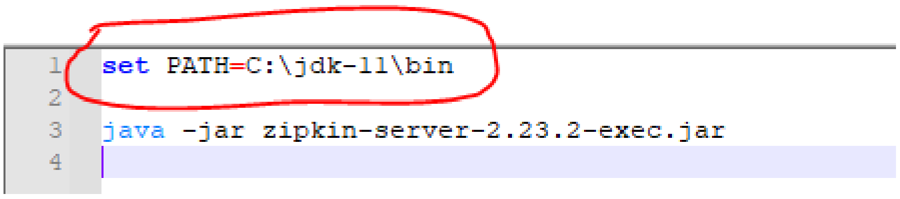
Lab 9

Part 1

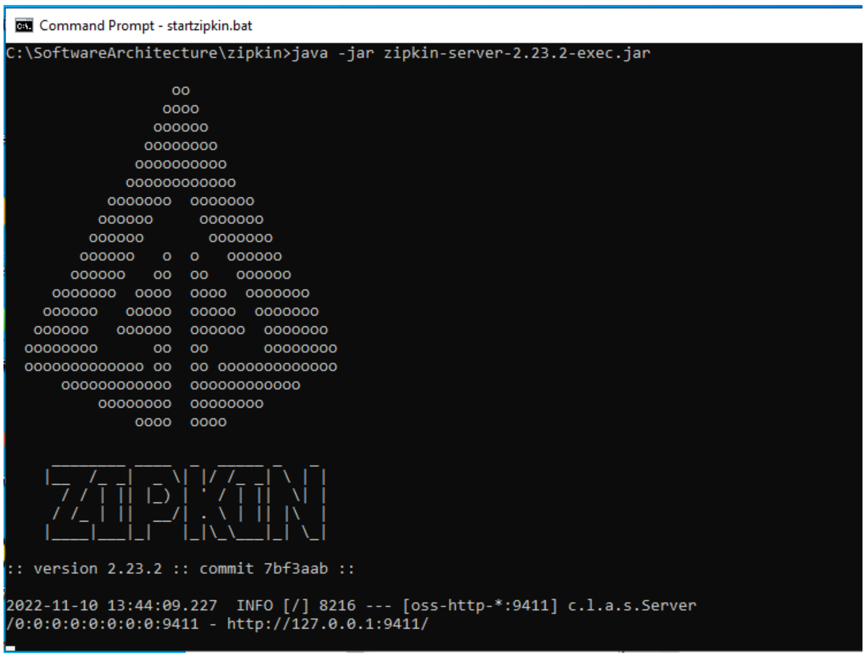
First open the file C:\SoftwareArchitecture\zipkin\startzipkin.bat:



```
set PATH=C:\jdk-11\bin

java -jar zipkin-server-2.23.2-exec.jar
```

Change the first line as follows: **set PATH=C:\jdk-11\bin**

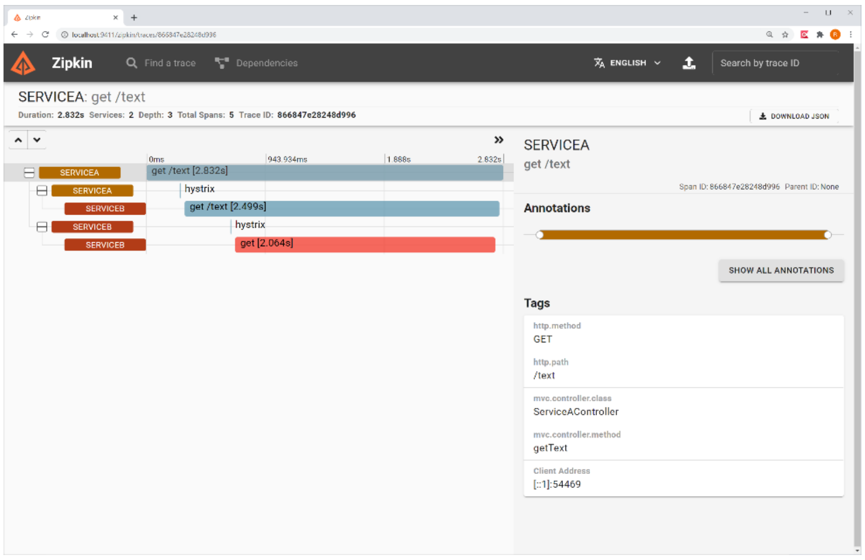
Then run the file C:\SoftwareArchitecture\zipkin\startzipkin.bat



```
C:\SoftwareArchitecture\zipkin>java -jar zipkin-server-2.23.2-exec.jar

:: version 2.23.2 :: commit 7bf3aab ::

2022-11-10 13:44:09.227  INFO [/] 8216 --- [oss-http-*:9411] c.l.a.s.Server
/0:0:0:0:0:0:0:0:9411 - http://127.0.0.1:9411/
```

In the ProductService to the StockService that you wrote in lab 8 add the Zipkin and Sleuth libraries and configuration.
Then open the zipkin console on http://localhost:9411/zipkin

You can click the search button, then see the traces between the services, and you can see the dependencies between the services.



SERVICEA: get /text

Duration: **2.832s** Services: **2** Depth: **3** Total Spans: **5** Trace ID: **866847e28248d996**

```
                0ms            943.934ms        1.888s         2.832s
SERVICEA        get /text [2.832s]
  SERVICEA            hystrix
    SERVICEB           get /text [2.499s]
  SERVICEB                  hystrix
    SERVICEB                 get [2.064s]
```

SERVICEA
get /text

Span ID: 866847e28248d996 Parent ID: None

Annotations

Tags

http.method
GET

http.path
/text

mvc.controller.class
ServiceAController

mvc.controller.method
getText

Client Address
[::1]:54469

Part 2

Elasticsearch

Start elasticsearch by dubble clicking **…\elasticsearch-7.10.0\bin\elasticsearch.bat**

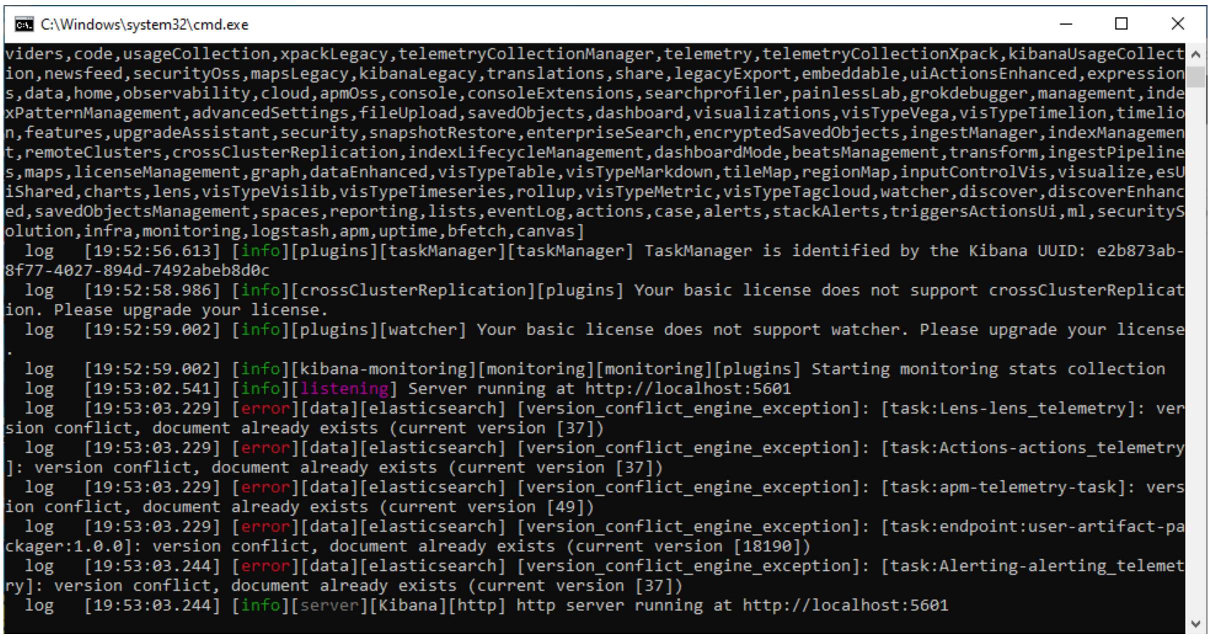
Logstash

Start logstash by dubble clicking **…\logstash-7.10.1\startLogstash.bat**

Kibana

Start kibana by dubble clicking **…\kibana-7.10.0-windows-x86_64\startkibana.bat**

