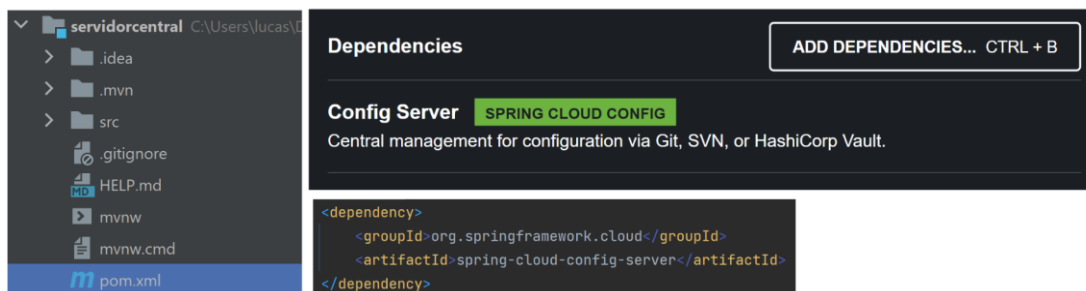


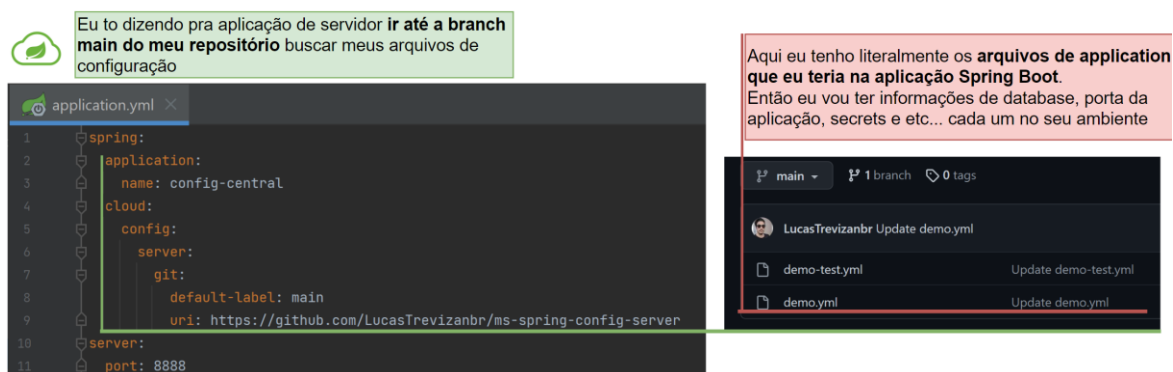
Spring Cloud Config - Server

A ideia desse projeto é fornecer suporte para **configurações externalizadas em ambientes distribuídos**, quando eu falo de **configurações** eu estou me referindo ao **application.properties/yml** de um projeto Spring Boot.

O **servidor centralizado** de configurações é simplesmente uma **aplicação Spring** que carrega a dependência de “config server”:



Algumas configurações são necessárias aqui, por exemplo, da onde eu vou pegar os **application.yml/properties** : (não esquecer a anotação **@EnableConfigServer**):



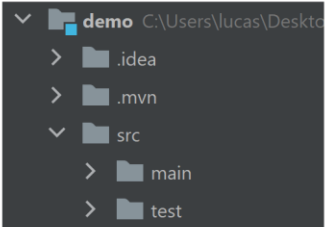
Agora eu só preciso **rodar esse projeto centralizador** e fazer **requests seguindo a convenção que eu defini**, no caso, eu vou **chamar o nome do application.yml / o ambiente de prefixo** e obter um **Json** com suas **configurações respectivas**:



Spring Cloud Config – Client

Beleza, o **server config** está com as configurações **centralizadas e expõe elas através de endpoints específicos**. Mas eu preciso que meus micros serviços usem essas configurações.

Para que um micro serviço seja um cliente do servidor de configuração ele precisa de uma dependência de (adivinha só) **“config client”**:



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

Dependencies


ADD DEPENDENCIES...

Config Client

SPRING CLOUD CONFIG

Client that connects to a Spring Cloud Config Server to fetch the application's configuration.

A maneira mais simples de fazer a aplicação buscar as configurações do servidor de configuração é utilizando duas propriedades do Spring:



O import especifica o endereço onde a aplicação de "config server" está rodando, com base nisso a aplicação cliente busca as informações

O name é extremamente importante, pois aqui definimos o nome da aplicação para que ela saiba qual arquivo buscar

```
spring:
  config:
    import: optional:configserver:http://localhost:8888
  application:
    name: demo
server:
  port: 8080
```

```
main] o.s.c.c.c.ConfigServerConfigDataLoader : Fetching config from server at http://localhost:8888
main] o.s.c.c.c.ConfigServerConfigDataLoader : Located environment: name=demo, profiles=[default], label=null, version=fe2131ae0172c969ddaf0a864350b03f30ff5c904, state=null
```

Com as propriedades sendo buscadas do servidor de configuração é necessário que o código consiga acessar a elas.

Existem algumas maneiras de fazer isso, basicamente é tu usar o **application.yml/properties** na sua aplicação, então dá pra usar o valor injetado com **@Value**:

Pegando o valor direto do arquivo, usando como referencia o nome da propriedade

```
@RestController
public class ControllerTest {

    private final DemoPropertiesSourceConfig demoPropertie

    @Value(value = "${test.config}")
    private String valorDaPropertieInjetadaEmAtributo;

    @GetMapping("/testandoSoAtributo")
    public String testaLegalDogLoko() {
        return valorDaPropertieInjetadaEmAtributo;
    }
}
```

localhost:8080/testandoSoAtributo

Profile padrão

Da pra usar um **Objeto específico** que representa a configuração e então injeta-lo em algum lugar:

Na anotação podemos configurar o prefixo das propriedades que queremos.
Nos atributos referenciamos ESPECÍCAMENTE os atributos de configuração daquele prefixo

```
@Component
@ConfigurationProperties("test")
public class DemoPropertiesSourceConfig {

    private String Config;

    public DemoPropertiesSourceConfig() {
    }

    public String getConfig() { return Config; }

    public void setConfig(String config) { Config = config; }
}
```

```
@RestController
public class ControllerTest {

    private final DemoPropertiesSourceConfig demoPropertiesSourceConfig;

    @Value(value = "${test.config}")
    private String valorDaPropriedadeInjetadaEmAtributo;

    public ControllerTest(DemoPropertiesSourceConfig demoPropertiesSourceConfig) {
    }

    @GetMapping("/testandoLegalDog")
    public String testaLegalDog(){
        return demoPropertiesSourceConfig.getConfig();
    }
}
```

← → ↺ 🏠 ⓘ localhost:8080/testandoLegalDog

Profile padrão