**I am tasked with building an infrastructure for a web application using AWS with Terraform. The infrastructure needs to be highly available across multiple Availability Zones and consist of the following components:**

1. **1. A VPC with subnets across 2 availability zones.**
2. **2. Two EC2 instances (web servers) in an Auto Scaling Group with a load balancer distributing traffic between them.**
3. **3. Security groups to allow HTTP traffic from the internet to the load balancer and SSH traffic to the web servers only from a specific CIDR block.**
4. **4. Use Terraform modules to encapsulate reusable components for VPC, Auto Scaling Group, and Security Groups.**
5. **5. The EC2 instances should have a provisioner to install NGINX on each instance after it's launched.**
6. **6. Use a remote backend (such as S3) to store the Terraform state securely.**
7. **7. Fetch an existing AMI for the EC2 instances dynamically using a data source.**
8. **8. Ensure the instance type of the EC2 instances can be dynamically selected based on an environment variable (like dev, prod).**
9. **9. Set up an output that will return the DNS name of the load balancer.**
10. **Requirements:**
11. **1. Create Terraform Modules:**
12. **VPC Module: Should create a VPC with subnets in 2 availability zones.**
13. **Auto Scaling Module: Should create the auto-scaling group for EC2 instances.**
14. **Security Group Module: Should create security groups for the load balancer and EC2 instances.**
15. **Use a Terraform provisioner to install NGINX on the EC2 instances.**
16. **Use the AMI ID fetched dynamically using a data source.**
17. **Store the state in an S3 bucket and enable state locking with DynamoDB.**
18. **Use a variable for environment (dev or prod). In the dev environment, use t2.micro instances; in the prod environment, use t3.medium instances.**
19. **Output the DNS of the Load Balancer.**

------------------------------------------------------------------------------------------------------

**Objective**

*To design and provision a highly available web application infrastructure using Terraform, incorporating modularity, remote state management, dynamic AMI selection, and conditional configuration based on environments.*

**1. Directory Structure**

terraform-project/

├── main.tf

├── variables.tf

├── outputs.tf

├── provider.tf

├── modules/

│ ├── vpc/

│ │ ├── main.tf

│ │ ├── variables.tf

│ ├── security\_group/

│ │ ├── main.tf

│ │ ├── variables.tf

│ ├── auto\_scaling/

│ ├── main.tf

│ ├── variables.tf

**3. Execution Steps**

1. **Initialize Terraform**:

terraform init

1. **Validate Configuration**:

terraform validate

1. **Plan Infrastructure**:

terraform plan -var="environment=dev"

1. **Apply Configuration**:

terraform apply -var="environment=dev" -auto-approve