NORTON UNIVERSITY
COMPUTER STUDY
GRADUATE

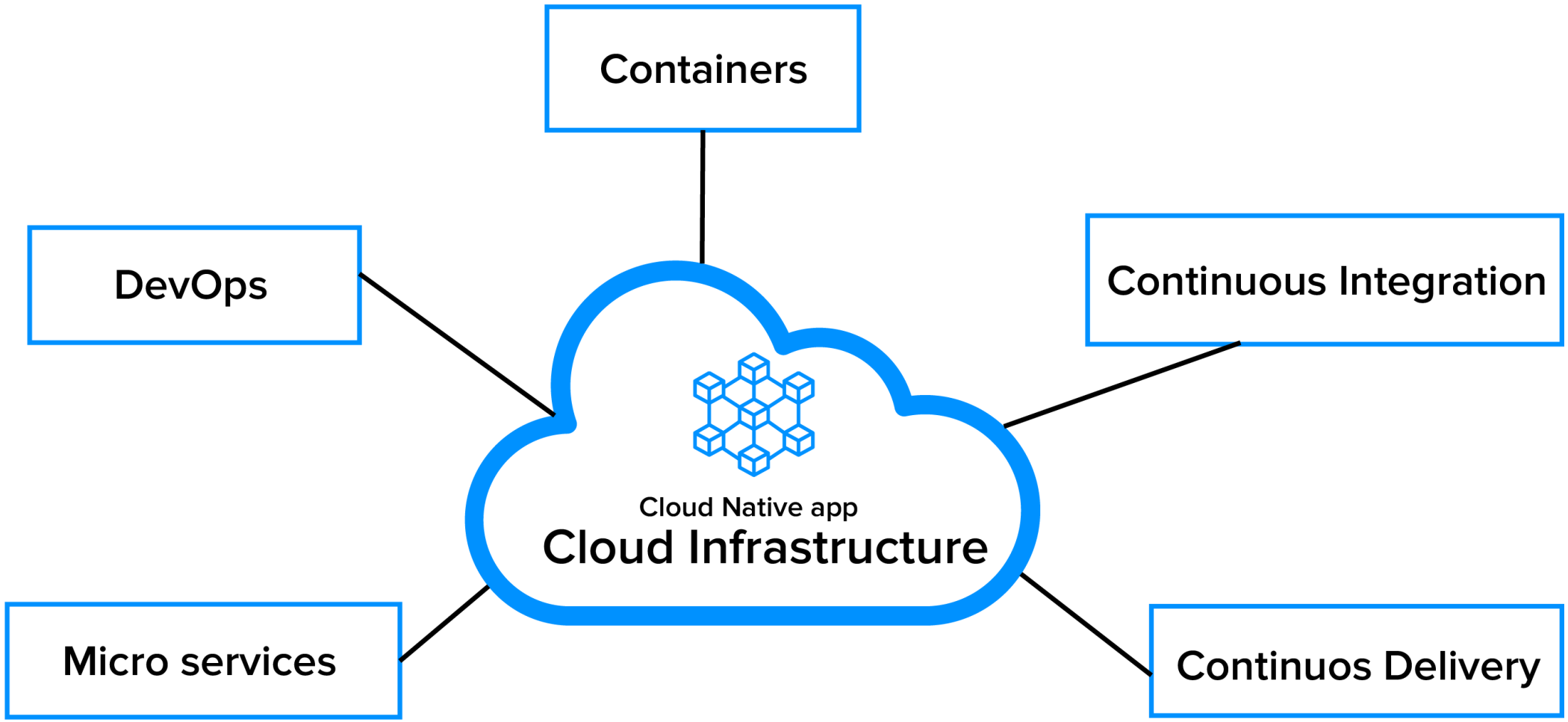
WEB DEVELOPMENT

SPRING CLOUD

LIM Sengdy, PhD
Mathematics, Computer Science, and USCT
HP: 012 946242
Email: lsengdy9@rac.gov.kh
2015-2026

WELCOME

Cloud represents a modern, dynamic, and distributed infrastructure model—public, private, or hybrid—rather than just a remote server. It refers to leveraging on-demand computing services like container orchestration (e.g., Kubernetes), microservices, and serverless computing to build scalable, resilient, and manageable applications. .


Containers
DevOps
Continuous Integration
Cloud Native app
Cloud Infrastructure
Micro services
Continuos Delivery

SPRING CLOUD

Spring Cloud is an open-source framework built on top of Spring Boot that simplifies the development of distributed, cloud-native microservices applications. It provides tools to manage common patterns like service discovery, configuration, circuit breakers, and routing, making it easier to build and deploy resilient applications in cloud environments.

What is Spring Cloud ?

Spring Cloud Contract

Spring Cloud Kubernetes

Spring Cloud Netflix

Spring Cloud Stream

Spring Cloud Sleuth

Spring Cloud Config

Spring Cloud Function

Spring Cloud Gateway

Spring Cloud Common

Spring Cloud Task

Spring Cloud Vault

Spring Cloud Circuit Breaker

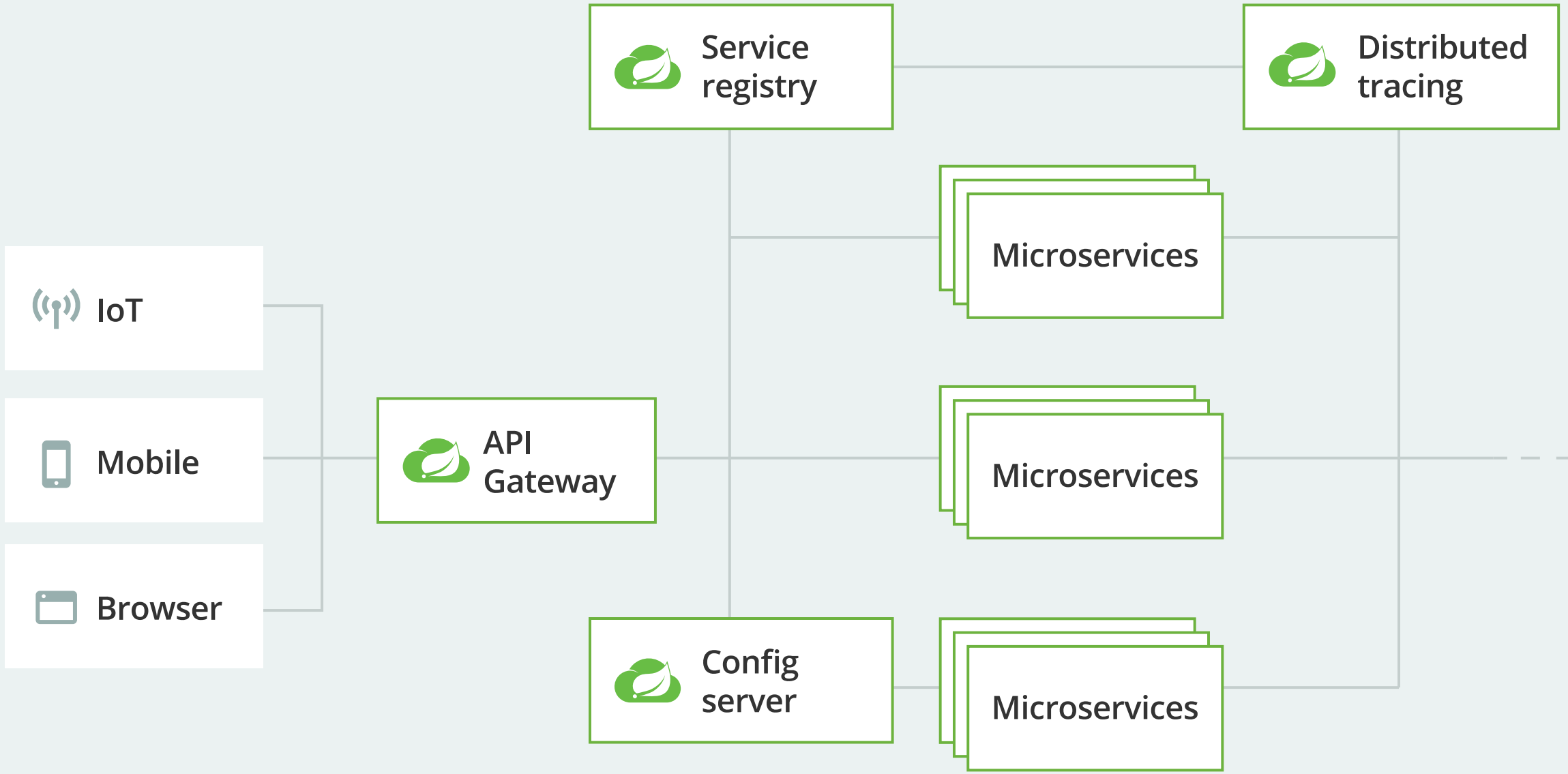
Spring Cloud for AWS

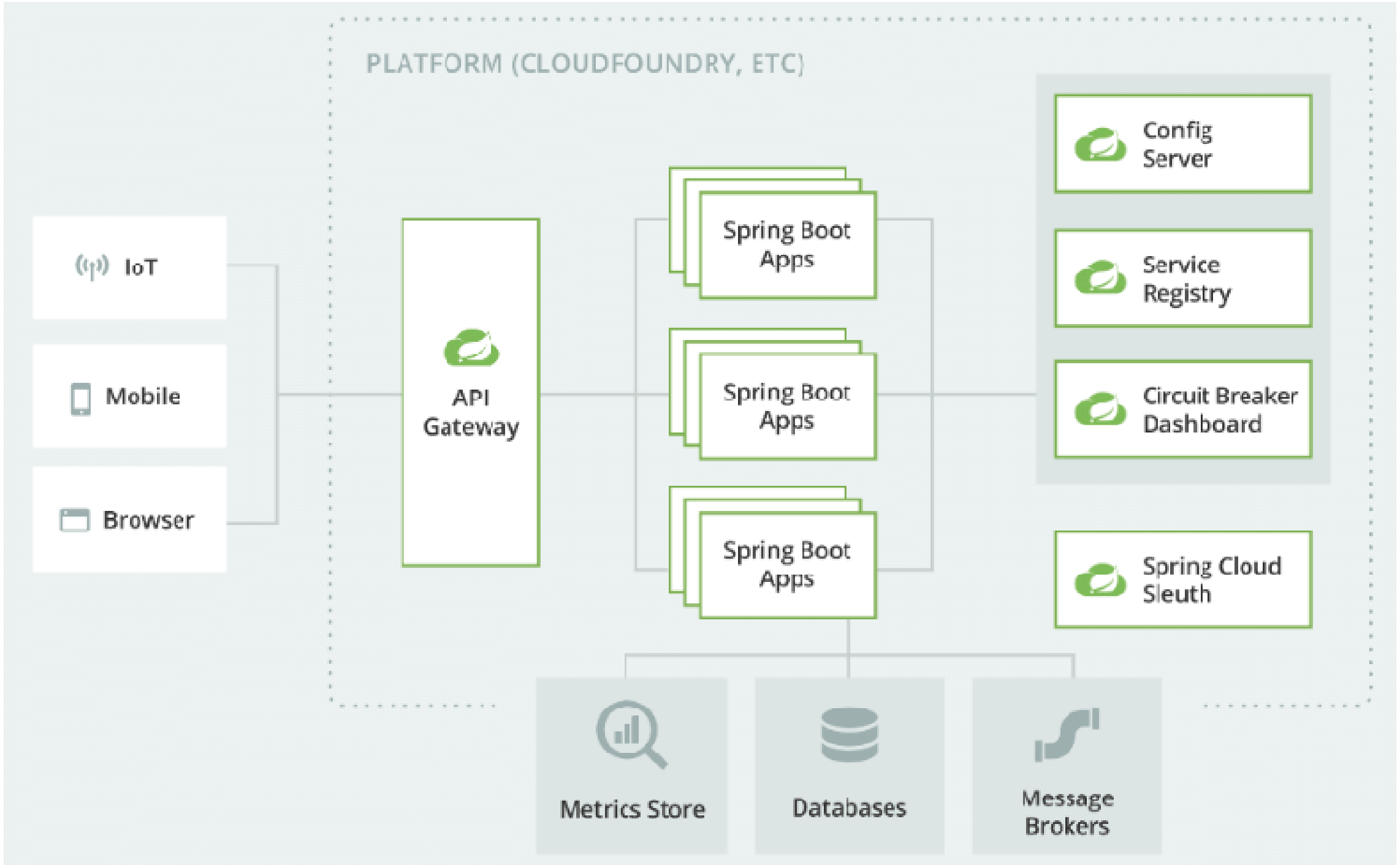
Spring Cloud Bus

Spring Cloud Security

Spring Cloud Architecture

Built on Spring Boot, it simplifies managing complex, scalable architectures across different environments, such as cloud providers or on-premise, by offering pre-built components for common challenges.

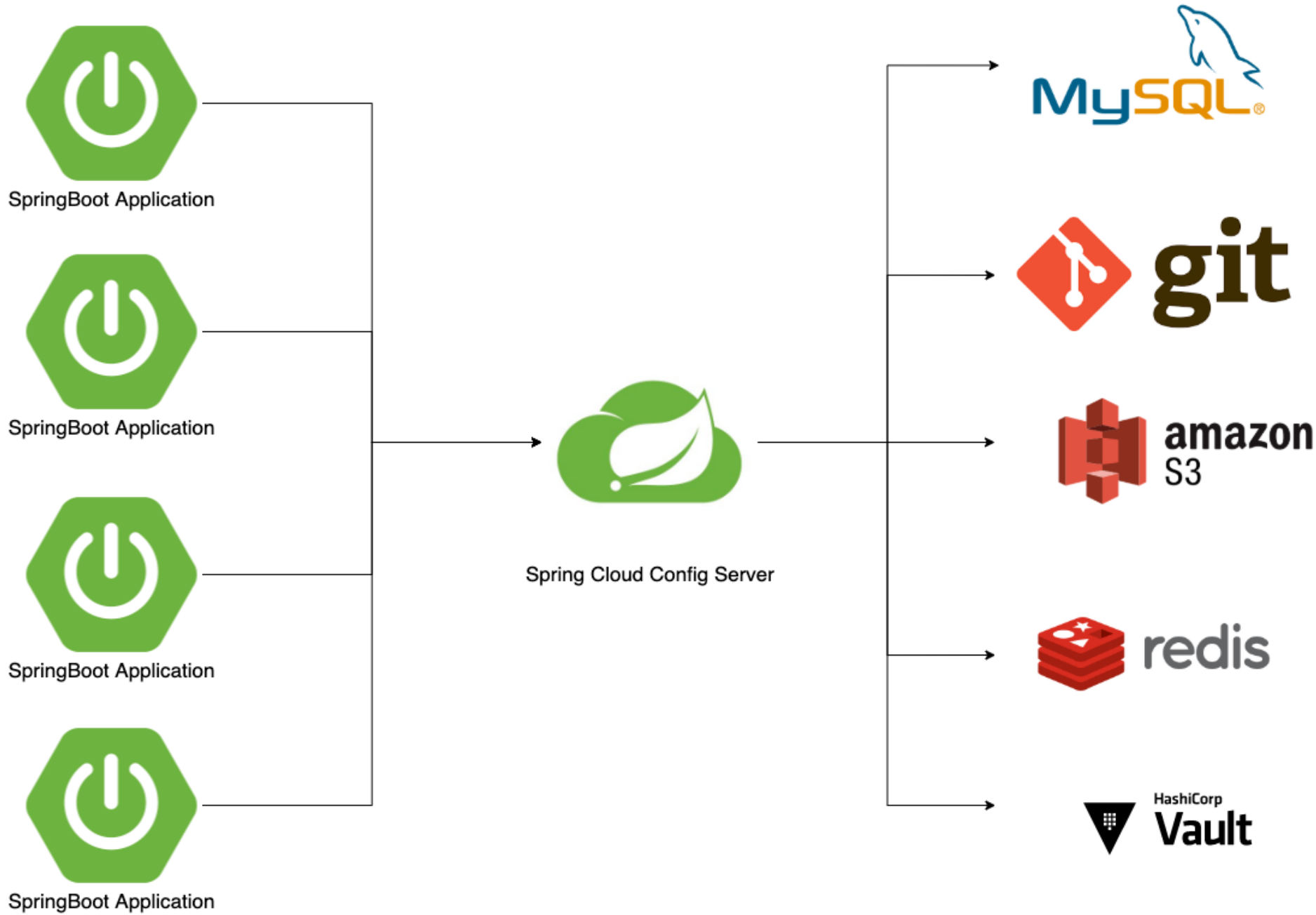

Service registry
Distributed tracing
Microservices
IoT
Mobile
Browser
API Gateway
Microservices
Config server
Microservices


PLATFORM (CLOUDFOUNDRY, ETC)
IoT
Mobile
Browser
API Gateway
Spring Boot Apps
Spring Boot Apps
Spring Boot Apps
Config Server
Service Registry
Circuit Breaker Dashboard
Spring Cloud Sleuth
Metrics Store
Databases
Message Brokers

- **Spring Cloud Config**

In the cloud, configuration can’t simply be embedded inside the application.

Spring Cloud Config provides a centralized, externalized, and version-controlled way to manage application configurations in a distributed system, especially useful for microservices. It allows you to store configuration data outside of your application code, making it easier to manage and update configurations across different environments and services.


SpringBoot Application
SpringBoot Application
SpringBoot Application
SpringBoot Application
Spring Cloud Config Server
MySQL
git
amazon
S3
redis
HashiCorp
Vault

+ Config Server

It is a centralized, version-controlled system for managing external configuration properties for microservices and other applications. It acts as a central configuration service that allows applications to retrieve their configuration from a single, consistent source, ensuring that all microservices use the same configuration values. This approach simplifies configuration management, especially in distributed systems like microservices, by centralizing the process and enabling dynamic updates without requiring application restarts.

```
<dependency> <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-config-server</artifactId> </dependency>
```

+ Config Client

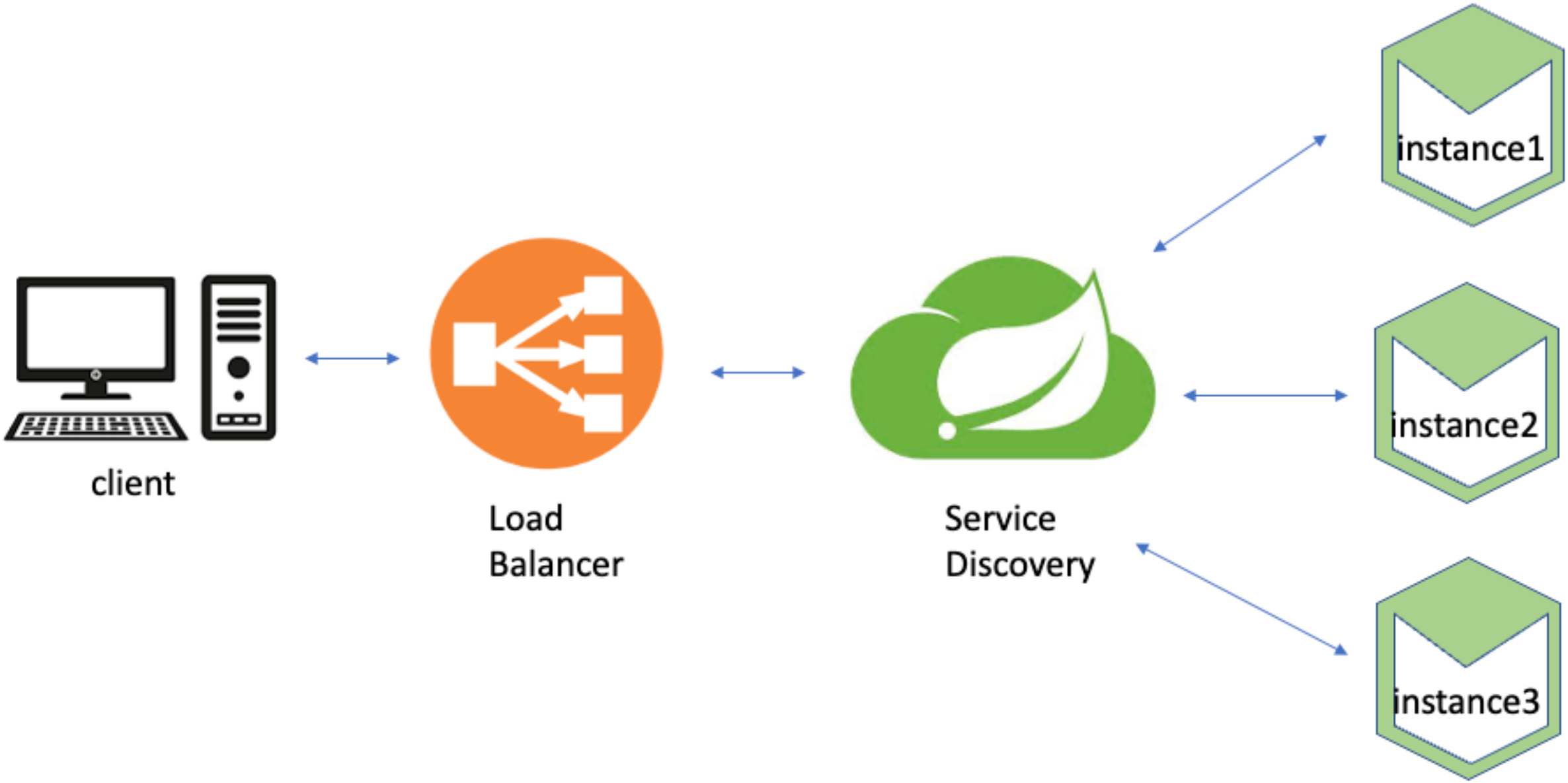
Spring Cloud Config Client allows Spring Boot applications to retrieve their configuration properties from a central **configuration server**, like the Spring Cloud Config Server. It acts as a client, fetching configuration data from a centralized location rather than having it embedded within the application itself. This enables dynamic updates to application configurations without requiring application restarts.

