

CLOUD COMPUTING LAB EXAM

Submitted to: Engr. Waqas Saleem

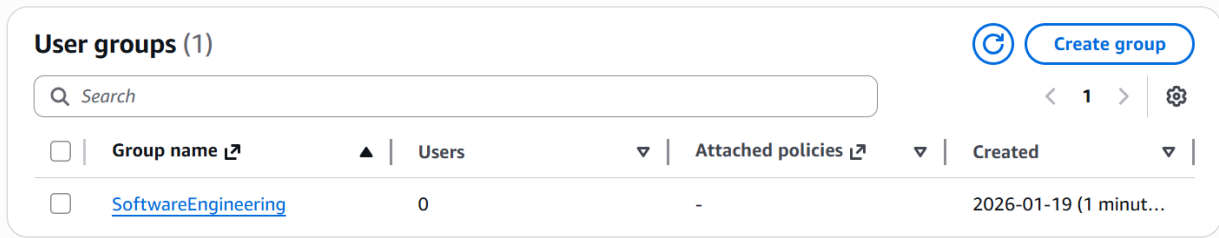
Engr. Muhammad Shahzad

Submitted by: Maham Saleem

Registration no.: 2023-BSE-035

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

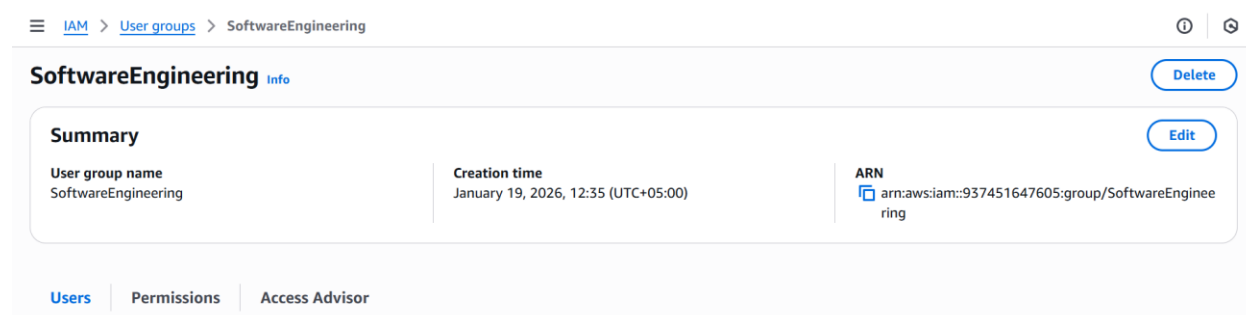
q1_create_group.png



The screenshot shows the 'User groups' page in the AWS IAM console. At the top, there is a 'Create group' button. Below it is a search bar. A table lists the groups:

<input type="checkbox"/>	Group name ↗	▲	Users	▼	Attached policies ↗	▼	Created	▼
<input type="checkbox"/>	SoftwareEngineering		0		-		2026-01-19 (1 minut...	

q1_group_details.png

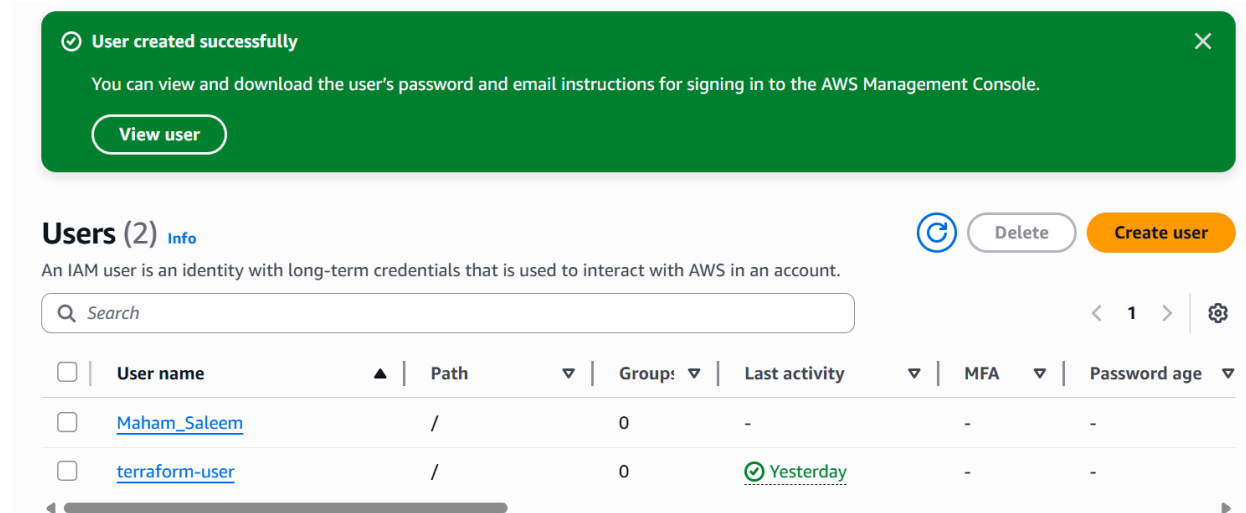


The screenshot shows the details for the 'SoftwareEngineering' group. The breadcrumb trail is 'IAM > User groups > SoftwareEngineering'. There are 'Delete' and 'Edit' buttons. The 'Summary' section contains the following information:

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

Below the summary, there are tabs for 'Users', 'Permissions', and 'Access Advisor'. The 'Users' tab is currently selected.

q1_create_user.png



The screenshot shows a green success message: 'User created successfully'. Below the message is a 'View user' button. Below the message, the 'Users' page is displayed. The breadcrumb trail is 'IAM > Users'. There are 'Delete' and 'Create user' buttons. A search bar is present. A table lists the users:


<input type="checkbox"/>	User name	▲	Path	▼	Groups	▼	Last activity	▼	MFA	▼	Password age	▼
<input type="checkbox"/>	Maham_Saleem		/		0		-		-		-	
<input type="checkbox"/>	terraform-user		/		0		✓ Yesterday		-		-	

q1_user_details.png

Maham_SaleemInfoDelete

Summary

ARN

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

Console access

Disabled

Access key 1

Create access key

Created

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

Last console sign-in

-

Permissions


Groups

Tags

Security credentials

Last Accessed

Permissions policies (1)




Remove

Add permissions

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 SoftwareEngineeringX

Maham_SaleemInfoDelete

Summary

ARN

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

Console access

Disabled

Access key 1

Create access key

Created

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

Last console sign-in

-

Permissions

Groups (1)

Tags

Security credentials

Last Accessed

User groups membership

Remove

Add user to groups

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.

q1_group_membership.png

User groups membership

Remove

Add user to groups

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

▲

☐

SoftwareEngineering

-

Attached policies

↗

▼

IAM

User groups

SoftwareEngineering

1

Q

Identity and Access Management (IAM)

Q 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

SoftwareEngineering

Creation time

January 19, 2026, 12:35 (UTC+05:00)

ARN

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.

