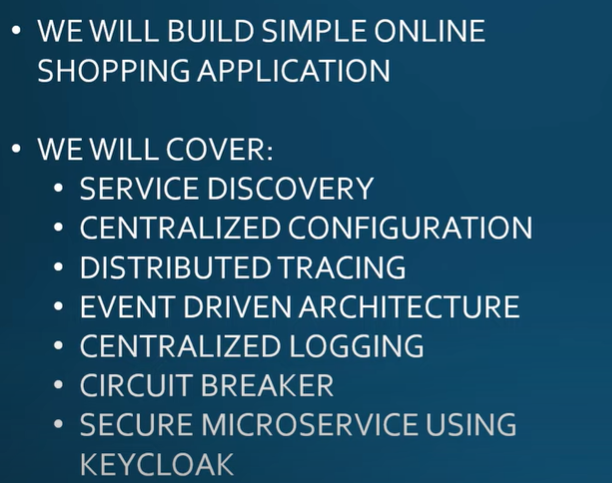
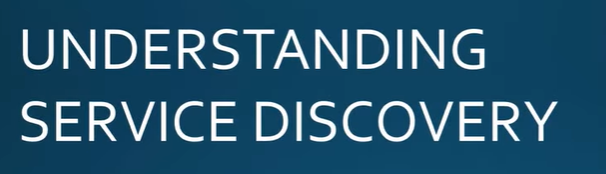
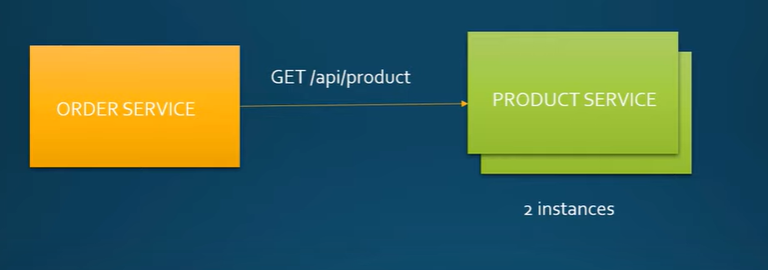
**Spring Boot Microservices Project - Part 1 - Spring Boot Config Server & Eureka Service Discovery**

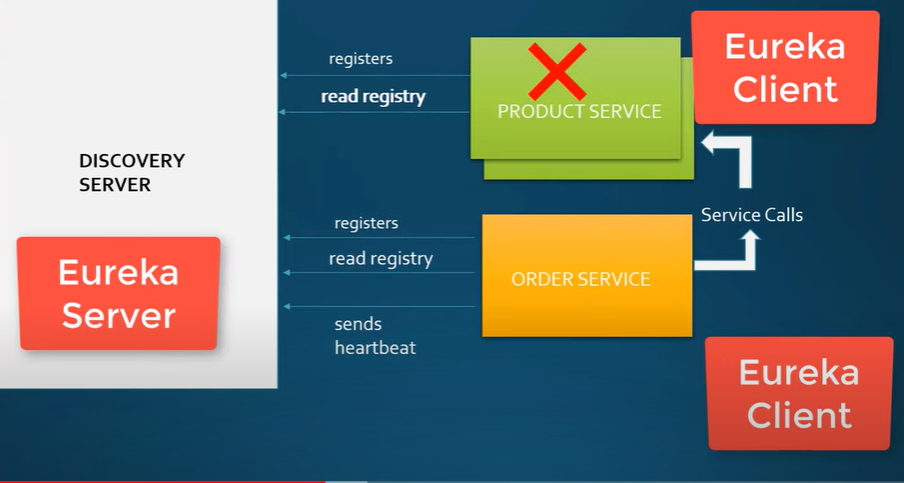


<https://programmingtechie.com/2021/03/24/spring-boot-microservices-project-tutorial-part-1/>



What is Service Discovery in Microservices? Microservices service discovery is **a way for applications and microservices to locate each other on a network**. Service discovery implementations within microservices architecture discovery includes both: a central server (or servers) that maintain a global view of addresses.



