Bonmoja Take-home assignment: Senior DevOps Engineer

Context

This assignment simulates real-world work: provisioning AWS infrastructure, automating deployment, and setting up observability — with a focus on **RDS, ECS, DynamoDB, SQS, and SNS**.

Your task

Design, provision, and document an AWS-based infrastructure that simulates running a simple messaging system using a containerized HTTP service.

Note: You do not need to write custom application code. Use the following public Docker image to simulate a service:

- HTTP service: `hashicorp/http-echo`

This will serve as your workload for ECS. You may also note alternatives in your documentation if you'd use something different in production.

Time estimate

This assignment is designed to be completed in 3 - 4 hours. If you find yourself spending significantly more time, feel free to note areas where you would spend more effort in a real-world scenario. The most important parts of this assignment are your use of AWS and ECS to provision, deploy, and manage infrastructure. Focus on the infrastructure, automation, and documentation.

Requirements

1. Infrastructure as code

Provision the following using Terraform (or CloudFormation):

- VPC with public/private subnets, routing tables, and a NAT gateway
- ECS (Fargate) cluster running the HTTP service
- RDS (PostgreSQL) instance in private subnets
- DynamoDB table (e.g., for session or metadata storage)
- SQS queue and SNS topic (with at least one subscription)
- IAM roles and policies following least-privilege principles
- Security groups and access control between components

2. CI/CD pipeline

Set up GitHub Actions or CircleCI config to:

- Build and push Docker images to Amazon ECR
- Validate Terraform syntax
- Deploy infrastructure (can be to a staging environment)
- Deploy ECS service
- Run a post-deploy health check script for the HTTP service

3. Automation and scripting

Write a script (`scripts/health_check.sh` or `.py`) that:

- Sends a request to the HTTP service
- Logs results and prints a warning if the service is not responding

4. Monitoring and alerting

Define via Terraform (or document) the following:

- CloudWatch log groups for ECS service
- At least two CloudWatch alarms, e.g.:
    - RDS CPU usage > 80% for 5 minutes
    - SQS queue depth > 100 messages for 10 minutes

5. Cost optimization

- List at least two actionable AWS cost-saving strategies (e.g., Spot tasks, Savings Plans) and describe the trade-offs.
- Pick one service (e.g., RDS, ECS, DynamoDB) and explain how you'd optimize it for cost.

Deliverables

A **GitHub repository** with your solution, including:

- `README.md`: Setup instructions, architecture diagram, and rationale
- `SOLUTION.md`: Architecture overview, trade-offs, security/monitoring notes, and cost strategies
- Code/config files, including:
    - Terraform files
    - CI/CD configuration
    - Health check script

Demo video (5 - 10 minutes): Record a video (Loom is recommended) explaining your solution, including your thought process, key design decisions, and trade-offs.