Submitted By: Zunaira Noor

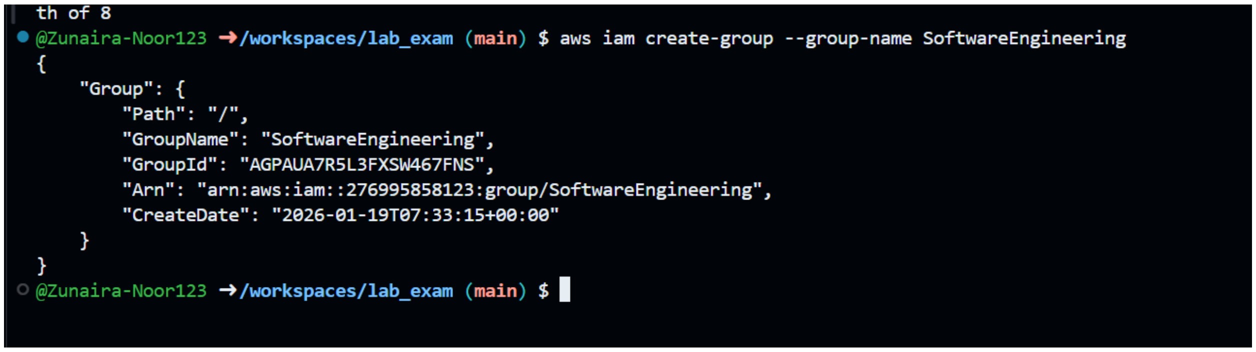
Submitted to: Sir Shoaib

Lab Exam

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

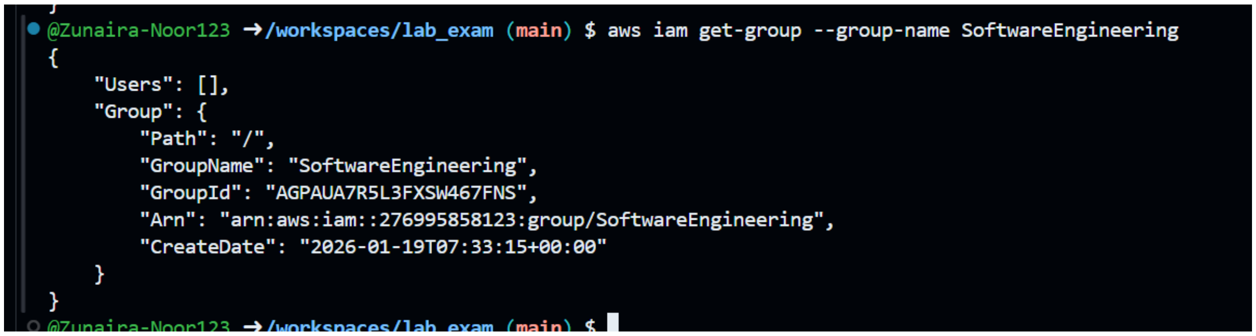
Create IAM group SoftwareEngineering using AWS CLI

Group



```
th of 8
● @Zunaira-Noor123 ➜/workspaces/lab_exam (main) $ aws iam create-group --group-name SoftwareEngineering
{
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPAUA7R5L3FXSW467FNS",
        "Arn": "arn:aws:iam::276995858123:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:33:15+00:00"
    }
}
○ @Zunaira-Noor123 ➜/workspaces/lab_exam (main) $
```

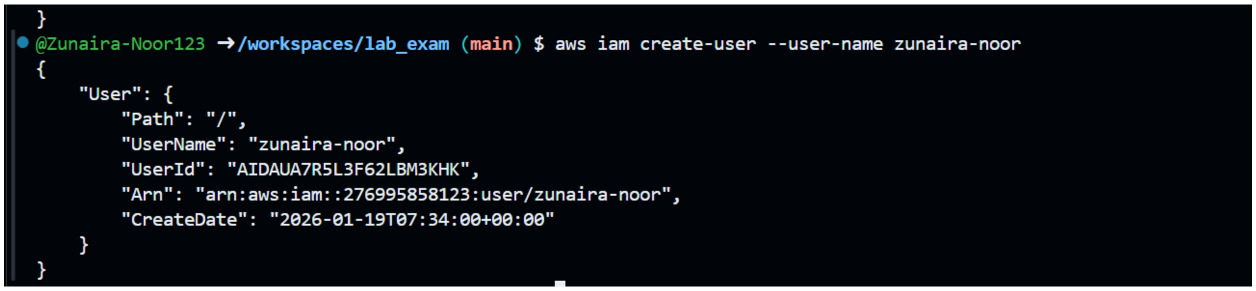
Details



```
● @Zunaira-Noor123 ➜/workspaces/lab_exam (main) $ aws iam get-group --group-name SoftwareEngineering
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "SoftwareEngineering",
        "GroupId": "AGPAUA7R5L3FXSW467FNS",
        "Arn": "arn:aws:iam::276995858123:group/SoftwareEngineering",
        "CreateDate": "2026-01-19T07:33:15+00:00"
    }
}
○ @Zunaira-Noor123 ➜/workspaces/lab_exam (main) $
```

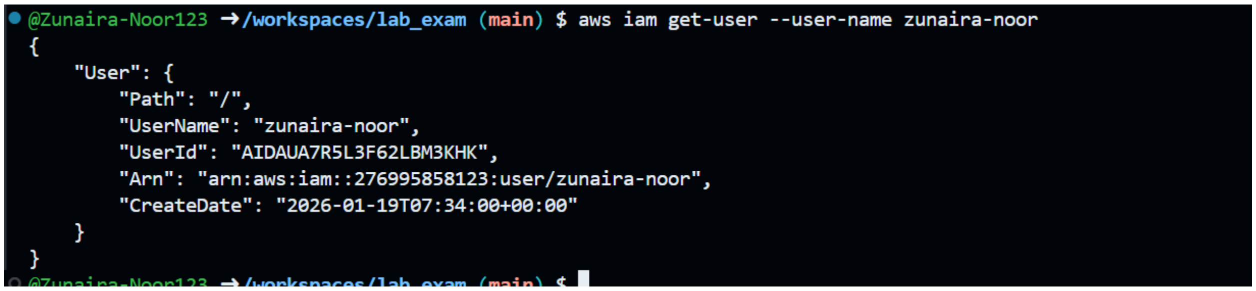
Create IAM user (your name) and view details

User

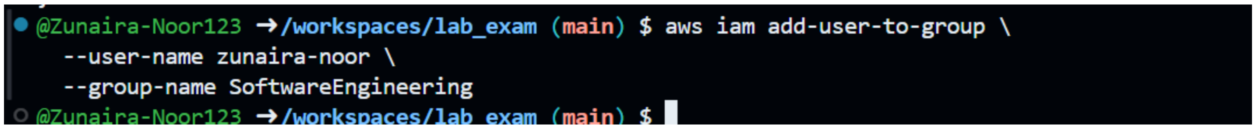


```
}
● @Zunaira-Noor123 ➜/workspaces/lab_exam (main) $ aws iam create-user --user-name zunaira-noor
{
    "User": {
        "Path": "/",
        "UserName": "zunaira-noor",
        "UserId": "AIDAUA7R5L3F62LBM3KHK",
        "Arn": "arn:aws:iam::276995858123:user/zunaira-noor",
        "CreateDate": "2026-01-19T07:34:00+00:00"
    }
}
```

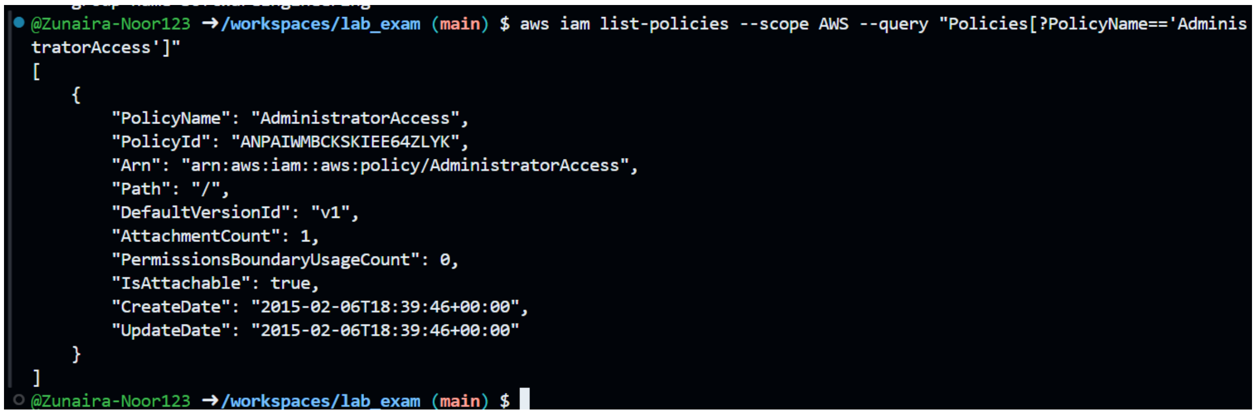
Details


```
@Zunaira-Noor123 →/workspaces/lab_exam (main) $ aws iam get-user --user-name zunaira-noor
{
    "User": {
        "Path": "/",
        "UserName": "zunaira-noor",
        "UserId": "AIDAUA7R5L3F62LBM3KHK",
        "Arn": "arn:aws:iam::276995858123:user/zunaira-noor",
        "CreateDate": "2026-01-19T07:34:00+00:00"
    }
}
@Zunaira-Noor123 →/workspaces/lab_exam (main) $
```

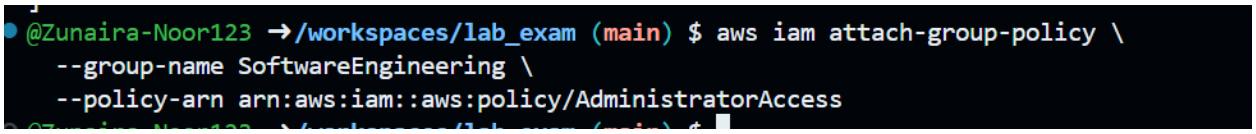
Add the IAM user to the SoftwareEngineering group


```
@Zunaira-Noor123 →/workspaces/lab_exam (main) $ aws iam add-user-to-group \
  --user-name zunaira-noor \
  --group-name SoftwareEngineering
@Zunaira-Noor123 →/workspaces/lab_exam (main) $
```

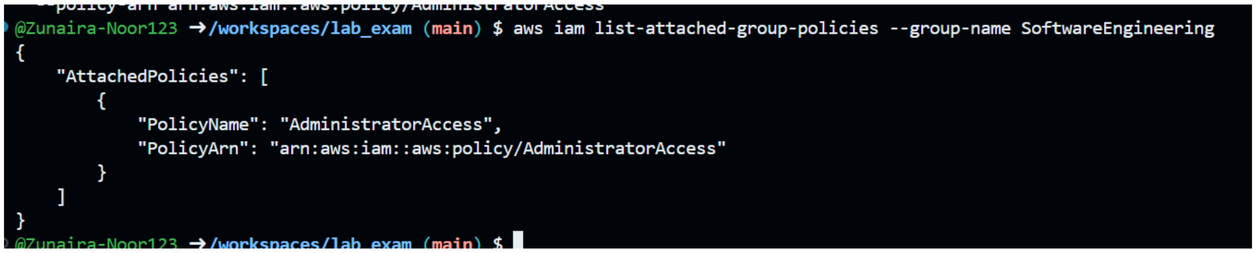
Attach AdministratorAccess managed policy to the SoftwareEngineering group


```
@Zunaira-Noor123 →/workspaces/lab_exam (main) $ aws iam list-policies --scope AWS --query "Policies[?PolicyName=='Adminis
tratorAccess']"
[
    {
        "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"
    }
]
@Zunaira-Noor123 →/workspaces/lab_exam (main) $
```