Wait till kibana has started.



```
viders,code,usageCollection,xpackLegacy,telemetryCollectionManager,telemetry,telemetryCollectionXpack,kibanaUsageCollect
ion,newsfeed,securityOss,mapsLegacy,kibanaLegacy,translations,share,legacyExport,embeddable,uiActionsEnhanced,expression
s,data,home,observability,cloud,apmOss,console,consoleExtensions,searchprofiler,painlessLab,grokdebugger,management,inde
xPatternManagement,advancedSettings,fileUpload,savedObjects,dashboard,visualizations,visTypeVega,visTypeTimelion,timelio
n,features,upgradeAssistant,security,snapshotRestore,enterpriseSearch,encryptedSavedObjects,ingestManager,indexManagemen
t,remoteClusters,crossClusterReplication,indexLifecycleManagement,dashboardMode,beatsManagement,transform,ingestPipeline
s,maps,licenseManagement,graph,dataEnhanced,visTypeTable,visTypeMarkdown,tileMap,regionMap,inputControlVis,visualize,esU
iShared,charts,lens,visTypeVislib,visTypeTimeseries,rollup,visTypeMetric,visTypeTagcloud,watcher,discover,discoverEnhanc
ed,savedObjectsManagement,spaces,reporting,lists,eventLog,actions,case,alerts,stackAlerts,triggersActionsUi,ml,securityS
olution,infra,monitoring,logstash,apm,uptime,bfetch,canvas]
  log   [19:52:56.613] [info][plugins][taskManager][taskManager] TaskManager is identified by the Kibana UUID: e2b873ab-
8f77-4027-894d-7492abeb8d0c
  log   [19:52:58.986] [info][crossClusterReplication][plugins] Your basic license does not support crossClusterReplicat
ion. Please upgrade your license.
  log   [19:52:59.002] [info][plugins][watcher] Your basic license does not support watcher. Please upgrade your license
.
  log   [19:52:59.002] [info][kibana-monitoring][monitoring][monitoring][plugins] Starting monitoring stats collection
  log   [19:53:02.541] [info][listening] Server running at http://localhost:5601
  log   [19:53:03.229] [error][data][elasticsearch] [version_conflict_engine_exception]: [task:Lens-lens_telemetry]: ver
sion conflict, document already exists (current version [37])
  log   [19:53:03.229] [error][data][elasticsearch] [version_conflict_engine_exception]: [task:Actions-actions_telemetry
]: version conflict, document already exists (current version [37])
  log   [19:53:03.229] [error][data][elasticsearch] [version_conflict_engine_exception]: [task:apm-telemetry-task]: vers
ion conflict, document already exists (current version [49])
  log   [19:53:03.229] [error][data][elasticsearch] [version_conflict_engine_exception]: [task:endpoint:user-artifact-pa
ckager:1.0.0]: version conflict, document already exists (current version [18190])
  log   [19:53:03.244] [error][data][elasticsearch] [version_conflict_engine_exception]: [task:Alerting-alerting_telemet
ry]: version conflict, document already exists (current version [37])
  log   [19:53:03.244] [info][server][Kibana][http] http server running at http://localhost:5601
```

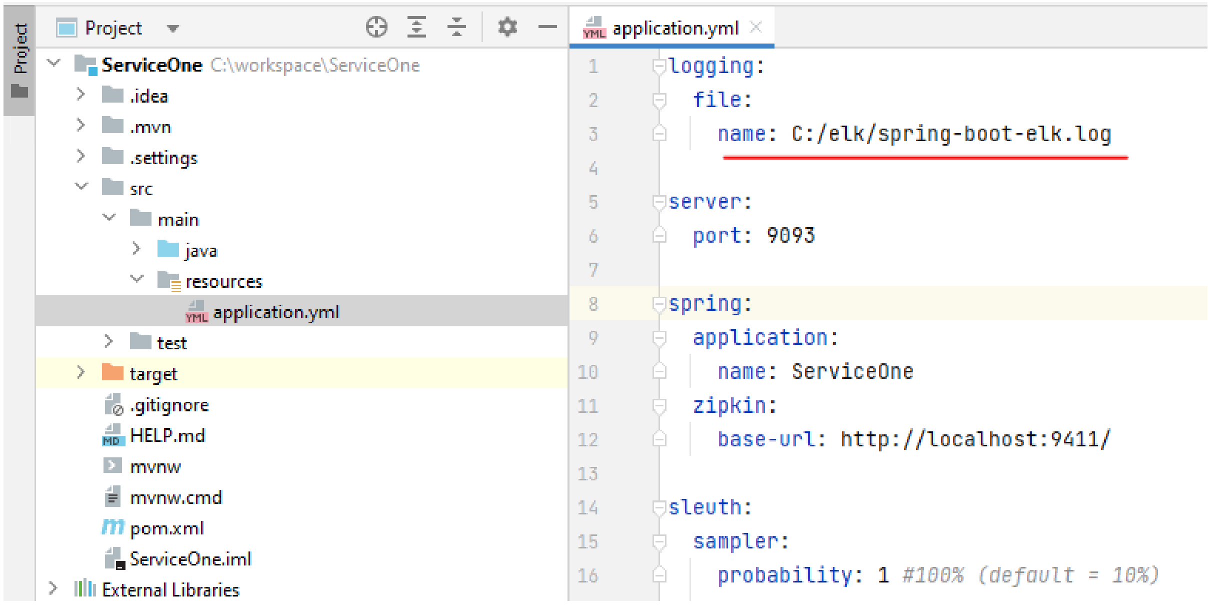
Open the file **…/logstash-6.10.1/logstash.conf:**

```
input {
  file {
    type => "java"
    path => " C:/elk/spring-boot-elk.log"
    codec => multiline {
      pattern => "^%{YEAR}-%{MONTHNUM}-%{MONTHDAY} %{TIME}.*"
      negate => "true"
      what => "previous"
    }
  }
}

output {
  stdout {
    codec => rubydebug
  }
   file {
   path => " C:/elk/testlog.log"
   create_if_deleted => true
  }

  # Sending log events to elasticsearch
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

Logstash will monitor log messages in the file **C:/elk/spring-boot-elk.log** and then write these messages to its console, to the file **C:/elk/testlog.log** and send them to elasticsearch.

Now we have to create a spring boot service that writes log messages to **C:/elk/spring-boot-elk.log**

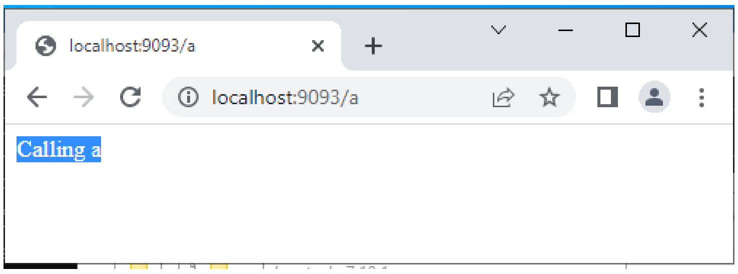
Given is the project **ServiceOne.** Modify applications.yml so that this applications writes the log information in the file **C:/elk/spring-boot-elk.log**



