

---

# Workshop Hands On – Spring Cloud Services

---

## Contents

1. Clone and build projects.....	2
2. Run Spring Cloud Services locally .....	4
3. Run microservices and clients locally.....	6
4. Test and observe locally .....	9
5. Create Spring Cloud Services on PAS .....	17
6. Push microservices and clients to PAS .....	22
7. Test and observe on Cloud .....	26

## 1. Clone and build projects

1. Create a directory and navigate to the directory by executing the following command:

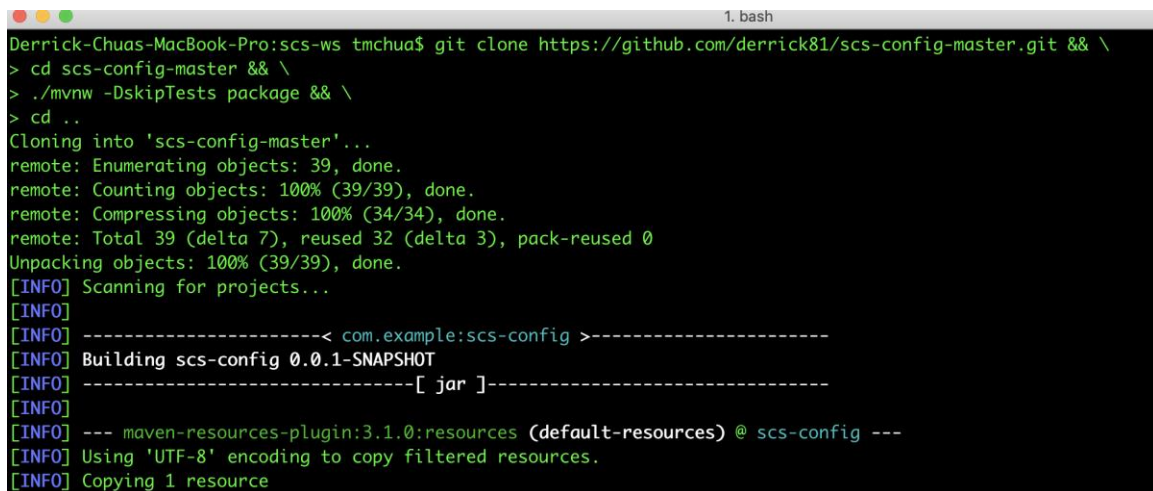
```
mkdir ~/scs-ws && cd ~/scs-ws
```



```
1. bash
Derrick-Chuas-MacBook-Pro:~ tmchua$ mkdir ~/scs-ws && cd ~/scs-ws
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$ pwd
/Users/tmchua/scs-ws
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$
```

2. Clone the Spring Cloud Config project by running the following command:

```
git clone https://github.com/derrick81/scs-config-master.git && \
cd scs-config-master && \
./mvnw -DskipTests package && \
cd ..
```



```
1. bash
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$ git clone https://github.com/derrick81/scs-config-master.git && \
> cd scs-config-master && \
> ./mvnw -DskipTests package && \
> cd ..
Cloning into 'scs-config-master'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 39 (delta 7), reused 32 (delta 3), pack-reused 0
Unpacking objects: 100% (39/39), done.
[INFO] Scanning for projects...
[INFO] -----< com.example:scs-config >-----
[INFO] Building scs-config 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ scs-config ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
```

3. Clone the Spring Cloud Eureka project by running the following command:

```
git clone https://github.com/derrick81/scs-eureka-master.git && \
cd scs-eureka-master && \
./mvnw -DskipTests package && \
cd ..
```

```

1. bash
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$ git clone https://github.com/derrick81/scs-eureka-master.git && \
> cd scs-eureka-master && \
> ./mvnw -DskipTests package && \
> cd ..
Cloning into 'scs-eureka-master'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 33 (delta 4), reused 27 (delta 1), pack-reused 0
Unpacking objects: 100% (33/33), done.
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:scs-eureka >-----
[INFO] Building scs-eureka 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ scs-eureka ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]

```

4. Clone the 4 microservices and client projects by running the following command:

```

git clone https://github.com/derrick81/scs-clients-master.git && \
cd scs-clients-master/employee-service && \
./mvnw -DskipTests package && \
cd ../employee-feign-client
./mvnw -DskipTests package && \
cd ../bookstore-service
./mvnw -DskipTests package && \
cd ../bookstore-ui-feign-hystrix
./mvnw -DskipTests package && \
cd ../..

