**Spring Boot and Spring Cloud**

Spring Boot and Spring Cloud are both part of the Spring ecosystem, but they serve different purposes.

Spring Boot is a framework that makes it easy to create standalone, production-grade Spring-based applications. It provides a set of pre-configured dependencies and an opinionated approach to configuration, making it quick and easy to get started with Spring. Spring Boot is often used to build microservices, which are small, independently deployable services that work together to form a larger application.

Spring Cloud, on the other hand, is a set of tools and frameworks that helps developers build and deploy microservices-based applications. It provides a suite of tools for service registration, discovery, and configuration, as well as distributed tracing and fault tolerance. Spring Cloud builds on top of Spring Boot, adding additional functionality that is specifically designed for microservices-based architectures.

In summary, Spring Boot is a framework for building standalone Spring-based applications, while Spring Cloud is a set of tools and frameworks for building and deploying microservices-based applications. Spring Boot is often used as the foundation for microservices-based applications, while Spring Cloud provides additional functionality and tools that are specifically designed for microservices.

Both Spring Boot and Spring Cloud are well-suited for building microservices-based architectures, as they provide a solid foundation for building and deploying small, independent services that work together to form a larger application. By using these frameworks, developers can create scalable and flexible applications that can adapt to changing business requirements over time.

A black screen with white text

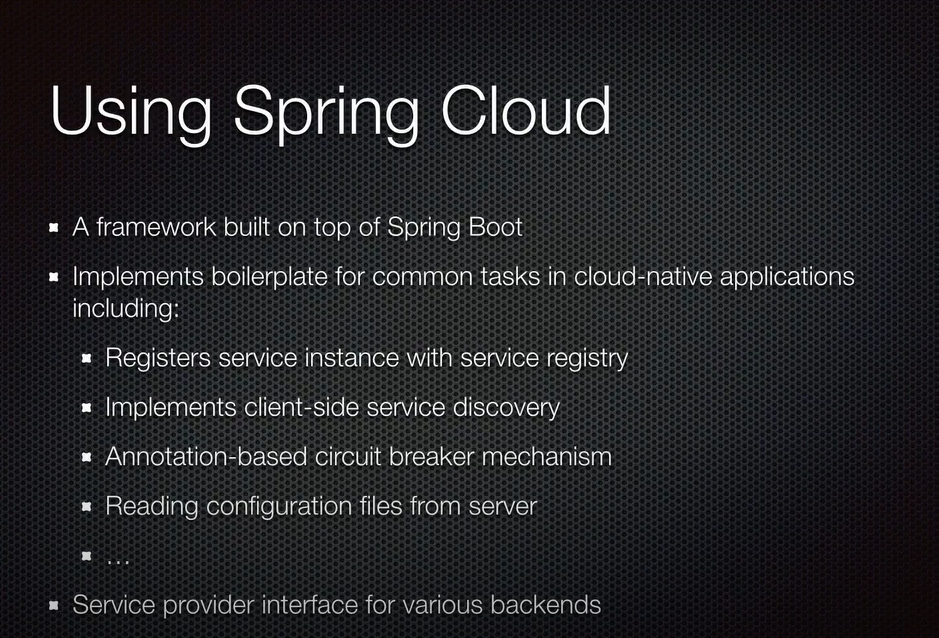
Description automatically generated

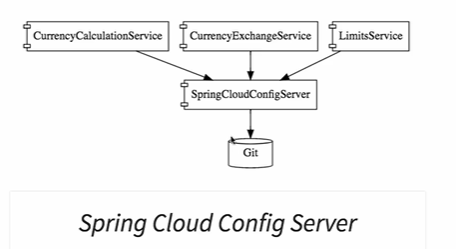
A black screen with white text

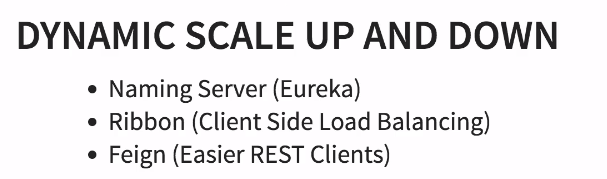
Description automatically generated

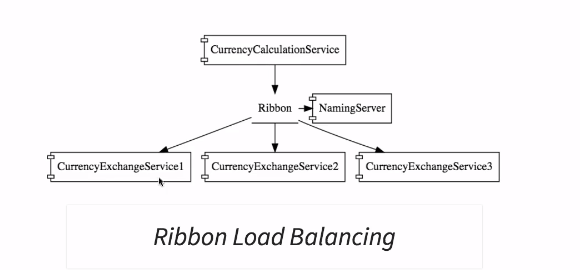
A black and white text on a black background

