Name : Reena Qureshi

Reg No: 2023-BSE-052

Section: V-B

LAB EXAM

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

Create IAM group softwareengineering using AWS CLI

```
@reenaqureshi /workspaces/LabExam $ aws iam create-group --group-name SoftwareEngineering
{
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPAZCN5ZTDXHODXXCN3M",
        "Arn": "arn:aws:iam::623705168110:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:32:23+00:00"
    }
}
```

Create IAM user (your name) and view details

```
@reenaqureshi /workspaces/LabExam $ aws iam create-user --user-name ReenaQureshi
{
    "User": {
        "Path": "/",
        "UserName": "ReenaQureshi",
        "UserId": "AIDAZCN5ZTDXBCQJ5ZKSE",
        "Arn": "arn:aws:iam::623705168110:user/ReenaQureshi",
        "CreateDate": "2026-01-19T07:33:18+00:00"
    }
}
```

Add the IAM user to the softwareengineering group

```
@reenaqureshi /workspaces/LabExam $ aws iam add-user-to-group --user-name ReenaQureshi --group-name SoftwareEngineering
@reenaqureshi /workspaces/LabExam $ aws iam get-group --group-name SoftwareEngineering
{
    "Users": [
        {
            "Path": "/",
            "UserName": "ReenaQureshi",
            "UserId": "AIDAZCN5ZTDXBCQJ5ZKSE",
            "Arn": "arn:aws:iam::623705168110:user/ReenaQureshi",
            "CreateDate": "2026-01-19T07:33:18+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPAZCN5ZTDXHODXXCN3M",
        "Arn": "arn:aws:iam::623705168110:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:32:23+00:00"
    }
}
```

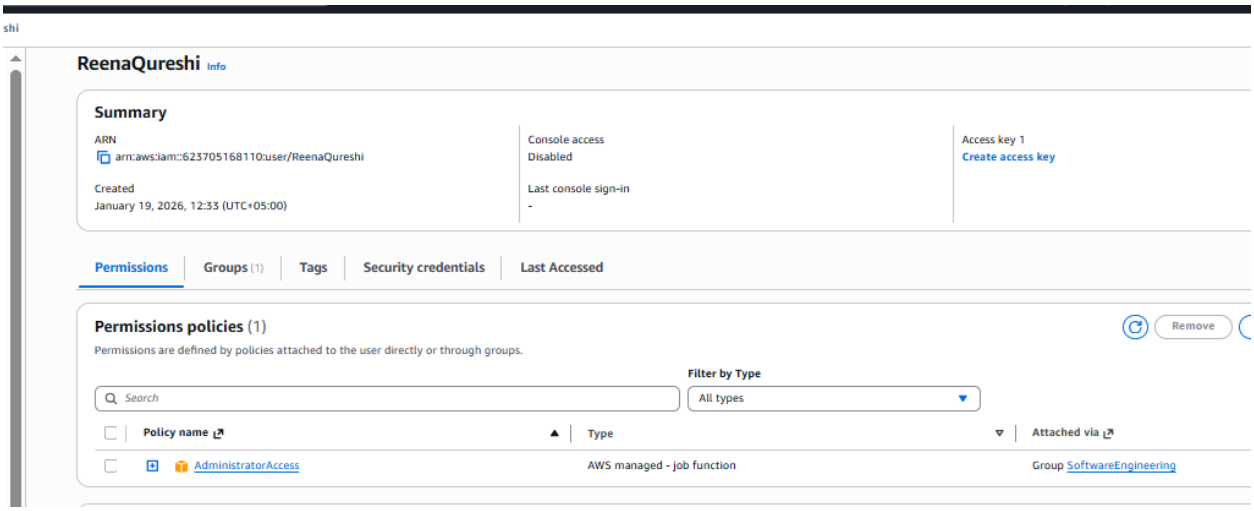
Attach administratoraccess managed policy to the softwareengineering group