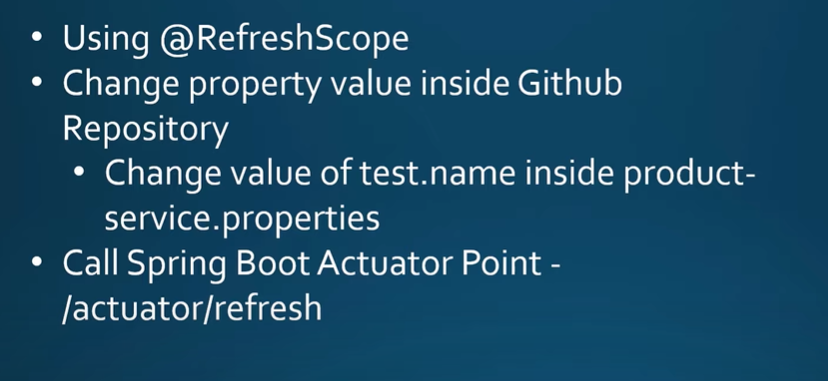


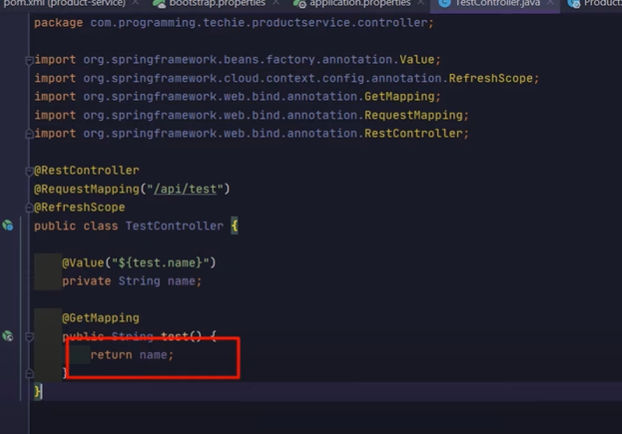
To create the discovery server, we need to just add one dependency – **Eureka Server**, once you download the project to your machine, we can enable the Eureka Server functionality by adding the **@EnableEurekaServer** annotation.

