***DevOps Assessment Submission – Darwix AI***

# 1. Infrastructure as Code (Terraform Configuration)

The provided Terraform script provisions a scalable, secure, and highly available AWS infrastructure to host a web application using NGINX on EC2. The configuration includes auto-scaling, load balancing, and robust networking.

### Key Components:

#### *1.1. VPC & Subnets*

- Custom VPC: `10.0.0.0/16`
- Two Public Subnets and Two Private Subnets across Two Availability Zones

#### *1.2. Networking*

- **Internet Gateway:** Provides public access.
- **NAT Gateway:** Located in the public subnet for outbound access from private subnets.
- **Route Tables:** Configured for public and private subnet routing.

#### *1.3. Security Groups*

- **ALB Security Group:** Allows HTTP/HTTPS from the internet.
- **EC2 Security Group:** Restricts HTTP access to EC2 instances only from the ALB.
- **SSH Security Group:** Allows SSH access only from trusted IP addresses.

#### *1.4. Compute*

- **Launch Template:** Configured to install and start NGINX with a sample static page.
- **Auto Scaling Group:** (Min: 1, Max: 3) deployed in private subnets using the launch template.

### *1.5. Load Balancer*

- **Application Load Balancer (ALB):** Deployed across public subnets.
- **Target Group:** Forwards traffic to EC2 instances.
- **ALB Listener:** Routes HTTP traffic to the target group.

### *1.6. Deployment*

- **EC2 Instances Registration:** Instances are registered with the ALB via Auto Scaling Group (ASG) attachment.
- **User Data:** Bootstraps NGINX and displays a welcome message.

*Main.tf script is included with the PDF file.*

---

## 2. EC2 Bootstrapping and Configuration

The bootstrapping process automates the initial setup of EC2 instances using a shell script. Below is an overview of the steps involved in the script execution.

### What the Script Does:

- **System Update:** Ensures all packages are up-to-date using `yum update`.
- **Install & Configure NGINX:**
  - Installs NGINX via Amazon Linux extras.
  - Enables and starts the NGINX service.
  - Deploys a simple HTML welcome page (`index.html`).
- **Install CloudWatch Logs Agent:**
  - Installs the AWS CloudWatch Logs agent to push logs to Amazon CloudWatch.
  - Configures log groups and streams for:
    - `/var/log/messages` (system logs)
    - NGINX access and error logs.
- **Region Configuration:** Sets the AWS region for the log agent (`us-east-1`).

- **Enable & Start Logging Service:** Activates the log agent to ensure logs are pushed during runtime and boot.

*Shell scripts (cloudwatch.sh and user-data.sh) are included with the PDF file.*

---

## 3. CI/CD Automation Design

A Jenkins pipeline has been created to automate the Continuous Integration and Continuous Deployment process for the application. Below is the overview of the pipeline.

### Jenkins CI/CD Pipeline Overview:

1. **Checkout:** Pulls source code from the GitHub repository (master branch).
2. **Build:** Copies the `index.html` file into the `build/` directory.
3. **Package:** Zips the `build/` folder into `app.zip`.
4. **Upload:** Uploads `app.zip` to an S3 bucket (`s3-bucket-for-darvixassessment`) using Jenkins credentials.
5. **Deploy:** Triggers the deployment by updating the Auto Scaling Group (ASG) capacity to refresh EC2 instances.

### Trigger Mechanism:

- The pipeline is triggered via a GitHub webhook. Every push to the repository automatically starts the build and deployment process.

### Rollback Steps:

- **Artifact Rollback:** In case of failure, the artifact can be rolled back to a previous version stored in S3.
- **ASG Rollback:** The Auto Scaling Group can be reset to the previous launch template version.

### Environment Promotion Strategy:

- **Staging Deployment:** Initially deploy to the staging ASG.
- **Production Deployment:** On approval, promote to production via Jenkins.

*Jenkinsfile is included with the PDF file.*

---

## 4. Security and Compliance Walkthrough

Security best practices have been followed to ensure the system adheres to both security and compliance requirements. Below are the implemented security measures:

### IAM Roles & Policies:

- Created an **EC2 role** with permissions for CloudWatch Logs and read-only access to S3.
- **Jenkins IAM User** has scoped access for `s3:PutObject` and `autoscaling:UpdateAutoScalingGroup`.
- Enforced **bucket policies** to restrict access and enforce SSL.

### Secret Management:

- Ensured that no hardcoded credentials are present in scripts.
- Recommended using **AWS Secrets Manager** or **SSM Parameter Store** with **KMS** for securely storing sensitive data.

### Encryption:

- **Data at Rest:** Enabled **SSE (Server-Side Encryption)** for S3 and encrypted EBS volumes.
- **Data in Transit:** Enforced **HTTPS** for ALB and S3 access.

### Governance:

- **CloudTrail:** Enabled to log all API activities across AWS.
- **GuardDuty:** Enabled for continuous security monitoring to detect threats and anomalies.

*EC2Role.json policy file is included with the PDF file.*

---

**Attachments:**

- main.tf (Terraform script)
- cloudwatch.sh (Bootstrapping script for CloudWatch)
- user-data.sh (EC2 bootstrapping script)
- Jenkinsfile (CI/CD pipeline)
- EC2role.json (IAM policy for EC2 role)