SRE AWS Technical Challenge

Welcome to Our Technical Challenge!

Our goal is to make the hiring process a true reflection of your skills while respecting your time. Rather than multiple technical interviews or live coding sessions, this take-home challenge lets you work at your own pace, in your own environment, and show us what you can really do.

Here's the deal:

- We expect this challenge to take a few hours. If you want to dive deeper, explore more features, or fine-tune your solution, feel free—but it's not expected. We value work-life balance, and we're not here to overwhelm you.
- This challenge is meant to give you a representation of what we work on. No trick questions, no gotchas, just a real-world problem you can dig into.

What we're looking for:

- How you think through and solve problems.
- Your technical skills, including code quality and approach.
- Communication and clarity in how you present your work, whether in code or supporting notes.

When you're done, send us a link to your GitHub repository so we can dive into your solution and learn more about how you work.

If you run into any questions or need anything clarified along the way, don't hesitate to reach out.

Douglas Francis

Cloud Services Managing Principal

Challenge Overview

Part One: Deployable Infrastructure

Create a proof-of-concept AWS environment using Terraform. The environment will host a basic web server with proper network segmentation and security controls.

You must use Terraform modules (your own or any open source). Coalfire's modules are encouraged but optional. (https://github.com/orgs/Coalfire-CF/repositories?q=visibility:public+terraform-aws)

Your challenge when submitted should be all within a public GitHub repository, with your code, a diagram that depicts your solution and any notes you have from going through the challenge. The main repo README should cover your solution, provide deployment steps and your notes and commentary from going through the challenge.

Any detail not provided in the scenario or requirements is up to your discretion.

Technical requirements

Network

- 1 VPC – 10.1.0.0/16
- 3 subnets, spread evenly across two availability zones.
    - Application, Management, Backend. All /24
    - Management should be accessible from the internet
    - All other subnets should NOT be accessible from internet

Compute

- ec2 in an ASG running Linux (your choice) in the application subnet
    - SG allows SSH from management ec2, allows web traffic from the Application Load Balancer. No external traffic
    - Script the installation of Apache
    - 2 minimum, 6 maximum hosts
    - t2.micro sized
- 1 ec2 running Linux (your choice) in the Management subnet
    - SG allows SSH from a single specific IP or network space only
    - Can SSH from this instance to the ASG
    - t2.micro sized

Supporting Infrastructure

- One ALB that sends web traffic to the ec2's in the ASG.

The goal is working, deployable code — it can be minimal but must meet the requirements.

Part Two – Operational Analysis and Improvement Plan

Once you've deployed your environment (or completed the code), assume you're the SRE responsible for operating it.

Document in your README:

- **Analysis of your own deployed infra**
    - What security gaps exist?
    - What availability issues?
    - Cost optimization opportunities?

- Operational shortcomings (e.g., no backups, no monitoring)?

- **Improvement Plan**
  - List *specific* changes you'd make to improve security, resilience, cost, maintainability.
  - Prioritize them (what would you fix first and why?).
  - Include at least 2 *implemented improvements* in code or scripts (e.g., tightening SG rules, adding CloudWatch alarms, setting bucket policies).
- **Runbook-style notes**
  - How would someone else deploy and operate your environment?
  - How would you respond to an outage for the EC2 instance?
  - How would you restore data if the S3 bucket were deleted?

Deliverables

1. Public GitHub repository containing:
   - All Terraform configurations
   - Architecture diagram
   - README including:
     - Solution overview
     - Deployment instructions
     - Design decisions and assumptions
     - References to resources used
     - Assumptions made
     - Improvement plan with priorities
     - Analysis of operational gaps
   - Evidence of successful deployment against the criteria. (e.g., screenshots, CLI output, Terraform apply logs)

Evaluation Criteria

We evaluate tech challenges based on

- Code Quality
  - Terraform best practices
  - Module usage

    - Correct resources deployed
    - Clear, maintainable code
- Operational thinking
    - Clear identification of risks
    - Realistic, prioritized improvements
    - Security controls and AWS best practices
- Architecture Design
    - Diagram clarity
    - Resource organization
- Documentation
    - Clear instructions
    - Well-documented assumptions
    - Proper reference citations
- Problem-Solving Approach
    - Solutions to challenges encountered
    - Design decisions
    - Demonstrated understanding of tradeoffs

Guidelines

- Work independently – no collaboration
- We do encourage use of web resources (Stack Overflow, Reddit, technical blogs, etc.) if used provide links as part of your documentation.
- Document any assumptions and design decisions you make.
- Be realistic about scope. Partial solutions are fine if well documented.
- Questions welcome – reach out if you need clarification