**ERROR:** **A child container failed during start**

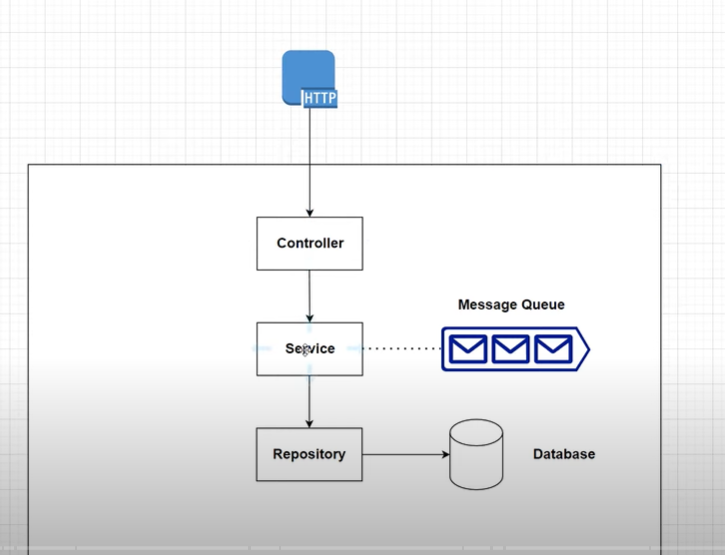
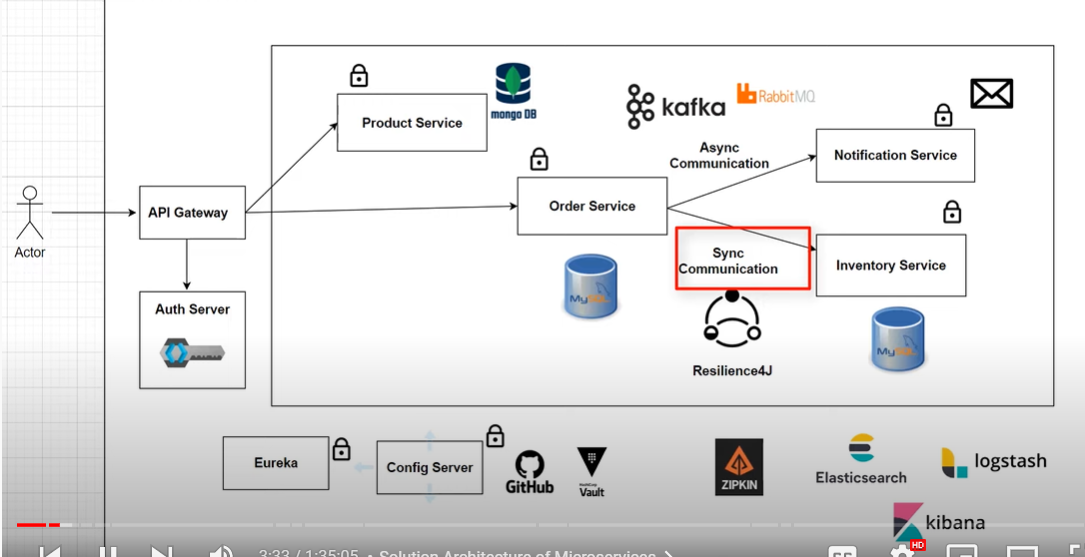
[**https://superuser.com/questions/1569942/i-run-tomcat-9-and-it-keeps-saying-grave-a-child-container-failed-during-start**](https://superuser.com/questions/1569942/i-run-tomcat-9-and-it-keeps-saying-grave-a-child-container-failed-during-start)

****

****

****

**Then going to add actuator dependency.**

****

What is @RequiredArgsConstructor in spring boot?

The @RequiredArgsConstructor annotation **generates the constructors with one parameter for each field needing special handling**. The final fields annotated with @NonNull are the parameters of the required constructor because they need special handling.

What does @SLF4J annotation do?

Annotation Type Slf4j  
  
**Causes lombok to generate a logger field**. Complete documentation is found at the project lombok features page for lombok log annotations. This annotation is valid for classes and enumerations.

**Integration Testing in Our product Service:**

For integration testing we need to install Test Container Library.

Is API testing integration testing?

API (application programming interface) testing is performed at the message layer without GUI. **It is a part of integration testing** that determines whether the APIs meet the testers' expectations of functionality, reliability, performance, and security.

What is the difference between unit testing and integration testing and end to end testing?

**In unit testing, each module of the software is tested separately.** **In integration testing, all modules of the software are tested combined**.

What is a test container?

Testcontainers is **a Java library that supports JUnit tests, providing lightweight, throwaway instances of common databases, Selenium web browsers, or anything else that can run in a Docker container**.

**WE NEED TO ADD THIS DEPENDENCY:**

<dependency>

<groupId>org.testcontainers</groupId>

<artifactId>testcontainers</artifactId>

<version>1.17.3</version>

<scope>test</scope>

</dependency>

BETTER TO USE MULTIPLE TEST CONTAINER DEPENDENCY

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.testcontainers</groupId>

<artifactId>testcontainers-bom</artifactId>

<version>1.17.3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

After adding this dependency we need to add mongoDB module in our Application.

<dependency>

<groupId>org.testcontainers</groupId>

<artifactId>mongodb</artifactId>

<version>1.17.3</version>

<scope>test</scope>

</dependency>

Add this to pom.xml

Apart from this as we are JUNIT5 to write the test. We also need to add Junit module.

