CC Lab Exam
Name: Anara Hayat
Reg#No: 2023-BSE-008

```
Default region name [None]: me-central-1
Default output format [None]: json
@Anara-hayat → /workspaces/Lab_exam (main) $ aws iam create-user --user-name anara
{
    "User": {
        "Path": "/",
        "UserName": "anara",
        "UserId": "AIDATUFNGT6BF6SHLWGN6",
        "Arn": "arn:aws:iam::249471344514:user/anara",
        "CreateDate": "2026-01-19T07:55:53+00:00"
    }
}
@Anara-hayat → /workspaces/Lab_exam (main) $ aws iam create-group --group-name softwareengineering
{
    "Group": {
        "Path": "/",
        "GroupName": "softwareengineering",
        "GroupId": "AGPATUFNGT6BF3E5JOINE",
        "Arn": "arn:aws:iam::249471344514:group/softwareengineering",
        "CreateDate": "2026-01-19T07:56:25+00:00"
    }
}
@Anara-hayat → /workspaces/Lab_exam (main) $
```

```
}
@Anara-hayat → /workspaces/Lab_exam (main) $ aws iam get-group --group-name softwareengineering
{
    "Users": [],
    "Group": {
        "Path": "/",
        "GroupName": "softwareengineering",
        "GroupId": "AGPATUFNGT6BF3E5JOINE",
        "Arn": "arn:aws:iam::249471344514:group/softwareengineering",
        "CreateDate": "2026-01-19T07:56:25+00:00"
    }
}
@Anara-hayat → /workspaces/Lab_exam (main) $
```

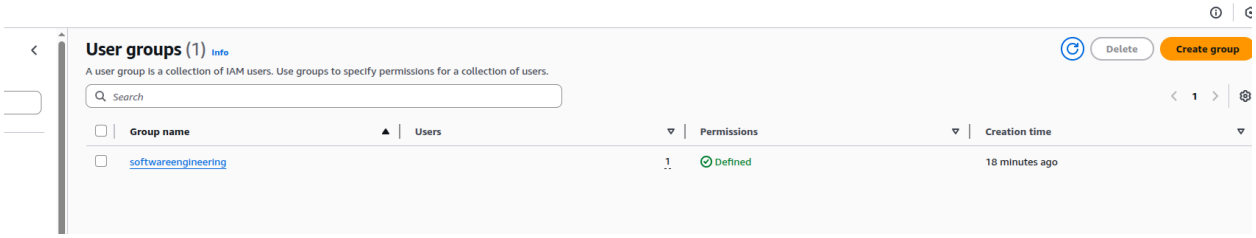
```
    }
}
@Anara-hayat → /workspaces/Lab_exam (main) $ aws iam get-user --user-name anara
{
    "User": {
        "Path": "/",
        "UserName": "anara",
        "UserId": "AIDATUFNGT6BF6SHLWGN6",
        "Arn": "arn:aws:iam::249471344514:user/anara",
        "CreateDate": "2026-01-19T07:55:53+00:00"
    }
}
@Anara-hayat → /workspaces/Lab_exam (main) $
```

```
   aws <command> <subcommand> help

@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam add-user-to-group --user-name anara --group-name softwareengineering
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam get-group --group-name softwareengineering
{
    "Users": [
        {
            "Path": "/",
            "UserName": "anara",
            "UserId": "AIDATUFNGT6BF6SHLWGN6",
            "Arn": "arn:aws:iam::249471344514:user/anara",
            "CreateDate": "2026-01-19T07:55:53+00:00"
        }
    ],
    "Group": {
        "Path": "/",
        "GroupName": "softwareengineering",
        "GroupId": "AGPATUFNGT6BF3E5JOINE",
        "Arn": "arn:aws:iam::249471344514:group/softwareengineering",
        "CreateDate": "2026-01-19T07:56:25+00:00"
    }
}
@Anara-hayat ➜ /workspaces/Lab_exam (main) $
```

