CLOUD COMPUTING LAB EXAM

Submitted to: Engr. Waqas Saleem

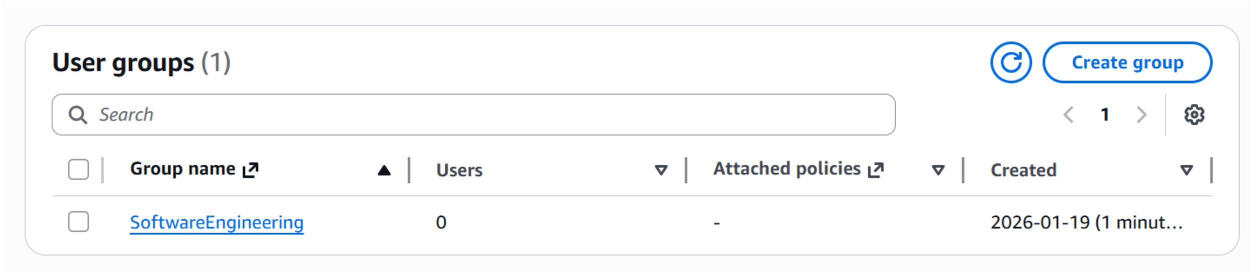
Engr. Muhammad Shahzad

Submitted by: Maham Saleem

Registration no.: 2023-BSE-035

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

q1_create_group.png

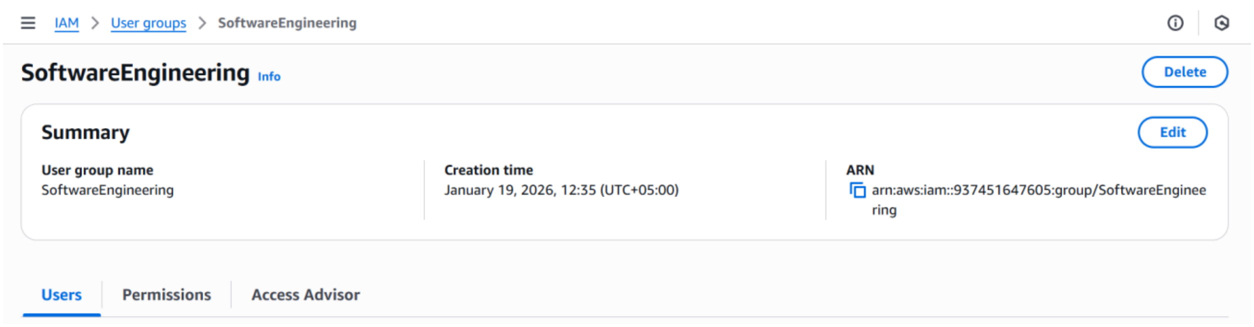


User groups (1)

Search

Group name	Users	Attached policies	Created
SoftwareEngineering	0	-	2026-01-19 (1 minut…

q1_group_details.png



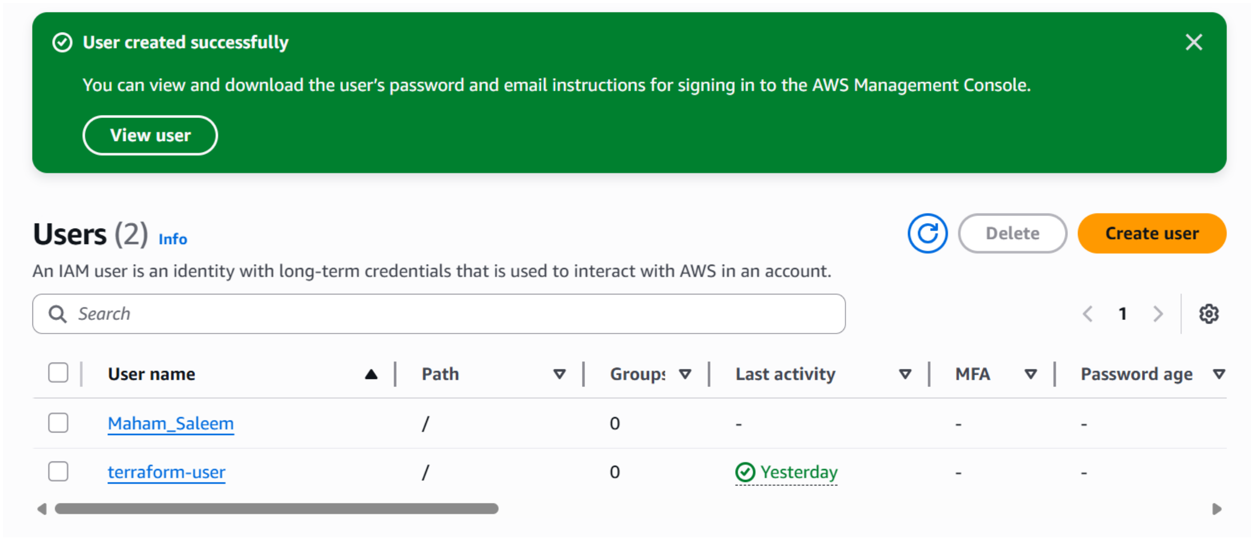
IAM > User groups > SoftwareEngineering

SoftwareEngineering Info

Summary

User group name	Creation time	ARN
SoftwareEngineering	January 19, 2026, 12:35 (UTC+05:00)	arn:aws:iam::937451647605:group/SoftwareEngineering

q1_create_user.png



User created successfully

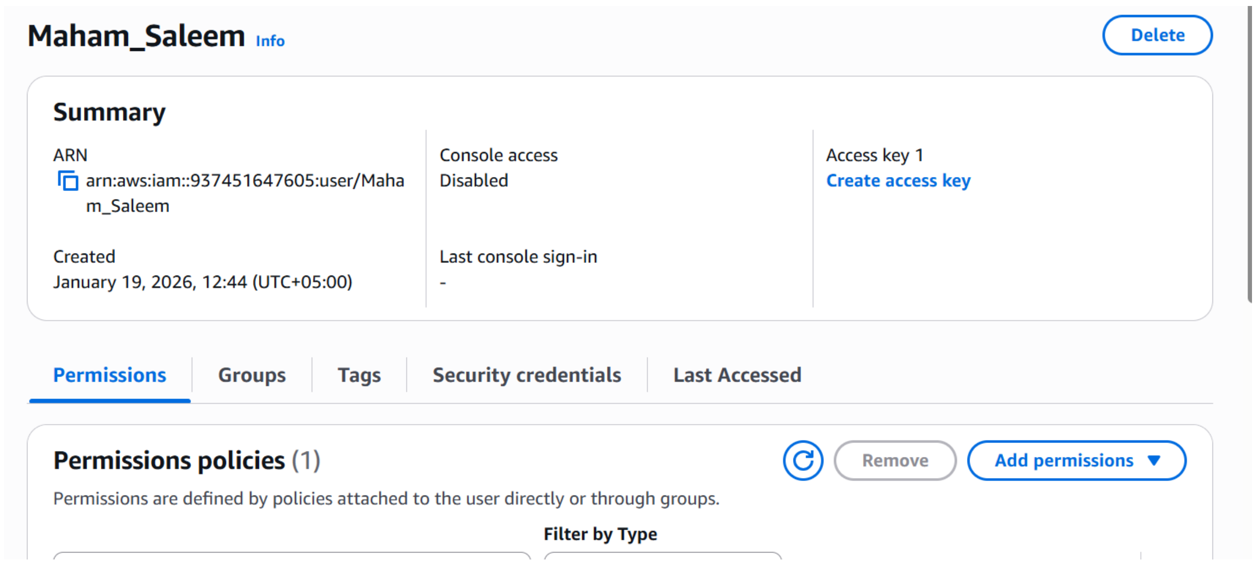
You can view and download the user's password and email instructions for signing in to the AWS Management Console.