```

```

1. bash
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$ git clone https://github.com/derrick81/scs-clients-master.git && \
> cd scs-clients-master/employee-service && \
> ./mvnw -DskipTests package && \
> cd ../employee-feign-client
Cloning into 'scs-clients-master'...
remote: Enumerating objects: 138, done.
remote: Counting objects: 100% (138/138), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 138 (delta 29), reused 133 (delta 26), pack-reused 0
Receiving objects: 100% (138/138), 69.73 KiB | 157.00 KiB/s, done.
Resolving deltas: 100% (29/29), done.
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:employee-service >-----
[INFO] Building employee-service 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ employee-service ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 1 resource
[INFO]

```

## 2. Run Spring Cloud Services locally

1. Open a **new terminal** and execute the following commands to run Spring Cloud Config:

```
cd ~/scs-ws/scs-config-master && \
./mvnw spring-boot:run
```



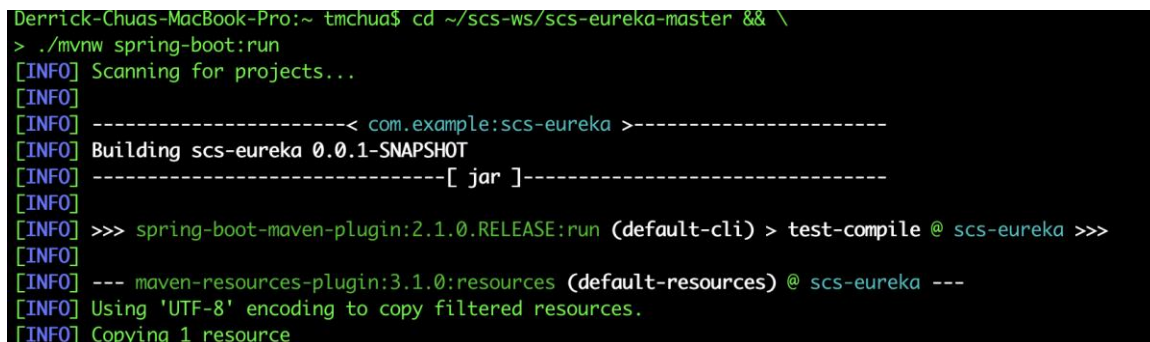
```
1. java
Derrick-Chuas-MacBook-Pro:scs-ws tmchua$ cd ~/scs-ws/scs-config-master && \
> ./mvnw spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:scs-config >-----
[INFO] Building scs-config 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.1.0.RELEASE:run (default-cli) > test-compile @ scs-config >>>
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ scs-config ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 1 resource
[INFO]
```

2. You should observe the following lines in the console output:

```
2019-04-19 15:50:17.377 INFO 7788 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebSe
rver : Tomcat started on port(s): 8888 (http) with context path ''
2019-04-19 15:50:17.379 INFO 7788 --- [          main] c.e.scsconfig.ScsConfigApplication
: Started ScsConfigApplication in 13.023 seconds (JVM running for 15.224)
```

3. Open a **new terminal** and execute the following commands to run Spring Cloud Eureka:

```
cd ~/scs-ws/scs-eureka-master && \
./mvnw spring-boot:run
```



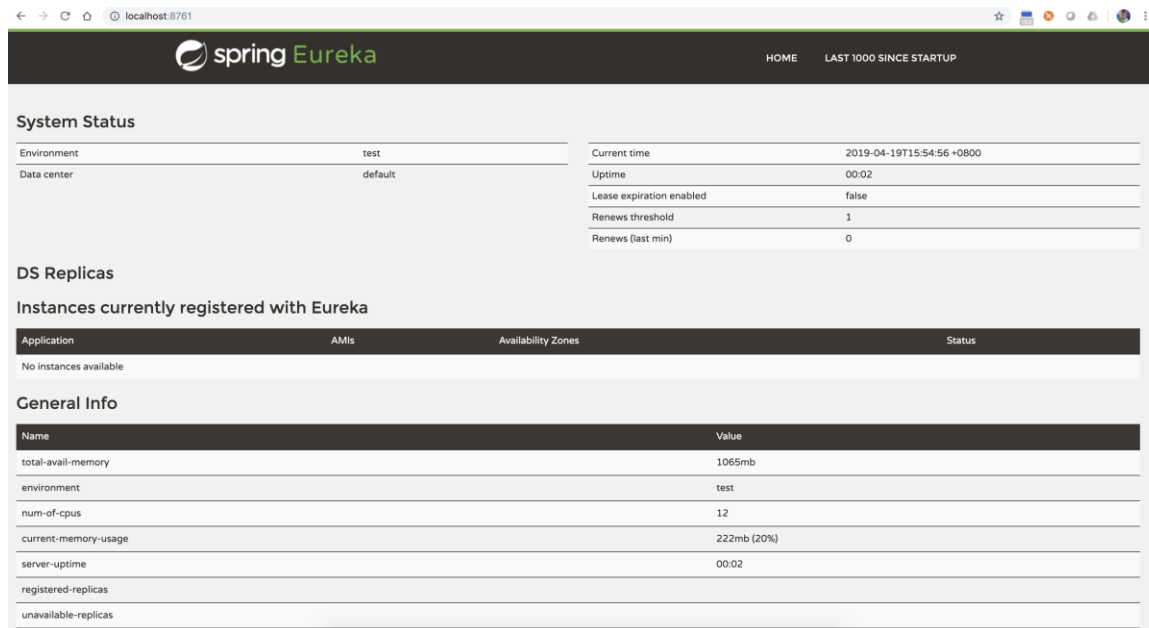
```
Derrick-Chuas-MacBook-Pro:~ tmchua$ cd ~/scs-ws/scs-eureka-master && \
> ./mvnw spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:scs-eureka >-----
[INFO] Building scs-eureka 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.1.0.RELEASE:run (default-cli) > test-compile @ scs-eureka >>>
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ scs-eureka ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
```

