

1. The Git repository is used to store the configuration since Git is pretty good at tracking and storing changes.
2. The Config server keeps itself up to date with the Git repository, and serves HTTP requests with configurations for the clients.
3. The Config clients request the configuration from the config server, which sets the properties in the application.

**Setting up your Git repository**

The configuration is maintained in a Git repository. If you have an application called 'configclient' you can specify it as seen below (or on [Github](https://github.com/rpicloud/guide-cloud-config/blob/master/configclient.yml)).

configuration:

projectName: configclient

server:

port: 8000

message:

greeting: Hello from the configuration

**Save this file with name configclient.yml**

Configuration: projectName shoule be same as file name.

Server: port is port number wer client application will be running

Message: greetings is plain string message.

### Setting up the server

To set up the server you must:

* Create spring cloud starter project and add required dependencies.
* Annotate the main application with @EnableConfigServer.
* Add a bootstrap.yml (under resources)
* Add the Git repository uri in the key spring.cloud.config.server.git.uri(which is created in step 1 **configclient.yml**)
* Set the application name as follows: spring.application.name=config-service (optionally) set server.port=8888

The resulting bootstrap.yml is shown below.

spring:

application:

name: config-service

cloud:

config:

server:

git:

uri: https://github.com/kotechachirag/cloud-config

server:

port: 8888

Start the server by running mvn spring-boot:run from within the folder.   
You should now be able to see the server at <http://localhost:8888/configclient/default>   
If you didn't change server.port it is running at 8080(8080 is default port).

### Setting up the client

To set up the client you must:

* add a bootstrap.yml (under resources)
* add the application name as seen below
* add the uri to the cloud config server as seen below

spring:

application:

name: configclient

cloud:

config:

uri: http://localhost:8888

uri: <http://localhost:8888> is pointing the server which we have configured above.

**Rest controller to check spring configuration properties.**

@RefreshScope

@Component

@RestController

public class Greeter {

@Value("${message.greeting}")

String greeting;

@Value("${server.port}")

int port;

@Value("${configuration.projectName}")

String projectName;

@RequestMapping(value = "/", produces = "application/json")

public List<String> index(){

List<String> env = Arrays.asList(

"message.greeting is: " + greeting,

"server.port is: " + port,

"configuration.projectName is: " + projectName

);

return env;

}

}

Now you should see a page telling you what the configuration server returned to you when you visit <http://localhost:8000/>.

Complete code you can checkout from github at below url.

<https://github.com/kotechachirag/cloud-config>