Monolith to Distributed

Microservices Series

Matthew Oberlies @ Revature 7/22/2020

REVATURE

Overview

Application will leverage the Netflix Stack, except substituting Consul and Spring Cloud Gateway for Eureka and Zuul, respectively

Consul is a service-mesh solution to service discovery and communication

It has many helpful features that we do not have time for, such as a key-value store

Overview

A few Netflix projects have entered Maintenance mode, and the community has started to shift to other technologies

Spring Cloud Gateway is an abstraction layer for Zuul, which have been moved to maintenance mode

Eureka has not been moved to maintenance mode however Consul has been gaining in popularity recently

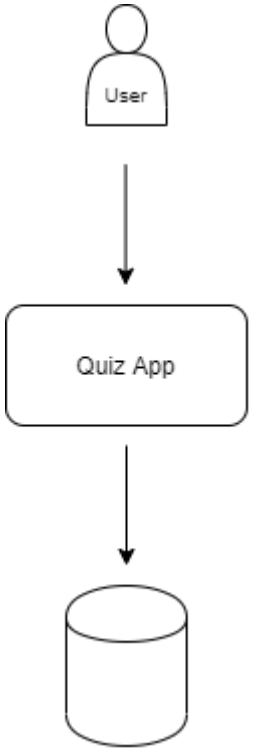
Initial Monolith

We have the backend for a quiz application that is designed to help associates quiz themselves on various topics throughout training

There is a supply of flashcards of various topics and difficulties

Quizzes are constructed as a collection of flashcards

Initial Monolith

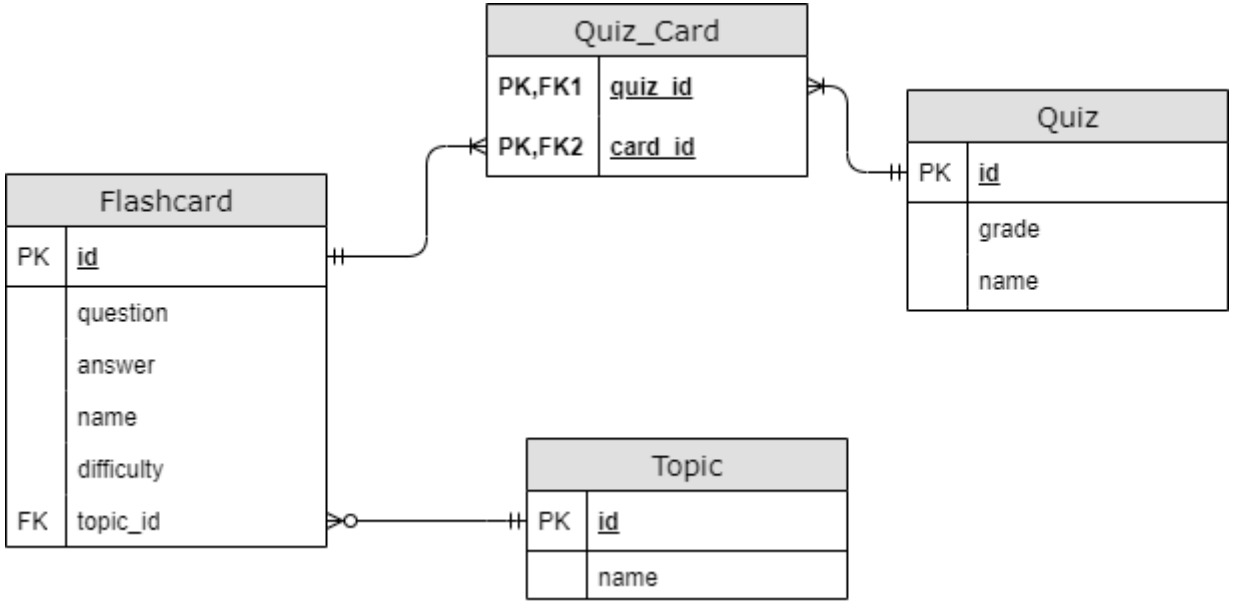


The application uses Spring Boot and Spring Data for ease of development

The Controller layer has most CRUD operations, but not all, for both quizzes and flashcards