4. You should observe the following lines in the console output:

```
2019-04-19 15:53:04.397 INFO 7820 --- [ Thread-14] e.s.EurekaServerInitializerConfiguration : Sta
rted Eureka Server
2019-04-19 15:53:04.414 INFO 7820 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
cat started on port(s): 8761 (http) with context path ''
2019-04-19 15:53:04.415 INFO 7820 --- [          main] .s.c.n.e.s.EurekaAutoServiceRegistration : Upd
ating port to 8761
2019-04-19 15:53:04.418 INFO 7820 --- [          main] c.e.scseureka.ScsEurekaApplication : Sta
rted ScsEurekaApplication in 13.865 seconds (JVM running for 16.179)
```

- Using a browser, navigate to the Spring Cloud Eureka web UI via the following URL:

<http://localhost:8761>



The screenshot shows the Spring Cloud Eureka web UI in a browser window. The address bar shows 'localhost:8761'. The page has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'.

### System Status

Environment	test	Current time	2019-04-19T15:54:56 +0800
Data center	default	Uptime	00:02
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

### DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

### General Info

Name	Value
total-avail-memory	1065mb
environment	test
num-of-cpus	12
current-memory-usage	222mb (20%)
server-uptime	00:02
registered-replicas	
unavailable-replicas	



## 3. Run microservices and clients locally

1. Open a **new terminal** and execute the following commands to run Employee microservice:

```
cd ~/scs-ws/scs-clients-master/employee-service && \
./mvnw spring-boot:run
```

```
2019-04-19 15:56:10.286 INFO 7972 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_EMPLOYEE-SERVICE/192.168.2.90:employee-service:8081: registering service...
2019-04-19 15:56:10.335 INFO 7972 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_EMPLOYEE-SERVICE/192.168.2.90:employee-service:8081 - registration status: 204
2019-04-19 15:56:10.348 INFO 7972 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
cat started on port(s): 8081 (http) with context path ''
2019-04-19 15:56:10.350 INFO 7972 --- [main] .s.c.n.e.s.EurekaAutoServiceRegistration : Upd
ating port to 8081
2019-04-19 15:56:10.356 INFO 7972 --- [main] c.e.e.EmployeeServiceApplication : Sta
rted EmployeeServiceApplication in 30.952 seconds (JVM running for 33.58)
```

2. Open a **new terminal** and execute the following commands to run Employee feign client:

```
cd ~/scs-ws/scs-clients-master/employee-feign-client && \
./mvnw spring-boot:run
```

```
2019-04-19 15:56:05.850 INFO 7981 --- [main] com.netflix.discovery.DiscoveryClient : Saw
local status change event StatusChangeEvent [timestamp=1555660565850, current=UP, previous=STARTING]
2019-04-19 15:56:05.852 INFO 7981 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_EMPLOYEE-FEIGN-CLIENT/192.168.2.90:employee-feign-client:8082: registering service...
2019-04-19 15:56:05.934 INFO 7981 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
cat started on port(s): 8082 (http) with context path ''
2019-04-19 15:56:05.939 INFO 7981 --- [main] c.e.e.EmployeeFeignClientApplication : Sta
rted EmployeeFeignClientApplication in 19.282 seconds (JVM running for 22.039)
2019-04-19 15:56:06.052 INFO 7981 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_EMPLOYEE-FEIGN-CLIENT/192.168.2.90:employee-feign-client:8082 - registration status: 204
```

3. Open a **new terminal** and execute the following commands to run Bookstore microservice:

```
cd ~/scs-ws/scs-clients-master/bookstore-service && \
./mvnw spring-boot:run
```

```
2019-04-19 15:56:12.818 INFO 7994 --- [main] com.netflix.discovery.DiscoveryClient : Saw
local status change event StatusChangeEvent [timestamp=1555660572818, current=UP, previous=STARTING]
2019-04-19 15:56:12.819 INFO 7994 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_BOOKSTORE-SERVICE/192.168.2.90:bookstore-service:8083: registering service...
2019-04-19 15:56:12.861 INFO 7994 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_BOOKSTORE-SERVICE/192.168.2.90:bookstore-service:8083 - registration status: 204
2019-04-19 15:56:12.868 INFO 7994 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
cat started on port(s): 8083 (http) with context path ''
2019-04-19 15:56:12.872 INFO 7994 --- [main] c.e.b.BookstoreServiceApplication : Sta
rted BookstoreServiceApplication in 19.052 seconds (JVM running for 21.835)
```

