

Lab 10

Part 1

Write a ProductCommandService where you can add, delete and update products. Products have a productNumber, name and price.

Write a StockCommandService where you can add, delete and update the number of products we have in stock. A Stock class has the attributes productNumber, quantity.

Write a ProductQueryService where you can retrieve products. Products have a productNumber, name, price and numberInStock.

Give all services their own Mongo collection.

Make sure that if products or quantity in stock is changed, this will show up in the products we get back from the ProductQueryService service.

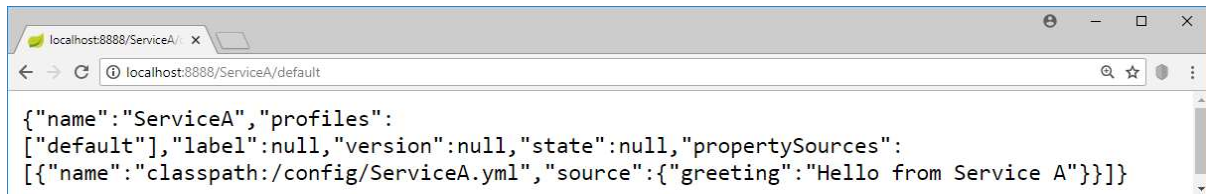
Part 2

Given are the projects ConfigServer, ServiceAApplication and ServiceBApplication

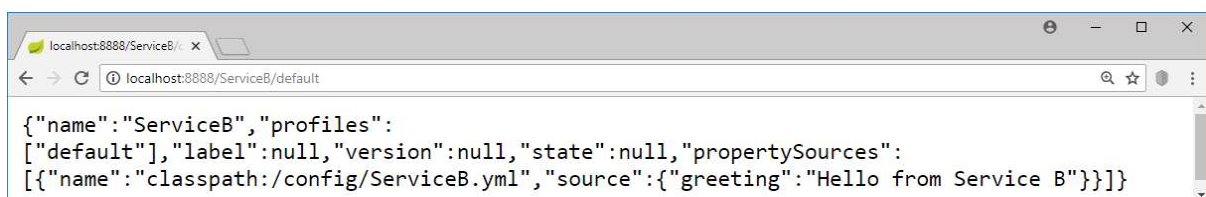
First run the ConfigServer and check if it works:

We can check if the configserver works correctly with the url :

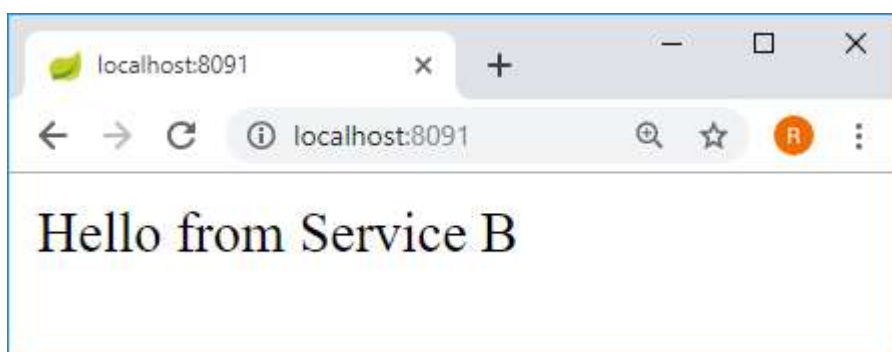
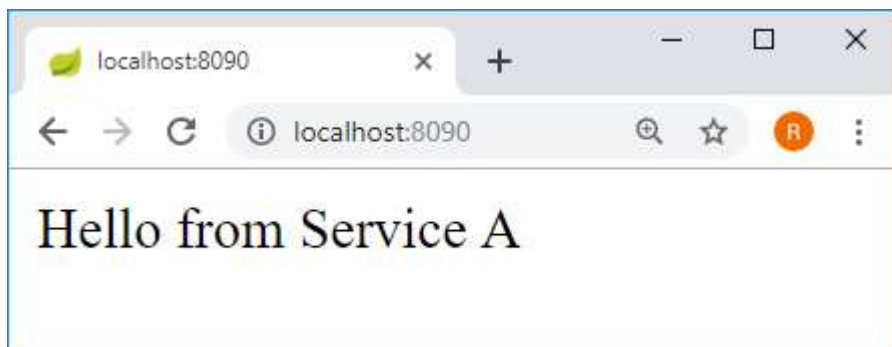
<http://localhost:8888/ServiceA/default>



