Spring Cloud

2020.0.3

OVERVIEW | LEARN | SAMPLES

Spring Cloud provides tools for developers to quickly build some of the common patterns in distributed systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, micro-proxy, control bus, short lived microservices and contract testing). Coordination of distributed systems leads to boiler plate patterns, and using Spring Cloud developers can quickly stand up services and applications that implement those patterns. They will work well in any distributed environment, including the developer's own laptop, bare metal data centres, and managed platforms such as Cloud Foundry.

Features

Spring Cloud focuses on providing good out of box experience for typical use cases and extensibility mechanism to cover others.

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Distributed messaging
- Short lived microservices (tasks)
- Consumer-driven and producer-driven contract testing

Talks and videos

- Distributed Applications with Spring Cloud: Spring Office Hours
- Beginner's Guide To Spring Cloud

Getting Started

Generating A New Spring Cloud Project

The easiest way to get started is visit start.spring.io, select your Spring Boot version and the Spring Cloud projects you want to use. This will add the corresponding Spring Cloud BOM version to your Maven/Gradle file when you generate the project.

Adding Spring Cloud To An Existing Spring Boot Application

If you an existing Spring Boot app you want to add Spring Cloud to that app, the first step is to determine the version of Spring Cloud you should use. The version you use in your app will depend on the version of Spring Boot you are using.

The table below outlines which version of Spring Cloud maps to which version of Spring Boot.

Table 1. Release train Spring Boot compatibility (see here for more detailed information).

Release Train	Spring Boot Generation
2023.0.x aka Leyton	3.3.x, 3.2.x
2022.0.x aka Kilburn	3.0.x, 3.1.x (Starting with 2022.0.3)
2021.0.x aka Jubilee	2.6.x, 2.7.x (Starting with 2021.0.3)
2020.0.x aka Ilford	2.4.x, 2.5.x (Starting with 2020.0.3)
Hoxton	2.2.x, 2.3.x (Starting with SR5)
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

Spring Cloud Dalston, Edgware, Finchley, Greenwich, 2020.0 (aka Ilford), 2021.0 (aka Jubilee), and 2022.0 (aka Kilburn) have all reached end of life status and are no longer supported.

Bug fixes and backwards compatible features are added to each release train via a service release (SR). Once you determine which version of Spring Cloud to use, you should use the latest service release for that release train. You can find the latest service release information on our release notes page.

Now that you know which release train to use and the latest service release for that release train you are ready to add the Spring Cloud BOM to your application.

```
<properties>
    <spring-cloud.version>2023.0.2</spring-cloud.version>
</properties>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${spring-cloud.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

```
plugins {
  id 'java'
  id 'org.springframework.boot' version '3.3.0'
  id 'io.spring.dependency-management' version '1.1.4'
}

repositories {
  mavenCentral()
}

ext {
  set("springCloudVersion", "2023.0.2")
}

dependencyManagement {
  imports {
    mavenBom "org.springframework.cloud:spring-cloud-dependencies:${springCloudVersion}"
  }
}
```

It is recommended that you use release train BOM `spring-cloud-dependencies` This is a BOM-only version and it just contains dependency management and no plugin declarations or direct references to Spring or Spring Boot. You can Spring Boot parent POM, or use the BOM from Spring Boot (`spring-boot-dependencies`) to manage Spring Boot versions.

Just like Spring Boot, many Spring Cloud projects include starters that you can add as dependencies to add various cloud native features to your project. In many cases, many features are enabled purely by adding the starter to your classpath. The starter names are documented within the individual projects. Below is an example of how you would add a Spring Cloud Config Client and a Spring Cloud Netflix Eureka client to your application.

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

```
dependencies {
  implementation 'org.springframework.cloud:spring-cloud-starter-config'
  implementation 'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'
  //...
}
```

Main Projects

Spring Cloud Config

Centralized external configuration management backed by a git repository. The configuration resources map directly to Spring `Environment` but could be used by non-Spring applications if desired.

Spring Cloud Gateway

Spring Cloud Gateway is an intelligent and programmable router based on Spring Framework and Spring Boot.

Spring Cloud Netflix

Integration with Eureka Services Discovery from Netflix OSS.

Spring Cloud Consul

Service discovery and configuration management with Hashicorp Consul.

Spring Cloud Data Flow

A cloud-native orchestration service for composable microservice applications on modern runtimes. Easy-to-use DSL, drag-and-drop GUI, and REST-APIs together simplifies the overall orchestration of microservice based data pipelines.

Spring Cloud Function

Spring Cloud Function promotes the implementation of business logic via functions. It supports a uniform programming model across serverless providers, as well as the ability to run standalone (locally or in a PaaS).

Spring Cloud Stream

A lightweight event-driven microservices framework to quickly build applications that can connect to external systems. Simple declarative model to send and receive messages using Apache Kafka or RabbitMQ between Spring Boot apps.

Spring Cloud Stream Applications

Spring Cloud Stream Applications are out of the box Spring Boot applications providing integration with external middleware systems such as Apache Kafka, RabbitMQ etc. using the binder abstraction in Spring Cloud Stream.

Spring Cloud Task

A short-lived microservices framework to quickly build applications that perform finite amounts of data processing. Simple declarative for adding both functional and non-functional features to Spring Boot apps.

Spring Cloud Task App Starters

Spring Cloud Task App Starters are Spring Boot applications that may be any process including Spring Batch jobs that do not run forever, and they end/stop after a finite period of data processing.

Spring Cloud Zookeeper

Service discovery and configuration management with Apache Zookeeper.

Spring Cloud Contract

Spring Cloud Contract is an umbrella project holding solutions that help users in successfully implementing the Consumer Driven Contracts approach.

Spring Cloud OpenFeign

Spring Cloud OpenFeign provides integrations for Spring Boot apps through autoconfiguration and binding to the Spring Environment and other Spring programming model idioms.

Spring Cloud Bus

An event bus for linking services and service instances together with distributed messaging. Useful for propagating state changes across a cluster (e.g. config change events).

Spring Cloud Open Service Broker

Provides a starting point for building a service broker that implements the Open Service Broker API.

Release Trains

Spring Cloud is an umbrella project consisting of independent projects with, in principle, different release cadences. To manage the portfolio a BOM (Bill of Materials) is published with a curated set of dependencies on the individual project. Go here to read about the Release Train naming conventions.

Quickstart Your Project
Bootstrap your application with Spring Initializr.

Get the Spring newsletter
Stay connected with the Spring newsletter
SUBSCRIBE