

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": "ANPAIWBCKSKIE64ZLYK",
        "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

User groups (1) Info

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

<input type="checkbox"/>	Group name	▲ Users	▼ Permissions	▼ Creation time
<input type="checkbox"/>	SoftwareEngineering	1	Defined	16 minutes ago

Users (2) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

<input type="checkbox"/>	User name	▲ Path	▼ Groups	▼ Last activity	▼ MFA	▼ Password age	▼ Console
<input type="checkbox"/>	Admin	/	0	2 minutes ago	-	24 days	2 min
<input type="checkbox"/>	SeeratFatima	/	1	-	-	-	-

The screenshot shows the AWS IAM User group summary page for 'SoftwareEngineering'. It displays the user group name, creation time, and ARN. Below this, there are tabs for 'Users (1)', 'Permissions' (which is selected), and 'Access Advisor'. Under the 'Permissions' tab, it shows 'Permissions policies (1) Info'. A note says 'You can attach up to 10 managed policies.' There is a search bar, a filter dropdown set to 'All types', and a table showing one policy named 'AdministratorAccess' which is an 'AWS managed - job function'. There are buttons for 'Simulate', 'Remove', and 'Add permissions'.

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

Configure the AWS provider

The screenshot shows a terminal window with several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal content displays Terraform configuration code:

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

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

Define input variables

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
}

variable "subnet_cidr_block" {
  description = "CIDR block for the subnet"
  type        = string
}

variable "availability_zone" {
  description = "Availability zone for the subnet"
  type        = string
}

variable "env_prefix" {
  description = "Environment name prefix (e.g. dev, prod)"
  type        = string
}

variable "instance_type" {
  description = "EC2 instance type"
  type        = string
}
```

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



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

data "http" "my_ip" {
    url = "https://icanhazip.com"
}

locals {
    my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}

~
```

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
vpc_cidr_block      = "10.0.0.0/16"
subnet_cidr_block   = "10.0.10.0/24"
availability_zone   = "eu-north-1a"
env_prefix          = "dev"
instance_type       = "t3.micro"

~  
~  
~  
~  
~
```

Run Terraform commands and capture outputs

```
● @SeratFatima00 → /workspaces/Lab_Exam (main) $ terraform init
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Finding latest version of hashicorp/http...
- Using previously-installed hashicorp/aws v5.100.0
- Installing hashicorp/http v3.5.0...
- Installed hashicorp/http v3.5.0 (signed by HashiCorp)

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

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

```
GNU nano 7.2                               hosts
13.60.244.14 ansible_user=ec2-user  ansible_ssh_private_key_file=~/ssh/id_ed25519 ansible_ssh_common_args='"-o StrictHostKeyChecking=no"'
```

```
[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
```