Q Search

< 1 >

⚙

☐

User name ↗

▲

Groups

▼

Last activity

▼

Creation time

▼

☐

Maham_Saleem

1

None

3 minutes ago

q1_find_admin_policy.png

Other permission policies (1109)

Q AdministratorAccess

Filter by Type

All types

4 matches

< 1 >

⚙

☐

Policy name

▲

Type

▼

Used as

▼

Description

☐

AdministratorAccess-Am...

AWS managed

None

Grants account administrative permis...

☐

AdministratorAccess-AW...

AWS managed

None

Grants account administrative permis...

☐

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

Delete

Summary Edit

User group name SoftwareEngineering	Creation time January 19, 2026, 12:35 (UTC+05:00)	ARN 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 ▼

< 1 > ⚙️

<input type="checkbox"/>	Policy name	Type	Attached entities
<input type="checkbox"/>	AdministratorAccess	AWS managed - job function	3

q1_list_group_policies.png

Permissions policies (1) [Info](#)

You can attach up to 10 managed policies.

[Simulate](#)

[Remove](#)

[Add permissions](#)

Filter by Type

All types

< 1 >

<input type="checkbox"/>	Policy name ↗	▲ Type ▼	Attached entities ▼
<input type="checkbox"/>	AdministratorAccess	AWS managed - job function	3

q1_console_group.png

User groups (1) [Info](#)

↺

Delete

Create group

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

🔍

Search

<

1

>

⚙️

<input type="checkbox"/>	Group name	▲	Users	▼	Permissions	▼	Creation time	▼
<input type="checkbox"/>	SoftwareEngineering		1		✔️ Defined		24 minutes ago	

q1_console_user_in_group.png

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.

< 1 > ⚙️

<input type="checkbox"/>	User name ↗	Groups	Last activity	Creation time
<input type="checkbox"/>	Maham_Saleem	1	None	15 minutes ago

q1_console_group_policy.png

Permissions policies (1) Info Simulate Remove Add permissions

You can attach up to 10 managed policies.

Filter by Type All types

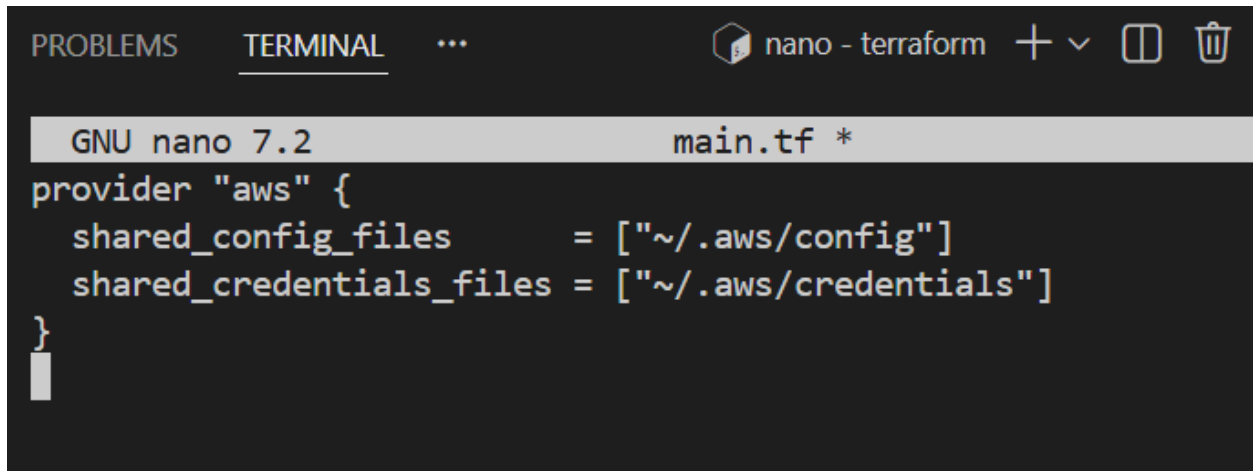
< 1 > ⚙️

<input type="checkbox"/>	Policy name ↗	Type	Attached entities
<input type="checkbox"/>	AdministratorAccess	AWS managed - job function	3

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● @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
```

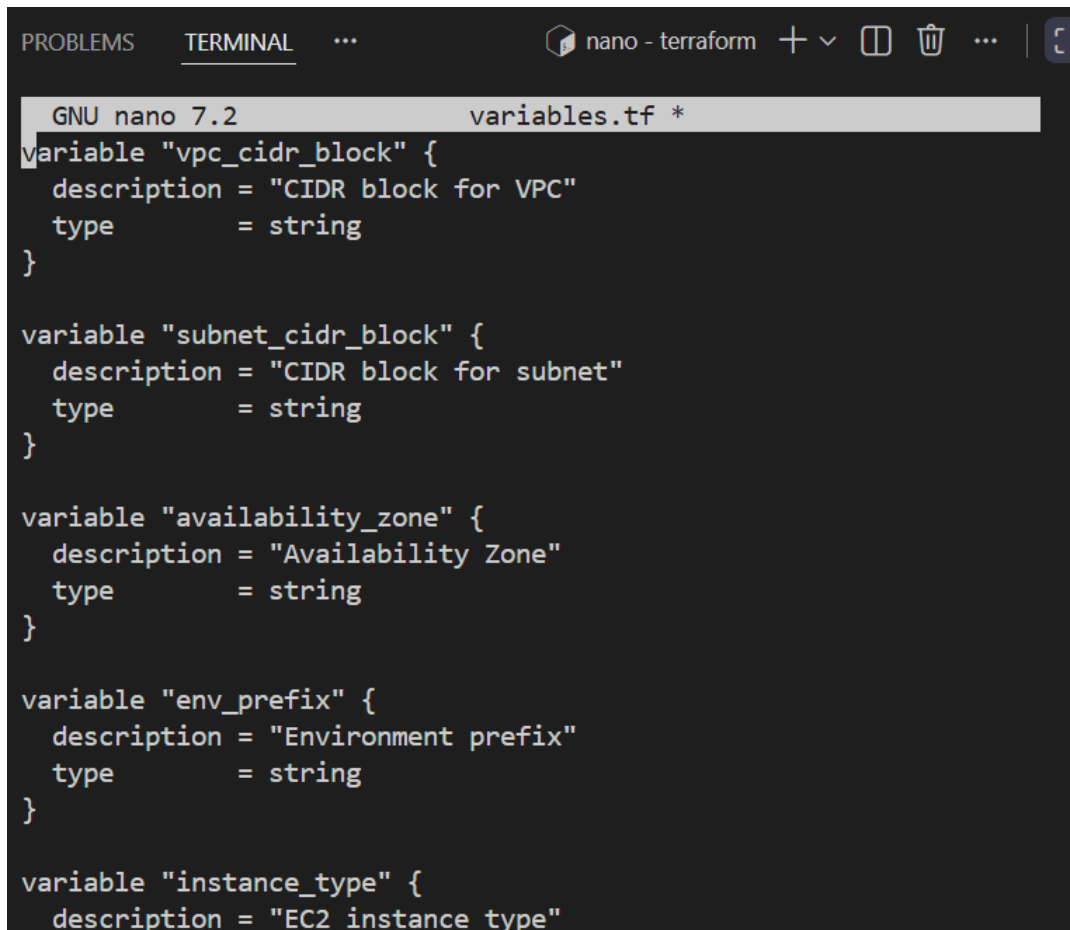
q2_provider.png