And <http://localhost:8888/ServiceB/default>



Then run ServiceAApplication and ServiceBApplication and test if the applications work correctly:

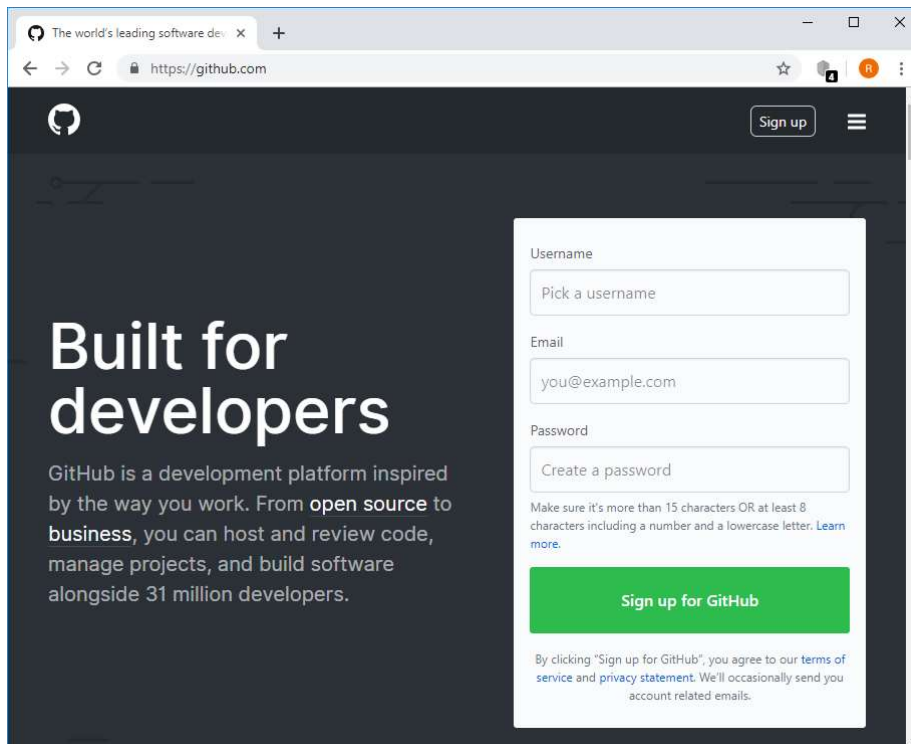


Modify the configuration in the ConfigServer and check if this modification is shown in ServiceAApplication and ServiceBApplication (You have to restart the applications)

Part 3

First we need to create a GitHub account

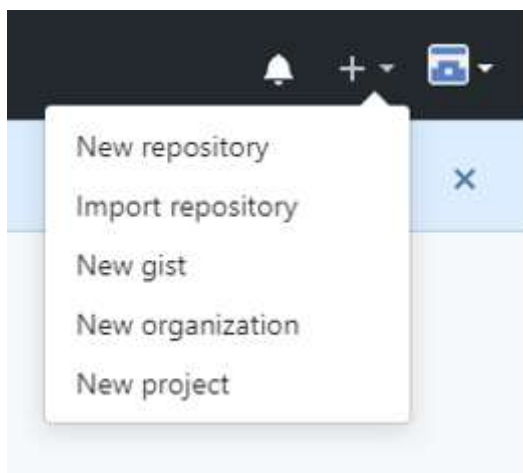
Go to <https://github.com/> and Sign up for a free GitHub account.