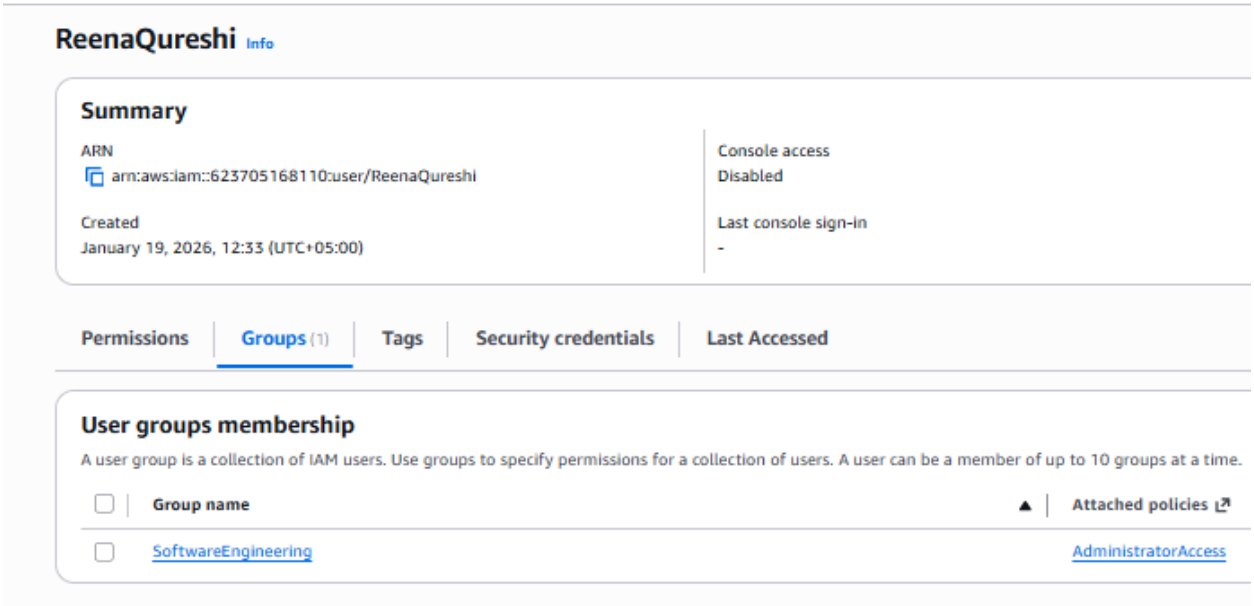
```
@reenaqureshi 🖹 /workspaces/LabExam $ aws iam list-policies --query 'Policies[?PolicyName==`AdministratorAccess`].[PolicyName,Arn]' --output table
-------------------------------------------------------------------
|                          ListPolicies                           |
+---------------------+-----------------------------------------+
|  AdministratorAccess |  arn:aws:iam::aws:policy/AdministratorAccess  |
+---------------------+-----------------------------------------+
@reenaqureshi 🖹 /workspaces/LabExam $ aws iam attach-group-policy \
>   --group-name SoftwareEngineering \
>   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

List attached policies of the softwareengineering group

```
@reenaqureshi 🖹 /workspaces/LabExam $ 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

```
@reenaqureshi /workspaces/LabExam $ aws sts get-caller-identity
{
    "UserId": "AIDAZCN5ZTDXBBDUSAKUZ",
    "Account": "623705168110",
```

```
@reenaqureshi /workspaces/LabExam $ cat main.tf
# Provider
provider "aws" {
  region                   = "me-central-1"
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
```

Define input variables

```
@reenaqureshi /workspaces/LabExam $ cat variables.tf
variable "vpc_cidr_block" {
  description = "CIDR block for VPC"
  type        = string
}

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

variable "availability_zone" {
  description = "Availability zone"
  type        = string
}

variable "env_prefix" {
  description = "Environment prefix"
  type        = string
}

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

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_1" {
  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

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

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

