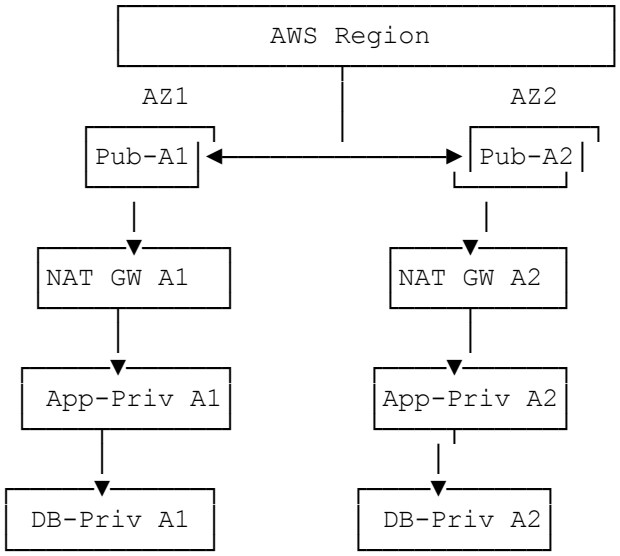
Project Report: Terraform-Based AWS Architecture (3-Tier)

🎯 Objective

To deploy a **highly available**, **scalable**, and **secure 3-tier architecture** across 2 Availability Zones using Terraform, complete with frontend (web), backend (app), and database (DB) layers.

🏗️ Terraform Architecture Overview

```
markdown
CopyEdit
```



📦 Components & Configuration

✅ Region & AZs

- **1 AWS Region**
- **2 Availability Zones** used for redundancy

✅ Networking

- **1 VPC**
- **6 Subnets:**
  - 2 Public (Web Tier)
  - 2 Private (App Tier)
- **IGW**: Attached to VPC for internet access
- **NAT Gateway (1 per Public Subnet)**: Allows private subnets to access internet securely
- **Route Tables**:
  - Public Subnet → IGW route
  - Private Subnets → NAT GW routes

✅ Auto Scaling & Load Balancing

- **Launch Templates**:
  - Web/Frontend AMI and config
  - App/Backend AMI and config
- **Auto Scaling Groups**:
  - For both **Frontend (Web)** and **Backend (App)** layers
  - Multi-AZ deployments
- **Application Load Balancers**:
  - **Public ALB**: Handles external HTTP/HTTPS traffic (Web Tier)
  - **Private ALB**: Internal routing to App Tier
- **Listener Rules**:
  - HTTP → HTTPS redirect
  - Path/Host-based rules for internal routing
- **Target Groups**: Mapped to ASG instances (FE and BE)

✅ Compute (EC2)

- **PIP (for Bastion/ALB)**
- **Key Pair for SSH**
- **AMI for Web and App instances**
- **Instance Types**: e.g., `t2.micro`, `t3.medium`
- **EBS Volume Configs**
- **Network Security Groups**:
  - Separate NSGs for Web, App, and DB
  - Allow minimum required ports (SSH, HTTP/S, DB access)
- **Metadata & Tags** for management and cost control

✅ Database Layer (RDS)

- **Multi-AZ RDS Deployment**
- Private subnet only, no public access
- **MySQL/PostgreSQL Engine**
- Security Group allows access from App tier only
- Automatic backups and failover configuration

🗃️ Terraform Modules & Files

- **Modules**:
  - `vpc/`, `subnet/`, `ec2/`, `alb/`, `rds/`, `asg/`, etc.
- **Files**:
  - `main.tf`: Orchestrates modules
  - `variables.tf`: Input variables
  - `terraform.tfvars`: Environment-specific values
  - `outputs.tf`: Export key values like DNS, IPs, etc.
  - `backend.tf`: Remote state backend (e.g., S3 + DynamoDB)

🧪 Testing & Validation

- Verified route propagation and subnet association
- Simulated load to test ASG scale-out/in
- Accessed web layer via ALB DNS and validated backend response
- Ensured RDS was accessible only from App layer