A screenshot of the GitHub website's sign-up page. The page has a dark blue header with the GitHub logo on the left and a 'Sign up' button on the right. The main content area is dark blue with the text 'Built for developers' in large white font. Below this, it says 'GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 31 million developers.' On the right side, there is a white sign-up form with fields for 'Username' (placeholder: 'Pick a username'), 'Email' (placeholder: 'you@example.com'), and 'Password' (placeholder: 'Create a password'). Below the password field, there is a note: 'Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)' At the bottom of the form is a green button that says 'Sign up for GitHub'. Below the button, there is a small disclaimer: 'By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy statement](#). We'll occasionally send you account related emails.'

Make sure you remember your password.

Once you have a github account, we first create a new repository.


In the upper right corner, next to your avatar or identicon, click **+** and then select **New repository**.



Create a new repository

A repository contains all project files, including the revision history.

Owner

 renespring ▾

Repository name *

springcloud ✓

Great repository names are short and memorable. Need inspiration? How about **didactic-pancake**?

Description (optional)

Repository for the spring cloud training



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾



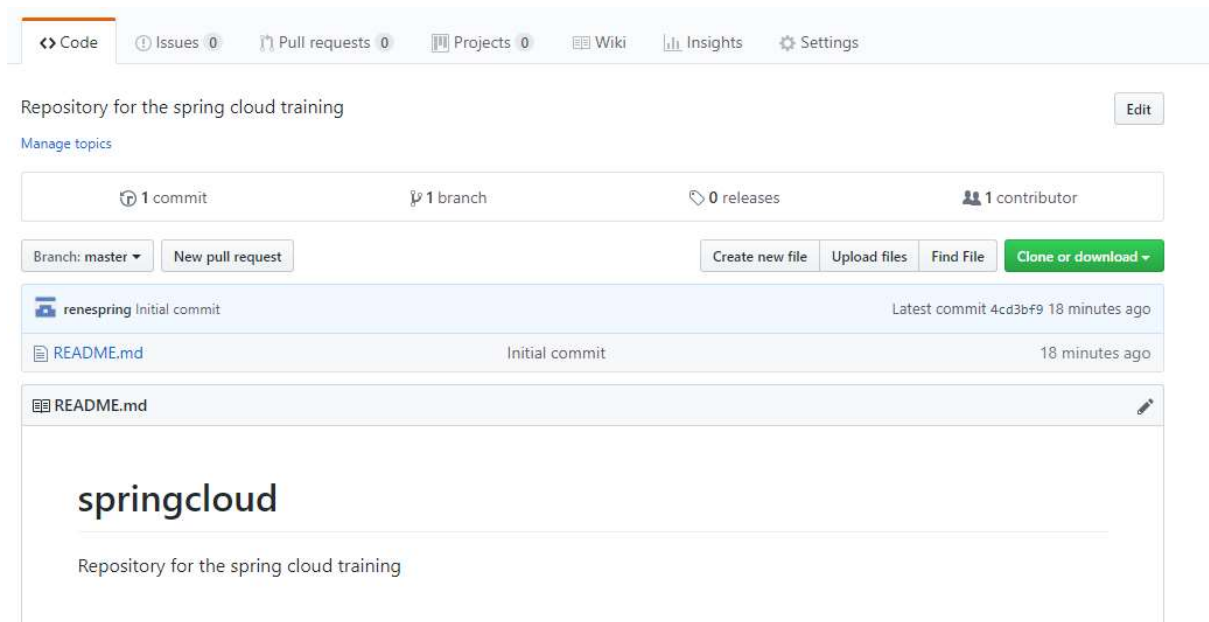
Create repository

Name your repository **springcloud**.