```yaml
logging:
  file:
    name: C:/elk/spring-boot-elk.log

server:
  port: 9093

spring:
  application:
    name: ServiceOne
  zipkin:
    base-url: http://localhost:9411/

sleuth:
  sampler:
    probability: 1 #100% (default = 10%)
```

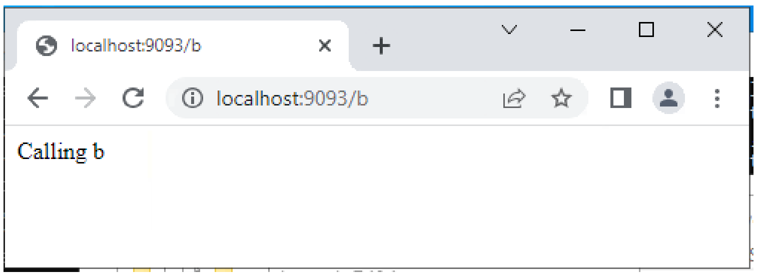
```java
@RestController
public class ServiceOneController {
	private static final Logger logger =
LoggerFactory.getLogger(ServiceOneController.class.getName());

	@RequestMapping("/a")
    public String one() {
        logger.info("Calling a");
        return "Calling a";
    }

	@RequestMapping("/b")
    public String two() {
        logger.debug("Calling b");
        return "Calling b";
    }
}
```

Start the application.



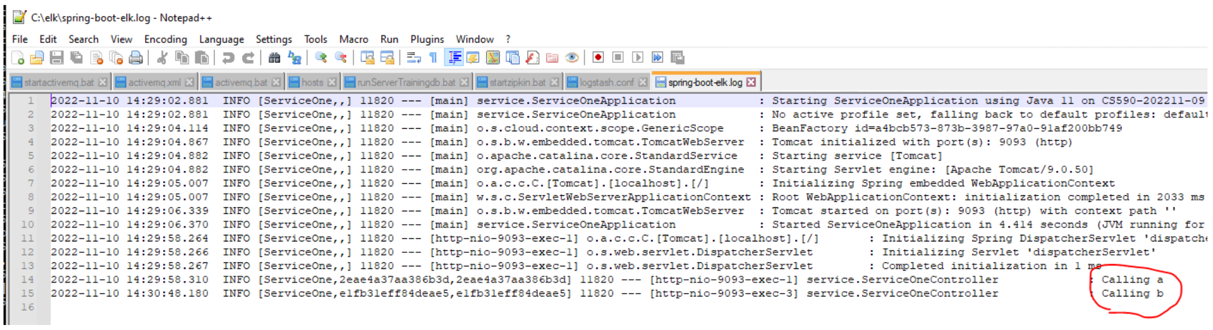
Calling a

Then call the service a few times



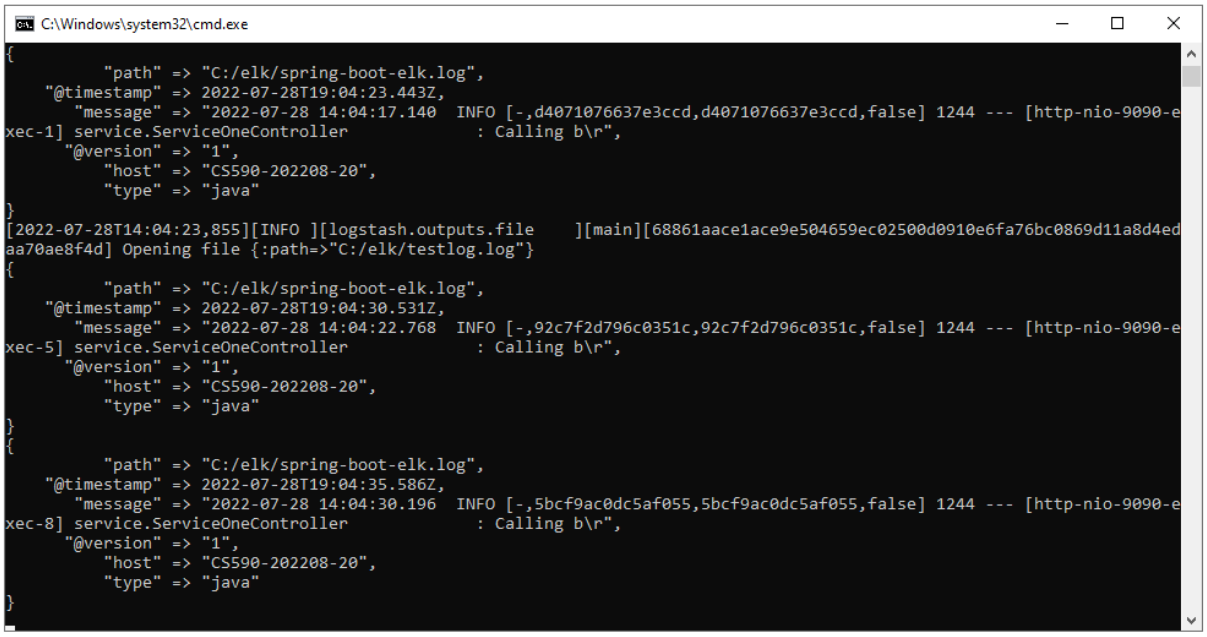
Calling b

Now check the file **C:\elk\spring-boot-elk.log** and check that every REST call write a log line in the log file.


C:\elk\spring-boot-elk.log - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
startactivemq.bat
activemq.xml
activemq.bat
hosts
runServerTrainingdb.bat
startzipkin.bat
logstash.conf
spring-boot-elk.log