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
Maham_Saleem	/	0	-	-	-
terraform-user	/	0	Yesterday	-	-

q1_user_details.png



Maham_Saleem Info

Delete

Summary

ARN
arn:aws:iam::937451647605:user/Maham_Saleem

Created
January 19, 2026, 12:44 (UTC+05:00)

Console access
Disabled

Last console sign-in
-

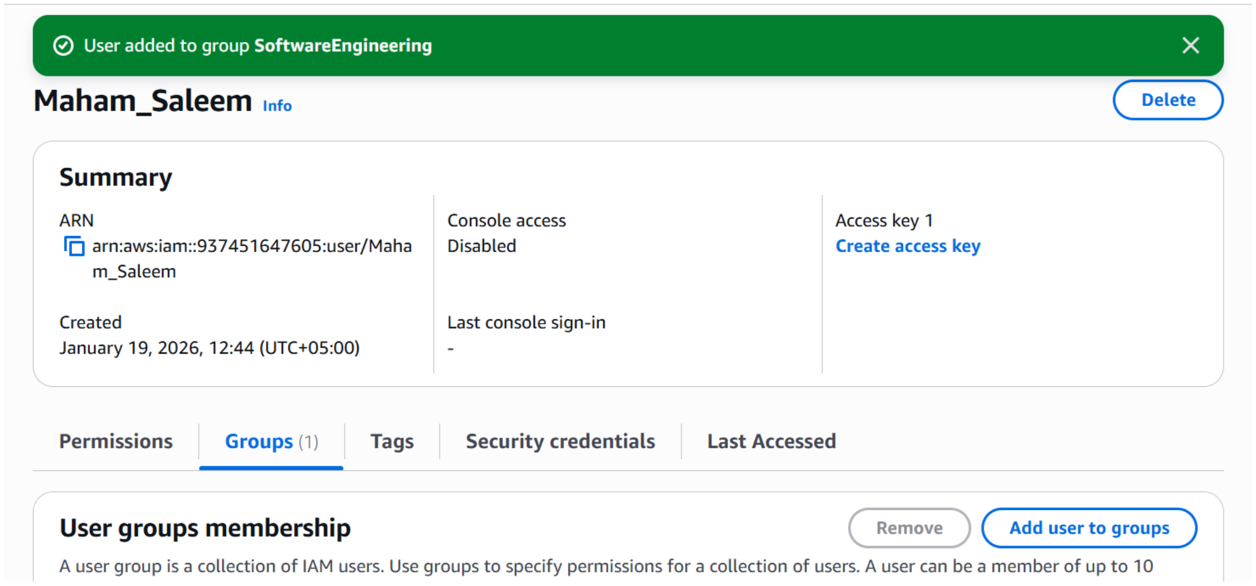
Access key 1
Create access key

Permissions policies (1)

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

Filter by Type

q1_add_user_to_group.png



User added to group **SoftwareEngineering**

Maham_Saleem Info

Delete

Summary

ARN
arn:aws:iam::937451647605:user/Maham_Saleem

Created
January 19, 2026, 12:44 (UTC+05:00)

Console access
Disabled

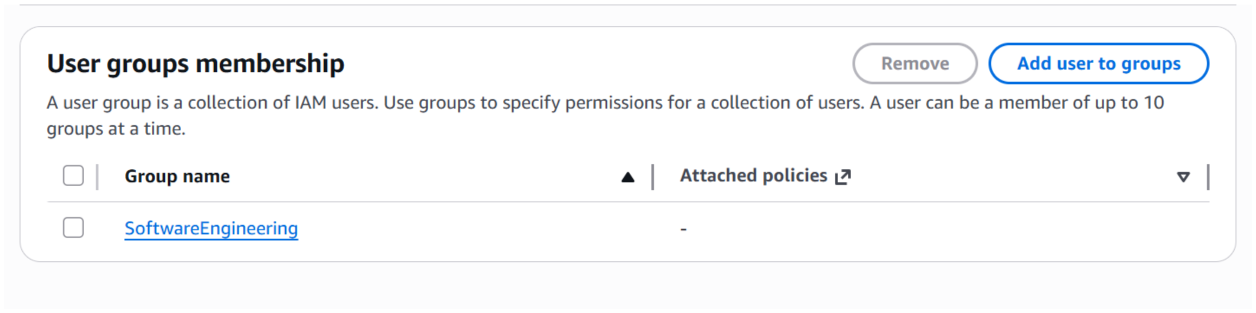
Last console sign-in
-

Access key 1
Create access key

User groups membership

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users. A user can be a member of up to 10

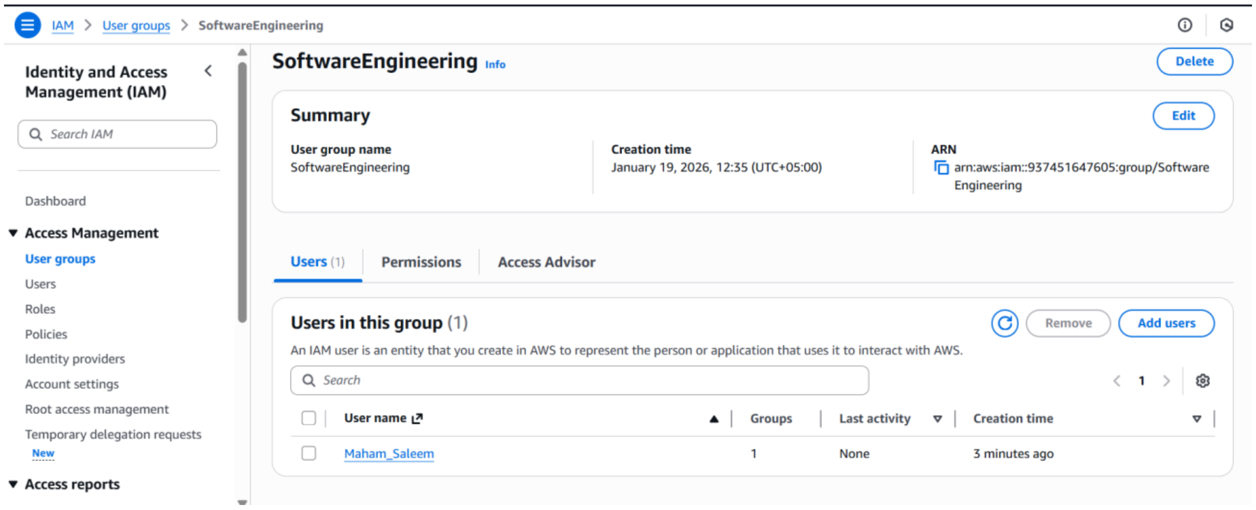
q1_group_membership.png



User groups membership

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users. A user can be a member of up to 10 groups at a time.

Group name	Attached policies
SoftwareEngineering	-


IAM > User groups > SoftwareEngineering

Identity and Access Management (IAM)

Search IAM

Dashboard

Access Management
- User groups
- Users
- Roles
- Policies
- Identity providers
- Account settings
- Root access management
- Temporary delegation requests New

Access reports

SoftwareEngineering Info

Delete

Summary

Edit

User group name	Creation time	ARN
SoftwareEngineering	January 19, 2026, 12:35 (UTC+05:00)	arn:aws:iam::937451647605:group/SoftwareEngineering

Users (1) | Permissions | Access Advisor

Users in this group (1)

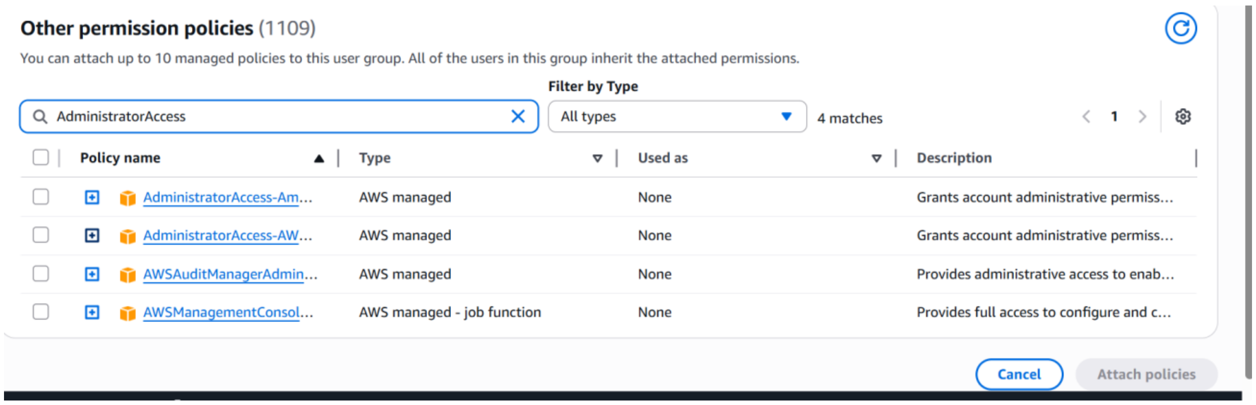
Remove | Add users

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

Search

User name	Groups	Last activity	Creation time
Maham_Saleem	1	None	3 minutes ago

q1_find_admin_policy.png


Other permission policies (1109)

You can attach up to 10 managed policies to this user group. All of the users in this group inherit the attached permissions.