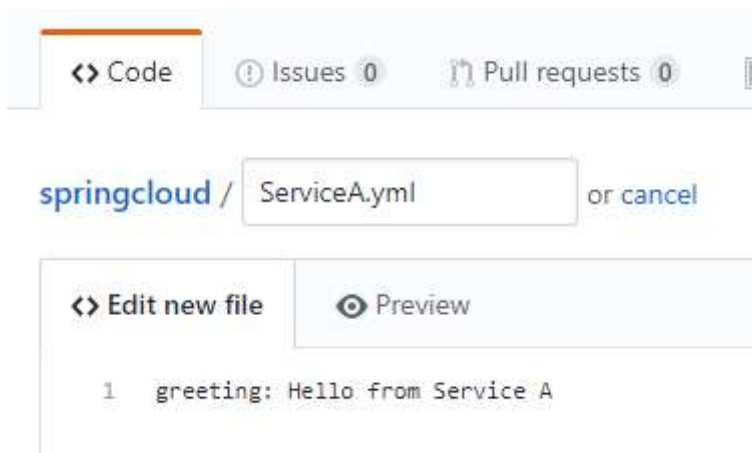
Write a short description.

Select **Initialize this repository with a README**

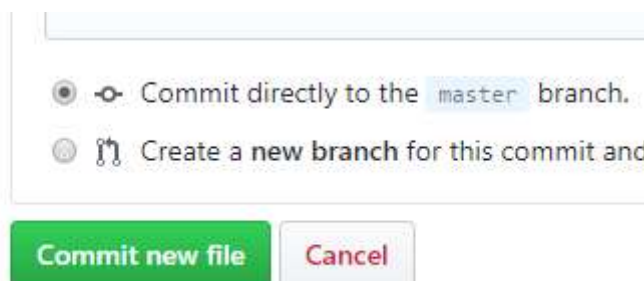
Click **Create Repository**



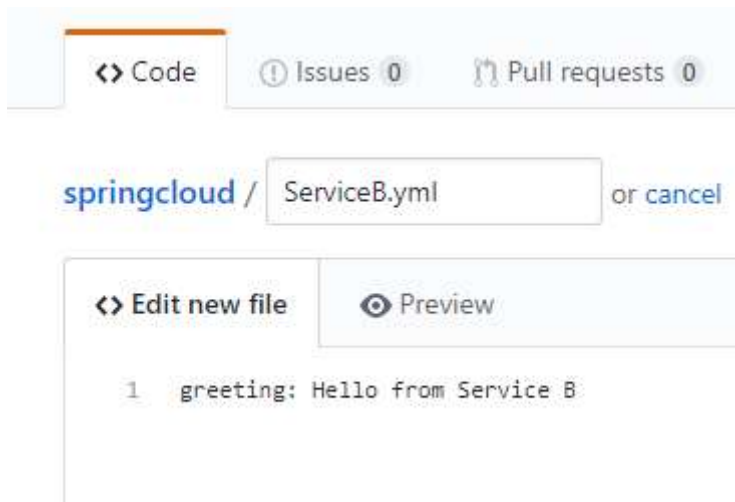
Click the **Create new file** button.



Name the file **ServiceA.yml** and enter the text
greeting: Hello from ServiceA



Then click the **Commit new file** button.



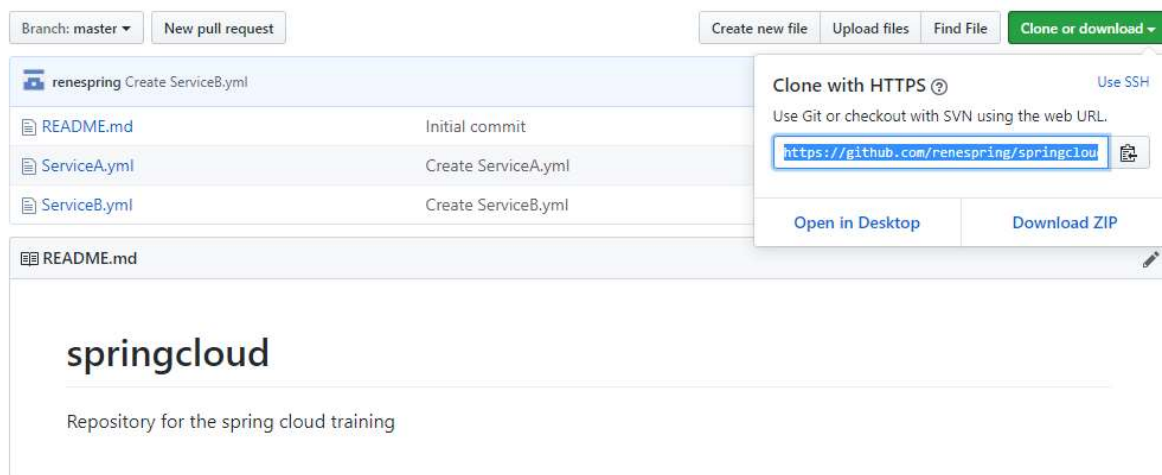
In a similar way, create a **ServiceB.yml** file.