List attached policies of the SoftwareEngineering group

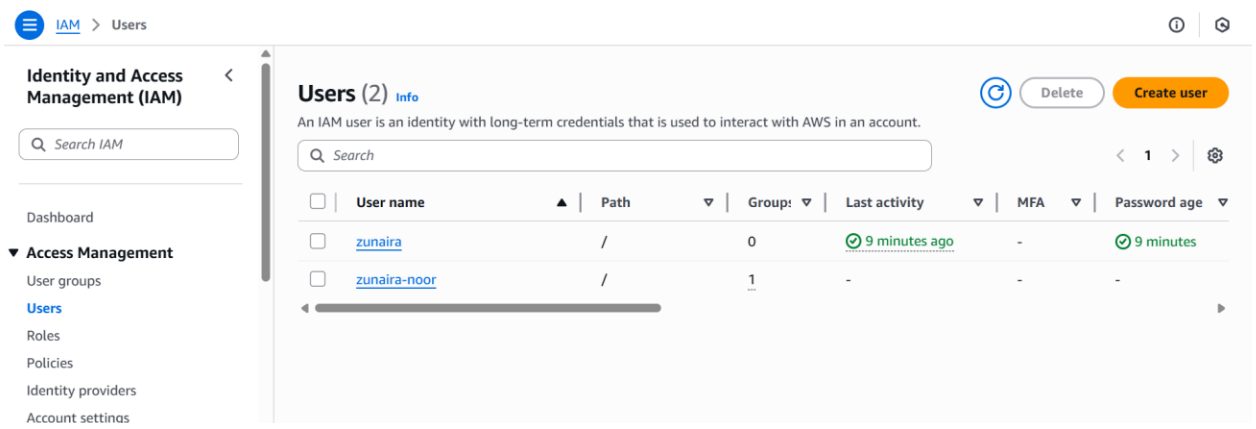

```
@Zunaira-Noor123 →/workspaces/lab_exam (main) $ aws iam attach-group-policy \
  --group-name SoftwareEngineering \
  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
```

Verify IAM configuration in AWS Management Console


```
@Zunaira-Noor123 →/workspaces/lab_exam (main) $ aws iam list-attached-group-policies --group-name SoftwareEngineering
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        }
    ]
}
@Zunaira-Noor123 →/workspaces/lab_exam (main) $
```

Verify IAM configuration in AWS Management Console

IAM user



IAM > Users

Identity and Access Management (IAM)

Search IAM

Dashboard

Access Management

User groups

Users

Roles

Policies

Identity providers

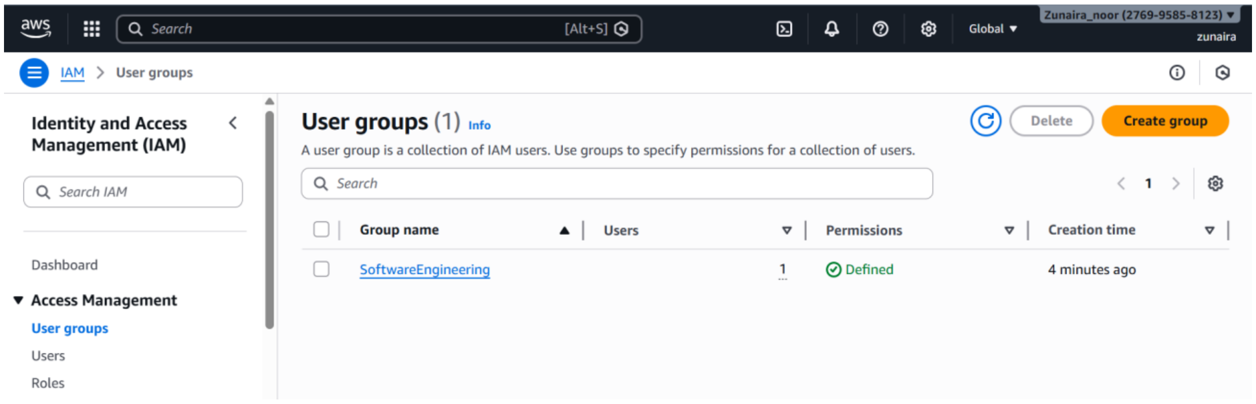
Account settings

Users (2) Info

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

User name	Path	Groups	Last activity	MFA	Password age
zunaira	/	0	9 minutes ago	-	9 minutes
zunaira-noor	/	1	-	-	-

SoftwareEngineering group



Zunaira_noor (2769-9585-8123)

zunaira

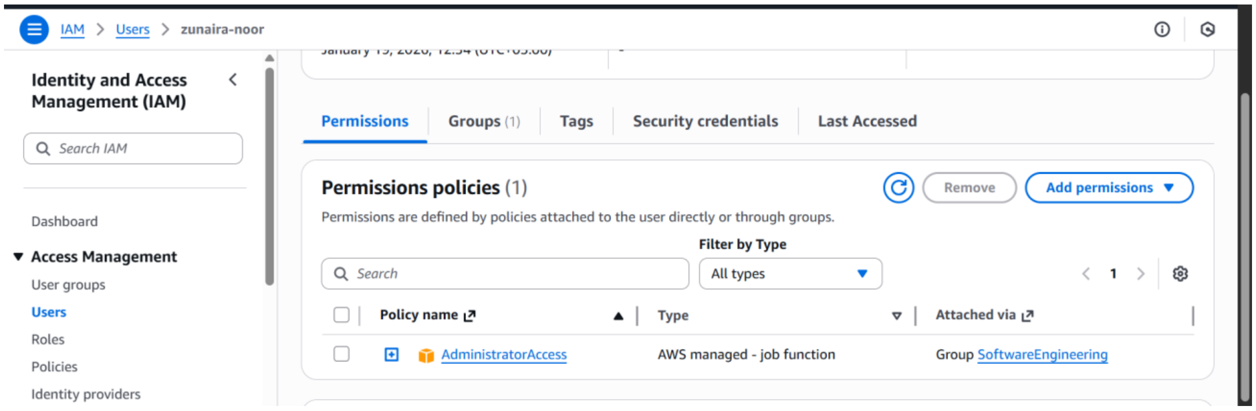
IAM > User groups

User groups (1) Info

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

Group name	Users	Permissions	Creation time
SoftwareEngineering	1	Defined	4 minutes ago

AdministratorAccess



IAM > Users > zunaira-noor

Permissions | Groups (1) | Tags | Security credentials | Last Accessed

Permissions policies (1)

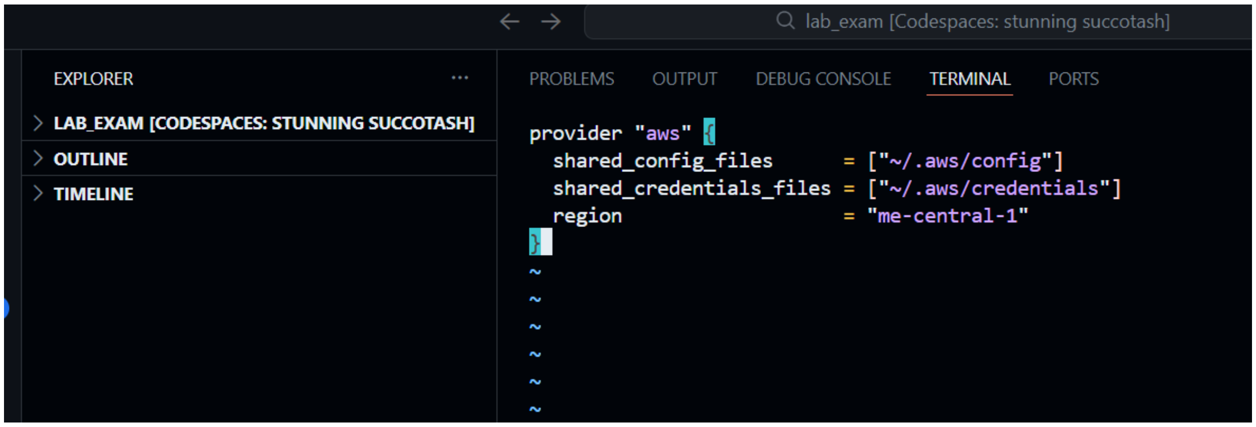
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type: All types

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Group SoftwareEngineering

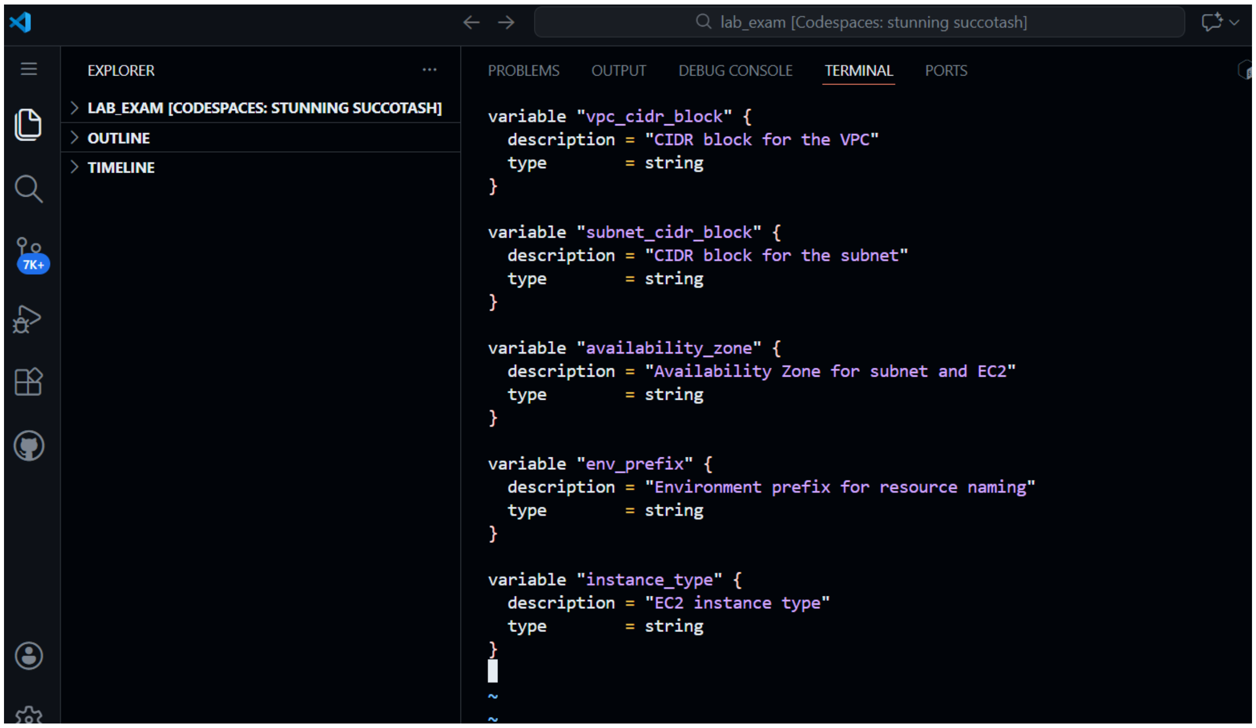
Q2 – Terraform Lab: Simple AWS Environment with Nginx over HTTPS (30 marks)

Configure the AWS provider


lab_exam [Codespaces: stunning succotash]
EXPLORER
LAB_EXAM [CODESPACES: STUNNING SUCCOTASH]
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

