

**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' section of the AWS IAM console. A single group named 'SoftwareEngineering' is listed. The group has 0 users attached and was created on 2026-01-19 (1 minute ago). The interface includes a search bar, sorting options for Group name, Users, Attached policies, and Created, and a 'Create group' button.

q1\_group\_details.png

The screenshot shows the detailed view of the 'SoftwareEngineering' user group. It includes a summary section with the group's name, creation time (January 19, 2026, 12:35 (UTC+05:00)), and ARN (arn:aws:iam::937451647605:group/SoftwareEngineering). There are tabs for Users, Permissions, and Access Advisor, with 'Users' currently selected.

q1\_create\_user.png

The screenshot shows the 'Users' section of the AWS IAM console. Two users are listed: 'Maham\_Saleem' and 'terraform-user'. Both users have 0 attached policies and were last active yesterday. The interface includes a search bar, sorting options for User name, Path, Group, Last activity, MFA, and Password age, and buttons for Create user, Delete, and View user.

q1\_user\_details.png

The screenshot shows the 'Maham\_Saleem' user details page. At the top right is a 'Delete' button. Below it is a 'Summary' section with the following details:

ARN arn:aws:iam::937451647605:user/Maha_m_Saleem	Console access Disabled	Access key 1 <a href="#">Create access key</a>
Created January 19, 2026, 12:44 (UTC+05:00)	Last console sign-in -	

Below the summary are tabs for 'Permissions', 'Groups', 'Tags', 'Security credentials', and 'Last Accessed'. The 'Permissions' tab is selected. Under 'Permissions policies (1)', there is a 'Remove' button and an 'Add permissions' button. A note says 'Permissions are defined by policies attached to the user directly or through groups.' There is also a 'Filter by Type' dropdown.

q1\_add\_user\_to\_group.png

A green success message at the top left reads '(✓) User added to group SoftwareEngineering'. The rest of the page is identical to the previous screenshot, showing the 'Maham\_Saleem' user details with the 'Groups' tab selected. The 'Groups (1)' section shows the 'SoftwareEngineering' group added, with 'Remove' and 'Add user to groups' buttons.

q1\_group\_membership.png

The screenshot shows the 'User groups membership' section for the 'Maham\_Saleem' user. It displays the 'SoftwareEngineering' group with 'Remove' and 'Add user to groups' buttons. A note below states: '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.' The 'Attached policies' section is empty.

The screenshot shows the AWS IAM SoftwareEngineering user group details page. The left sidebar includes a search bar and navigation links for Identity and Access Management (IAM), User groups, Users, Roles, Policies, Identity providers, Account settings, Root access management, and Temporary delegation requests. The main content area displays the SoftwareEngineering user group summary, showing it was created on January 19, 2026, at 12:35 (UTC+05:00) with an ARN of arn:aws:iam:937451647605:group/SoftwareEngineering. It lists one user, Maham\_Saleem, and provides options to edit or delete the group.

## 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.

The screenshot shows the 'Attach policies' dialog box. A search bar at the top contains the text 'AdministratorAccess'. A 'Filter by Type' dropdown is set to 'All types' and shows '4 matches'. Below the search bar is a table with columns: Policy name, Type, Used as, and Description. The table lists four AWS managed policies: 'AdministratorAccess-Am...', 'AdministratorAccess-AW...', 'AWSAuditManagerAdmin...', and 'AWSManagementConsol...'. At the bottom right of the dialog are 'Cancel' and 'Attach policies' buttons.

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...

## q1\_attach\_admin\_policy.png

Policies attached to this user group.

**SoftwareEngineering** Info

**Summary**

User group name: SoftwareEngineering

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

ARN: arn:aws:iam::937451647605:group/SoftwareEngineering

**Permissions**

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

**Actions:**

## 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.

**SoftwareEngineering**

**Actions:**

Search

Group name	Users	Permissions	Creation time
SoftwareEngineering	1	Defined	24 minutes ago

q1\_console\_user\_in\_group.png

The screenshot shows the 'Users in this group' section of the AWS IAM console. It displays a single user, Maham\_Saleem, with the following details:

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

Buttons at the top right include 'Remove' and 'Add users'.

q1\_console\_group\_policy.png

The screenshot shows the 'Permissions policies' section of the AWS IAM console. It displays one policy, AdministratorAccess, with the following details:

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

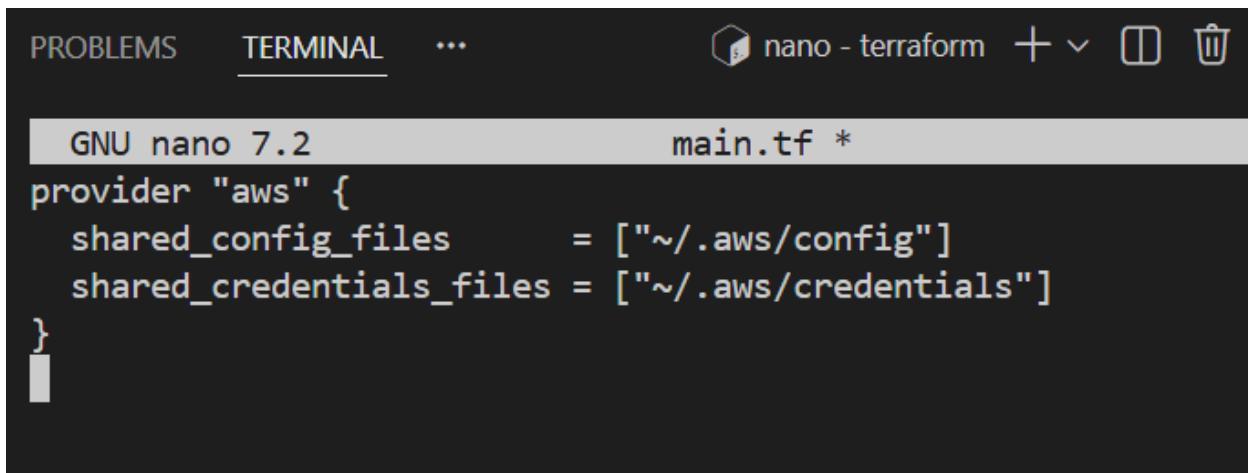
Buttons at the top right include 'Simulate', 'Remove', and 'Add permissions'.

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

The screenshot shows the VS Code terminal tab titled 'TERMINAL'. The output window displays a series of nano command executions, likely part of a Terraform workflow:

- @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

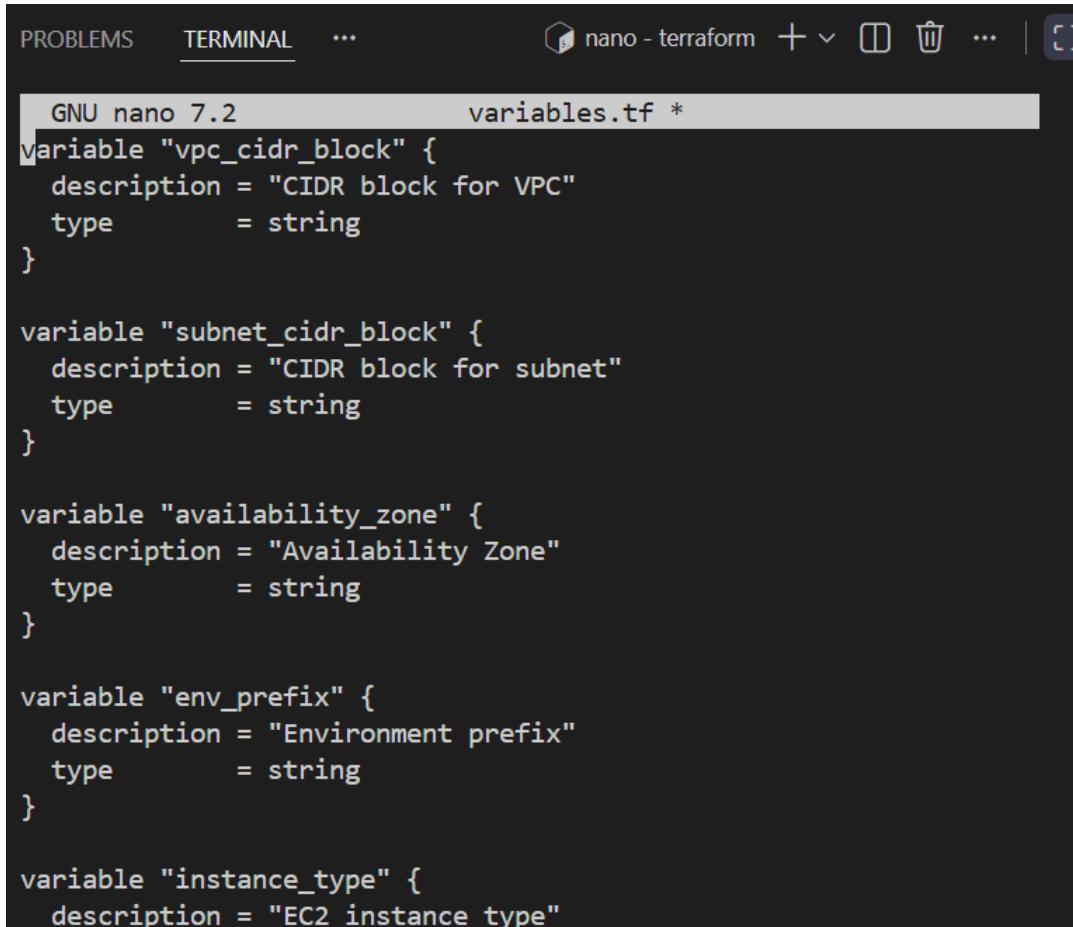


PROBLEMS TERMINAL ... nano - terraform + × ⌂ ⌂

```
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 + × ⌂ ⌂ ... | ⌂

```
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 screenshot shows a terminal window with the title "nano - terraform". The file is named "main.tf" and contains the following Terraform configuration:

```
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 screenshot shows a terminal window titled "nano - terraform" displaying Terraform configuration code. The code defines an Internet Gateway ("myapp\_igw") and a Default Route Table ("myapp\_rt"). The Internet Gateway is associated with a VPC, has a specific name, and includes tags. The Default Route Table is also associated with the VPC, has a specific name, and includes tags. The configuration uses AWS provider resources like "aws\_internet\_gateway" and "aws\_default\_route\_table".

```
PROBLEMS TERMINAL ...
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"
  }
}

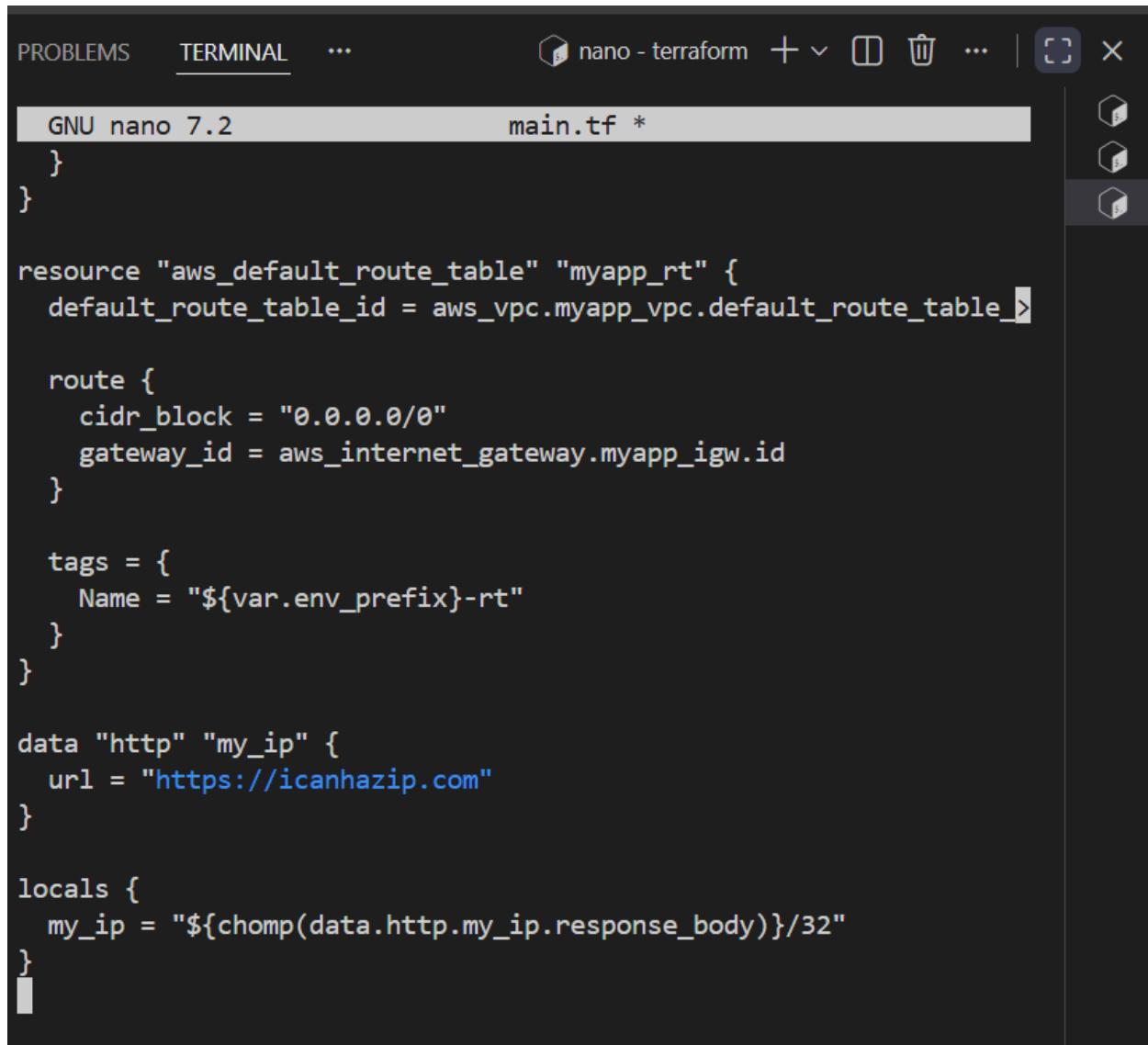
resource "aws_default_route_table" "myapp_rt" {
  default_route_table_id = aws_vpc.myapp_vpc.default_route_table_id