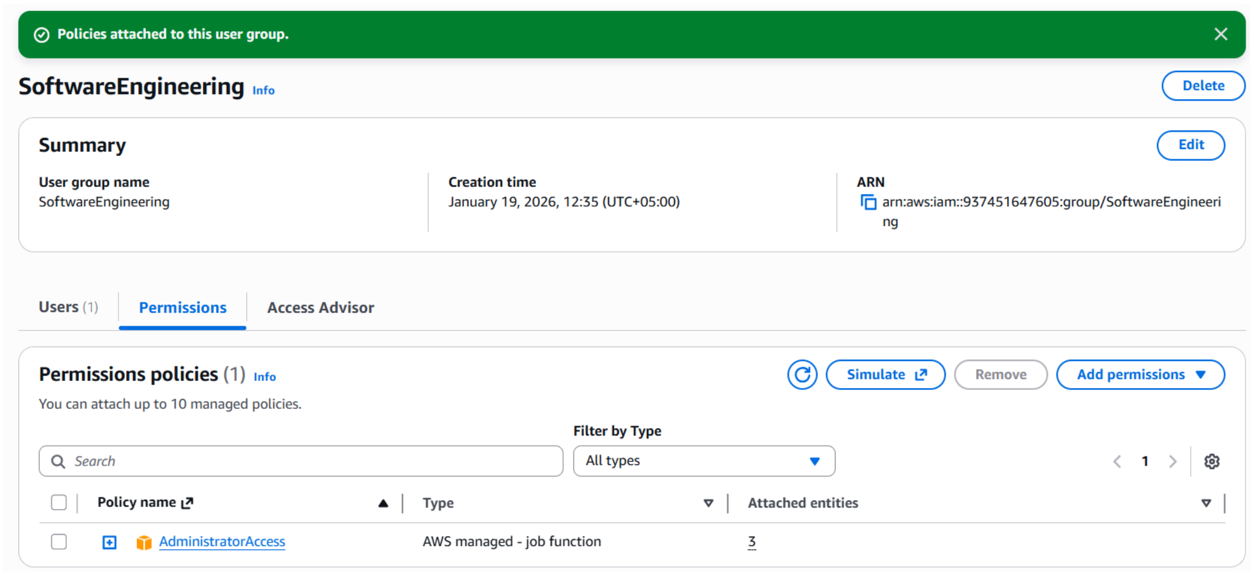
Filter by Type

AdministratorAccess | All types | 4 matches

Policy name	Type	Used as	Description
AdministratorAccess-Am…	AWS managed	None	Grants account administrative permiss…
AdministratorAccess-AW…	AWS managed	None	Grants account administrative permiss…
AWSAuditManagerAdmin…	AWS managed	None	Provides administrative access to enab…
AWSManagementConsol…	AWS managed - job function	None	Provides full access to configure and c…

Cancel | Attach policies

q1_attach_admin_policy.png



Policies attached to this user group.

SoftwareEngineering Info

Summary

User group name	Creation time	ARN
SoftwareEngineering	January 19, 2026, 12:35 (UTC+05:00)	arn:aws:iam::937451647605:group/SoftwareEngineering

Users (1) | Permissions | Access Advisor

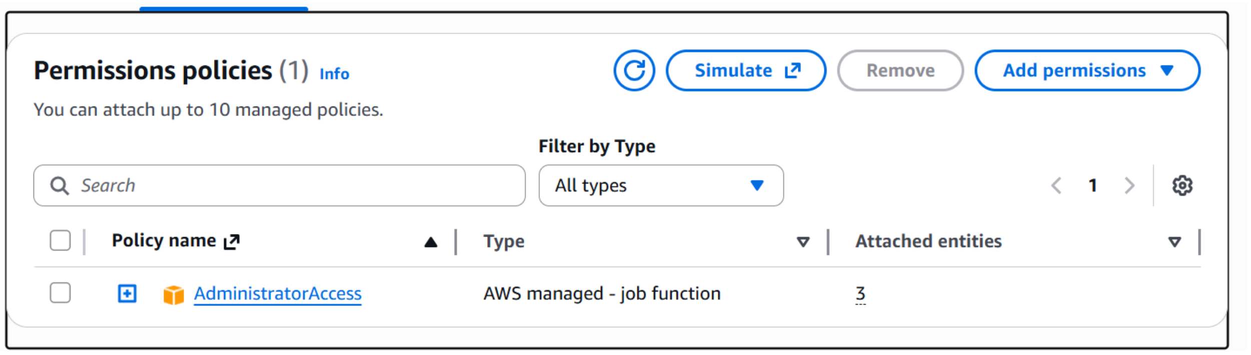
Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	3

q1_list_group_policies.png



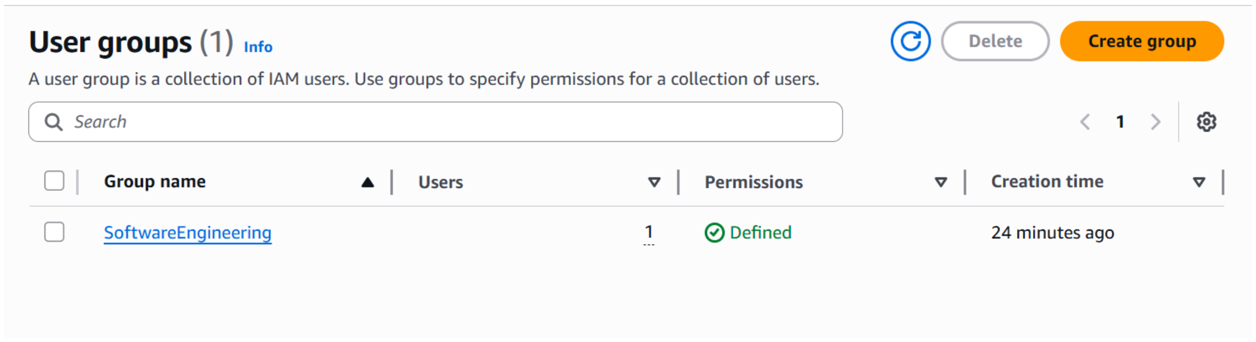
Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	3

q1_console_group.png

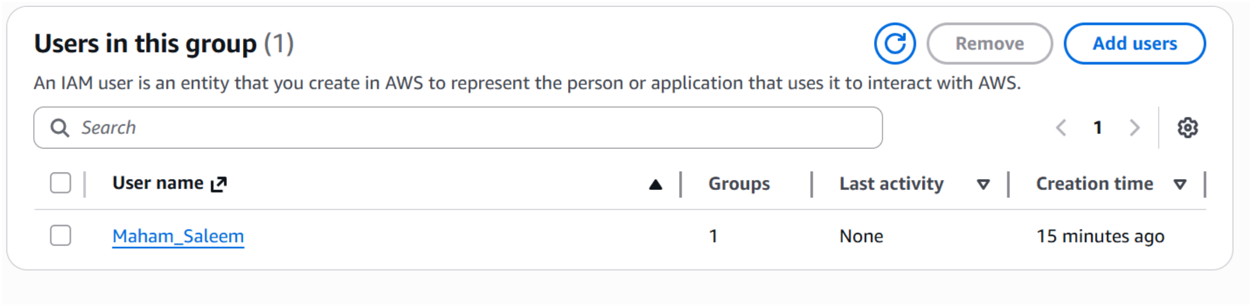


User groups (1) Info

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	24 minutes ago

q1_console_user_in_group.png

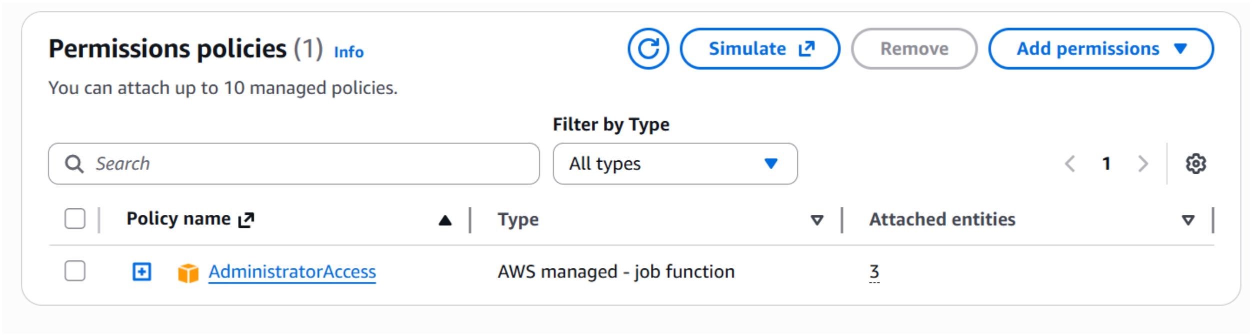


Users in this group (1)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

User name	Groups	Last activity	Creation time
Maham_Saleem	1	None	15 minutes ago

q1_console_group_policy.png



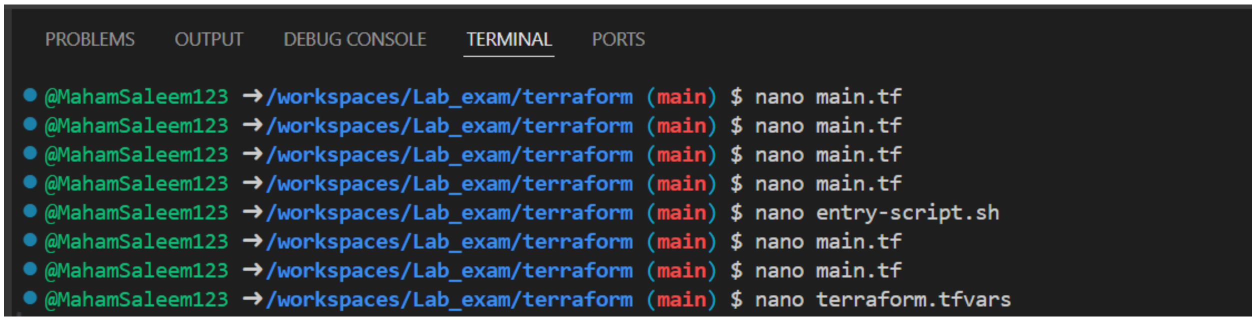
Permissions policies (1)

You can attach up to 10 managed policies.