<dependency>

<groupId>org.testcontainers</groupId>

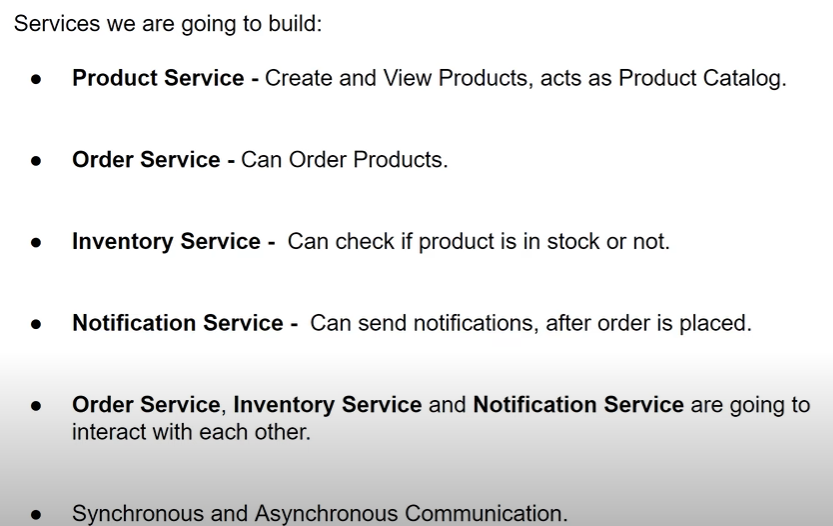
<artifactId>junit-jupiter</artifactId>

<version>1.17.3</version>

<scope>test</scope>

</dependency>

Add this to pom.xml



What is @document annotation in spring boot?

@Document is an annotation provided by Spring data project. It is **used to identify a domain object, which is persisted to MongoDB**. So you can use it to map a Java class into a collection inside MongoDB. If you don't use Spring Data, you don't need this annotation.

What is the use of @NoArgsConstructor and @AllArgsConstructor?

The constructor will throw a NullPointerException if any of the parameters intended for the fields marked with @NonNull contain null . The order of the parameters match the order in which the fields appear in your class. **@AllArgsConstructor generates a constructor with 1 parameter for each field in your class**.

What is @data annotation in spring boot?

@Data is **a convenient shortcut annotation that bundles the features of @ToString , @EqualsAndHashCode , @Getter / @Setter and @RequiredArgsConstructor together**: In other words, @Data generates all the boilerplate that is normally associated with simple POJOs (Plain Old Java Objects) and beans: getters for all fields, ..

What is @builder annotation in spring boot?

The @Builder annotation **produces complex builder APIs for your classes**. @Builder lets you automatically produce the code required to have your class be instantiable with code such as: Person. builder() .

**@Transactional annotation**

This annotation is used to combine the series of operations/action which need to be performed either all or none.

for eg. In a banking application debit and credit both operations need to perform to transfer the fund.

Here the transaction annotation will combine both of the operation into single unit, eighther both will take place at once or non of them will be performed (if any exceptions exist)

We can customise the transaction with various properties like propagation, rollback, timeout, no rollback.

You use @Transcational when concurrent calls on your API can affect each other.

Let's say you want to add a **Person** (you retreive data from somewhere, create a new **Person** from data and add it to a list of persons). Let's assume in order to create a Person you need a partner attribute which is another Person.

Before creating a Person you would search the partner by Id somehwere and add it to the new Person partner attribute. But what if during all this Queries the partneryou wanted to add is deleted somewhere (let's say in the database due to some other query). You'll end up not having the object you requested.

If you use @Transactional Spring makes sure all the required data is safe until the **Transaction** ends. Once it ends, the delete request from the partner would take place and then you'll have some logic to remove it from the new **Person** object. But that would take place afterwards.

