# Spring Cloud Config

## Introduction of the Topic

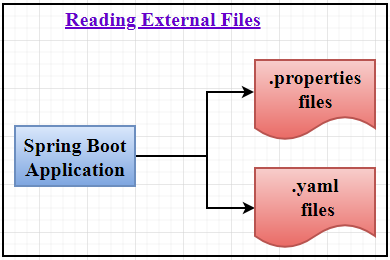
Microservices approach now has become an industry standard for any new API development, and almost all the organizations are promoting it. Spring cloud provides excellent tools to build these microservice on top of the Spring boot framework.

## Content

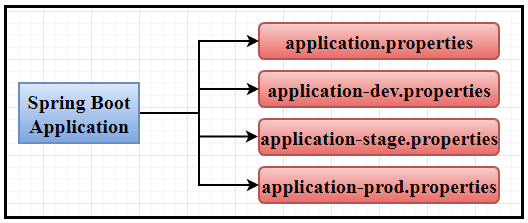
### Why do we require Spring Cloud Config?

#### Updating the properties without restarting the servers

SpringBoot provides lot of flexibility in externalizing configuration properties via properties or YAML files.



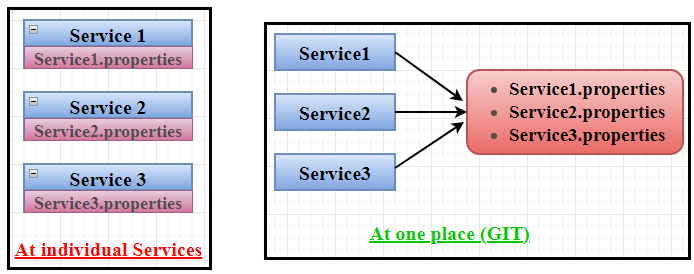
We can also configure properties for each environment (dev, qa, prod etc)separately.



But once the application is started, we can not update the properties at runtime. If we change the properties, we need to restart the application to use the updated configuration properties.

#### Managing the properties at one place

Also, in the context of large number of MicroService based applications, we want the ability to configure and manage the configuration properties of all MicroServices from a centralized place.



#### Maintaining the Common properties

If we store the common properties like database props at individual services, if there is any change, we need to update in all the services.

|  |
| --- |
|  |

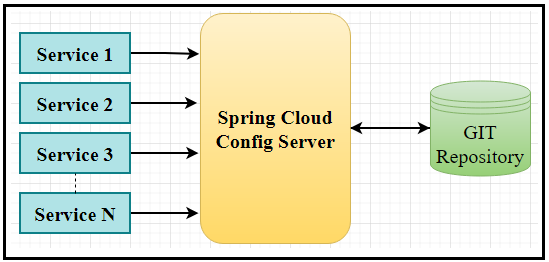
What are some of the alternatives in placing our configuration details?

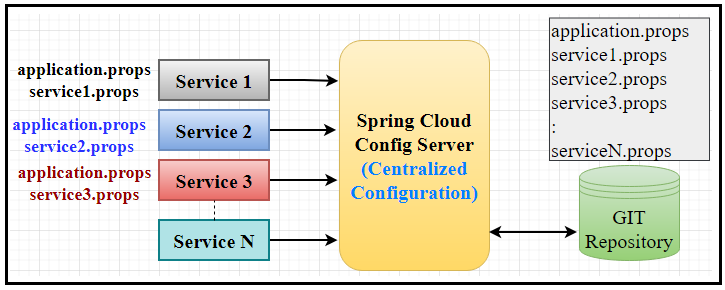
* Placing the configuration as environment variables - but there is a limit on creation of environment variables
* Placing them in external files – but access to such file system is difficult from cloud.

### What is Spring Cloud Config Server?

Spring Cloud Config provides server-side and client-side support for externalized configuration in a distributed system.

We can use Spring Cloud Config Server to centralize all the application configurations and use Spring Cloud Config Client module from the applications to consume configuration properties from Config Server. We can also update the configuration properties at runtime without restarting the application.





We can place the common properties in “application.properties” file, So that they will be available across all the services.

### Fail Fast

* spring.cloud.config.fail-fast = true

The service won’t start if the Config Server is not available.

* spring.cloud.config.fail-fast = false

The service will start even the config service is not available. Properties will use the default values.

### Geting Latest Configuration

To load the latest configurations without restarting the applications, disable the security and enable the actuator “refresh” endpoint.

|  |
| --- |
| management:  security:  enabled: false  endpoints:  web:  exposure:  include: health, refresh, bus-refresh |

Adding **@RefreshScope** annotation on the controller.

|  |
| --- |
| @RestController  @RequestMapping("/test")  **@RefreshScope**  **public** **class** PropertiesController {  @Value("${message}")  **private** String message;  } |

### Intervals Configuration

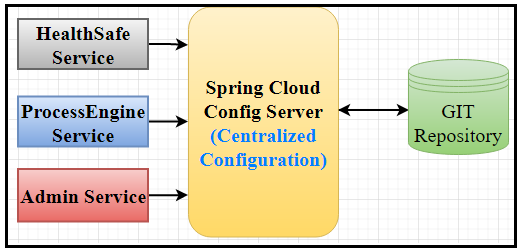
The config-server is contacted by the clients only once, during the start of the project. Therefore, any changes made to the configuration after the application starts will not be reflected in the application.

We can also configure the clients retry attempts to contact the config server using the properties like:

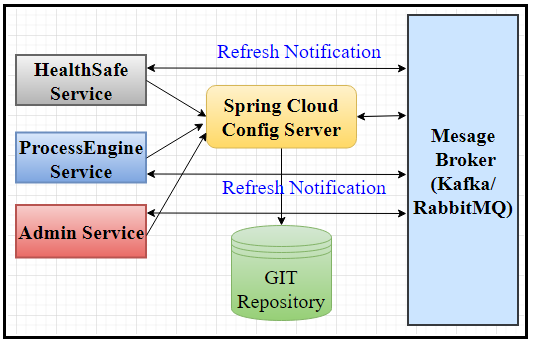
|  |  |  |
| --- | --- | --- |
| **Property** | **Default** | **Usage** |
| initialInterval | 1000ms | Initial retry interval in ms |
| multiplier | 1time | Multiplier for next interval |
| maxInterval | 2000 ms | Max interval for backoff |
| maxAttempts | 6 times | Max number of attempts |

## Demo

### Spring Cloud Config Server



### Spring Cloud Bus



### Disadvantages of using Spring Cloud Config

* Depending on the external repository like GIT, SVN…etc.
* Creating an extra service (Spring Cloud Config Server) to deal with the GIT repository.
* Implementing all the services to deal with Spring Cloud Config Server.
* Need to send a refresh call to get the updated configurations for GIT.

## Reference Links

<https://cloud.spring.io/spring-cloud-config/reference/html/>

https://cloud.spring.io/spring-cloud-static/spring-cloud-config/2.0.2.RELEASE/single/spring-cloud-config.html

<https://howtodoinjava.com/spring-cloud/spring-cloud-config-server-git/>

<https://cloud.spring.io/spring-cloud-static/Dalston.SR5/multi/multi__spring_cloud_config_client.html>

## Q&A