**Spring Boot Microservices with an account-to-account transaction**

This source code was developed for Java based microservices with account-to-account transaction as a part of CBD technical assessment.

### Overview of the microservices

CBD banking API concept with spring boot based micro services architecture. Initially I will develop the core API which will evolve as a full-fledged REST API collection until deployments where Fund transfer service deals with the account-to-account transaction with payment and bank core service using Java 11, Spring boot 2.4.5, Spring Cloud, Netflix Eureka, Zipkin, Jitpack and Keycloak.

### Purpose of the microservices

- Fund transfer service (cbd-fund-transfer-service) – This is the service that handles all the fund transfers between accounts and this API will push messages to a centralized RabbitMQ queue to use from the Notification service.

- User service (cbd-user-service) – This service includes all the operations under the User such as registrations and retrieval. Additionally, this API consumes keycloak REST API to register and manage the user base while using the local PostgreSQL database as well.

- Payment service (cbd-payments-service) – This service will include all the API endpoints to process Utility payments in this project and that will push notification messages to RabbitMQ as well.

- Notification service – This API is registered under the service registry but consumes all the messages from RabbitMQ and pushes necessary notifications to the end users.

- Banking core service – This is the banking core service that acts as a dummy banking core with accounts, users, transaction details, and processors for banking transactions.

### Technology Stack

1. Java 11

2. Spring Boot 2.4.5

3. Netflix Eureka Service Registry

4. Netflix Eureka Service Client

5. Spring Cloud API Gateway

6. Spring Cloud Config Server

7. Zipkin

8. Spring Cloud Sleuth

9. Open Feign

10. RabbitMQ

11. Prometheus

12. Jitpack

13. MySQL

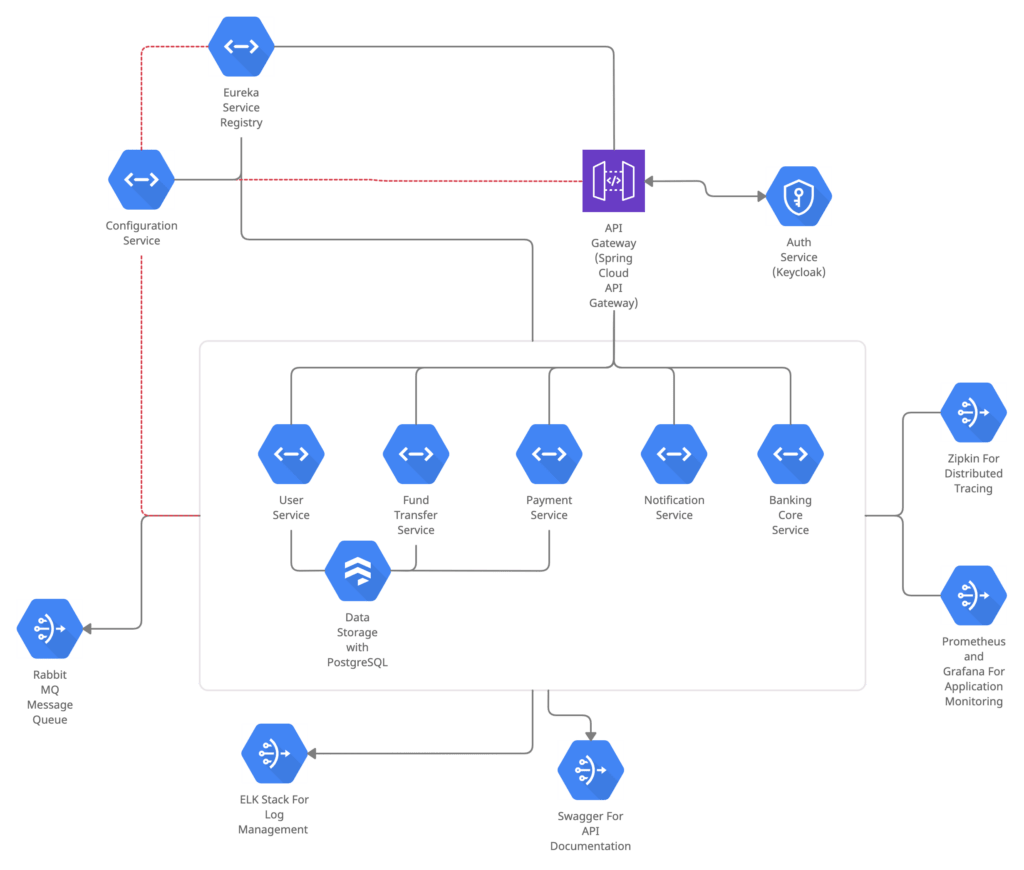
14. Keycloak

15. Docker / Docker Compose

16. Kubernetes

17. Keycloak

**Base Project Architecture**

[](https://github.com/javatodev/internet-banking-concept-microservices/blob/main/README.md)

**Main microservices**

1. Microservices – Service Registration and Discovery with Spring Cloud Netflix
2. Microservices – Setup API Gateway Using Spring Cloud Gateway
3. Microservices – Authentication, and Authorization with Keycloak
4. Microservices – CBD core Banking Service Implementation
5. Microservices – User Service Implementation
6. Microservices – Fund Transfer Service Implementation
7. Microservices – Communication with Spring Cloud Open Feign
8. Microservices – Exception Handling
9. Microservices – Centralized Configurations with Spring Cloud Config