4. Open a **new terminal** and execute the following commands to run Bookstore feign hystrix client:

```
cd ~/scs-ws/scs-clients-master/bookstore-ui-feign-hystrix && \
./mvnw spring-boot:run
```

```
2019-04-19 15:56:22.256 INFO 8003 --- [main] com.netflix.discovery.DiscoveryClient : Saw
local status change event StatusChangeEvent [timestamp=1555660582256, current=UP, previous=STARTING]
2019-04-19 15:56:22.258 INFO 8003 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_BOOKSTORE-UI/192.168.2.90:bookstore-ui:8084: registering service...
2019-04-19 15:56:22.295 INFO 8003 --- [nfoReplicator-0] com.netflix.discovery.DiscoveryClient : Dis
coveryClient_BOOKSTORE-UI/192.168.2.90:bookstore-ui:8084 - registration status: 204
2019-04-19 15:56:22.303 INFO 8003 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tom
cat started on port(s): 8084 (http) with context path ''
2019-04-19 15:56:22.307 INFO 8003 --- [main] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/employee-service.properties
2019-04-19 15:56:22.307 INFO 8003 --- [main] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/employee-feign-client.properties
2019-04-19 15:56:22.307 INFO 8003 --- [main] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/bookstore-ui.properties
```

- Observe the Spring Cloud Config terminal console output and you should see the following:

```
2019-04-19 15:55:58.940 INFO 7788 --- [nio-8888-exec-1] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/employee-service.properties
2019-04-19 15:55:59.800 INFO 7788 --- [nio-8888-exec-2] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/employee-feign-client.properties
2019-04-19 15:56:18.455 INFO 7788 --- [nio-8888-exec-7] o.s.c.c.s.e.NativeEnvironmentRepository : Adding property source: file:/var/folders/nw/dwd3y25930nggcm6yh3q947h0000gp/T/config
-repo-3141964626437442138/bookstore-ui.properties
```

- Refresh the Spring Cloud Eureka web UI and you should see the following:

The screenshot shows the Spring Cloud Eureka web UI. The top navigation bar includes 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment (test), data center (default), current time (2019-04-19T16:01:28 +0800), uptime (00:08), lease expiration enabled (false), renewals threshold (8), and renewals (last min) (8).
- EMERGENCY!** A red warning message: "EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE."
- DS Replicas:** A section for distributed system replicas.
- Instances currently registered with Eureka:** A table listing applications, AMIs, availability zones, and status.
- General Info:** A table showing system metrics like total-avail-memory (1065mb) and environment (test).

Application	AMIs	Availability Zones	Status
BOOKSTORE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.2.90:bookstore-service:8083
BOOKSTORE-UI	n/a (1)	(1)	UP (1) - 192.168.2.90:bookstore-ui:8084
EMPLOYEE-FEIGN-CLIENT	n/a (1)	(1)	UP (1) - 192.168.2.90:employee-feign-client:8082
EMPLOYEE-SERVICE	n/a (1)	(1)	UP (1) - 192.168.2.90:employee-service:8081

Name	Value
total-avail-memory	1065mb
environment	test

- You should also observe the registration from Spring Cloud Eureka's terminal console output:

```
2019-04-19 15:56:06.049 INFO 7820 --- [nio-8761-exec-1] c.n.e.registry.AbstractInstanceRegistry : Registered instance EMPLOYEE-FEIGN-CLIENT/192.168.2.90:employee-feign-client:8082 with status UP (replication=false)
2019-04-19 15:56:10.334 INFO 7820 --- [nio-8761-exec-2] c.n.e.registry.AbstractInstanceRegistry : Registered instance EMPLOYEE-SERVICE/192.168.2.90:employee-service:8081 with status UP (replication=false)
2019-04-19 15:56:12.860 INFO 7820 --- [nio-8761-exec-6] c.n.e.registry.AbstractInstanceRegistry : Registered instance BOOKSTORE-SERVICE/192.168.2.90:bookstore-service:8083 with status UP (replication=false)
2019-04-19 15:56:22.294 INFO 7820 --- [nio-8761-exec-9] c.n.e.registry.AbstractInstanceRegistry : Registered instance BOOKSTORE-UI/192.168.2.90:bookstore-ui:8084 with status UP (replication=false)
```