We have now 2 yml files:

renespring Create ServiceB.yml	
README.md	Initial commit
ServiceA.yml	Create ServiceA.yml
ServiceB.yml	Create ServiceB.yml

Now we need to change application.properties from the ConfigServer so that it uses the GIT repository instead of the local file repository.

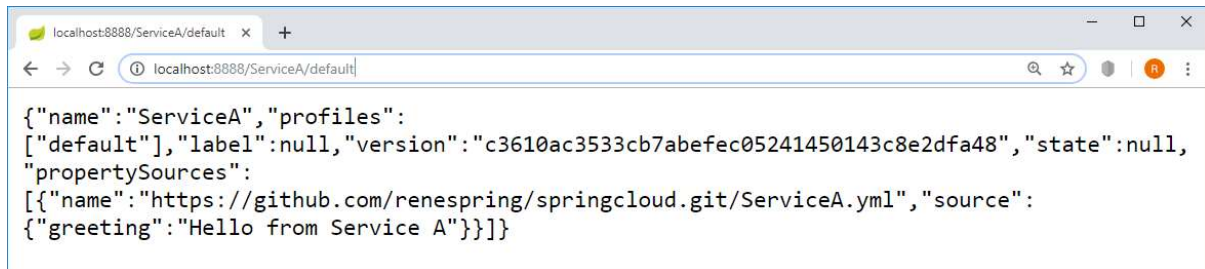
We can get the URL to our git repository by clicking the **Clone or download** button:



Change application.properties so that the property: spring.cloud.config.server.git.uri points to your git repository

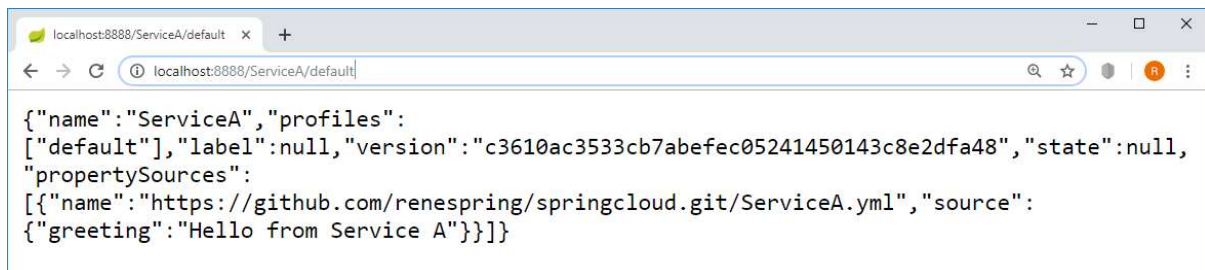
```
server.port=8888  
  
spring.cloud.config.server.git.uri=https://github.com/renespring/springcloud.git
```

Now start (or restart) the ConfigServer and check if it works correctly:



A screenshot of a web browser window with the address bar showing 'localhost:8888/ServiceA/default'. The main content area displays a JSON configuration for 'ServiceA' with a default profile. The configuration includes a version, state, and a property source pointing to a GitHub repository. The 'greeting' property is set to 'Hello from Service A'.

```
{  
  "name": "ServiceA",  
  "profiles": [  
    "default",  
    {  
      "label": null,  
      "version": "c3610ac3533cb7abefec05241450143c8e2dfa48",  
      "state": null,  
      "propertySources": [  
        {  
          "name": "https://github.com/renespring/springcloud.git/ServiceA.yml",  
          "source": {  
            "greeting": "Hello from Service A" }  
        }  
      ]  
    }  
  ]  
}
```



