws_vpc.myapp_vpc.id

  tags = {
    Name = "${var.env_prefix}-igw"
  }
}

resource "aws_default_route_table" "myapp_default_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }

  tags = {
    Name = "${var.env_prefix}-rt"
  }
}
```

Discover public IP and compute `/32` CIDR using data + locals

Configure the default security group in the VPC

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
resource "aws_default_security_group" "myapp_default_sg" {
  vpc_id = aws_vpc.myapp_vpc.id

  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [locals.my_ip]
  }

  ingress {
    description = "HTTP from anywhere"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "HTTPS from anywhere"
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    description = "Allow all outbound traffic"
    from_port   = 0
    to_port     = 0
```

Create an AWS key pair for SSH

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
resource "aws_key_pair" "server_key" {
  key_name   = "serverkey"
  public_key = file("~/.ssh/id_ed25519.pub")
}
~
```

Create the EC2 instance resource

```
resource "aws_instance" "myapp_ec2" {
  ami                    = "ami-0a1b2c3d4e5f67890" # Replace with actual Amazon Linux 2023 AMI in eu-north-1
  instance_type          = var.instance_type
  subnet_id              = aws_subnet.myapp_subnet.id
  vpc_security_group_ids = [aws_default_security_group.myapp_default_sg.id]
  availability_zone      = var.availability_zone
  associate_public_ip_address = true
  key_name               = aws_key_pair.server_key.key_name
  user_data              = file("entry-script.sh")

  tags = {
    Name = "${var.env_prefix}-ec2-instance"
  }
}
```

Create `entry-script.sh` to configure Nginx + HTTPS

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
#!/bin/bash
# entry-script.sh
# This script installs Nginx, sets up HTTPS, and serves a custom page

# Update package index
sudo dnf update -y

# Install Nginx
sudo dnf install -y nginx

# Create directory for self-signed certificate
sudo mkdir -p /etc/nginx/ssl

# Generate self-signed TLS certificate (valid for 365 days)
sudo openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -keyout /etc/nginx/ssl/selfsigned.key \
  -out /etc/nginx/ssl/selfsigned.crt \
  -subj "/C=US/ST=State/L=City/O=Organization/OU=Org/CN=example.com"

# Backup default Nginx config
sudo mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak

# Create new Nginx config with HTTPS
cat <<EOF | sudo tee /etc/nginx/nginx.conf
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

events {
```

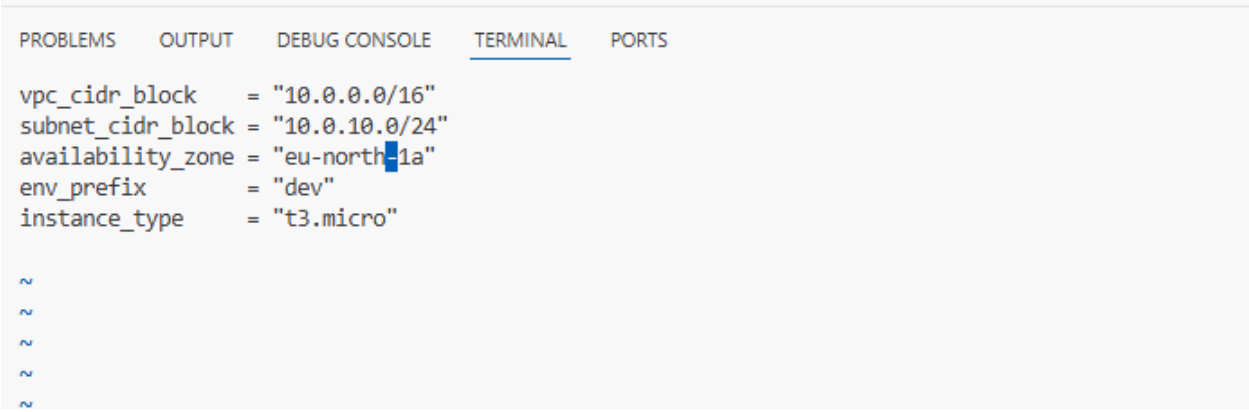
```
events {
    worker_connections 1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    server {
        listen 80;
        server_name _;
        return 301 https://\$host\$request_uri;
    }

    server {
        listen 443 ssl;
        server_name _;

        ssl_certificate     /etc/nginx/ssl/selfsigned.crt;
        ssl_certificate_key /etc/nginx/ssl/selfsigned.key;

        location / {
            root   /usr/share/nginx/html;
            index  index.html;
        }
    }
}
EOF
```

