Hasanalp Temiz / hasanalp.tmz@gmail.com

Go Lang Interview Case Scenario - Developer Documentation

Overview

This documentation provides a comprehensive guide for understanding and setting up the Golang applications developed to fulfill the requirements outlined in the interview case scenario. The applications consist of a job and a microservice, each serving distinct purposes and functionalities.

Contents

1.Introduction

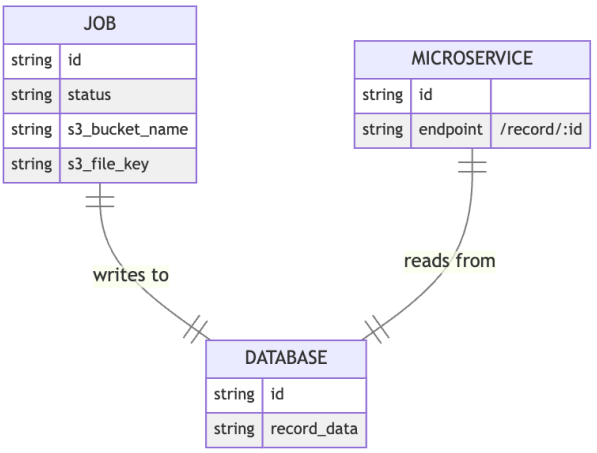
In general terms, a job service (worker) and a microservice (api) were written in the project. Although it does not fully comply with the microservice principles (one service, one db) , a proper architecture has been developed in line with the purpose of the project.

2. Architecture Overview

The architecture consists of two main components:

1. **Job (Worker):** Fetches JSON Lines files from AWS S3, processes them, and writes the data to a database.While doing this, it performs the operations in parallel for each json file entered as objectKeys in the main.go file. Also, after fetching the Json files from S3, it inserts the data into the mongoDB collection with insertMany using the bulk insert method. According to my research, insertMany is quite optimal up to 2 m data in terms of performance and power consumed. In addition, the files fetched from S3 and successfully written to the db are recorded in the log text called history.txt and the process is not repeated if the same file input is entered.

2. **Microservice (Api):** Provides a RESTful endpoint to fetch a record by ID from the database.It has been developed in a modular way and in accordance with microservice principles. The service has gained a great deal of extensibility thanks to the compound index feature added to the records collection in mongoDB and is currently very performant when retrieving records. The compound index method is capable of high performance even with much larger data counts.



3. Testing

Job (Worker) and Microservice (Api) integration tests were written for both sides. Tests were written for S3 and BulkInsert operations in Worker and GetRecord endpoint in Microservice.Therefore, the project can be tested using the "go test" command with the _test.go pages of the relevant classes with the dependencies provided without the need to run the project locally.

4.API Specification

The microservice provides a RESTful endpoint:

- Endpoint: /api/v1/record/{id}
- Method: GET
- Description: Retrieves a record by ID from the database.

5.Conclusion

As a result, the requirements of the project have been almost completely fulfilled in the development. Worker and Api microservices were written to be very clean, readable, modular and high performance. Although I have some additional ideas in the development mind, I'm happy with the end result.

Kind Regards,
Hasanalp.