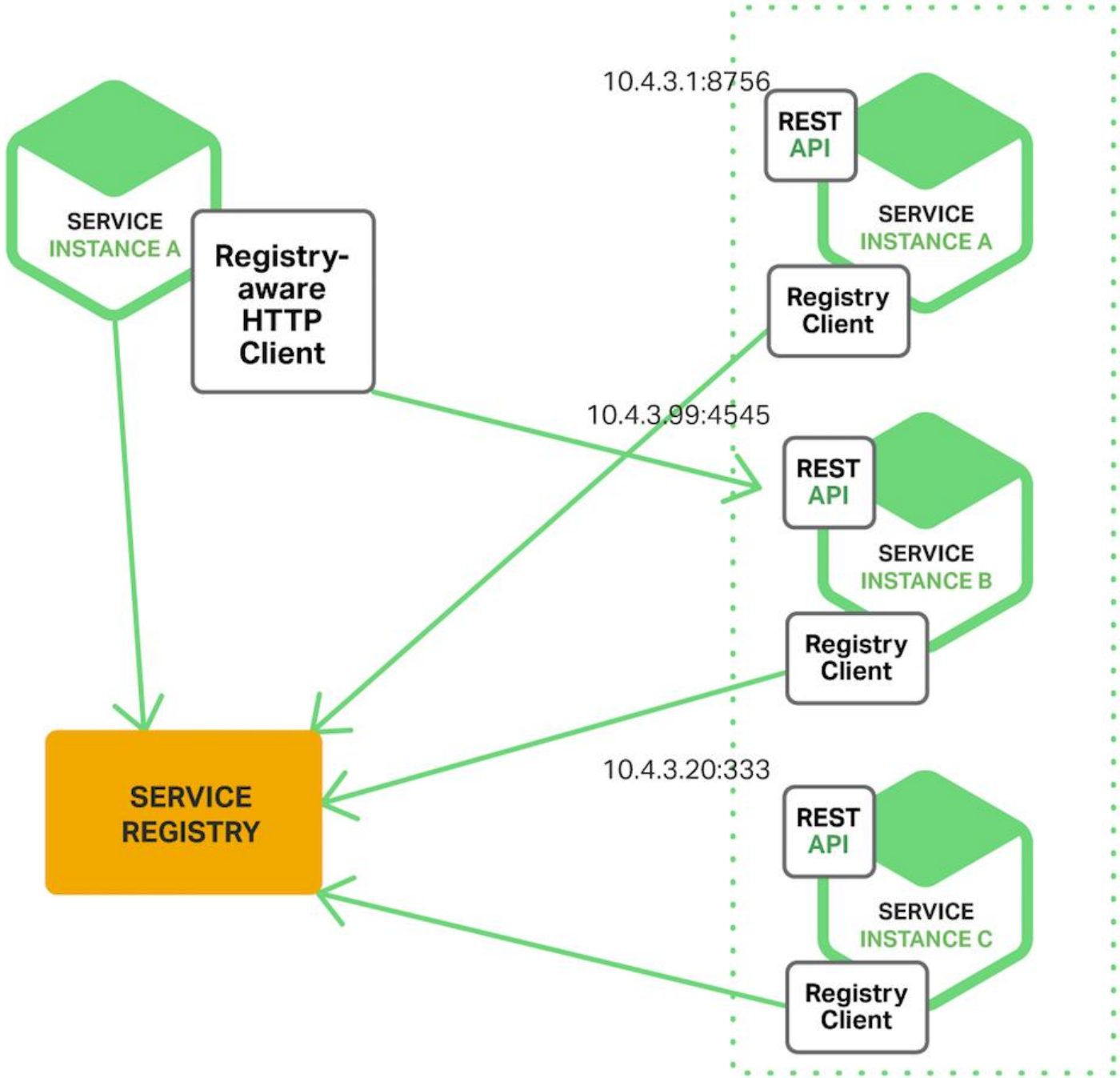
```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-config</artifactId>
</dependency>
```

- **Spring Cloud Discovery**

In the cloud, applications can’t always know the exact location of other services.

Spring Cloud Discovery is a mechanism within the Spring Cloud framework that enables services within a distributed system to dynamically locate and communicate with each other. It allows services to register themselves with a service registry and then allows other services to discover and connect to them without needing to know their physical locations (IP addresses and ports) in advance.

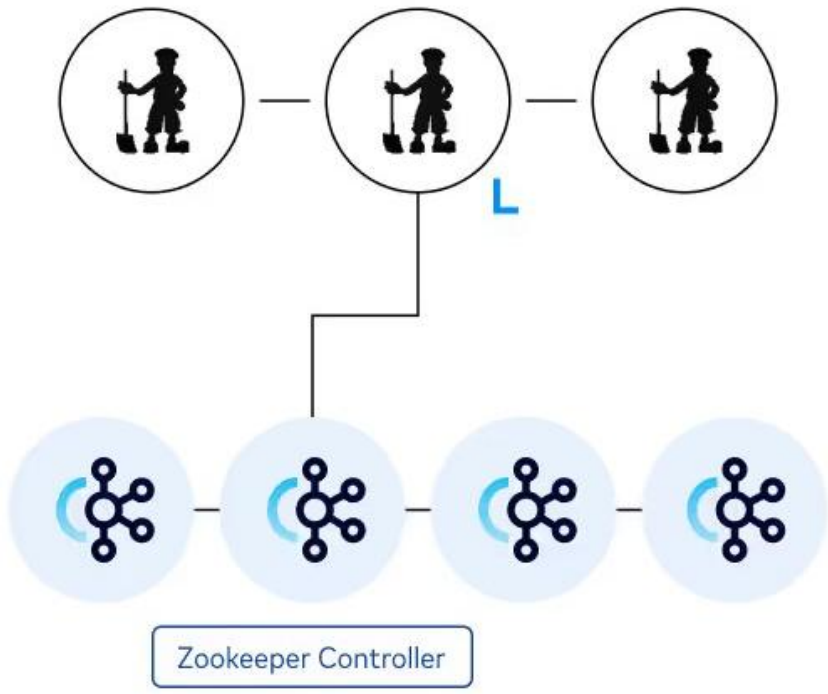

client
Load
Balancer
Service
Discovery
instance1
instance2
instance3


10.4.3.1:8756
REST API
SERVICE INSTANCE A
Registry Client
SERVICE INSTANCE A
Registry-aware HTTP Client
10.4.3.99:4545
REST API
SERVICE INSTANCE B
Registry Client
SERVICE REGISTRY
10.4.3.20:333
REST API
SERVICE INSTANCE C
Registry Client

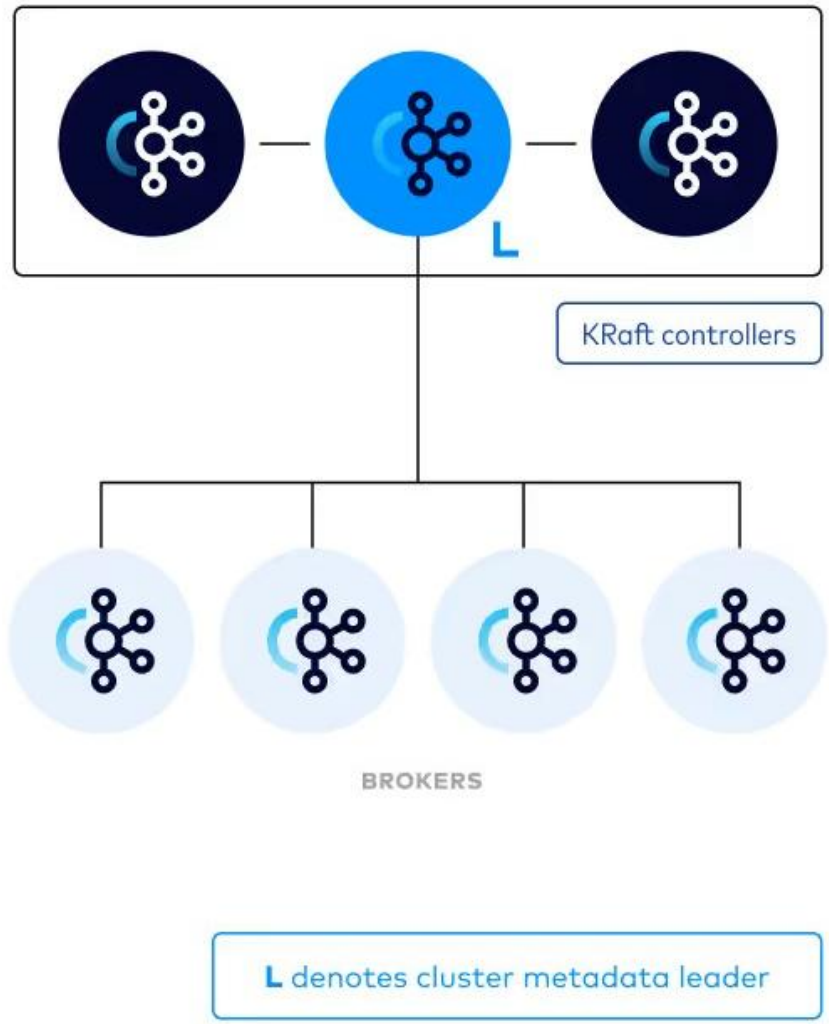
+ Kafka Raft (Kraft)

Apache Kafka is moving away from relying on **Apache ZooKeeper Discovery** for its metadata management and controller functionality. Instead, it's using a new internal consensus mechanism called Kafka Raft (**KRaft**), based on the Raft consensus algorithm. This shift simplifies Kafka's architecture and improves its scalability and performance.

Zookeeper



KRaft



L denotes cluster metadata leader

+ Kubernetes

Kubernetes service discovery is a core mechanism that allows applications (Pods) to dynamically find and communicate with each other using stable DNS names or environment variables, abstracting away the ephemeral nature of Pod IP addresses. This ensures application resilience and scalability in a dynamic containerized environment.

+ Consul Discovery

Spring Cloud Consul Discovery is a Spring Cloud project that integrates Spring Boot applications with HashiCorp's Consul for service discovery and registration. It simplifies the process of registering services with Consul, allowing them to be discovered and used by other applications in a distributed environment. This integration is achieved through auto-configuration and binding to the Spring Environment and other Spring programming model idioms.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-consul-discovery</artifactId>
</dependency>
```

- **Spring Cloud Circuit Breaker**

Distributed systems can be unreliable. Requests might encounter timeouts or fail completely.

Spring Cloud Circuit Breaker is a Spring Cloud module that provides an abstraction layer for integrating circuit breaker patterns into Spring Boot applications. It allows developers to easily incorporate resilience patterns like circuit breakers, which help prevent cascading failures in distributed systems, by providing a consistent API across different circuit breaker implementations.

Circuit Breaker

+ Resilience4j

Resilience4j provides a way to integrate the circuit breaker pattern into Spring Boot applications using the Resilience4j library. It offers an abstraction layer over different circuit breaker implementations, allowing developers to easily incorporate resilience patterns like circuit breaking, rate limiting, and retries. Resilience4j itself is a lightweight, fault-tolerance library that provides various resilience patterns, including circuit breakers.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-circuitbreaker-resilience4j
</artifactId> </dependency>
```

- **Spring Cloud Routing**

Spring Cloud Routing, specifically through Spring Cloud Gateway, is a mechanism for directing network traffic to different microservices within a distributed system. It acts as an API gateway, providing a single entry point for external requests and routing them to the appropriate internal services. This simplifies communication between clients and microservices, handling tasks like routing, filtering, and security.

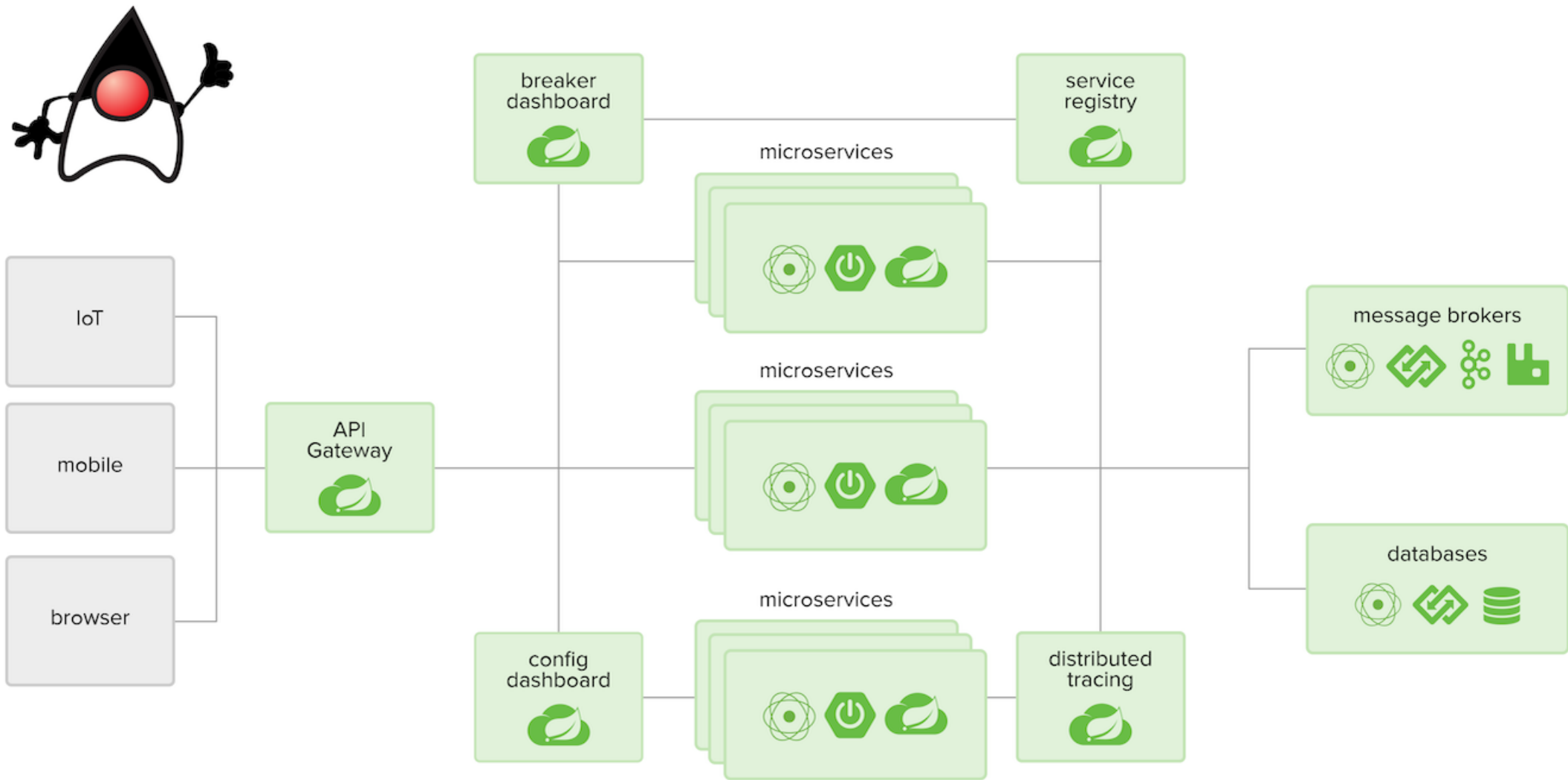


Diagram from https://spring.io

+ Gateway

With so many clients and servers in play, it's often helpful to include an API gateway in your cloud architecture. A gateway can take care of securing and routing messages, hiding services, throttling load, and many other useful things.

Spring Cloud Gateway (Reactive Gateway) is a routing and filtering mechanism built on the Spring Framework that acts as an API gateway for microservices.

```
<dependency>  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-gateway-server-
                    webmvc</artifactId></dependency>
or <artifactId>spring-cloud-starter-gateway-server-webflux </artifactId>
```

+ Cloud LoadBalancer

Spring Cloud LoadBalancer is a **client-side** load balancing solution for Spring Boot applications, particularly useful in microservices architectures. It distributes incoming requests across multiple instances of a service, preventing any single instance from being overwhelmed and improving overall application availability and performance.

```
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-loadbalancer</artifactId>
</dependency>
```

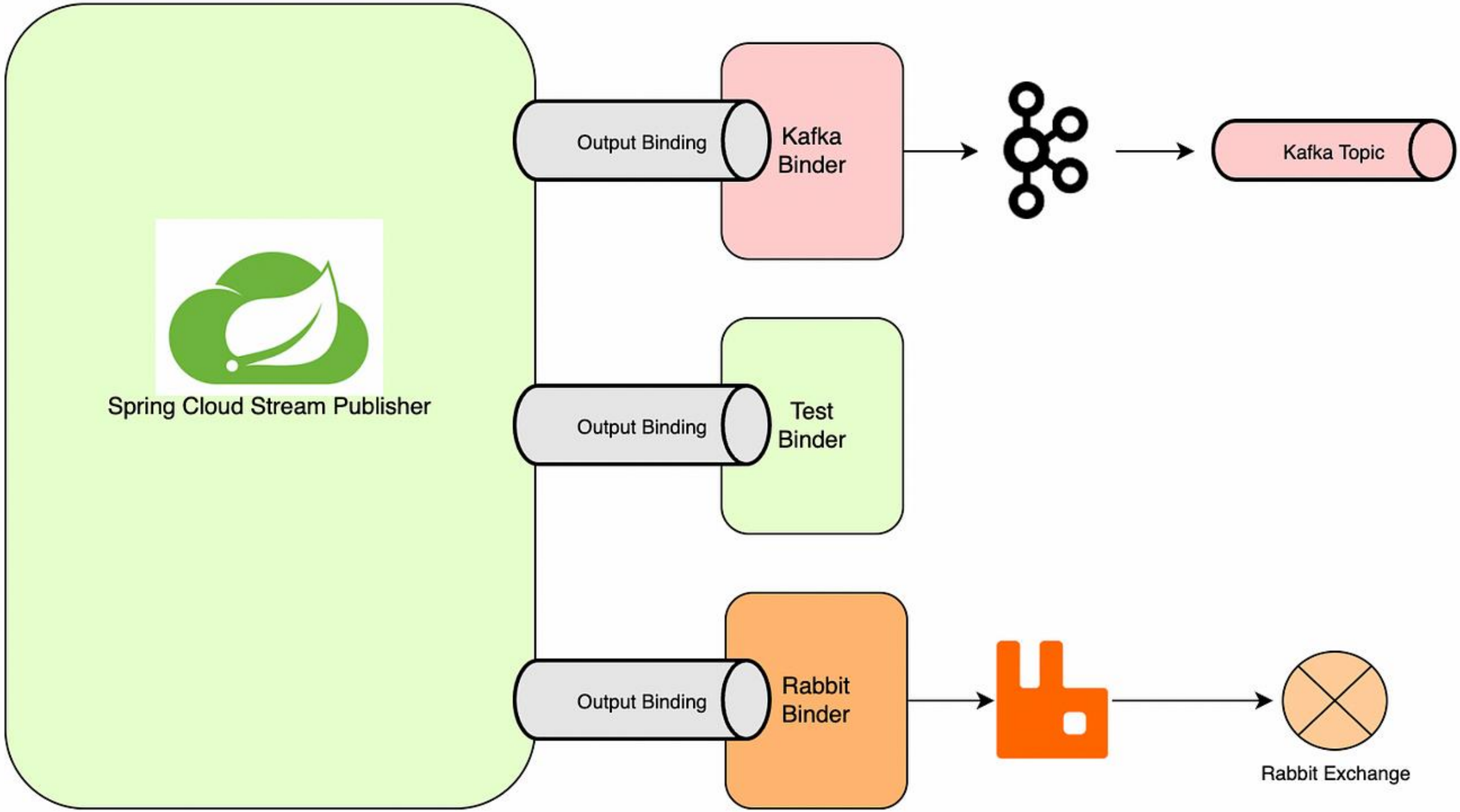
+ OpenFeign

Spring Cloud OpenFeign is a declarative web service client that simplifies the process of writing HTTP clients for interacting with RESTful services in Spring Boot applications, particularly within microservices architectures. It leverages interfaces and annotations to define how your application should communicate with external services, reducing boilerplate code and making the process more concise and maintainable.

