BYOD 2

CSV -205

Reg No: 12303493

Part 1: Project Setup & Remote State Management (25 Marks)
This part focuses on establishing a collaborative and stable environment before creating any infrastructure.

1. Provider & Versioning (5 Marks)

- Create a providers.tf file.
- Configure the aws provider with a constraint to use a stable version (~> 5.0 or later).
- Use a terraform block to enforce a minimum Terraform version (e.g., required_version = ">= 1.3.0").

2. AWS Remote Backend Setup (15 Marks)

- **Prerequisite:** The S3 bucket and DynamoDB table must exist before running terraform init. You must create these resources manually using the AWS Console or AWS CLI.
    - **S3 Bucket:** Name it [your-name]-terraform-state-bucket.
    - **DynamoDB Table:** Name it [your-name]-terraform-lock-table with the primary key LockID (String type).
- Create a dedicated file named **backend.tf**.
- Configure the s3 backend within the terraform block in backend.tf.
    - Use your newly created bucket and table names.
    - Set the state file key to "prod/webserver/terraform.tfstate".
    - Set the region to us-east-1 (or your preferred region).
    - **Security:** Enable encrypt for server-side encryption.

3. Initialization (5 Marks)

- Run terraform init.
- **Deliverable:** A screenshot showing a successful initialization that explicitly states: **"Successfully configured the backend 's3'."**

```
❯ terraform init -reconfigure
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.26.0

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.
```

Part 2: Networking (30 Marks)

Build the necessary network plumbing.

1. The VPC (10 Marks)

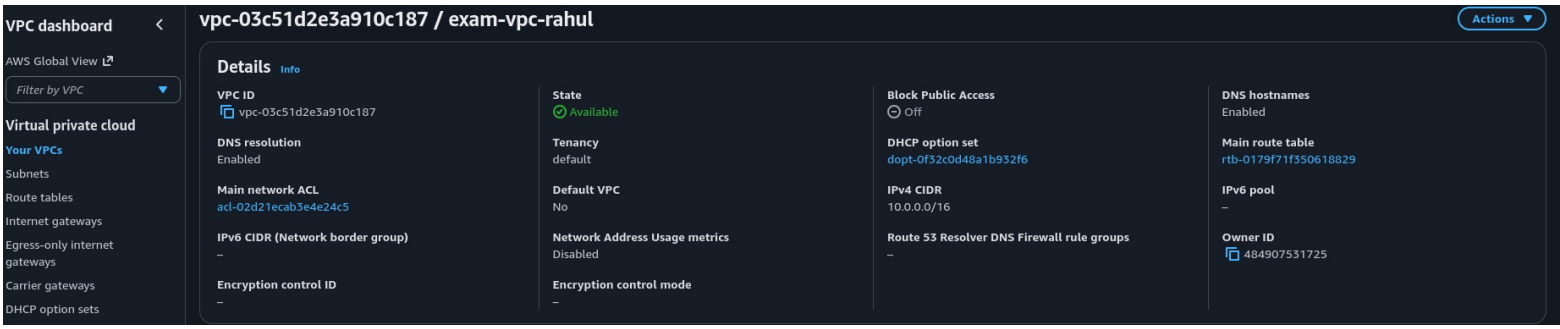
- Create a networking.tf file.
- Define an **AWS VPC** using the CIDR block **10.0.0.0/16**.
- **Tagging:** Tag the VPC with Name = "exam-vpc-[your-name]".

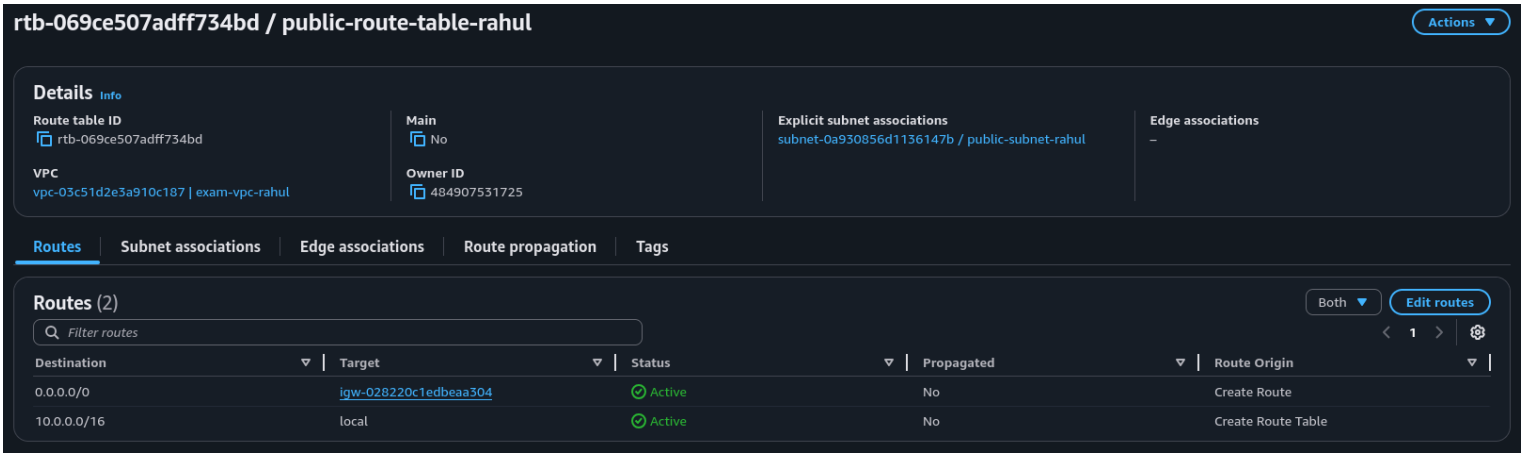
2. Internet Access (10 Marks)

