Table of Contents

List of Code Snippets

List of Tables

Solutions

1 Key Value Store Orchestration

1.1 Question 1

What is the license of the original example code from the textbook? (Hint: read the 'LICENSE' file)

Upon further inspection of the LICENSE file within the project repository, I discovered that the textbook's original sample code is licensed under the Apache License, Version 2.0. Users are free to use, modify, and distribute the program under the terms of this very liberal license. It's noteworthy to note that retaining correct credit depends on the license's need to preserve copyright notices and disclaimers.

1.2 Question 2

Open-source software may use other licenses like GNU General Public License (GNU GPL) and GNU Affero General Public License (GNU AGPL). What are the difference between GNU GPL and GNU AGPL for cloud services?

Components	GNU GPL	GNU AGPL
Network Use	No explicit guidelines for using the network	In Section 13, which talks about using a network to obtain software.
Disclosure of Source Code	Only necessary when software is distributed	Needed even in cases when software is only accessed virtually (cloud services, for example)
Impact of Cloud Services	Can be utilized, if not disseminated, without revealing changes	Requires disclosure of changes, even when they are solely utilized on servers.
Release of Source Code Trigger	Software distribution	Software distribution or network-based access
Cloud providers' flexibility	More adaptable since changes may be kept confidential	More stringent, demanding that changes be disclosed
Integration	Compatible with AGPL	GPL-compatible but more restrictive
Range of Use	focuses on the dissemination of software	Encompasses network-based access as well as distribution.

Table 1: Comparative differences between GNU GPL and GNU AGPL.

1.3 Question 3

The kvs service depends on the postgres service as it need to load the transaction log from the database. The dependency is defined in docker-compose.yml so that postgres starts before kvs. However, the log messages above indicate that kvs still need to retry connecting to postgres. Why?

It was discovered that there are a few reasons why kvs needs to try connecting to Postgres again even when the requirement is specified in docker-compose.yml:

1. Although postgres begins before kvs thanks to docker-compose, postgres may not be completely functional and open for connections when kvs starts. It might take longer for the postgres service to initialize, build the required tables, or carry out other starting tasks.
2. Although the containers themselves have started, it is also possible that the network connecting them isn't instantly accessible or fully established.
3. In distributed systems, including retry logic for service connections is truly recommended practice. By doing this, the system's overall resilience and dependability are increased, making it more capable of handling brief service interruptions or network problems.
4. Docker Compose's 'depends_on' feature does not wait for the services within containers to be completely ready to accept connections; instead, it just makes sure that containers start in the correct sequence.
5. By adding a retry mechanism, kvs can guarantee that it connects only when Postgres is prepared to receive connections, strengthening and dependable the system.

2 Manageability in Environment Variables

2.1 Question 4

Explain how 'service.go' is modified for Section III and the modified part of the 'service.go' file.

```go
func initializeTransactionLog() error {
    host := os.Getenv("POSTGRES_HOST")
    dbName := os.Getenv("POSTGRES_DB")
    user := os.Getenv("POSTGRES_USER")
    password := os.Getenv("POSTGRES_PASSWORD")

    connStr := fmt.Sprintf("host=%s dbname=%s user=%s password=%s sslmode=disable",
        host, dbName, user, password)

    // code block remains same
}
```

Code 1: Changes to service.go

The 'service.go' file was altered as shown in Code 1 to replace the hardcoded values for database connection settings with environment variables. In particular, the 'initializeTransactionLog' method was modified to obtain data for the host, dbName, user, and password using os.Getenv().

2.2 Question 5

The modified 'docker-compose.yml' file for Section III.

The following code block shows the modified version of the docker-compose.yml file.

```yaml
services:
  kvs:
    build: .
    ports:
      - "8080:8080"
    environment:
      - POSTGRES_HOST=postgres
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    depends_on:
      - postgres
```

Code 2: Modified docker-compose.yml file.

3 Fault Tolerance Testing

3.1 Question 6

Is there any data loss if one service is down? Why?

Since the data is kept in a postgres database, there shouldn't be any loss if the kvs service goes down. There may be a brief loss of data for any activities performed during the postgres service outage. This is a result of kvs' inability to write to an inaccessible database.

3.2 Question 7

If there is, what would you like to do?

To lessen the possibility of data loss during a Postgres outage, you could:

1. A KVS queuing mechanism to hold requests in case the database isn't accessible.

2. Retry systems for unsuccessful database queries.
3. A caching layer in kvs to provide up-to-date information even in the event of a database outage.
4. To provide high availability for PostgreSQL, use database replication or clustering.