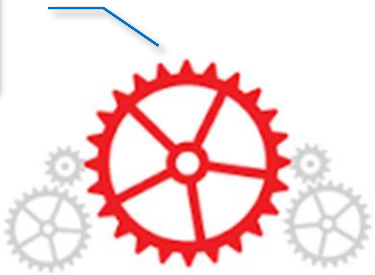
Micro services - Overview

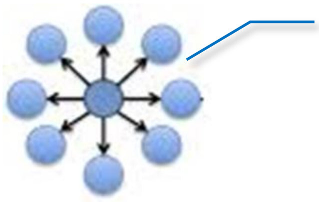
What is Micro Service?

The main objective of the micro-services implementation is to split up the application as separate service for each core and API service functionality and it should be deployed independently on cloud. We have chosen the reactive programming language from spring.io family project with a set of components that can be used to implement our operations model. Spring Cloud integrates the Netflix components in the spring environment in a very nice way using auto configuration and convention over configuration similar to how Spring Boot works.

Why Micro Services Architecture?



We chose micro services architecture to write each functionality as a separate service for core and API functionality and it helps us to achieve the continuous delivery and integration.



Patterns in Micro Services Architecture

- API Gateway
- Service registry
- Service discovery

Patterns in Micro services Architecture

API Gateway	1. Choose to build the application as a set of micro-services. 2. Decide how the application client's will interact with the micro services. 3. With a monolithic application there is just one set of (typically replicated, load-balanced) endpoints. 4. In a micro services architecture, however each micro services exposes a set of what are typically fine-grained endpoints.
Service Registry	1. Service registry helps to determine the location of service instances to send request to the corresponding service 2. Here, we have used the Netflix Eureka to register a service that are available to be registered in service registry server and it can be identified through the router.
Service Discovery	1. In a monolithic application, services invoke one another through language-level method or procedure calls. 2. But, in a modern micro services based application typically runs in a virtualized environments where the number of instances of a service and their locations changes dynamically. 3. Each service can be identified using router that are registered with service registry server.

Micro services Architecture via Netflix Components

We have used the Netflix components to accomplish the above micro services architecture patterns. T

Operations Component	Spring, Netflix OSS
Service Discovery server	Netflix Eureka
Edge Server	Netflix Zuul
Central configuration server	Spring Cloud Config Server
Dynamic Routing and Load Balancer	Netflix Ribbon
OAuth 2.0 protected API's	Spring Cloud + Spring Security OAuth2
Monitoring	Netflix Hystrix dashboard and turbine

Major Components of Netflix

Netflix Eureka - Service Discovery Server

Netflix Eureka allows micro services to register themselves at runtime as they appear in the system landscape.