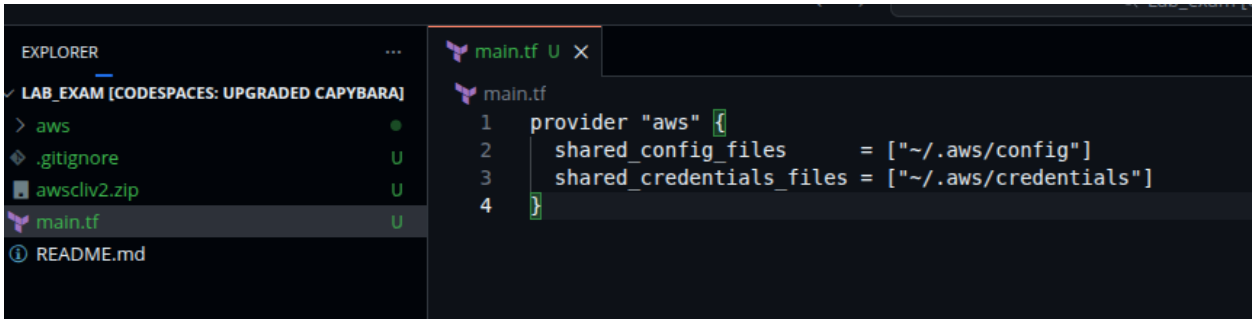
```
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam list-policies --query "Policies[?PolicyName=='AmazonEC2FullAccess' ].{Name:PolicyName, ARN:Arn}" --ou
tput text
arn:aws:iam::aws:policy/AmazonEC2FullAccess     AmazonEC2FullAccess
@Anara-hayat ➜ /workspaces/Lab_exam (main) $
```

```
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam list-policies --query "Policies[?PolicyName=='AmazonEC2FullAccess' ].{Name:PolicyName, ARN:Arn}" --ou
tput text
arn:aws:iam::aws:policy/AmazonEC2FullAccess     AmazonEC2FullAccess
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam attach-group-policy --group-name softwareengineering --policy-arn arn:aws:iam::aws:policy/AmazonEC2Fu
llAccess
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ aws iam list-attached-group-policies --group-name softwareengineering
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonEC2FullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"
        }
    ]
}
@Anara-hayat ➜ /workspaces/Lab_exam (main) $
```

