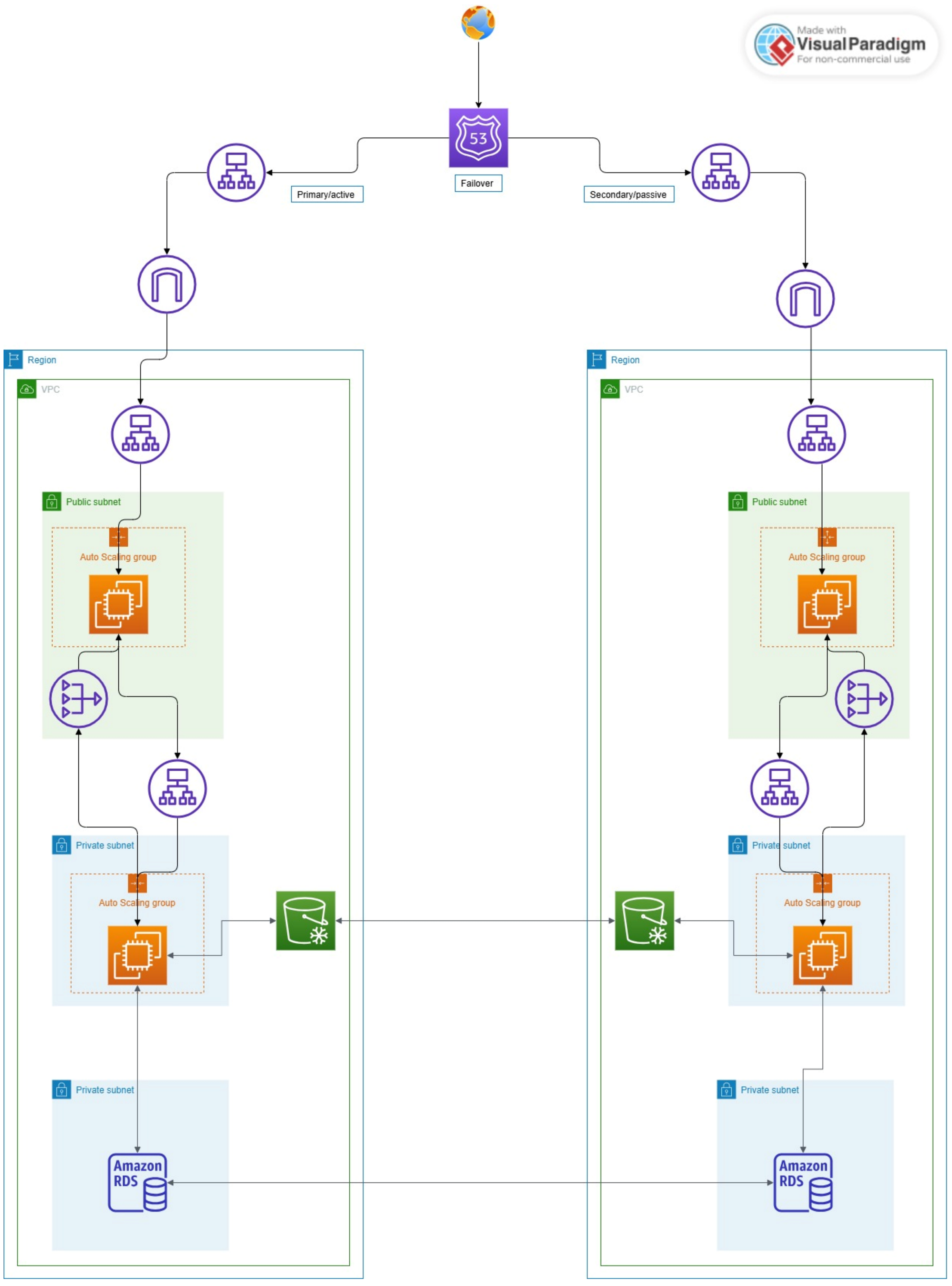
Goal:
To design and implement a multi-region disaster recovery solution using AWS services, where infrastructure is deployed in two AWS regions. The solution will replicate data across regions, ensure high availability, and implement a failover mechanism for disaster recovery. The project involves using Terraform to provision infrastructure, and CI/CD pipelines to ensure both regions remain synchronized for disaster recovery and failover.

Services:

Component	**Role in Architecture**
Amazon VPC	Creates isolated networks in each region to host your resources securely.
Amazon S3	Stores data (e.g., app data, backups) and replicates it to another region using **S3 Cross-Region Replication (CRR)**.
Amazon RDS	Hosts your databases; uses **read replica in another region** or **Aurora Global Database** for cross-region sync.
Amazon Route 53	Manages DNS routing; configured with **health checks** and **failover policies** so it redirects traffic to the DR region if the primary fails.
Terraform	Automates infrastructure deployment in both regions, ensuring consistency and version control.
Jenkins / AWS CodePipeline	Automates CI/CD workflows to deploy code and infra updates to both regions simultaneously.

Architecture:

To implement a three tier architecture using aws, with implementation of aws services done by terraform script via ci cd pipeline in Jenkins.



Module Structure:

Create a terraform project to launch the aws services. We will be using a modular approach where we create modules for each services and use them in main.tf

```
DisasterRecovery/
main.tf             # Root module orchestrator
variables.tf          # Variable declarations
terraform.tfvars       # Environment-specific values
networking/           # VPC, subnets, route tables
security_groups/       # Tiered security group definitions
alb_asg/              # ALB and Auto Scaling Group modules
rds/                 # RDS instance configuration
s3/                  # Cross-region replicated S3 buckets
```

Architecture Components

Networking & VPC

- Two VPCs (one per region)
- Three subnet tiers per VPC:
  - **Public Subnets:** Host public ALBs and Web EC2 instances
  - **Application Subnets:** Host internal ALBs and App EC2 instances
  - **Database Subnets:** Host RDS in private subnets
- NAT Gateways in public subnets for outbound internet access from private subnets

Security Groups

- **External ALB SG:** Allows inbound HTTP/HTTPS traffic from the internet
- **Web EC2 SG:** Allows inbound HTTP traffic from the ALB
- **App EC2 SG:** Allows inbound traffic from Web tier on port 8080
- **DB SG:** Allows inbound MySQL traffic from the App tier only

Compute & Load Balancing

- **4 ALBs:**
    - 2 Web ALBs (one per region, public)
    - 2 App ALBs (one per region, internal)
- **4 Auto Scaling Groups:**
    - Web and App tiers in each region
- Launch templates install Apache and configure test endpoints for validation

Storage

- RDS MySQL instance deployed in the primary region (us-east-1)
- Cross-region S3 replication for backups and Terraform state sharing

High Availability & Disaster Recovery

- **Route 53 Failover Routing:**
    - Primary ALB (us-east-1) receives traffic when healthy
    - Secondary ALB (us-east-2) automatically takes over if the primary becomes unhealthy
- Multi-region architecture ensures minimal downtime and robust resilience