```
# Default Route Table
resource "aws_default_route_table" "main_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

```
@reenaqureshi 🖹 /workspaces/LabExam $ cat locals.tf
# Get your public IP
data "http" "my_ip" {
  url = "https://icanhazip.com"
}

locals {
  my_ip = "${chomp(data.http.my_ip.response_body)}/32"
}
@reenaqureshi 🖹 /workspaces/LabExam $
```

Configure the default security group in the VPC

```
# Default Security Group
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"
  }
}
```

Create an AWS key pair for SSH

```
# SSH Key Pair
resource "aws_key_pair" "ssh_key" {
  key_name   = "serverkey"
  public_key = file("~/.ssh/id_ed25519.pub")
}
```

Create the EC2 instance resource

```
resource "aws_instance" "myapp_server" {
  ami                         = "ami-05524d6658fcf35b6"  # Amazon Linux 2023
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet_1.id
  vpc_security_group_ids      = [aws_default_security_group.default_sg.id]
  availability_zone           = var.availability_zone
  associate_public_ip_address = true
  key_name                    = aws_key_pair.ssh_key.key_name

  user_data = file("./entry-script.sh")

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

Create entry-script.sh to configure Nginx + HTTPS

```
  GNU nano 7.2                                                    entry-script.s
#!/bin/bash
set -e

# Update system
yum update -y

# Install Nginx
yum install -y nginx

# Get instance metadata using IMDSv2
TOKEN=$(curl -s -X PUT "http://169.254.169.254/latest/api/token" \
  -H "X-aws-ec2-metadata-token-ttl-seconds: 21600")

PUBLIC_IP=$(curl -s -H "X-aws-ec2-metadata-token: $TOKEN" \
  http://169.254.169.254/latest/meta-data/public-ipv4)

# Create SSL certificate
mkdir -p /etc/ssl/private /etc/ssl/certs
openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
  -keyout /etc/ssl/private/selfsigned.key \
  -out /etc/ssl/certs/selfsigned.crt \
  -subj "/CN=$PUBLIC_IP" \
  -addext "subjectAltName=IP:$PUBLIC_IP"

# Configure Nginx
cat > /etc/nginx/nginx.conf <<'NGINX_EOF'
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                    '$status $body_bytes_sent "$http_referer" '
                    '"$http_user_agent" "$http_x_forwarded_for"';
```

```
    access_log /var/log/nginx/access.log main;

    # HTTPS server
    server {
        listen 443 ssl;
        server_name _;

        ssl_certificate /etc/ssl/certs/selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/selfsigned.key;

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

    # HTTP server (redirect to HTTPS)
    server {
        listen 80;
        server_name _;
        return 301 https://$host$request_uri;
    }
}
NGINX_EOF

# Create custom HTML page with YOUR NAME
cat > /usr/share/nginx/html/index.html <<'HTML_EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Reena's Terraform Environment</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            padding: 50px;
```

```
        }
    }

    # HTTP server (redirect to HTTPS)
    server {
        listen 80;
        server_name _;
        return 301 https://$host$request_uri;
    }
}
NGINX_EOF

# Create custom HTML page with YOUR NAME
cat > /usr/share/nginx/html/index.html <<'HTML_EOF'
<!DOCTYPE html>
<html>
<head>
    <title>Reena's Terraform Environment</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
            color: white;
            padding: 50px;
            text-align: center;
        }
        .container {
            background: rgba(255, 255, 255, 0.1);
            padding: 40px;
            border-radius: 15px;
            backdrop-filter: blur(10px);
        }
        h1 { font-size: 3em; margin: 0; }
        p { font-size: 1.5em; }
    </style>
</head>
<body>
    <div class="container">
        <h1>  This is Reena's Terraform Environment</h1>
        <p>Deployed with Terraform </p>
```

Add Terraform output for public IP