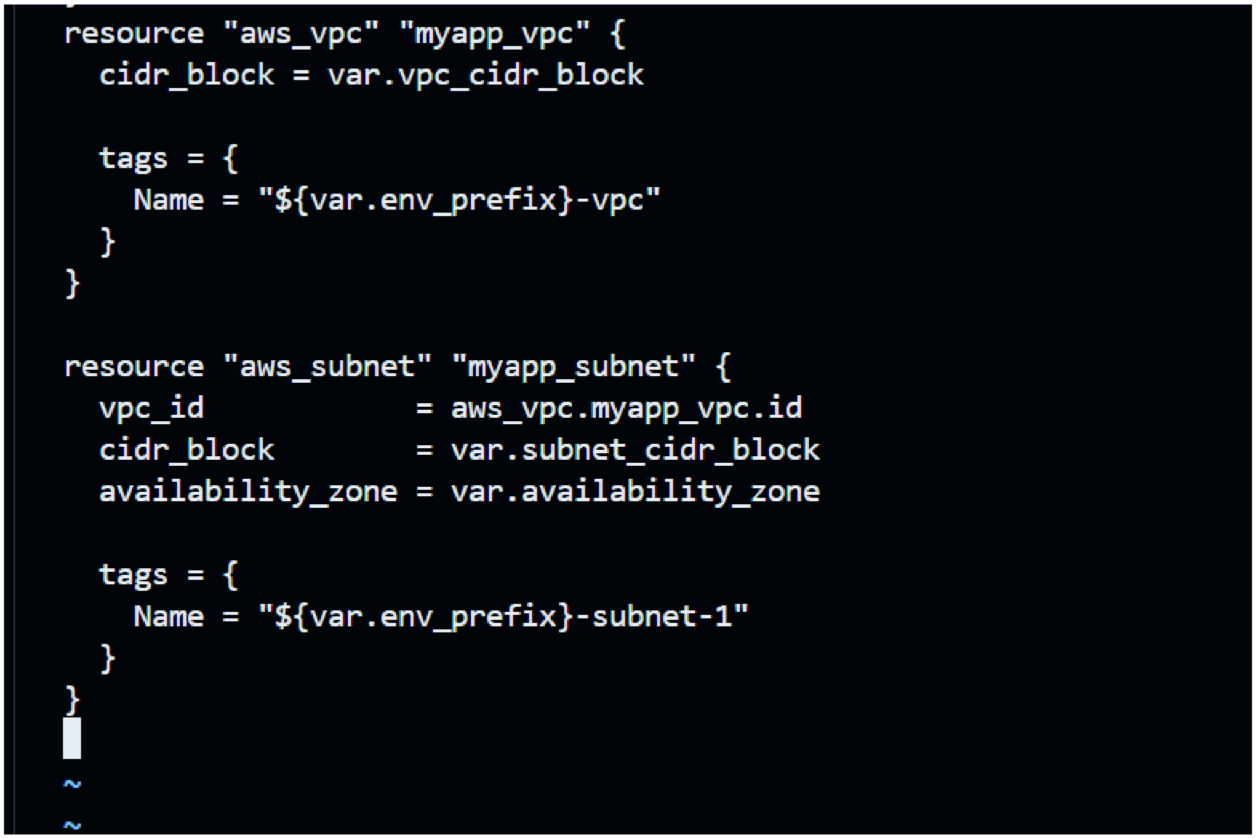
```
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
  region                   = "me-central-1"
}
~
~
~
~
~
~
```

Define input variables


lab_exam [Codespaces: stunning succotash]
EXPLORER
LAB_EXAM [CODESPACES: STUNNING SUCCOTASH]
OUTLINE
TIMELINE
7K+
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

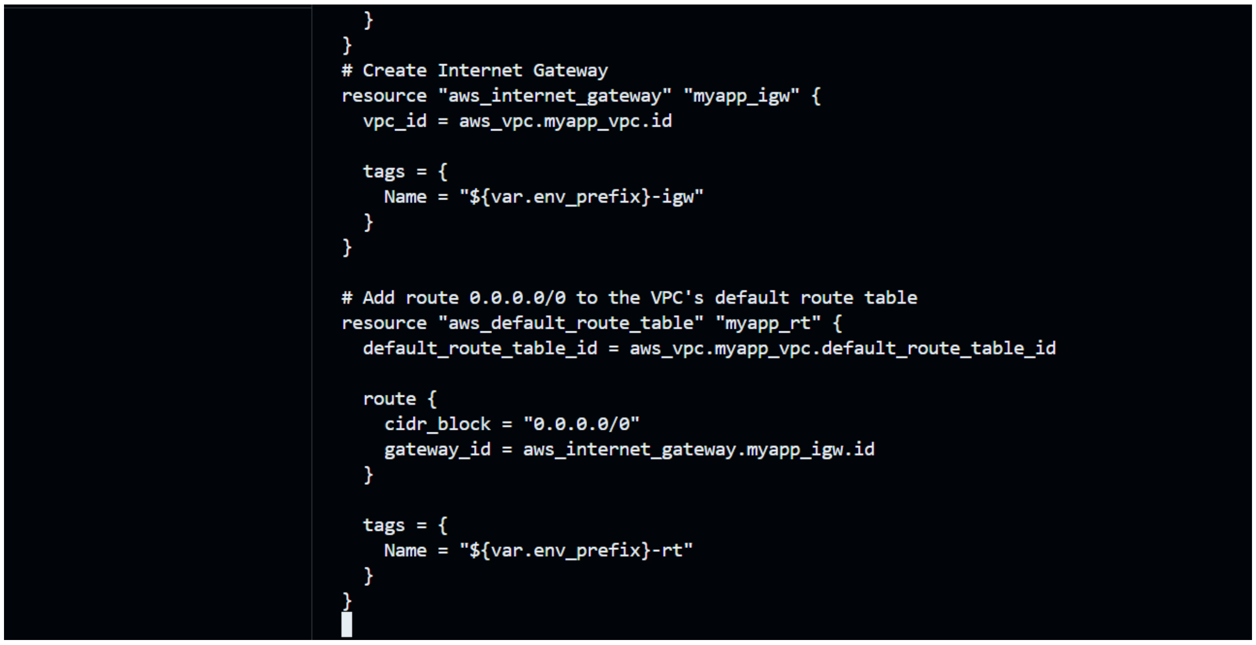
```
variable "vpc_cidr_block" {
  description = "CIDR block for the VPC"
  type        = string
}

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

variable "availability_zone" {
  description = "Availability Zone for subnet and EC2"
  type        = string
}

variable "env_prefix" {
  description = "Environment prefix for resource naming"
  type        = string
}

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

~
~
```

Create VPC and subnet



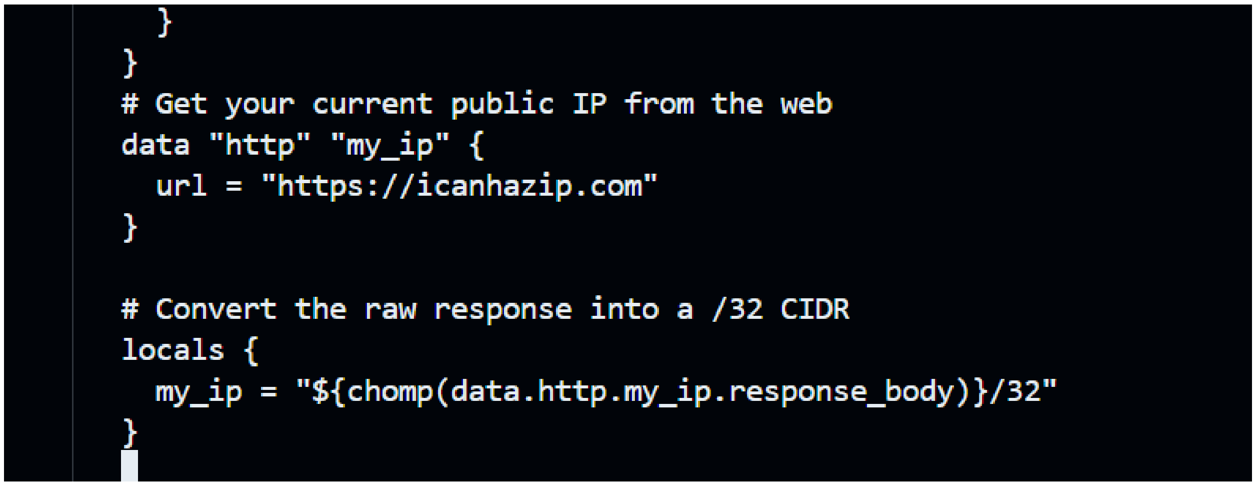
```
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block

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

resource "aws_subnet" "myapp_subnet" {
  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



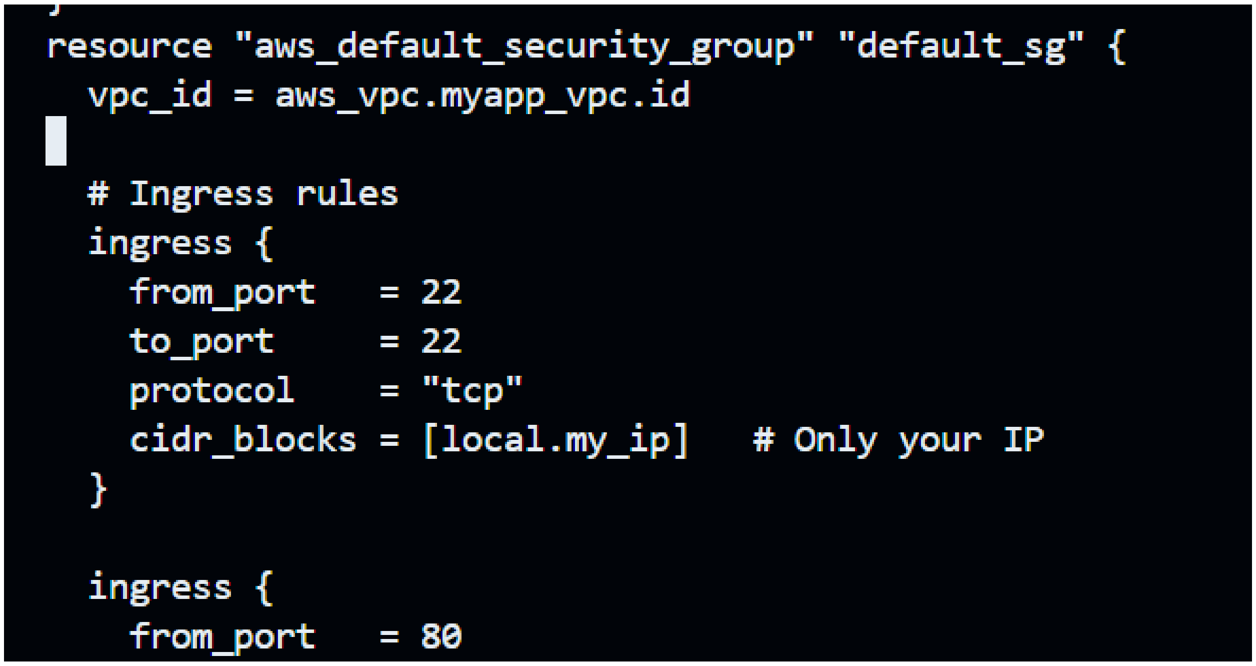
```
  }
}
# Create Internet Gateway
resource "aws_internet_gateway" "myapp_igw" {
  vpc_id = aws_vpc.myapp_vpc.id

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

# Add route 0.0.0.0/0 to the VPC's default route table
resource "aws_default_route_table" "myapp_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