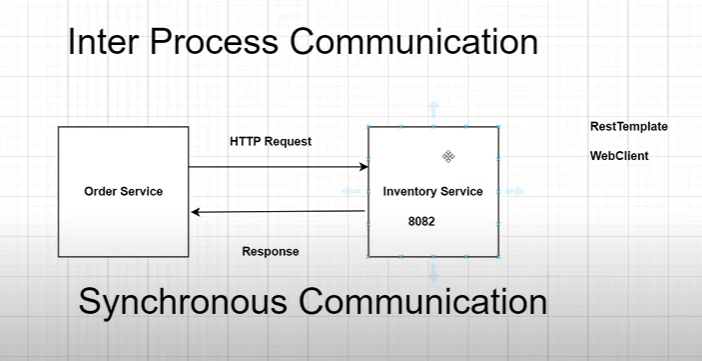
You use @Transactional to ensure safety on your "Transactions".

Migrate to Maven Multi-Module Project

Saari services ko 1 new project banakar us mai add karengay then module add karengay har service ka artifact id parent waali honi chaiye bs then src folder dlt kardengay parent project mai sai then har services ki pom.xml mai sai dependency copy kar k main k module mai karengay or dependency management or parent main project ki pom.xml mai add kardengay

Phr har module ka src dlt kar k jo services banaye hai unka src add kardengay

Phr maven clean verify karengay right side par intellij k likha hoga usko open kar k maven k logo par click kar k maven clean verify likhengay



What is the difference of synchronous and asynchronous?

**Synchronous = happens at the same time.** **Asynchronous = doesn't happen at the same time**. With synchronous learning, participants can receive immediate feedback. With asynchronous learning, the participants can learn at their own pace.

In synchronous communication, the order service will wait for the response from inventory whether the product is available or not.

We can use a web client or rest template for synchronous communication. The rest template class is in maintenance mode moreover, spring boot is recommending to use of a web client as an alternative because it has a more modern API and also supports sync, async, and streaming scenarios.

## Difference between WebClient and RestTemplate

#### 1.1. RestTemplate is blocking

RestTemplate are blocking in nature and uses one thread-per-request model of Java Servlet API. It means that RestTemplate will wait for the response everytime it dispatches a request to the remote server.

Creating and closing the URL connections is a costly operation. For using RestTemplate in our production class applications, we must use a [HTTP connection pooling](https://howtodoinjava.com/spring-boot2/resttemplate/resttemplate-httpclient-java-config/).

Please note that **RestTemplate is thread-safe** and a single instance can be shared across multiple connections at any time.

For each request, RestTemplate craetes a new Thread and uses it for request-response lifecyle. After sending a request, RestTemplate waits for the response from the server until a configured timeout is reached. This action blocks the thread.

#### WebClient is non-blocking

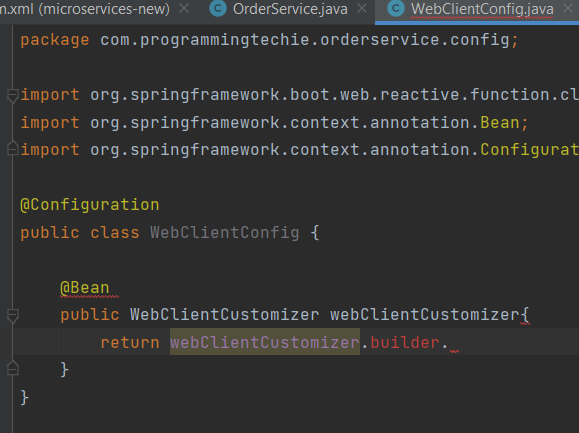
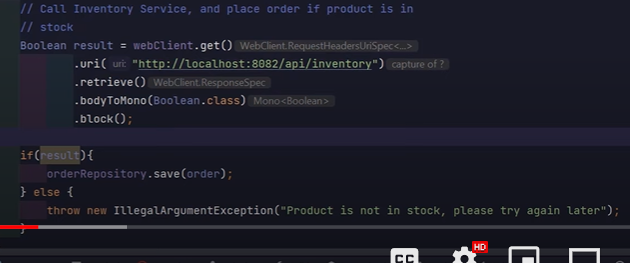
Opposite to RestTemplate, WebClient is asynchronous and non-blocking in nature. It follows **events-diven architecture** from reactive framework of Spring WebFlux.