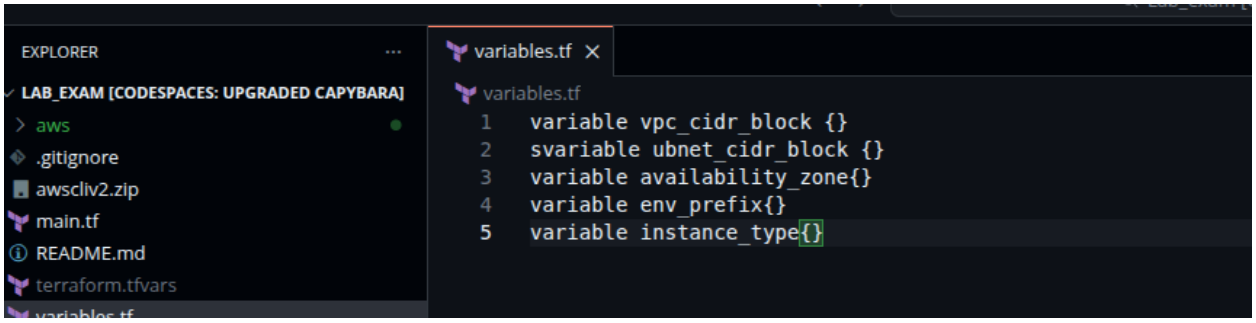


Q2

1



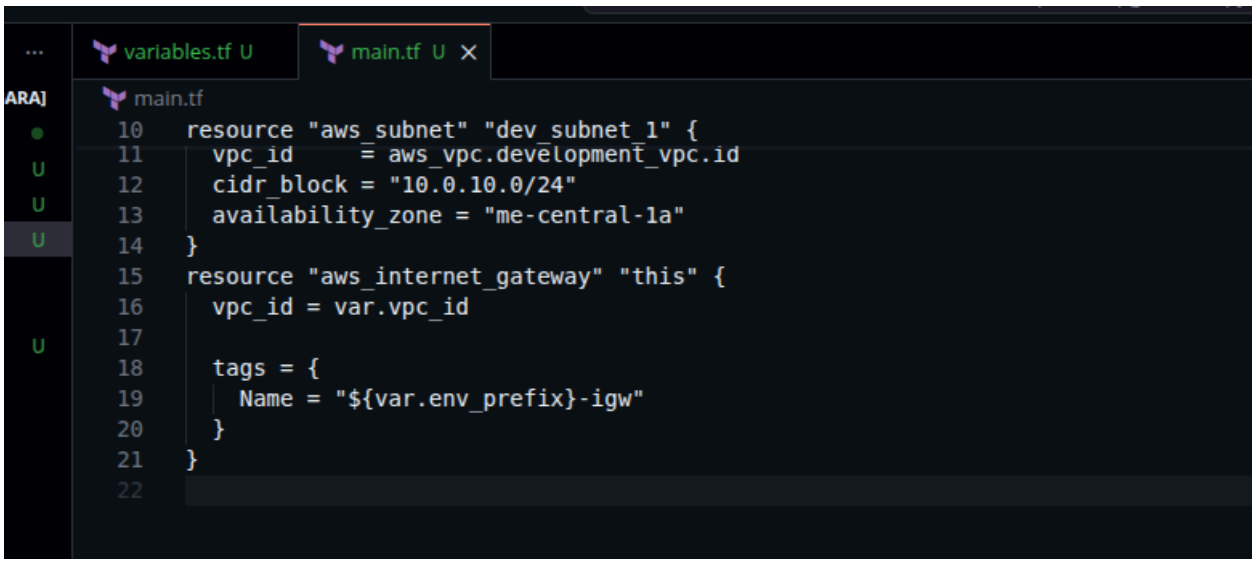
2


EXPLORER
LAB_EXAM [CODESPACES: UPGRADED CAPYBARA]
aws
.gitignore
awscliv2.zip
main.tf
README.md
terraform.tfvars
variables.tf
variables.tf
1 variable vpc_cidr_block {}
2 svariable ubnet_cidr_block {}
3 variable availability_zone{}
4 variable env_prefix{}
5 variable instance_type{}