```
2022-11-10 14:29:02.881  INFO [ServiceOne,,] 11820 --- [main] service.ServiceOneApplication            : Starting ServiceOneApplication using Java 11 on CS590-202211-09
2022-11-10 14:29:02.881  INFO [ServiceOne,,] 11820 --- [main] service.ServiceOneApplication            : No active profile set, falling back to default profiles: defaul
2022-11-10 14:29:04.114  INFO [ServiceOne,,] 11820 --- [main] o.s.cloud.context.scope.GenericScope     : BeanFactory id=a4bcb573-873b-3987-97a0-91af200bb749
2022-11-10 14:29:04.867  INFO [ServiceOne,,] 11820 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 9093 (http)
2022-11-10 14:29:04.882  INFO [ServiceOne,,] 11820 --- [main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-11-10 14:29:04.882  INFO [ServiceOne,,] 11820 --- [main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.50]
2022-11-10 14:29:05.007  INFO [ServiceOne,,] 11820 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-11-10 14:29:05.007  INFO [ServiceOne,,] 11820 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2033 ms
2022-11-10 14:29:06.339  INFO [ServiceOne,,] 11820 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 9093 (http) with context path ''
2022-11-10 14:29:06.370  INFO [ServiceOne,,] 11820 --- [main] service.ServiceOneApplication            : Started ServiceOneApplication in 4.414 seconds (JVM running for
2022-11-10 14:29:58.264  INFO [ServiceOne,,] 11820 --- [http-nio-9093-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatche
2022-11-10 14:29:58.266  INFO [ServiceOne,,] 11820 --- [http-nio-9093-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2022-11-10 14:29:58.267  INFO [ServiceOne,,] 11820 --- [http-nio-9093-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2022-11-10 14:29:58.310  INFO [ServiceOne,2eae4a37aa386b3d,2eae4a37aa386b3d] 11820 --- [http-nio-9093-exec-1] service.ServiceOneController             : Calling a
2022-11-10 14:30:48.180  INFO [ServiceOne,e1fb31eff84deae5,e1fb31eff84deae5] 11820 --- [http-nio-9093-exec-3] service.ServiceOneController             : Calling b
```

Also you should see that every time a line is added in the log file, logstash will shows this line in the console.

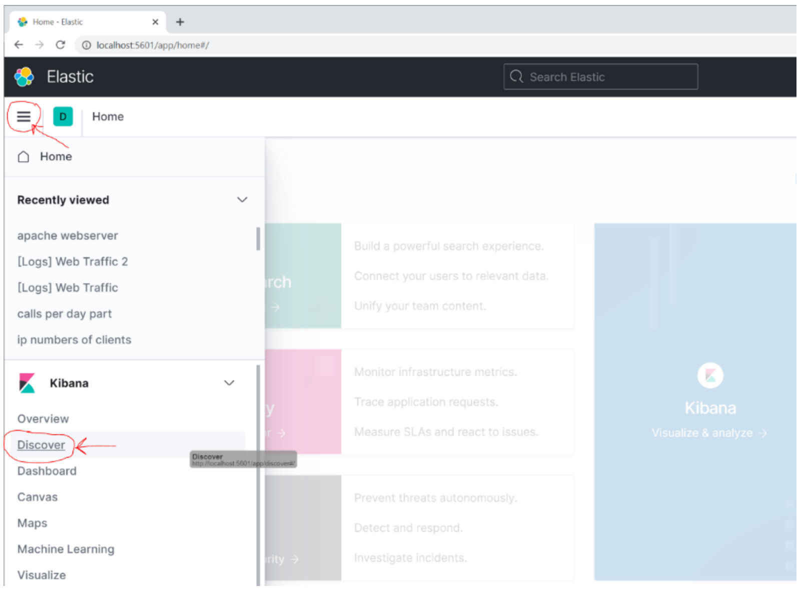


C:\Windows\system32\cmd.exe

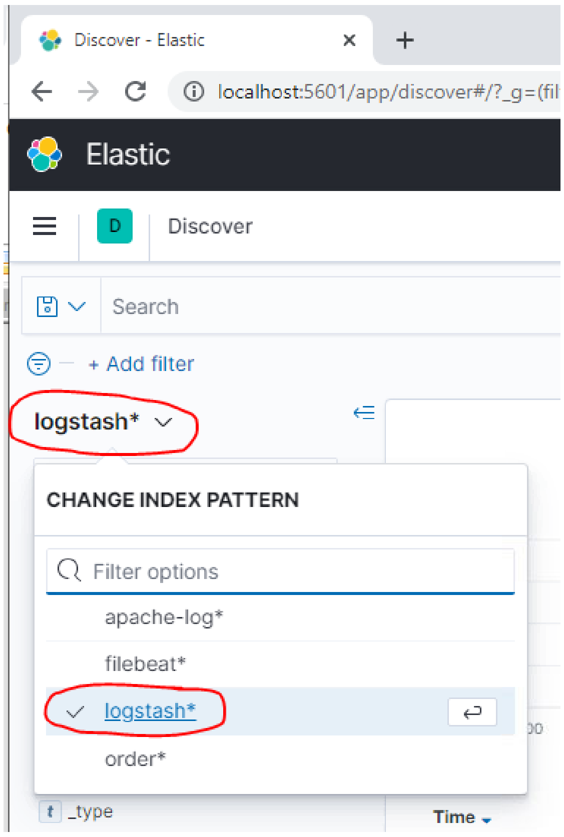
```
{
          "path" => "C:/elk/spring-boot-elk.log",
    "@timestamp" => 2022-07-28T19:04:23.443Z,
       "message" => "2022-07-28 14:04:17.140  INFO [-,d4071076637e3ccd,d4071076637e3ccd,false] 1244 --- [http-nio-9090-e
xec-1] service.ServiceOneController             : Calling b\r",
      "@version" => "1",
          "host" => "CS590-202208-20",
          "type" => "java"
}
[2022-07-28T14:04:23,855][INFO ][logstash.outputs.file    ][main][68861aace1ace9e504659ec02500d0910e6fa76bc0869d11a8d4ed
aa70ae8f4d] Opening file {:path=>"C:/elk/testlog.log"}
{
          "path" => "C:/elk/spring-boot-elk.log",
    "@timestamp" => 2022-07-28T19:04:30.531Z,
       "message" => "2022-07-28 14:04:22.768  INFO [-,92c7f2d796c0351c,92c7f2d796c0351c,false] 1244 --- [http-nio-9090-e
xec-5] service.ServiceOneController             : Calling b\r",
      "@version" => "1",
          "host" => "CS590-202208-20",
          "type" => "java"
}
{
          "path" => "C:/elk/spring-boot-elk.log",
    "@timestamp" => 2022-07-28T19:04:35.586Z,
       "message" => "2022-07-28 14:04:30.196  INFO [-,5bcf9ac0dc5af055,5bcf9ac0dc5af055,false] 1244 --- [http-nio-9090-e
xec-8] service.ServiceOneController             : Calling b\r",
      "@version" => "1",
          "host" => "CS590-202208-20",
          "type" => "java"
}
```

This log record will also be send to elasticsearch. We can check this in kibana.

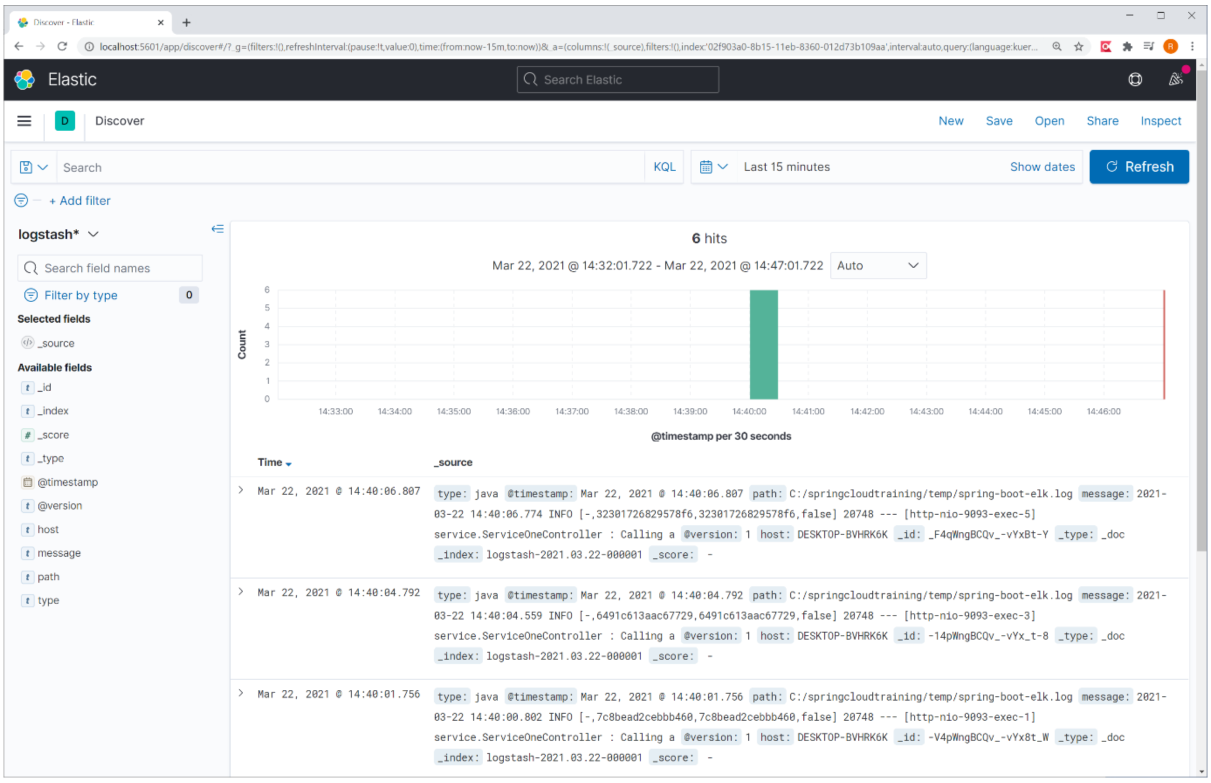
Open the browser on http://localhost:5601



Then click the **Discover** tab



Select the **logstash*** index pattern



