BÀI TẬP MODULE BACKEND

Người thực hiện: Nguyễn Đại Nghĩa (nghiand)

Yêu cầu:

- Triển khai các services lên container của docker
- Viết các API gRPC mô phorng các chức năng:
    - Đăng ký user mới
    - Kiểm tra thông tin KYC
    - Tạo tài khoản tiết kiệm: với các thông tin số dư, kỳ hạn, lãi suất, lãi ước tính,...
    - Truy vấn thông tin tài khoản tiết kieemj
    - Rút tiền từ tài khoản tiêết kiệm: rút đúng ngày đáo hạn, ruút sớm
    - Truy vấn danh sách tài khoản tiết kiệm của một khách hàng
    - Truy vấn danh sách tài khoản tiết kiệm dựa trên các fields:
        - KYC
        - Số dư tối thiểu
        - Ngày đáo hạn (khoảng thời gian)
        - Kỳ hạn
    - Phân trang danh sách kết quả trả về, cung cấp các thông tin về số lượng và tổng số dư của các tài khoản thỏa điều kiện
- Produce message mỗi khi có tài khoản mới được tạo, để một hệ thống bên ngoài có thể nhận

Thực hiện

Các công nghệ / thư viện đã tìm hiểu và sử dụng

gRPC

- Định nghĩa message và service để giao tiếp giữa các service
- Các API gRPC cơ bản:
    - Đăng ký mới user
    - Kiểm tra KYC level
    - Mở tài khoản tiết kiệm
    - Truy vấn thông tin tài khoản
    - Lấy danh sách tài khoản đáp ứng các yêu cầu
    - Rút tiền, cập nhật số dư tài khoản
    - Truy vấn thông tin user

- Tìm kiếm user hoặc danh sách user theo yêu cầu

```
❯ docker exec -it banksystem-saving-client-1 /bin/bash
root@e11d34a0e758:/app# ./saving-client
address mid container: mid-savingInput action:
1. Register User
2. Check KYC level
3. Open SavingAccount
4. Account Inquiry
5. Withdrawal
6. Search accounts by Filters ***NEW***

7. Search all saving accounts by UserID
8. Search User by ID card number
9. Search User by Account ID
10. Search Users by Filters
```

(demo console client)

Go tag injection

- Đánh thêm các custom tag cho các fields của message trong proto.

```
message SavingAccount {
  // @gotags: es:"id"
  string id = 1;

  // @gotags: es:"user_id"
  string user_id = 2 [json_name = "user_id"];

  // @gotags: es:"balance"
  int64 balance = 3;

  string term_type = 4 [json_name = "term_type"];

  int32 term = 5;

  // @gotags: es:"term_in_days"
  int32 term_in_days = 6 [json_name = "term_in_days"];
```

- Sau khi generate các file .pb.go, sử dụng go tag injection để lấy được các fields cần thiết, từ đó lấy được các thông tin phục vụ cho việc indexing document trong elasticseach

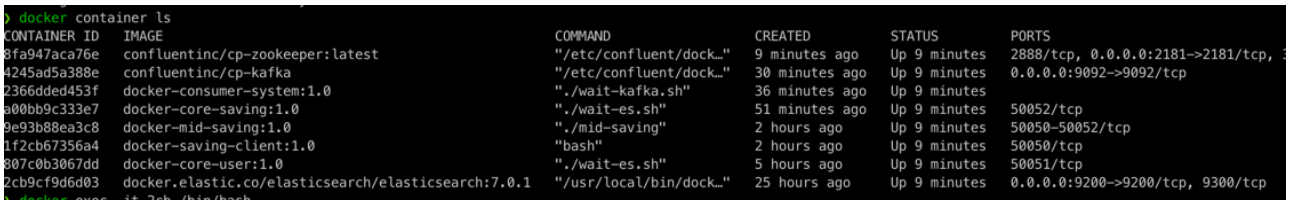
Reflection

- Tìm hiểu các kỹ thuật reflection cơ bản trong go
- Sử dụng trong đọc các object: ví dụ đọc các tag quy ước trong proto để lấy ra các fields phù hợp để index vào elasticsearch

Docker

- Tìm hiểu các khái niệm cơ bản: image, container, dockerhub, pulling, tạo custom image. Khởi chạy, dừng, xóa các container và image.
- Chèn lệnh bash vào container bằng docker exec
- Viết dockerfile cho từng service, đóng gói các services thành các images và triển khai thành các container
- Tạo network riêng và config để các container giao tiếp với nhau : sử dụng biến môi trường và tên service trong các địa chỉ