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

- ## Spring Cloud Messaging

Spring Cloud Messaging, specifically Spring Cloud Stream, is a framework built on top of Spring Boot and Spring Integration that simplifies the development of message-driven microservices. It provides a consistent programming model for building scalable, event-driven applications that interact with message brokers like **Kafka** and **RabbitMQ**.


Spring Cloud Stream Publisher
Output Binding
Kafka Binder
Kafka Topic
Output Binding
Test Binder
Output Binding
Rabbit Binder
Rabbit Exchange

+ Cloud Stream

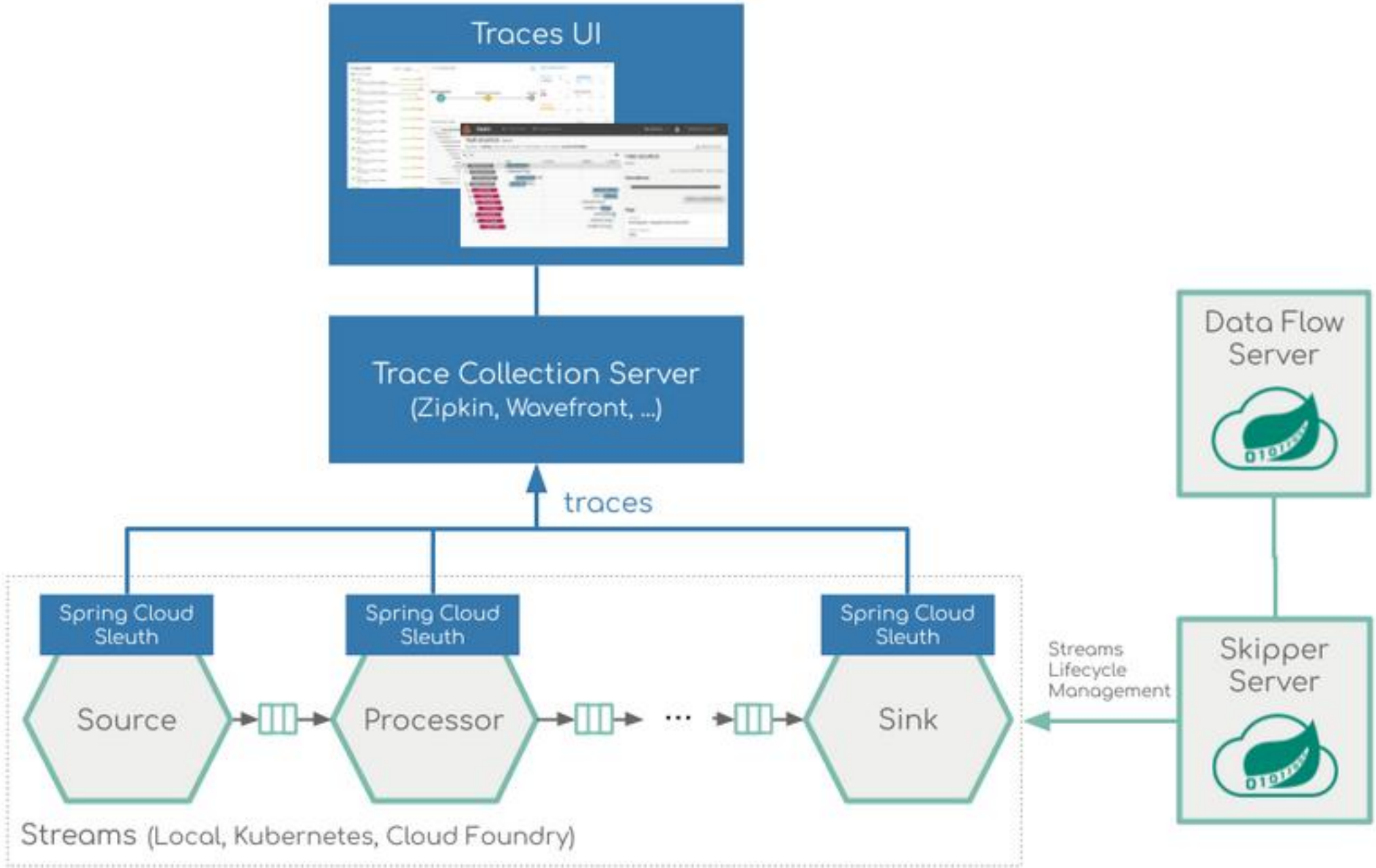
Spring Cloud Stream is a framework that simplifies building event-driven or message-driven microservices by providing a consistent programming model for interacting with messaging systems. It abstracts the complexities of message broker interaction, allowing developers to focus on business logic rather than integration details. Essentially, it acts as a bridge between your Spring Boot applications and message brokers like **Kafka** or **RabbitMQ**.

