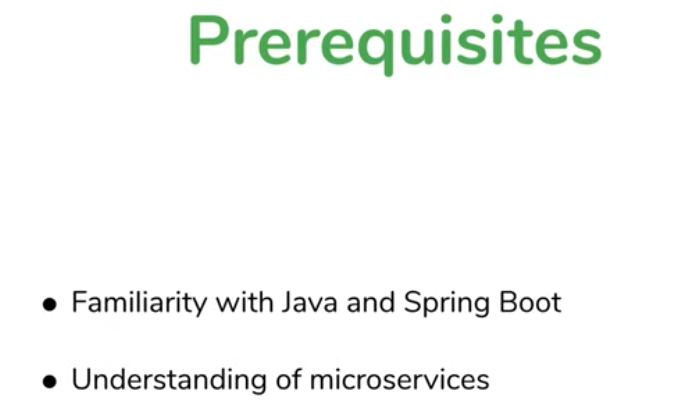
Spring BOOT Micro service - Level 3 Microservice Configurtaion

- configure microservices

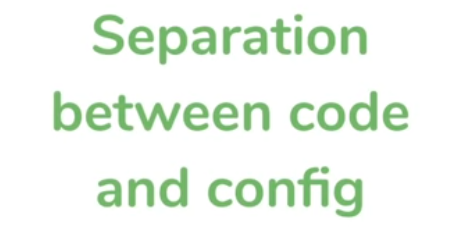
-things to do make microserive easily confgiurable and manageable



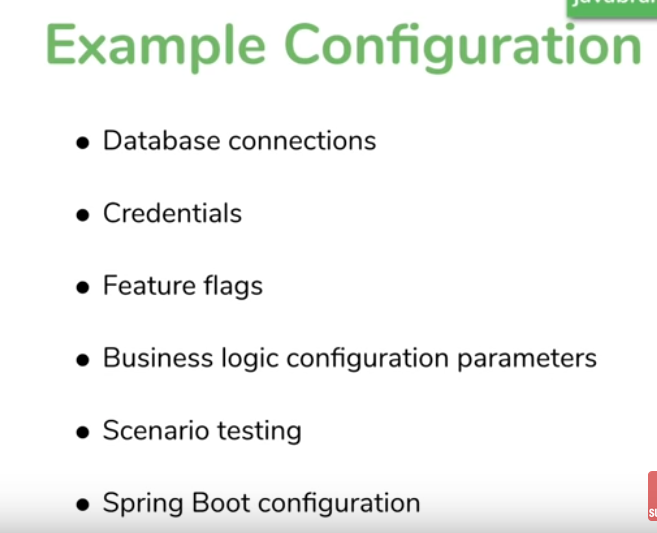
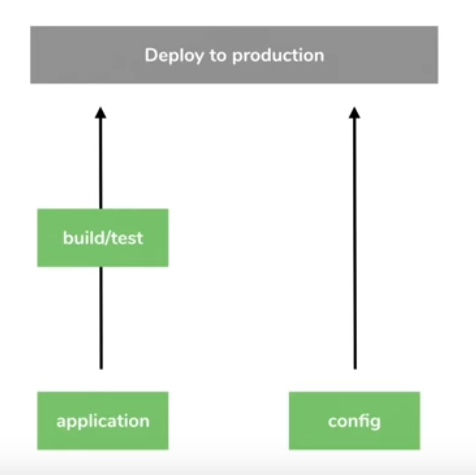




Configuration is very essential for any production application



Ex : db connection details in property file or xml file instead of source code

’

- If somethign change is source it has to be built r tested then moved to prodcution

- wt if there is only need to change connection pool in config no need to build test directly we can do in production

- config has to be optimized , fast, easy and realtime

Types of Config Files

1 XML files

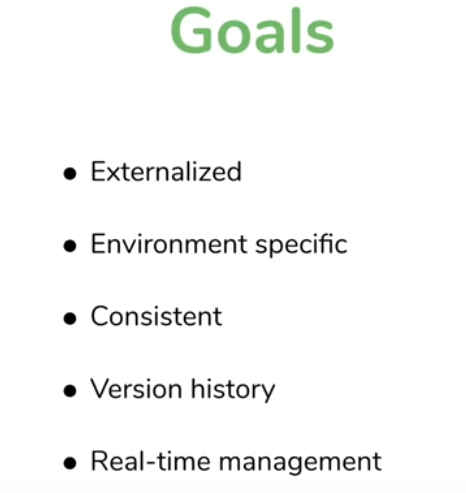
2 Propertie files, YAML files ,JSON (close to YAML) Commonly used

What is the big deal?

It bigger deal with microservice

- multiple microservices

- multiple instance of each microservice



Externalized already we know since taking appliaction.properties file



We will see out access values from propertie files

@Value(“${key}”)

@Value(“${greetMeet}”)

Private String greetMeet;

app.name=My NAme

app.descriotion=description for my app is ${app.name}

Properties file still will be inside jar even we put property from code and put it in jar file

What if externalizing if jar holds property file inside even we have advantage if its inside the jar we have ways to override it



Propertie file inside jar how to change it with out effecting jar file and making externalize it

Open terminal

Target folder will have my jar or war file

Java -jar app\_name.jar => it will take values in the application.properties file and run it

Suggetion 1 : create application.properties in same path where jar is present

Add same key with new value

And run

Java -jar app\_name.jar

Now 1st it will run propertie file inside jar later it seem same jar outside of it so it overrides it

Suggestion 2 : using Command lines

Java -jar app\_name.jar --key=”msg to add fr key”

1st internal property file nxt exteranle then anything in command line

Suggestion 3 : system properties can be accessed using @value

@value anntoation is an essential way to get value from property files

3 tricks of @Value

1 st 

Assiagn string value directly

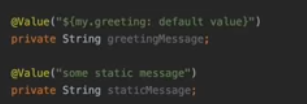


2nd

For ex I dnt have key in propertie files still I used in controller I get build it will treated like a bean