```
Time                        _source
Mar 22, 2021 @ 14:40:06.807 type: java @timestamp: Mar 22, 2021 @ 14:40:06.807 path: C:/springcloudtraining/temp/spring-boot-elk.log message: 2021-03-22 14:40:06.774 INFO [-,32301726829578f6,32301726829578f6,false] 20748 --- [http-nio-9093-exec-5] service.ServiceOneController : Calling a @version: 1 host: DESKTOP-BVHRK6K _id: _F4qWngBCQv_-vYxBt-Y _type: _doc _index: logstash-2021.03.22-000001 _score: -
Mar 22, 2021 @ 14:40:04.792 type: java @timestamp: Mar 22, 2021 @ 14:40:04.792 path: C:/springcloudtraining/temp/spring-boot-elk.log message: 2021-03-22 14:40:04.559 INFO [-,6491c613aac67729,6491c613aac67729,false] 20748 --- [http-nio-9093-exec-3] service.ServiceOneController : Calling a @version: 1 host: DESKTOP-BVHRK6K _id: -14pWngBCQv_-vYx_t-8 _type: _doc _index: logstash-2021.03.22-000001 _score: -
Mar 22, 2021 @ 14:40:01.756 type: java @timestamp: Mar 22, 2021 @ 14:40:01.756 path: C:/springcloudtraining/temp/spring-boot-elk.log message: 2021-03-22 14:40:00.802 INFO [-,7c8bead2cebbb460,7c8bead2cebbb460,false] 20748 --- [http-nio-9093-exec-1] service.ServiceOneController : Calling a @version: 1 host: DESKTOP-BVHRK6K _id: -V4pWngBCQv_-vYx8t_W _type: _doc _index: logstash-2021.03.22-000001 _score: -
```

You will now see log records appear on your screen.
You can also click the Auto-Refresh option in the upper hand corner on the right so that Kibana will update itself every 5 seconds.

Now create a new **ServiceTwo** project, and let it write its output to **C:/elk/spring-boot-elk2.log**

Then we have to tell logstash to also monitor this log file. Modify the file: **…\logstash-6.7.0/logstash.conf** so that we also see the logging of ServiceTwo in kibana. Everytime you change logstash.conf you have to restart logstash.

Part 3

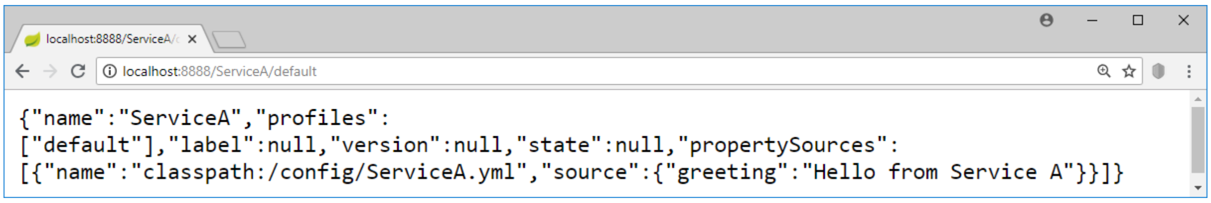
Add the Hystrix circuit breaker to the remote call from the ProductService to the StockService. Test its working.

Part 4

Given are the projects ConfigServer, ServiceAApplication and ServiceBApplication

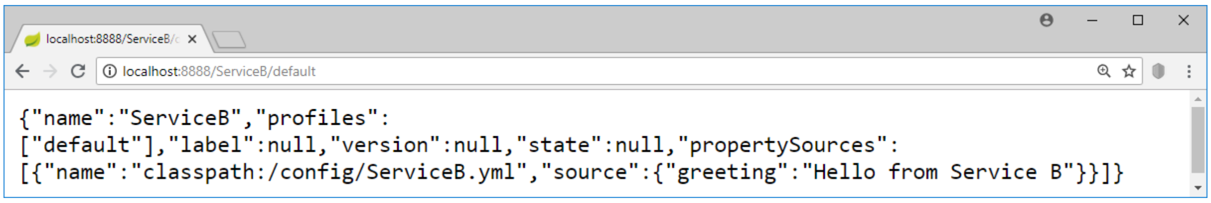
First run the ConfigServer and check if it works:

We can check if the configserver works correctly with the url : http://localhost:8888/ServiceA/default



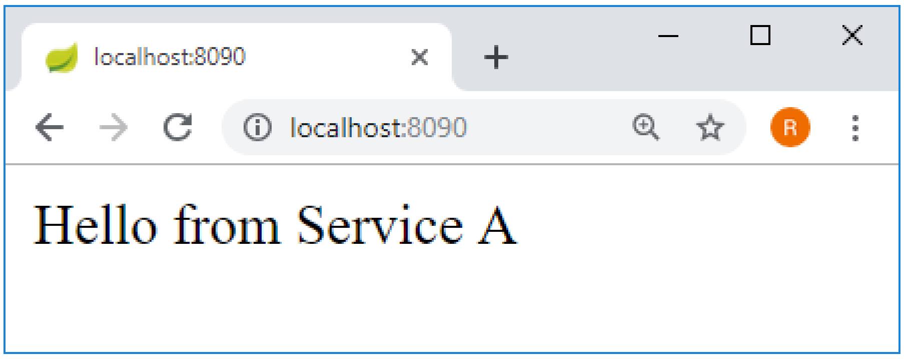
```
{"name":"ServiceA","profiles":
["default"],"label":null,"version":null,"state":null,"propertySources":
[{"name":"classpath:/config/ServiceA.yml","source":{"greeting":"Hello from Service A"}}]}
```

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

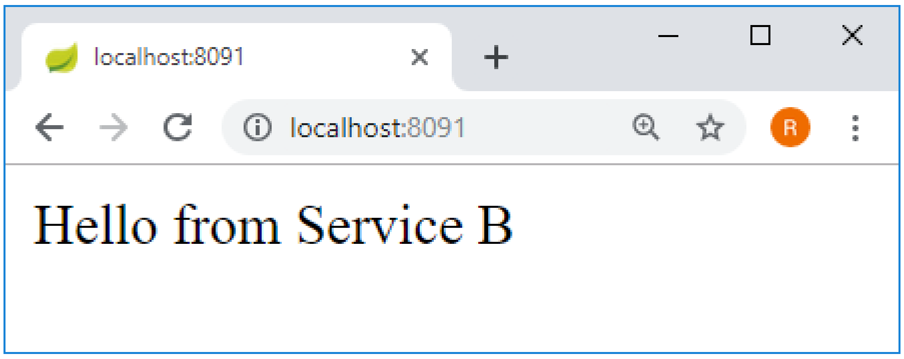