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-stream</artifactId>
</dependency>
```

+ Cloud Bus

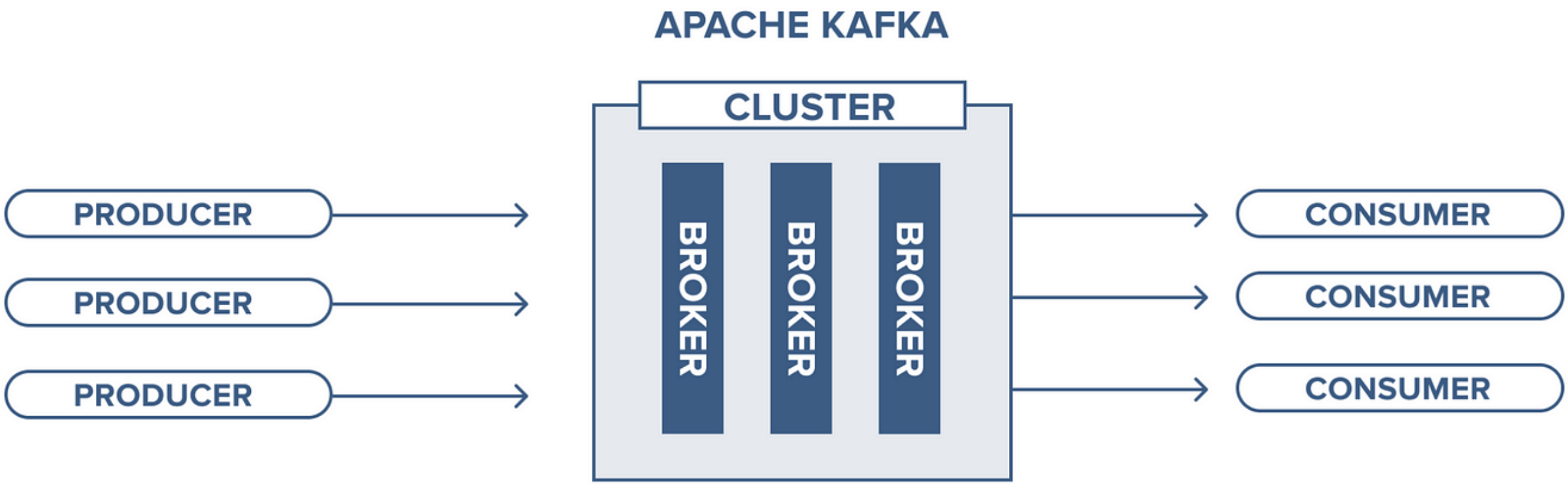
Spring Cloud Bus is a project within the Spring Cloud framework that links nodes in a distributed system using a lightweight message broker, enabling the broadcasting of state changes or management instructions across multiple microservices. It acts as a distributed actuator, facilitating the propagation of configuration changes or other management events among microservices.

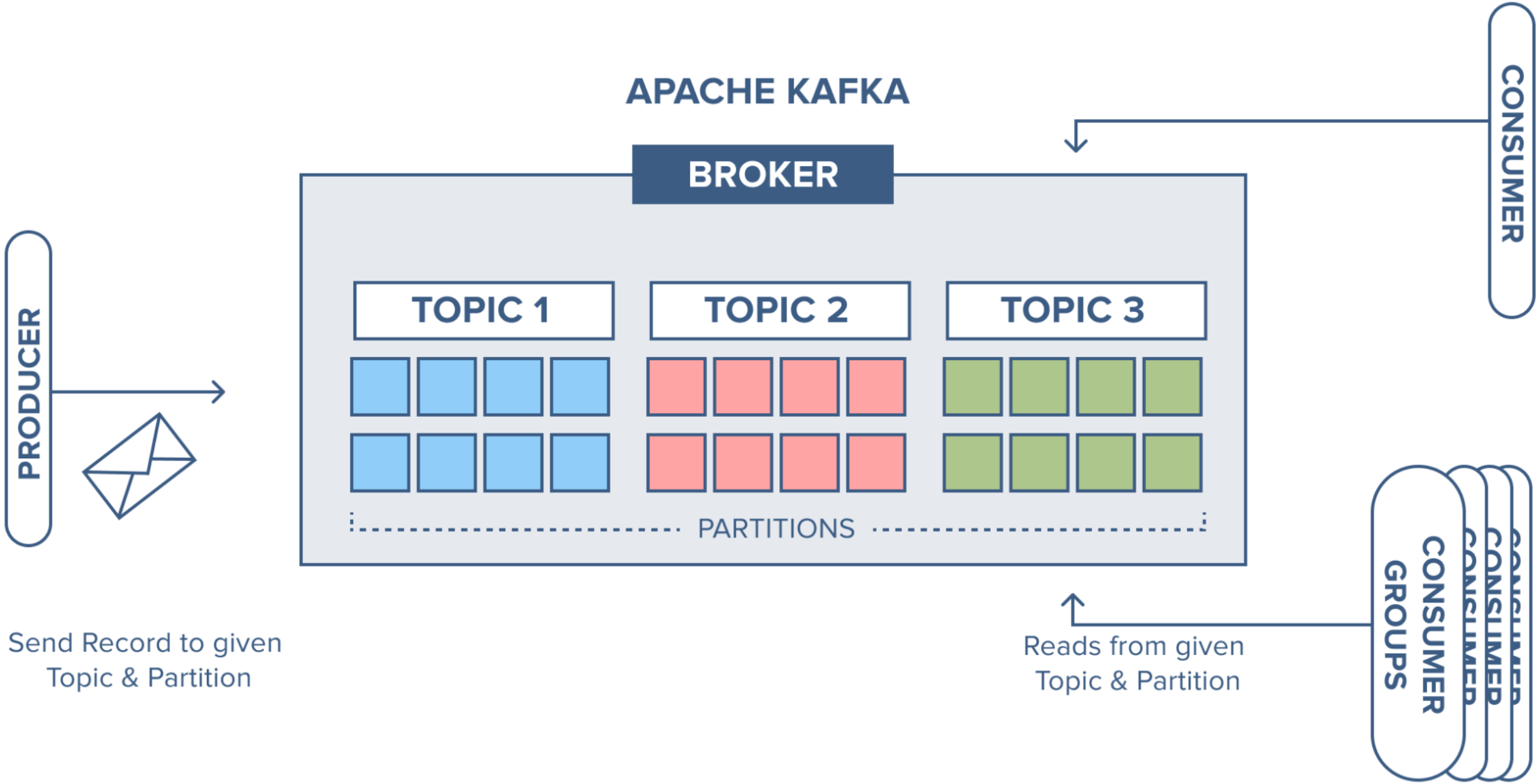
```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-bus</artifactId>
</dependency>
```

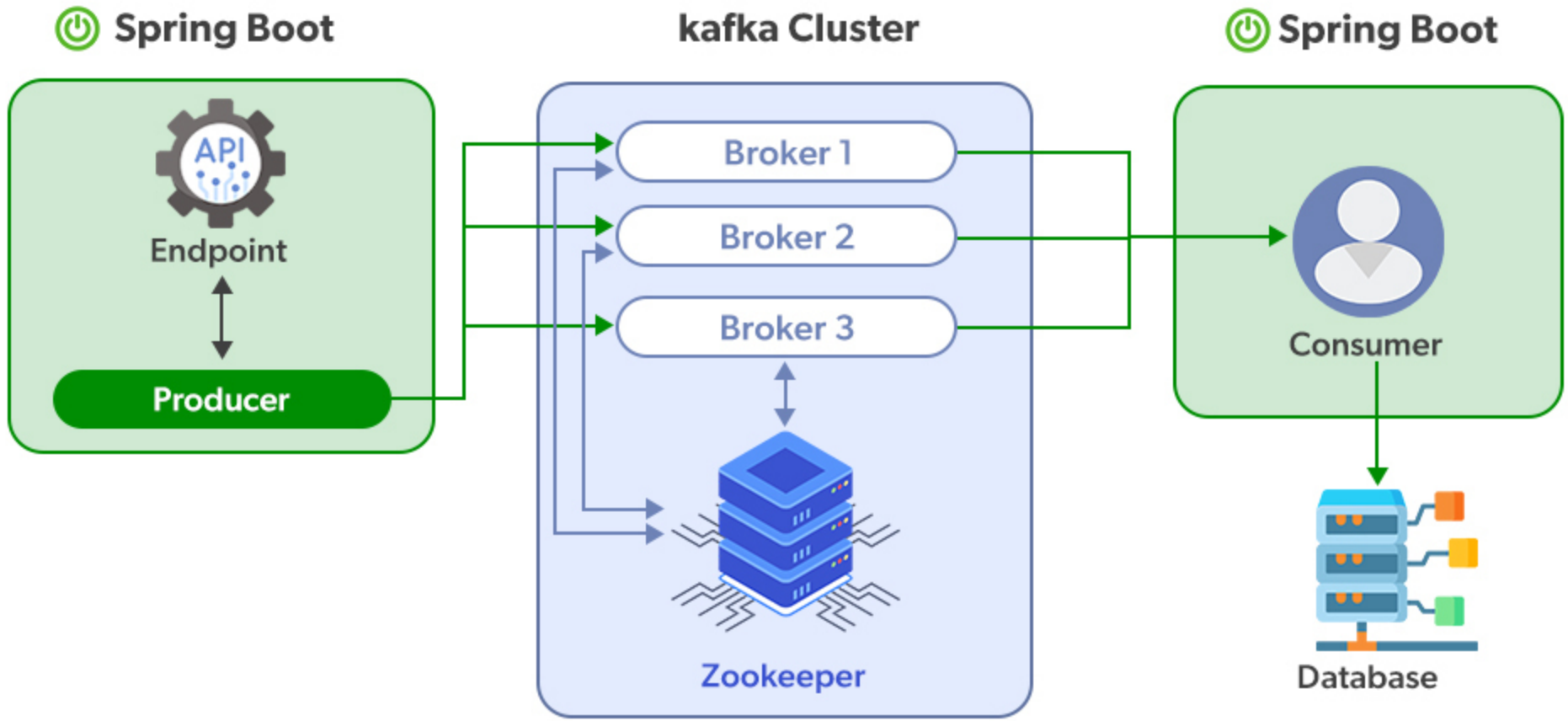

Traces UI
Trace Collection Server
(Zipkin, Wavefront, ...)
traces
Data Flow
Server
Spring Cloud
Sleuth
Spring Cloud
Sleuth
Spring Cloud
Sleuth
Source
Processor
Sink
Streams
Lifecycle
Management
Skipper
Server
Streams (Local, Kubernetes, Cloud Foundry)

Apache Kafka

Apache Kafka is a distributed, fault-tolerant, high-throughput, and scalable platform for building real-time data pipelines and streaming applications. It's fundamentally a distributed **publish-subscribe** messaging system, acting as a central nervous system for data in many organizations. Key concepts include producers, consumers, topics, partitions, brokers, and the publish-subscribe model.


APACHE KAFKA
CLUSTER
BROKER
BROKER
BROKER
PRODUCER
PRODUCER
PRODUCER
CONSUMER
CONSUMER
CONSUMER

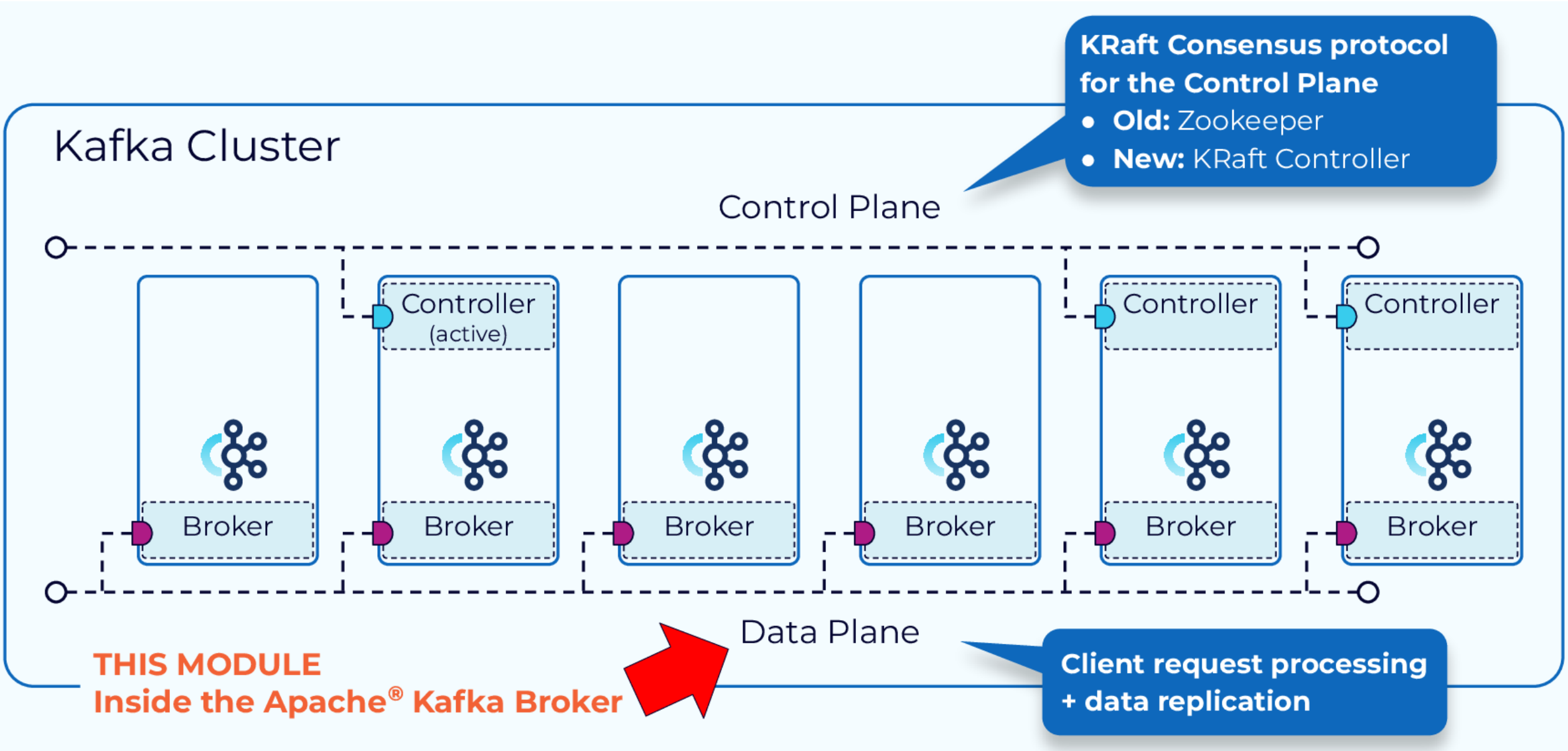

APACHE KAFKA
BROKER
TOPIC 1
TOPIC 2
TOPIC 3
PARTITIONS
PRODUCER
CONSUMER
CONSUMER GROUPS
Send Record to given
Topic & Partition
Reads from given
Topic & Partition


Spring Boot
kafka Cluster
Spring Boot
API
Endpoint
Producer
Broker 1
Broker 2
Broker 3
Zookeeper
Consumer
Database

Cluster

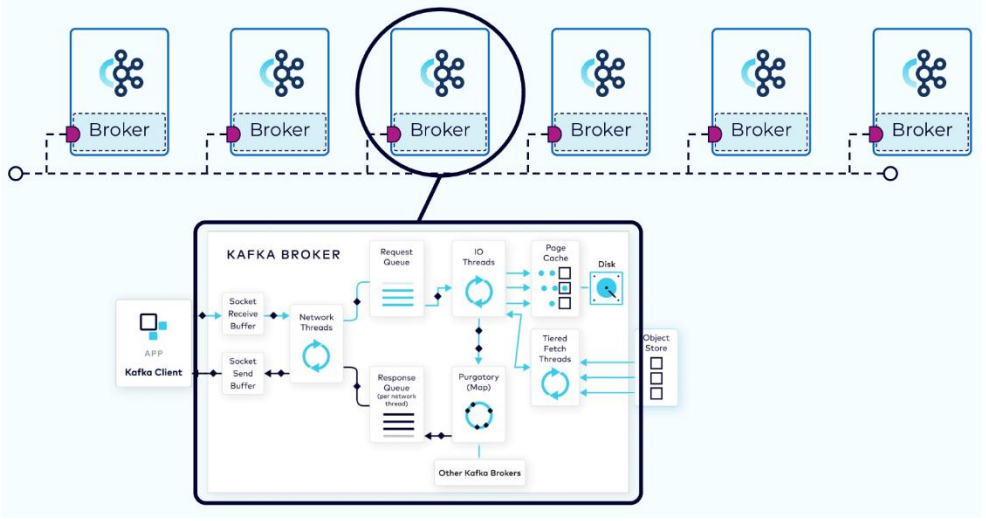
A Kafka cluster is a distributed system comprised of **multiple brokers** (servers) that work together to manage and distribute data streams. It's designed for high throughput, low latency, and fault tolerance, enabling efficient real-time data processing and storage.

The functions within a Kafka cluster are broken up into a **data plane** and a **control plane**. The **control plane** handles management of all the metadata in the cluster. The **data plane** deals with the actual data that we are writing to and reading from Kafka.


KRaft Consensus protocol for the Control Plane
• Old: Zookeeper
• New: KRaft Controller
Kafka Cluster
Control Plane
Controller (active)
Controller
Controller
Broker
Broker
Broker
Broker
Broker
Broker
Data Plane
THIS MODULE
Inside the Apache® Kafka Broker
Client request processing + data replication

- **Broker**

A Kafka broker is a **server** within an Apache Kafka cluster that acts as the central point for handling data. It receives messages from producers, stores them, and serves them to consumers upon request. Essentially, it's the workhorse of Kafka, managing data storage, delivery, and replication.



Producer

A Kafka Producer is a client application that publishes (**writes**) data to Kafka topics within a Kafka cluster. It acts as the source of data, sending messages to Kafka brokers, which are responsible for storing and managing the data. Producers utilize the Kafka client library to interact with the Kafka cluster, enabling them to send streams of data to specified topics.

Consumer

A Kafka consumer is a client application that **reads** and **processes** data (events) from Kafka topics. It subscribes to topics, pulls messages, and then processes them, often updating databases, triggering other actions, or sending the data elsewhere. Consumers play a vital role in Kafka's event-driven architecture, working alongside producers and the Kafka broker.

- **Consumer Group**

It is a set of consumers working together to read and process messages from one or more Kafka topics. Each consumer within a group subscribes to a subset of the topic's partitions, allowing for parallel consumption and processing of messages.

- **Topic**

Kafka topic is a fundamental concept that acts as a categorized stream of messages. Think of it as a named channel or feed where producers write data (messages) and consumers read data.

- ## Partitions

Topic is divided into **partitions**, which are distributed across multiple brokers for scalability and fault tolerance. Each partition acts as an ordered, immutable log.

- ## Offsets

Unique identifiers for messages within **a partition**. Kafka uses offsets to track the position of consumers within a partition.

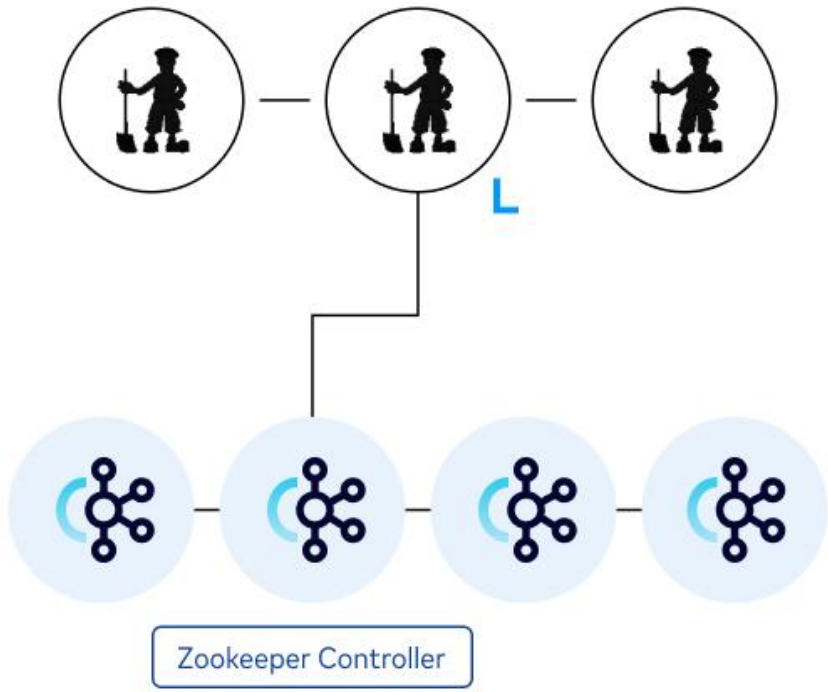
Apache Kafka Raft

Apache Kafka Raft (KRaft) mode is a new consensus protocol that enable Kafka to operate without **ZooKeeper** for metadata management.

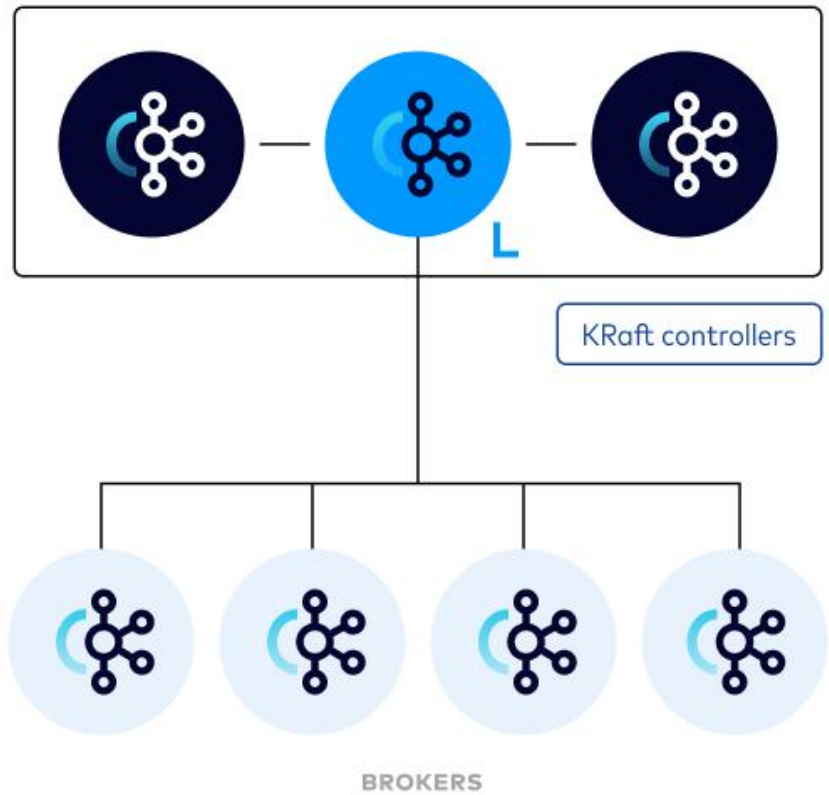
KRaft simplifies Kafka's architecture by managing metadata internally using the Raft consensus algorithm, improving reliability and simplifying cluster management.

In KRaft mode, Kafka brokers themselves elect a **controller** (or controllers) to manage cluster metadata using the Raft consensus protocol.

Zookeeper



KRaft

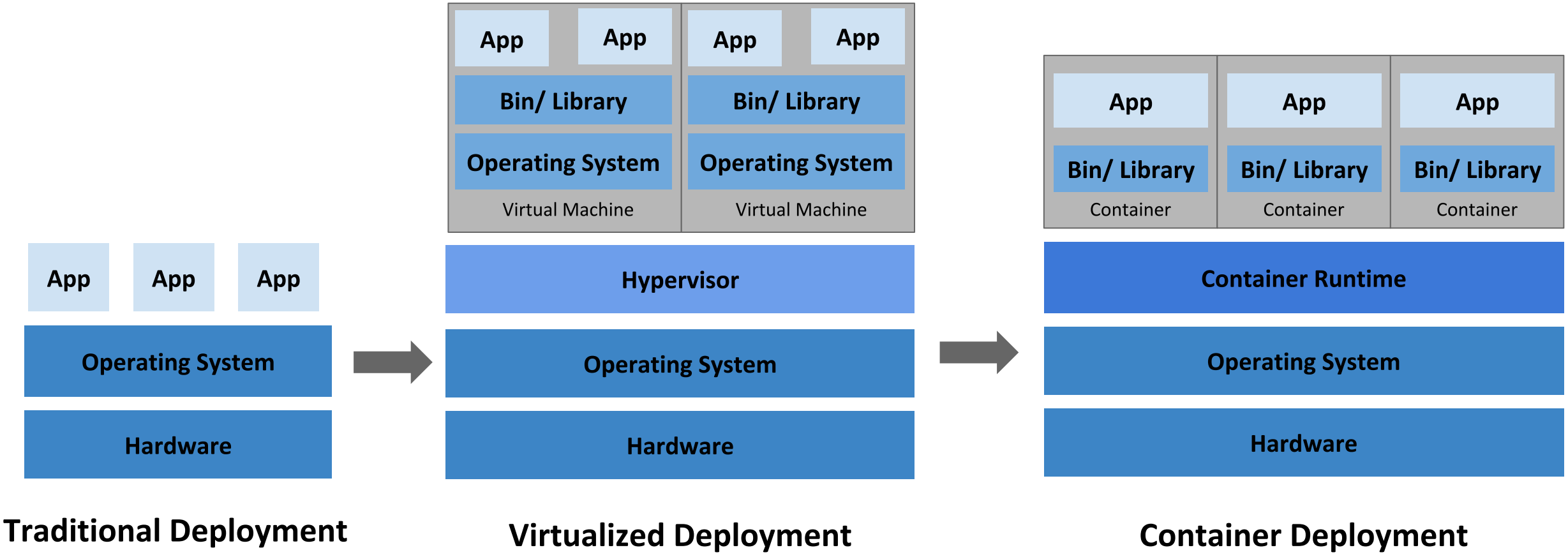


L denotes cluster metadata leader

KUBERNETES

Kubernetes is a portable, extensible, open source platform for managing containerized workloads and services that facilitate both declarative configuration and automation. It has a large, rapidly growing ecosystem.

Use Kubernetes to manage your **services**, and use Kafka (with KRaft) inside Kubernetes to handle the **messaging** and event-streaming for your Discover service.


App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Container Deployment

- **Orchestration**

orchestration is the automated process of deploying, managing, scaling, and networking containerized applications throughout their lifecycle. It acts as a system that ensures applications run as intended by continuously reconciling the actual system state with a user-defined desired state, handling complex tasks that would be difficult to manage manually at scale.

It groups containers into logical units called "**pods**" and schedules them across nodes (servers).