Filter by Type: All types

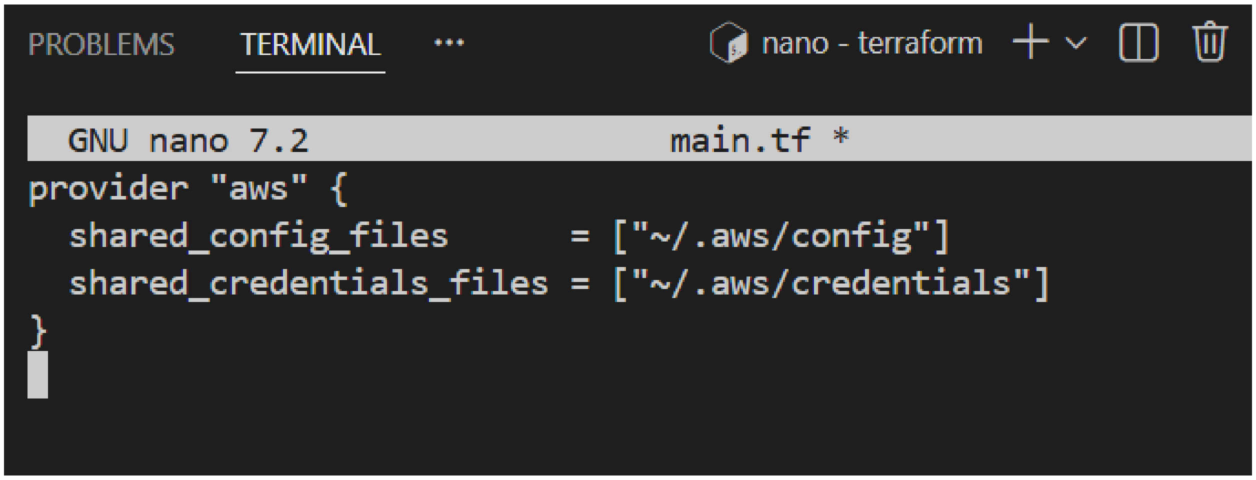
Policy name	Type	Attached entities
AdministratorAccess	AWS managed - job function	3

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



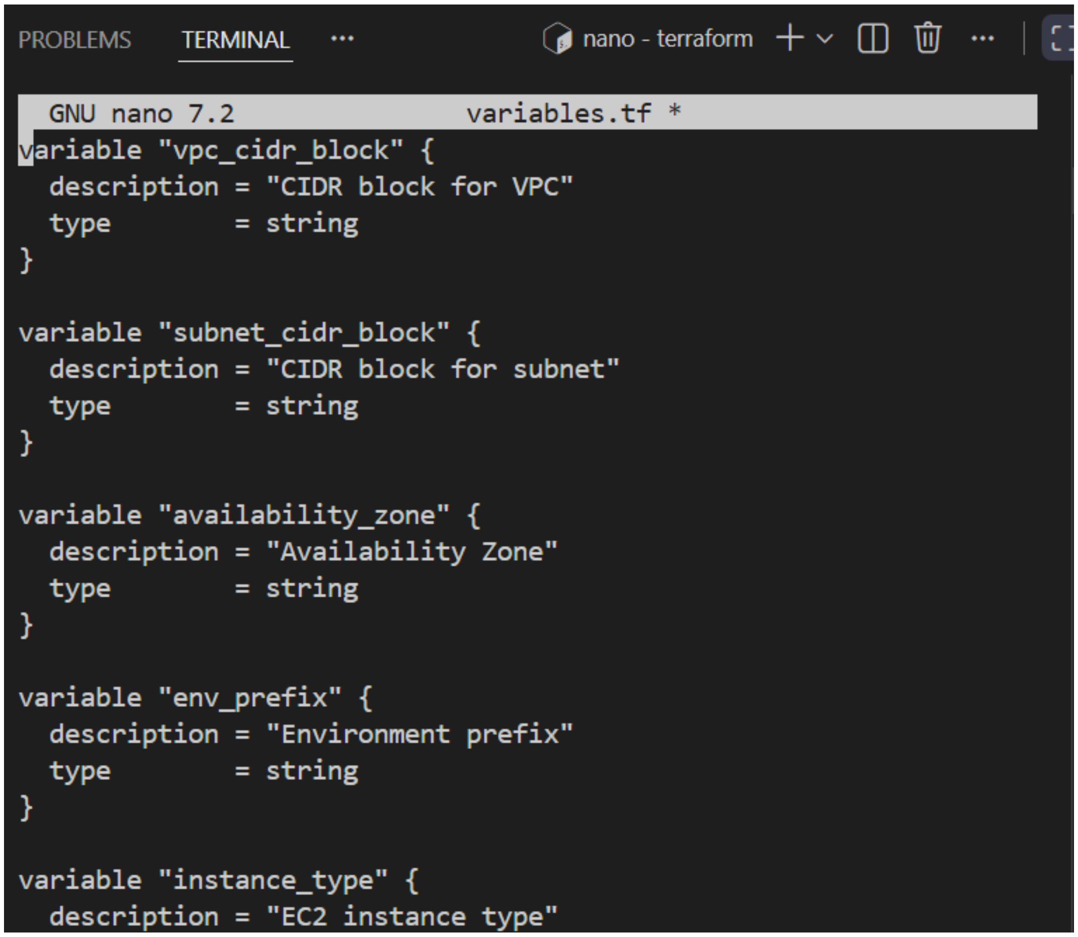
```
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano entry-script.sh
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano main.tf
@MahamSaleem123 →/workspaces/Lab_exam/terraform (main) $ nano terraform.tfvars
```

q2_provider.png



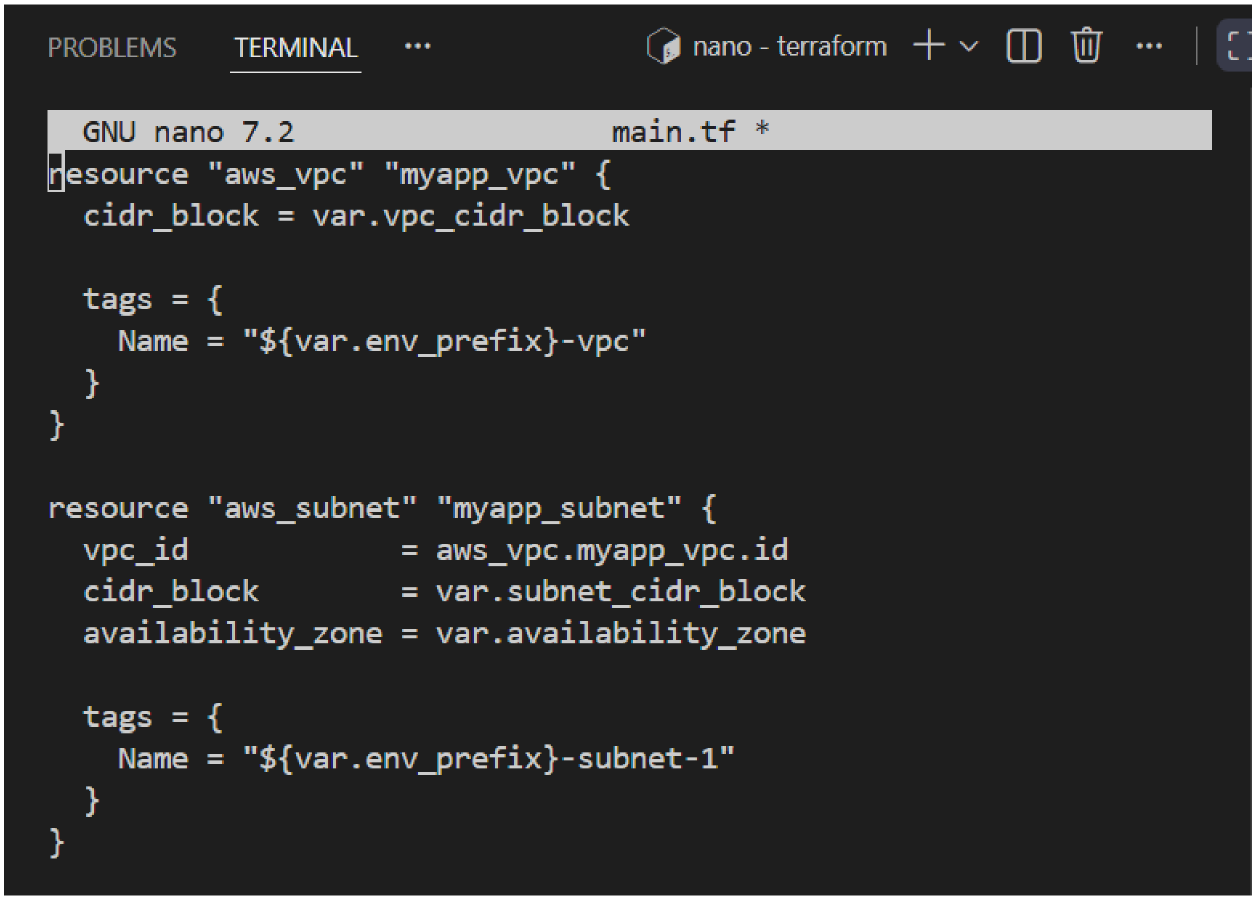
```
  GNU nano 7.2                        main.tf *
provider "aws" {
  shared_config_files      = ["~/.aws/config"]
  shared_credentials_files = ["~/.aws/credentials"]
}
```