  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



The screenshot shows a terminal window with the title "nano - terraform". The file being edited is "main.tf". The code is written in Terraform, defining an AWS Default Route Table and a local variable to store the IP address from an HTTP request.

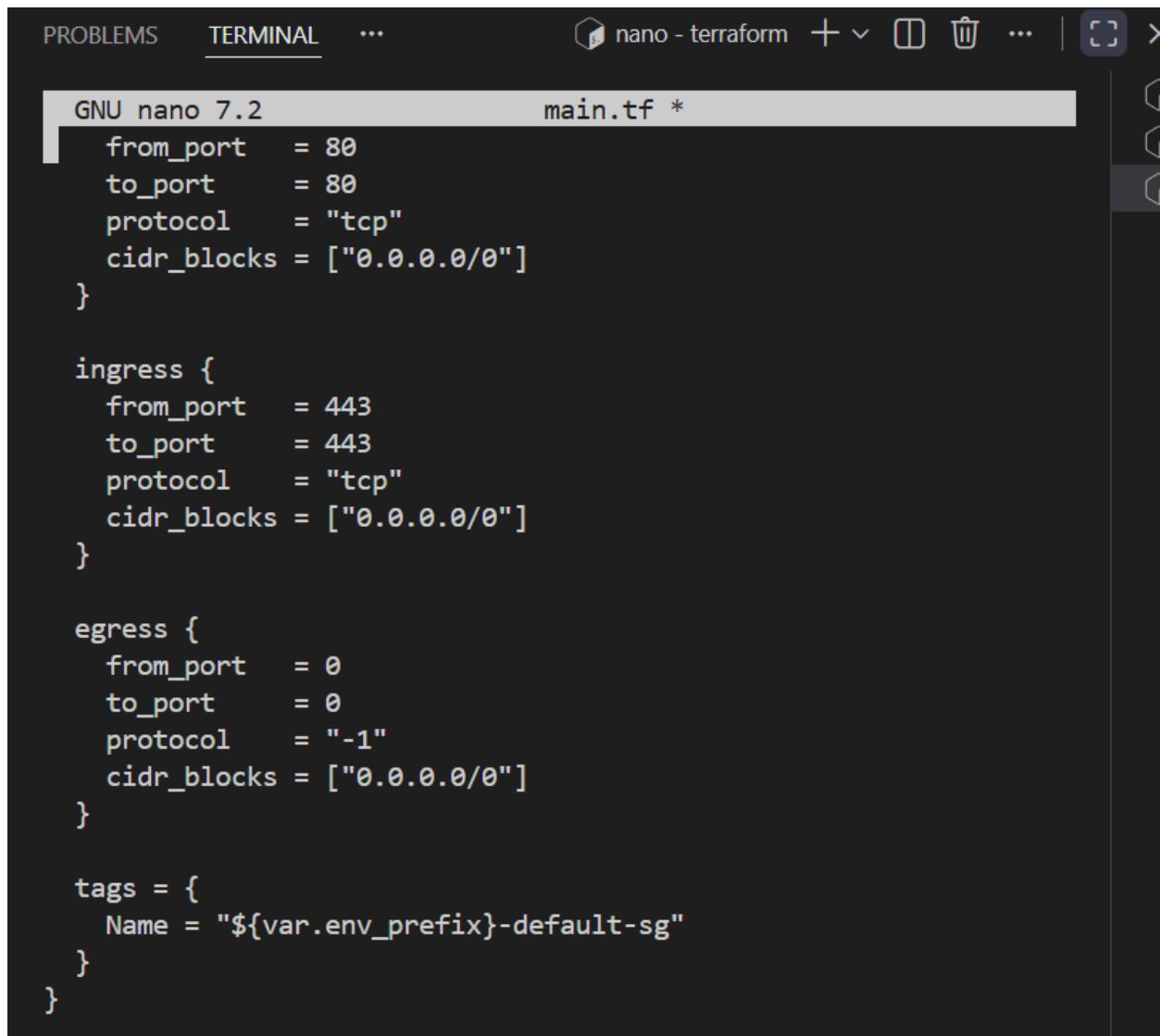
```
GNU nano 7.2          main.tf *
```

```