```
@reenaqureshi  /workspaces/LabExam $ cat outputs.tf
output "ec2_public_ip" {
  description = "EC2 instance public IP"
  value       = aws_instance.myapp_server.public_ip
}
@reenaqureshi  /workspaces/LabExam $
```

Set variable values for apply time

```
@reenaqureshi ⍰ /workspaces/LabExam $ cat 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"
@reenaqureshi ⍰ /workspaces/LabExam $
```

Run Terraform commands and capture outputs

```
@reenaqureshi ⍰ /workspaces/LabExam $ @reenaqureshi ⍰ /workspaces/LabExam $ nano entry-script.sh
@reenaqureshi ⍰ /workspaces/LabExam $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Finding latest version of hashicorp/http...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (signed by HashiCorp)
- Installing hashicorp/http v3.5.0...
- Installed hashicorp/http v3.5.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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,

commands will detect it and remind you to do so if necessary.
```

Terraform plan

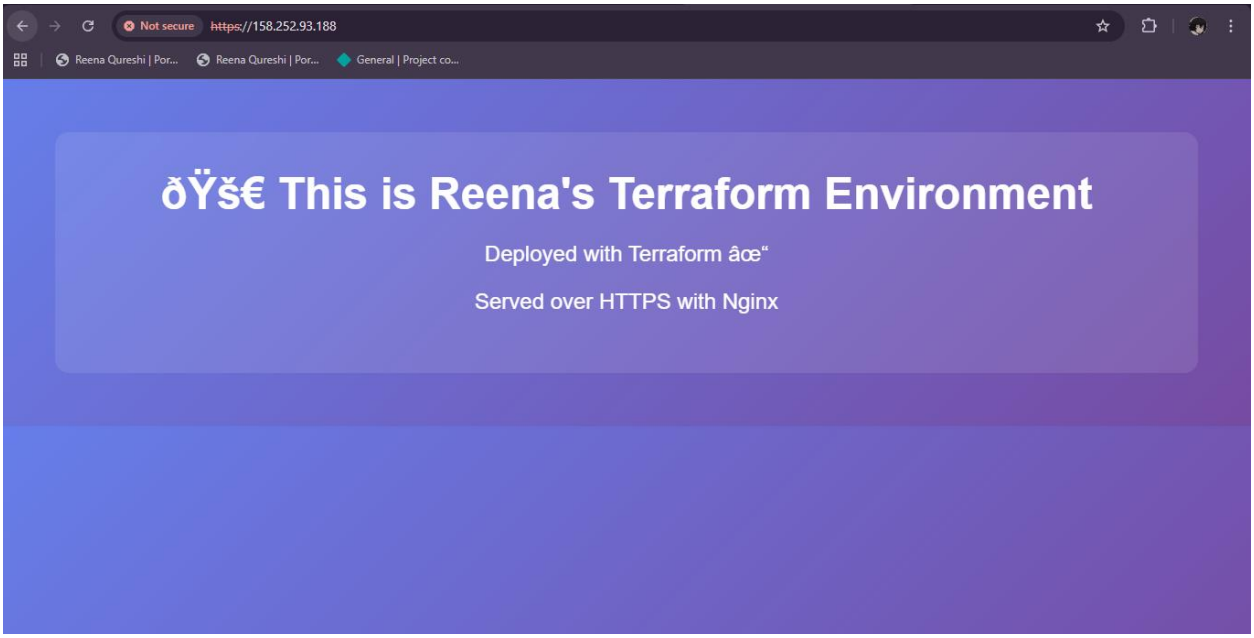
```
      + owner_id                                = (known after apply)
      + region                                  = "me-central-1"
      + tags                                    = {
          + "Name" = "dev-vpc"
        }
      + tags_all                                = {
          + "Name" = "dev-vpc"
        }
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + ec2_public_ip = (known after apply)
```