Add Terraform output for public IP

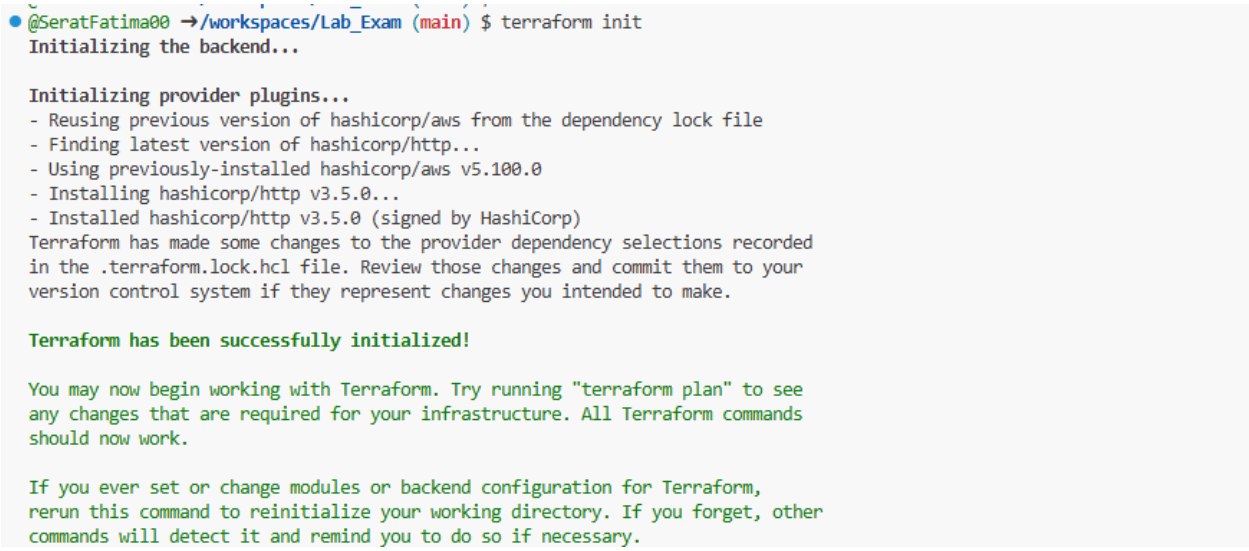
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
output "ec2_public_ip" {
  description = "Public IP of the EC2 instance"
  value       = aws_instance.myapp_ec2.public_ip
}
~
~
~
```

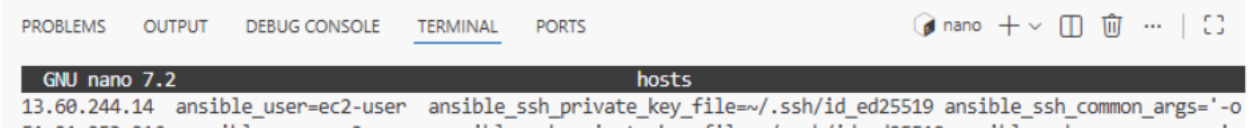
Set variable values for apply time



Run Terraform commands and capture outputs



Q3 – Ansible Playbook for EC2 Web Server Using Q2 Instance (10 marks)



```
[defaults]
host_key_checking = False
interpreter_python = /usr/bin/python3
~
~
~
```

```
---
- name: Configure nginx web server
  hosts: ec2
  become: true
  tasks:
    - name: install nginx and update cache
      yum:
        name: nginx
        state: present
        update_cache: yes

    - name: start nginx server
      service:
        name: nginx
        state: started
~
~
~
~
```

```
PLAY [Configure nginx web server] ***********************************************************************************************************************

TASK [Enable and install nginx from amazon-linux-extras] ************************************************************************************************
changed: [13.51.241.6]

TASK [start nginx server] *******************************************************************************************************************************
changed: [13.51.241.6]

PLAY RECAP **********************************************************************************************************************************************
13.51.241.6                : ok=2    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```