```
{"name":"ServiceB","profiles":
["default"],"label":null,"version":null,"state":null,"propertySources":
[{"name":"classpath:/config/ServiceB.yml","source":{"greeting":"Hello from Service B"}}]}
```

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



Hello from Service A

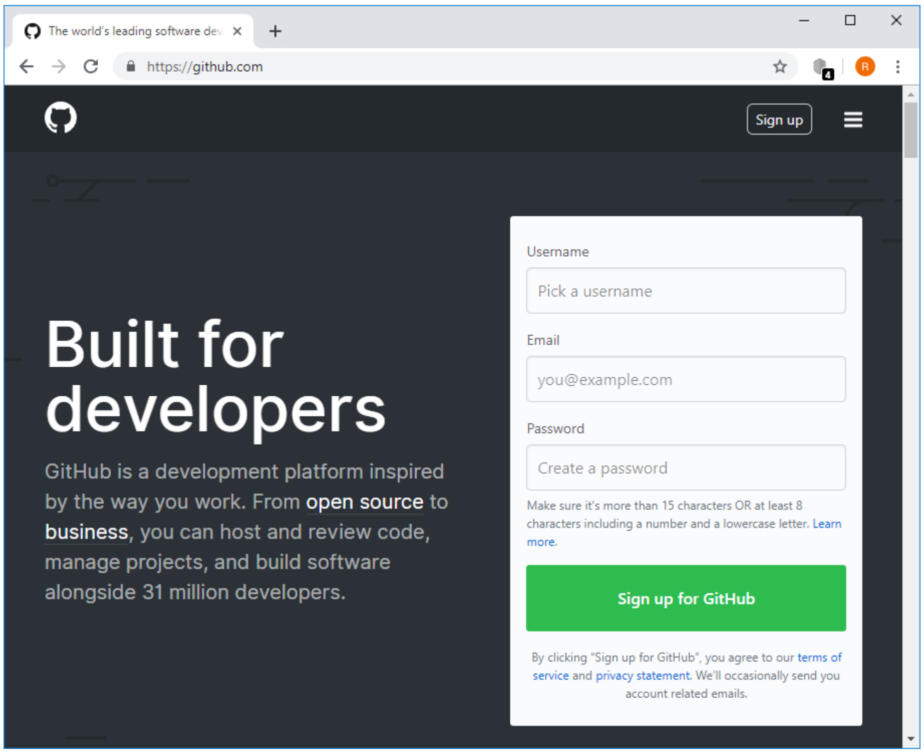


Hello from Service B

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

<u>Part 5</u>
First we need a GitHub account

Go to https://github.com/ and Sign up for a free GitHub account (if you don't have an account yet).



Built for developers

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.

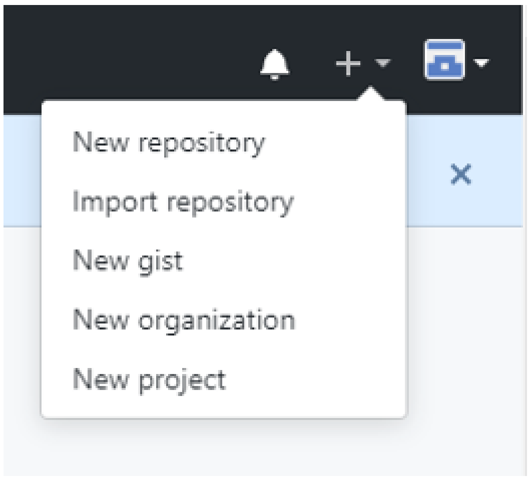
Make sure it's more than 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

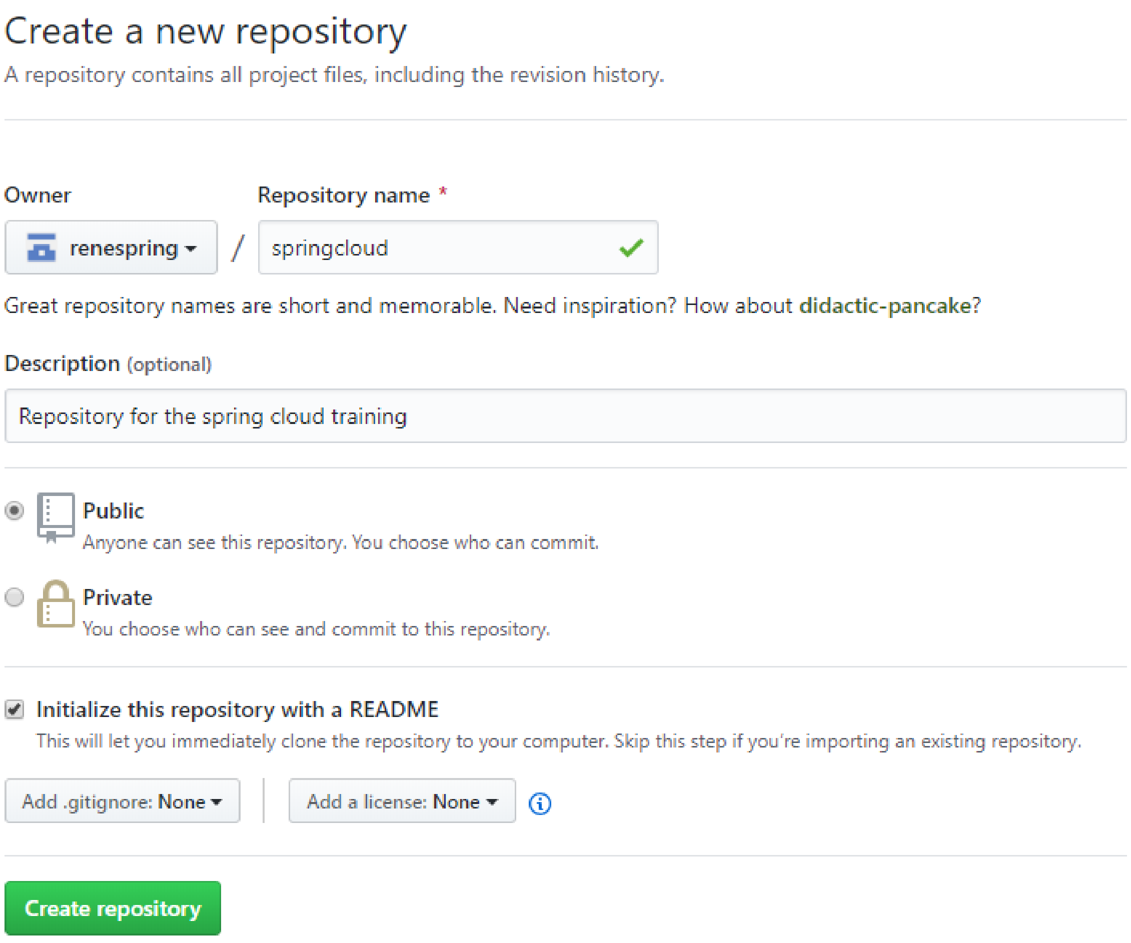
By clicking "Sign up for GitHub", you agree to our terms of service and privacy statement. We'll occasionally send you account related emails.

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

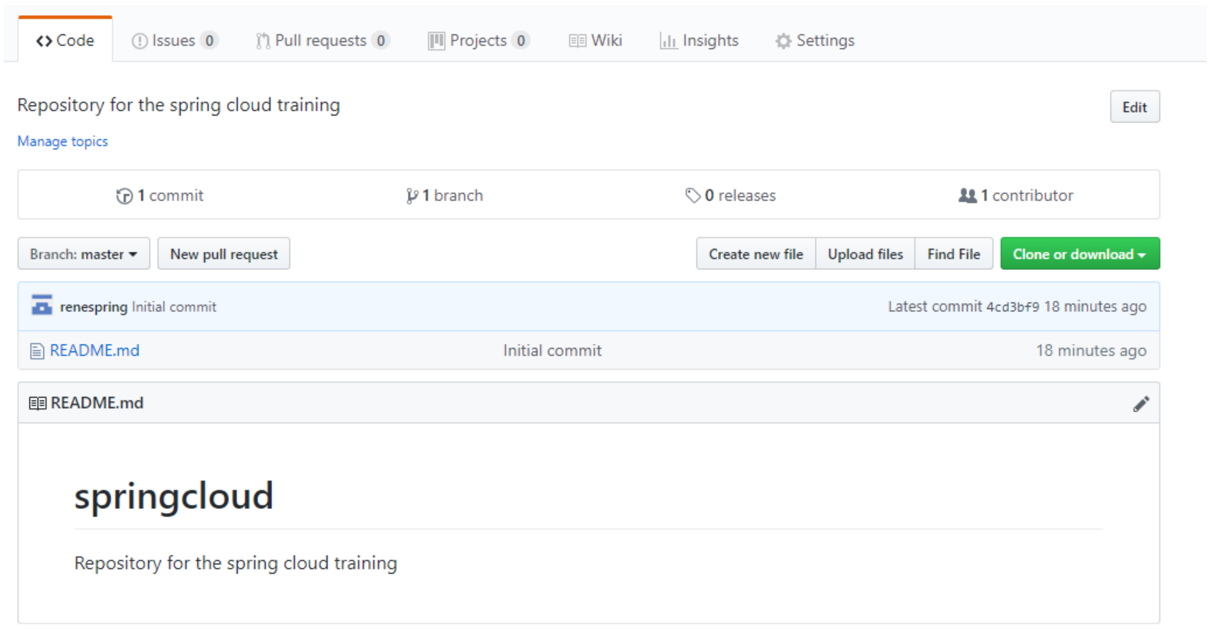
[✓ Public] Anyone can see this repository. You choose who can commit.

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

Name your repository **springcloud**.
Write a short description.
Select **Initialize this repository with a README**
Click **Create Repository**



Repository for the spring cloud training

1 commit | 1 branch | 0 releases | 1 contributor

renespring Initial commit — Latest commit 4cd3bf9 18 minutes ago

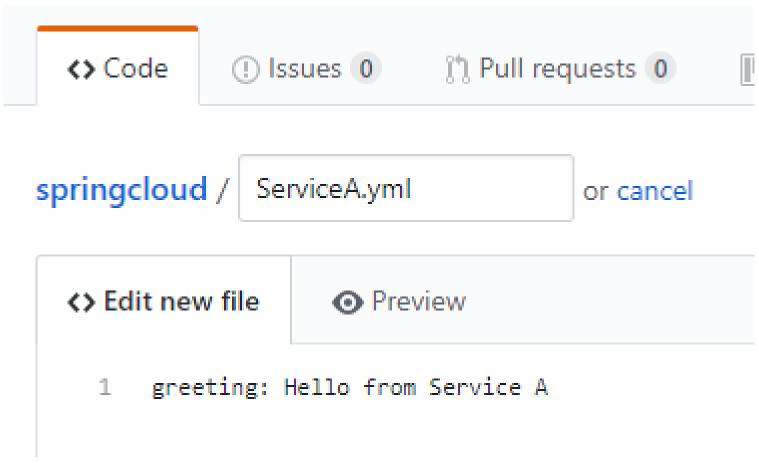
README.md — Initial commit — 18 minutes ago

README.md

springcloud

Repository for the spring cloud training

Click the **Create new file** button.

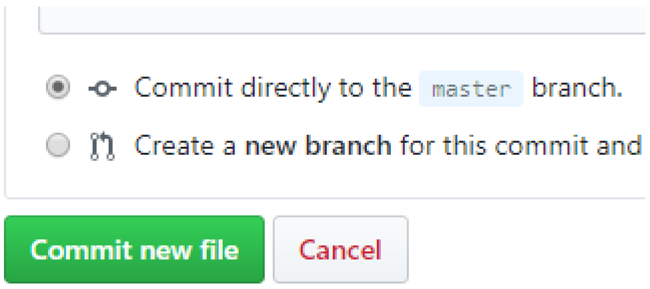


ServiceA.yml