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 screenshot shows a terminal window titled "nano - terraform" displaying a Terraform configuration file named "main.tf". The file contains the following code:

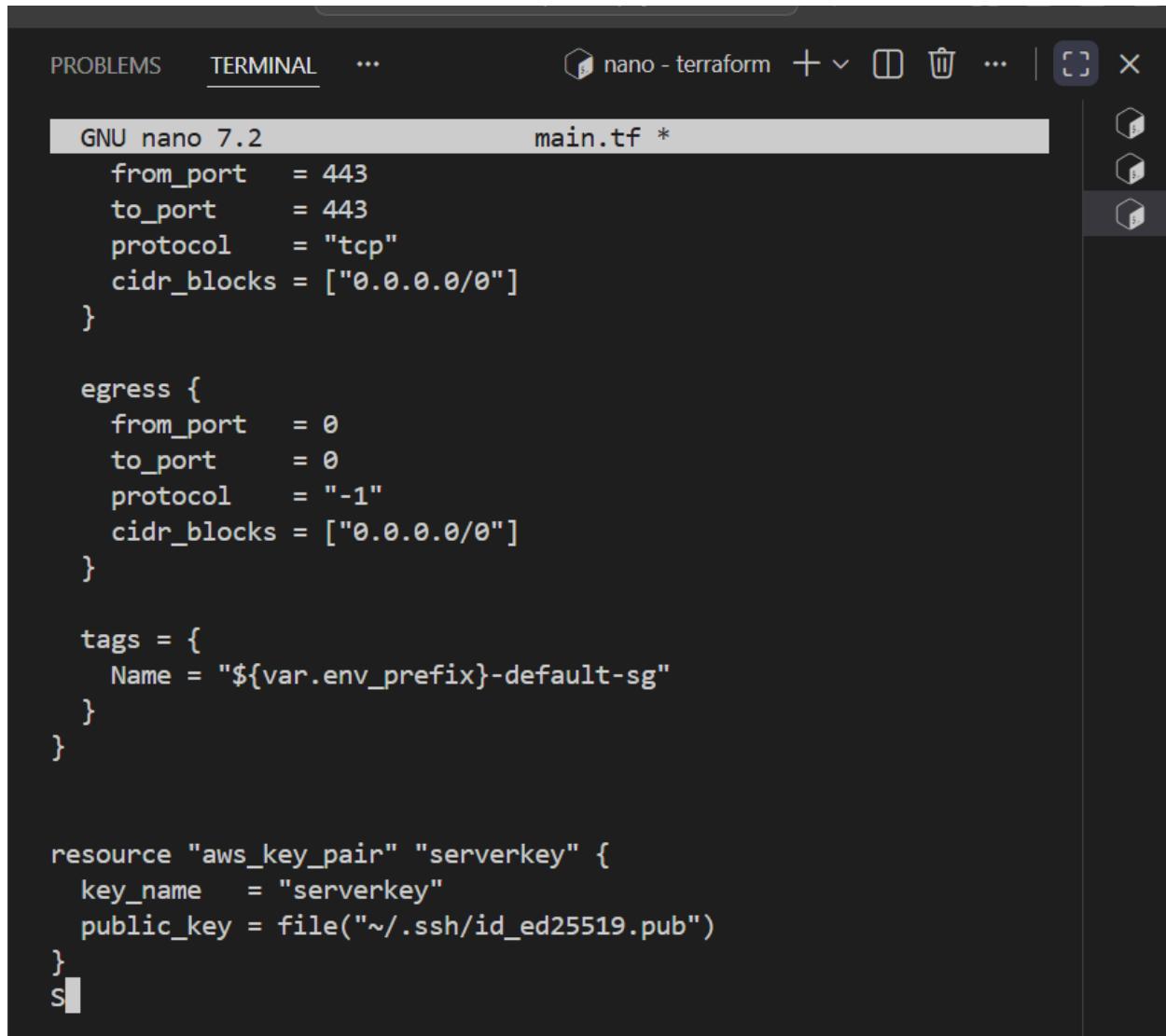
```
GNU nano 7.2                               main.tf *
  from_port     = 80
  to_port       = 80
  protocol      = "tcp"
  cidr_blocks  = ["0.0.0.0/0"]
}