- Viết docker compose: liệt kê các services và cấu hình để các container khởi chạy đúng kịch bản và giao tiếp được với nhau
- Một số services khi chạy cần phụ thuộc vào service khác đã sẵn sàng, ví dụ các core services cần lấy được các elasticsearch.Client. Do đó sử dụng ENTRYPOINT để chạy file bash để tạo lệnh chờ thích hợp



```
> docker container ls
CONTAINER ID   IMAGE                                          COMMAND                  CREATED          STATUS          PORTS
8fa947aca76e   confluentinc/cp-zookeeper:latest               "/etc/confluent/dock…"   9 minutes ago    Up 9 minutes    2888/tcp, 0.0.0.0:2181->2181/tcp, 3
4245ad5a388e   confluentinc/cp-kafka                          "/etc/confluent/dock…"   30 minutes ago   Up 9 minutes    0.0.0.0:9092->9092/tcp
2366dded453f   docker-consumer-system:1.0                     "./wait-kafka.sh"        36 minutes ago   Up 9 minutes
a00bb9c333e7   docker-core-saving:1.0                         "./wait-es.sh"           51 minutes ago   Up 9 minutes    50052/tcp
9e93b88ea3c8   docker-mid-saving:1.0                          "./mid-saving"           2 hours ago      Up 9 minutes    50050-50052/tcp
1f2cb67356a4   docker-saving-client:1.0                       "bash"                   2 hours ago      Up 9 minutes    50050/tcp
807c0b3067dd   docker-core-user:1.0                           "./wait-es.sh"           5 hours ago      Up 9 minutes    50051/tcp
2cb9cf9d6d03   docker.elastic.co/elasticsearch/elasticsearch:7.0.1   "/usr/local/bin/dock…"   25 hours ago     Up 9 minutes    0.0.0.0:9200->9200/tcp, 9300/tcp
> docker exec -it 2cb /bin/bash
```

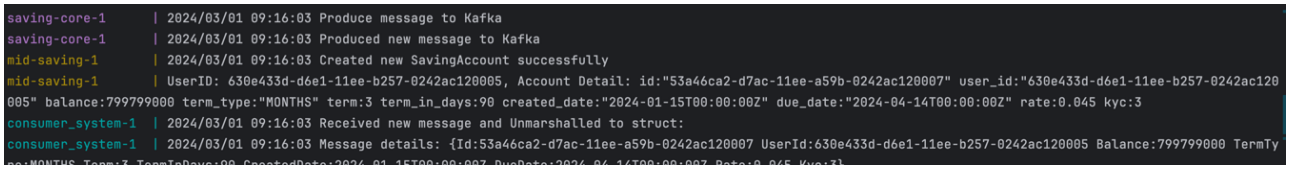
Search Engine: Elastic Search

- Sử dụng image có sẵn trên dockerhub để chạy container
- Mapping đường dẫn dữ liệu với thư mục cố định trên máy đang chạy
- 2 index sử dụng trong bài tập: saving, user
- Tạo indexRequest và Do để thực hiện ghi mới một document vào index
- Tạo updateRequest và Do để thực hiện cập nhật giá trị của một document cho trước
- Tạo các query đơn giản: match, term cho các field quan tâm
- Viết thư viện để sinh query cho các truy vấn liên quan tới nhiều fields: mỗi điều kiện là một map[string]interface{}, tùy thuộc vào phép so sánh và kiểu dữ liệu của field cần lọc sẽ viết hàm sinh query tương ứng, sau đó đưa tất cả query con vào 1 mảng để tạo ra câu query cuối cùng. Khi yêu cầu truy vấn về sau mở rộng có thể tái sử dụng mà không cần viết thêm query riêng biệt (đang tiếp tục mở rộng)
- Tìm kiếm và lọc kết quả dựa trên ràng buộc của các fields: kyc, term, min_balance, duedate_early, duedate_late (còn tiếp tục mở rộng). Giá trị nhập vào theo yêu cầu lọc, hoặc -1 để bỏ qua điều kiện lọc của field này. Ví dụ: kyc = -1, term = -1, min_balance = 5000000, duedate_early = 01022024, duedate_late = 30042024 để tìm kiếm tất cả tài khoản tiết kiệm tối thiểu 5 triệu, có ngày đáo hạn từ 01/02/2024 đến 30/04/2024
- Thực hiện phân trang cho kết quả tìm kiếm: sử dụng from, size. Tương ứng trong message là page_index và page_size. Ở lần truy vấn đầu tiên sẽ lấy về tổng số kết quả "hits", giá trị này trả về response và phát sinh số trang kết quả phù hợp. Ở client có thể tiếp tục nhập index của trang khác để