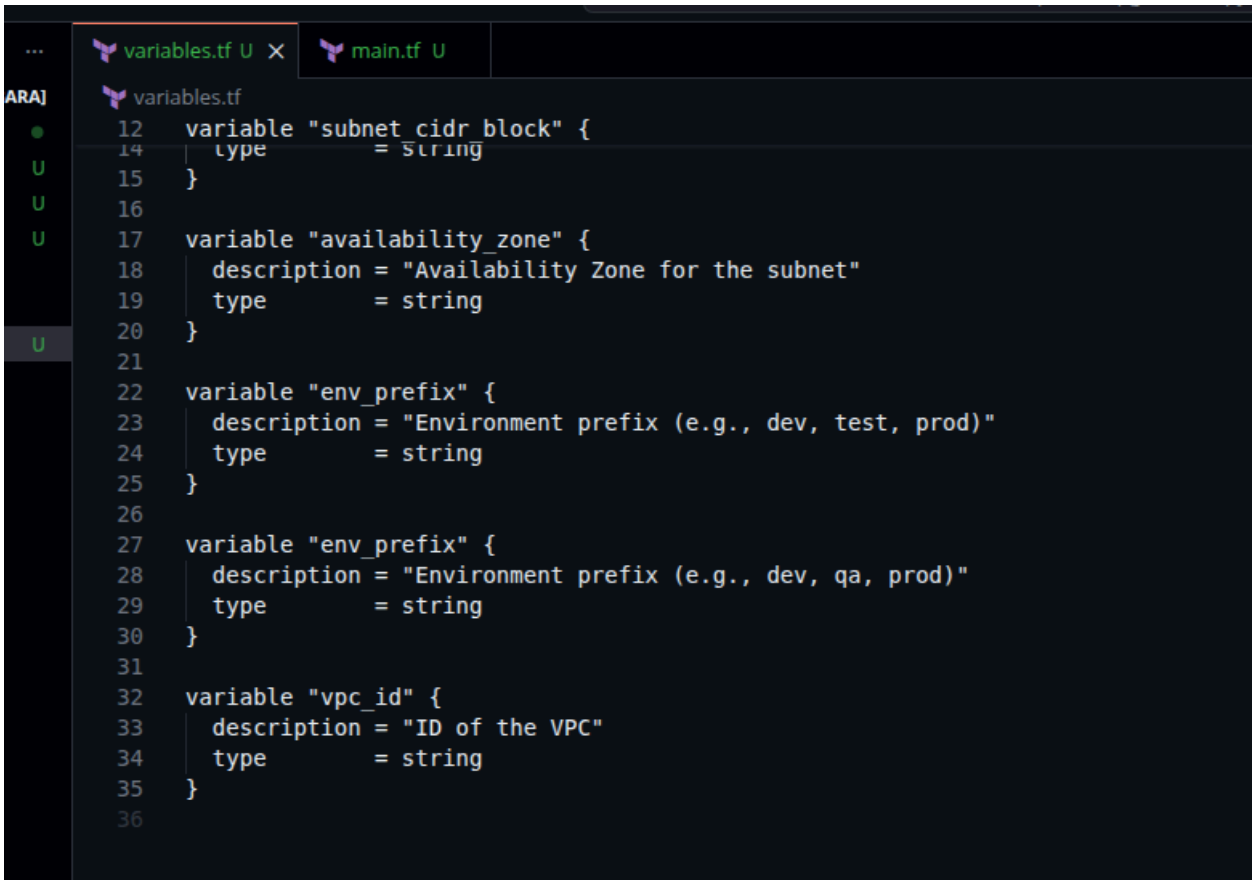
3


variables.tf U
main.tf U
main.tf
10 resource "aws_subnet" "myapp_subnet_1" {
11 vpc_id = aws_vpc.myapp_vpc.id
12 cidr_block = var.subnet_cidr_block
13 availability_zone = var.availability_zone
14
15 tags = {
16 Name = "${var.env_prefix}-subnet-1"
17 }
18 }
19 resource "aws_vpc" "myapp_vpc" {
20 cidr_block = var.vpc_cidr_block
21
22 tags = {
23 Name = "${var.env_prefix}-vpc"
24 }
25 }
26 resource "aws_internet_gateway" "this" {