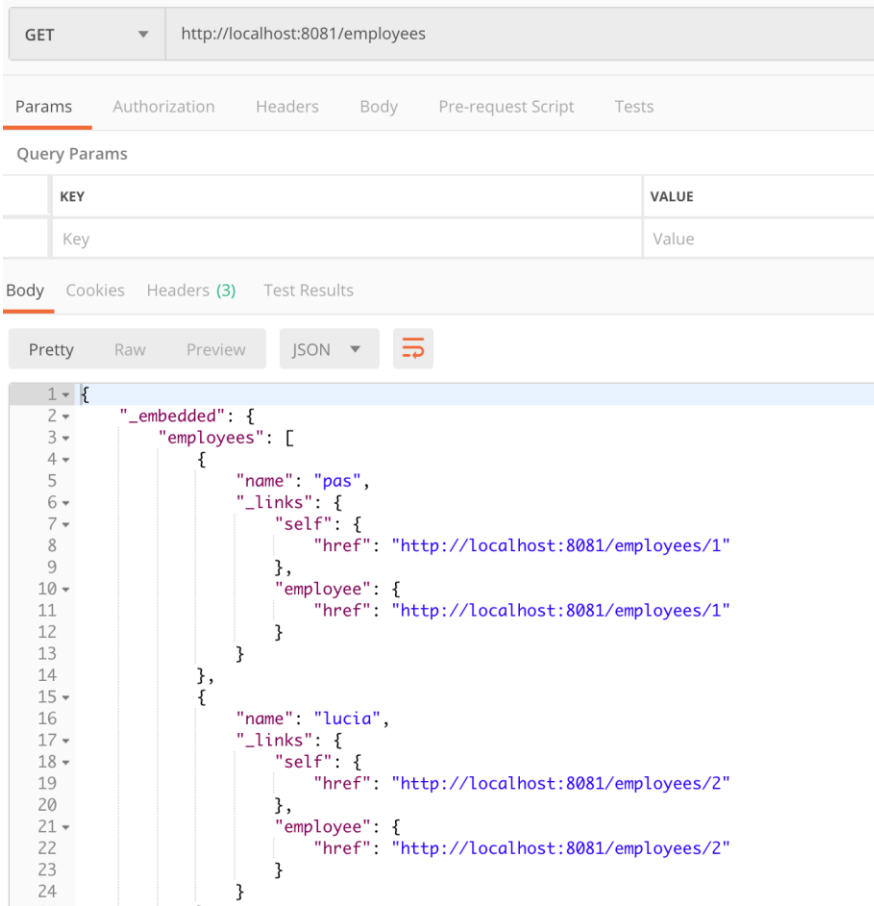


## 4. Test and observe locally

1. Invoke the employee microservice “employees” endpoint directly by executing the following in Postman.

GET <http://localhost:8081/employees>

You should observe the following if it is running successfully.



The screenshot shows the Postman interface with a GET request to <http://localhost:8081/employees>. The response is displayed in the Body tab, formatted as JSON. The response is an array of two objects, each representing an employee with a self-link and an employee link.

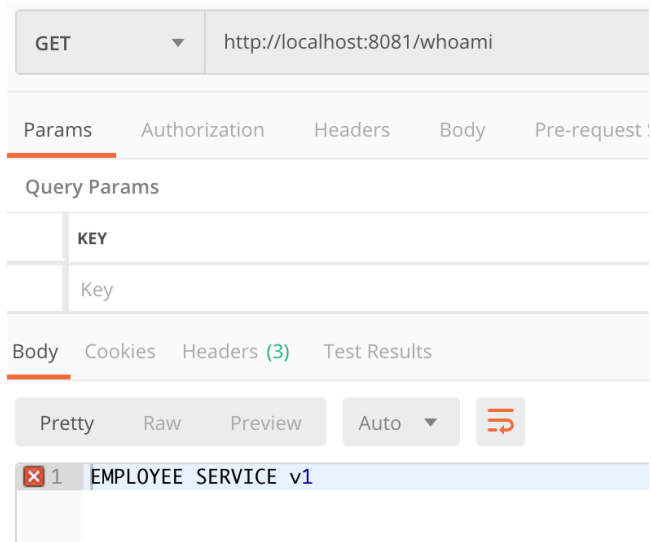
```

1 {
2   "_embedded": {
3     "employees": [
4       {
5         "name": "pas",
6         "_links": {
7           "self": {
8             "href": "http://localhost:8081/employees/1"
9           },
10          "employee": {
11            "href": "http://localhost:8081/employees/1"
12          }
13        }
14      },
15      {
16        "name": "lucia",
17        "_links": {
18          "self": {
19            "href": "http://localhost:8081/employees/2"
20          },
21          "employee": {
22            "href": "http://localhost:8081/employees/2"
23          }
24        }
25      }
26    ]
27  }
28 }
  
```

2. Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET <http://localhost:8081/whoami>

You should observe the following if it is running successfully.