```
greeting: Hello from Service A
```

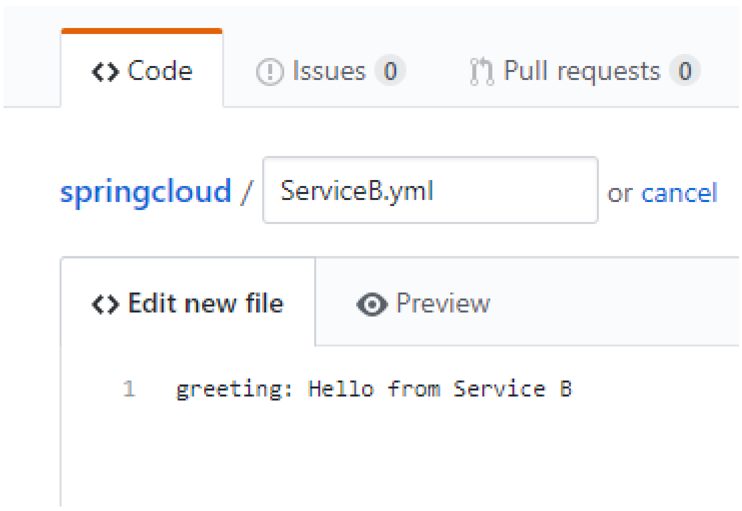
Name the file **ServiceA.yml** and enter the text

greeting: Hello from ServiceA



[● Commit directly to the master branch.]
○ Create a new branch for this commit and

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



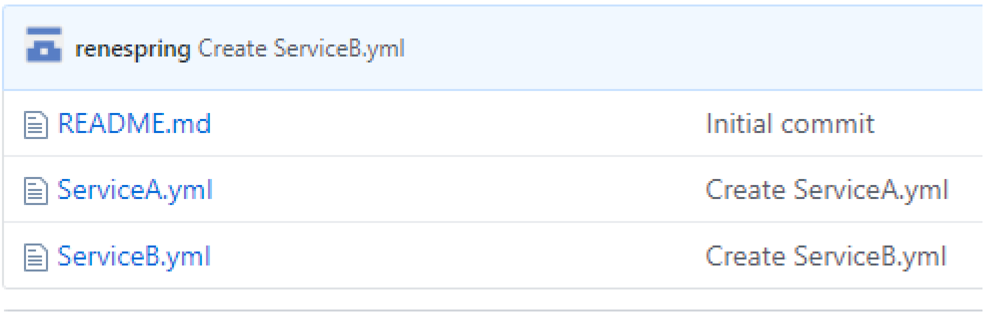
springcloud / ServiceB.yml or cancel

Edit new file | Preview

