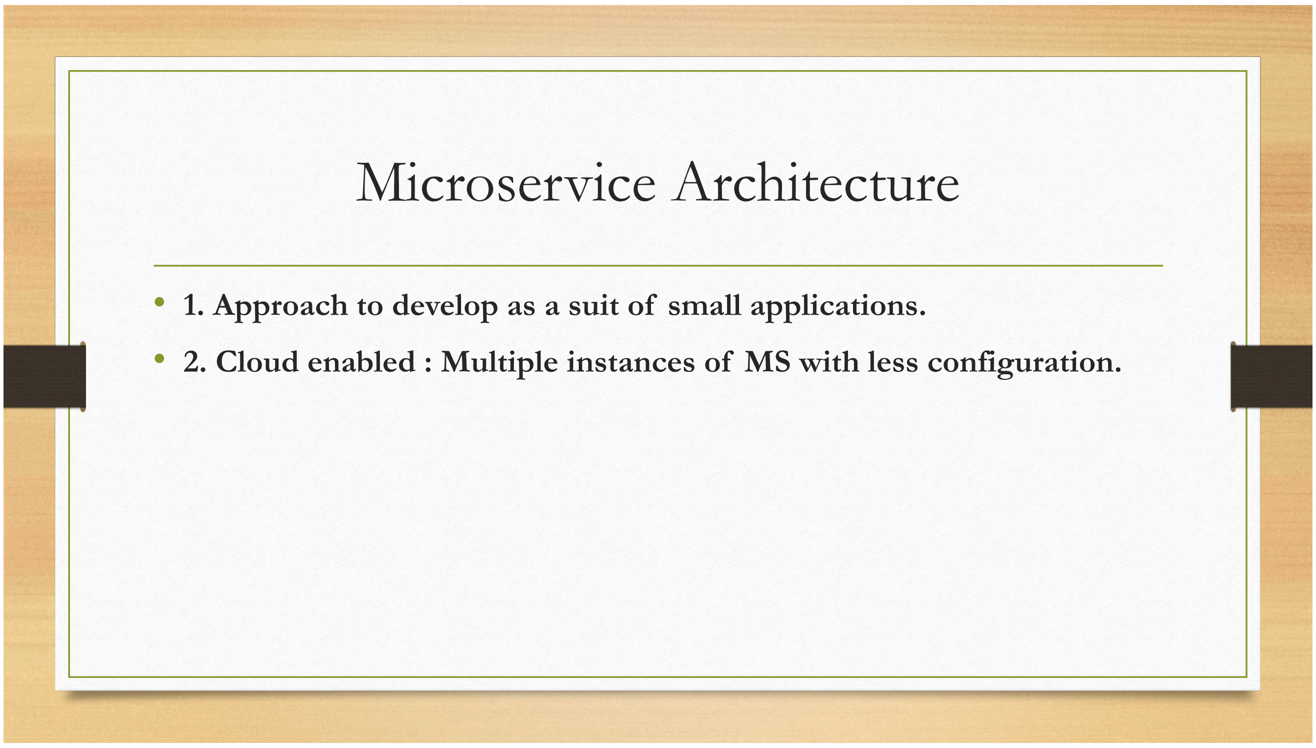
Spring Cloud Introduction



Microservice Architecture

- 1. Approach to develop as a suit of small applications.
- 2. Cloud enabled : Multiple instances of MS with less configuration.

Challenges with Microservices development

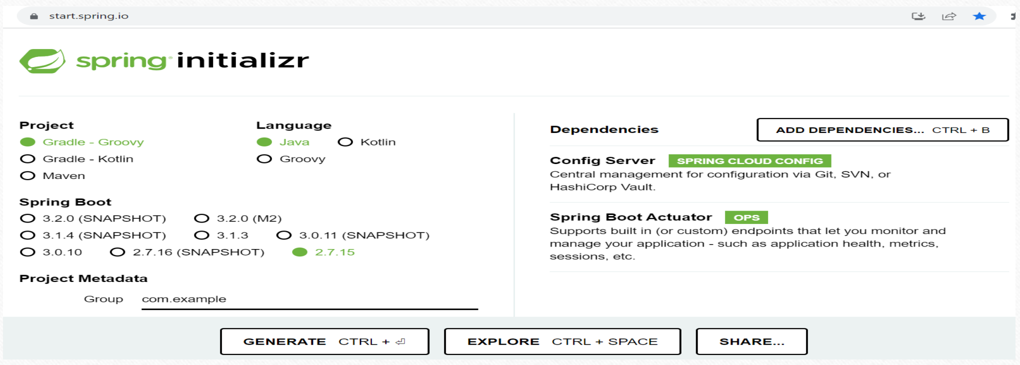
- **1. Well defined boundary: Evolutionary approach**
- **2. Configuration Management: Spring Cloud Config**
- **3. Scale up and Down: Ribbon+ Eureka(Naming Server) OR Spring Cloud Load balancer+ Eureka(Naming Server)**
- **4. Visibility: Zipkin + sleuth, Netflix API gateway**
- **5. Fault Tolerance : Hystrix, Resilence4j**

Spring Cloud Configuration

- **1. Server Application :** Which connect to the git server having properties
- **2. Client's Application :** Which want to use configuration from git server.
- **3. Git Repository :** Location where properties files are kept.

