Assignment

Distributed AI Engineer – Coding Challenge

Overview

This challenge tests your ability to **design scalable AI infrastructure components** on a single machine while simulating a distributed environment. You will build a **microservice-based pipeline** that mimics distributed processing with parallel workers, fault tolerance, and inference orchestration.

Task: Build a Simulated Distributed Inference System

Create a **multi-service AI inference system** that runs multiple worker processes (or threads) simulating distributed nodes. The coordinator should dispatch inference tasks to these workers, handle retries, and return results.

Requirements

1. Microservice Inference Architecture

- Create a **coordinator service** (REST or gRPC) to manage inference requests.
- Spawn at least **3 worker services/processes**, each loading a model (e.g., small variant of CLIP, BERT, MobileNet).
- Workers must expose an HTTP or gRPC endpoint to receive inference tasks.

2. Simulated Distribution & Fault Tolerance

- Simulate network delay, failure (e.g., random crash, timeout).
- Coordinator must:

    - Track worker status (heartbeat or health checks).
    - Retry or redirect tasks on failure.
    - Log task assignments and errors.

3. Batching & Queuing

- Implement a basic **queue** at the coordinator to batch multiple incoming requests.
- Dispatch them to available workers efficiently (round-robin or load-aware).

4. Logging and Monitoring

- Log every task with metadata: request ID, worker ID, latency, retry count.
- Provide a basic dashboard or CLI output showing active workers and load.

5. Model Usage

- Each worker runs a **small pre-trained model** (e.g., from torchvision, transformers).
- Use the model to process dummy text/image input (you choose modality).

Bonus Points

- Use **Docker Compose** to simulate services.
- Use **async I/O** (e.g., asyncio, aiohttp, or FastAPI).
- Provide a **test script** that sends bursty traffic and prints success/failure per request.
- Include a **Streamlit or CLI monitor** for live request status.

Submission Guidelines

Submit a GitHub repo with:

- README: Architecture, how to run, test, simulate failure.
- Dockerfile or setup instructions.
- Example requests and expected outputs.

Submission

Files

Upload File(s)

or drag and drop here

Notes

Write here...

Submit Assignment