- dnt want to these caues I want some fallback to handle this issue if property key not exist

@Value(“${key : def value}”)



3rd Get Array

My.key=one,two,three

@value(“${My.key}”)

Private List<String> listValues;

Look like separated by comma set to list separated by comma

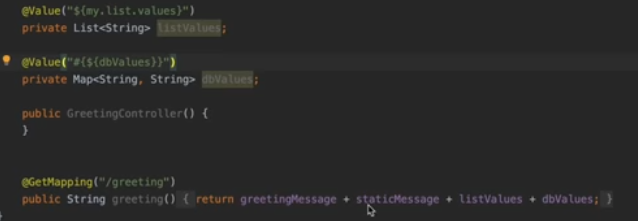
Get Key value pair

Db.value={ConnectionString:”http://path”.username:”sachin”,password:”xxxx”}

Evalate this has expression by using #{${key}} what # does it treats things inside curly brace after it has sprignexpresion lang and evvaluate it

@value(“#{${dbvalues}}”)

Private Map<String ,String> dbvalues;



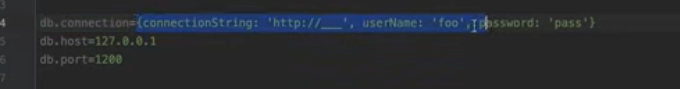
To access each element of map

@ConfigurationProperties

- specfic annotaion related to pulling config values from property files specfic when we have group of values and pull it ones and make it available

More Powerfull way to pull up the properties from property file

For example we have bunch of keys like



Instead We add @value to one by one to property keys we will create a bean

db.connection=oracle

db.host=127.0.0.1

db,port=1521

/\*\*

\*

\* **@author** --SACHIN

\*

\* **@ConfigurationProperties**("db") :> prefix to look up values from

\* properties dnt want look up everything just prefix like db.connection

\* 1st it will look for properties start with "db" and any of the name

\* matches member variable and set the values create a new instance of

\* DBSetting and set values to memborable variables

\*

\* **@Configuration** :> telling to create a bean of it and access it in any

\* controller by autowiring it

\*

\* **@Autowired** private DBSetting dbSetting;

\*

\*

\*

\*/

@Configuration

@ConfigurationProperties("db")

@Getter

@Setter

public class DBSetting {

private String connection;

private String host;

private int port;

}

If I set db.port to foo db.port=foo but in class file its int for port while strt it will give error Instead

This kind of type checking also works for variable annotated with @value and **@ConfigurationProperties** also

If we have single value are single data then @Value is a gud usage

Ex list,are some single value details

For @ConfigurationProperties we have 2 advantage Group value together to get it and another reason is it wil end up as spring bean we can use it in multiple location,

Ex db connection

Spring Boot Actuator

- specfic end points its exposes in boot application

- exposes specfic REST endpoints to get all config properties by our self or by spring boot

Add acutator dependency

-limitied exposed endpoints by default need to mention expose endpoints not gud in Production

Management.endpoints.web.exposure.include=\*

Localhost:8080/actuator/configprops

- expose all config properties and even ours also

Using YAML files

YAML format : Yet Another Markup lang / Yaml aint markp lang

{ and \* cnt understood by YAML so use “{}” “\*” use with “” for those calues if its yml file

Indendent has to space not tab avoid tabs

Spring Profiles

Till now we achieved Externalized(using property files,yml file)

- To make it enviornment specifc





Solution : Spring Profile

There is no profile says at start

No active profile set , falling back to default profile application.properties/yml is default

Application-<profilename>.extn

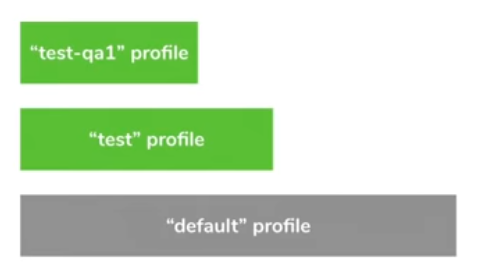
application-test.yml

application-dev.yml

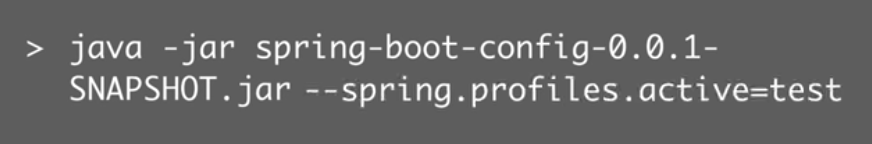
In default application file mention spring.profiles.active: test

Default profile always active

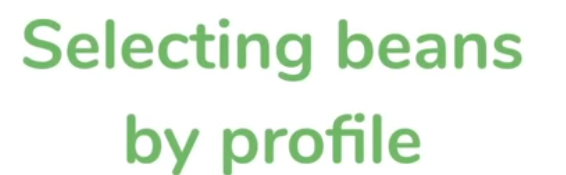
We can more then one active profile



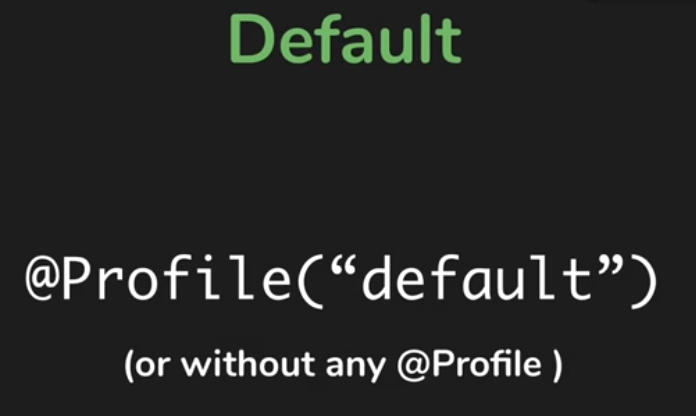
Command lines



Not only for configuring values it can used it is used for Selecting beans by profile