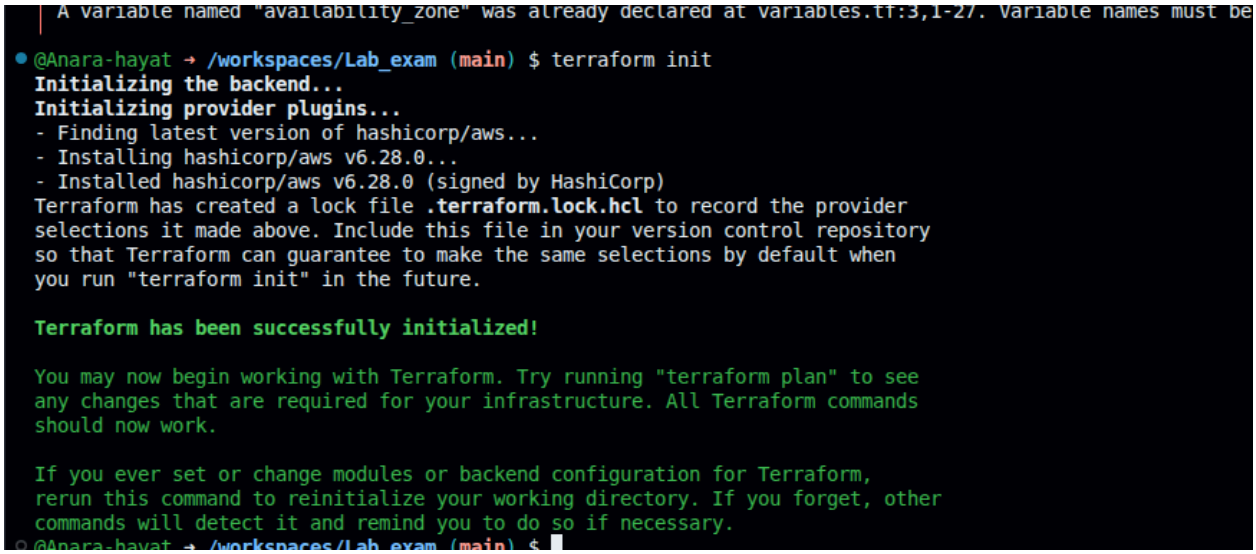
4


variables.tf U
main.tf U
main.tf
10 resource "aws_subnet" "dev_subnet_1" {
11 vpc_id = aws_vpc.development_vpc.id
12 cidr_block = "10.0.10.0/24"
13 availability_zone = "me-central-1a"
14 }
15 resource "aws_internet_gateway" "this" {
16 vpc_id = var.vpc_id
17
18 tags = {
19 Name = "${var.env_prefix}-igw"
20 }
21 }
22

5


variables.tf U
main.tf U
variables.tf
12 variable "subnet_cidr_block" {
14 type = string
15 }
16
17 variable "availability_zone" {
18 description = "Availability Zone for the subnet"
19 type = string
20 }
21
22 variable "env_prefix" {
23 description = "Environment prefix (e.g., dev, test, prod)"
24 type = string
25 }
26
27 variable "env_prefix" {
28 description = "Environment prefix (e.g., dev, qa, prod)"
29 type = string
30 }
31
32 variable "vpc_id" {
33 description = "ID of the VPC"
34 type = string
35 }
36

12


A variable named "availability_zone" was already declared at variables.tf:3,1-27. Variable names must be
@Anara-hayat → /workspaces/Lab_exam (main) $ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v6.28.0...
- Installed hashicorp/aws v6.28.0 (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,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

7

```
locals {
  my_ip = "${chomp(data.http.my_public_ip.response_body)}/32"
}
locals {
  effective_vpc_id = var.vpc_id != null ? var.vpc_id : aws_vpc.main.id
}
```

8

```
resource "aws_default_security_group" "this" {
  vpc_id = var.vpc_id

  # SSH — only from your public IP
  ingress {
    description = "SSH from my IP"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = [local.my_ip]
  }

  # HTTP — open to the world
  ingress {
    description = "HTTP"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  # HTTPS — open to the world
  ingress {
    description = "HTTPS"
    from_port   = 443
    to_port     = 443
```

```
@Anara-hayat ➜ /workspaces/Lab_exam (main) $ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/codespace/.ssh/id_ed25519):
/home/codespace/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/codespace/.ssh/id_ed25519
Your public key has been saved in /home/codespace/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:pzijqM1ALaUJZ8VrtSzEpuJ1ss/uyDeVbsEywUumuVc codespace@codespaces-d7e6b9
The key's randomart image is:
+--[ED25519 256]--+
|   o.            |
|   .= .          |
|. o=.+ .         |
|oo*o+*o          |
|.*.oO.+ S .      |
|...+ + E o       |
|.   + X o        |
| +.o.B =         |
|..+o*oo          |
+----[SHA256]-----+
```

9

```
resource "aws_instance" "ec2" {
  ami                    = "ami-0aeeebd8d2ab47354" # Amazon Linux 2023, hard-coded
  instance_type          = var.instance_type
  subnet_id              = aws_subnet.main.id      # replace with your subnet resource
  vpc_security_group_ids = [aws_default_security_group.this.id]
  availability_zone      = var.availability_zone
  key_name               = "serverkey"
  associate_public_ip_address = true

  # User data script
  user_data = file("entry-script.sh")

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

10

$ entry-script.sh

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

# Update the system
yum update -y

# Install Nginx and OpenSSL
yum install -y nginx openssl

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

# Generate a self-signed certificate valid for 1 year
openssl req -x509 -nodes -days 365 \
  -subj "/C=US/ST=State/L=City/O=Organization/CN=example.com" \
  -newkey rsa:2048 \
  -keyout /etc/nginx/ssl/selfsigned.key \
  -out /etc/nginx/ssl/selfsigned.crt

# Backup default Nginx config
cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup

```

11

variables.tf

```
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"

```