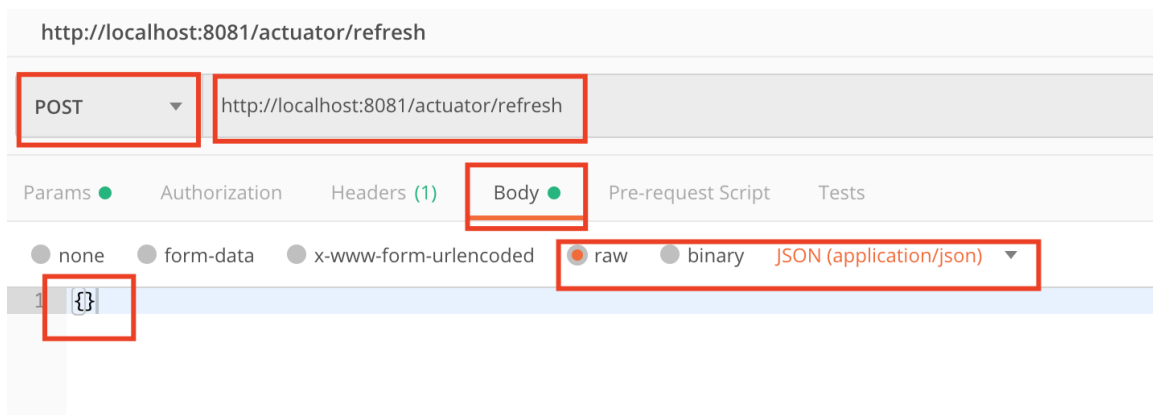


- Wait for the instructor to update employee microservice config in the github config store to reflect "v2".

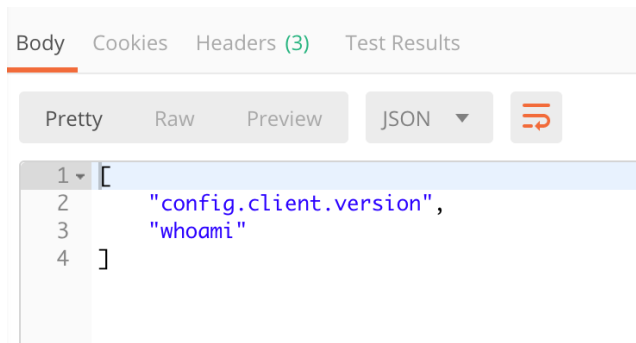


- Refresh the employee microservice's copy of the configuration by executing the following command in Postman.

POST http://localhost:8081/actuator/refresh  
 Content-Type: application/json  
 Body: {}



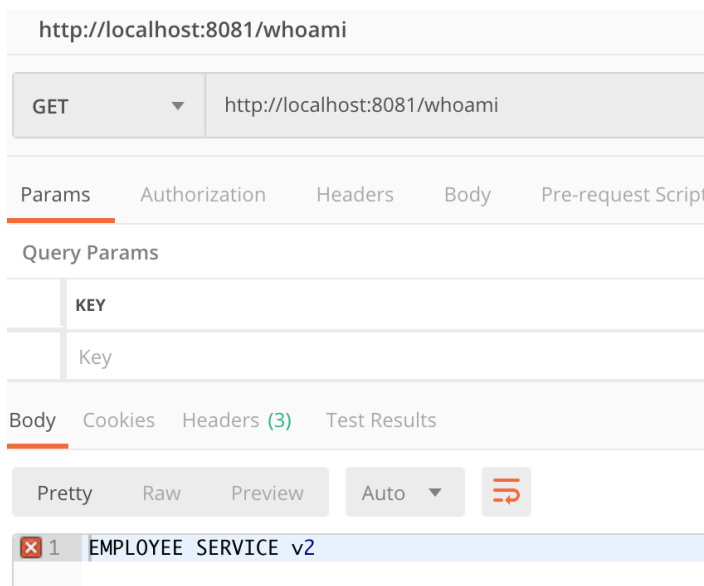
You should observe the following if executed successfully.



- Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET <http://localhost:8081/whoami>

You should observe the following (a change in the version number) if it is running successfully.



- Wait for the instructor to revert employee microservice config in the github config store to reflect “v1”.

Branch: master ▼ [scs-config-store](#) / [employee-service.properties](#)