@value r @Configuration properties or spring injection

Using Environment object we can look what the object and profile we have

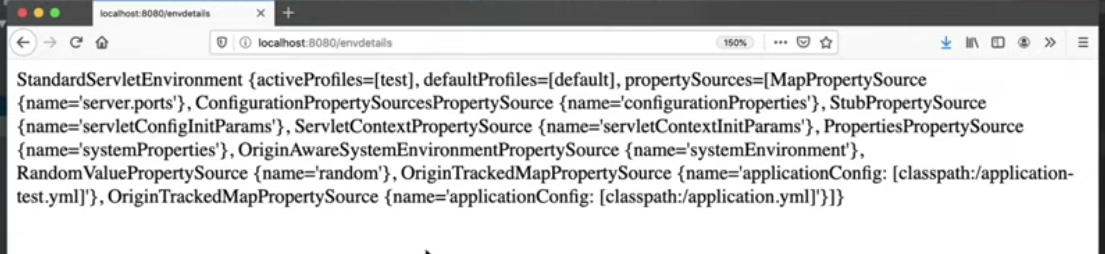
Environment is a bean / object we can autowire it into our class

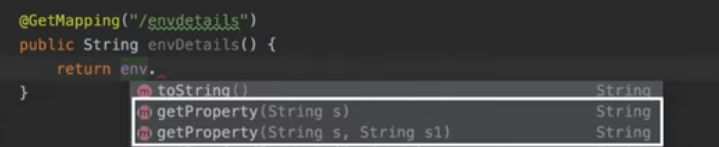
@autowire

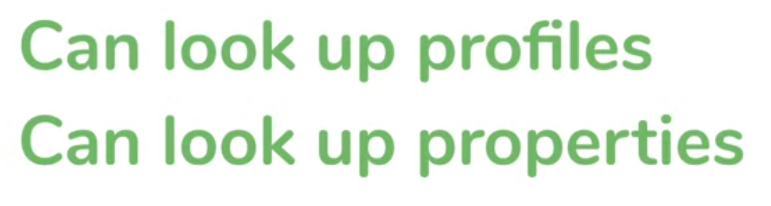
Private Environament env;

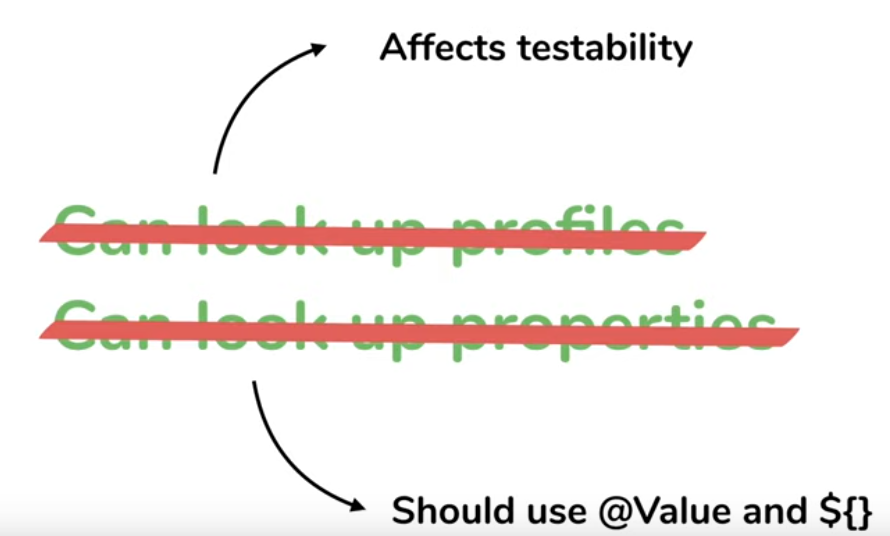
It is from spring pacaage itself









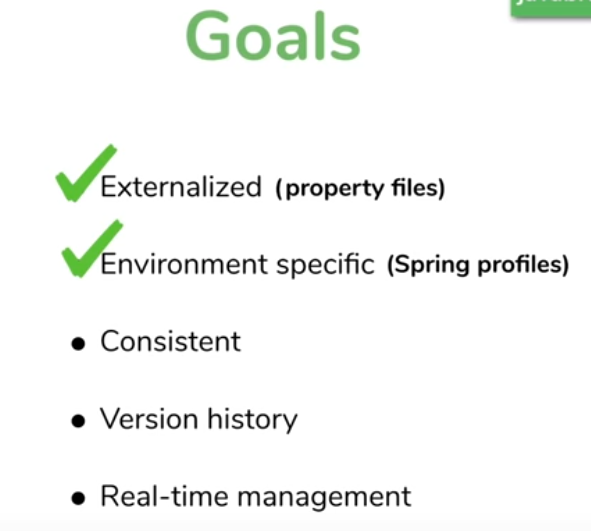


Spring Cloud config server



Till now configuration is ok since we are using only one microservice nw gng for multiple microservice