tiếp tục xem các kết quả khác

```
Input DueDateRange - earliest date
-1
Input DueDateRange - latest date
-1
Input minimum balance
1
Page Size
6
2024/03/01 09:18:31 Calling SearchAccountsByFilters
Hits: 15
Total balance: 1609757700
Page 1:
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
| ID                                   | UserID                               |     Balance | KYC | Term in Days | Due Date                 |
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
| 3761f0e8-d6db-11ee-be25-0242ac120004 | 77100a2a-d6d4-11ee-b3e2-0242ac120005 |  152678000 | 2   | 180           | 2024-09-04T00:00:00Z     |
| ddffcc57-d6da-11ee-be25-0242ac120004 | 77100a2a-d6d4-11ee-b3e2-0242ac120005 |   10000000 | 2   | 21            | 2024-03-22T00:00:00Z     |
| 3bd4770c-d773-11ee-86df-0242ac120004 | 20ecd4a7-d773-11ee-96b0-0242ac120005 |    6500000 | 2   | 90            | 2024-02-29T00:00:00Z     |
| 21d0f43c-d6db-11ee-be25-0242ac120004 | 66f85ba1-d6d2-11ee-bdb2-0242ac120005 |   17200000 | 2   | 21            | 2024-02-10T00:00:00Z     |
| e16bd1c6-d6ea-11ee-ab88-0242ac120004 | c1545a8a-d6ea-11ee-afe8-0242ac120005 |     123000 | 2   | 90            | 2024-09-13T00:00:00Z     |
| 4822594f-d773-11ee-86df-0242ac120004 | 20ecd4a7-d773-11ee-96b0-0242ac120005 |   38000000 | 2   | 90            | 2024-04-24T00:00:00Z     |
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
See another page result?: (-1 for exit)
2
2024/03/01 09:18:33 Hits: 6
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
| ID                                   | UserID                               |     Balance | KYC | Term in Days | Due Date                 |
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
| 0e67e215-d6db-11ee-be25-0242ac120004 | 66f85ba1-d6d2-11ee-bdb2-0242ac120005 |           1 | 2   | 180           | 2024-07-13T00:00:00Z     |
| ee7b265e-d6da-11ee-be25-0242ac120004 | 77100a2a-d6d4-11ee-b3e2-0242ac120005 |   75000000 | 2   | 90            | 2024-05-30T00:00:00Z     |
| 5d4e5863-d773-11ee-86df-0242ac120004 | 20ecd4a7-d773-11ee-96b0-0242ac120005 |  150000000 | 2   | 180           | 2024-06-29T00:00:00Z     |
| 69a84402-d773-11ee-86df-0242ac120004 | 20ecd4a7-d773-11ee-96b0-0242ac120005 | 1000000000 | 2   | 90            | 2024-06-13T00:00:00Z     |
| 30624a6b-d7a7-11ee-a8bc-0242ac120007 | a03001b7-d7a5-11ee-aa7a-0242ac120008 |    6800000 | 2   | 180           | 2024-08-13T00:00:00Z     |
| b3f7d3e7-d7a5-11ee-b4d1-0242ac120007 | a03001b7-d7a5-11ee-aa7a-0242ac120008 |  100000000 | 2   | 180           | 2023-08-11T00:00:00Z     |
+--------------------------------------+--------------------------------------+-------------+-----+---------------+--------------------------+
```

Message broker: Kafka

- Chạy các container từ các images có sẵn: kafka và zookeeper
- Cài đặt trong docker-compose để chạy zookeeper và kafka, expose các cổng cần thiết để các dịch vụ khác nhau có thể truy cập, produce và consume các message
- Tạo delay time bằng file .sh ở ENTRYPOINT của kafka để đảm bảo zookeeper đã khởi chạy và ready
- Trong core-saving service, mỗi khi ghi mới thành công một SavingAccount, sẽ gọi thư viện và method cần thiết để produce một messange. Topic được quy ước là "NewSavingAccountCreated". core-saving chỉ có nhiệm vụ produce message và không quan tâm service nào khác sẽ consume message này
- Viết một service đơn giản tên là consumer-system. Service này có địa chỉ của kafka, được config trong docker-compose, và lắng nghe topic "NewSavingAccountCreated". Trong service này đã được định nghĩa struct SavingAccount phù hợp để có thể unmarshal từ data là []byte thành object của struct SavingAccount
- 

Database: (đang thực hiện)

- Sử dụng TiKV để lưu các struct user, saving_account.
- Các yêu cầu truy vấn sẽ được sử dụng elasticsearch tìm kiếm, sau khi có id sẽ gọi tiếp TiKV database để lấy dữ liệu