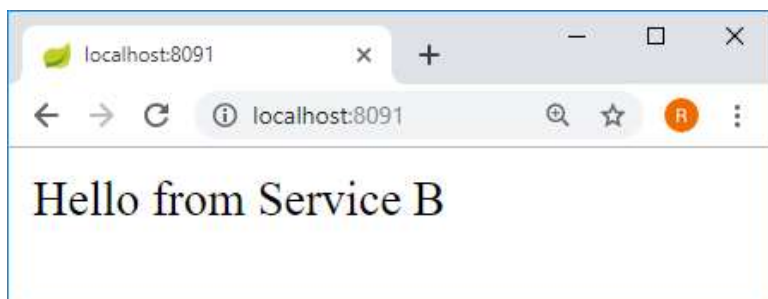
A second screenshot of the same web browser window, showing the same JSON configuration for 'ServiceA' as above.

```
{  
  "name": "ServiceA",  
  "profiles": [  
    "default",  
    {  
      "label": null,  
      "version": "c3610ac3533cb7abefec05241450143c8e2dfa48",  
      "state": null,  
      "propertySources": [  
        {  
          "name": "https://github.com/renespring/springcloud.git/ServiceA.yml",  
          "source": {  
            "greeting": "Hello from Service A" }  
        }  
      ]  
    }  
  ]  
}
```

Also check if ServiceA and ServiceB are still working correctly:

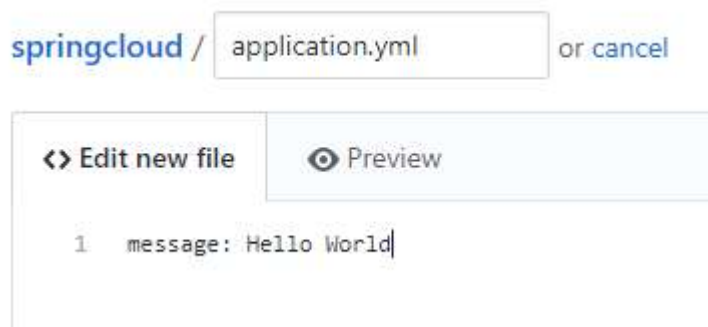


A screenshot of a web browser window with the address bar showing 'localhost:8090'. The main content area displays the text 'Hello from Service A'.

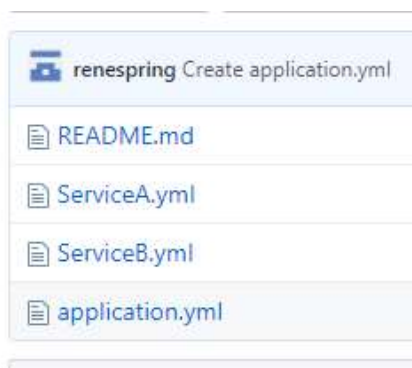


A screenshot of a web browser window with the address bar showing 'localhost:8091'. The main content area displays the text 'Hello from Service B'.

In GitHub, create a new file called **application.yml** and enter the **message** property:



We now have 3 configuration files:



In ServiceAApplication, change the controller as follows:

```
@RestController
@RefreshScope
public class ServiceAController {
    @Value("${greeting}")
    private String greeting;

    @Value("${message}")
    private String message;

    @RequestMapping("/")
    public String getName() {
        return message+" , "+greeting;
    }
}
```

Do the same for ServiceBApplication, and restart the services. Check now if the shared configuration **message** is picked up by both services:

What to hand in?

1. A zip file containing all services for part 1
2. A zip file containing all services for part 2
3. Write a readme.txt file with the following statement and sign with your name:

I hereby declare that this submission is my own original work and to the best of my knowledge it contains no materials previously published or written by another person. I am aware that submitting solutions that are not my own work will result in an NC of the course.

[your name as signature]