- **Automation**

It automates deployment, scaling (up/down), and management of containerized apps.

- **Resource Management**

It efficiently allocates hardware resources and manages storage.

- **Declarative Configuration**

You define the desired state, and Kubernetes works to maintain it.

- **Service Discovery & Load Balancing**

Kubernetes can expose a container using a DNS name or IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.

- **Self-Healing**

Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Spring Cloud Lab 1

Spring cloud Routing

Gateway/Reactive Gateway

Spring Initializr

start.spring.io

spring initializr

Project

- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language

- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot

- ○ 4.0.0 (SNAPSHOT)
- ○ 3.5.4 (SNAPSHOT)
- ● 3.5.3
- ○ 3.4.8 (SNAPSHOT)
- ○ 3.4.7

Project Metadata

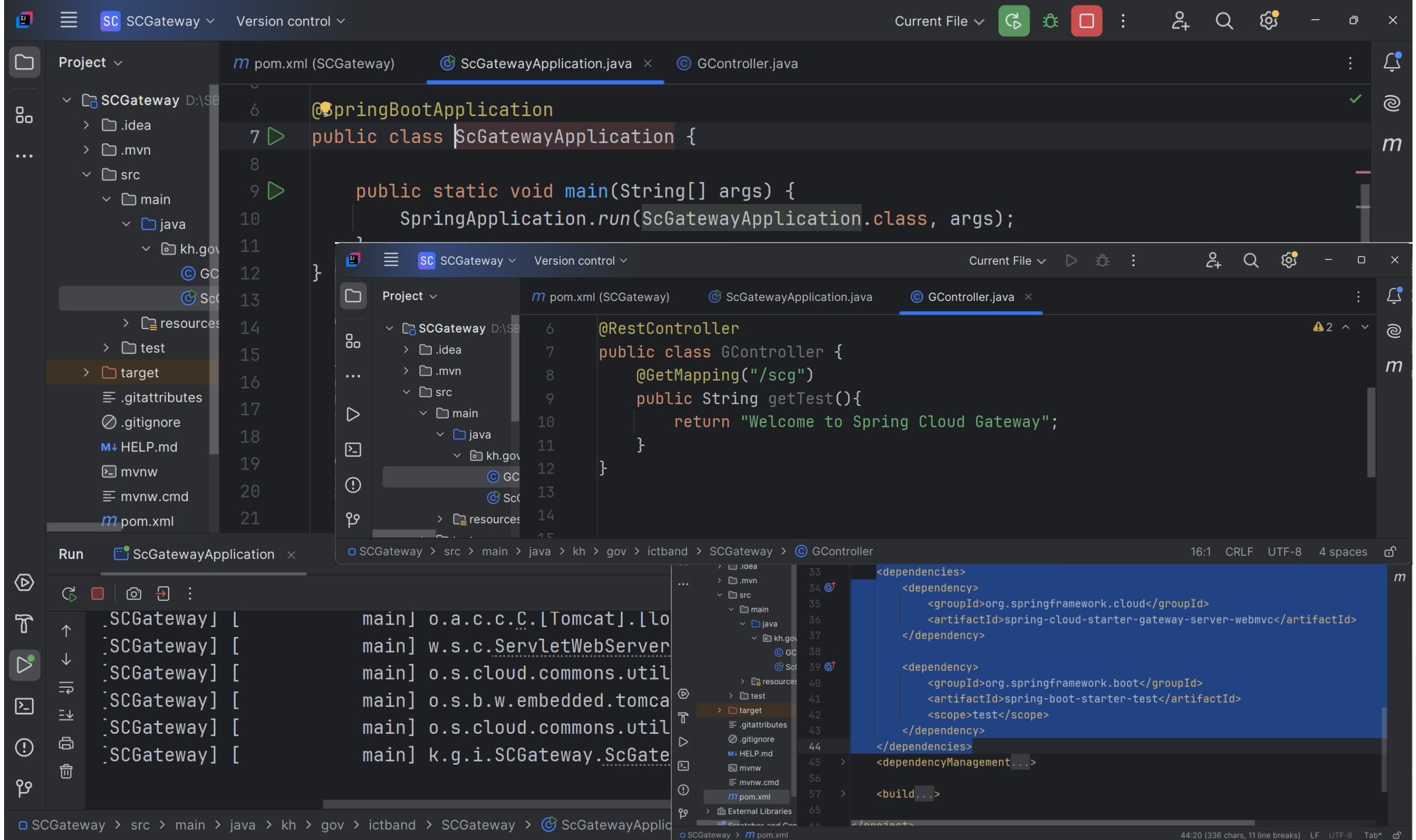
Group	kh.gov.ictband
Artifact	SCGateway
Name	SCGateway
Description	Spring Cloud Gateway for Spring Boot
Package name	kh.gov.ictband.SCGateway
Packaging	● Jar ○ War
Java	○ 24 ● 21 ○ 17

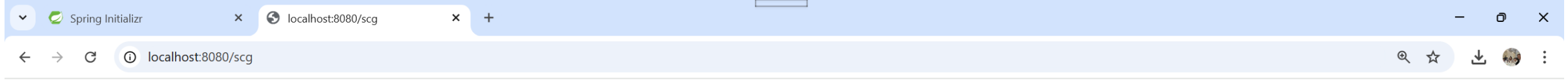
Dependencies — ADD DEPENDENCIES... CTRL + B

Gateway SPRING CLOUD ROUTING

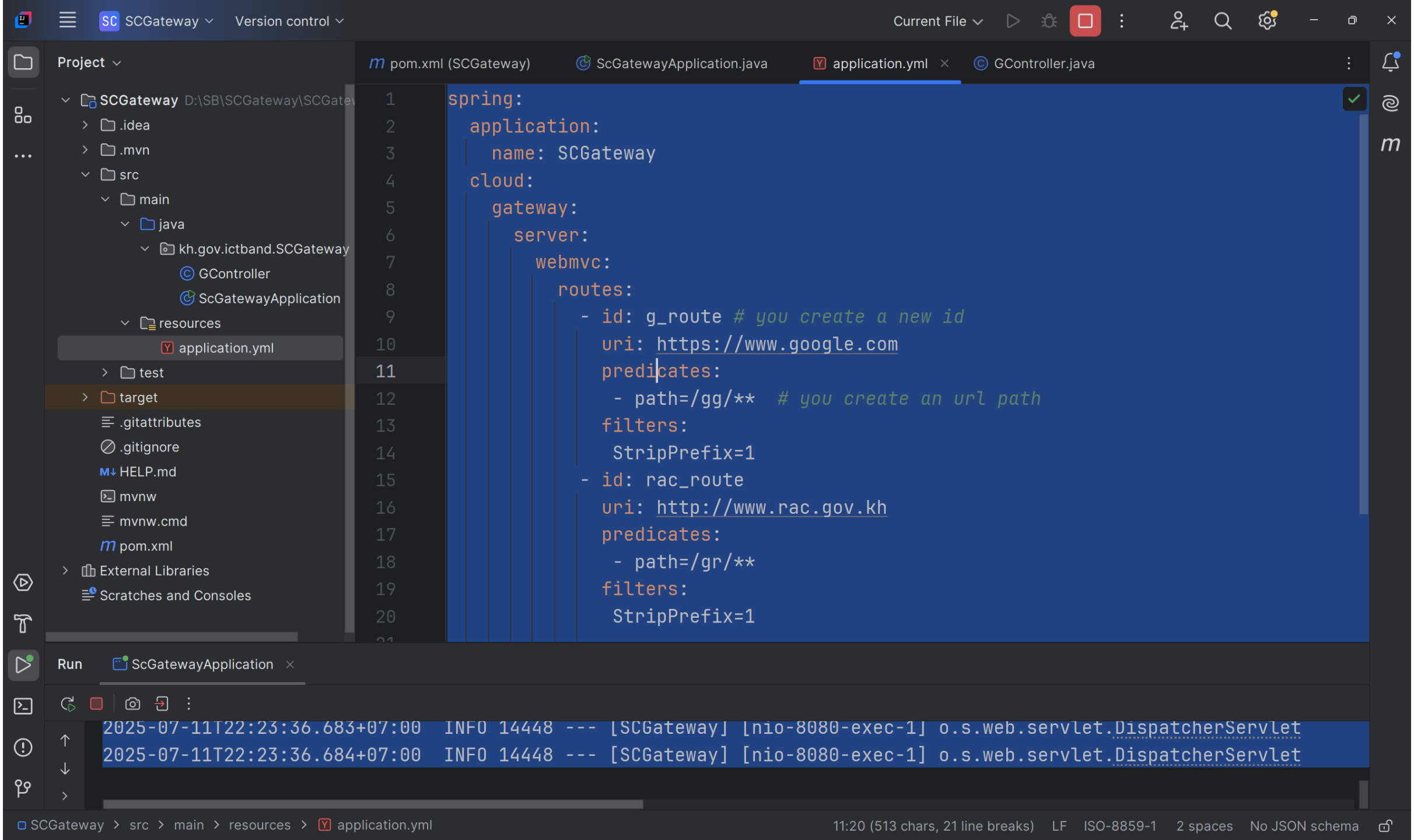
Provides a simple, yet effective way to route to APIs in Servlet-based applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | ...


SCGateway
Version control
Current File
Project
pom.xml (SCGateway)
ScGatewayApplication.java
GController.java
@SpringBootApplication
public class ScGatewayApplication {
public static void main(String[] args) {
SpringApplication.run(ScGatewayApplication.class, args);
}
}
@RestController
public class GController {
@GetMapping("/scg")
public String getTest(){
return "Welcome to Spring Cloud Gateway";
}
}
Run
ScGatewayApplication
SCGateway D:\SB
.idea
.mvn
src
main
java
kh.gov
resources
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SCGateway] [main] o.a.c.c.C.[Tomcat].[lo
SCGateway] [main] w.s.c.ServletWebServer
SCGateway] [main] o.s.cloud.commons.util
SCGateway] [main] o.s.b.w.embedded.tomca
SCGateway] [main] o.s.cloud.commons.util
SCGateway] [main] k.g.i.SCGateway.ScGate
SCGateway > src > main > java > kh > gov > ictband > SCGateway > GController
16:1 CRLF UTF-8 4 spaces
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-gateway-server-webmvc</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<dependencyManagement...>
<build...>
External Libraries
44:20 (336 chars, 11 line breaks) LF UTF-8 Tab*



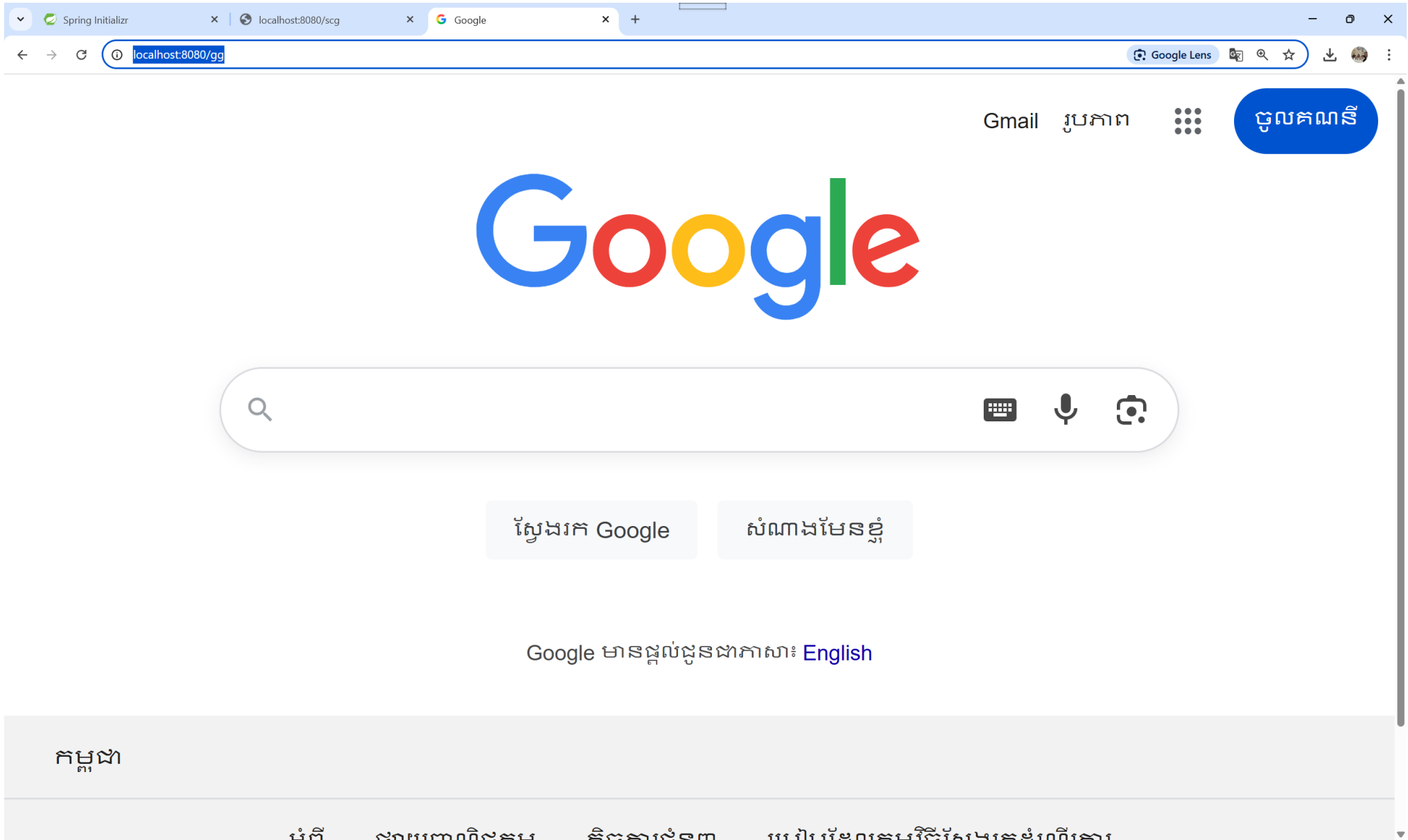
Welcome to Spring Cloud Gateway

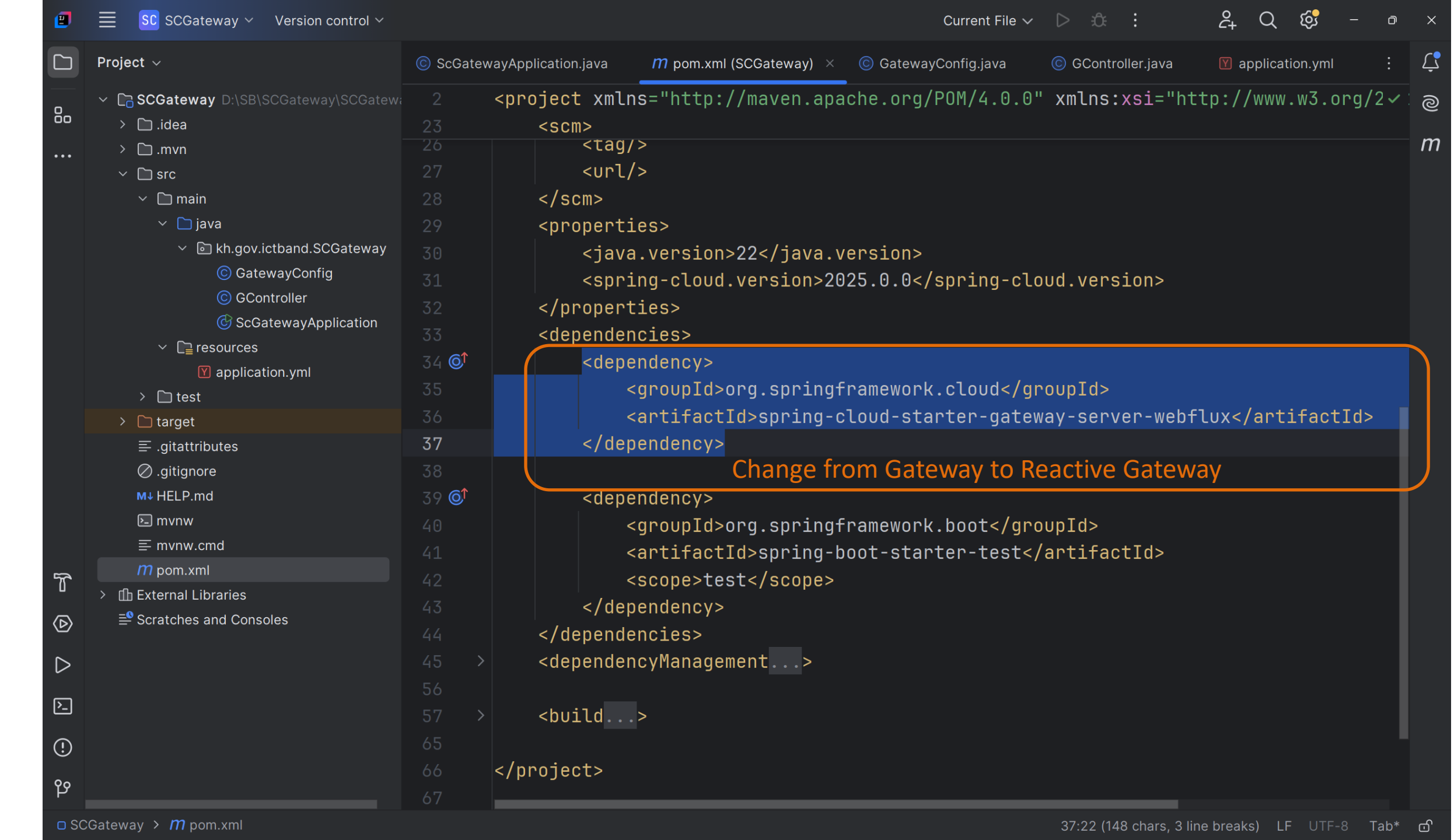

SCGateway
Version control
Current File
Project
SCGateway D:\SB\SCGateway\SCGate
.idea
.mvn
src
main
java
kh.gov.ictband.SCGateway
GController
ScGatewayApplication
resources
application.yml
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
pom.xml (SCGateway)
ScGatewayApplication.java
application.yml
GController.java
spring:
application:
name: SCGateway
cloud:
gateway:
server:
webmvc:
routes:
- id: g_route # you create a new id
uri: https://www.google.com
predicates:
- path=/gg/** # you create an url path
filters:
StripPrefix=1
- id: rac_route
uri: http://www.rac.gov.kh
predicates:
- path=/gr/**
filters:
StripPrefix=1
Run
ScGatewayApplication
2025-07-11T22:23:36.683+07:00 INFO 14448 --- [SCGateway] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2025-07-11T22:23:36.684+07:00 INFO 14448 --- [SCGateway] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
SCGateway > src > main > resources > application.yml
11:20 (513 chars, 21 line breaks)
LF
ISO-8859-1
2 spaces
No JSON schema

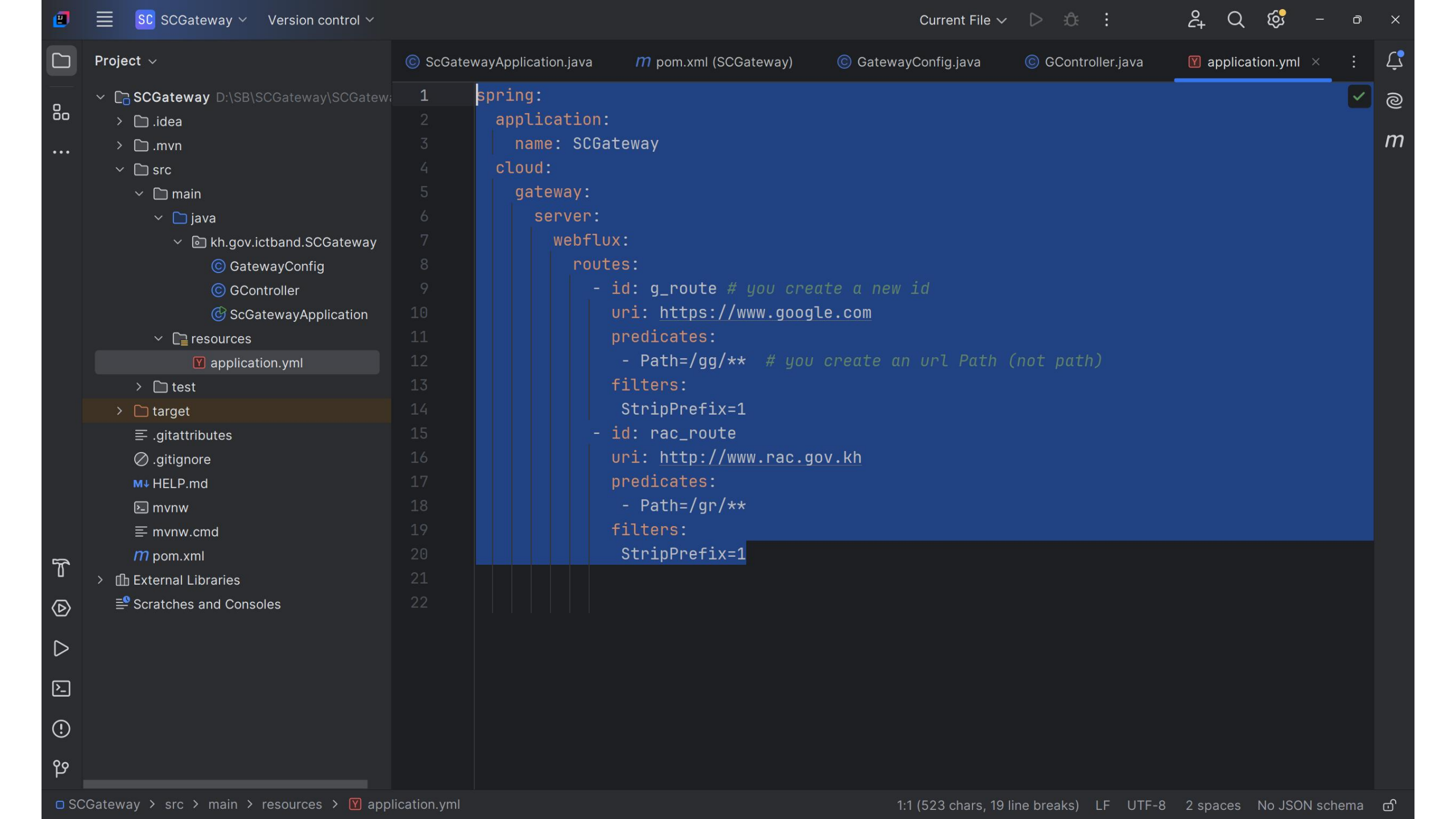
Spring Initializr | localhost:8080/scg

localhost:8080/scg

Welcome to Spring Cloud Gateway



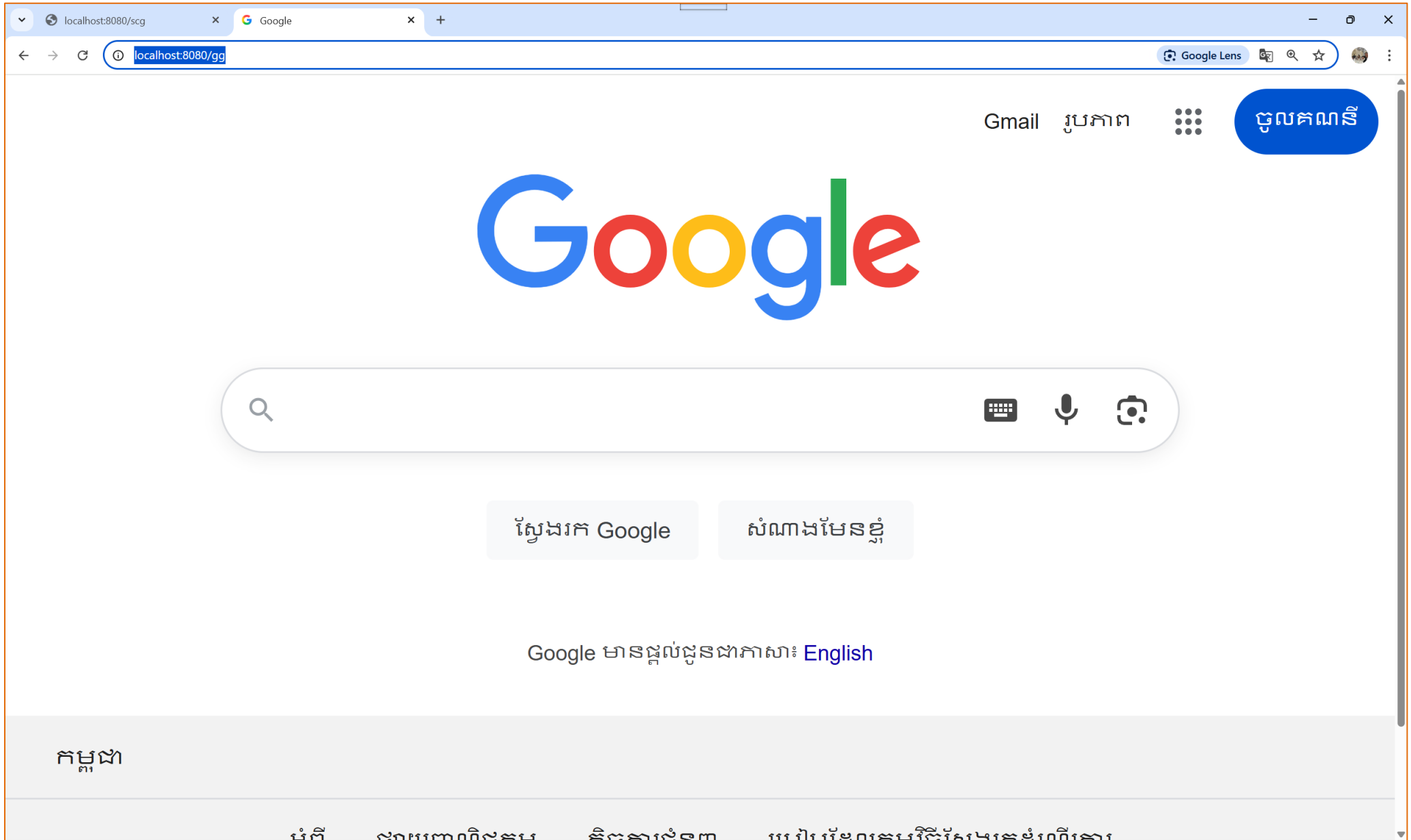

SCGateway
Version control
Current File
Project
SCGateway D:\SB\SCGateway\SCGatew
.idea
.mvn
src
main
java
kh.gov.ictband.SCGateway
GatewayConfig
GController
ScGatewayApplication
resources
application.yml
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
ScGatewayApplication.java
pom.xml (SCGateway)
GatewayConfig.java
GController.java
application.yml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2
<scm>
<tag/>
<url/>
</scm>
<properties>
<java.version>22</java.version>
<spring-cloud.version>2025.0.0</spring-cloud.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-gateway-server-webflux</artifactId>
</dependency>
Change from Gateway to Reactive Gateway
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<dependencyManagement...>
<build...>
</project>
SCGateway > pom.xml
37:22 (148 chars, 3 line breaks)
LF
UTF-8
Tab*


SCGateway
Version control
Current File
Project
SCGateway D:\SB\SCGateway\SCGatew
.idea
.mvn
src
main
java
kh.gov.ictband.SCGateway
GatewayConfig
GController
ScGatewayApplication
resources
application.yml
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
ScGatewayApplication.java
pom.xml (SCGateway)
GatewayConfig.java
GController.java
application.yml
spring:
application:
name: SCGateway
cloud:
gateway:
server:
webflux:
routes:
- id: g_route # you create a new id
uri: https://www.google.com
predicates:
- Path=/gg/** # you create an url Path (not path)
filters:
StripPrefix=1
- id: rac_route
uri: http://www.rac.gov.kh
predicates:
- Path=/gr/**
filters:
StripPrefix=1
SCGateway > src > main > resources > application.yml
1:1 (523 chars, 19 line breaks)
LF
UTF-8
2 spaces
No JSON schema

localhost:8080/scg

localhost:8080/scg

Welcome to Spring Cloud Gateway



SCGateway Version control Current File

Project

SCGateway D:\SB\SCGateway\SCG
- .idea
- .mvn
- src
  - main
    - java
      - kh.gov.ictband.SCGatew
        - GatewayConfig
        - GController
        - ScGatewayApplicatic
    - resources
      - application.yml
  - test
- target
- .gitattributes
- .gitignore
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

External Libraries

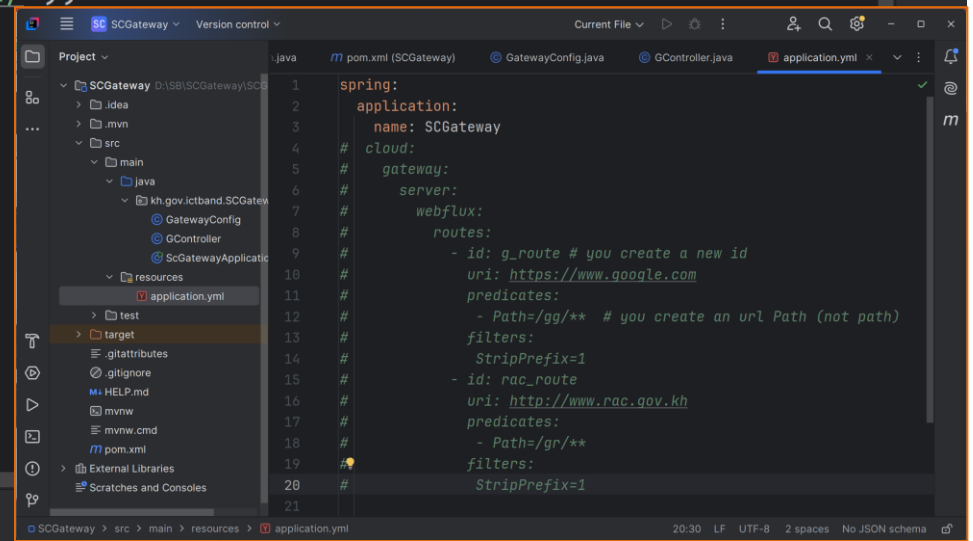
Scratches and Consoles

ScGatewayApplication.java | pom.xml (SCGateway) | GatewayConfig.java | GController.java | application.yml

