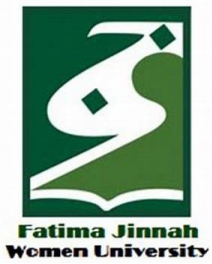
CLOUD COMPUTING LAB

FINAL EXAM



Submitted To:

Engr. Shoaib

Submitted By:

Komal Kashif

BSE V-A

2023-BSE-031

Q1 – AWS IAM Setup Using AWS CLI and Console Verification

1. Create IAM group SoftwareEngineering using AWS CLI

```
@KomalKashif →/workspaces/Lab_exam (main) $ aws iam create-group --group-name So
ftwareEngineering
{
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3TFVF2NRZXXNDGYSK",
        "Arn": "arn:aws:iam::797096399715:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:38:01+00:00"
    }
}
```

```
@KomalKashif →/workspaces/Lab_exam (main) $ aws iam get-group --group-name Softw
areEngineering
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3TFVF2NRZXXNDGYSK",
        "Arn": "arn:aws:iam::797096399715:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:38:01+00:00"
    }
}
```

2. Create IAM user (your name) and view details

```
@KomalKashif →/workspaces/Lab_exam (main) $ aws iam create-user --user-name Koma
lKashif
{
    "User": {
        "Path": "/",
        "UserName": "KomalKashif",
        "UserId": "AIDA3TFVF2NR5PGSDGMJQ",
        "Arn": "arn:aws:iam::797096399715:user/KomalKashif",
        "CreateDate": "2026-01-19T07:39:35+00:00"
    }
}
```

```
@KomalKashif →/workspaces/Lab_exam (main) $ aws iam get-user --user-name KOmalKa
shif
{
    "User": {
        "Path": "/",
        "UserName": "KomalKashif",
        "UserId": "AIDA3TFVF2NR5PGSDGMJQ",
        "Arn": "arn:aws:iam::797096399715:user/KomalKashif",
        "CreateDate": "2026-01-19T07:39:35+00:00"
    }
}
```

3. Add the IAM user to the SoftwareEngineering group

```
@KomalKashif ➜/workspaces/Lab_exam (main) $ aws iam add-user-to-group \
  --user-name KomalKashif \
  --group-name SoftwareEngineering
```

```
@KomalKashif ➜/workspaces/Lab_exam (main) $ aws iam get-group --group-name Softw
areEngineering
{
    "Users": [
        {
            "Path": "/",
            "UserName": "KomalKashif",
            "UserId": "AIDA3TFVF2NR5PGSDGMJQ",
            "Arn": "arn:aws:iam::797096399715:user/KomalKashif",
            "CreateDate": "2026-01-19T07:39:35+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPA3TFVF2NRZXXNDGYSK",
        "Arn": "arn:aws:iam::797096399715:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:38:01+00:00"
    }
}
```

4. Attach AdministratorAccess managed policy to the SoftwareEngineering group

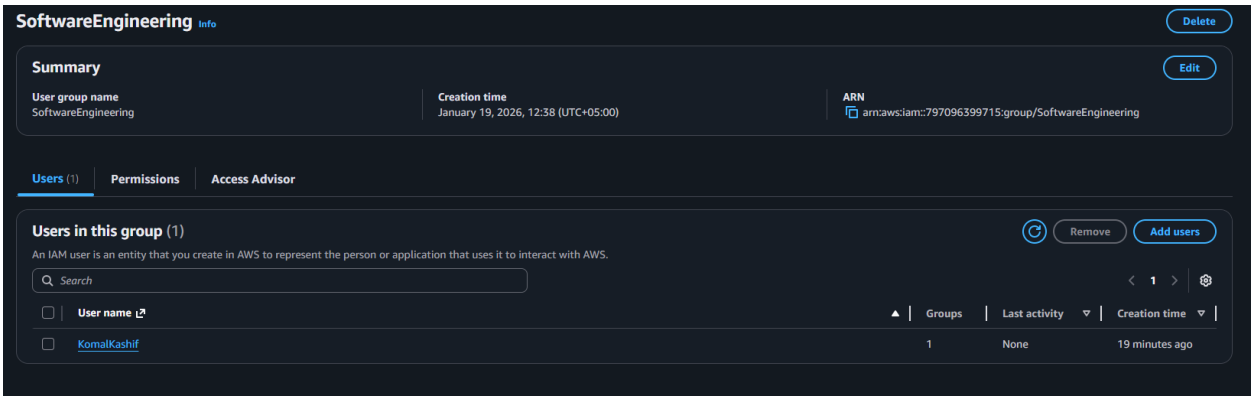
```
@KomalKashif ➜/workspaces/Lab_exam (main) $ aws iam get-policy \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
{
    "Policy": {
        "PolicyName": "AdministratorAccess",
        "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
        "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 4,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "Provides full access to AWS services and resources.",
        "CreateDate": "2015-02-06T18:39:46+00:00",
        "UpdateDate": "2015-02-06T18:39:46+00:00",
        "Tags": []
    }
}
```