```
PROBLEMS  TERMINAL  ...  nano - terraform  +  v  [ ]  [X]

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

q2_provider.png



```
PROBLEMS  TERMINAL  ...  nano - terraform  +  v  [ ]  [X]

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

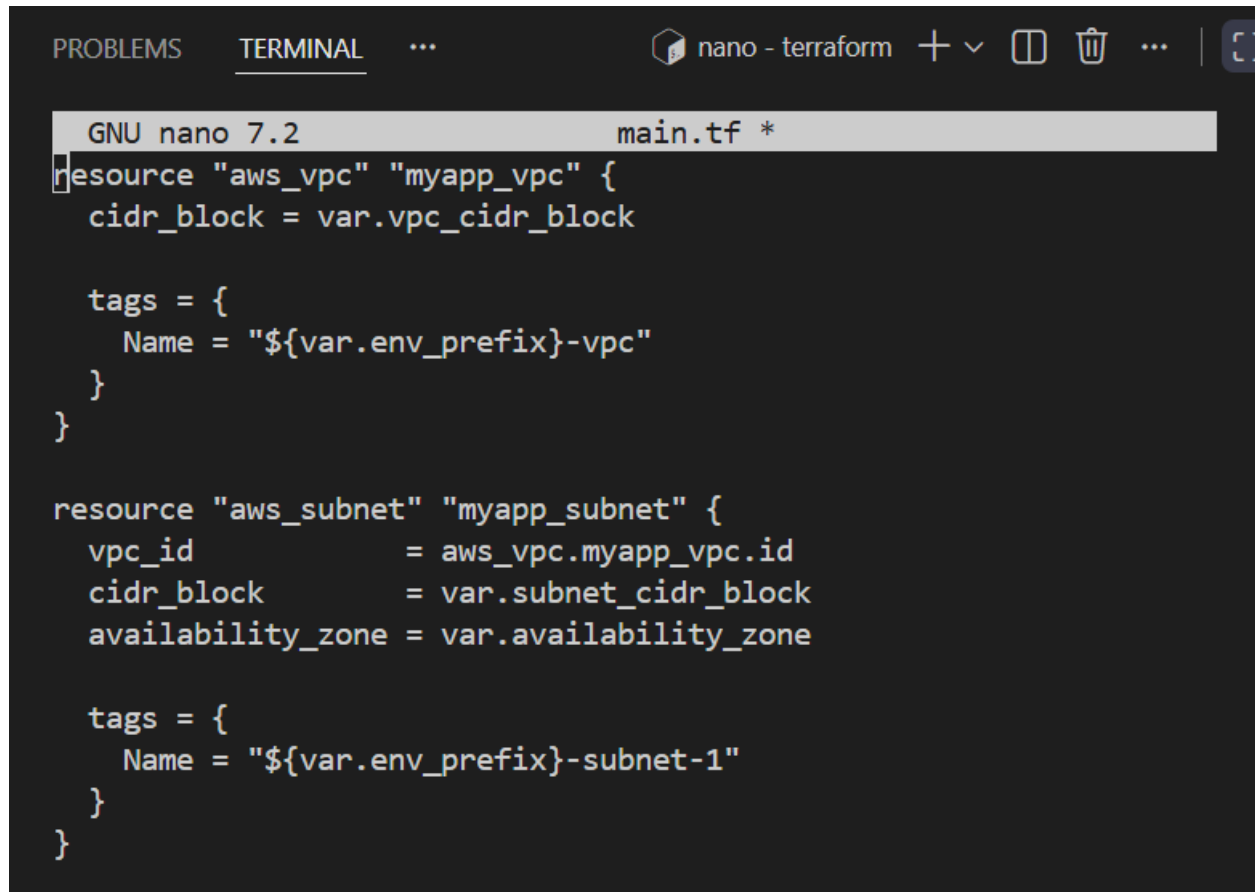
variable "subnet_cidr_block" {
  description = "CIDR block for subnet"
  type        = string
}

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



The image shows a terminal window with a dark background. At the top, there is a header bar with the text "PROBLEMS", "TERMINAL", and "...". To the right of this bar is a tab labeled "nano - terraform" with a plus sign, a dropdown arrow, and icons for a window, trash, and a search icon. Below the header bar, the terminal shows the prompt "GNU nano 7.2" and the filename "main.tf *". The main content of the terminal is a Terraform configuration file. It starts with a resource block for "aws_vpc" named "myapp_vpc". This block has a "cidr_block" attribute set to "var.vpc_cidr_block" and a "tags" block. The "tags" block has a "Name" attribute set to "\${var.env_prefix}-vpc". Below this is another resource block for "aws_subnet" named "myapp_subnet". This block has three attributes: "vpc_id" set to "aws_vpc.myapp_vpc.id", "cidr_block" set to "var.subnet_cidr_block", and "availability_zone" set to "var.availability_zone". It also has a "tags" block with a "Name" attribute set to "\${var.env_prefix}-subnet-1". The configuration ends with a closing brace for the "aws_subnet" resource.

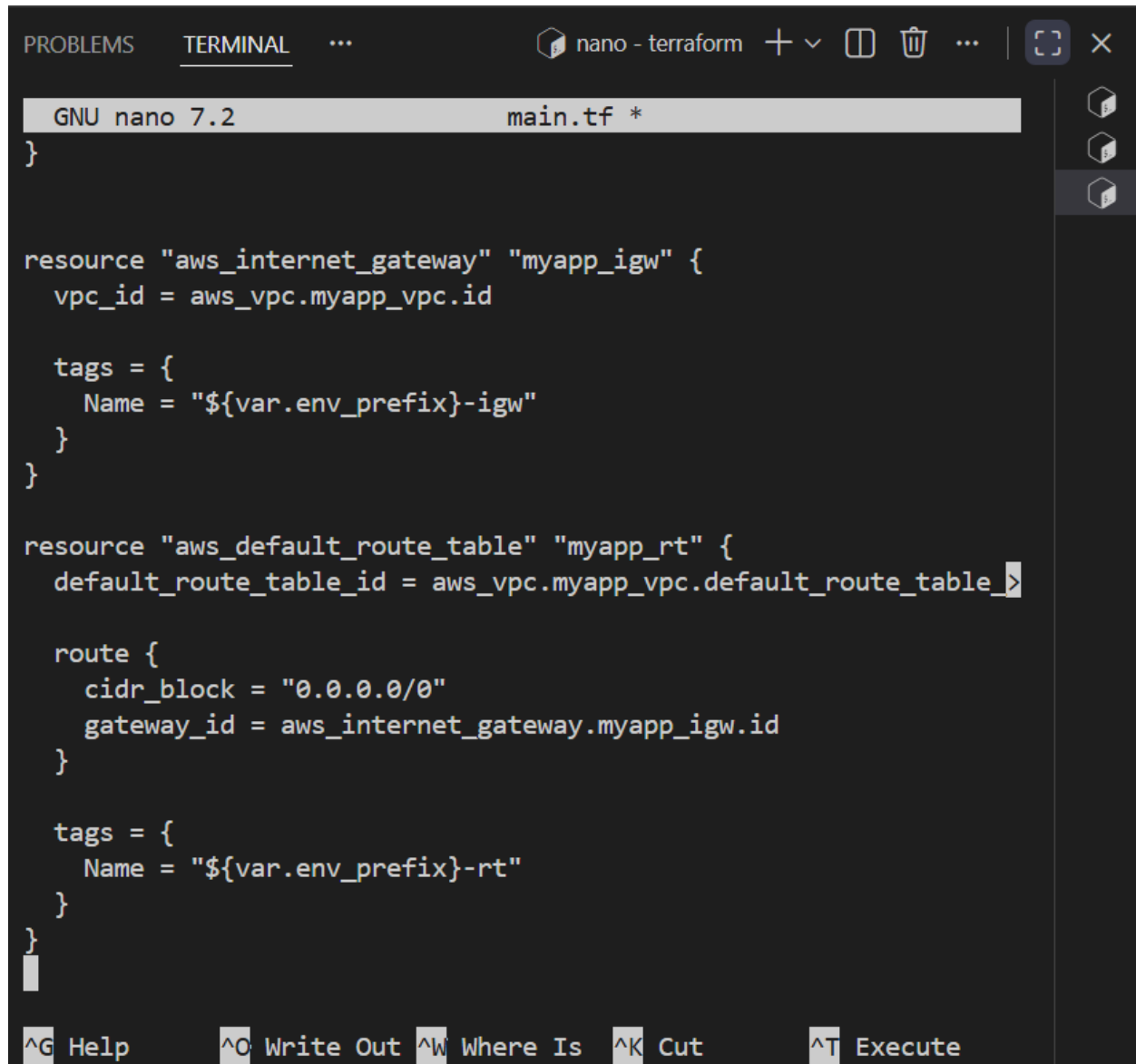
```
GNU nano 7.2 main.tf *
resource "aws_vpc" "myapp_vpc" {
  cidr_block = var.vpc_cidr_block