q2_provider.png



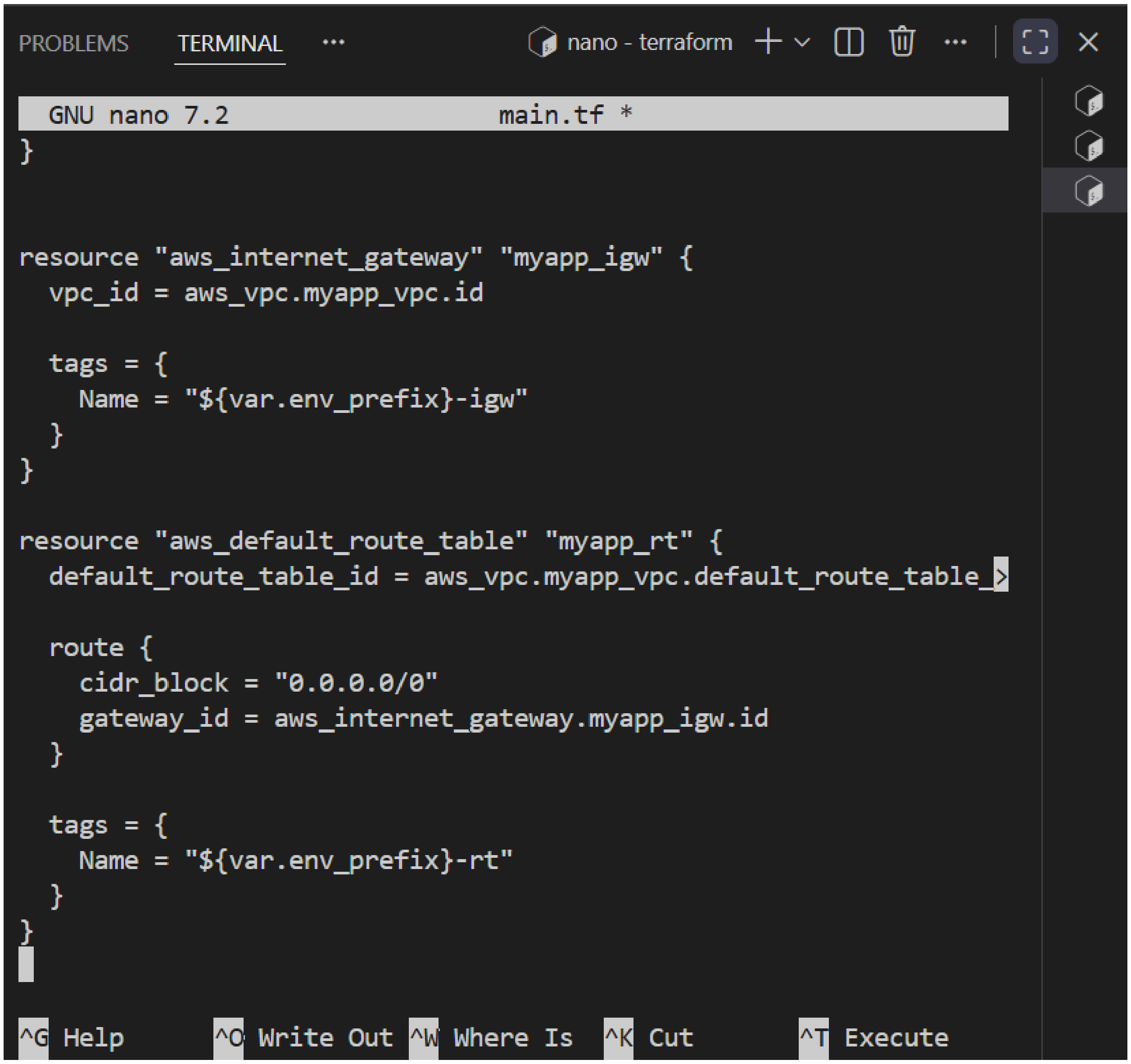
```
  GNU nano 7.2                  variables.tf *
variable "vpc_cidr_block" {
  description = "CIDR block for VPC"
  type        = string
}

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"
```

q2_vpc_subnet.png



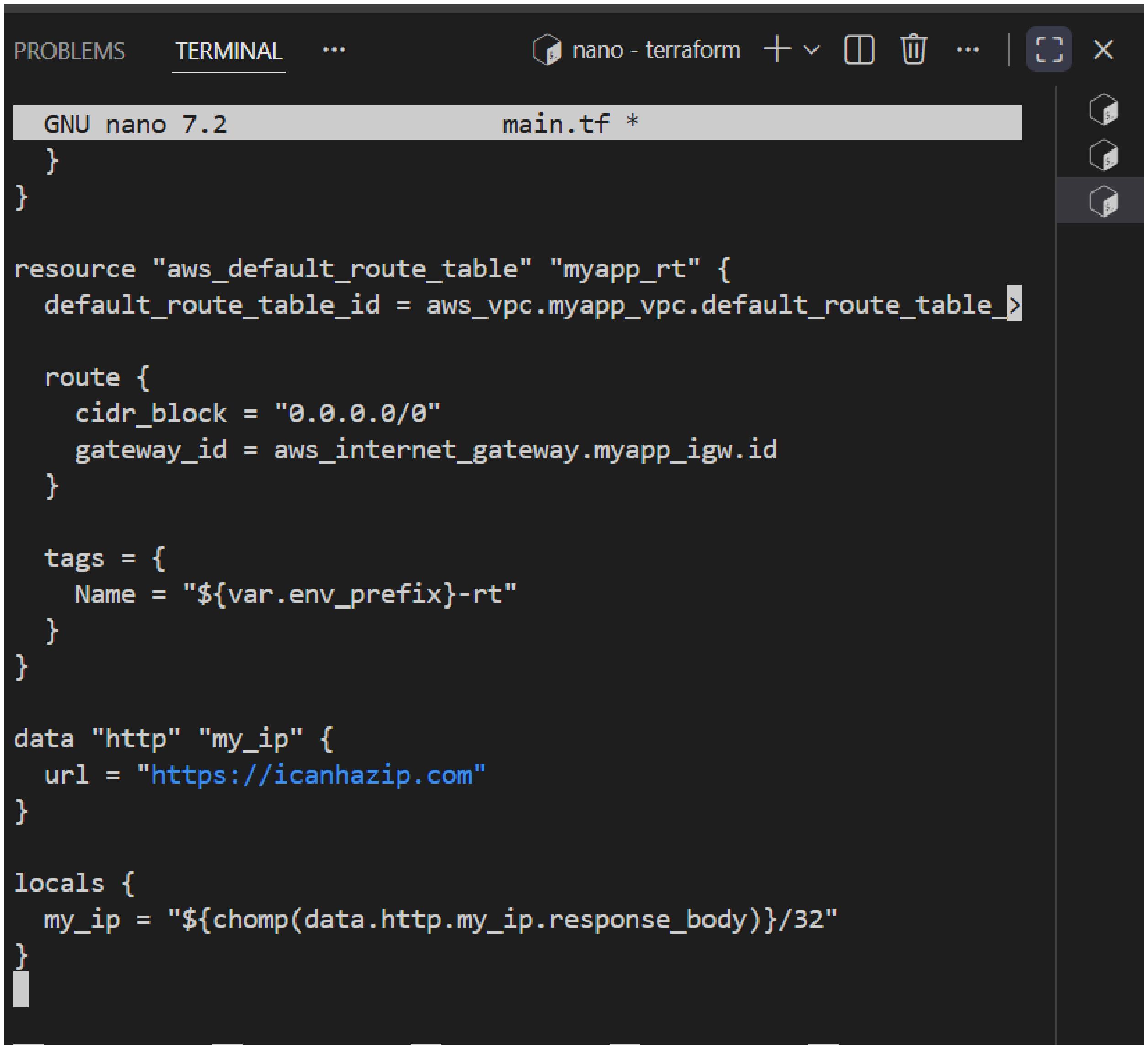
```
  GNU nano 7.2                              main.tf *
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block

  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"
  }
}
```