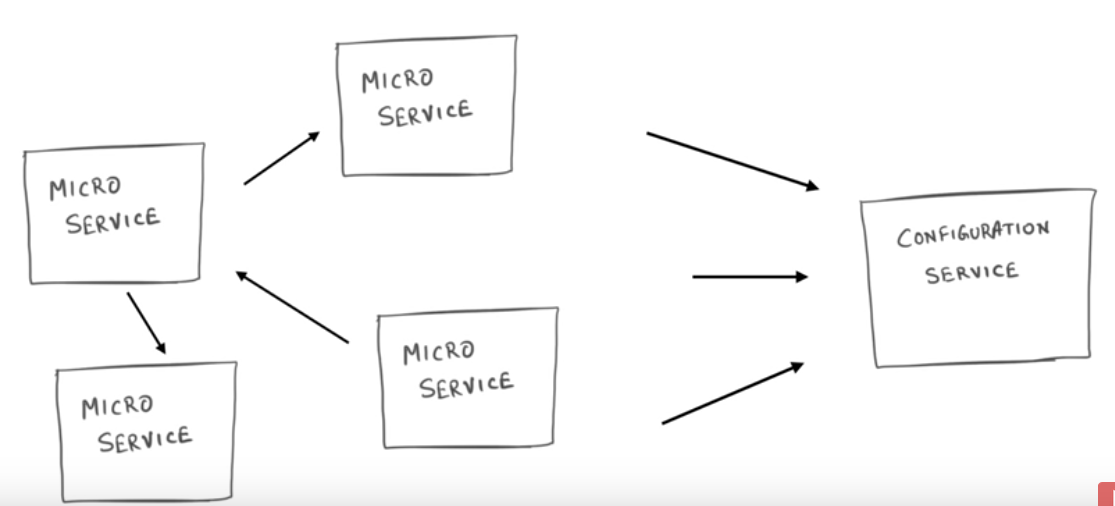




Micro service its important all microservice should be using same configuration file we need to same consistane and reliable config

- it cant be possible if each microservice having its config files

Solution : take out create separate configuration service have that to be single source and let it manage all config things





One source provide that all information

Apache - Zookeeper => very popular solution for maintain centalized config



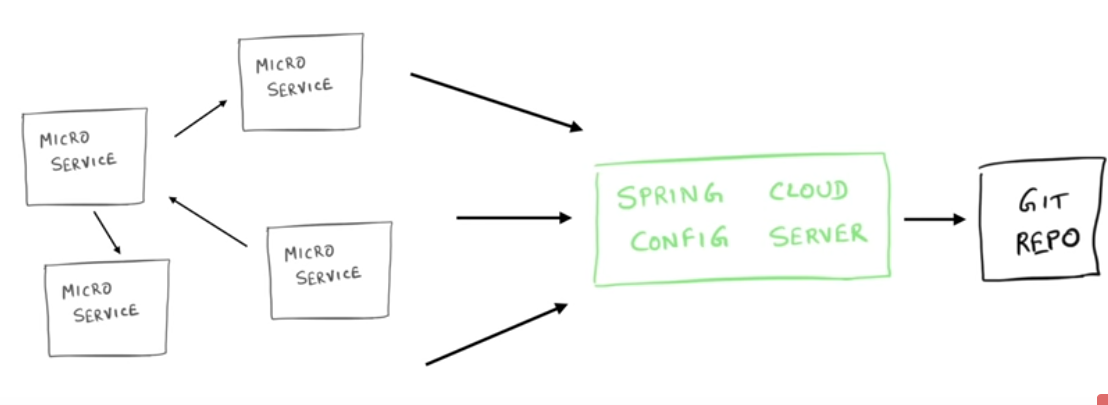
-Spring Cloud Configuration server widly used In context of spring Boot Microservice

Configuration service stroes the configuration it can be in DB/Sourec code source code ask redepoly on change

- we want push to production with out build

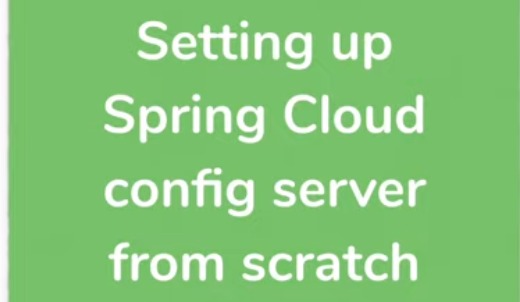
Popular Version controller we can use is Git Repo

Spring Cloud Configuration server connect to Git Repo to get info



When config is in Git Repo and Spring cloud Config server use Git Repo to config Check changes in Git Repo then push to Git Repo same will be used by Config micro services

- make changes in Git Repo all will get it and verison control also taken care



Setting Spring Cloud Server from Scratch

Spring.io

Create boot project

Add config server dependency telling its config server

Build it

In main class add @EnableConfigServer

Now where to pull the value from

@SpringBootApplication

@EnableConfigServer

public class SpringCloudConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(SpringCloudConfigServerApplication.class, args);