```java
@Configuration
public class GatewayConfig {

    // if you would like to use RouteLocator,
    // change Gateway to Reactive Gateway dependency
    @Bean
    public RouteLocator routeLocator(RouteLocatorBuilder builder) {
        return builder.routes()
         .route("g_route", r -> r.path("/gg/**")  // "g_route" is id
            .filters(f -> f.stripPrefix(1))      // Remove /gg/ from the path
            .uri("https://www.google.com/"))    // Direct URI for a specific service
         .route("rac_route", r -> r.host("*.rac.gov.kh")
            .and()
            .path("/gr/**")
            .filters(f -> f.stripPrefix(1))
            .uri("http://www.rac.gov.kh/"))
         .build();
    }
}
```

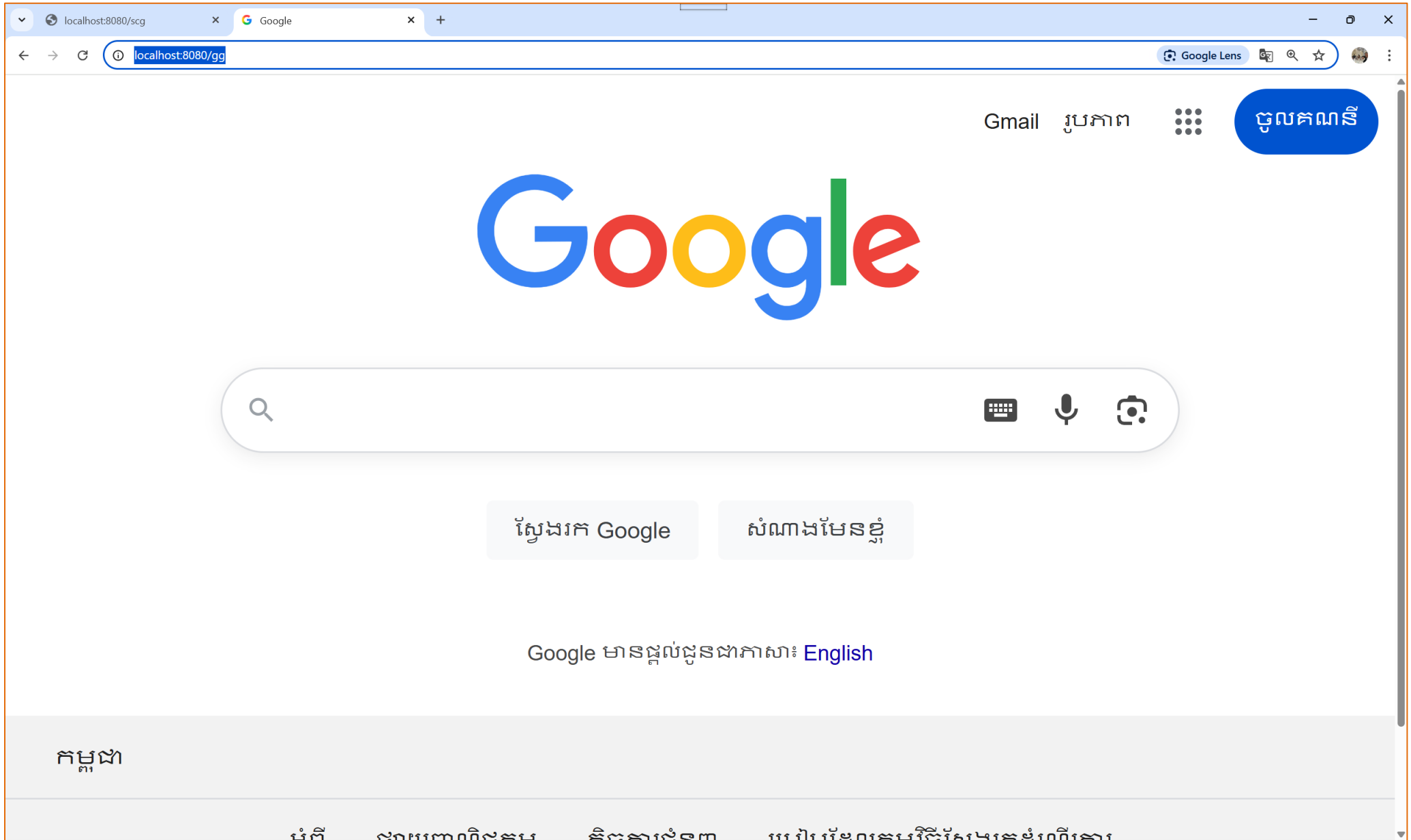
SCGateway > src > main > java > kh > gov > ictband > SCGateway > GatewayConfig > routeLocator



localhost:8080/scg

localhost:8080/scg

Welcome to Spring Cloud Gateway



Spring Cloud Lab 2

Spring cloud Routing

Reactive Gateway + Microservices

Spring Initializr

start.spring.io

spring initializr

Project

- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language

- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot

- ○ 4.1.0 (SNAPSHOT)
- ○ 4.1.0 (M1)
- ○ 4.0.3 (SNAPSHOT)
- ● 4.0.2
- ○ 3.5.11 (SNAPSHOT)
- ○ 3.5.10

Project Metadata

Group	kh.edu.ictband	Artifact	ServiceOne
Name	ServiceOne		
Description	Serivce One for Spring Boot		
Package name	kh.edu.ictband.ServiceOne		

- Packaging: ● Jar ○ War
- Configuration: ● Properties ○ YAML
- Java: ● 25 ○ 21 ○ 17

Dependencies — ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | ...

Spring Initializr

start.spring.io

spring initializr

Project

- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language

- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot

- ○ 4.1.0 (SNAPSHOT)
- ○ 4.1.0 (M1)
- ○ 4.0.3 (SNAPSHOT)
- ● 4.0.2
- ○ 3.5.11 (SNAPSHOT)
- ○ 3.5.10

Project Metadata

Group	kh.edu.ictband
Artifact	ServiceTwo
Name	ServiceTwo
Description	Serivce Two for Spring Boot
Package name	kh.edu.ictband.ServiceTwo
Packaging	● Jar ○ War
Configuration	● Properties ○ YAML
Java	● 25 ○ 21 ○ 17

Dependencies — ADD DEPENDENCIES... CTRL + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | ...

Spring Initializr

start.spring.io

spring initializr

Project

- ○ Gradle - Groovy
- ○ Gradle - Kotlin
- ● Maven

Language

- ● Java
- ○ Kotlin
- ○ Groovy

Spring Boot

- ○ 4.1.0 (SNAPSHOT)
- ○ 4.1.0 (M1)
- ○ 4.0.3 (SNAPSHOT)
- ● 4.0.2
- ○ 3.5.11 (SNAPSHOT)
- ○ 3.5.10

Project Metadata

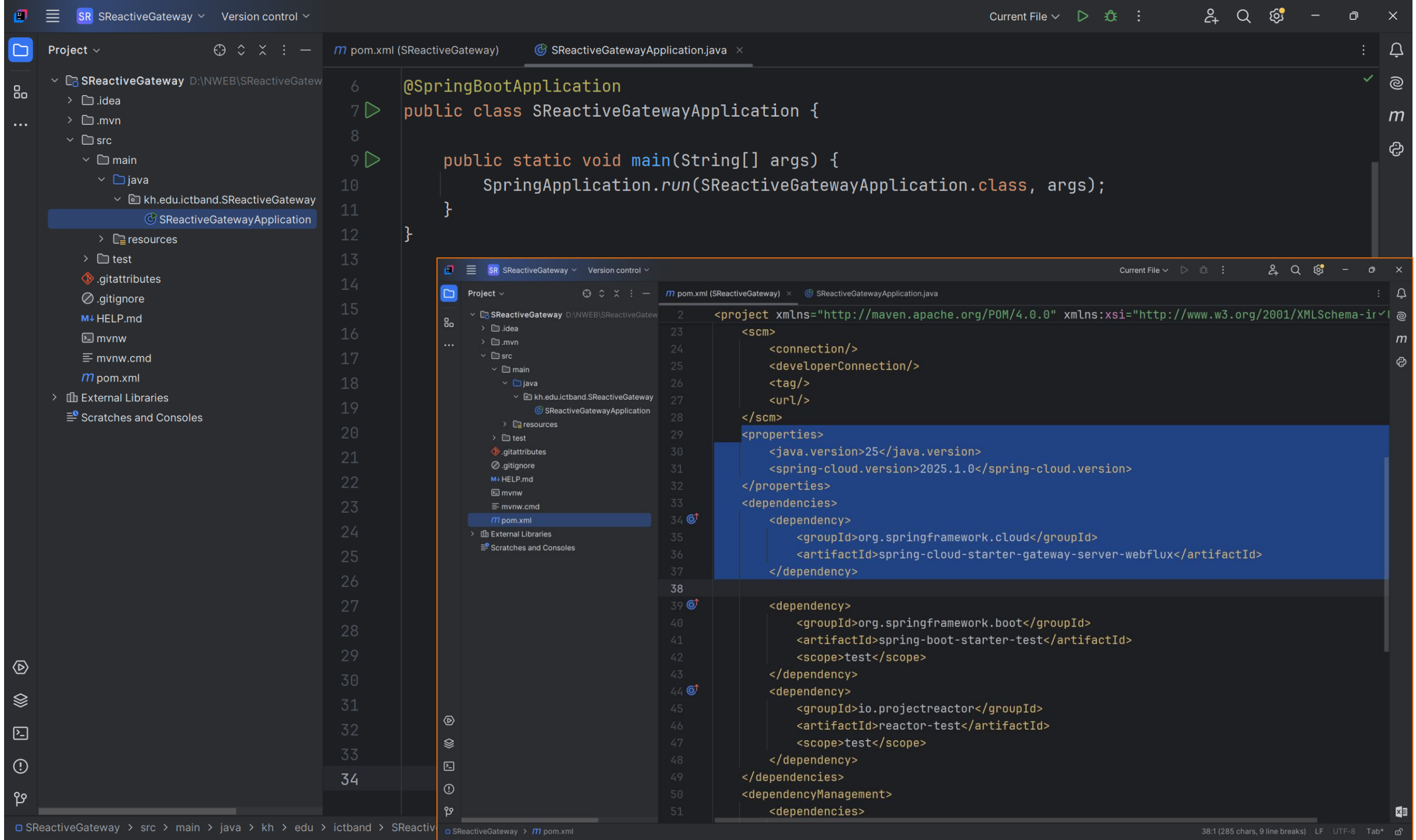
Group	kh.edu.ictband
Artifact	SReactiveGateway
Name	SReactiveGateway
Description	Spring Cloud Reactive Gateway for Spring Boot
Package name	kh.edu.ictband.SReactiveGateway
Packaging	● Jar ○ War
Configuration	● Properties ○ YAML
Java	● 25 ○ 21 ○ 17

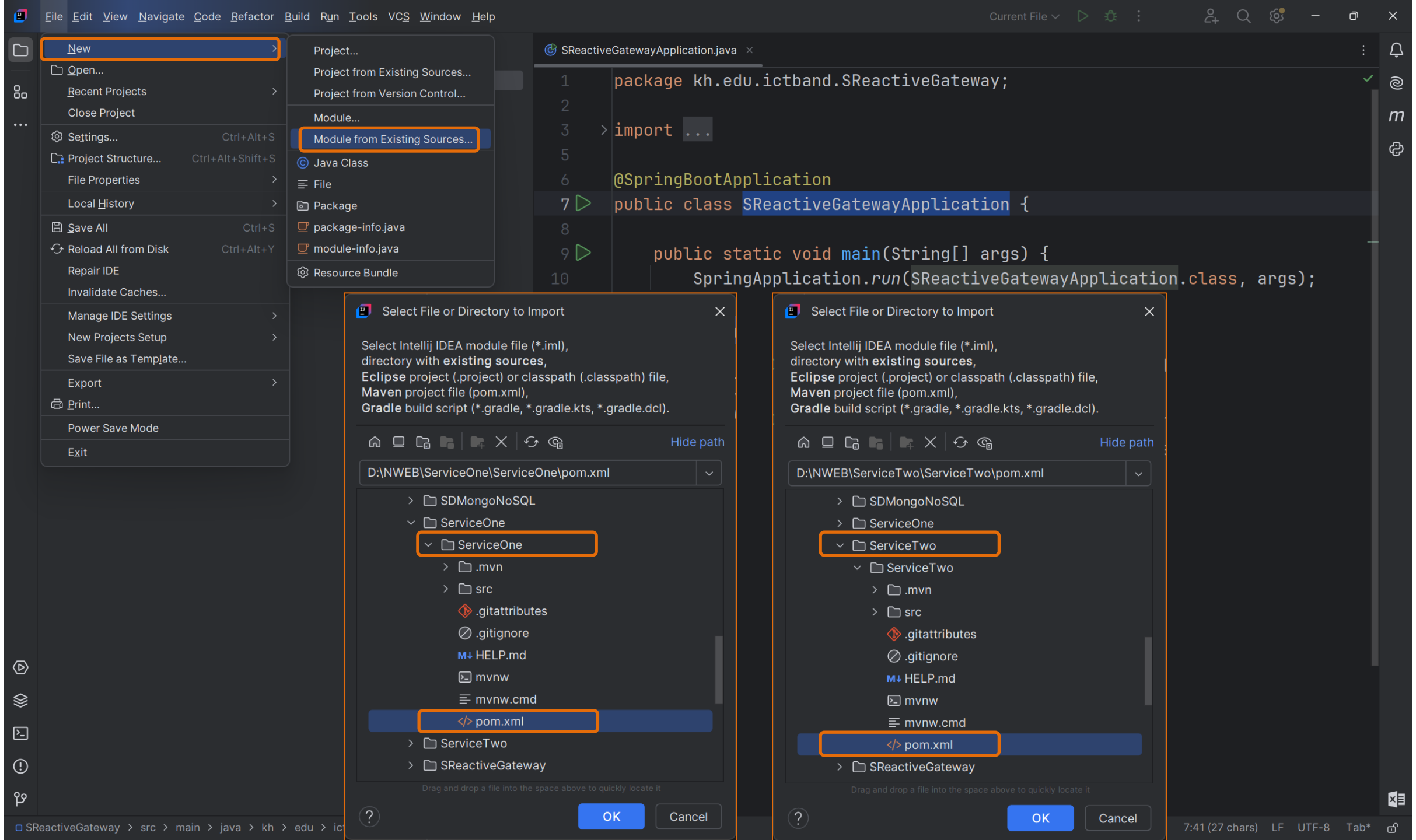
Dependencies — ADD DEPENDENCIES... CTRL + B

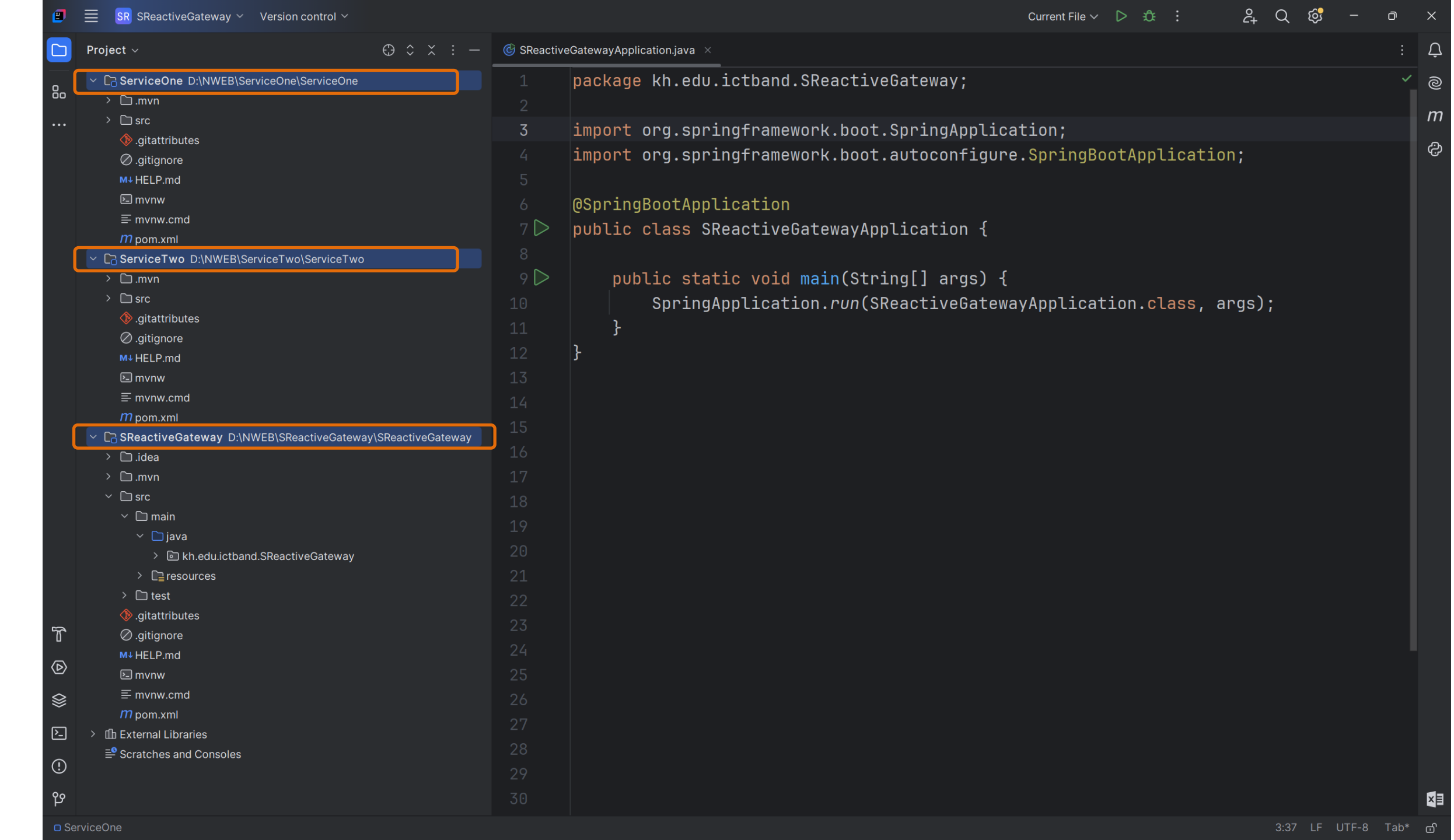
Reactive Gateway SPRING CLOUD ROUTING

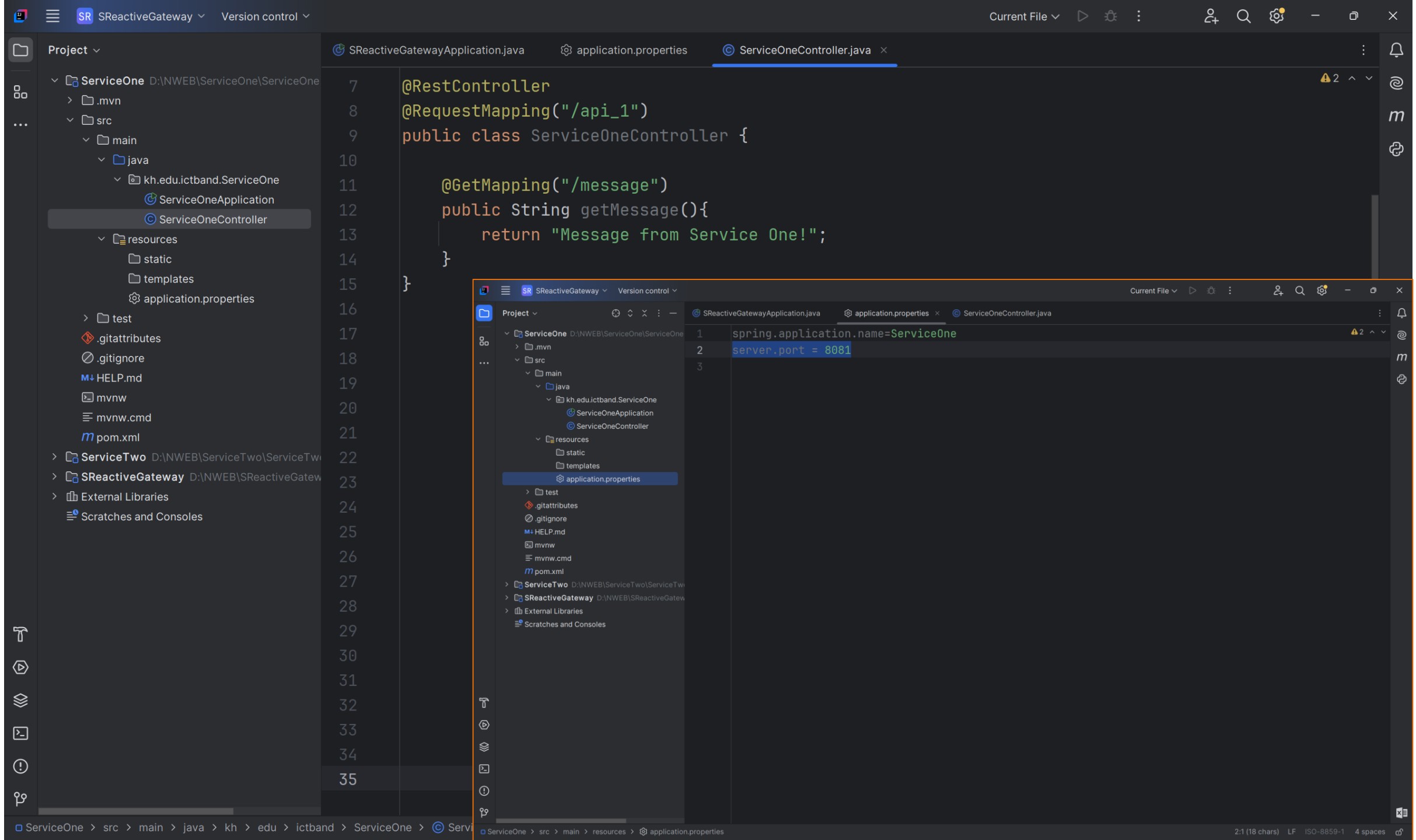
Provides a simple, yet effective way to route to APIs in reactive applications. Provides cross-cutting concerns to those APIs such as security, monitoring/metrics, and resiliency.

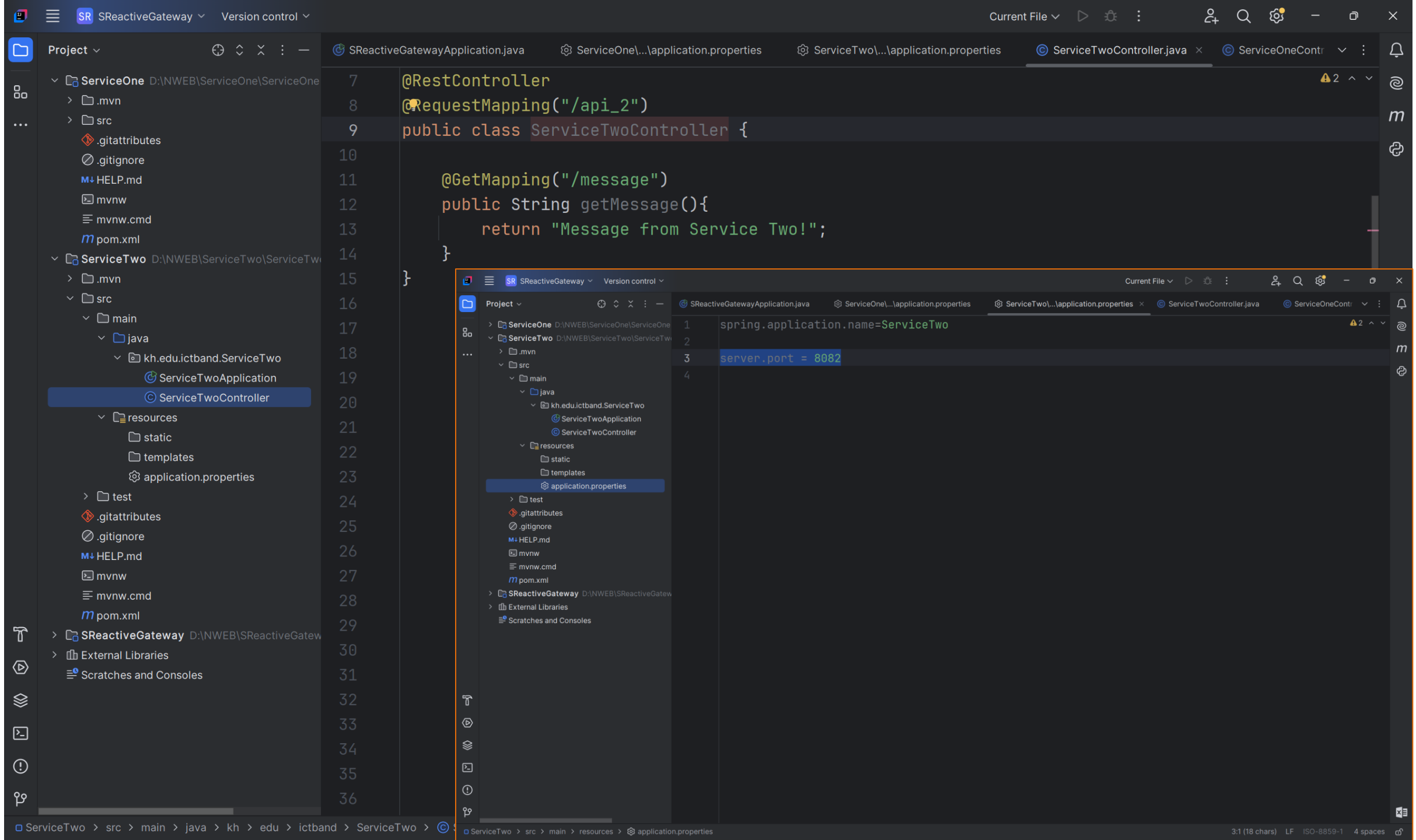
GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | ...

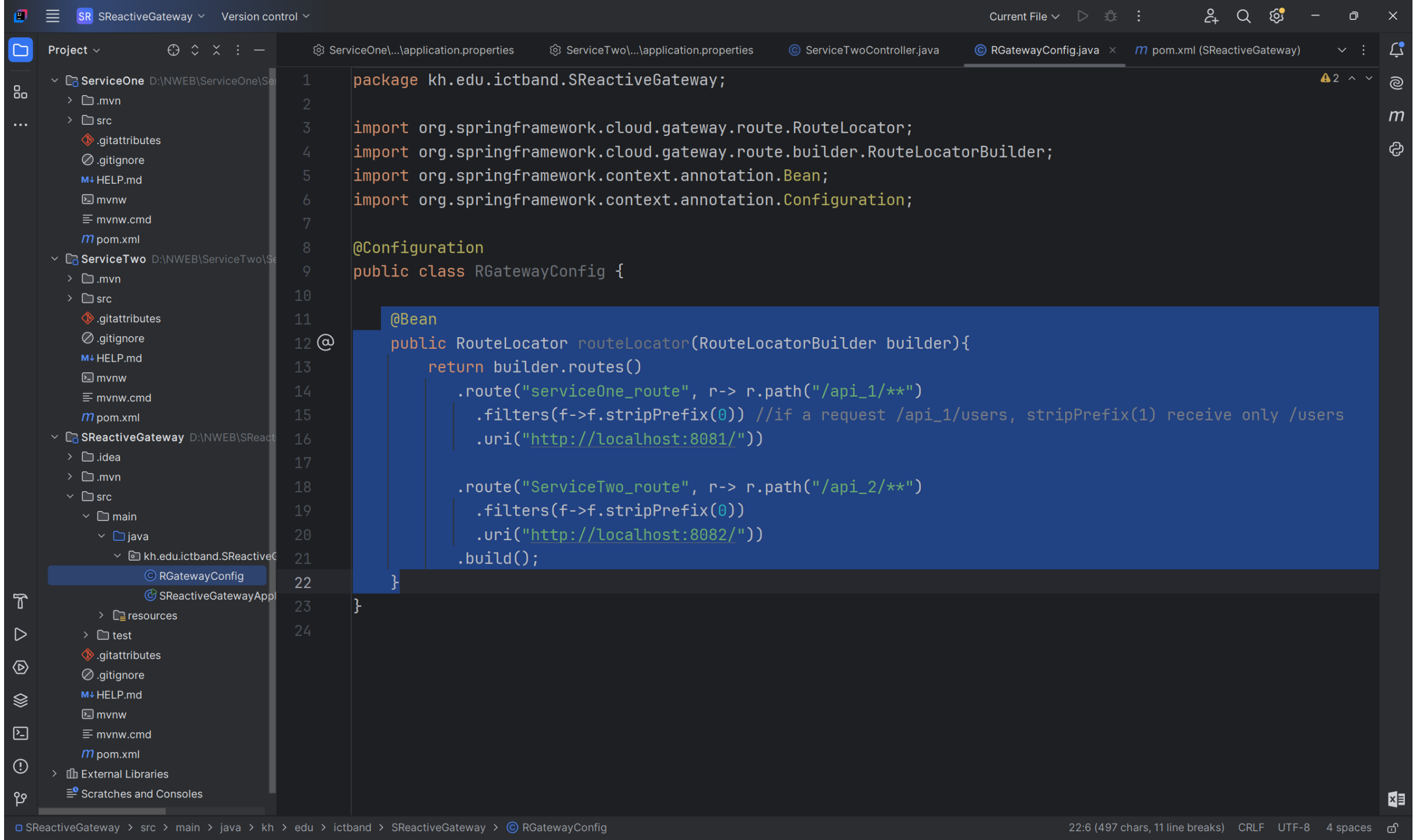

SReactiveGateway
Version control
Current File
Project
SReactiveGateway D:\NWEB\SReactiveGatew
.idea
.mvn
src
main
java
kh.edu.ictband.SReactiveGateway
SReactiveGatewayApplication
resources
test
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
pom.xml (SReactiveGateway)
SReactiveGatewayApplication.java
@SpringBootApplication
public class SReactiveGatewayApplication {
public static void main(String[] args) {
SpringApplication.run(SReactiveGatewayApplication.class, args);
}
}
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-ir
<scm>
<connection/>
<developerConnection/>
<tag/>
<url/>
</scm>
<properties>
<java.version>25</java.version>
<spring-cloud.version>2025.1.0</spring-cloud.version>
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-gateway-server-webflux</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
<dependency>
<groupId>io.projectreactor</groupId>
<artifactId>reactor-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<dependencyManagement>
<dependencies>
SReactiveGateway › src › main › java › kh › edu › ictband › SReactiv
SReactiveGateway › pom.xml
38:1 (285 chars, 9 line breaks) LF UTF-8 Tab*

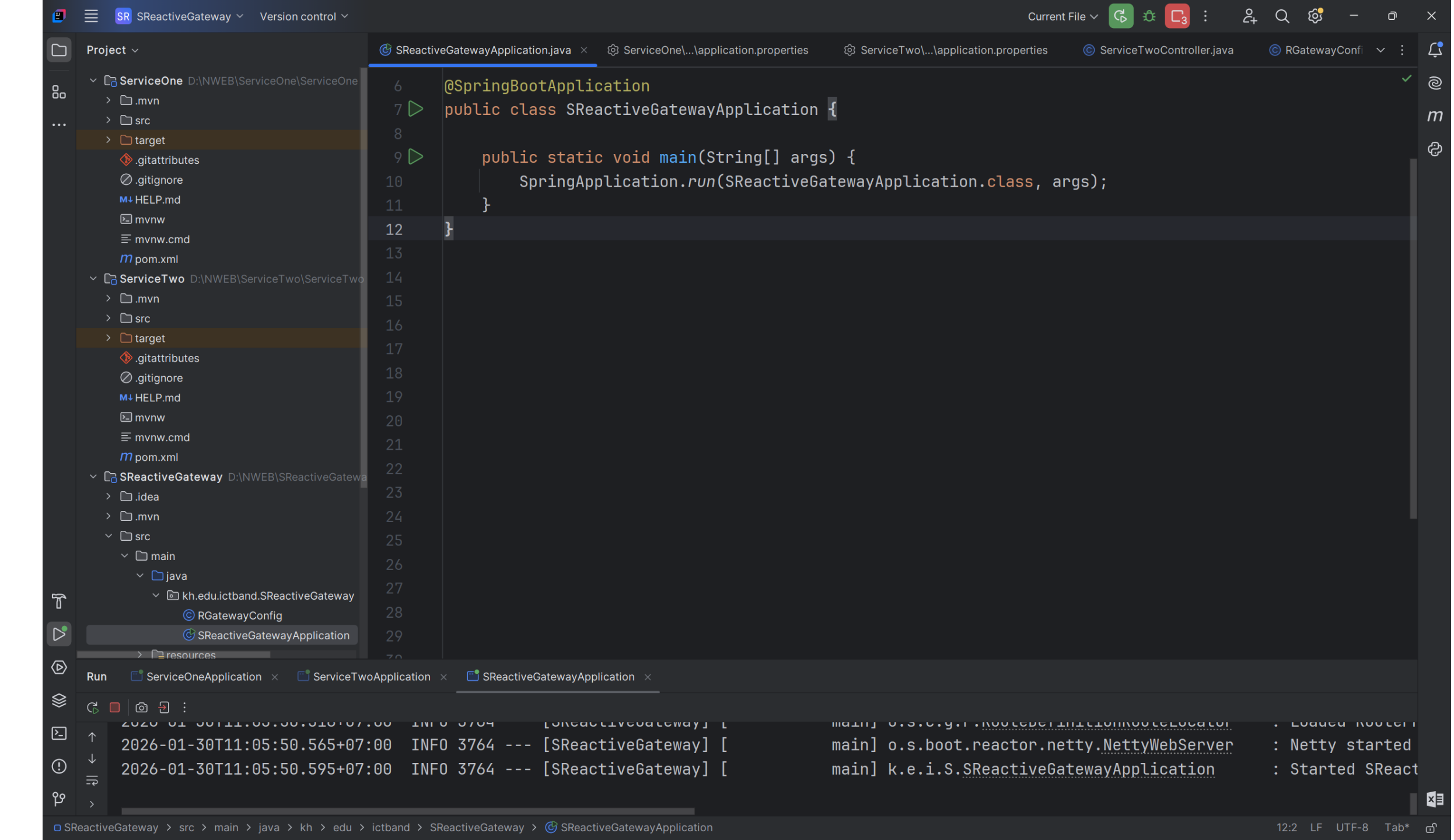

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
New
Open...
Recent Projects
Close Project
Settings... Ctrl+Alt+S
Project Structure... Ctrl+Alt+Shift+S
File Properties
Local History
Save All Ctrl+S
Reload All from Disk Ctrl+Alt+Y
Repair IDE
Invalidate Caches...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Exit
Project...
Project from Existing Sources...
Project from Version Control...
Module...
Module from Existing Sources...
Java Class
File
Package
package-info.java
module-info.java
Resource Bundle
SReactiveGatewayApplication.java
package kh.edu.ictband.SReactiveGateway;
import ...
@SpringBootApplication
public class SReactiveGatewayApplication {
public static void main(String[] args) {
SpringApplication.run(SReactiveGatewayApplication.class, args);
Select File or Directory to Import
Select Intellij IDEA module file (*.iml),
directory with existing sources,
Eclipse project (.project) or classpath (.classpath) file,
Maven project file (pom.xml),
Gradle build script (*.gradle, *.gradle.kts, *.gradle.dcl).
Hide path
D:\NWEB\ServiceOne\ServiceOne\pom.xml
SDMongoNoSQL
ServiceOne
ServiceOne
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo
SReactiveGateway
Drag and drop a file into the space above to quickly locate it
OK
Cancel
Select File or Directory to Import
Select Intellij IDEA module file (*.iml),
directory with existing sources,
Eclipse project (.project) or classpath (.classpath) file,
Maven project file (pom.xml),
Gradle build script (*.gradle, *.gradle.kts, *.gradle.dcl).
Hide path
D:\NWEB\ServiceTwo\ServiceTwo\pom.xml
SDMongoNoSQL
ServiceOne
ServiceTwo
ServiceTwo
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SReactiveGateway
Drag and drop a file into the space above to quickly locate it
OK
Cancel
SReactiveGateway › src › main › java › kh › edu
7:41 (27 chars) LF UTF-8 Tab*

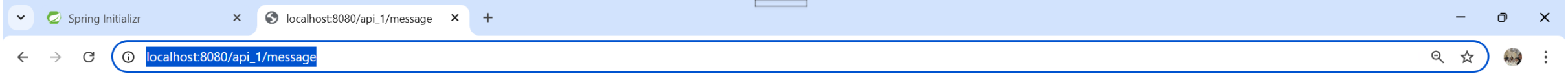

SReactiveGateway
Version control
Current File
Project
ServiceOne D:\NWEB\ServiceOne\ServiceOne
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo D:\NWEB\ServiceTwo\ServiceTwo
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SReactiveGateway D:\NWEB\SReactiveGateway\SReactiveGateway
.idea
.mvn
src
main
java
kh.edu.ictband.SReactiveGateway
resources
test
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
SReactiveGatewayApplication.java
1 package kh.edu.ictband.SReactiveGateway;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class SReactiveGatewayApplication {
8
9 public static void main(String[] args) {
10 SpringApplication.run(SReactiveGatewayApplication.class, args);
11 }
12 }
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
ServiceOne
3:37 LF UTF-8 Tab*


SReactiveGateway
Version control
Current File
Project
ServiceOne D:\NWEB\ServiceOne\ServiceOne
.mvn
src
main
java
kh.edu.ictband.ServiceOne
ServiceOneApplication
ServiceOneController
resources
static
templates
application.properties
test
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo D:\NWEB\ServiceTwo\ServiceTwo