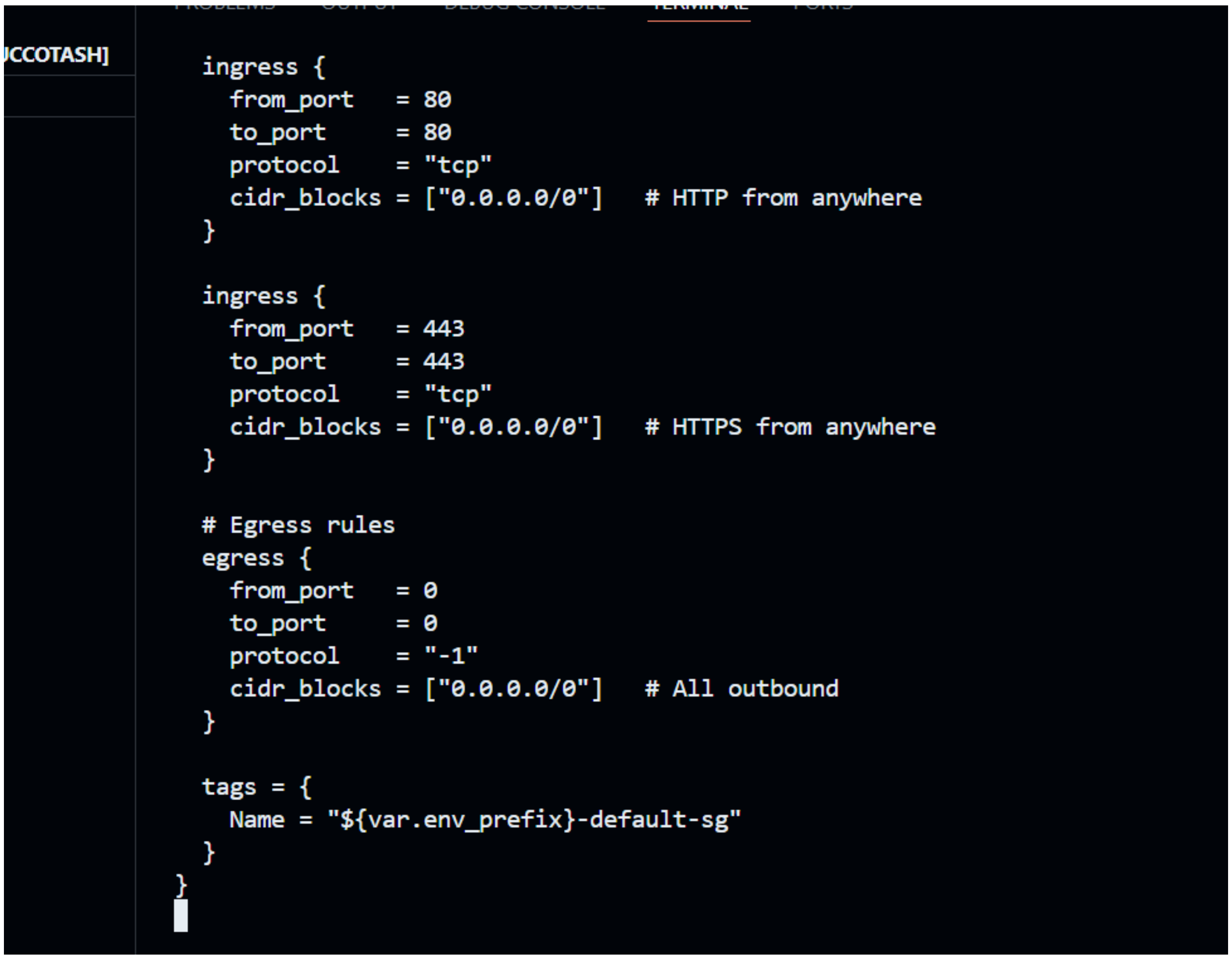


```
    }
  }
  # Get your current public IP from the web
  data "http" "my_ip" {
    url = "https://icanhazip.com"
  }

  # Convert the raw response into a /32 CIDR
  locals {
    my_ip = "${chomp(data.http.my_ip.response_body)}/32"
  }
```

Configure the default security group in the VPC

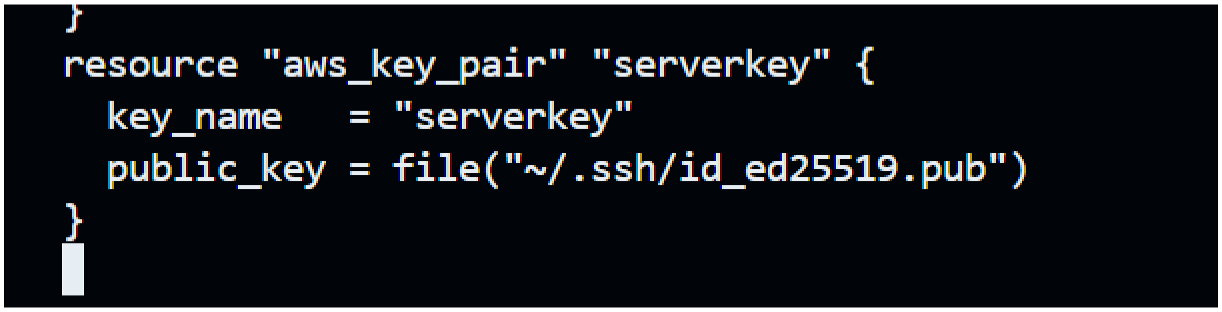


```
resource "aws_default_security_group" "default_sg" {
  vpc_id = aws_vpc.myapp_vpc.id

  # Ingress rules
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [local.my_ip]   # Only your IP
  }

  ingress {
    from_port   = 80
```



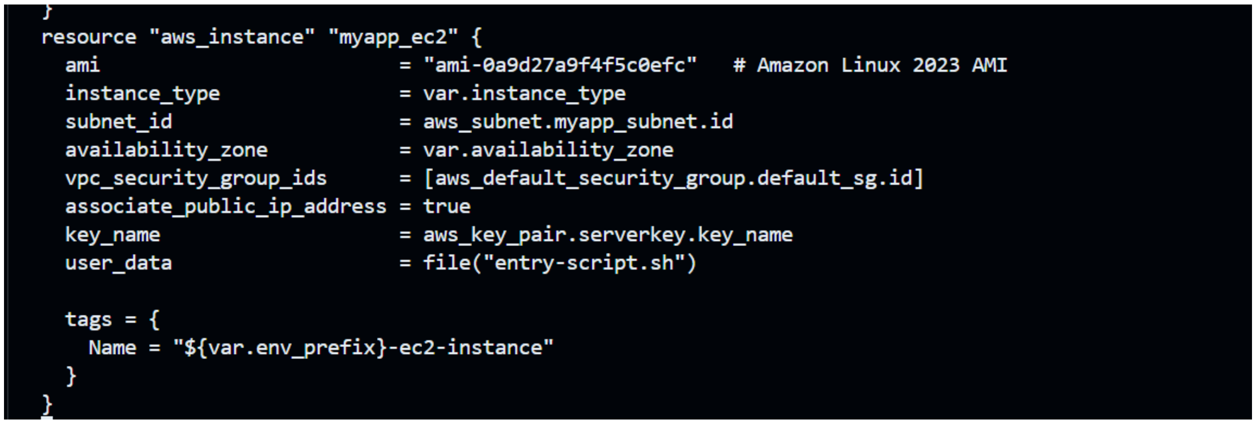
```
  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]   # HTTP from anywhere
  }

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

  # Egress rules
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]   # All outbound
  }

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

Create an AWS key pair for SSH



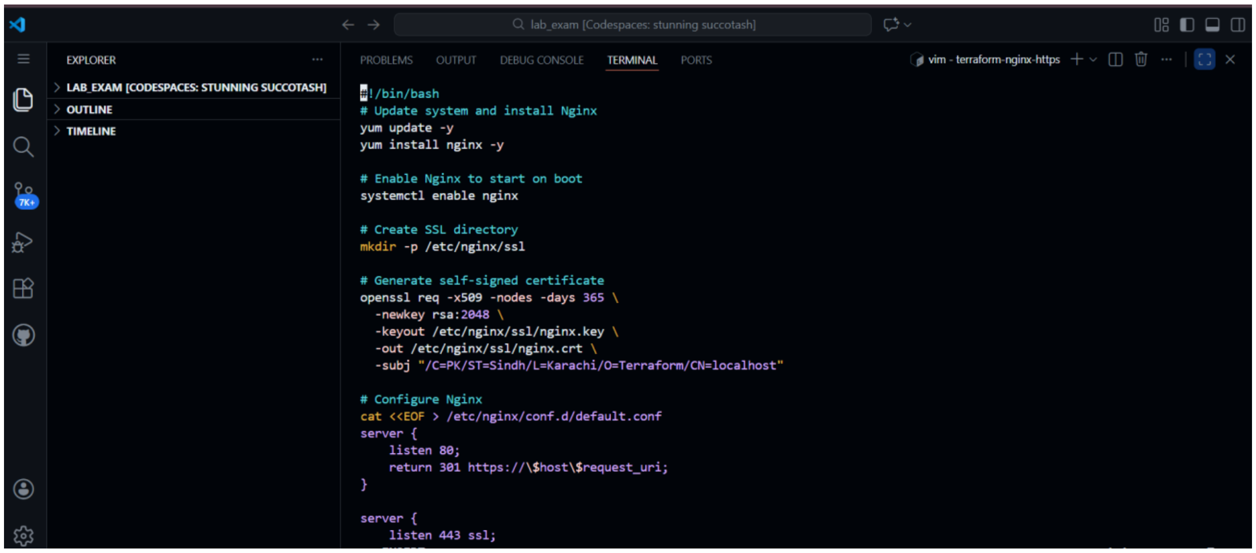
```
}
resource "aws_key_pair" "serverkey" {
  key_name   = "serverkey"
  public_key = file("~/.ssh/id_ed25519.pub")
}
```

Create the EC2 instance resource



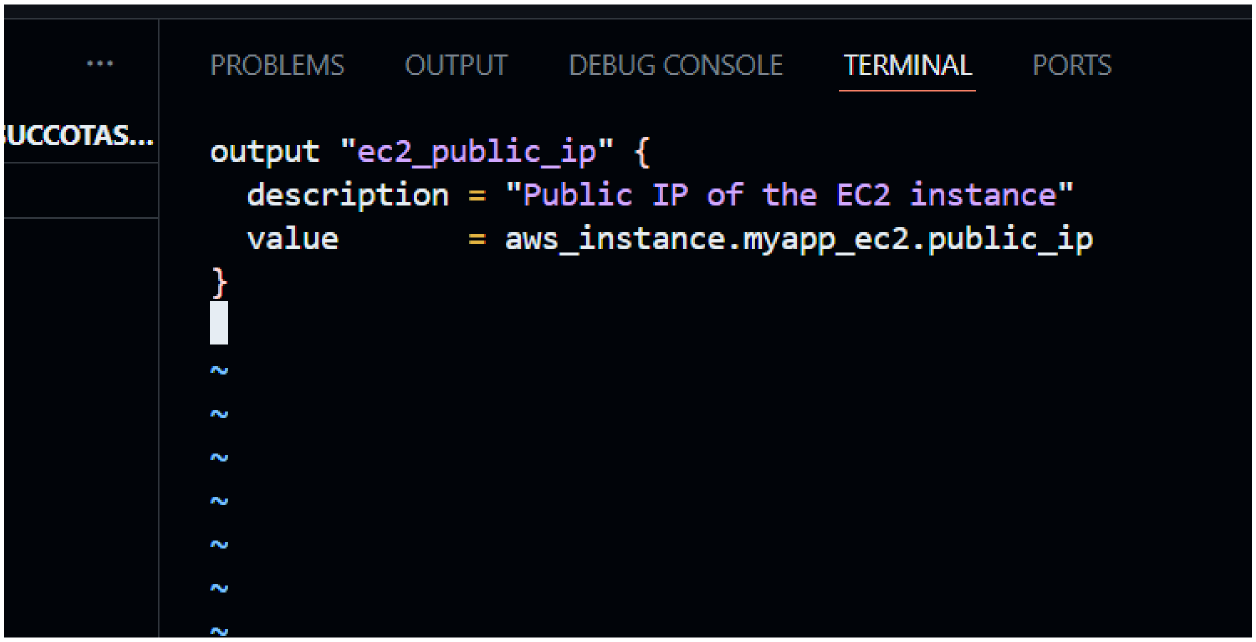
```
}
resource "aws_instance" "myapp_ec2" {
  ami                         = "ami-0a9d27a9f4f5c0efc"   # Amazon Linux 2023 AMI
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet.id
  availability_zone           = var.availability_zone
  vpc_security_group_ids      = [aws_default_security_group.default_sg.id]
  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"
  }
}
```

Create entry-script.sh to configure Nginx + HTTPS


lab_exam [Codespaces: stunning succotash]
EXPLORER
LAB_EXAM [CODESPACES: STUNNING SUCCOTASH]
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
vim - terraform-nginx-https

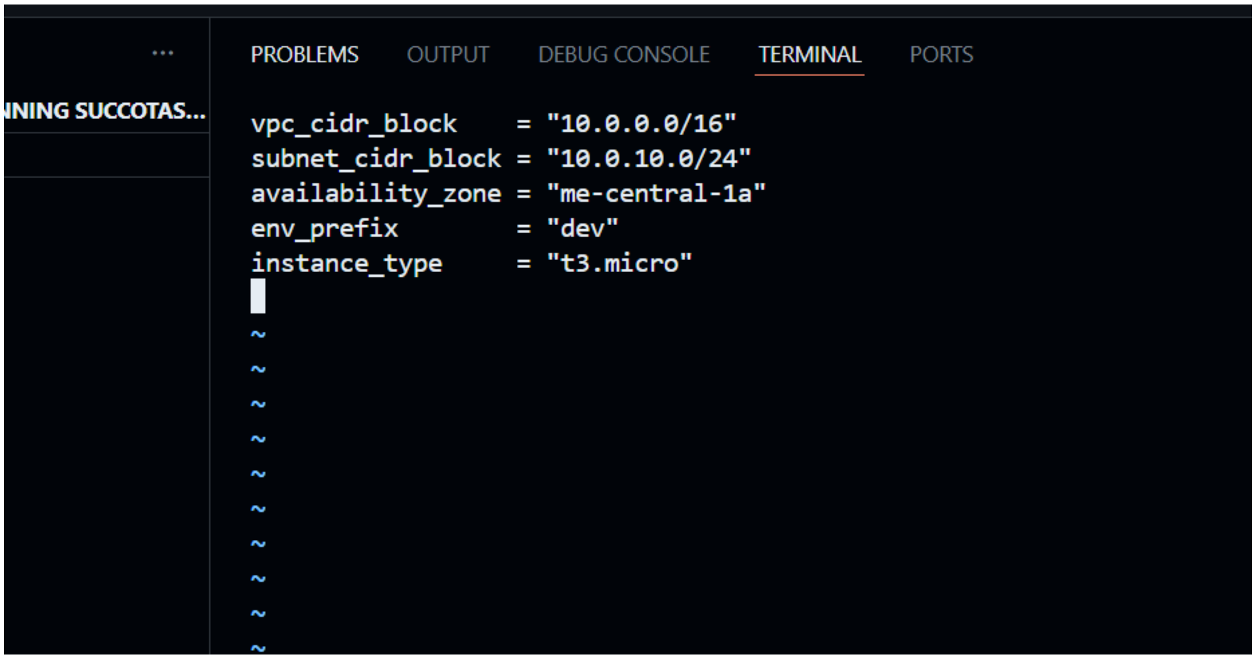
```
#!/bin/bash
# Update system and install Nginx
yum update -y
yum install nginx -y