 derrick81 Update employee-service.properties

1 contributor

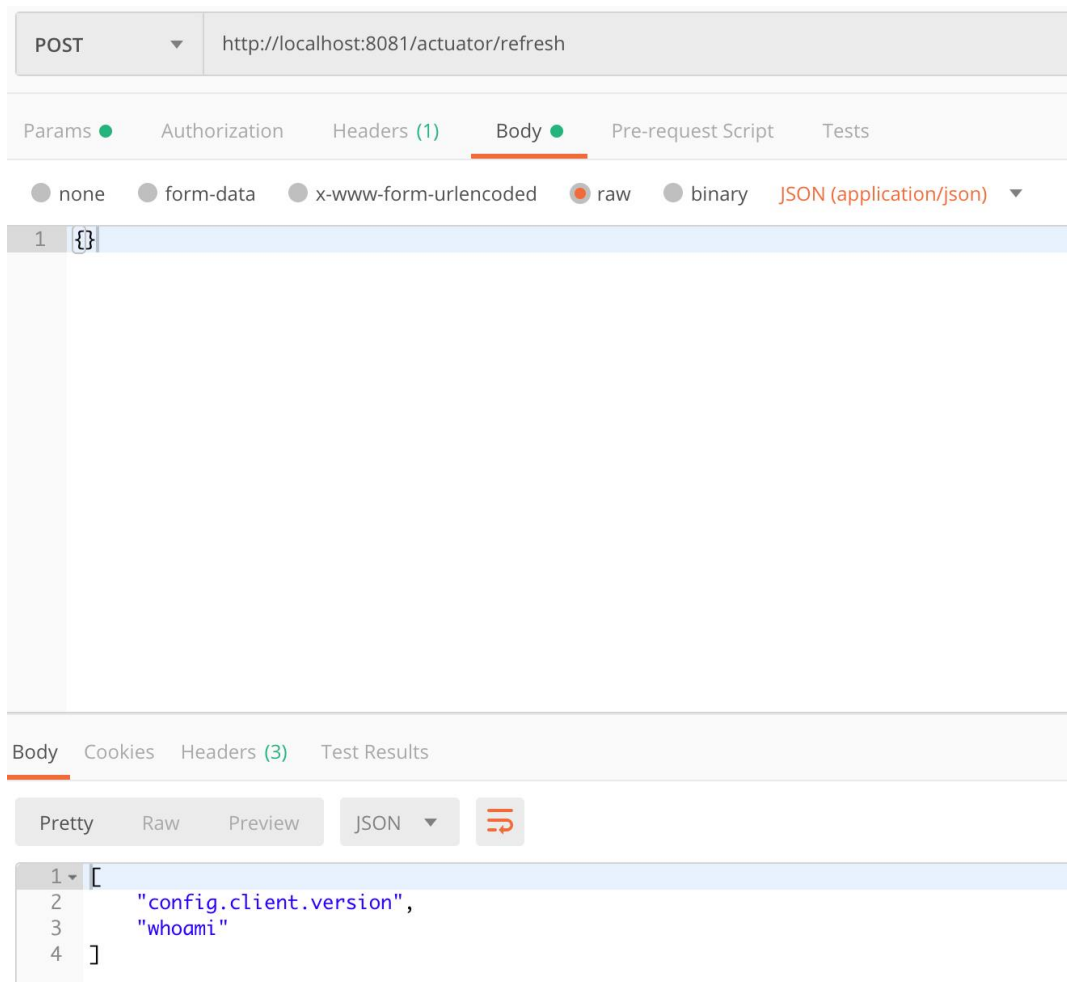
Executable File | 3 lines (1 sloc) | 28 Bytes

```
1 whoami=EMPLOYEE SERVICE v1
2
```

7. Refresh the employee microservice's copy of the configuration by executing the following command in Postman.

POST <http://localhost:8081/actuator/refresh>  
Content-Type: application/json

You should observe the following if executed successfully.



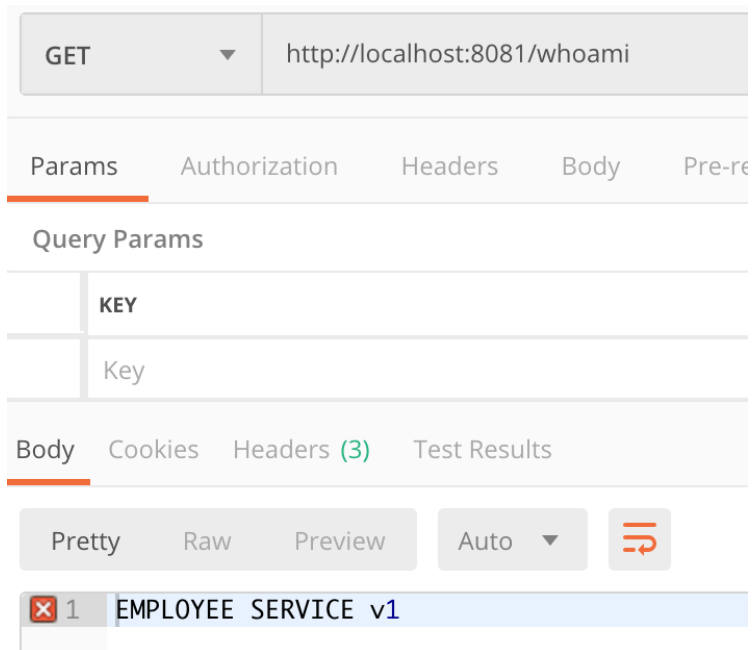
The screenshot shows the Postman interface. The top bar indicates a POST request to `http://localhost:8081/actuator/refresh`. The 'Body' tab is selected, showing a JSON body of `{}`. The 'Headers' tab shows `Content-Type: application/json`. The 'Test Results' tab is also visible. The response body is shown in the bottom section, displaying a JSON array:

```
[
  "config.client.version",
  "whoami"
]
```

8. Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET <http://localhost:8081/whoami>

You should observe the following (a change in the version number) if it is running successfully.

