**Load Balancer Documentation**

Overview

The Load Balancer is designed to dynamically route incoming requests to different API endpoints based on specified criteria. It employs various routing strategies and captures metrics to optimize performance and ensure efficient load distribution.

**Design Choices**

**Architecture**

The load balancer follows a simple middleware architecture in an Express.js environment:

* **Middleware:** Utilizes Express middleware for logging requests and routing.
* **Routing:** Implements dynamic routing based on API type (REST, GraphQL, gRPC) or randomized selection for load balancing.

**Technologies Used**

* **Node.js**: Backend environment for building the load balancer.
* **Express.js**: Framework for handling HTTP requests and middleware.
* **Axios**: HTTP client for making requests to API endpoints.
* **JavaScript (ES6+)**: A programming language for application logic.

**Routing Strategies**

* **Dynamic Routing**: Determines endpoint based on request headers (`x-api-type`), allowing customization per request.
* **Randomized Routing:** Simulates load balancing by randomly selecting an API endpoint.
* **Error Handling**: Logs errors and returns appropriate HTTP status codes for failed requests.

**Mock APIs**

* **REST API:** Simulates REST API responses with varying response times.
* **GraphQL API:** Simulates GraphQL API responses with random delays.
* **gRPC API:** Simulates gRPC API responses with randomized delays.

**Setup Instructions**

**Prerequisites**

* **Node.js:** Install Node.js and npm (Node Package Manager) from [nodejs.org](https://nodejs.org).
* **Express.js:** Ensure Express.js is installed as a project dependency (`npm install express`).
* **Axios:** Install Axios for making HTTP requests (`npm install axios`).

**Installation:**

1. **Clone the repository:**

git clone <repository-url>

cd load-balancer-project

1. **Install dependencies:**

**npm install**

**Running the Load Balancer**

1. **Start the mock APIs:**

* **REST API: node rest-api.js**
* **GraphQL API: node graphql-api.js**
* **gRPC API: node grpc-api.js**

1. **Start the Load Balancer:**

npm start

1. **The load balancer will run on http://localhost:3000**

**Environment Variables**

* PORT: Specify the load balancer's port (`PORT=3000 npm start`).

**Usage Instructions**

**Routing Requests**

* **Dynamic Routing:** Include the `x-api-type` header with values (`rest`, `graphql`, `grpc`) to route requests to specific APIs.
* **Randomized Routing**: Omit the `x-api-type` header to route requests across available APIs randomly.

**Logging and Metrics**

* **Request Logging**: Middleware logs request details including URL and duration.
* **Error Handling**: Logs errors for failed requests and returns appropriate status codes.

**1. REST API Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-api-type: rest

**2. GraphQL API Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-api-type: graphql

**3. gRPC API Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-api-type: grpc

**4. Random Routing Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + (No specific header for API type to let the load balancer choose randomly)

**5. FIFO Queue Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-queue-type: fifo

**6. Priority Queue Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-queue-type: priority

**7. Round-Robin Queue Test**

* **URL:** http://localhost:3000
* **Method:** GET
* **Headers:**
  + x-queue-type: roundRobin