  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



The image shows a terminal window with the nano text editor open. The editor is editing a file named 'main.tf'. The code defines two Terraform resources: an 'aws_internet_gateway' named 'myapp_igw' and an 'aws_default_route_table' named 'myapp_rt'. The gateway is associated with 'aws_vpc.myapp_vpc.id' and has a tag 'Name' with the value '\${var.env_prefix}-igw'. The route table is associated with 'aws_vpc.myapp_vpc.default_route_table_id' and has a route for '0.0.0.0/0' pointing to 'aws_internet_gateway.myapp_igw.id'. It also has a tag 'Name' with the value '\${var.env_prefix}-rt'. The bottom of the window shows nano editor shortcuts: ^G Help, ^O Write Out, ^W Where Is, ^K Cut, and ^T Execute.

```
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"
  }
}

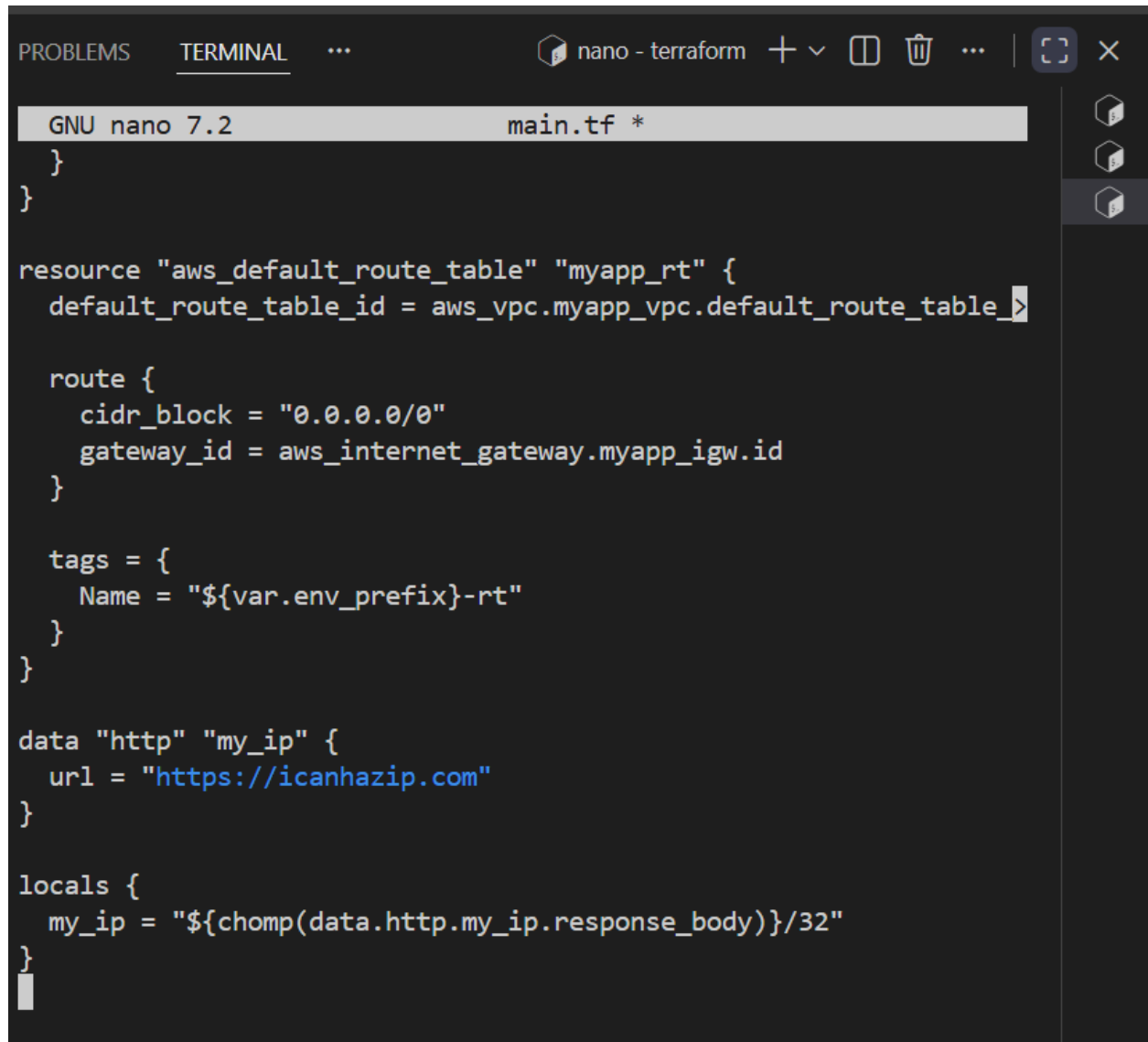
resource "aws_default_route_table" "myapp_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

  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