q2_igw_route_table.png



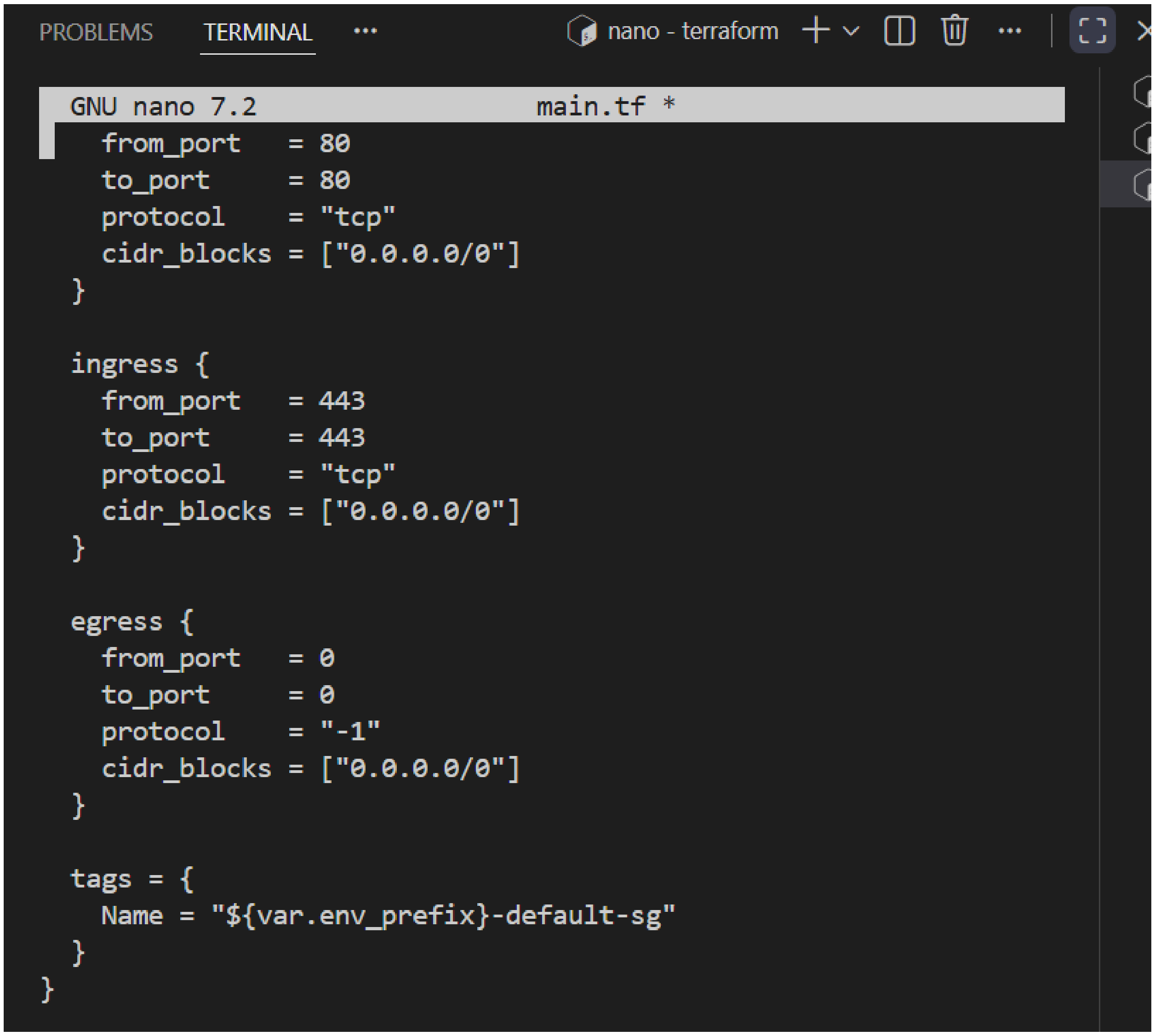
```
  GNU nano 7.2                      main.tf *
}


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"
  }
}


^G Help         ^O Write Out ^W Where Is ^K Cut          ^T Execute
```

