Take-Home / Live Coding Exam - Infrastructure

Outline:

1. Problem Statement
2. Tasks
3. Deliverables
4. Evaluation Criteria

Problem Statement:

Scenario:

You are part of an engineering team tasked with building a scalable and secure data platform on AWS to support machine learning operations (MLOps). The platform should include the following components:

- An S3 bucket for data storage.
- An EC2 instance for running ETL jobs.
- An RDS instance for storing processed data.
- A CI/CD pipeline using GitHub Actions to automate deployments.
- Monitoring and logging for all components.

Your task is to use Terraform for infrastructure provisioning and Python for scripting the automation of ETL processes.

Tasks:

1. Infrastructure Provisioning with Terraform:

a. S3 Bucket:
- Create an S3 bucket named `<your-name>-hadrian-ml-data-bucket` with versioning enabled.

b. EC2 Instance:
- Provision an EC2 instance with the latest Ubuntu AMI.
- Configure security groups to allow SSH and HTTP access.
- Use user data to install Docker and Docker Compose.

c. RDS Instance:
- Create an RDS instance using PostgreSQL.
- Configure security groups to allow access from the EC2 instance.

d. CI/CD Pipeline:
- Use GitHub Actions to create a CI/CD pipeline that deploys changes to the EC2 instance.

2. ETL Automation with Python:

a. Data Ingestion:
- Write a Python script to upload sample data to the S3 bucket.
  (CSV example data checked into git and deployed as seed data in your python program is fine is fine)

b. ETL Job:
- Write a Python script to run on the EC2 instance that:
  - Downloads data from the S3 bucket.
  - Processes the data (e.g., simple transformation, i.e. change one word or a regex).
  - Uploads the processed data to the RDS instance.

3. Monitoring and Logging:

a. CloudWatch:
- Configure CloudWatch to monitor EC2 instance metrics.
- Set up log streaming for the EC2 instance to CloudWatch Logs.

4. Documentation:

- Provide a README file explaining:
  - Infrastructure setup.
  - How to run the ETL scripts.
  - How to trigger the CI/CD pipeline.
  - Monitoring and logging configuration.

—

Deliverables:

1. Terraform Scripts:
   - All `.tf` files required to set up the infrastructure.

2. Python Scripts:
   - Python scripts for data ingestion and ETL job.

3. CI/CD Configuration:
   - GitHub Actions workflow files for the CI/CD pipeline setup.

4. Documentation:
   - README file with detailed instructions.

5. Additional Configurations:
   - Any additional configuration files (e.g., Dockerfiles, CloudWatch config).

—

Evaluation Criteria:

1. Completeness:
   - All required components are provisioned and configured correctly.
   - Python scripts perform the expected tasks.

2. Code Quality:
   - Code is well-structured, commented, and follows best practices.

3. Documentation:
   - README is clear, concise, and provides all necessary information.

4. Scalability and Security:

   - Infrastructure is designed to be scalable and secure.
   - Security best practices are followed (e.g., least privilege, encrypted storage).

5. Automation:
   - CI/CD pipeline is correctly configured and automates deployments.

6. Monitoring and Logging:
   - Proper monitoring and logging configurations are in place.

Submission:

- Create a GitHub repository with all the deliverables.
- Provide a link to the repository along with any additional instructions if necessary.