It uses an in-memory database

Initial Monolith



REVATURE

Goal

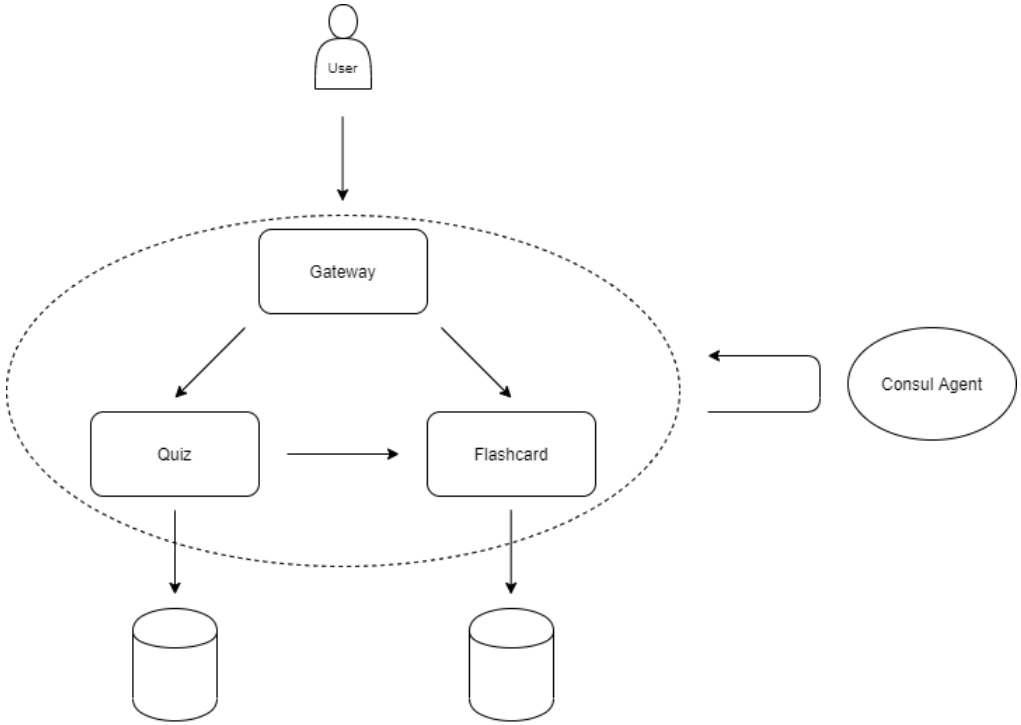
Break Services

API Gateway

Service Discovery

Live Example

New Architecture



REVATURE

How could we evolve this further?