```
PROBLEMS  TERMINAL  ...  nano - terraform  + v  [ ]  [X]  ...  [ ]  [X]
GNU nano 7.2  main.tf *
}
}

resource "aws_default_route_table" "myapp_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

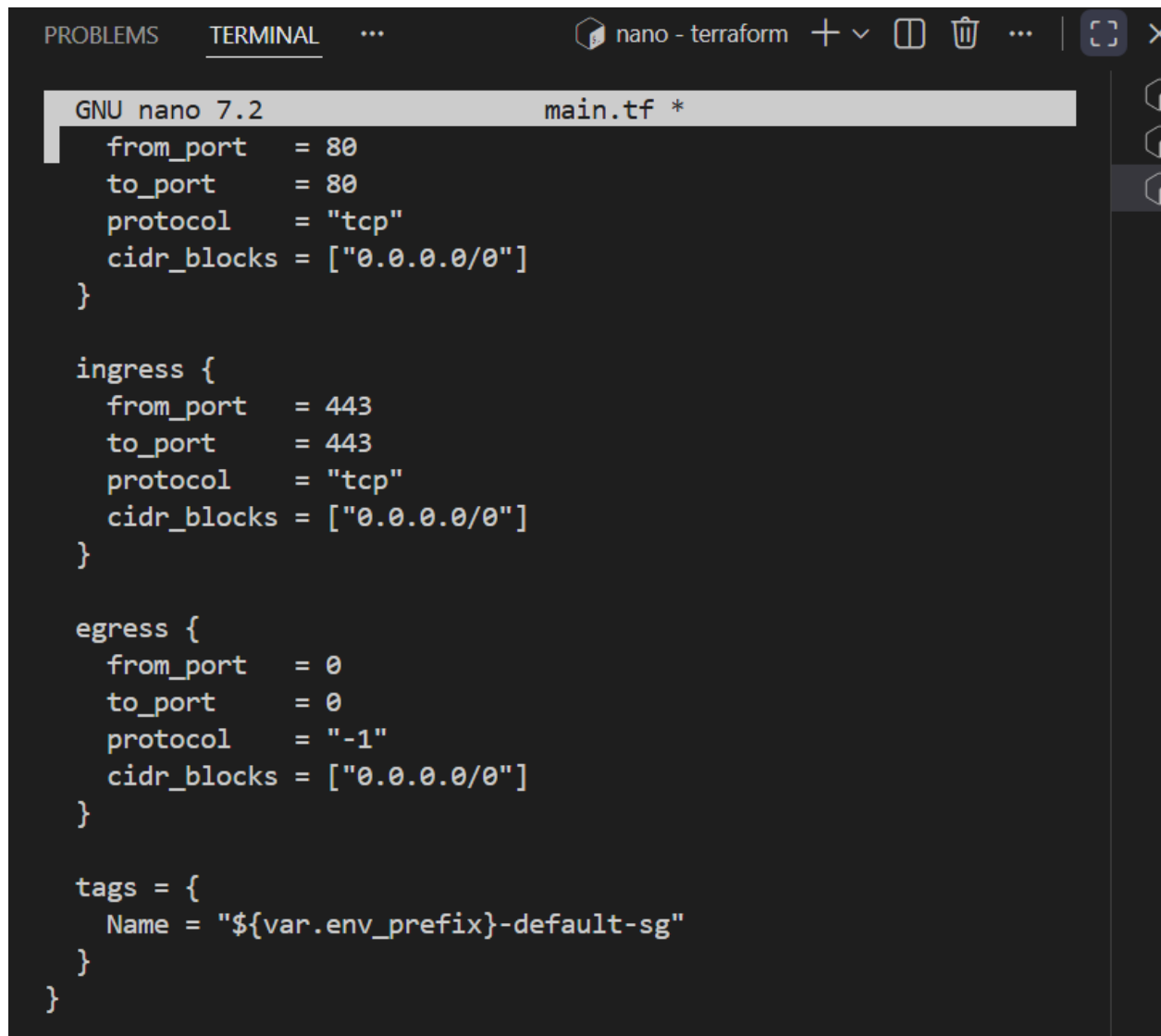
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.myapp_igw.id
  }