SReactiveGateway D:\NWEB\SReactiveGatew
External Libraries
Scratches and Consoles
SReactiveGatewayApplication.java
application.properties
ServiceOneController.java
@RestController
@RequestMapping("/api_1")
public class ServiceOneController {
@GetMapping("/message")
public String getMessage(){
return "Message from Service One!";
}
}
spring.application.name=ServiceOne
server.port = 8081
ServiceOne › src › main › java › kh › edu › ictband › ServiceOne
ServiceOne › src › main › resources › application.properties

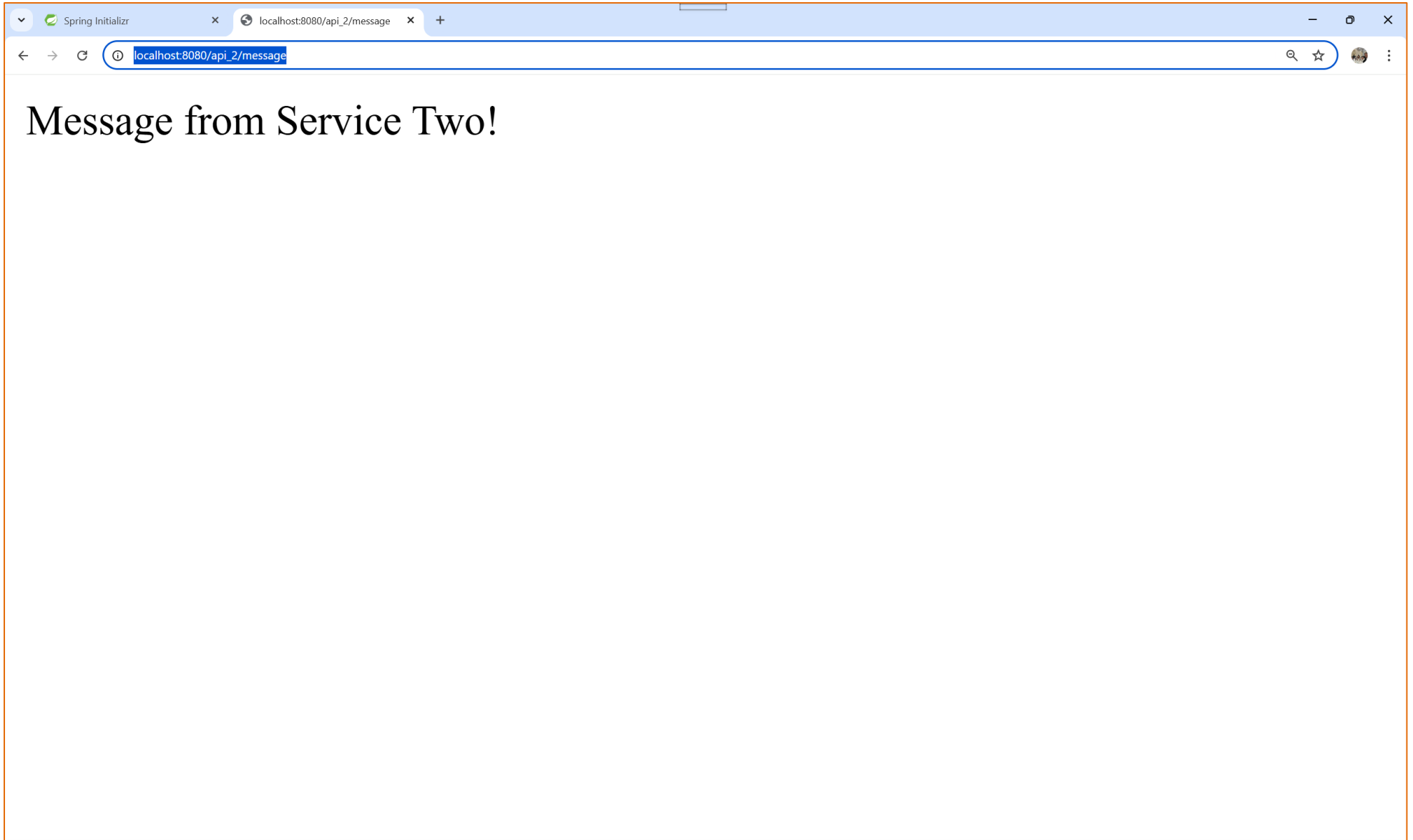

SReactiveGateway
Version control
Current File
Project
ServiceOne D:\NWEB\ServiceOne\ServiceOne
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo D:\NWEB\ServiceTwo\ServiceTwo
.mvn
src
main
java
kh.edu.ictband.ServiceTwo
ServiceTwoApplication
ServiceTwoController
resources
static
templates
application.properties
test
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SReactiveGateway D:\NWEB\SReactiveGatew
External Libraries
Scratches and Consoles
SReactiveGatewayApplication.java
ServiceOne\...\application.properties
ServiceTwo\...\application.properties
ServiceTwoController.java
ServiceOneContr
@RestController
@RequestMapping("/api_2")
public class ServiceTwoController {
@GetMapping("/message")
public String getMessage(){
return "Message from Service Two!";
}
}
spring.application.name=ServiceTwo
server.port = 8082
ServiceTwo › src › main › java › kh › edu › ictband › ServiceTwo ›
ServiceTwo › src › main › resources › application.properties
3:1 (18 chars)
LF
ISO-8859-1
4 spaces


SReactiveGateway
Version control
Current File
Project
ServiceOne D:\NWEB\ServiceOne\Se
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo D:\NWEB\ServiceTwo\Se
.mvn
src
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SReactiveGateway D:\NWEB\SReact
.idea
.mvn
src
main
java
kh.edu.ictband.SReactiveG
RGatewayConfig
SReactiveGatewayAppl
resources
test
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
ServiceOne\...\application.properties
ServiceTwo\...\application.properties
ServiceTwoController.java
RGatewayConfig.java
pom.xml (SReactiveGateway)
package kh.edu.ictband.SReactiveGateway;
import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class RGatewayConfig {
@Bean
public RouteLocator routeLocator(RouteLocatorBuilder builder){
return builder.routes()
.route("serviceOne_route", r-> r.path("/api_1/**")
.filters(f->f.stripPrefix(0)) //if a request /api_1/users, stripPrefix(1) receive only /users
.uri("http://localhost:8081/"))
.route("ServiceTwo_route", r-> r.path("/api_2/**")
.filters(f->f.stripPrefix(0))
.uri("http://localhost:8082/"))
.build();
}
}
SReactiveGateway › src › main › java › kh › edu › ictband › SReactiveGateway › RGatewayConfig
22:6 (497 chars, 11 line breaks)
CRLF
UTF-8
4 spaces


SReactiveGateway
Version control
Current File
Project
ServiceOne D:\NWEB\ServiceOne\ServiceOne
.mvn
src
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
ServiceTwo D:\NWEB\ServiceTwo\ServiceTwo
.mvn
src
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
SReactiveGateway D:\NWEB\SReactiveGatewa
.idea
.mvn
src
main
java
kh.edu.ictband.SReactiveGateway
RGatewayConfig
SReactiveGatewayApplication
resources
SReactiveGatewayApplication.java
ServiceOne\...\application.properties
ServiceTwo\...\application.properties
ServiceTwoController.java
RGatewayConfi
@SpringBootApplication
public class SReactiveGatewayApplication {
public static void main(String[] args) {
SpringApplication.run(SReactiveGatewayApplication.class, args);
}
}
Run
ServiceOneApplication
ServiceTwoApplication
SReactiveGatewayApplication
2026-01-30T11:05:50.565+07:00 INFO 3764 --- [SReactiveGateway] [main] o.s.boot.reactor.netty.NettyWebServer : Netty started
2026-01-30T11:05:50.595+07:00 INFO 3764 --- [SReactiveGateway] [main] k.e.i.S.SReactiveGatewayApplication : Started SReact
SReactiveGateway › src › main › java › kh › edu › ictband › SReactiveGateway › SReactiveGatewayApplication
12:2 LF UTF-8 Tab*



Message from Service One!

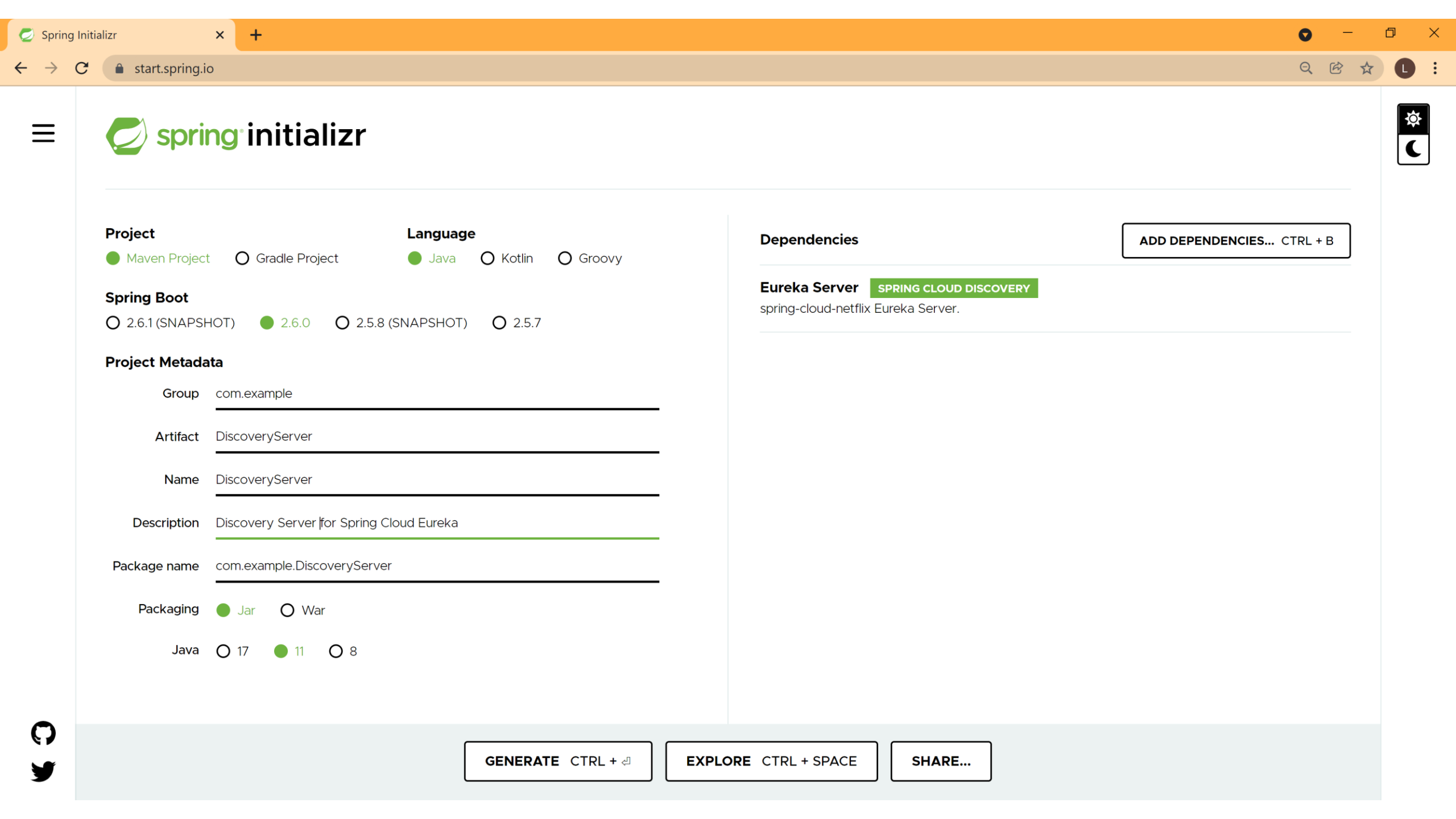


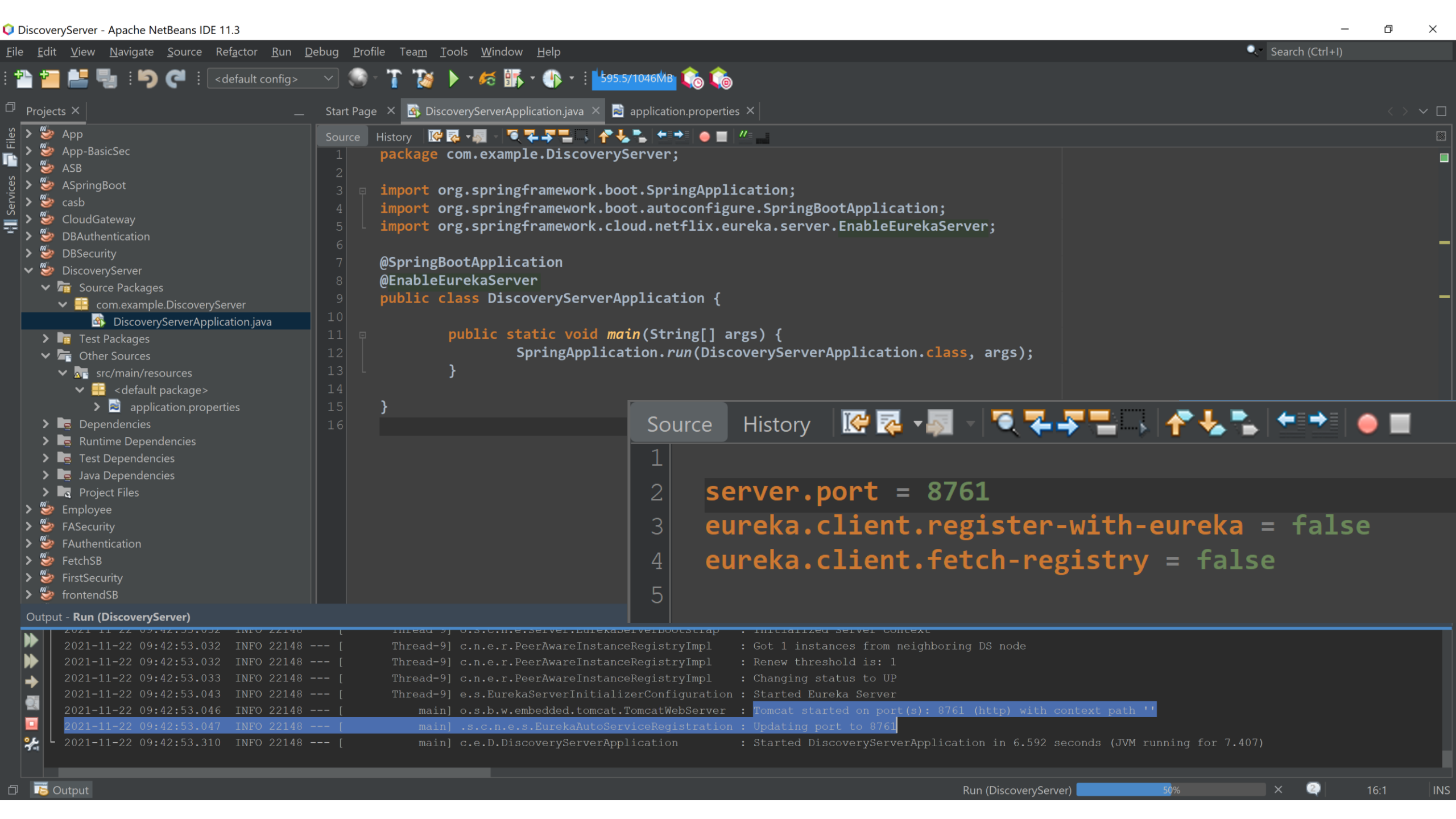
Message from Service Two!

Spring Cloud Lab 5

Spring Cloud Discovery

Eureka Server (Deprecated)


Spring Initializr
start.spring.io
spring initializr
Project
Maven Project
Gradle Project
Language
Java
Kotlin
Groovy
Spring Boot
2.6.1 (SNAPSHOT)
2.6.0
2.5.8 (SNAPSHOT)
2.5.7
Project Metadata
Group
com.example
Artifact
DiscoveryServer
Name
DiscoveryServer
Description
Discovery Server for Spring Cloud Eureka
Package name
com.example.DiscoveryServer
Packaging
Jar
War
Java
17
11
8
Dependencies
ADD DEPENDENCIES... CTRL + B
Eureka Server
SPRING CLOUD DISCOVERY
spring-cloud-netflix Eureka Server.
GENERATE CTRL + ⏎
EXPLORE CTRL + SPACE
SHARE...


DiscoveryServer - Apache NetBeans IDE 11.3
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
595.5/1046MB
Projects
Start Page
DiscoveryServerApplication.java
application.properties
App
App-BasicSec
ASB
ASpringBoot
casb
CloudGateway
DBAuthentication
DBSecurity
DiscoveryServer
Source Packages
com.example.DiscoveryServer
DiscoveryServerApplication.java
Test Packages
Other Sources
src/main/resources
<default package>
application.properties
Dependencies
Runtime Dependencies
Test Dependencies
Java Dependencies
Project Files
Employee
FASecurity
FAuthentication
FetchSB
FirstSecurity
frontendSB
Source History
1 package com.example.DiscoveryServer;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7 @SpringBootApplication
8 @EnableEurekaServer
9 public class DiscoveryServerApplication {
10
11 public static void main(String[] args) {
12 SpringApplication.run(DiscoveryServerApplication.class, args);
13 }
14
15 }
16
Source History
1
2 server.port = 8761
3 eureka.client.register-with-eureka = false
4 eureka.client.fetch-registry = false
5
Output - Run (DiscoveryServer)
2021-11-22 09:42:53.032 INFO 22148 --- [Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Got 1 instances from neighboring DS node
2021-11-22 09:42:53.032 INFO 22148 --- [Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Renew threshold is: 1
2021-11-22 09:42:53.033 INFO 22148 --- [Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Changing status to UP
2021-11-22 09:42:53.043 INFO 22148 --- [Thread-9] e.s.EurekaServerInitializerConfiguration : Started Eureka Server
2021-11-22 09:42:53.046 INFO 22148 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8761 (http) with context path ''
2021-11-22 09:42:53.047 INFO 22148 --- [main] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8761
2021-11-22 09:42:53.310 INFO 22148 --- [main] c.e.D.DiscoveryServerApplication : Started DiscoveryServerApplication in 6.592 seconds (JVM running for 7.407)
Output
Run (DiscoveryServer)
50%
16:1
INS

ServiceOneSB - Apache NetBeans IDE 11.3

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

854.1/1051MB

Projects

- DiscoveryServer
  - Source Packages
    - com.example.DiscoveryServer
      - DiscoveryServerApplication.java
  - Test Packages
  - Other Sources
  - Dependencies
  - Runtime Dependencies
  - Test Dependencies
  - Java Dependencies
  - Project Files
- Employee
- FASecurity
- FAuthentication
- FetchSB
- FirstSecurity
- frontendSB
- jwtSB
- JWTSS
- LatinToKhmer
- SBBA
- SBHTTPS
- ServiceOneSB
  - Source Packages
    - com.example.ServiceOneSB
      - ServiceOneController.java
      - ServiceOneSbApplication.java
  - Test Packages

Start Page | ServiceOneSbApplication.java | application.properties

Source History