q2_http_and_locals.png



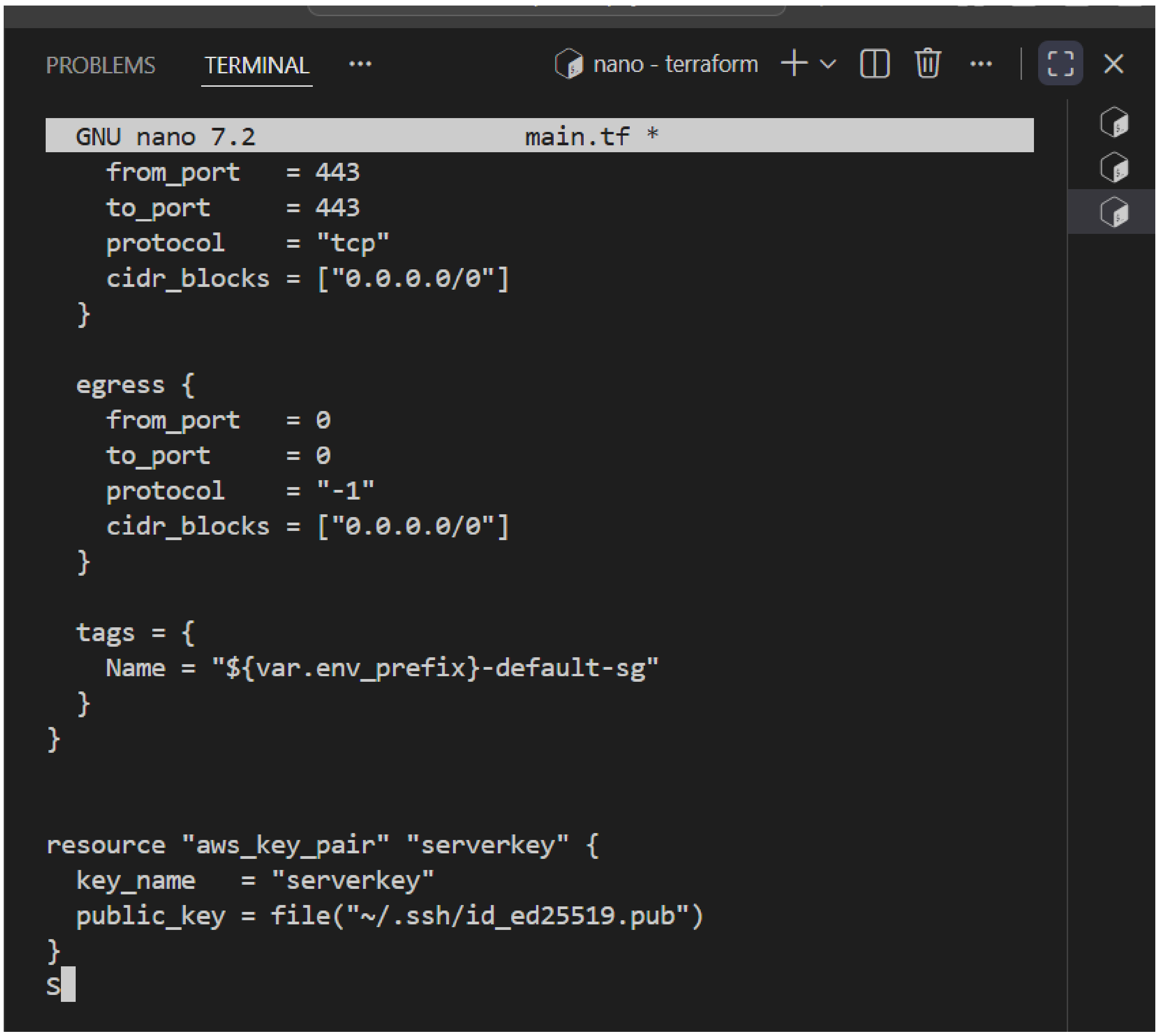
```
  GNU nano 7.2                         main.tf *
  }
}

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"
  }
}

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

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

q2_default_sg.png



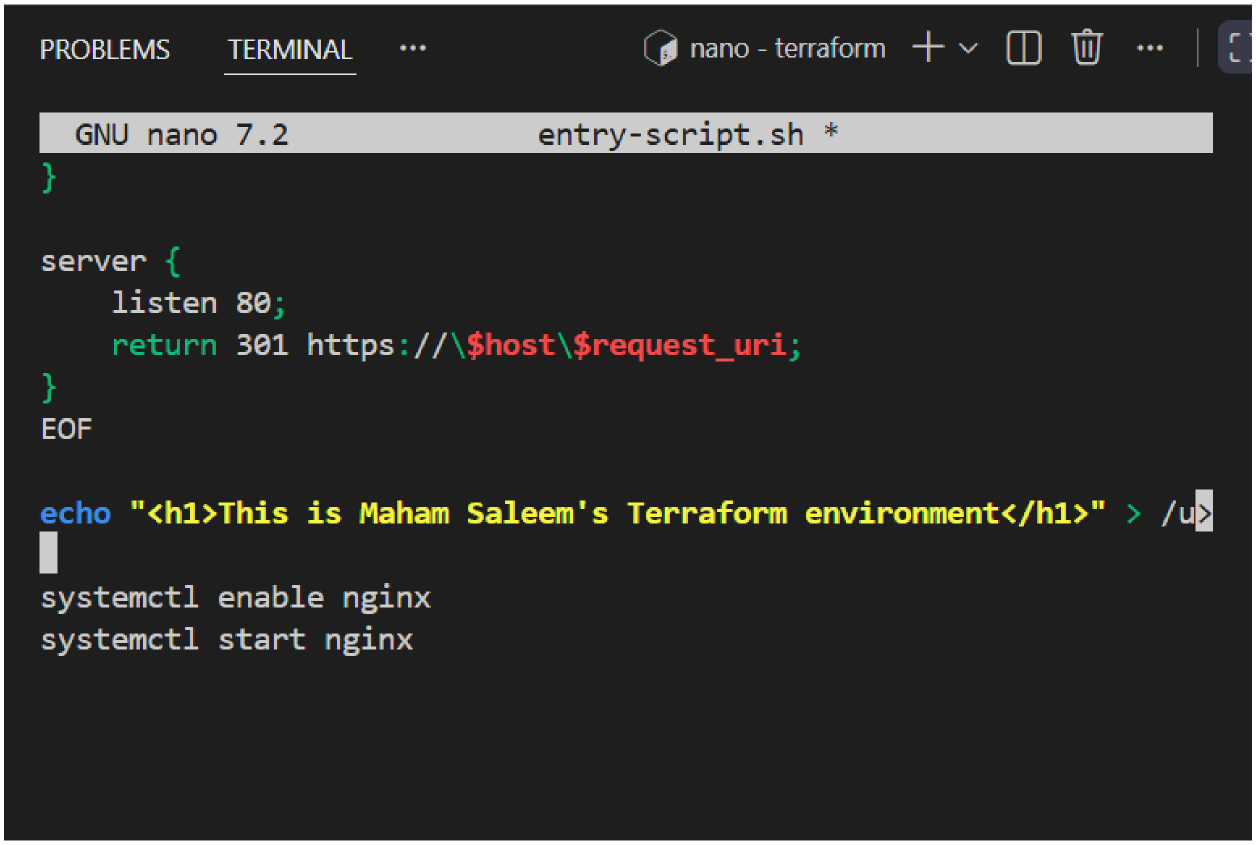
```
  GNU nano 7.2                   main.tf *
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  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"
  }
}
```

q2_keypair.png



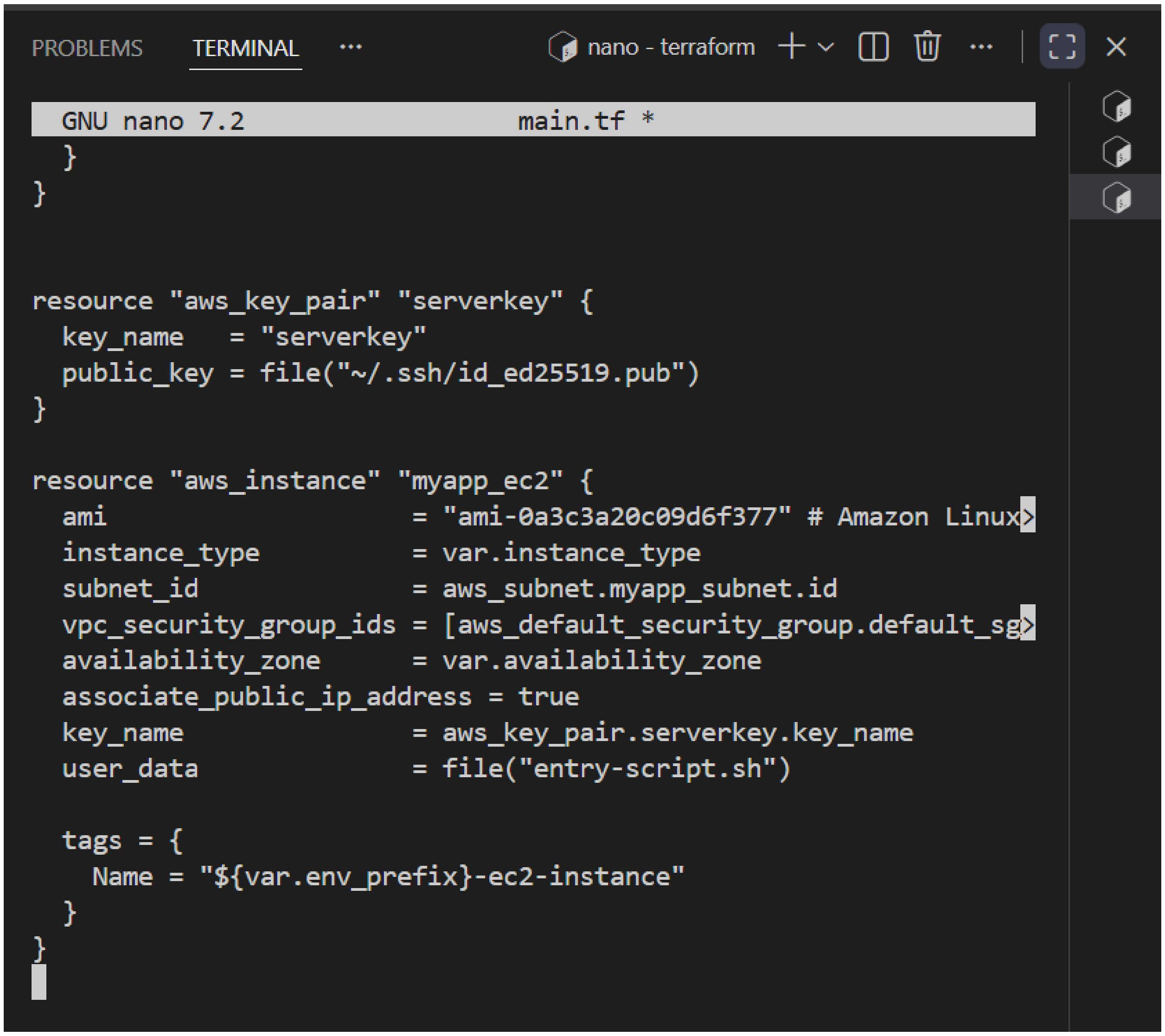
```
  GNU nano 7.2                              main.tf *
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

