Canoe DevOps Tech Assessment for Remotely

Goal

Demonstrate familiarity with Docker containers and some software development skills(especially how a RESTful API is built in a Docker image and deployed to ECS Fargate)

Tasks

1. Docker file for Hello World API service using a preferred language and framework
    a. This task portion should be used by the candidate to demonstrate the ability to create clean Docker images where build dependencies are not present in the final layer of the image and the process for the software is not running as root in the container.
    b. The image should have three API endpoints
        i. `GET /hello_world`
            1. Returns a 200 status code with `{ "message": "Hello World!" }` JSON response
        ii. `GET /current_time?name=some_name`
            1. Returns a 200 status code with `{ "timestamp": 1700000000, "message": "Hello some_name" }`
        iii. `GET /healthcheck` route
            1. Returns a 200 status code to indicate that the service is healthy
        iv. For every request the service should output a structured JSON log of the request
    c. A README.md markdown document explaining:
        i. the application
        ii. how to build a docker image
        iii. how to run the image locally
        iv. how to tag the image and push it to ECR
2. Terraform to build the following in a single AWS region. Does not have to be deployed to AWS by the candidate as to not incur charges, but we do expect a `terraform plan` to work on this Terraform code. We will be checking the syntax for correctness and demonstrated understanding how Terraform and the AWS provider works
    a. This task portion should be used by the candidate to demonstrate understanding in AWS networking, service configuration, module usage, and Terraform best practices
    b. VPC creation
        i. 3 private subnets with a NAT Gateway
        ii. Public subnet with routing configured to the private subnet
    c. ECS cluster creation
    d. ECR creation
        i. There should be a repository for the Docker image of the service from step 1.
    e. ECS service
        i. The service should run in the private subnets
        ii. The service should output logs to AWS Cloudwatch
    f. Load balancer
        i. Load balancer should be in the public subnet
        ii. Load balancer performs healthchecks against a target group
        iii. Load balancer should redirect non-HTTPs requests to HTTPs
    g. Include a diagram that shows the AWS infrastructure for the service