Technical Challenge: DevOps & Cloud Automation Engineer

1. Overview

The goal of this challenge is to evaluate your ability to design and implement a secure, scalable, and efficient **Continuous Delivery (CD)** lifecycle on **AWS** using **GitHub Actions** and **Infrastructure as Code (IaC)**.

2. The Scenario

You are required to deploy a containerized web application (e.g., Nginx, or a simple Node.js/Python app) onto **Amazon APP Runner or Amazon ECS Fargate**. The application must be accessible via a Load Balancer and must satisfy a specific networking requirement: **all outgoing traffic to the internet must originate from a static public IP address.**

Technical Requirements

A. Infrastructure as Code (IaC)

Define your infrastructure using **CloudFormation, Terraform or AWS CDK.**

- **Networking:** A VPC with Public and Private subnets across at least 2 Availability Zones.
- **Static Outbound IP:** Implement the necessary architecture (NAT Gateway + Elastic IP) so the application's outbound requests always use a predictable IP.
- **Compute:** An ECS Cluster running a Fargate Service.
- **Traffic Ingress:** An Application Load Balancer (ALB) to expose the service.

B. CI/CD Pipeline (GitHub Actions)

The pipeline must be orchestrated with the following stages:

1. **Validation:** Linting for IaC and security scanning
2. **Build:** Build the Docker image and push it to **Amazon ECR**.
3. **Preview:** Generate an infrastructure `diff` or `plan` and post it as a comment in the Pull Request.
4. **Deploy:** Automated deployment to a "Staging" environment upon merging to the main branch.

---

3. Security & Governance (Mandatory)

- **AWS Connection:** Determine and implement the most secure method for deployment from GitHub.
- **Least Privilege:** The IAM Role assumed by GitHub Actions must have the minimum permissions required for the deployment.
- **Network Segregation:** The application tasks must reside in **private subnets**.

---

4. Deliverables

1. **Source Code:** A link to a public repository (or invitation to a private one) containing the app code, IaC templates, and GitHub Workflow YAMLs.
2. **Architecture Diagram:** A diagram representing:
    - **Control Plane:** The CI/CD flow from `git push` to AWS authentication.
    - **Data Plane:** The VPC topology showing the path of an incoming request (Ingress) and the path of an outbound request (Egress via NAT/EIP).
3. **README.md:** Clear instructions on how to replicate the deployment and a brief explanation of the security choices made.

---

5. Follow-up Questions (Written Response)

Please include answers to the following in your documentation:

1. **Performance:** How would you optimize the GitHub Actions workflow to reduce Docker build times for a large application?
2. **Reliability:** If the IaC deployment fails halfway through, how do you ensure the environment remains stable?
3. **Observability:** How would you verify that the application is healthy after a deployment, and how would you automate a rollback if it isn't?
4. **Cost Optimization:** NAT Gateways incur hourly charges. If the app only needs to sync data with an external API once a day, what alternative would you propose to reduce costs?