}

}

Mention from where to pull from Git

Application.properties

spring.cloud.config.server.git.uri=${HOME}/code/configrepo

We can use git hub in live repo are we can create a local repo and use

Inside a folder

Run this commands





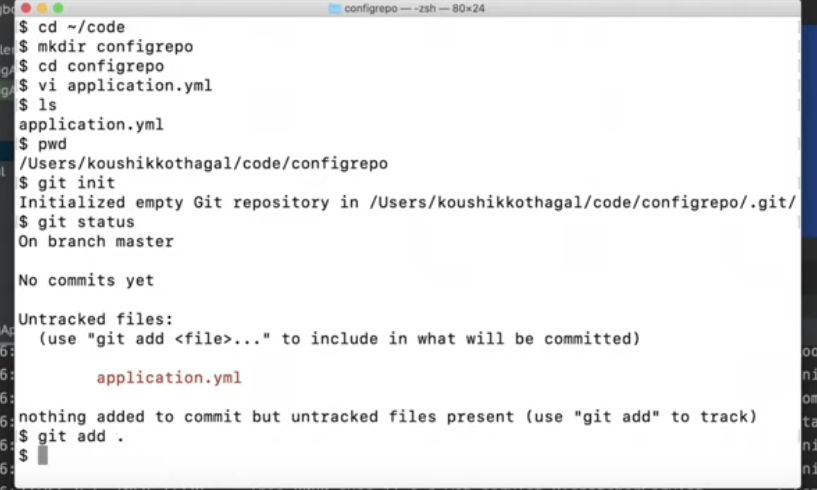
Now commit this code to local git repo

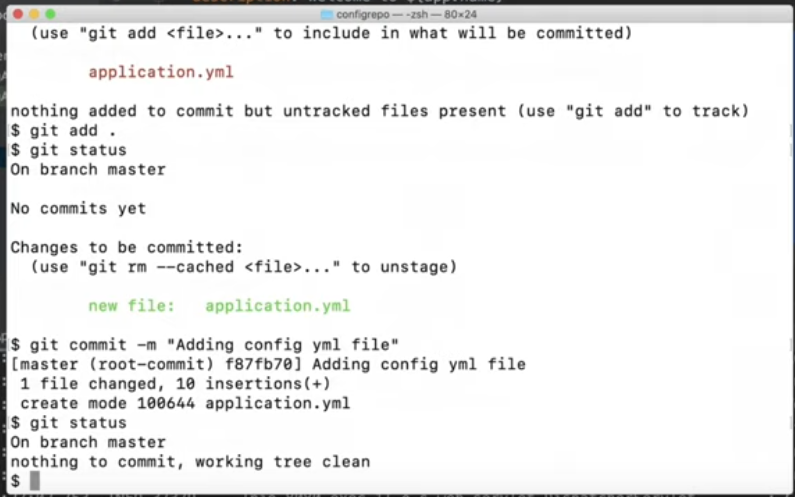
git init : to intialize the git empty repo

git status

git add .

git commit -m “add config file”



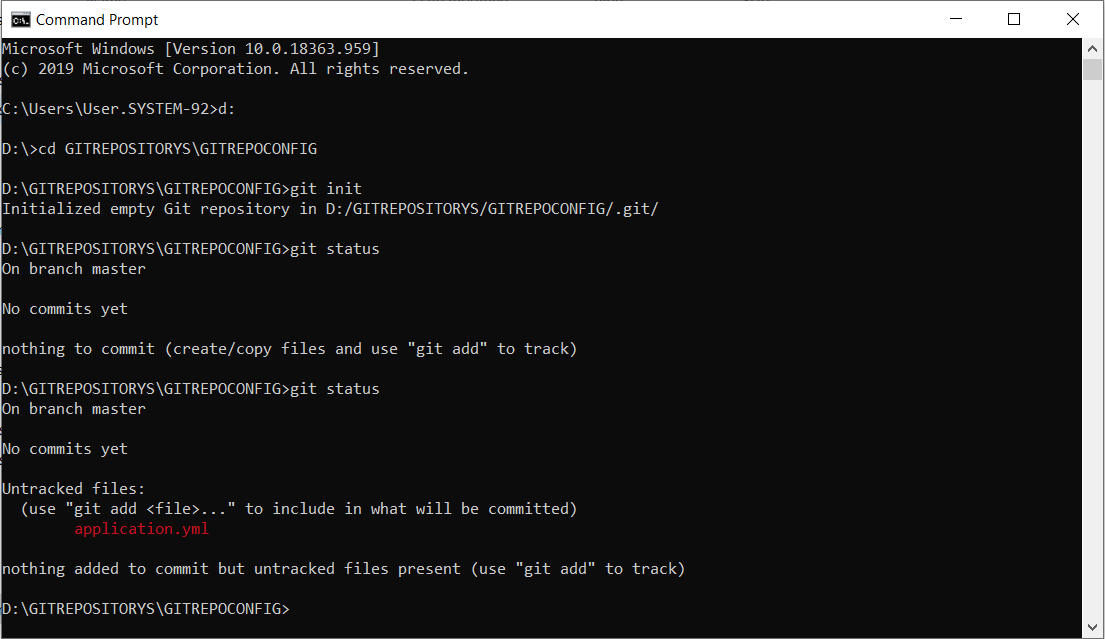


Now I can add the location of these local repo in applcation.properties

${HOME} is the system propeties for home location for unix

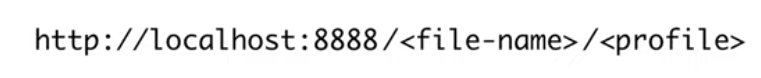
If we want secify file directly specify path it knows file protocol

spring.cloud.config.server.git.uri=${HOME}/code/configrepo



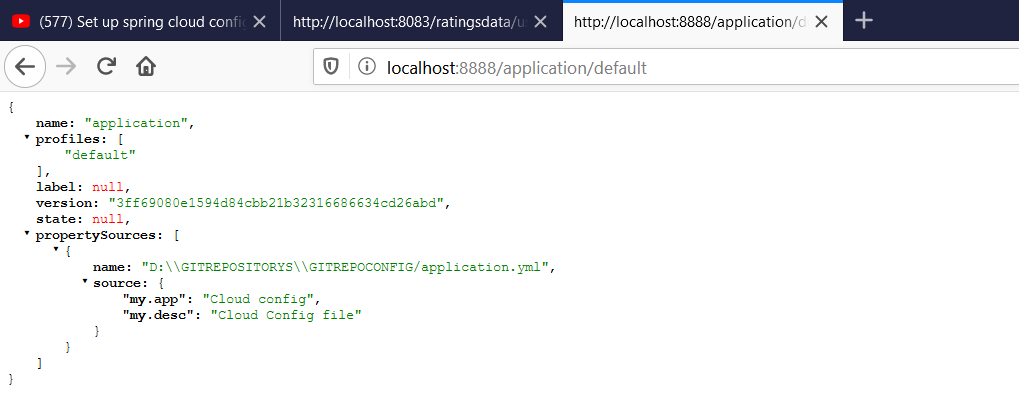


Whats the URL?