# Enable Nginx to start on boot
systemctl enable nginx

# Create SSL directory
mkdir -p /etc/nginx/ssl

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

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

server {
    listen 443 ssl;
```

Add Terraform output for public IP



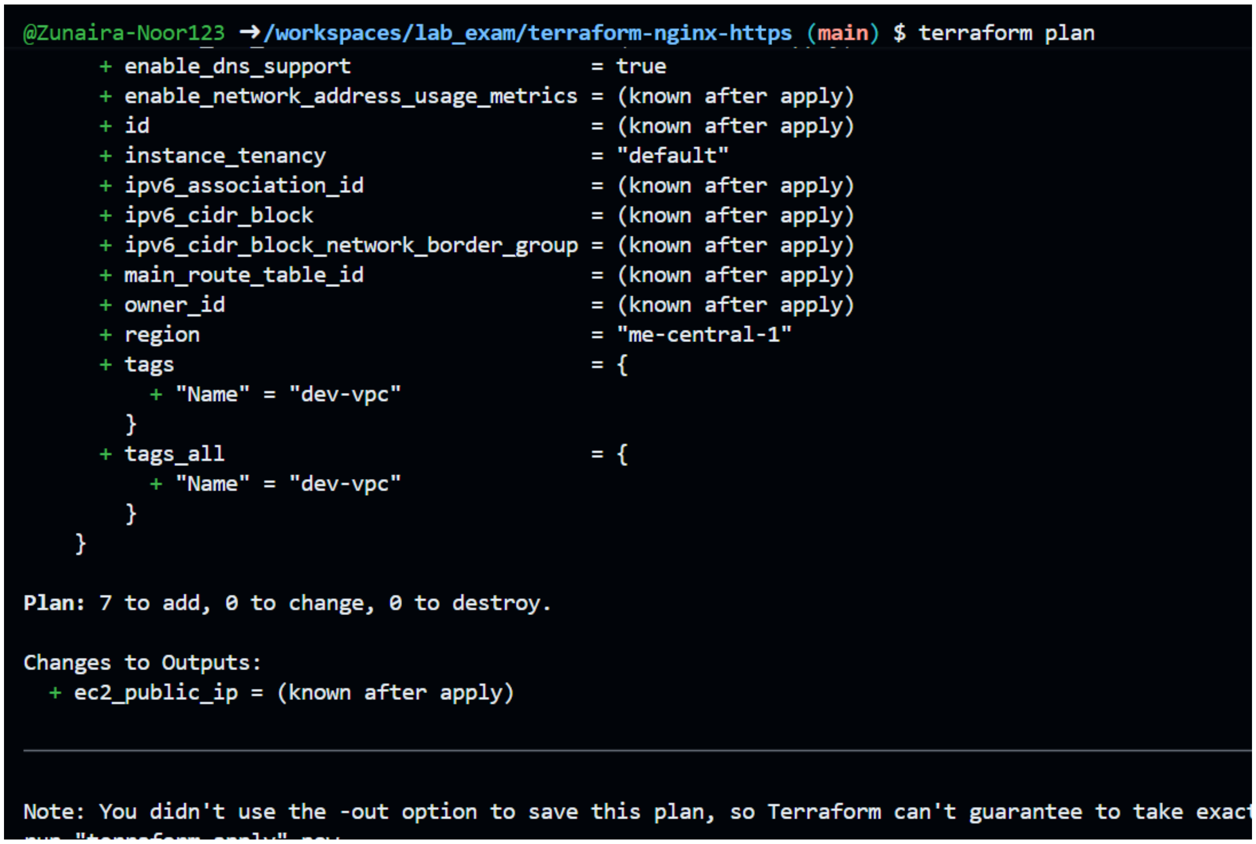
```
output "ec2_public_ip" {
  description = "Public IP of the EC2 instance"
  value       = aws_instance.myapp_ec2.public_ip
}

~
~
~
~
~
~
~
```

Set variable values for apply time

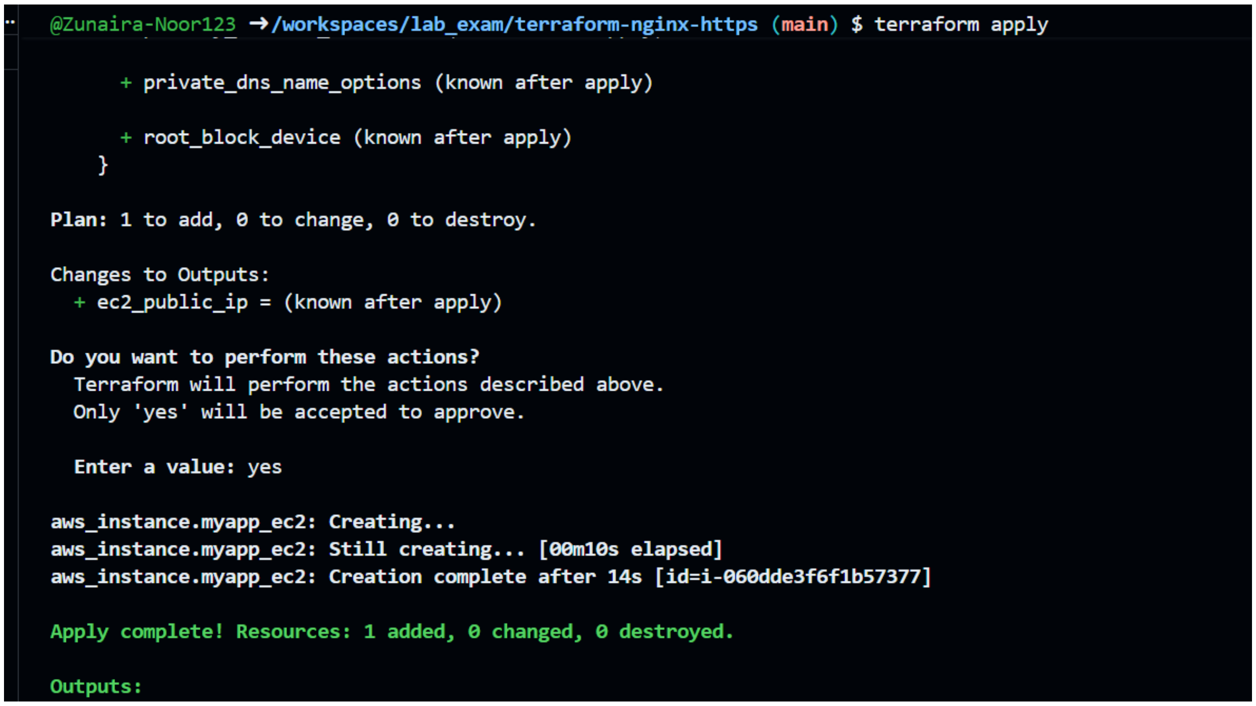


```
vpc_cidr_block    = "10.0.0.0/16"
subnet_cidr_block = "10.0.10.0/24"
availability_zone = "me-central-1a"
env_prefix        = "dev"
instance_type     = "t3.micro"

~
~
~
~
~
~
~
~
~
~
```

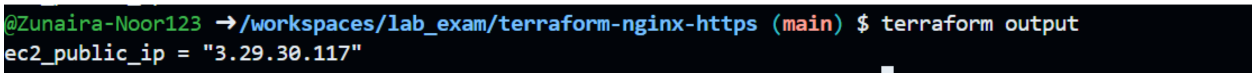
Run Terraform commands and capture outputs



```
@Zunaira-Noor123 ➜/workspaces/lab_exam/terraform-nginx-https (main) $ terraform plan
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + 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)

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exac
```



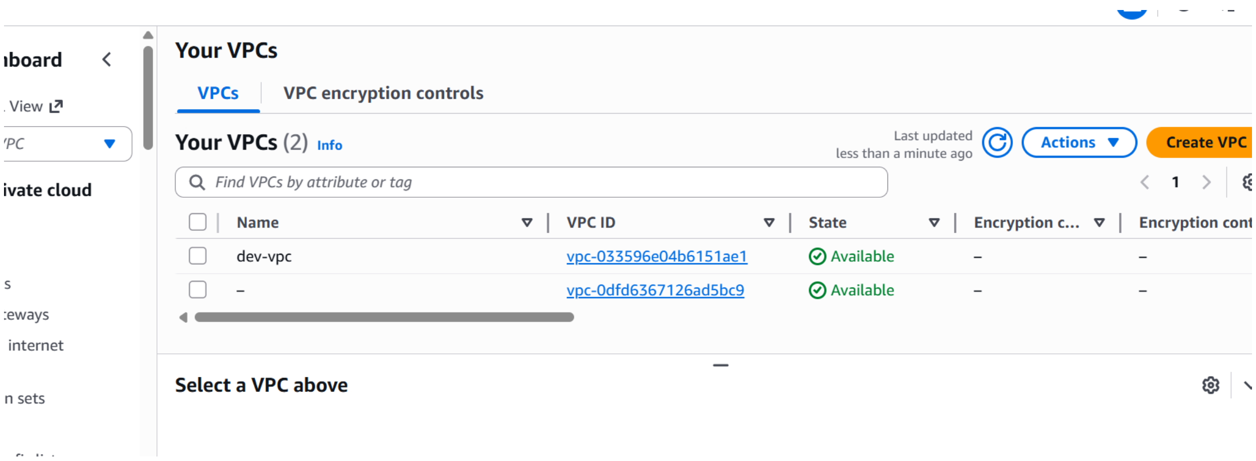
```
@Zunaira-Noor123 ➜/workspaces/lab_exam/terraform-nginx-https (main) $ terraform apply

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

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

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.myapp_ec2: Creating...
aws_instance.myapp_ec2: Still creating... [00m10s elapsed]
aws_instance.myapp_ec2: Creation complete after 14s [id=i-060dde3f6f1b57377]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:
```