Description automatically generated









**Naming service**

Service registration

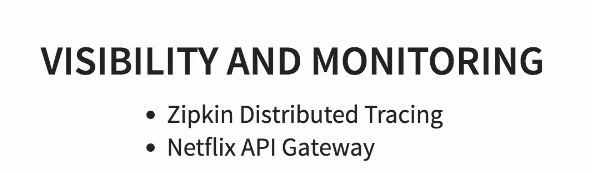
Service discovery

**Ribbon**

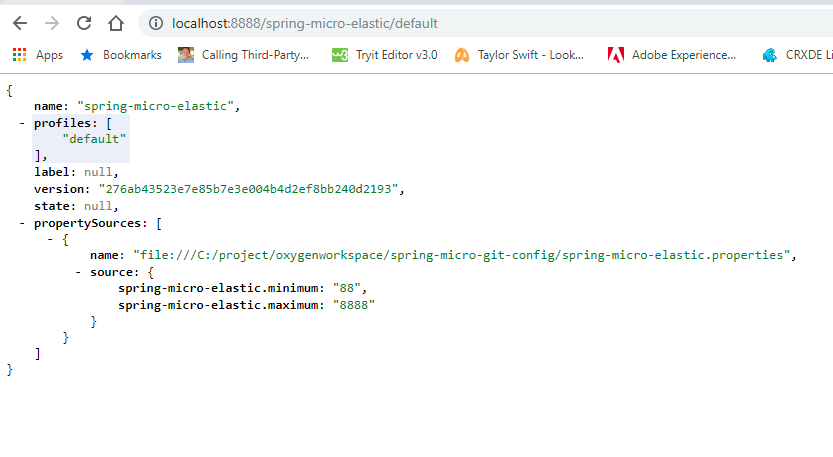
client side load balancing

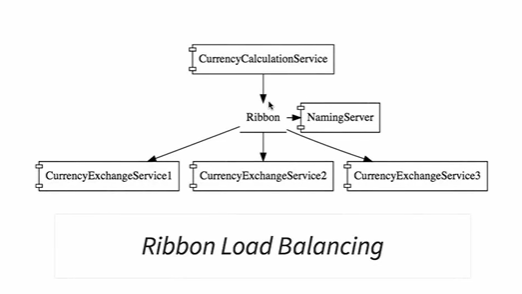
**Feign Client**

Rest client

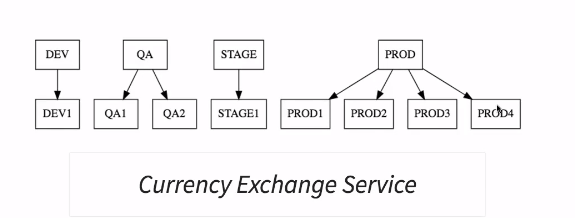


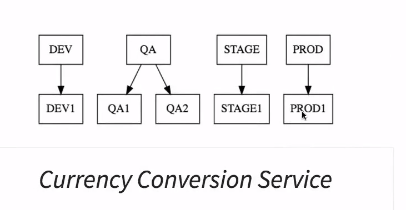








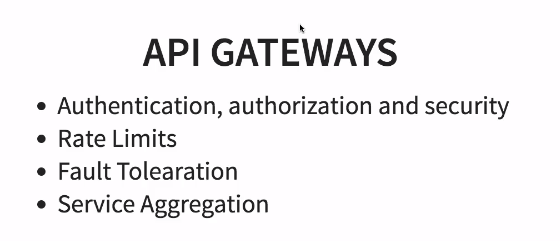




Ribbon helps in accessing services in load balanced way

Naming server is useful we need auto scale up or down. All the micro services will register themselves with the naming server (service registration) service1 contacts with naming server to find out how many instances of service2 are running currently.(this is called service discovery)

These should be done at api gateway level



**Starting Cassandra NoSQL db**

C:\project\softwares\apache-cassandra-3.11.4\bin>cassandra –f

Updates to Step 40 : Use spring-cloud-starter-zipkin and spring-rabbit

*The dependencies are ever changing with Spring Cloud and Spring Boot.*

If you are using Spring Boot Release >= 2.1.\*, you would need to use spring-cloud-starter-zipkin and spring-rabbit instead of spring-cloud-sleuth-zipkin and spring-cloud-starter-bus-amqp.

You would need to make this change in THREE pom.xmls - in currency-conversion-service, currency-exchange-service and zuul-api-gateway projects

**New Dependencies**

1. <dependency>
2. <groupId>org.springframework.cloud</groupId>
3. <artifactId>spring-cloud-starter-zipkin</artifactId>
4. </dependency>
5. <dependency>
6. <groupId>org.springframework.amqp</groupId>
7. <artifactId>spring-rabbit</artifactId>
8. </dependency>

**OLD Dependencies to be Replaced**

1. <dependency>
2. <groupId>org.springframework.cloud</groupId>
3. <artifactId>spring-cloud-sleuth-zipkin</artifactId>
4. </dependency>
6. <dependency>
7. <groupId>org.springframework.cloud</groupId>
8. <artifactId>spring-cloud-starter-bus-amqp</artifactId>
9. </dependency>