**TERRAFORM PROJECT**

**Objective:**  
Create a secure VPC environment in AWS using Terraform where an EC2 instance is running an Nginx web server. The EC2 instance should reside within a private subnet and should be accessible to the outside world via a load balancer. Traffic to the EC2 instance should be routed through a NAT gateway.  
  
Requirements:  
  
  
**VPC Setup:**  
a. Create a new VPC.  
b. Configure appropriate CIDR blocks for the VPC and subnets.  
c. Create at least two subnets: one public and one private.  
d. Associate the public subnet with an Internet Gateway.  
e. Set up a NAT Gateway in the public subnet.  
  
**EC2 Instance:**  
a. Launch an EC2 instance within the private subnet.  
b. Ensure the EC2 instance does not have a public IP.  
c. Install and run Nginx on the EC2 instance.  
Hint: You can use user\_data to bootstrap the EC2 instance with Nginx.  
  
  
**Load Balancer:**  
a. Set up a load balancer (either Application Load Balancer or Network Load Balancer based on your preference).  
b. Ensure that the Load Balancer routes traffic to the Nginx EC2 instance.  
c. The Load Balancer should be associated with the public subnet and should have a public-facing listener.  
  
**Routing:**  
a. Ensure that traffic from the load balancer to the EC2 instance is routed through the NAT Gateway.  
b. Set up necessary route tables and associations to ensure correct traffic flow.  
  
  
**Security Groups and Network ACLs:**  
a. Secure your infrastructure. Only allow necessary ports (e.g., HTTP/HTTPS for web traffic, SSH only from specific IPs, etc.).  
b. Ensure that the EC2 instance can reach the outside world for updates or any other necessary outbound communication.  
  
  
**Outputs:**  
a. Provide an output for the Load Balancer's DNS name so it can be easily accessed for testing.  
b. Have a file with store.tf to pass all critical values to the SSM Parameter Store.  
  
**Deliverables:**  
a. A well-organized Terraform directory structure with separate files for resources (e.g., vpc.tf, ec2.tf, etc.).  
b. Push your codes to your own personal github account and make it public. Share the link to the remote repository in the space provided for the answers here in Google Classroom.  
c. Your repository must have a README.md explaining:

* How to deploy the infrastructure.
* Any assumptions made.
* Steps to validate the setup (e.g., accessing the Nginx page via the Load Balancer).

d. A Terraform outputs.tf file containing relevant outputs.  
  
**Evaluation Criteria:**

* Correctness: The infrastructure should be set up as described and should be functional.
* Security: Ensure that there are no security loopholes. Resources in the private subnet, especially, should not be directly accessible from the public Internet.
* Code Quality: Terraform code should be clean, organized, and modular. Comments and documentation are a plus.
* Idempotency: Running terraform apply multiple times should not result in changes unless the configuration has changed.

**Bonus Points:**

* Implementing a Terraform module for reusable components.
* Automated tests for your infrastructure.
* Setting up an Nginx reverse proxy or any advanced configuration.