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

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


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

q2_entry_script.png



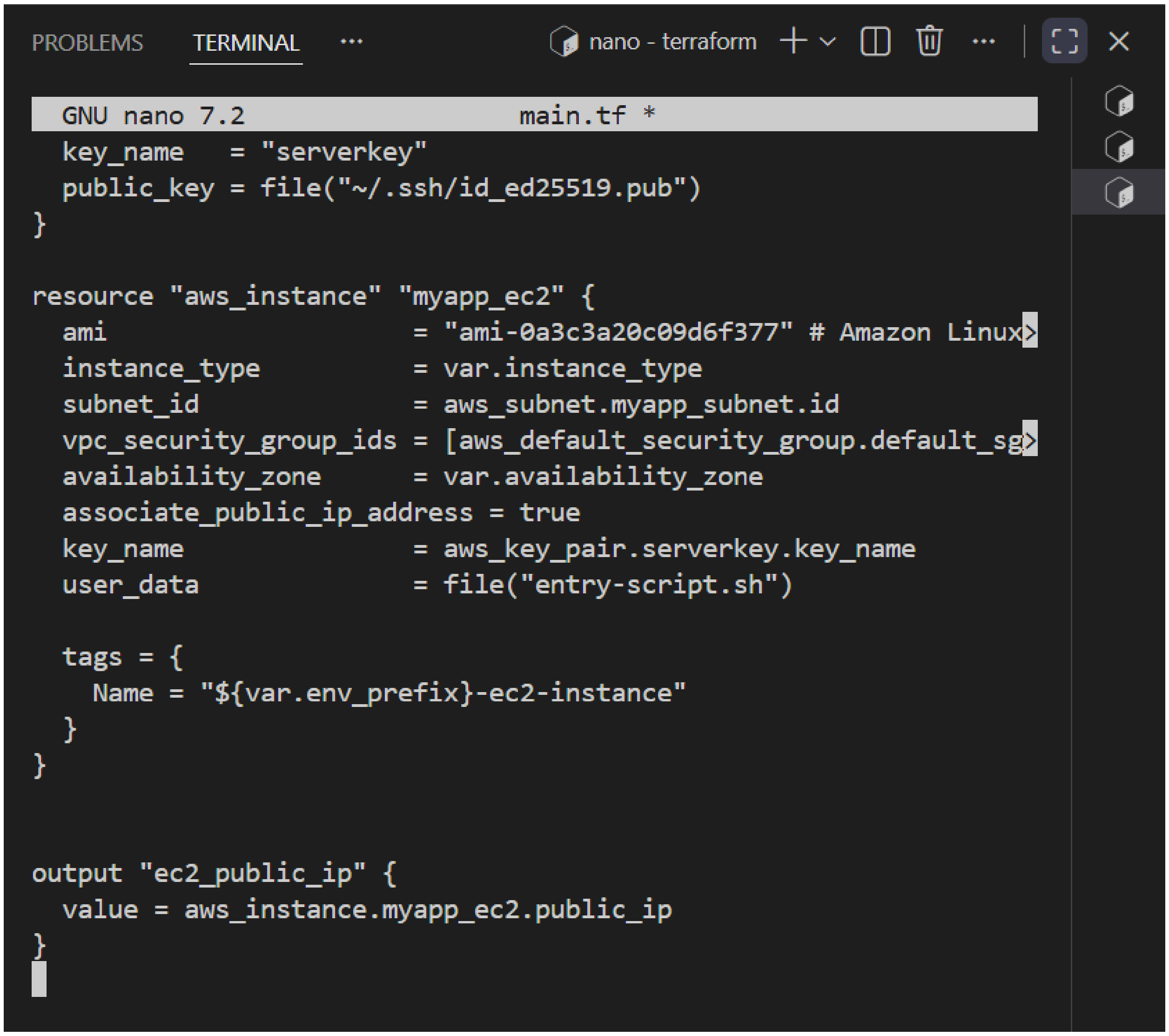
```
  GNU nano 7.2                entry-script.sh *
}

server {
    listen 80;
    return 301 https://\$host\$request_uri;
}
EOF

echo "<h1>This is Maham Saleem's Terraform environment</h1>" > /u>


systemctl enable nginx
systemctl start nginx
```

q2_ec2_resource.png



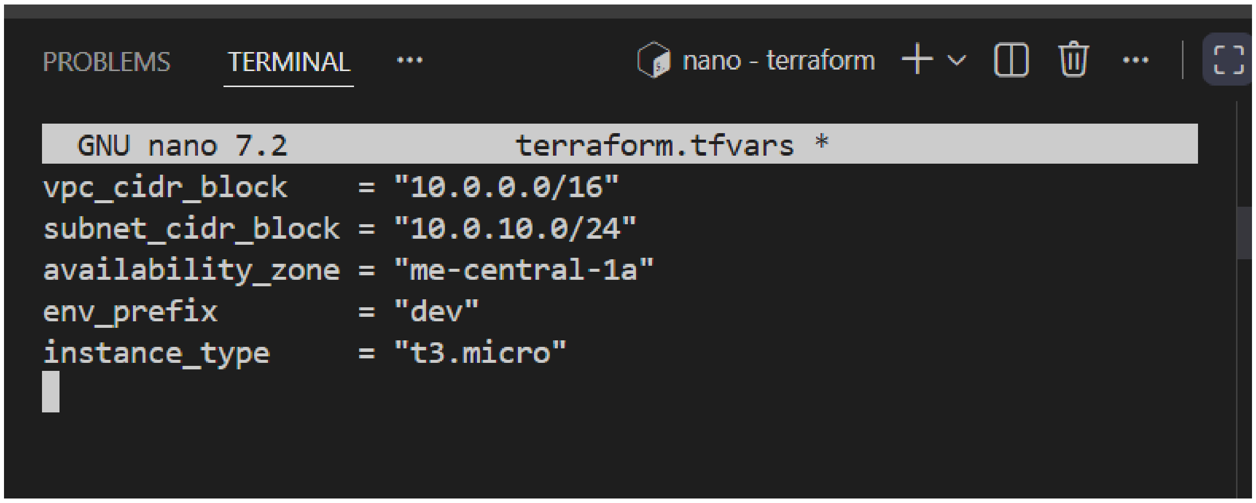
```
 GNU nano 7.2                     main.tf *
  }
}


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

resource "aws_instance" "myapp_ec2" {
  ami                         = "ami-0a3c3a20c09d6f377" # Amazon Linux>
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet.id
  vpc_security_group_ids = [aws_default_security_group.default_sg>
  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"
  }
}
```

q2_output_block.png



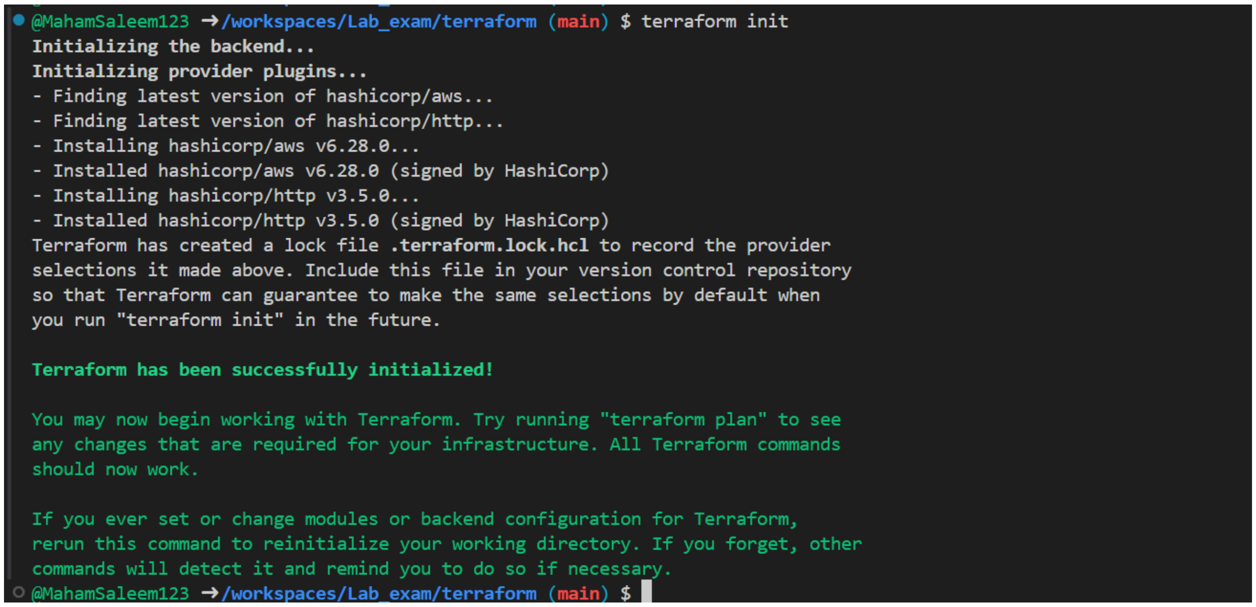
```
  GNU nano 7.2                    main.tf *
  key_name    = "serverkey"
  public_key = file("~/.ssh/id_ed25519.pub")
}

resource "aws_instance" "myapp_ec2" {
  ami                         = "ami-0a3c3a20c09d6f377" # Amazon Linux>
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.myapp_subnet.id
  vpc_security_group_ids = [aws_default_security_group.default_sg>
  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"
  }
}


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

q2_tfvars_or_vars.png



```
 GNU nano 7.2                     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"
```

q2_terraform_init.png



```
@MahamSaleem123 ➜ /workspaces/Lab_exam/terraform (main) $ 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,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
@MahamSaleem123 ➜ /workspaces/Lab_exam/terraform (main) $
```