```
@Zunaira-Noor123 ➜/workspaces/lab_exam/terraform-nginx-https (main) $ terraform output
ec2_public_ip = "3.29.30.117"
```

Verify Terraform resources in AWS console

VPC and Subnet



Your VPCs

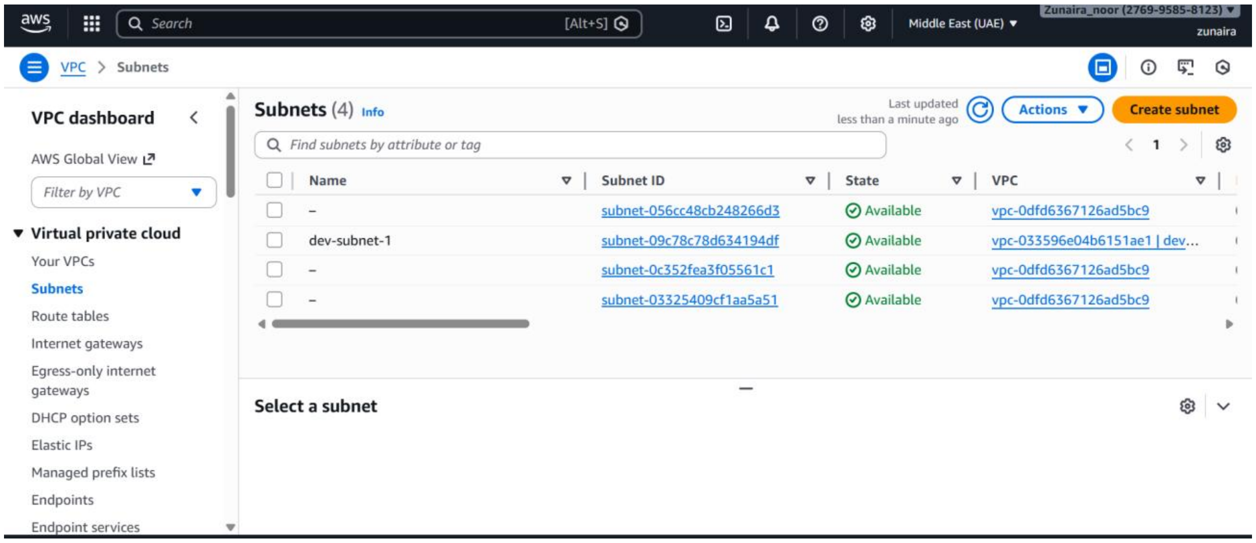
VPCs | VPC encryption controls

Your VPCs (2) Info

Last updated less than a minute ago

	Name	VPC ID	State	Encryption c...	Encryption cont
	dev-vpc	vpc-033596e04b6151ae1	Available	–	–
	–	vpc-0dfd6367126ad5bc9	Available	–	–

Select a VPC above



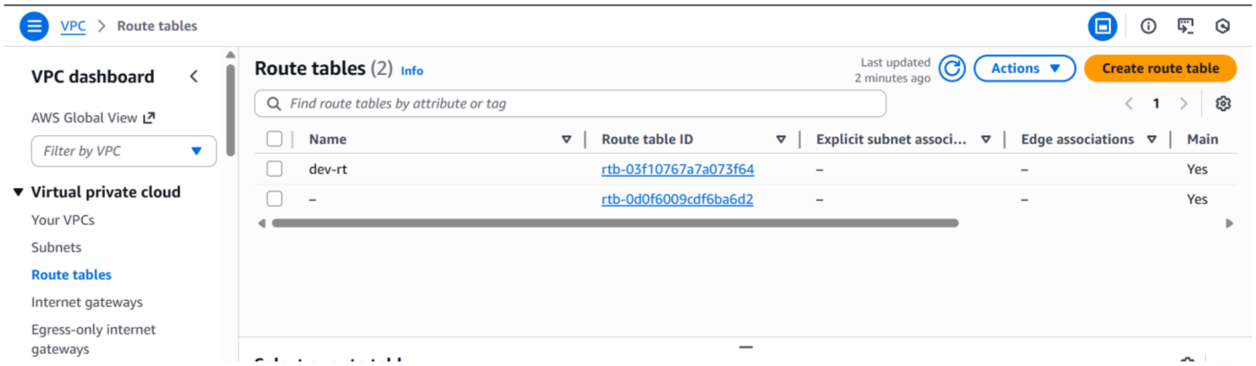
Subnets (4) Info

Last updated less than a minute ago

	Name	Subnet ID	State	VPC
	–	subnet-056cc48cb248266d3	Available	vpc-0dfd6367126ad5bc9
	dev-subnet-1	subnet-09c78c78d634194df	Available	vpc-033596e04b6151ae1 | dev...
	–	subnet-0c352fea3f05561c1	Available	vpc-0dfd6367126ad5bc9
	–	subnet-03325409cf1aa5a51	Available	vpc-0dfd6367126ad5bc9

Select a subnet

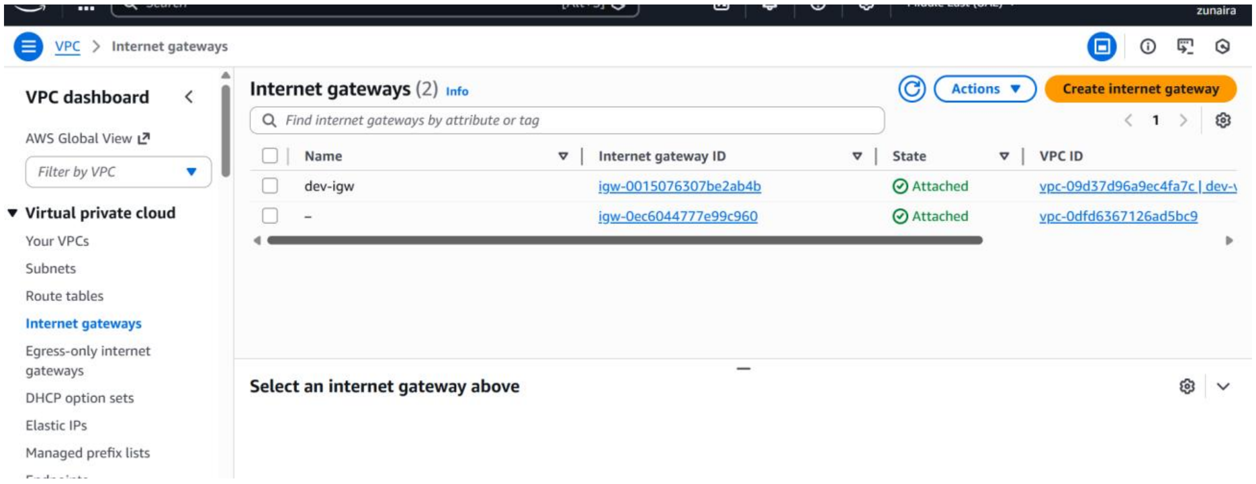
Internet Gateway and Route Table



Route tables (2) Info

Last updated 2 minutes ago

	Name	Route table ID	Explicit subnet associ...	Edge associations	Main
	dev-rt	rtb-03f10767a7a073f64	–	–	Yes
	–	rtb-0d0f6009cdf6ba6d2	–	–	Yes



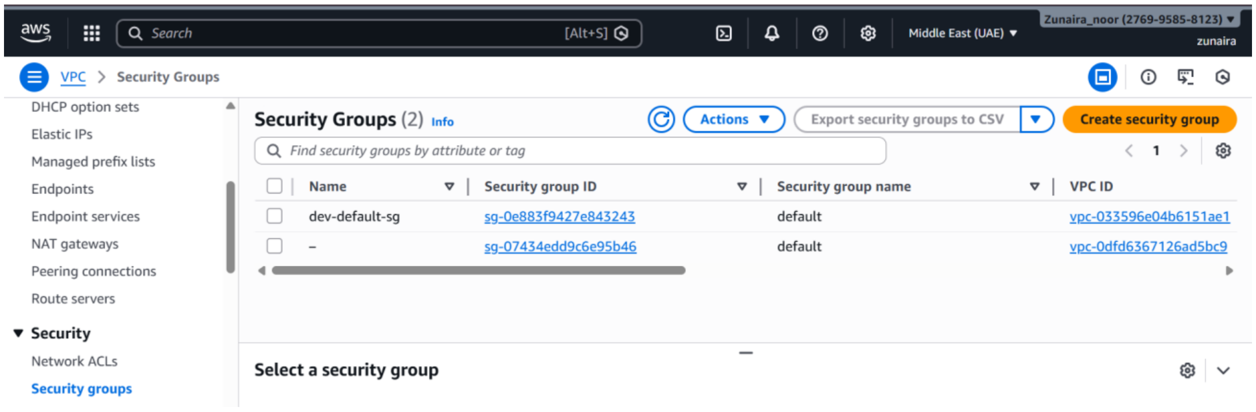
VPC > Internet gateways

Internet gateways (2) Info

Name	Internet gateway ID	State	VPC ID
dev-igw	igw-0015076307be2ab4b	Attached	vpc-09d37d96a9ec4fa7c | dev-v
–	igw-0ec6044777e99c960	Attached	vpc-0dfd6367126ad5bc9

Select an internet gateway above

Security Group



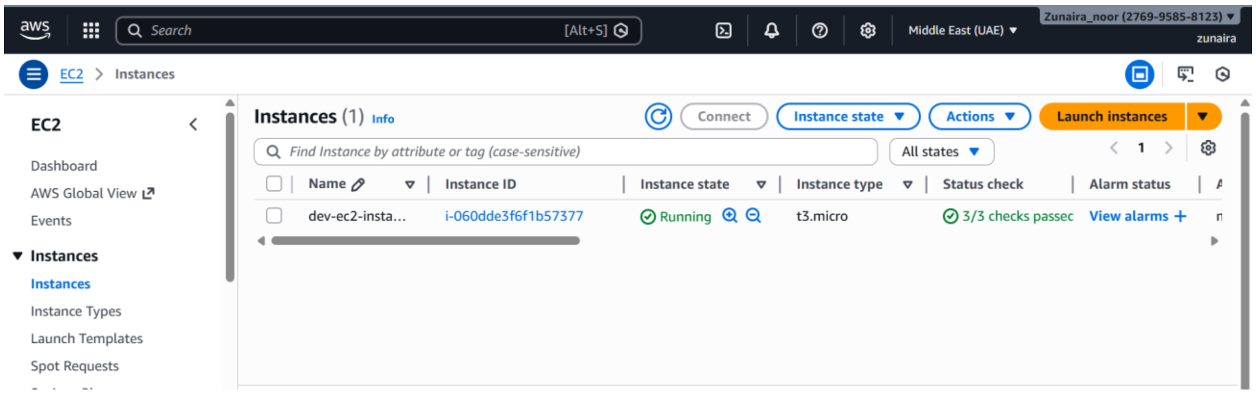
VPC > Security Groups

Security Groups (2) Info

Name	Security group ID	Security group name	VPC ID
dev-default-sg	sg-0e883f9427e843243	default	vpc-033596e04b6151ae1
–	sg-07434edd9c6e95b46	default	vpc-0dfd6367126ad5bc9

Select a security group

EC2 instance

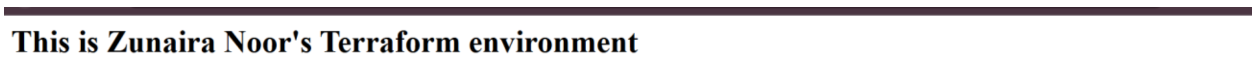


EC2 > Instances

Instances (1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
dev-ec2-insta…	i-060dde3f6f1b57377	Running	t3.micro	3/3 checks passec	View alarms +

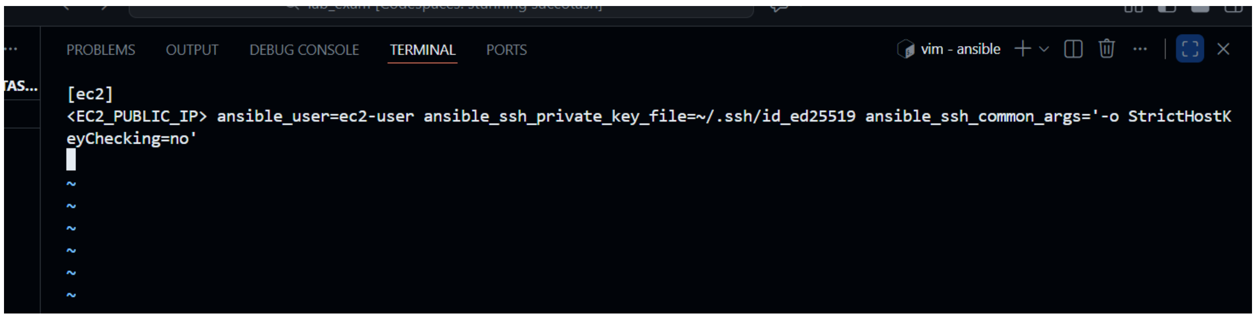
Verify HTTPS access from browser



This is Zunaira Noor's Terraform environment

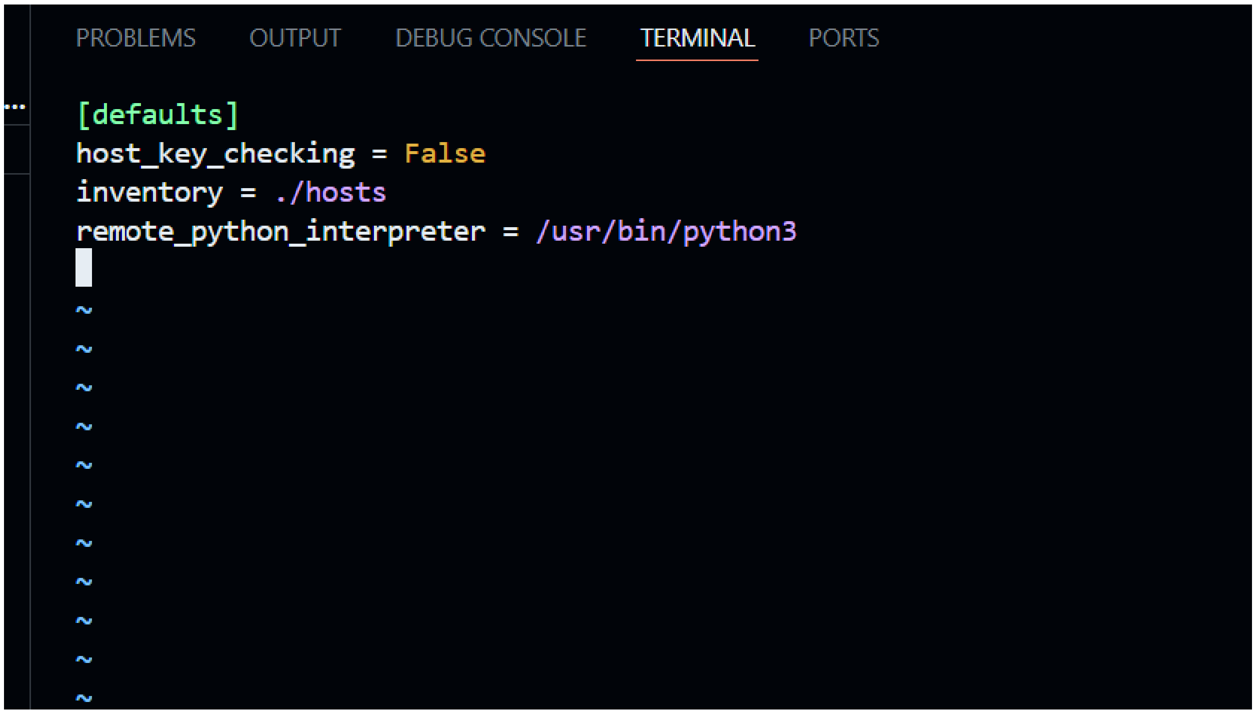
Q3 – Ansible Playbook for EC2 Web Server Using Q2 Instance (10 marks)

Create Ansible inventory file hosts



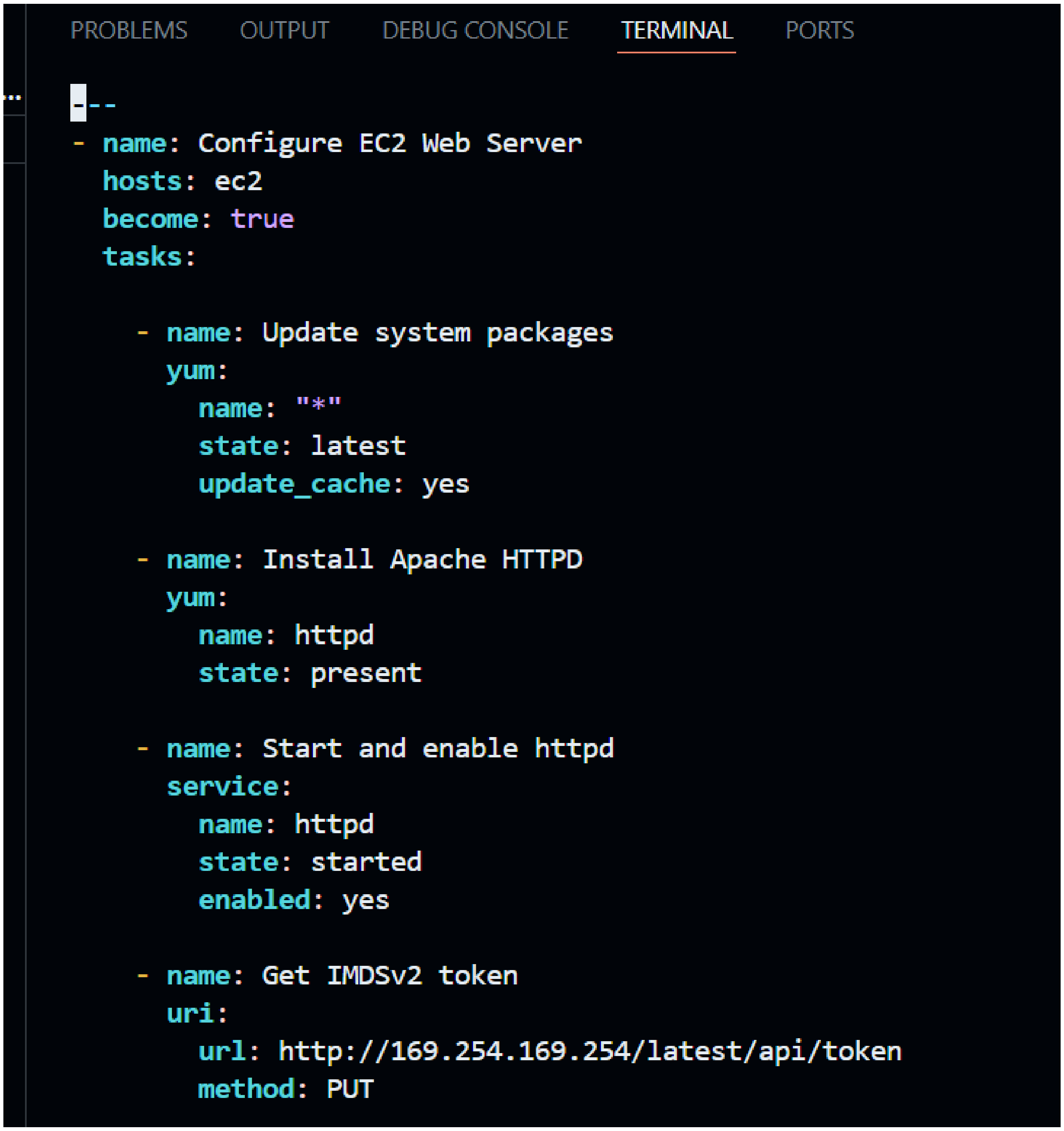
```
[ec2]
<EC2_PUBLIC_IP> ansible_user=ec2-user ansible_ssh_private_key_file=~/.ssh/id_ed25519 ansible_ssh_common_args='-o StrictHostK
eyChecking=no'