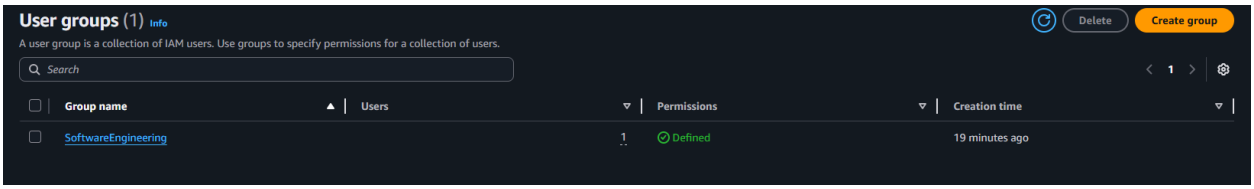
```
@KomalKashif →/workspaces/Lab_exam (main) $ aws iam attach-group-policy \
  --group-name SoftwareEngineering \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

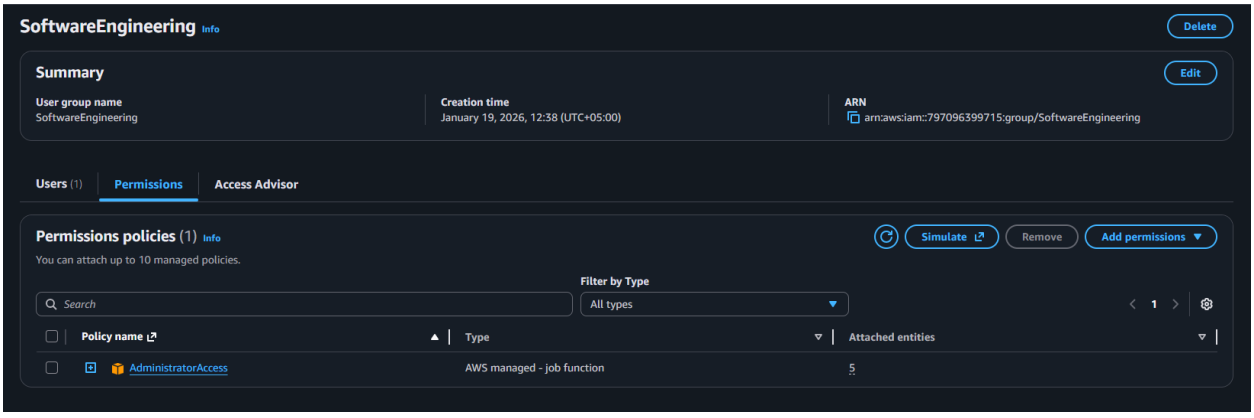
5. List attached policies of the SoftwareEngineering group

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

6. Verify IAM configuration in AWS Management Console







Q2 – Terraform Lab: Simple AWS Environment with Nginx over HTTPS

1. Configure the AWS provider

```
provider "aws" {
  region  = "me-central-1"
  profile = "default"
}
```

2. Define input variables

```
variable "vpc_cidr_block" {
  type = string
}

variable "subnet_cidr_block" {
  type = string
}

variable "availability_zone" {
  type = string
}

variable "env_prefix" {
  type = string
}

variable "instance_type" {
  type = string
}
```

3. Create VPC and subnet

```
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
  map_public_ip_on_launch = true

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

4. Create Internet Gateway and configure default route table

```
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_rt"
{
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_

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

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

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

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

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

6. Configure the default security group in the VPC

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

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [local.my_ip]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
  ingress
  {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress
  {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

7. Create an AWS key pair for SSH

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

8. Create the EC2 instance resource

```
resource "aws_instance" "myapp_ec2"
{
  ami                         = "ami-0d6d74f27c3c3e6c3"
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet.id
  vpc_security_group_ids      = [aws_default_security_group.defau
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = aws_key_pair.serverkey.key_name
  user_data                   = file("entry-script.sh")

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

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

terraform-q2 > $ entry-script.sh

```
#!/bin/bash

dnf update -y
dnf install -y nginx openssl

mkdir -p /etc/nginx/ssl

openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -keyout /etc/nginx/ssl/self.key \
  -out /etc/nginx/ssl/self.crt \
  -subj "/C=PK/ST=Lab/L=Terraform/O=Terraform/CN=localhost"

cat <<EOF > /etc/nginx/conf.d/terraform.conf
server {
    listen 80;
    return 301 https://\$host\$request_uri;
}
```

```
server {
    listen 443 ssl;
    ssl_certificate /etc/nginx/ssl/self.crt;
    ssl_certificate_key /etc/nginx/ssl/self.key;

    location / {
        return 200 "<h1>This is Komal Kashif's Terraform environm
    }
}
EOF

systemctl enable nginx
systemctl restart nginx
```

10. Add Terraform output for public IP

terraform-q2 > outputs.tf

```
output "ec2_public_ip" {
  value = aws_instance.myapp_ec2.public_ip
}
```

11. Set variable values for apply time

terraform-q2 > terraform.tfvars

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

12. Run Terraform commands and capture outputs

13. Verify Terraform resources in AWS console
14. Verify HTTPS access from browser