Encord DevOps Challenge

DOCKER APP

You have just joined an AI company, and on your first day, you are assigned the task of deploying a Node.js application inside a Docker container. The application requires a MySQL database to manage user data, so you'll need to run a separate MySQL container alongside it.

Here are the key requirements for the task:

- Node.js Version: 18
- MySQL Version: 8

Additionally, when running the Node.js container, you'll need to pass some environment variables to Docker, as the application relies on reading those values from the environment.

List of environment variables to pass to nodeJs app docker container:
- **DB_HOST**=<MYSQL_DB_HOSTNAME>
- **DB_USER**=<MYSQL_DB_NAME>
- **DB_PASSWORD**=<MYSQL_DB_PASSWORD>
- **DB_NAME**=<MYSQL_DB_NAME>

Preload the MySQL container with data using the init.sql file provided in the folder.

You have options to decide how you would like to achieve this task.

What's expected of the above task:

1. **Functionality:** Does the setup work as intended?
2. **Containerisation Best Practices:** Are the Dockerfiles and configurations optimised?
3. **Documentation:** Is the setup easy to understand and deploy?

We should be able to access the app on port 80.

Check working status:
On the host which is running docker we should be able to use curl and get status:
- Return APP status: **curl http://localhost/api/status**
- Return User list: **curl http://localhost/api/users**

Architecture Design Challenge

You have been tasked with designing the architecture for a simple web application. The app consists of the following components:

- **Frontend:** A React.js application that users interact with.
- **Backend:** A Python FastAPI that handles business logic.
- **Database:** A PostgreSQL database for storing user data.

The goal is to design a scalable, secure, and reliable architecture that can be deployed in a cloud environment. The architecture should consider high availability, security, monitoring, and CI/CD.

Requirements:

1. **Application Components:**
    - The frontend, backend, and database should be hosted in separate environments.
    - The solution should be scalable to handle increased load.
2. **Infrastructure Design:**
    - Design the deployment using either containerised services (e.g., Docker) or cloud-native services (e.g., AWS, GCP, Azure).
    - Propose a solution for managing the infrastructure, such as using Infrastructure as Code (IaC) tools (e.g., Terraform, CloudFormation, or Ansible).
3. **CI/CD Pipeline:**
    - Design a CI/CD pipeline to automatically test, build, and deploy the application.
    - The pipeline should handle different environments (e.g., development, staging, production).
4. **Security:**
    - Propose how to secure sensitive data (e.g., environment variables, database credentials).
    - Consider network security, such as setting up firewalls, VPCs, and private subnets.
5. **Monitoring and Logging:**
    - Include a strategy for monitoring the application and infrastructure (e.g., Prometheus, Grafana, CloudWatch, Google Cloud Dashboards).
    - Design a centralised logging solution to collect and analyse logs from all services (e.g., ELK Stack, Fluentd, Google Cloud Logging).
6. **High Availability and Fault Tolerance:**
    - Ensure the architecture is highly available and can recover from failures.
    - Propose a strategy for load balancing and failover.

Deliverables:

1. **Architecture Diagram:**
   - Create a visual representation of the architecture, including all components and how they interact. Tools like Lucidchart, draw.io, or any diagramming tool are acceptable.
2. **Design Document:**
   - Write a brief explanation of your design choices, covering:
     - Infrastructure setup (e.g., cloud provider, on-premise, hybrid).
     - CI/CD pipeline design.
     - Security measures.
     - Monitoring and logging approach.
     - High availability strategy.

Scripting Challenge (use Python or Bash)

Write a script that monitors a log file for specific keywords (e.g., "ERROR" or "FAIL") and sends an alert (prints a message) if the keyword is found within the last 10 minutes.

Detailed Requirements:

1. **Log File Monitoring:**
   - The script should continuously monitor a log file (sample logs: **systems.logs**) for occurrences of the keywords "ERROR" or "FAIL" in real-time.
2. **Time Filtering:**
   - The script should only consider logs from the last 10 minutes.
3. **Alert System:**
   - If the script finds any log entries with the keywords "ERROR" or "FAIL" in the last 10 minutes, it should:
     - Print an alert message (e.g., "ALERT: Error detected in the logs!").

The estimated completion time for this task is between 2~4 hours.

Please reach out to us, if you have any confusion or questions.