Just file name no extension

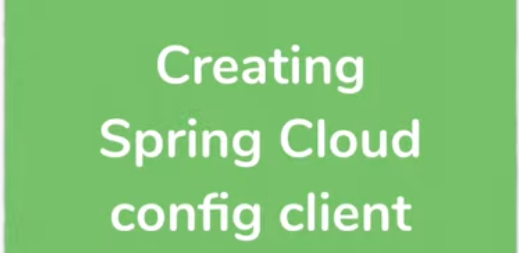


We get application property which is commited thanks to spring cloud I can access it easily I can make cahnge to yml file and commit it

Now spring cloud can talk to all microservices

Here no need to deploy that into prodcution just make change in git and commit it version will be easy

Now we need to config micro service to consume it cloud server



- a micro service connect to spring cloud config server get connection from there

Add microserviice project and add dependecny : config client

<dependency>

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

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

</dependency>

<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>

Now wr to look fr yml file

In applcation file add

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

Pull up the values from the application file

We need configuration dependent on microservice

Ex I need specpfic port for speacfic microservice not same generice

Create yml file in name of microservice add data



#to know name of microservice same as yml file name

#when app connects to spring cloud server Spring cloud repo look if any yaml file with same name as application then pick

spring.application.name=moviecatlog

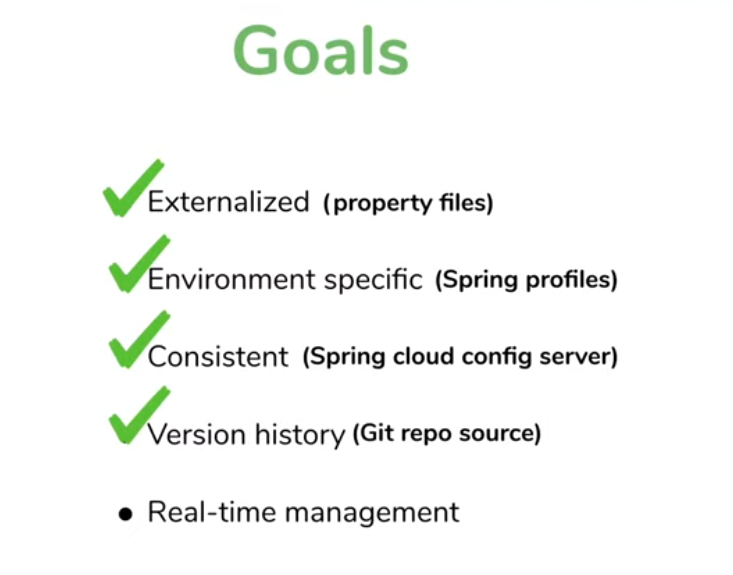
spring.application.name=movieinfo

spring.application.name=movierating

Above lines to be added in respective applciation.properties

And its dependcy in pom.xml

Generic put in yml file microservice specfic palce microservice file



So using Spring Cloud Server Config achived single configuration to all multiple servers via Git So version controlling is achived

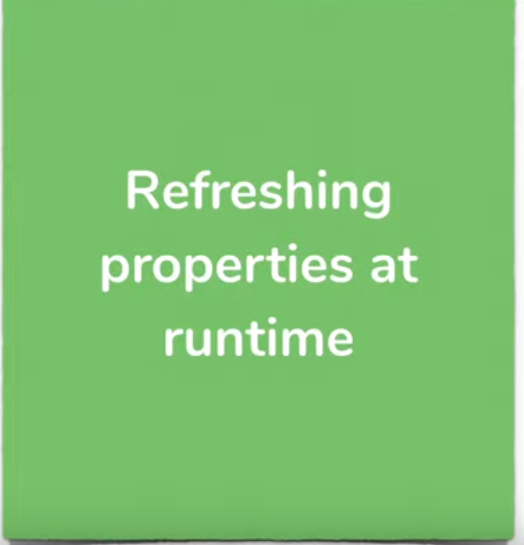
Application.properties will have common config for specfic Application related config can add in file starts with app name and same can be configued in propeties file

Make sure Commit is imp once content changed in Git Config files and Dependency add in all application

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

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

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



Real time Management

- refreshing properties at run time

- one thing is skipped in all above part changes

- make changes and sumbit to git repo we have to start microservice bcz propetie files are read when application statred

- we want new cahnges to be updated with out restating the server---- how by enabling refrshing in micro services

- change something in my config file fr example movierating.yml with new value and don’t commit

- call service of cloud config server

http://localhost:8888/movierating/default

- still changed not reflected commit the file to git repo then check

- now its showing the effected data with out restarting cloud config server

- but same value will not reflect the microservice appplication ex: movie-rating-service

- bcz Cloud config server always look fr current value always check fr the value in git repo no need restart in this case takes data from GIT repo

- challange is client to get updated bcz it cant always look for something like anything changed in config files properties files its not the proper way it takes process time and all in other hand Cloud config server is configured in a purpose to always check the changes in config properties

- so we need to restart the microservice app to get effected so alternate way to handle this is

We have dependency called actuator in pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

Add the above dependency and expose all endpoints are reuired one is also enough

- one the endpoint to call to to refresh it need to make post request to that endpoint - saying new changes in config refresh and get new config from cloud server

- even this will not fix all it will only applicable it any Class is annotated with @RefreshScopeso only that class will be refreshed if its using any data frm config Server and this annotaion helps to refresh bean

- ex below it make only changes to that class which have propeties data getting from Config server