~
~
~
~
~
~
```

Create project-level ansible.cfg



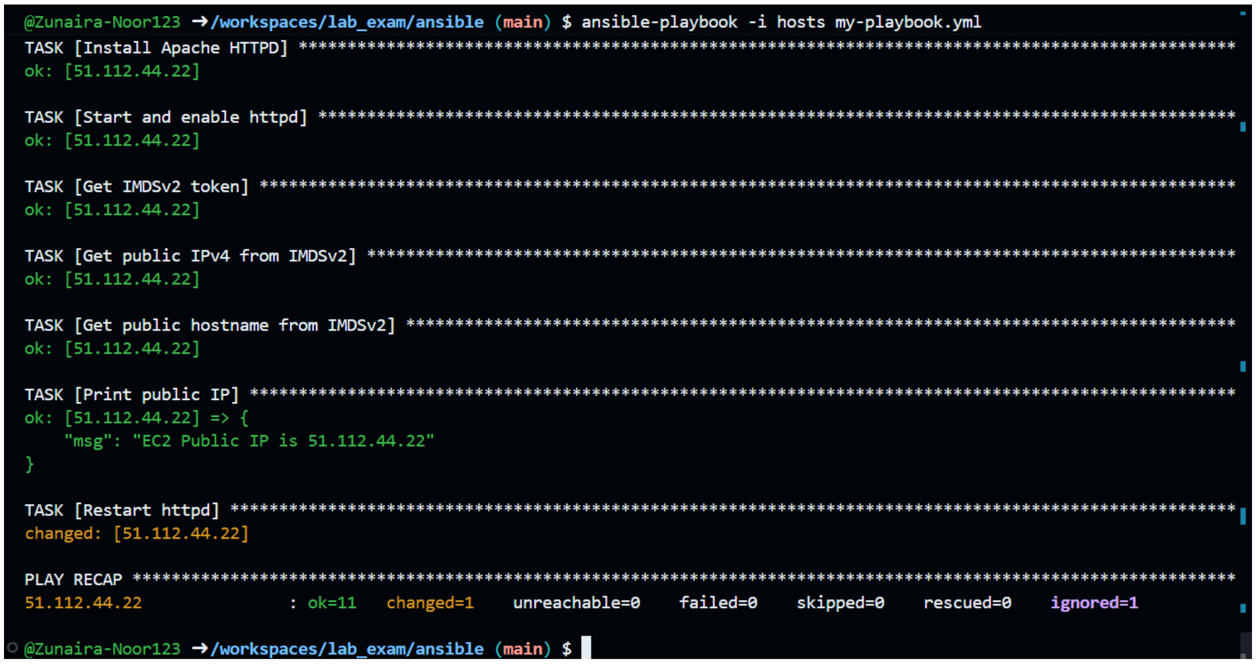
```
[defaults]
host_key_checking = False
inventory = ./hosts
remote_python_interpreter = /usr/bin/python3