REVATURE

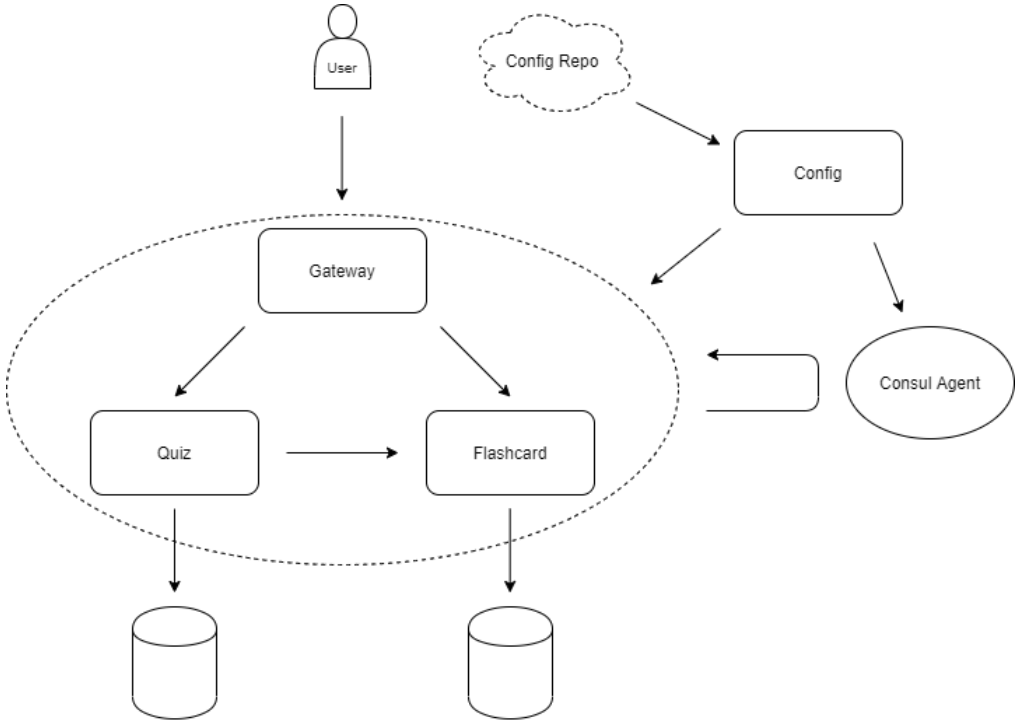
Next Steps

Centralized Configuration

Circuit Breaking

Eventual Consistency

Centralized Configuration



REVATURE

Circuit Breaking

In its current state, the quiz service may send a failing response for some operations if the flashcard service is down

Spring Cloud Circuit Breaker would allow our quiz service to fail gracefully; it won't perform the operation, but can provide a meaningful response

Hystrix is commonly used, but as it is maintenance mode, some are switching to Resilience4J

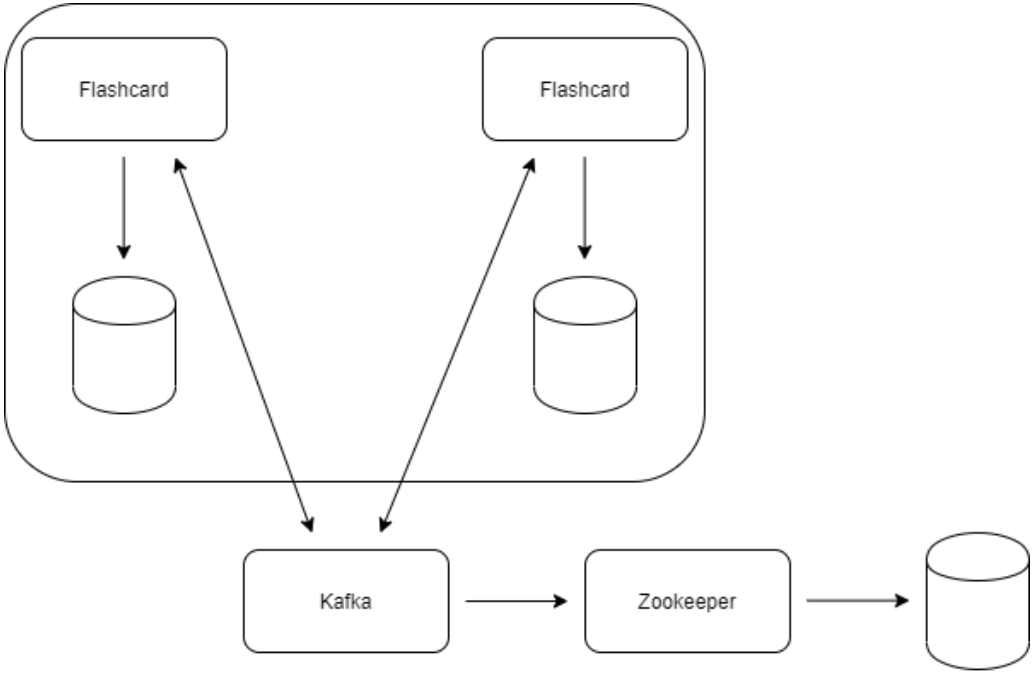
Eventual Consistency

If our flashcard service reaches a bottleneck at the database level, we might choose to replicate the database to accomplish efficient scaling for READ operations

The issue is that now our replicated data must be synchronized

We could use Apache Kafka as a Messaging Queue to broadcast messages about every change to the data, received by all instances of the flashcard service

Eventual Consistency



References

Consul Tutorial: https://learn.hashicorp.com/consul

Spring Cloud Gateway Docs: https://cloud.spring.io/spring-cloud-gateway/reference/html/

Example Repository: https://github.com/revaturelabs/microservices-series-example

Questions