@RestController

@RefreshScope

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

public class RatingResource {

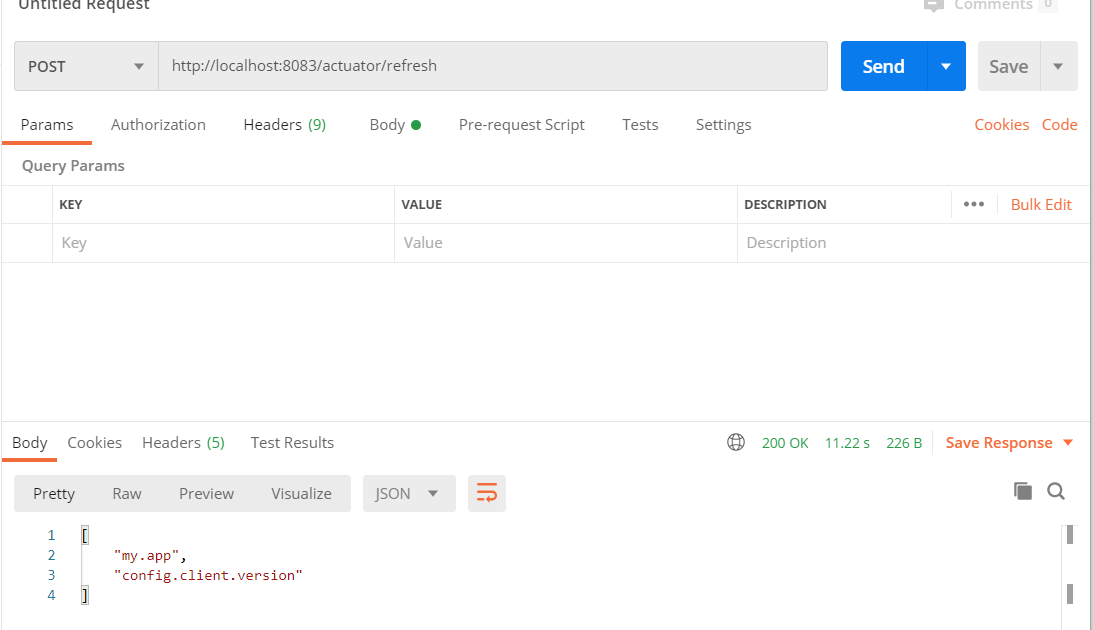
@Value("${my.app}")

private String app;

@Value("${my.desc}")

private String desc;

- using postman will make POST request to that endpoint automatically class with @refreshScope will refreshed and effected values will effect to it ex below it make only changes



Above is the endpoint refresh it has to be post request then which key is modified will get has response also

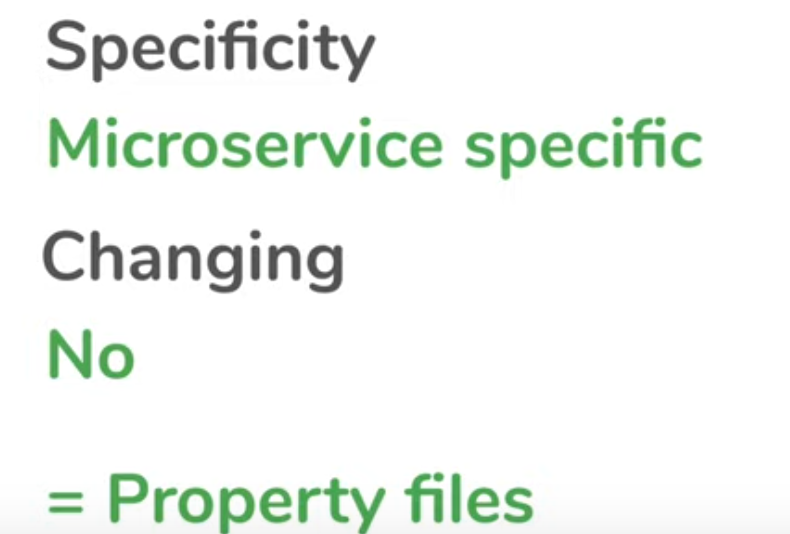
- we have achived dynamic config in live



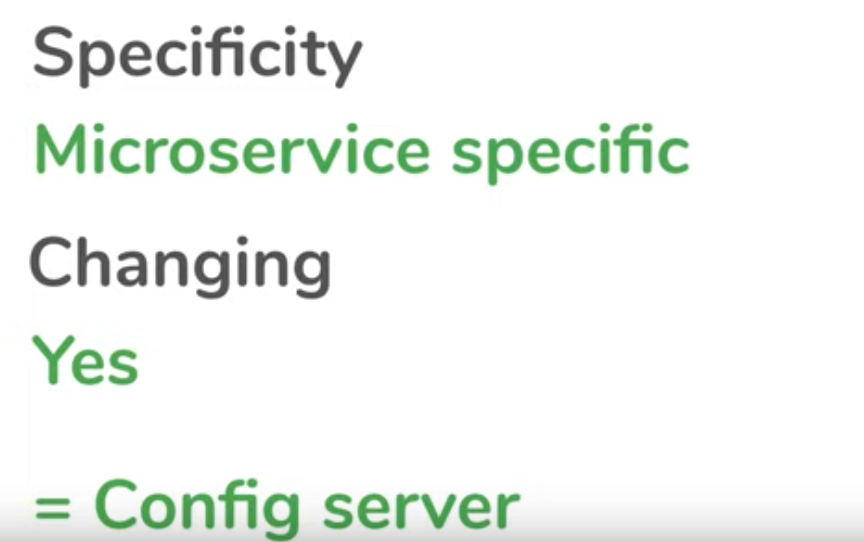
Strategis we need to follow for effective micro service config