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

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

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

q2_default_sg.png



The image shows a terminal window with the nano editor open. The title bar indicates the editor is running on a system named 'terraform'. The editor is editing a file named 'main.tf'. The code displayed is Terraform configuration for a security group. It defines three rules: one for port 80, one for port 443, and one for all traffic (protocol -1). The security group is tagged with the name derived from the environment prefix and the string '-default-sg'.

```
PROBLEMS  TERMINAL  ...  nano - terraform  +  -  [  ]  [X]  ...  [  ]  X

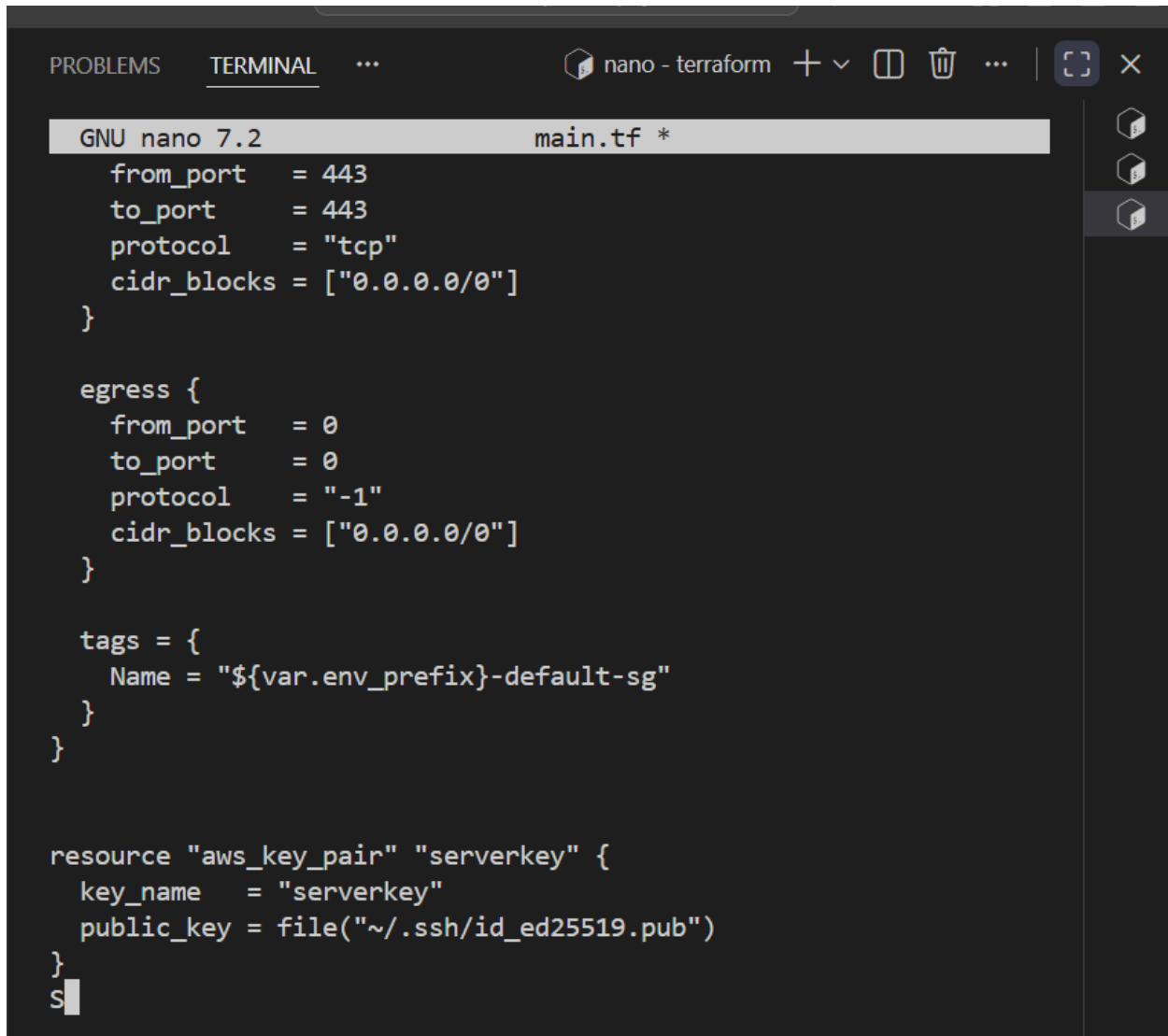
GNU nano 7.2  main.tf *
    from_port    = 80
    to_port      = 80
    protocol     = "tcp"
    cidr_blocks  = ["0.0.0.0/0"]
  }

  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



```
PROBLEMS  TERMINAL  ...  nano - terraform  +  -  [ ]  [X]  ...  [ ]  X

GNU nano 7.2  main.tf *

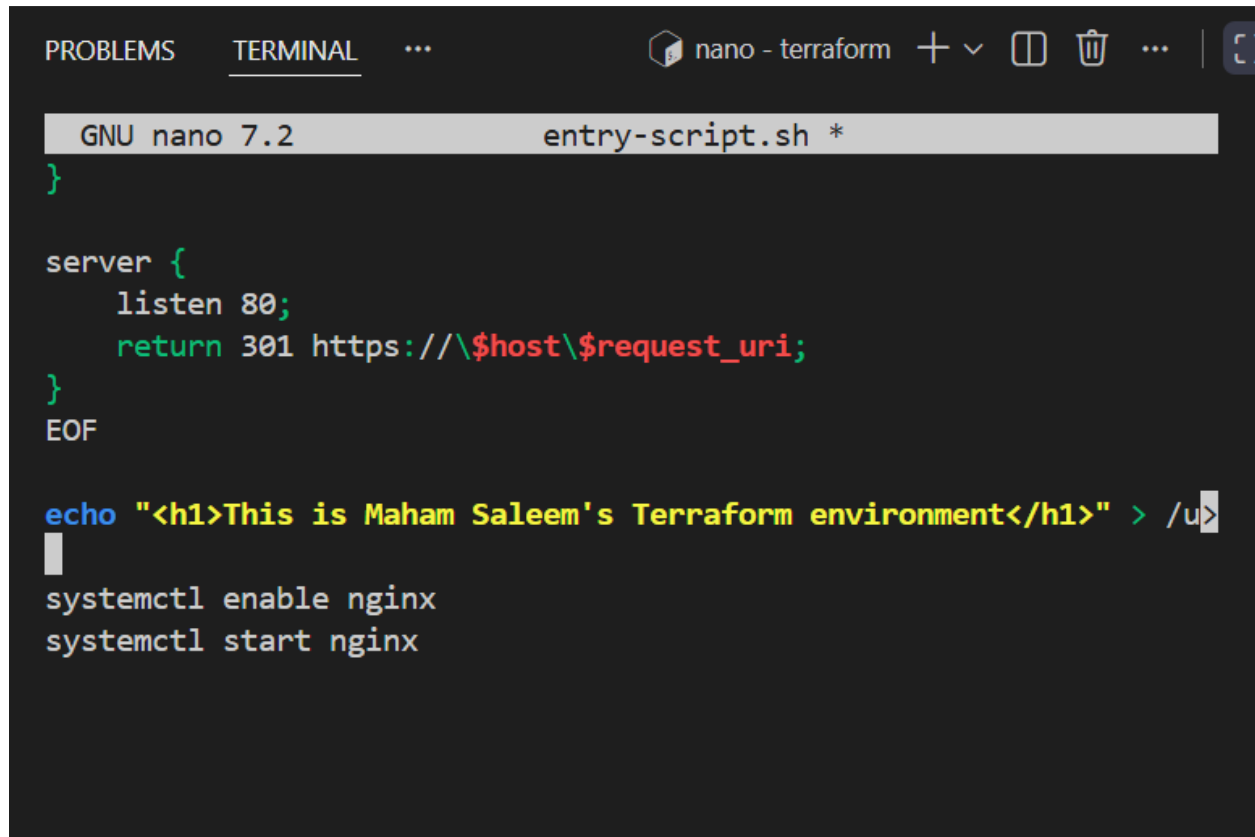
    from_port    = 443
    to_port      = 443
    protocol     = "tcp"
    cidr_blocks  = ["0.0.0.0/0"]
  }

