Technical Assignment: CI/CD Pipeline API

Objective:

Develop a minimal RESTful API that allows users to set up and manage a simple CI/CD pipeline configuration. The focus should be on showcasing your ability to design and implement a clean, efficient API, and to handle basic infrastructure interactions. Please make sure to make a list about the assumptions that you make while designing the solution and any clarifying questions you would make.

Task Overview:

- Scenario

  You are creating a RESTful API for a CI/CD system that is consumed by users of an Internal Developer Platform. The pipeline lets them define steps that target a specific GitHub repository and run a command. Each step can be one of three classes of commands:
  - Run: Run an arbitrary command (for linting, testing, etc).
  - Build: Build a container image from a given Dockerfile in the repository, and upload it to an Amazon ECR repository.
  - Deploy: Apply a given Kubernetes manifest in the repository to a Kubernetes cluster provided by the user.
- API Development:
  - Create a RESTful API with the following endpoints:
  - POST /pipelines:
    - Create a new CI/CD pipeline configuration.
    - Accepts JSON input detailing the pipeline stages (e.g., lint, test, build, deploy).
  - GET /pipelines/{id}:
    - Retrieve the configuration of an existing pipeline by ID.
  - PUT /pipelines/{id}:
    - Update an existing pipeline configuration.
  - DELETE /pipelines/{id}:
    - Delete a pipeline configuration.
  - POST /pipelines/{id}/trigger:

        - Trigger the execution of a pipeline.
- Pipeline Execution:
    - Simulate the execution of the pipeline stages by printing logs or status updates for each stage.
    - Handle basic error scenarios (e.g., invalid configurations, pipeline failures).
- Data Persistence:
    - Use an in-memory data structure (e.g., a simple dictionary or list) to store pipeline configurations during the session.
- Documentation:
    - Include a README file with instructions on how to run the service, example API requests, and any assumptions or limitations.
- Technical Requirements:
    - The API should be implemented in one of the following languages: **Go, Python, Nodejs or Rust.**
    - The code should be clean, well-structured, and include basic error handling.
    - The API should be able to run locally with minimal setup.

Evaluation Criteria:

- API Design and Implementation:
    - Ability to create a well-structured, RESTful API.
    - Effective use of the chosen language's features and libraries.
- Code Quality:
    - Clean, readable, and maintainable code.
    - Appropriate use of comments and documentation.
- Problem-Solving:
    - Handling of edge cases and error conditions.
- Testing:
    - Inclusion of basic unit tests or example calls to demonstrate functionality.
- Bonus (Optional):
    - Implement a simple CLI that interacts with the API.
    - Add basic authentication to the API endpoints.

Time Expectation:

2-3 hours to complete the assignment.
This version of the assignment is designed to be achievable within a couple of hours, while still allowing candidates to demonstrate their backend development skills, particularly in API design, error handling, and basic infrastructure interaction. It's focused enough to be doable in a short time frame, yet comprehensive enough to assess the key skills required for the role.

Submission:

After completing the assignment, upload the files to a Google Shared Drive. Share the link directly with the recruiter (**michal.szafraniec@wolt.com**), ensuring access for anyone with the link. This helps us share your work with the team efficiently. (Please note: Email attachments may be filtered out.)