```
aws_default_security_group.default_sg: Creating...
aws_internet_gateway.myapp_igw: Creation complete after 1s [id=igw-011a4633423098584]
aws_default_route_table.main_rt: Creating...
aws_subnet.myapp_subnet_1: Creation complete after 1s [id=subnet-035e4557aa5d66549]
aws_default_route_table.main_rt: Creation complete after 1s [id=rtb-0dbb6976f22e84bc8]
aws_default_security_group.default_sg: Creation complete after 3s [id=sg-055a2f8959cef8bfa]
aws_instance.myapp_server: Creating...
aws_instance.myapp_server: Still creating... [00m10s elapsed]
aws_instance.myapp_server: Creation complete after 13s [id=i-027dfaa6c018c8631]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

ec2_public_ip = "158.252.93.188"
@reenaqureshi /workspaces/LabExam $ terraform output
```

Terraform output

```
@reenaqureshi /workspaces/LabExam $ terraform output
ec2_public_ip = "158.252.93.188"
@reenaqureshi /workspaces/LabExam $
```

Verify HTTPS access from browser



QNO3 Ansible Playbook for EC2 Web Server Using Q2 Instance (10 marks)

Create Ansible inventory file hosts

```
@reenaqureshi ➜ /workspaces/LabExam/ansible $ cat hosts
[ec2]
158.252.93.188

[ec2:vars]
ansible_user=ec2-user
ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```

Create project-level ansible.cfg

```
@reenaqureshi ➜ /workspaces/LabExam/ansible $ cat ansible.cfg
[defaults]
host_key_checking = False
interpreter_python = /usr/bin/python3
inventory = ./hosts

[ssh_connection]
ssh_args = -o StrictHostKeyChecking=no
@reenaqureshi ➜ /workspaces/LabExam/ansible $
```

Create Ansible playbook (e.g. My-playbook.yml)

```
@reenaqureshi ⍰ /workspaces/LabExam/ansible $ cat my-playbook.yml
---
- name: Configure EC2 Web Server with Apache
  hosts: ec2
  become: true

  tasks:
    - name: Update system packages
      yum:
        name: '*'
        state: latest
        update_cache: yes

    - name: Stop nginx if running
      service:
        name: nginx
        state: stopped
        enabled: no
      ignore_errors: yes

    - name: Uninstall nginx to free port 80
      yum:
        name: nginx
        state: absent

    - name: Install Apache HTTPD
      yum:
        name: httpd
        state: present

    - name: Start and enable HTTPD
      service:
        name: httpd
        state: started
        enabled: yes

    - name: Get IMDSv2 token
      uri:
        url: http://169.254.169.254/latest/api/token
        method: PUT
        headers:
          X-aws-ec2-metadata-token-ttl-seconds: "21600"
        return_content: yes
      register: imds_token
```

```
    return_content: yes
  register: imds_token

- name: Get public IPv4 address
  uri:
    url: http://169.254.169.254/latest/meta-data/public-ipv4
    method: GET
    headers:
      X-aws-ec2-metadata-token: "{{ imds_token.content }}"
    return_content: yes
  register: public_ipv4

- name: Get public hostname
  uri:
    url: http://169.254.169.254/latest/meta-data/public-hostname
    method: GET
    headers:
      X-aws-ec2-metadata-token: "{{ imds_token.content }}"
    return_content: yes
  register: public_hostname

- name: Print public IP
  debug:
    msg: "Public IP: {{ public_ipv4.content }}"

- name: Print public hostname
  debug:
    msg: "Public Hostname: {{ public_hostname.content }}"

- name: Restart HTTPD service
  service:
    name: httpd
    state: restarted
```