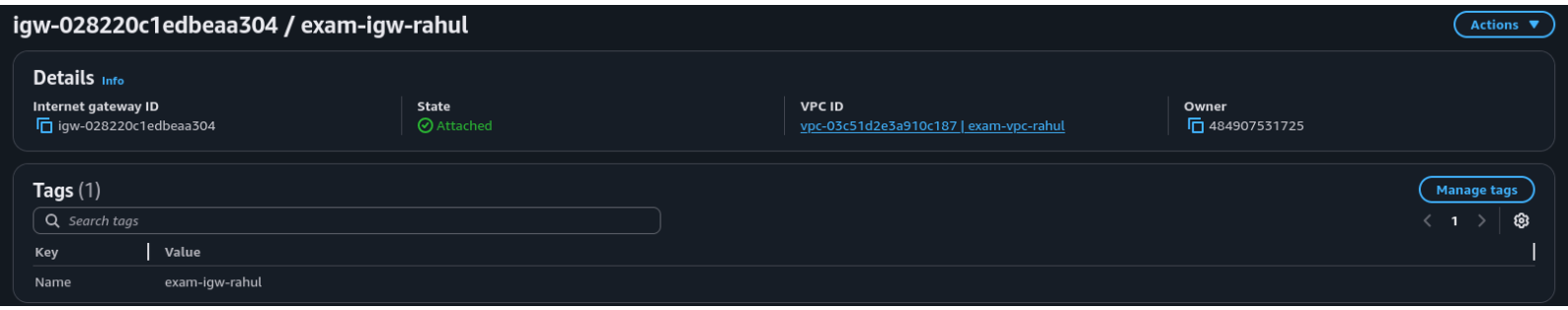
- Create an **Internet Gateway** and attach it to your VPC.
- Create a **Public Subnet** with CIDR **10.0.1.0/24**.
- **Critical:** Ensure map_public_ip_on_launch is set to true.

3. Routing (10 Marks)

- Create a custom **Route Table** for the public subnet.
- Add a route that directs traffic 0.0.0.0/0 to the Internet Gateway.
- Association: Explicitly associate this Route Table with your Public Subnet.







Part 3: Security & Identity (15 Marks)

Secure the instance and prepare for access.

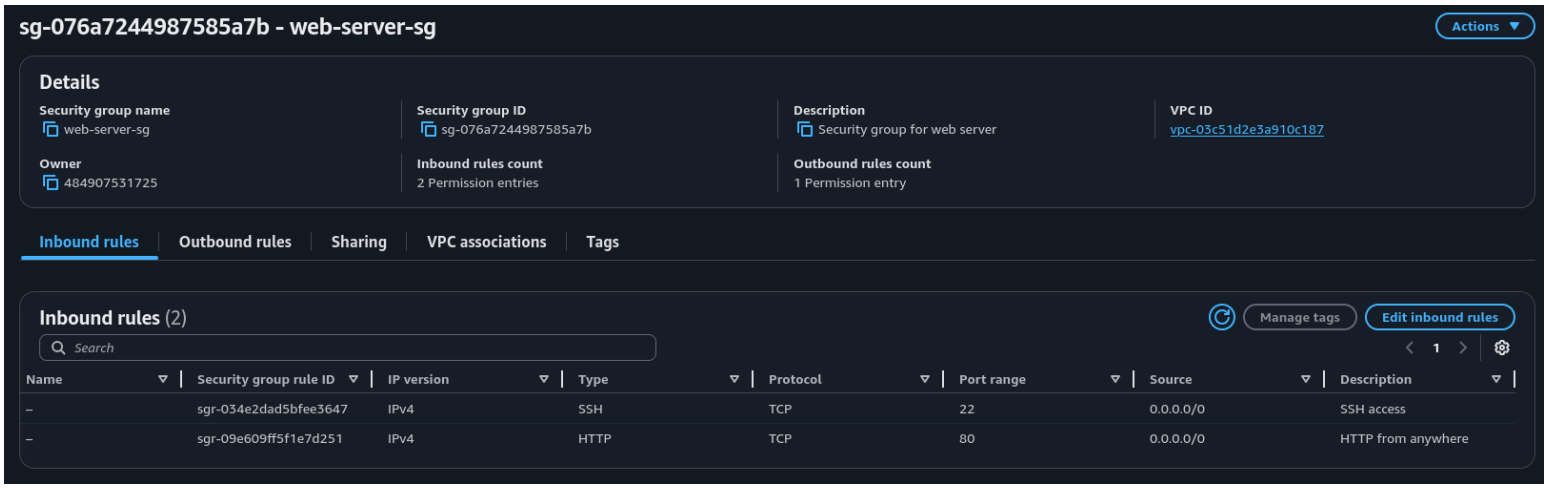
1. SSH Key Pair (5 Marks)