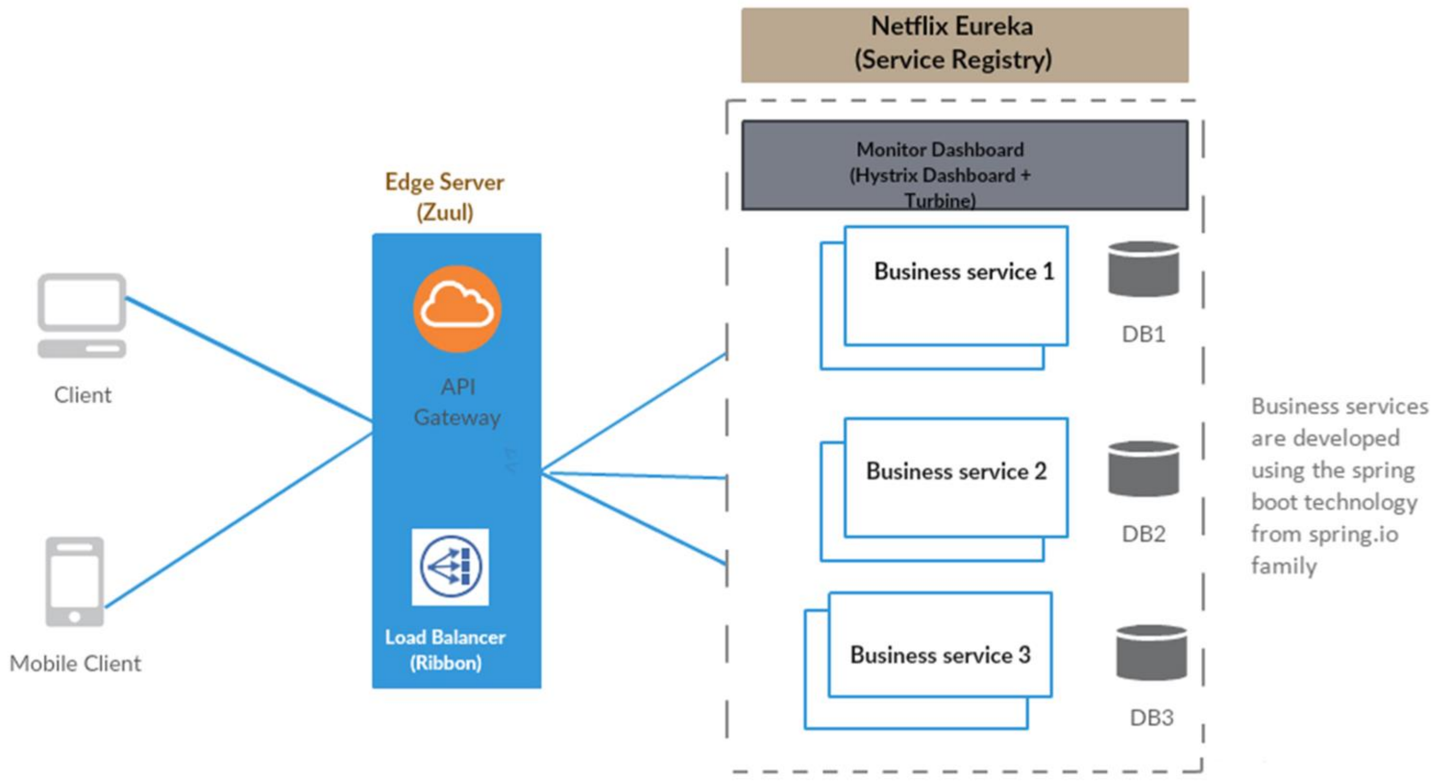
Netflix Ribbon - Dynamic Routing & Load Balancer

Netflix Ribbon can be used by service consumers to look up services at runtime. Ribbon uses the information available in Eureka to locate appropriate service instances. If more than one instance is found, Ribbon will apply load balancing to spread the requests over the available instances. Ribbon does not run as a separate service but instead as an embedded component in each service consumer.

Netflix Zuul - Edge Server

Zuul is (of course) our gatekeeper to the outside world, not allowing any unauthorized external requests pass through. Zulu also provides a well-known entry point to the micro services in the system landscape. Using dynamically allocated ports is convenient to avoid port conflicts and to minimize administration but it makes it of course harder for any given service consumer. Zuul uses Ribbon to look up available services and routes the external request to an appropriate service instance.

Spring Boot & Spring Cloud Netflix OSS - Micro Service Architecture



Micro Services with Spring Boot

Spring Boot is a brand new framework from the team at Pivotal, designed to simplify the bootstrapping and development of a new spring application. The framework takes an opinionated approach to configuration, freeing developers from the need to define boilerplate configuration.

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)
- Provide opinionated 'starter' POMs to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely **no code generation** and **no requirement for XML** configuration

Spring Cloud Netflix

Spring cloud Netflix provides Netflix OSS integrations for spring boot apps through auto configuration and binding to the spring environment and other spring programming models. With a few simple annotations we can quickly enable and configure common patterns inside application and to build large distributed systems with Netflix components. There are lot of features available with spring cloud Netflix. Here, we have listed some of the common features we have implemented with micro services with spring boot and Netflix,

Service Discovery	•Eureka instances can be registered and clients can discover the instances using spring managed beans.
Service Creation	•an embedded Eureka server can be created with declarative Java configuration
External Configuration	•a bridge from the Spring Environment (enables native configuration of Netflix components using Spring Boot conventions)
Router and Filter	•automatic registration of Zuul filters, and a simple convention over configuration approach to reverse proxy creation