Cloud Config Server

server-side and client-side support for externalized configuration in a distributed system

Introduction

- central place to manage external properties for applications across all environments.
- client and server map identically to the Spring **Environment** and **PropertySource** abstractions.
- They fit very well with Spring applications but can be used with any application running in any language
- The default implementation of the server storage backend uses **git**

Server Configuration

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

```
==========================================================================================
server.port=8888
spring.cloud.config.server.native.search-locations=file:///home/dr/Desktop/
spring.cloud.config.server.bootstrap=true
spring.profiles.active=native
spring.cloud.config.server.native:searchLocations=file://${user.dir}/cloud-configuration-repository
```

Server Configuration

```
import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.clo@SpringBootApplication

@EnableConfigServer

public class ConfigserverdemoApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigserverdemoApplication.class, args);

}

}
```

Sample Client

curl localhost:8888/foo/development

{"name":"foo","label":"master","propertySources":[
{"name":"https://github.com/scratches/config-repo/foo-development.properties","source":{"bar":"spam"}},
{"name":"https://github.com/scratches/config-repo/foo.properties","source":{"foo":"bar"}}
]}

Sample Client

curl localhost:8080/env


```
{
  "profiles":[],
  "configService:https://github.com/spring-cloud-samples/config-repo/bar.properties":{"foo":"bar"},
  "servletContextInitParams":{},
  "systemProperties":{...},
  ...
}
```

application.propreties

- spring.cloud.config.server.git.uri = https://github.com/tbarua1/testing
- spring.cloud.config.server.git.timeout = 4

```
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/tbarua1/testing
          Timeout: 4
```

in seconds, that the configuration server will wait to acquire an HTTP connection

HTTP service resources format

- /{application}/{profile}[/{label}]
- /{application}-{profile}.yml
- /{label}/{application}-{profile}.yml
- /{application}-{profile}.properties
- /{label}/{application}-{profile}.properties

where application is injected as the *spring.config.name* in the SpringApplication

Client Side Usage

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

application.yml

- if no application name is set, application will be used. To modify the name, the following property can be added to the bootstrap.properties file:
- spring.application.name: myapp

application.yml

- When this HTTP server runs, it picks up the external configuration from the default local config server on port **8888**. To modify the startup behavior, you can change the location of the config server by using bootstrap.properties (The bootstrap phase of an application context)
- spring.cloud.config.uri: http://myconfigserver.com

```
@SpringBootApplication
@RestController
public class Application {
    @RequestMapping("/")
    public String home() {
        return "Hello World!";
    }
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

application.yml

Spring Cloud Config Server

- HTTP resource-based API for external configuration (name-value pairs or equivalent YAML content).
- The server is embeddable in a Spring Boot application, by using the **@EnableConfigServer** annotation.

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

application.properties

- spring.config.name=configserver
- server.port: 8888
- spring.cloud.config.server.git.uri: file:///${user.home}/config-repo
- spring.cloud.config.server.accept-empty = false

Server would return a HTTP 404 status, if the application is not found.By default, this flag is set to true.

Creating application.properties in git Repository

- cd $HOME
- mkdir config-repo
- cd config-repo
- git init .
- echo info.foo: bar > application.properties
- git add -A .
- git commit -m "Add application.properties"

Environment Variables

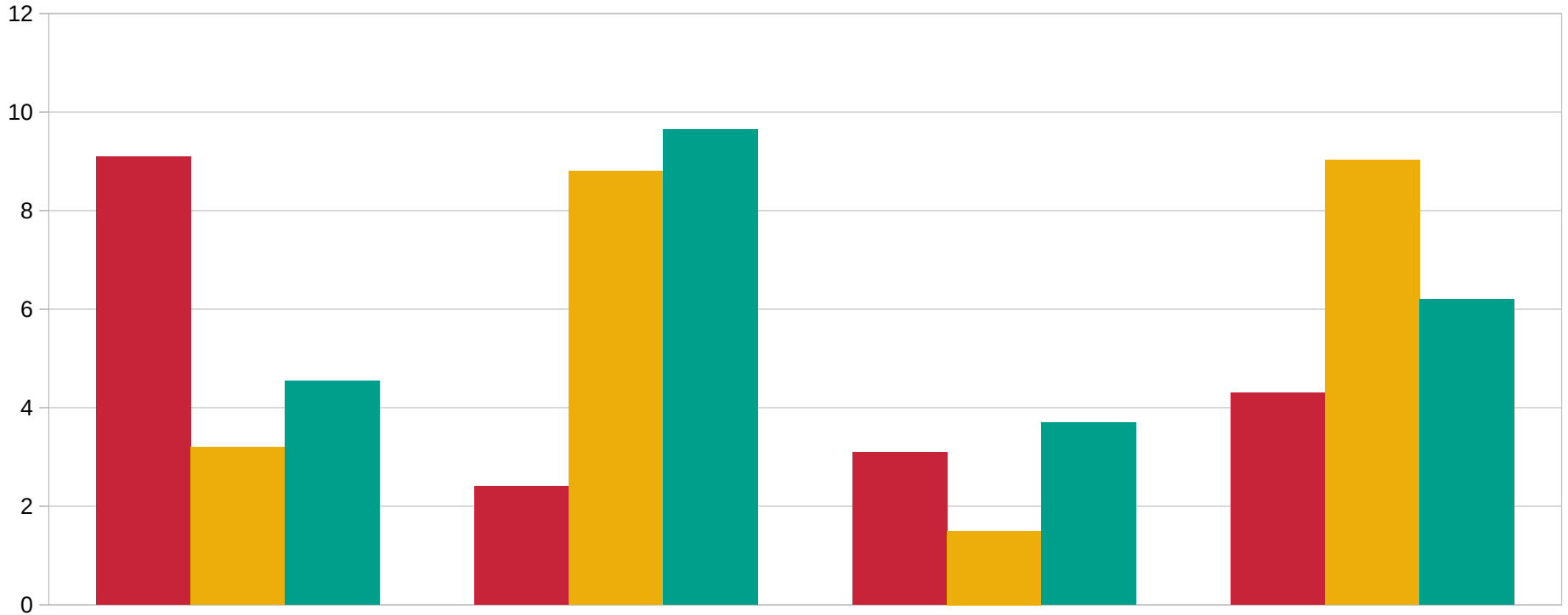
- **{application}**, which maps to spring.application.name on the client side.
- **{profile}**, which maps to spring.profiles.active on the client (comma-separated list).
- **{label}**, which is a server side feature labelling a "versioned" set of config files.

Multiple Repo Config

```
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/spring-cloud-samples/config-repo
          repos:
            simple: https://github.com/simple/config-repo
            special:
              pattern: special*/dev*,*special*/dev*
              uri: https://github.com/special/config-repo
            local:
              pattern: local*
              uri: file:/home/configsvc/config-reposspring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/spring-cloud-samples/config-repo
          repos:
            simple: https://github.com/simple/config-repo
            special:
              pattern: special*/dev*,*special*/dev*
              uri: https://github.com/special/config-repo
            local:
              pattern: local*
              uri: file:/home/configsvc/config-repo
```

Multiple Repo Config pattern

```
spring:
  cloud:
    config:
      server:
        git:
          uri: https://github.com/spring-cloud-samples/config-repo
          repos:
            development:
              pattern:
                - '*/development'
                - '*/staging'
              uri: https://github.com/development/config-repo
            staging:
              pattern:
                - '*/qa'
                - '*/production'
              uri: https://github.com/staging/config-repo
```



12
10
8
6
4
2
0