~
~
~
~
~
~
~
~
~
~
~
```

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

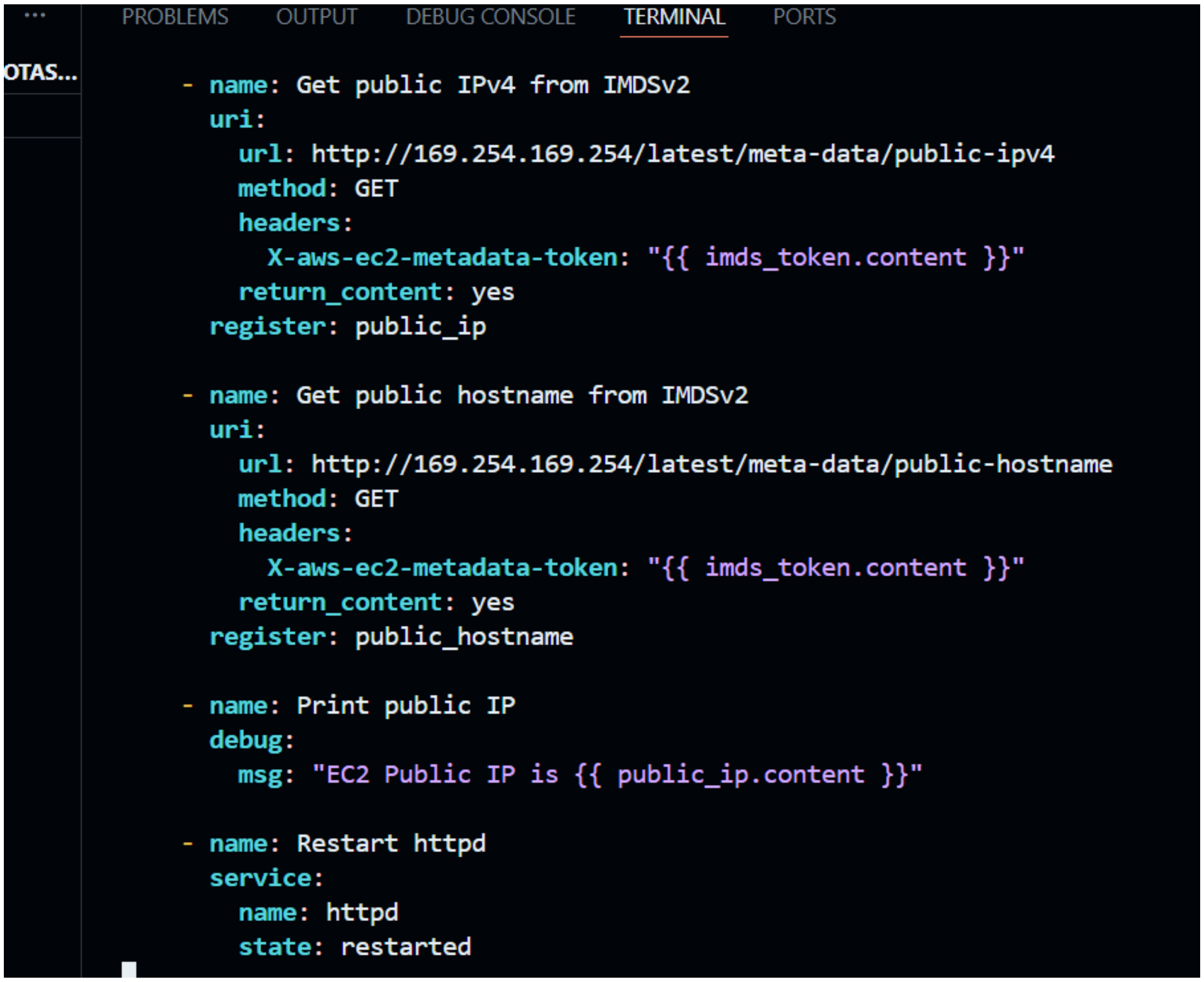


```
---
- name: Configure EC2 Web Server
  hosts: ec2
  become: true
  tasks:

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

    - 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
```

Run the Ansible playbook



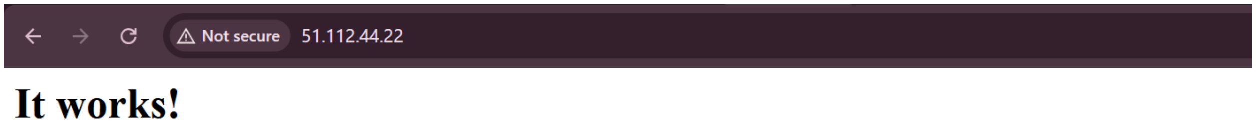
```
@Zunaira-Noor123 ➜ /workspaces/lab_exam/ansible (main) $ ansible-playbook -i hosts my-playbook.yml
TASK [Install Apache HTTPD] ***************************************************************************************
ok: [51.112.44.22]

TASK [Start and enable httpd] *************************************************************************************
ok: [51.112.44.22]

TASK [Get IMDSv2 token] *******************************************************************************************
ok: [51.112.44.22]

TASK [Get public IPv4 from IMDSv2] ********************************************************************************
ok: [51.112.44.22]

TASK [Get public hostname from IMDSv2] ****************************************************************************
ok: [51.112.44.22]

TASK [Print public IP] ********************************************************************************************
ok: [51.112.44.22] => {
    "msg": "EC2 Public IP is 51.112.44.22"
}

TASK [Restart httpd] **********************************************************************************************
changed: [51.112.44.22]

PLAY RECAP ********************************************************************************************************
51.112.44.22               : ok=11   changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1

@Zunaira-Noor123 ➜ /workspaces/lab_exam/ansible (main) $
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
- name: Get public IPv4 from IMDSv2
  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_ip

- name: Get public hostname from IMDSv2
  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: "EC2 Public IP is {{ public_ip.content }}"

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

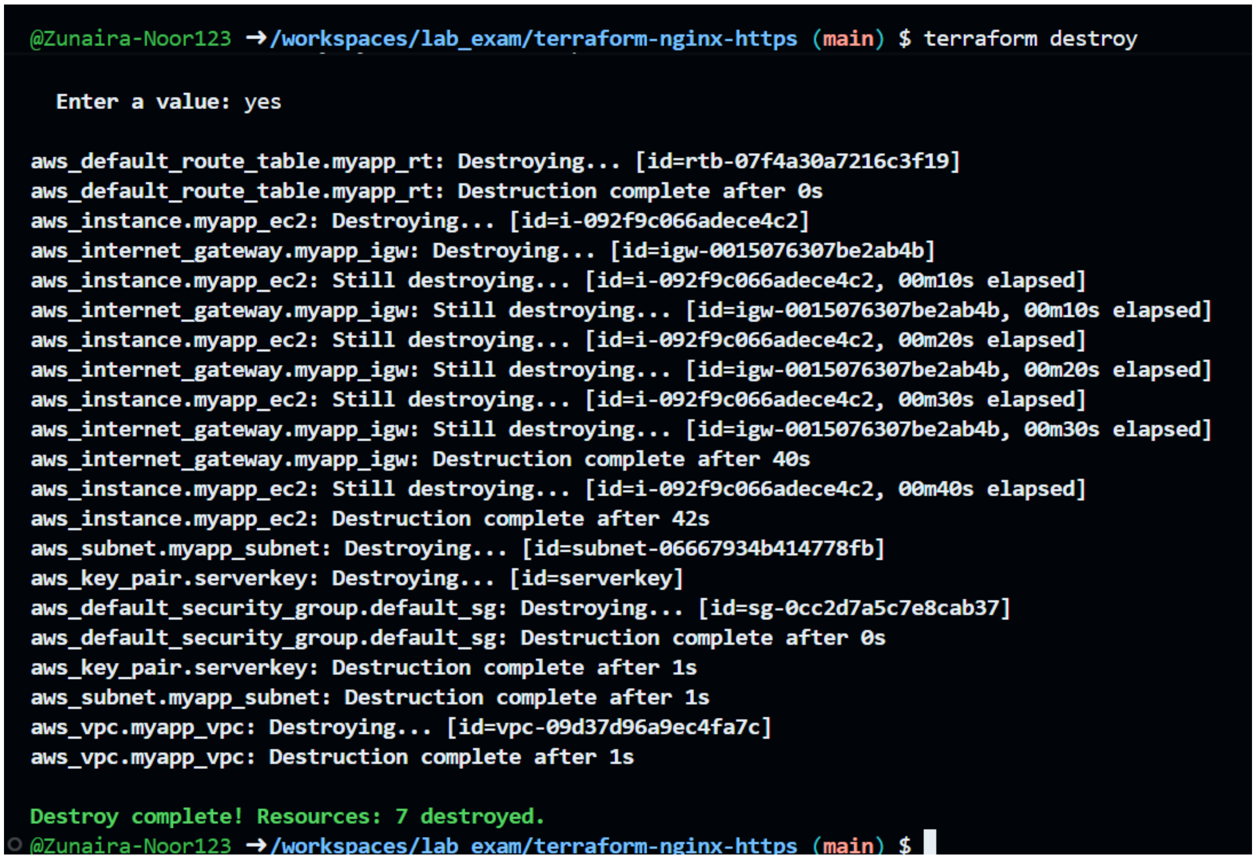
Verify HTTP access



It works!

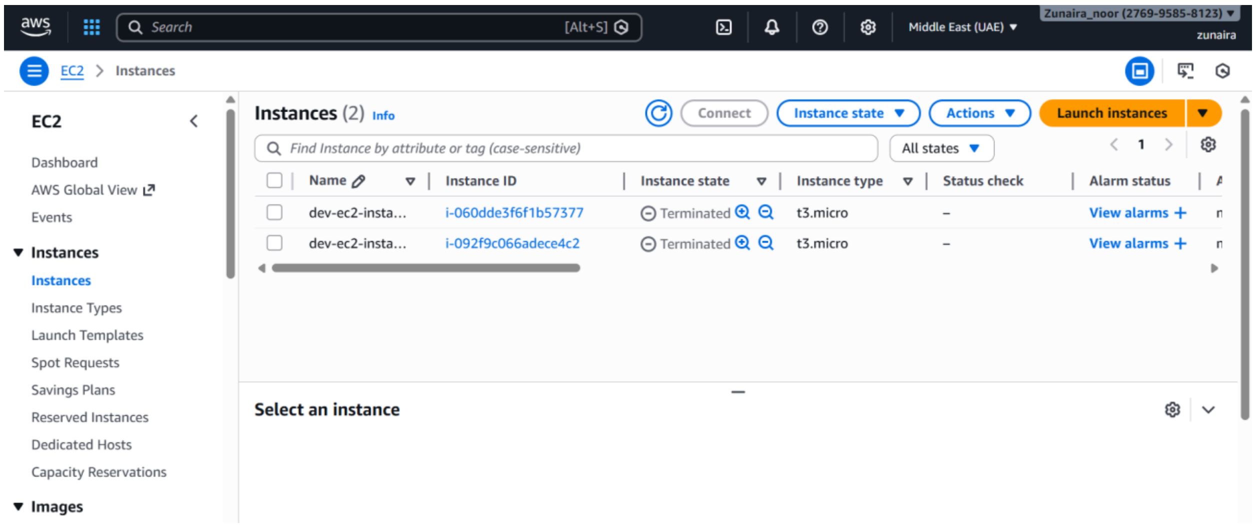
Cleanup (ungraded)

From your Terraform project directory (Q2):



```
@Zunaira-Noor123 →/workspaces/lab_exam/terraform-nginx-https (main) $ terraform destroy

  Enter a value: yes

aws_default_route_table.myapp_rt: Destroying... [id=rtb-07f4a30a7216c3f19]
aws_default_route_table.myapp_rt: Destruction complete after 0s
aws_instance.myapp_ec2: Destroying... [id=i-092f9c066adece4c2]
aws_internet_gateway.myapp_igw: Destroying... [id=igw-0015076307be2ab4b]
aws_instance.myapp_ec2: Still destroying... [id=i-092f9c066adece4c2, 00m10s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0015076307be2ab4b, 00m10s elapsed]
aws_instance.myapp_ec2: Still destroying... [id=i-092f9c066adece4c2, 00m20s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0015076307be2ab4b, 00m20s elapsed]
aws_instance.myapp_ec2: Still destroying... [id=i-092f9c066adece4c2, 00m30s elapsed]
aws_internet_gateway.myapp_igw: Still destroying... [id=igw-0015076307be2ab4b, 00m30s elapsed]
aws_internet_gateway.myapp_igw: Destruction complete after 40s
aws_instance.myapp_ec2: Still destroying... [id=i-092f9c066adece4c2, 00m40s elapsed]
aws_instance.myapp_ec2: Destruction complete after 42s
aws_subnet.myapp_subnet: Destroying... [id=subnet-06667934b414778fb]
aws_key_pair.serverkey: Destroying... [id=serverkey]
aws_default_security_group.default_sg: Destroying... [id=sg-0cc2d7a5c7e8cab37]
aws_default_security_group.default_sg: Destruction complete after 0s
aws_key_pair.serverkey: Destruction complete after 1s
aws_subnet.myapp_subnet: Destruction complete after 1s
aws_vpc.myapp_vpc: Destroying... [id=vpc-09d37d96a9ec4fa7c]
aws_vpc.myapp_vpc: Destruction complete after 1s

Destroy complete! Resources: 7 destroyed.
@Zunaira-Noor123 →/workspaces/lab_exam/terraform-nginx-https (main) $
```

In AWS console, verify that no lab-related EC2 instances remain running.



Zunaira_noor (2769-9585-8123)
zunaira
Middle East (UAE)

EC2 > Instances

Instances (2) Info

Find Instance by attribute or tag (case-sensitive)   All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
dev-ec2-insta…	i-060dde3f6f1b57377	Terminated	t3.micro	–	View alarms +
dev-ec2-insta…	i-092f9c066adece4c2	Terminated	t3.micro	–	View alarms +

Select an instance