# Spring Cloud Config

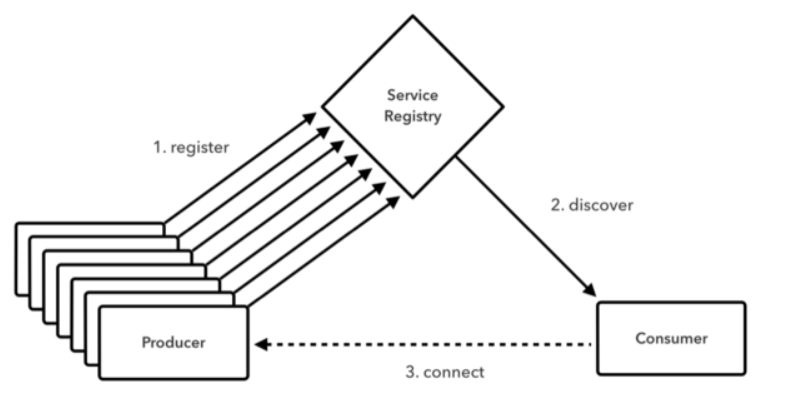
## What is Spring Cloud Config?

Spring Cloud Config provides server and client-side support for externalized configuration in a distributed system. With the Config Server you have a central place to manage external properties for applications across all environments. The concepts on both client and server map identically to the Spring Environment and PropertySource abstractions, so they fit very well with Spring applications, but can be used with any application running in any language. As an application moves through the deployment pipeline from dev to test and into production you can manage the configuration between those environments and be certain that applications have everything they need to run when they migrate. The default implementation of the server storage backend uses git so it easily supports labelled versions of configuration environments, as well as being accessible to a wide range of tooling for managing the content. It is easy to add alternative implementations and plug them in with Spring configuration.

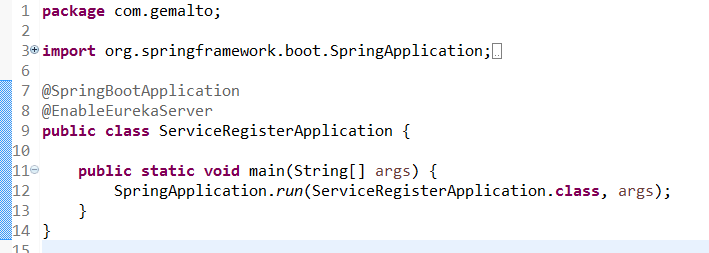
## How to use

### Setup Server Register

The discovery server. It contains a registry of services and a REST api that can be used to register a service, deregister a service, and discover the location of other services. Your applications can use the Service Registry to dynamically discover and call registered services.



A Eureka Service registry. can use Spring Cloud’s @EnableEurekaServer to standup a registry that other applications can talk to. This is a regular Spring Boot application with one annotation added to *enable* the service registry.



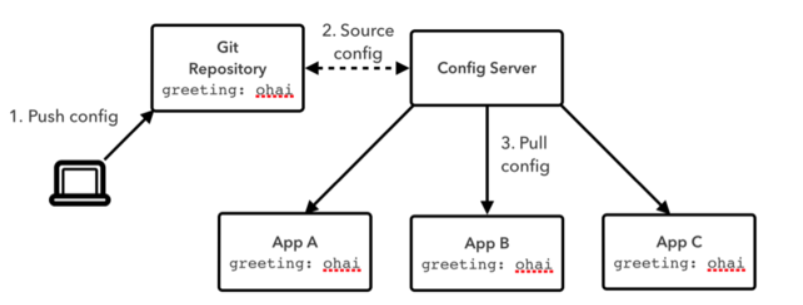
You can startup our register server as the following command：

**Java –jar serviceRegister-0.0.1-SNAPSHOT.jar**

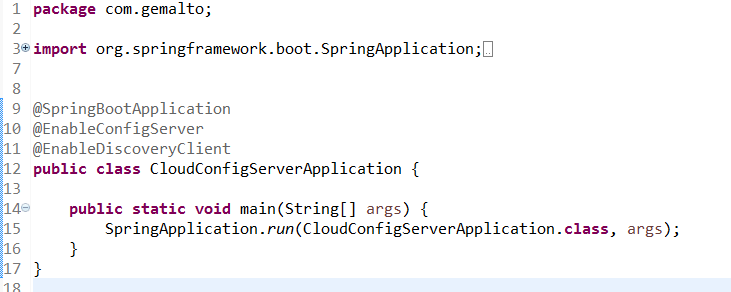
Server runs on 1111 port

### Setup Config Server

The Config Server we have a central place to manage external properties for applications with support for different environments. Configuration files in several formats like YAML,JOSN or properties are added to a Git repository. The server provides an REST API to get configuration values.



Building Config Server is very simple. Add dependency on org.springframework.cloud:spring-cloud-config-server and just put EnableConfigServer to the entry point class.



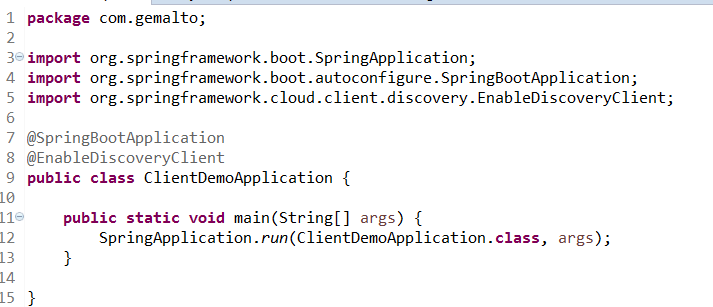
You can startup our register server as the following command：

**Java –jar config-0.0.1-SNAPSHOT.jar**

Server runs on 8888 port

### Setup Client

To use these features in an application, just build it as a Spring Boot application that depends on spring-cloud-config-client . When it runs it will pick up the external configuration from the default config server on port 8888 if it is running.



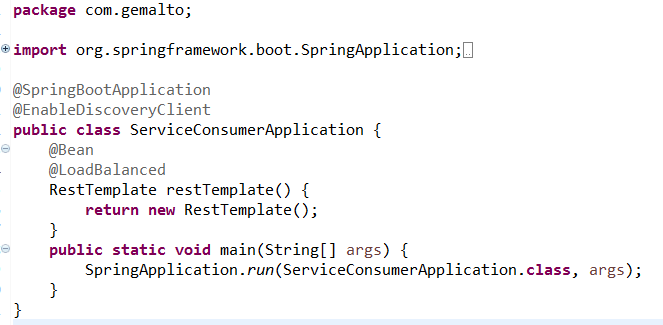
You can startup our register server as the following command：

**Java –jar clientDemo-0.0.1-SNAPSHOT.jar --server.port=8081**

Server runs on 8081 port

### Setup Server Consumer

Server Consumer is also a load-balancer so if you have multiple instances of a service available, it picks one for you.



You can startup our register server as the following command：

**Java –jar serviceConsumer-0.0.1-SNAPSHOT.jar**

Server runs on 3333 port