  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

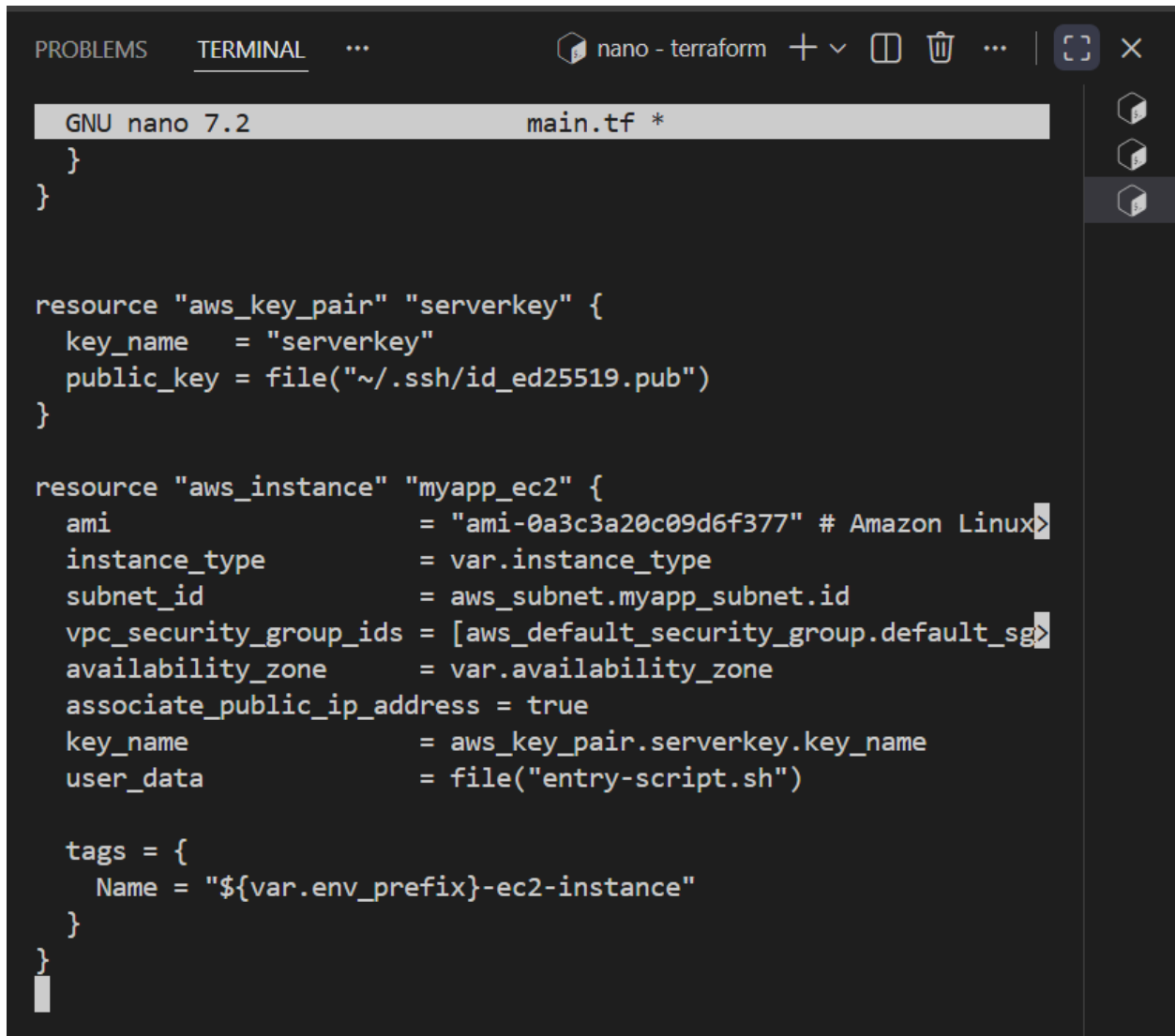


The image shows a terminal window with the nano text editor open. The editor is editing a file named 'entry-script.sh'. The current content of the file is as follows:

```
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
```

The terminal window has a dark background. The nano editor's status bar at the top shows 'GNU nano 7.2' and the filename 'entry-script.sh *'. The code is color-coded: green for curly braces, blue for 'server', red for 'return', and yellow for the echo command. The cursor is positioned at the end of the echo command line.