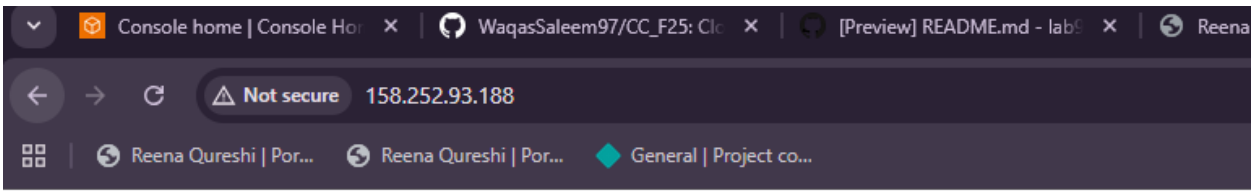
Run the Ansible playbook

Execute:

```
@reenaqureshi ➜ /workspaces/LabExam/ansible $ ansible-playbook -i hosts my-playbook.yml
[WARNING]: Ansible is being run in a world writable directory (/workspaces/LabExam/ansible), ignoring it as an ansible.cfg source. For more information see
https://docs.ansible.com/ansible/devel/reference_appendices/config.html#cfg-in-world-writable-dir

PLAY [Configure EC2 Web Server with Apache] ************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Update system packages] **************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Stop nginx if running] ***************************************************************************************************************************************
changed: [158.252.93.188]

TASK [Uninstall nginx to free port 80] *****************************************************************************************************************************
changed: [158.252.93.188]

TASK [Install Apache HTTPD] ****************************************************************************************************************************************
changed: [158.252.93.188]

TASK [Start and enable HTTPD] **************************************************************************************************************************************
changed: [158.252.93.188]

TASK [Get IMDSv2 token] ********************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Get public IPv4 address] *************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Get public hostname] *****************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Print public IP] *********************************************************************************************************************************************
ok: [158.252.93.188] => {
    "msg": "Public IP: 158.252.93.188"
}
```

```
TASK [Get public IPv4 address] *************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Get public hostname] *****************************************************************************************************************************************
ok: [158.252.93.188]

TASK [Print public IP] *********************************************************************************************************************************************
ok: [158.252.93.188] => {
    "msg": "Public IP: 158.252.93.188"
}

TASK [Print public hostname] ***************************************************************************************************************************************
ok: [158.252.93.188] => {
    "msg": "Public Hostname: "
}

TASK [Restart HTTPD service] ***************************************************************************************************************************************
changed: [158.252.93.188]

PLAY RECAP *********************************************************************************************************************************************************
158.252.93.188             : ok=12   changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Verify HTTP access



It works!

Cleanup (ungraded)

```
aws_instance.myapp_server: Still destroying... [id=i-027dfaa6c018c8631, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-011a4633423098584, 00m20s elapsed]
aws_instance.myapp_server: Still destroying... [id=i-027dfaa6c018c8631, 00m30s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-011a4633423098584, 00m30s elapsed]
aws_instance.myapp_server: Still destroying... [id=i-027dfaa6c018c8631, 00m40s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-011a4633423098584, 00m40s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 49s
aws_instance.myapp_server: Still destroying... [id=i-027dfaa6c018c8631, 00m50s elapsed]
aws_instance.myapp_server: Destruction complete after 52s
aws_key_pair.ssh_key: Destroying... [id=serverkey]
aws_subnet.myapp_subnet_1: Destroying... [id=subnet-035e4557aa5d66549]
aws_default_security_group.default_sg: Destroying... [id=sg-055a2f8959cef8bfa]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.ssh_key: Destruction complete after 1s
aws_subnet.myapp_subnet_1: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-05d875566487241e6]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@reenaqureshi ⊡ /workspaces/LabExam $ cd ansible
@reenaqureshi ⊡ /workspaces/LabExam/ansible $ cat hosts
[ec2]
158.252.93.188

[ec2:vars]
ansible_user=ec2-user
ansible_ssh_private_key_file=~/.ssh/id_ed25519
ansible_ssh_common_args='-o StrictHostKeyChecking=no'
```