1. Server Application

- Step 1.1: Create a spring-boot application using these springboot starter projects:


start.spring.io

spring initializr

Project
[✓ Gradle - Groovy]
Gradle - Kotlin
Maven

Language
[✓ Java] Kotlin
Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (M2)
3.1.4 (SNAPSHOT) 3.1.3 3.0.11 (SNAPSHOT)
3.0.10 2.7.16 (SNAPSHOT) [✓ 2.7.15]

Project Metadata
Group com.example

Dependencies

Config Server SPRING CLOUD CONFIG
Central management for configuration via Git, SVN, or HashiCorp Vault.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

- Step 1.2: Add @EnableConfigServer over Springboot main class

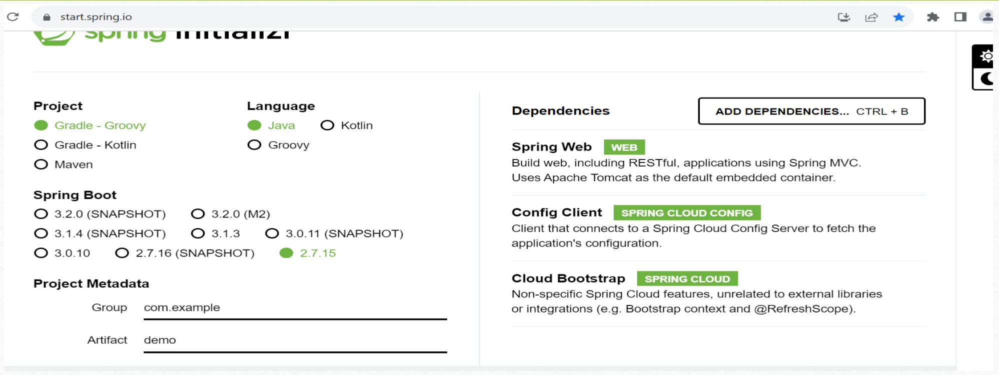
```
@EnableConfigServer
@SpringBootApplication
public class CloudconfigserverApplication {
    public static void main(String[] args) {
        SpringApplication.run(CloudconfigserverApplication.class, args);
    }
}
```

- Step 1.3: Specify the Git URL of the repository in application.properties.

```
spring.application.name= spring-cloud-config-server
server.port= 8888
spring.cloud.config.server.git.uri= https://github.com/aroopkumar/spring-cloud-config.git
#spring.cloud.config.server.git.uri= file:///C:/Users/Aroop.Kumar/Downloads/git-local-config-repository
spring.cloud.config.server.git.defaultLabel= master
```

2. Client's Application

- Step 2.1: Create a spring-boot application using these springboot starter projects:


start.spring.io

Project
● Gradle - Groovy
○ Gradle - Kotlin
○ Maven

Language
● Java ○ Kotlin
○ Groovy

Spring Boot
○ 3.2.0 (SNAPSHOT) ○ 3.2.0 (M2)
○ 3.1.4 (SNAPSHOT) ○ 3.1.3 ○ 3.0.11 (SNAPSHOT)
○ 3.0.10 ○ 2.7.16 (SNAPSHOT) ● 2.7.15

Project Metadata
Group com.example
Artifact demo

Dependencies ADD DEPENDENCIES... CTRL + B

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

Config Client SPRING CLOUD CONFIG
Client that connects to a Spring Cloud Config Server to fetch the application's configuration.

Cloud Bootstrap SPRING CLOUD
Non-specific Spring Cloud features, unrelated to external libraries or integrations (e.g. Bootstrap context and @RefreshScope).

- Step 2.2: Create Configuration class which can read the properties from cloud server

```
@Component
@ConfigurationProperties("limits-service")
public class CloudConfigurationReader {
   int minimum;
   int maximum;
      //Getter and Setter
}
```

- Step 2.3: Autowired Configuration class in your beans where you want to use it.

```
@RestController
@RequestMapping("/limits")
public class LimitController {
   @Autowired
   private CloudConfigurationReader configuration;
   @GetMapping("/getlimits")
   public CloudConfigurationReader getLimits(){
      return configuration;
   }
}
```

- Step3: Configure the Config Server Url in application.properties

```
spring.application.name= limits-service
spring.config.import=optional:configserver:${SPRING_CLOUD_CONFIG_URI:http:/
/localhost:8888}
spring.profiles.active= dev
```

3. Git Repository

- Step 3.1: Create multiple properties file based on springboot profiling.

1. limits-service.properties
2. limits-service-dev.properties
3. limits-service-pre.properties

```
limits-service.minimum=30
limits-service.maximum=300
```

Step 3.2: Adding config files to git

```
git init
git add -A
git commit -m "first commit"
git remote add origin https://github.com/aroopkumar/spring-cloud-config.git
git push --set-upstream origin master
```

Code Reference:

1. https://github.com/aroopkumar/cloudconfigserver.git
2. https://github.com/aroopkumar/cloudconfigclient.git
3. https://github.com/aroopkumar/spring-cloud-config.git

Thank you.