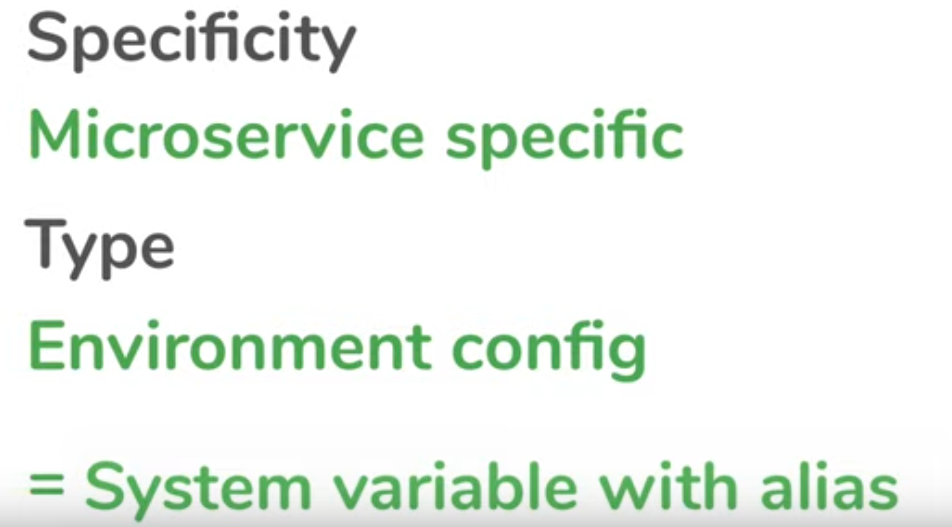
1 If ur not going to change the some spefic config information then keep it in property files



2 IF it micro service specfic and likly to be changes often keep it in config server rather then property files

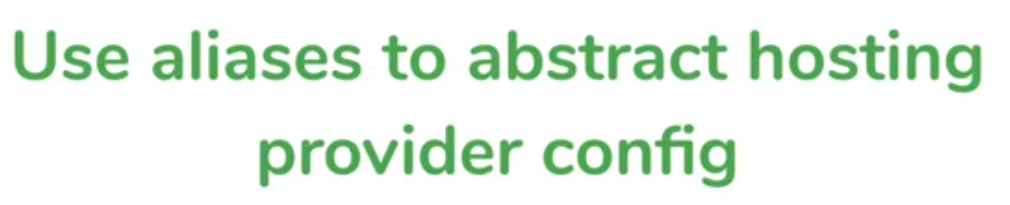


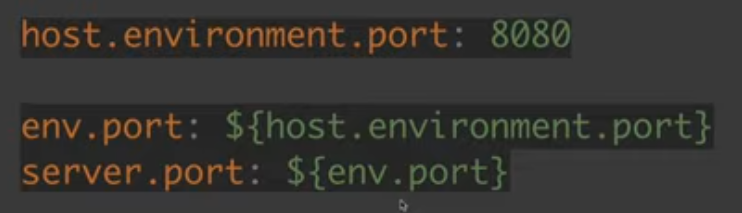
3 PATH variable / System variable





${HOME} = >system Variable fr home path





Security to Spring cloud config servers

Sol 1: using Spring security

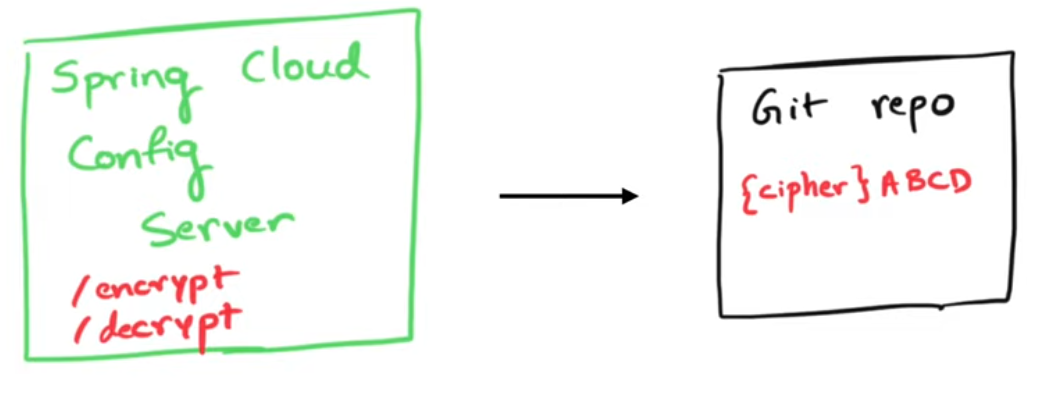


Sol 2: using encryption Spring cloud server will do encryption to the values in Git Repo and Do decryption while fecthings so actual value will not be visible in GIT Repo it will decrypted when it accessed

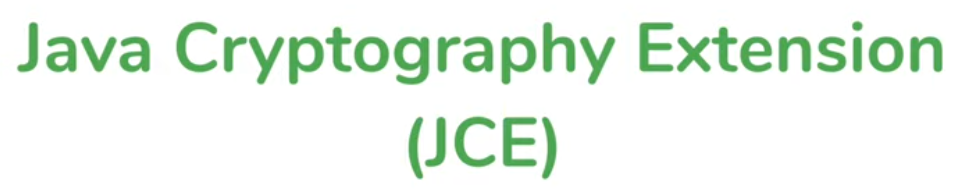


When we open config file in repo it looks like below cnt show orginal data

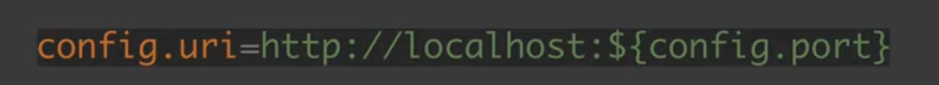




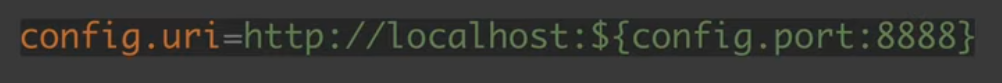
This will be responsible for Encryption and Decryption







Default value using : “then value I.e port no” ex ${config.port:8888} it will set 8888 as port if the config.port key not found in config



12 factors need to be looked while building Microservices

Please refefer this link for more details about 12 factors.