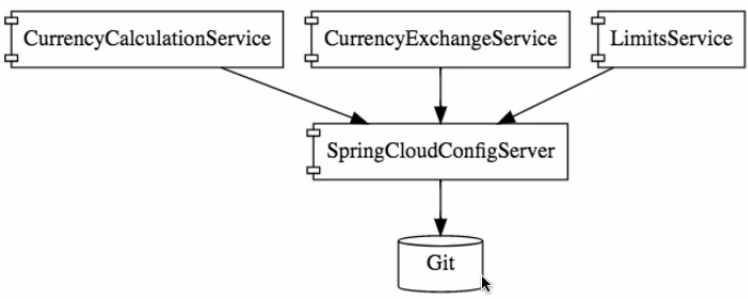
## Implementing config server

This config server must be a separate project



1. It’s a thought of externalizing the property files from our application and keeping it in a separate git repository, the great thing is all application properties files are present in 1 place,

* If u ask the config server with the name of property file , it will goto the git repository and fetch the related file and pick the respective environment file
* Its like a centralized configuration

# Reference

https://github.com/in28minutes/spring-microservices-v2/tree/main/03.microservices

This config server can store properties of all applications and all environments

Running on diff port

-DServer.port=8001

U can get the port infor as -- 



## Course update –add dependency

**If you are using 2.4.0, you need to add this dependency to the pom.xml:**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-bootstrap</artifactId>

</dependency>

## Debugging help

<https://github.com/in28minutes/in28minutes-initiatives/tree/master/The-in28Minutes-TroubleshootingGuide-And-FAQ#debugging-problems-with-spring-cloud-config-server>

Connect cloud config server to git

Ref code :- <https://github.com/in28minutes/spring-microservices/tree/master/03.microservices/limits-service>

spring.application.name=spring-cloud-config-server  
server.port=8888  
**spring.cloud.config.server.git.uri**=file:///C:/Users/Manideep/OneDrive/Desktop/deleteMee  
**spring.cloud.config.server.git.default-label**=master  
**spring.cloud.config.server.git.clone-on-start**=true

spring.cloud.config.server.git.uri= <file:///in28minutes/git/spring-microservices>

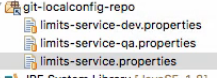
<This is the path where .git folder is present>

file:///C:/Users/Manideep/OneDrive/Desktop/deleteMee

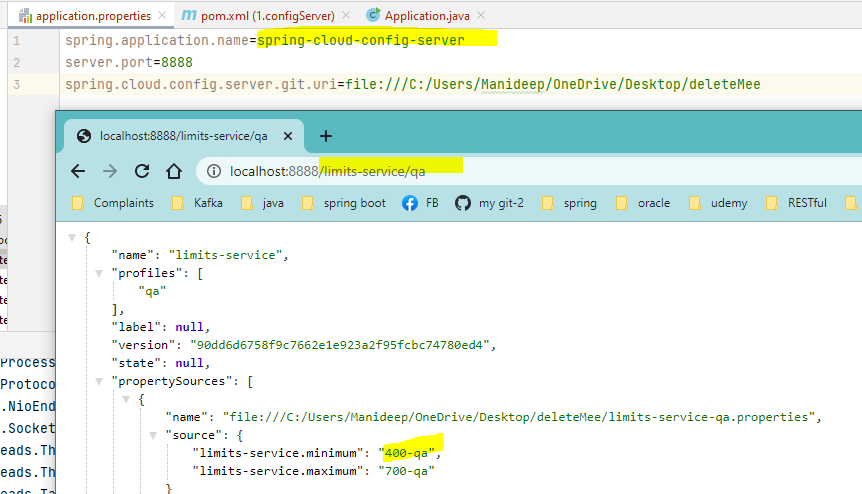
1. add @EnableConfigServer annotation on top of that main method

Therefore, eventhough ur cloud config server name is diff or if it is having a context root for ur spring cloud config server , while searching **we should not search** with that context-root , we should search only with that property file and env name

1. create all properties like below with environment names,
2. create a git repo and commit all like below
3. to create git repo click “**git init**” and commit command as 🡪 **git commit –m “abcd”**

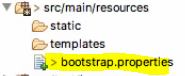


In the above limits-qa.properties will have the highest priority , if the expected value is not present in that property file it will pick from the default property file.



## Connect Limits-service to spring cloud config server

1. Rename the application.properties to bootstrap.properties



1. Place the config server git uri location in limits-service as below, but in latest versions the key name is changed

spring.cloud.config.uri=http://localhost:8888

Now this will fetch values from limits-service-**dev**.properties file , because we have activated dev profile , but generally this is not the correct way to implement , provide the profile name in vm arguments

If u keep qa, then it will fetch limits-service-qa.properties file.

Or use below property

Spring.config.import=optional:configserver:http:localhost:8888

spring.config.import=optional:configserver:http://myconfigserver.com

From ur Dev appn connect to QA property file

Soring.profiles.active=dev

spring.cloud.config.profile=qa – eventhough as per above ,if ur appn is running on dev environment

if ur cloud profile is activated as QA, ur dev app will connect to qa config server/ it will load the QA config property file

From ur appn fetch diff appn property files

spring.application.name= santhoshi

spring.cloud.config.name=manideep ,

Now eventhough ur appn is santhoshi as per above , since ur cloud config name is manideep it will fetch the manideep related property file

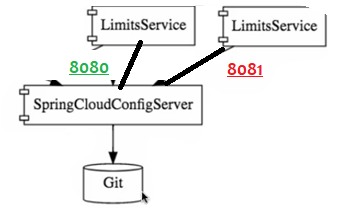
Reflect run time prop file changes in Appn on git commit

Generally, when there is a change in property file we will commit those changes to GIT.

But while appn is getting started at that point itself it used to fetch the property file changes from GIT, but once after the appn is started then if u make changes and if u want those prop file changes to reflect in appn then

Hit actuator refresh url

localhost:8080/actuator/refresh if u hit this POST request then latest prop file changes from git will be pulled and maintained in the spring context



Lets say if we have 2 instances of limits-services , if there are any configuration changes in property files after committing to git repo , if u want those prop file changes to reflect in ur appn then u have to trigger this url

Localhost:8080/actuator/refresh

If u have 1 more instance then trigger same request

Localhost:8081/actuator/refresh

If u have 100 such microservices u have to do the same approach which is the problem

### **Spring cloud bus**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-bus-amqp</artifactId>

</dependency>

If 10 apps are there , when property files are changed and committed to git repo

To reflect those changes we should manually hit actuator POST end point on each vm as per above

Instead of doing that on multiple nodes we should go with spring cloud bus

As and when appn is starting, all appns will be registered with spring cloud bus.

**So u have to hit below url only once , even if u have 10 vms**

**localhost:8080/actuator/bus-refresh this u can hit on any registered instance. The refresh will automatically happens on other vms too.**