Using WebClient, the client need not wait till response comes back. Instead it will be notified usign a **callback method** when there is a response from the server.

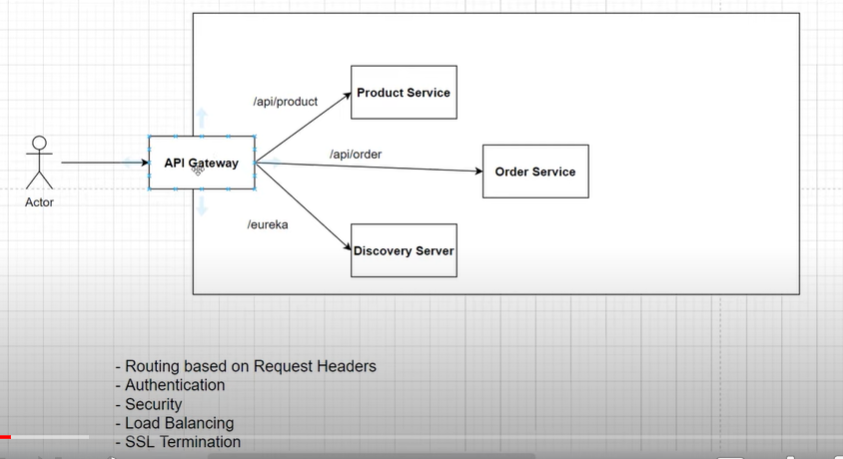
When we invoke an API through WebClient that returns a Mono or a Flux, it will return immediately. The results of the call will be delivered to us through the mono or flux callbacks when they become available.

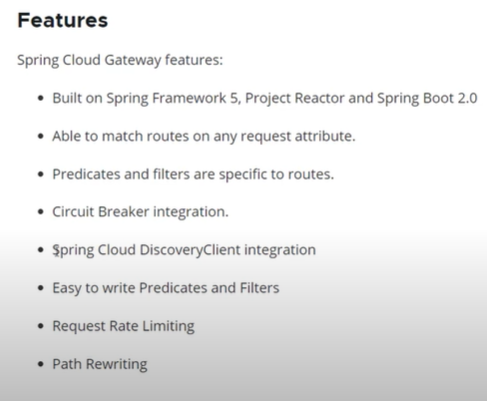
Please note that we can achieve RestTemplate like synchronous processing in WebClient using block() method.

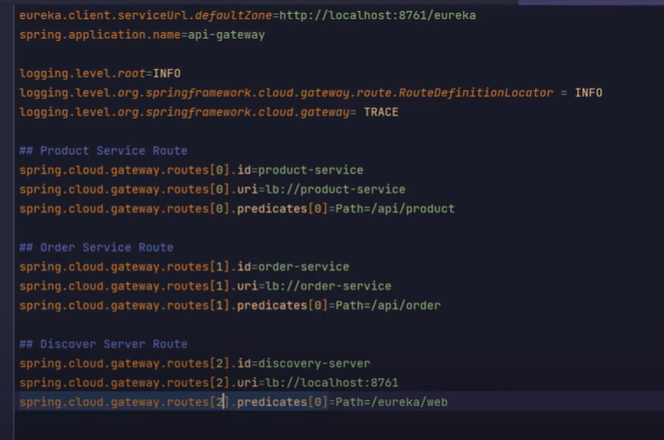
Steps:

* Create config package
* Create class webClientConfig and annotated this class with @Configuration
* Define a bean of type webclient
* Add spring web flux dependency to create bean webclient
* 
* Then inject private final Webclient webclient in order service.
* 
* So we are not done with implementation part you can observe that our inventory controller expecting a skucode as a path variable so whatever product you want to check inventory we have to pass sku code as part of the get request
* If you observe the order object each order can have multiple orderlineitems and each orderlineitems contains a sku code.
* For ex if user order 100 items har items ka alag sku code hoga or is tarah hundred request jayengi
* It takes so much time that’s why we collect all the sku code and then provide this list to inventory api

# API Gateway







What is predicate in Microservices?