```java
package com.example.ServiceOneSB;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class ServiceOneSbApplication {

	public static void main(String[] args) {
		SpringApplication.run(ServiceOneSbApplication.class, args);
	}
}
```

Source History

```
spring.application.name = service-One
# for multiple instances, port is set to 0
server.port = 0
eureka.instance.instance-id = ${spring.application.name}:${random.uuid}
spring.cloud.compatibility-verifier.enabled=false
```

Output

Run (DiscoveryServerApplication) | Run (ServiceOneSB)

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  <version>3.0.4</version>
</dependency>
```

```java
@RestController
@RequestMapping("/api1")
public class ServiceOneController {

    @GetMapping("/message")
    public String Message(){
        return "Message from Service One!";
    }
}
```

Output

ServiceTwoSB - Apache NetBeans IDE 11.3

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config>

730.9/1194MB

Projects

- FirstSecurity
- frontendSB
- jwtSB
- JWTSS
- LatinToKhmer
- SBBA
- SBHTTPS
- ServiceOneSB
  - Source Packages
  - Test Packages
  - Other Sources
  - Dependencies
  - Runtime Dependencies
  - Test Dependencies
  - Java Dependencies
  - Project Files
    - pom.xml
    - nbactions.xml
- ServiceTwoSB
  - Source Packages
    - com.example.ServiceTwoSB
      - ServiceTwoController.java
      - ServiceTwoSbApplication.java
  - Test Packages
  - Other Sources
    - src/main/resources
      - <default package>
        - application.properties
        - static
        - templates
  - Dependencies

...va | application.properties | application.properties | pom.xml [ServiceOneSB] | pom.xml [ServiceTwoSB] | ServiceTwoSbApplication.java

Source History

```java
package com.example.ServiceTwoSB;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class ServiceTwoSbApplication {

        public static void main(String[] args) {
                SpringApplication.run(ServiceTwoSbApplication.class, args);
        }

}
```

Source History

```
spring.application.name = service-Two
server.port = 0
eureka.instance.instance-id = ${spring.application.name}:${random.uuid}
spring.cloud.compatibility-verifier.enabled=false
```

Source History

```java
@RestController
@RequestMapping("/api2")
public class ServiceTwoController {

    @GetMapping("/message")
    public String Message(){
        return "Message from Service Two!";
    }
}
```

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  <version>3.0.4</version>
</dependency>
```

DiscoveryServer - Apache NetBeans IDE 11.3

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

<default config> 619.1/1265MB

Projects

- ASB
- ASpringBoot
- casb
- CloudGateway
- DBAuthentication
- DBSecurity
- DiscoveryServer
  - Source Packages
    - com.example.DiscoveryServer
      - DiscoveryServerApplication.java
  - Test Packages
  - Other Sources
  - Dependencies
  - Runtime Dependencies
  - Test Dependencies
  - Java Dependencies
  - Project Files
- Employee
- FASecurity
- FAuthentication
- FetchSB
- FirstSecurity
- frontendSB
- jwtSB
- JWTSS
- LatinToKhmer
- SBBA
- SBHTTPS

...es | pom.xml [ServiceOneSB] | pom.xml [ServiceTwoSB] | ServiceTwoSbApplication.java | ServiceTwoController.java | DiscoveryServerApplication.java

Source History

```
1  package com.example.DiscoveryServer;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
6
7  @SpringBootApplication
8  @EnableEurekaServer
9  public class DiscoveryServerApplication {
10
11         public static void main(String[] args) {
12                 SpringApplication.run(DiscoveryServerApplication.class, args);
13         }
14
15 }
16
```

Output

Run (DiscoveryServerApplication) | Run (ServiceOneSB) | Run (ServiceTwoSbApplication)

```
2021-11-22 15:00:33.269  INFO 29088 --- [           main] com.netflix.discovery.DiscoveryClient    : Saw local status change event StatusChangeEvent [timestamp=1637568033269, current=UP, previous=STARTI
2021-11-22 15:00:33.271  INFO 29088 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_SERVICE-TWO/service-Two:f2f01054-ac1b-4d8f-8b5c-0f7455079435: registering service...
2021-11-22 15:00:33.322  INFO 29088 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_SERVICE-TWO/service-Two:f2f01054-ac1b-4d8f-8b5c-0f7455079435 - registration status: 2
2021-11-22 15:00:34.545  INFO 29088 --- [           main] o.s.cloud.commons.util.InetUtils         : Cannot determine local hostname
2021-11-22 15:00:34.566  INFO 29088 --- [           main] c.e.S.ServiceTwoSbApplication            : Started ServiceTwoSbApplication in 8.38 seconds (JVM running for 8.853)
```

Output

Run (ServiceTwoSbApplication) (2 more...) 1:1 INS

System Status

Environment	N/A
Data center	N/A

Current time	2021-11-22T15:04:26 +0700
Uptime	00:29
Lease expiration enabled	false
Renews threshold	5
Renews (last min)	4

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
SERVICE-ONE	n/a (1)	(1)	UP (1) - service-One:83b86304-9bf8-4d16-bb16-6d95c69697a0
SERVICE-TWO	n/a (1)	(1)	UP (1) - service-Two:f2f01054-ac1b-4d8f-8b5c-0f7455079435

General Info

Name	Value
total-avail-memory	254mb
num-of-cpus	8
current-memory-usage	43mb (16%)
server-uptime	00:29
registered-replicas	
unavailable-replicas	
available-replicas	

Instance Info

Name	Value
ipAddr	192.168.121.1
status	UP

Questions

&

Answers