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



The screenshot shows a terminal window titled "nano - terraform" displaying Terraform configuration code. The code defines a security group (sg) with specific port rules and tags, and also creates an AWS key pair named "serverkey". The configuration is saved in a file named "main.tf".

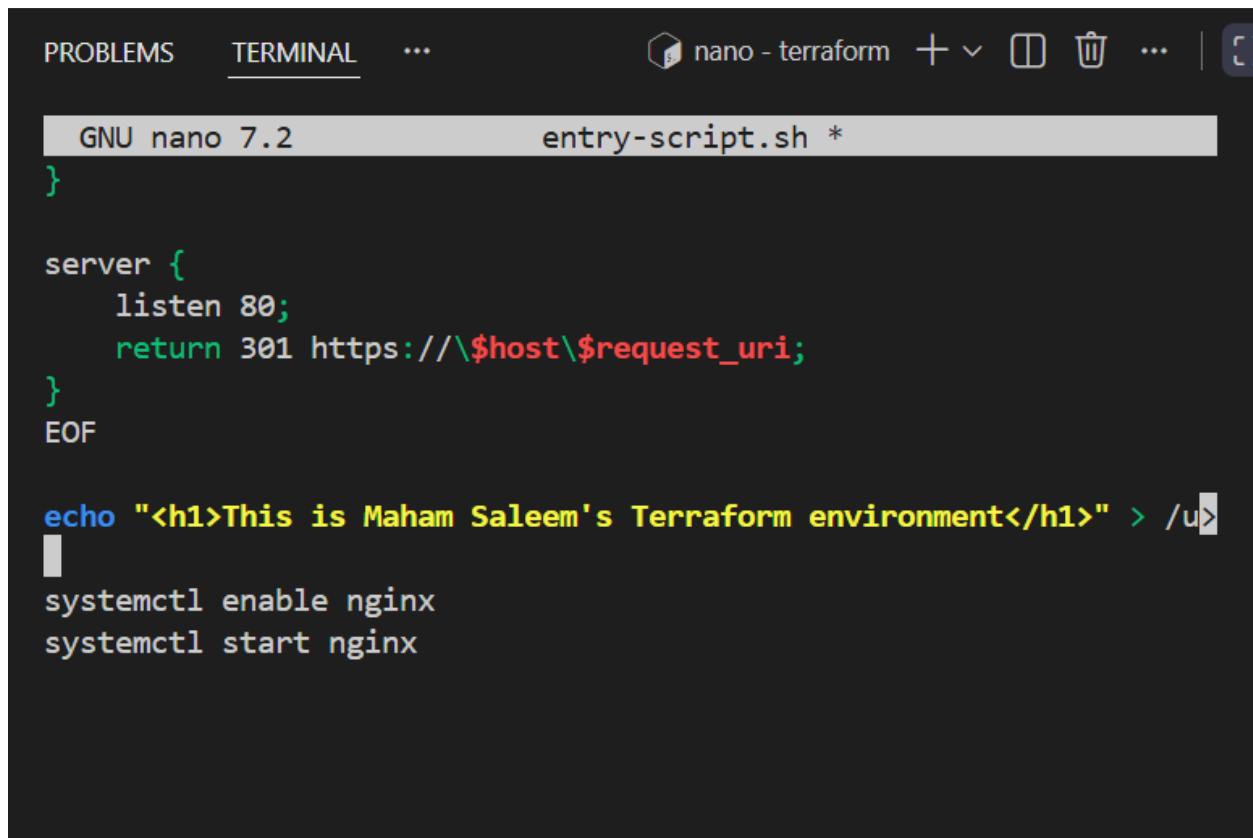
```
GNU nano 7.2                               main.tf *
from_port      = 443
to_port        = 443
protocol       = "tcp"
cidr_blocks   = ["0.0.0.0/0"]
}

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")
}
```