Predicate − These are **the set of criteria which should match for the incoming requests to be forwarded to internal microservices**. For example, a path predicate will forward the request only if the incoming URL contains that path.

There are many predicates:

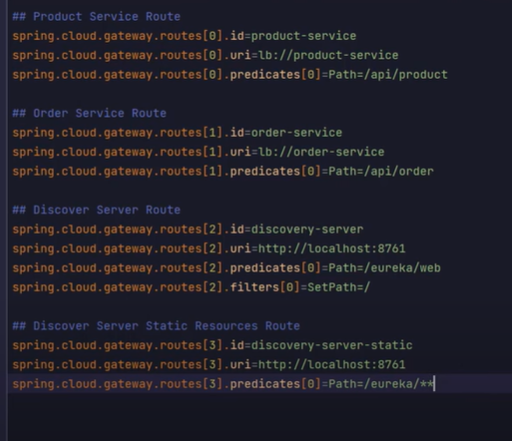
After

Before

Method

Cookie

Path Route predicate factory



### Spring Cloud Config Server

This module is used to externalize the configuration of our microservices into a centralized place.

If there are any changes in the configuration, our applications should be updated without the need to restart them.

There are many options to implement this Centralized Configuration eg: Using a **Git Repository**, or using **HashiCorp Consul** etc.

We are also going to use **HashiCorp Vault** as one of the backends to maintain the secrets for our application.

### Spring Cloud Bus

This module contains a light weight message broker implementation, which is mainly used to broadcast some messages to the other services.

In our project, we will use this module, to broadcast configuration changes in our Config Server.

### Spring Cloud Circuit Breaker

Inter-service Communication is common in the Microservice architecture, if one of the service is down, the other service which is communicating with it, should be able to handle this failure gracefully.

**Spring Cloud Circuit Breaker** provides an abstraction over different Circuit Breaker implementations like **Reslience4J**, **Hystrix**, **Sentinel** etc.

In our project, we are going to use **Spring Cloud Circuit Breaker** with **Resilience4J**

### Spring Cloud Sleuth

In Microservice Architecture, if there is any error, it’s hard to debug and trace it, **Spring Cloud Sleuth** provides us with the functionality to trace the inter-service calls.

### Spring Cloud Gateway

This library helps us to implement the [API Gateway pattern](https://microservices.io/patterns/apigateway.html), by hiding the complexity of our microservices from the external clients.

If a client want’s to connect to one of our services, all the traffic will be going through the API Gateway and the gateway will route the request to the appropriate service.

This library also handles the cross-cutting concerns like Security, Monitoring, Rate-limiting and Resiliency.

We will secure our microservices using **Keycloak** as Authorization Server together with the **Spring Cloud Gateway**.

### Spring Cloud Stream

This module mainly allows us to implement asynchronous communication between our microservices using event-driven architecture.

We are going to use RabbitMQ as a message broker to implement some event driven microservices.

### Distributed Logging using ELK Stack

Even though, this is not part of the **Spring Cloud**Stack, we are going to implement Centralized Logging for our microservices using **ELK Stack** (**Elasticsearch**, **Logstash** and **Kibana**)

The **spring-cloud-starter-bootstrap** is used to tell Spring framework to connect to the configuration server at the time of bootstrapping the application.

For this reason, we have to move our configuration from **application.properties** to **bootstrap.properties** file.