```
greeting: Hello from Service B
```

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

We have now 2 yml files:

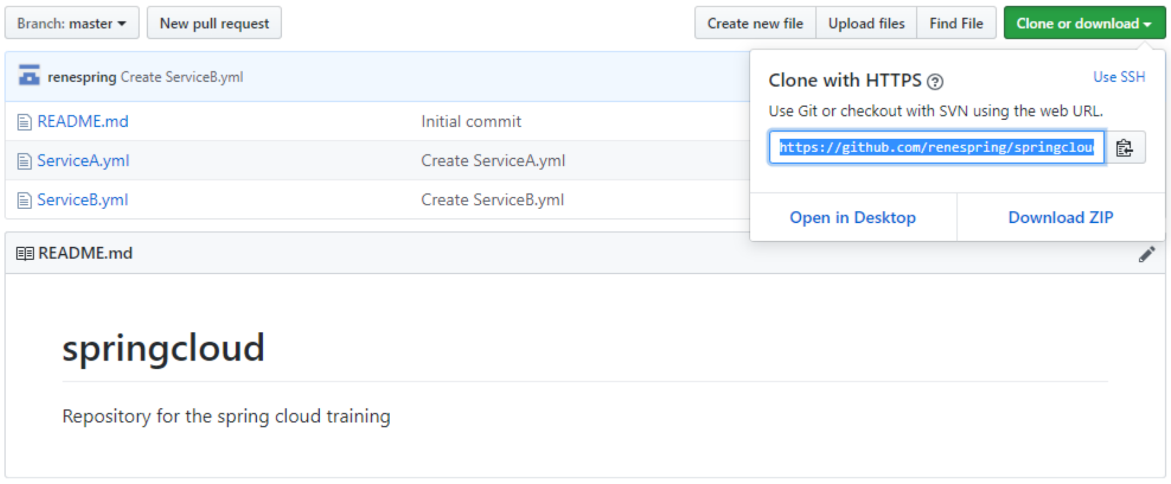


renespring Create ServiceB.yml

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 byn clicking the **Clone or download button**:



renespring Create ServiceB.yml

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

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

https://github.com/renespring/springclou

README.md

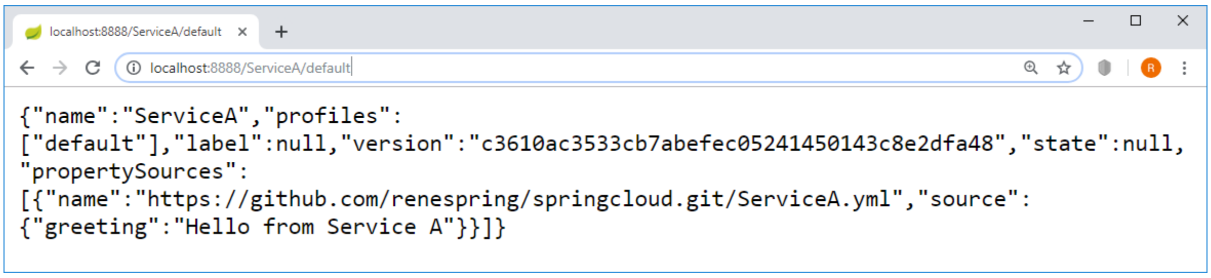
springcloud

Repository for the spring cloud training

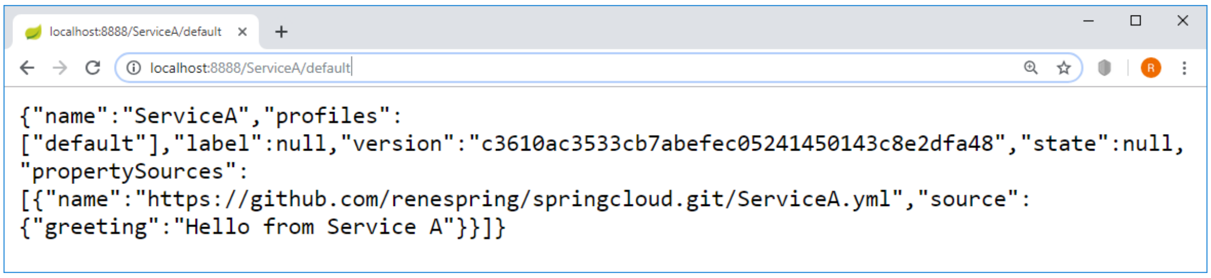
Change application.properties so that the property: spring.cloud.config.server.git.uri points to your git repository

```
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:

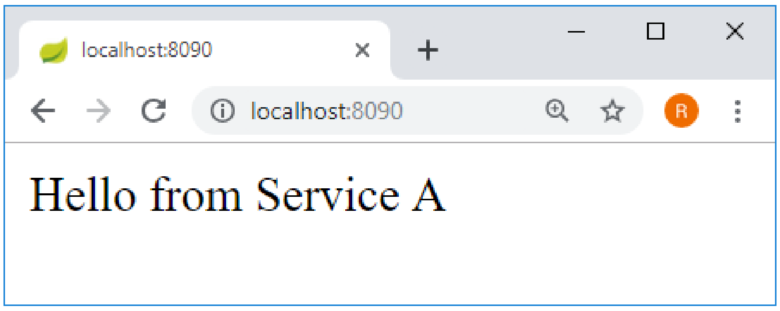


```
{"name":"ServiceA","profiles":
["default"],"label":null,"version":"c3610ac3533cb7abefec05241450143c8e2dfa48","state":null,
"propertySources":
[{"name":"https://github.com/renespring/springcloud.git/ServiceA.yml","source":
{"greeting":"Hello from Service A"}}]}
```

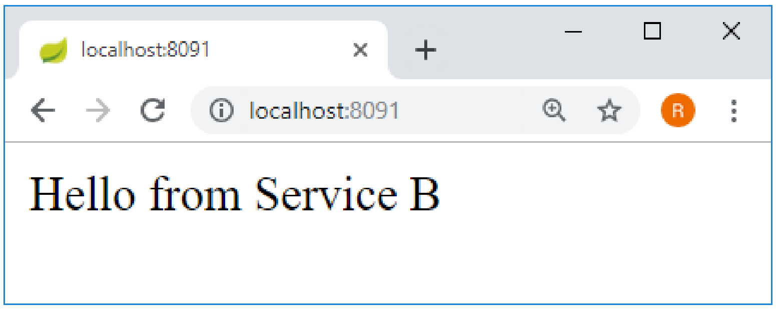


```
{"name":"ServiceA","profiles":
["default"],"label":null,"version":"c3610ac3533cb7abefec05241450143c8e2dfa48","state":null,
"propertySources":
[{"name":"https://github.com/renespring/springcloud.git/ServiceA.yml","source":
{"greeting":"Hello from Service A"}}]}
```

Also check if ServiceA and ServiceB are still working correctly:

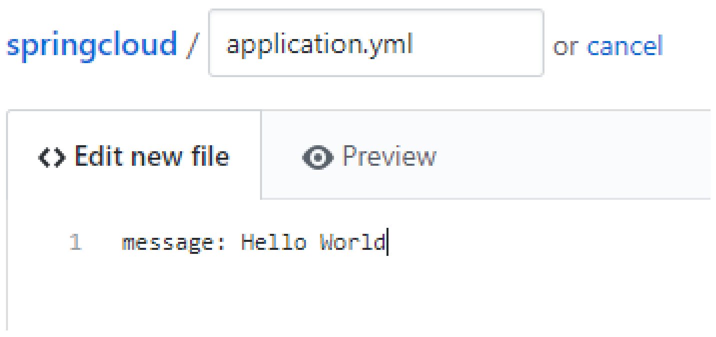


Hello from Service A



Hello from Service B

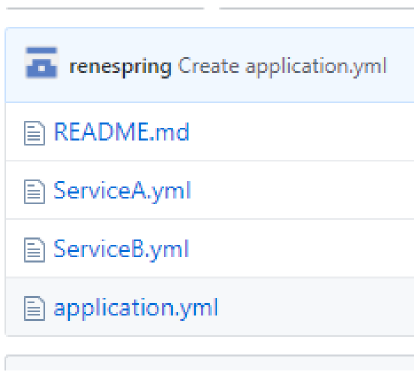
In GitHub, create a new file called **application.yml** and enter the **message** property:


springcloud / application.yml or cancel

Edit new file | Preview

```
message: Hello World
```

We now have 3 configuration files:


renespring Create application.yml

README.md

ServiceA.yml

ServiceB.yml

application.yml

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. A zip file containing all services for part 3
4. A zip file containing all services for part 5