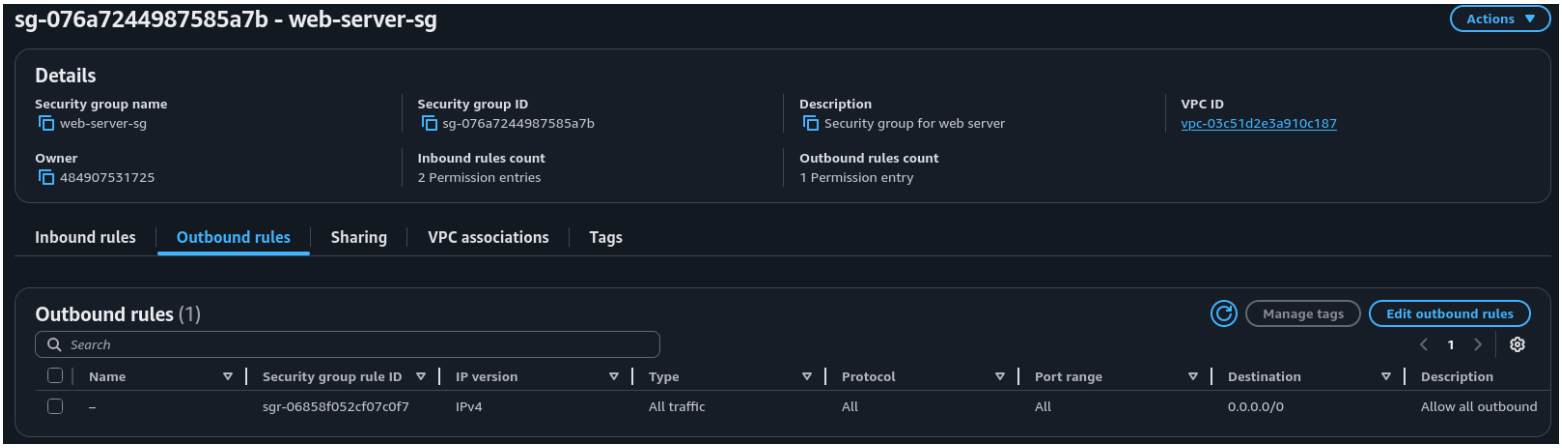
- Use the aws_key_pair resource to upload your local public key to AWS.

2. Security Groups (10 Marks)

- Create a Security Group named web-server-sg.
- **Ingress Rules:**
  - Allow HTTP (Port 80) from 0.0.0.0/0.
  - Allow SSH (Port 22) from your specific IP (or 0.0.0.0/0 if necessary for the exam).
- **Egress Rules:**
  - Allow all traffic outbound (0.0.0.0/0).

```
resource "aws_key_pair" "my_key" {
  key_name   = "rahul-key-pair"
  public_key = file("~/.ssh/id_rsa.pub")
}
```





Part 4: Compute, Variables & Outputs (30 Marks)

Deploy the server and finalize the configuration.

1. Variables (5 Marks)

- Create a variables.tf file.
- Define variables for: region, vpc_cidr, and instance_type.
- Reference these variables in your configuration.

2. The EC2 Instance (5 Marks)

- Deploy a **t3.micro** EC2 instance into your **Public Subnet**.
- Use a data source to look up the latest **Amazon Linux 2** AMI.
- Attach the correct Security Group and Key Pair.

3. User Data Configuration (15 Marks)

- Use the user_data argument.
- **The Script Logic:**
    1. Install the Nginx web server.
    2. Start and enable the Nginx service.
    3. **Custom Challenge:** Overwrite the default Nginx welcome page with an HTML file that contains the text:

"<h1>Project Genesis SUCCESS! Deployed by [Your Name]</h1>"

4. Outputs (5 Marks)

- Create an outputs.tf file.
- Output the **Web URL** of the server (e.g., http://<public-ip>).

Web url for website http://35.175.111.77/

GITHUB LINK : https://github.com/Rahulx0/BYOD2.git