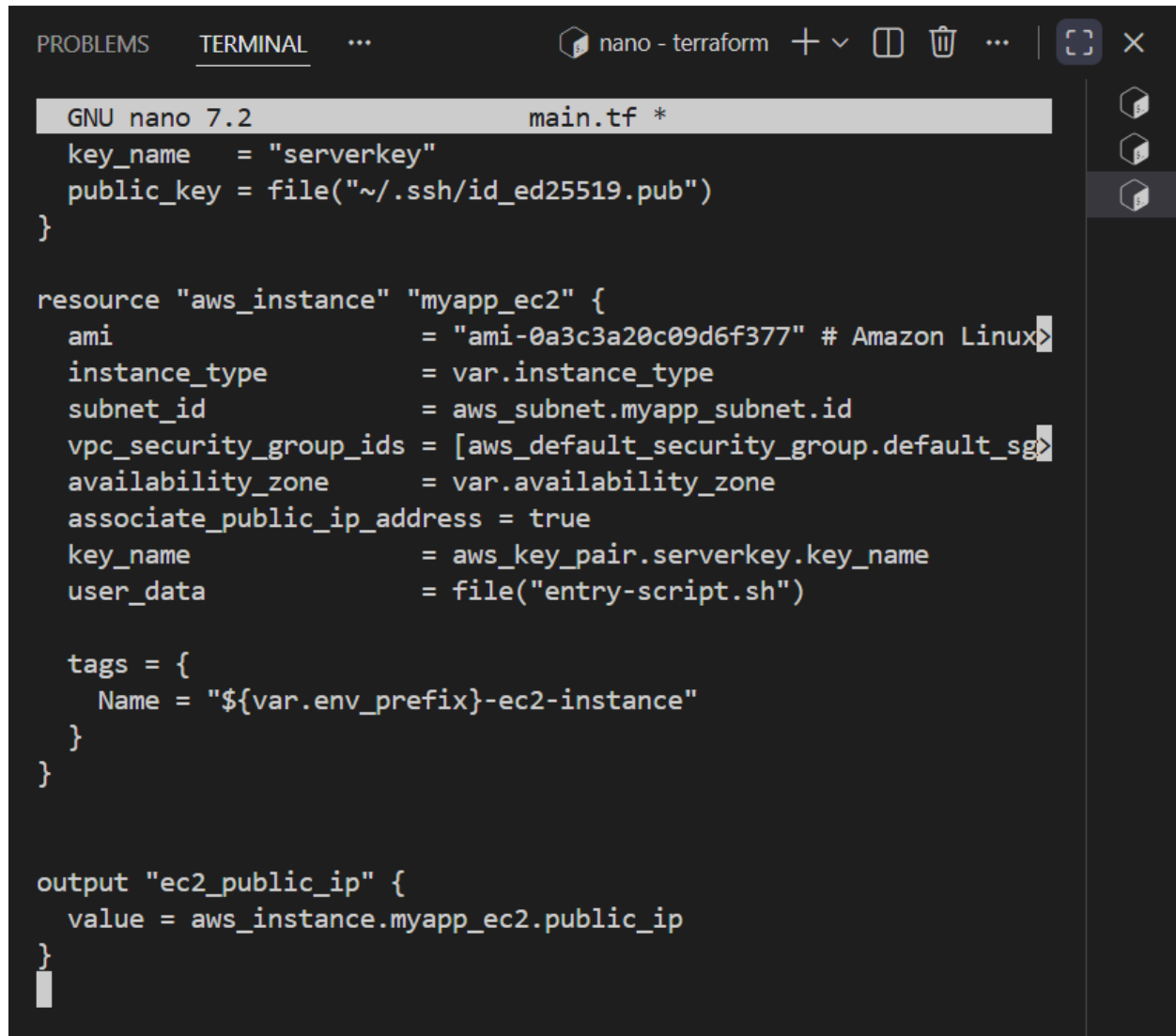
q2_ec2_resource.png



The image shows a terminal window with a dark background. The title bar at the top reads "nano - terraform" and includes standard window controls. Below the title bar, a status line indicates "GNU nano 7.2" and "main.tf *". The terminal displays Terraform configuration code for an AWS EC2 instance. The code defines an AWS key pair named "serverkey" and an EC2 instance named "myapp_ec2". The EC2 instance configuration includes attributes for AMI, instance type, subnet, security groups, availability zone, public IP address, key name, and user data. A tags block is also defined for the instance. The code is as follows:

```
}  
}  
  
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



The image shows a terminal window with a dark background. At the top, there's a header bar with 'PROBLEMS', 'TERMINAL', and a menu icon. To the right of the header, it says 'nano - terraform' followed by window management icons. The main area displays the content of a file named 'main.tf'. The code is written in a light-colored monospace font. It defines a key pair, an EC2 instance resource, and an output for the public IP address. The EC2 instance configuration includes AMI, instance type, subnet, security groups, availability zone, and tags. The output block 'ec2_public_ip' references the public IP of the EC2 instance.

```
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"
  }
}

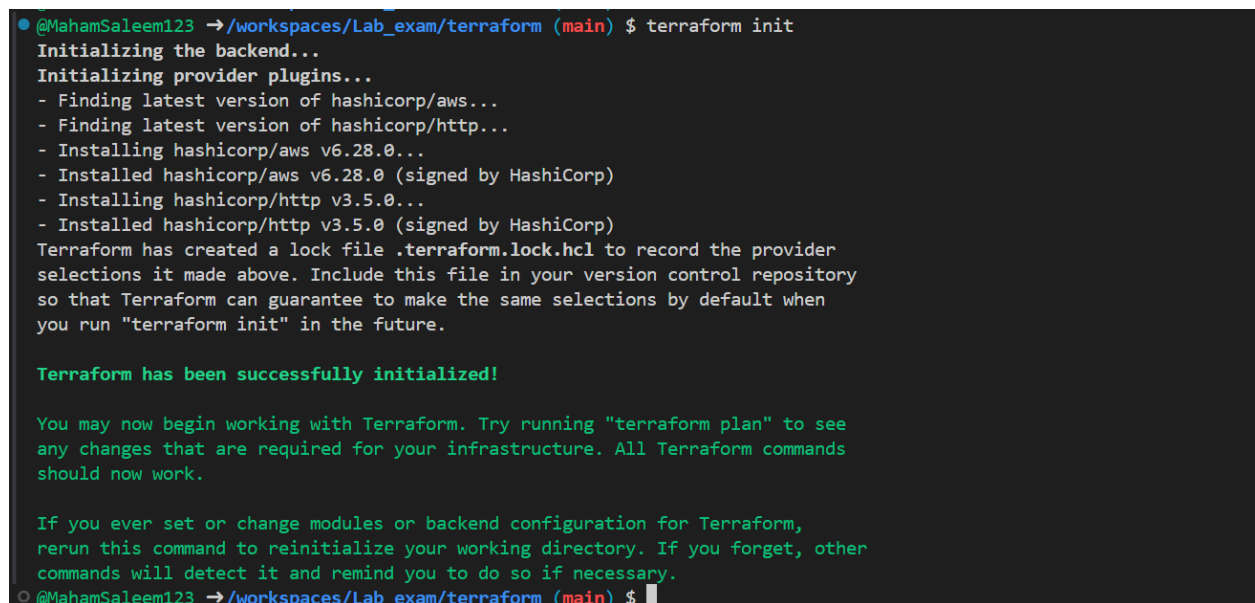
output "ec2_public_ip" {
  value = aws_instance.myapp_ec2.public_ip
}
```

q2_tfvars_or_vars.png



```
PROBLEMS  TERMINAL  ...  nano - terraform  +  -  [ ]  [X]  ...  [ ]  
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) $ |
```