q2\_entry\_script.png



The screenshot shows a terminal window titled "nano - terraform". The file being edited is "entry-script.sh". The code in the terminal 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
```

## q2\_ec2\_resource.png

The screenshot shows a terminal window titled "nano - terraform" displaying a Terraform configuration file named "main.tf". The file contains the following code:

```
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

The screenshot shows a terminal window titled "nano - terraform" displaying a Terraform configuration file named "main.tf". The code defines an AWS EC2 instance named "myapp\_ec2" with specific parameters like AMI, instance type, and security group. It also includes a key pair and user data. An output block is present to retrieve the public IP address of the 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.id]
  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
}
```

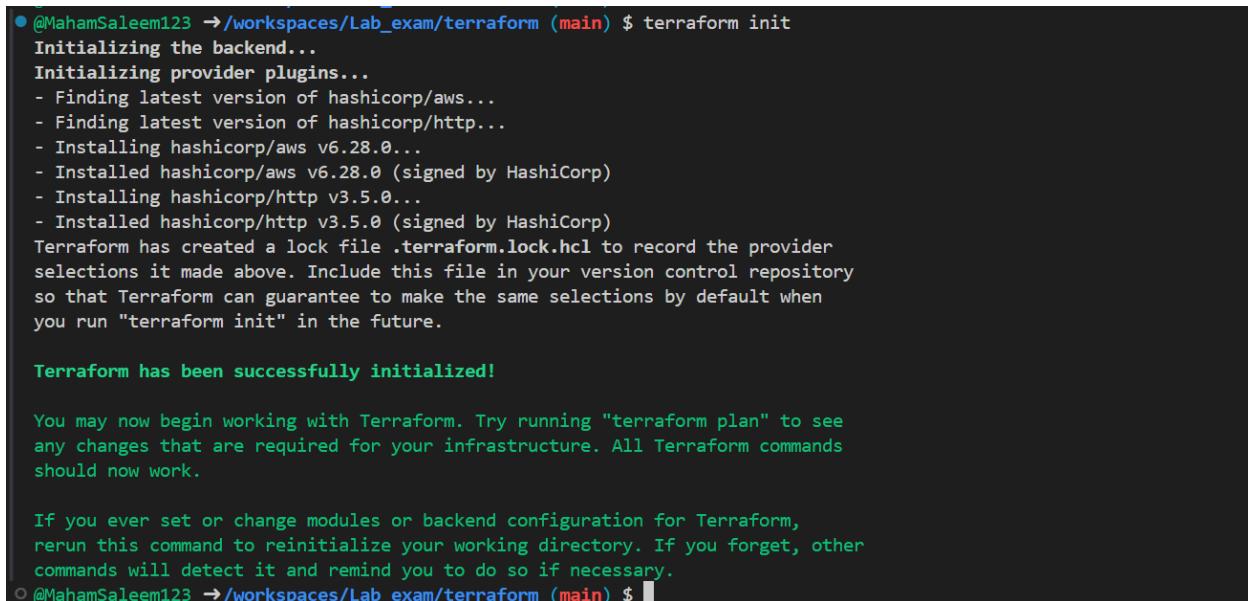
## q2\_tfvars\_or\_vars.png



The screenshot shows a terminal window with the title "nano - terraform". The file being edited is "terraform.tfvars". The content of the file is as follows:

```
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



The screenshot shows a terminal window with the title "Terminal". The command "terraform init" was run, and the output is as follows:

```
● @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) $
```