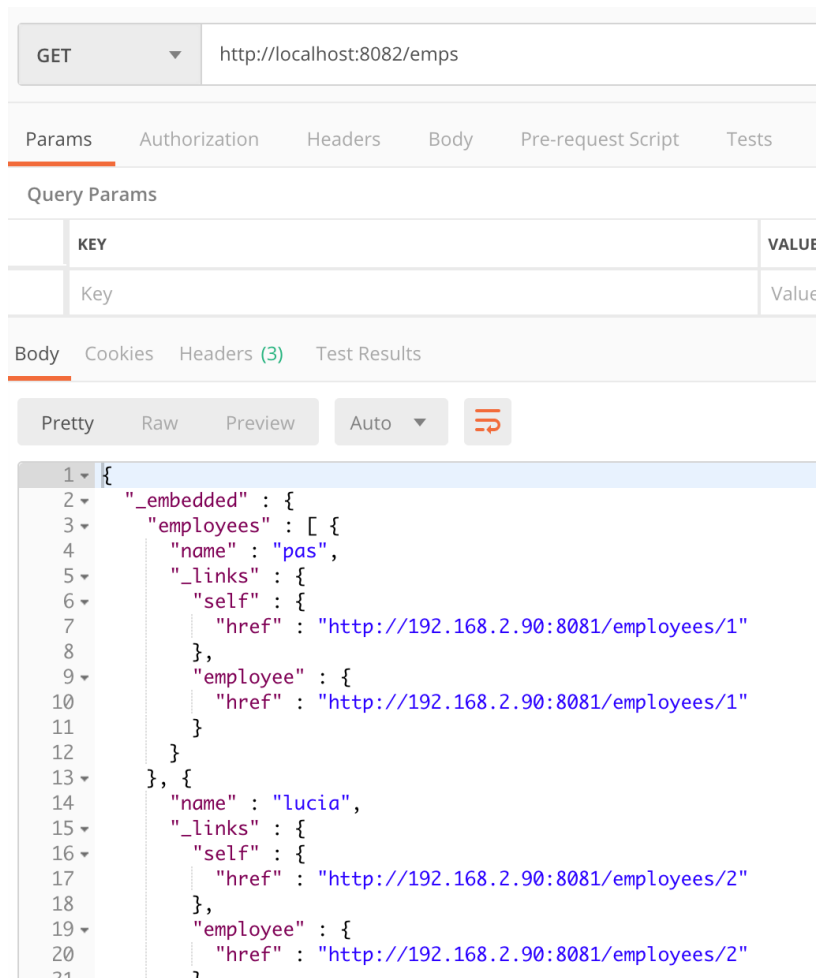


9. Now consume the employee microservice through a feign client. Invoke the client’s “employees” endpoint directly by executing the following in Postman.

GET <http://localhost:8082/emp>

You should observe the following if it is running successfully.





GET http://localhost:8082/emps

Params Authorization Headers Body Pre-request Script Tests

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (3) Test Results

Pretty Raw Preview Auto

```

1 {
2   "_embedded" : {
3     "employees" : [ {
4       "name" : "pas",
5       "_links" : {
6         "self" : {
7           "href" : "http://192.168.2.90:8081/employees/1"
8         },
9       "employee" : {
10        "href" : "http://192.168.2.90:8081/employees/1"
11      }
12    }, {
13      "name" : "lucia",
14      "_links" : {
15        "self" : {
16          "href" : "http://192.168.2.90:8081/employees/2"
17        },
18      "employee" : {
19        "href" : "http://192.168.2.90:8081/employees/2"
20      }
21    }
22  ]
23 }
  
```

- Now turn off the employee microservice to observe the clients response when the backing microservice is unavailable.  
Turn off the employee microservice by issuing the keyboard “Control+C” keystrokes on the terminal running the employee microservice.

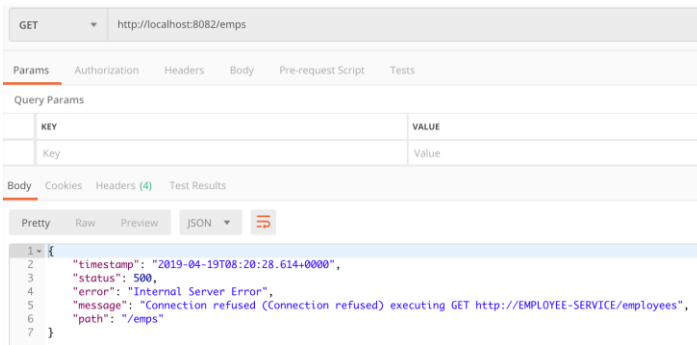
```

2019-04-19 16:19:46.016 INFO 7972 --- [ Thread-9] com.zaxxer.hikari.HikariDataSource : Hik
ariPool-2 - Shutdown initiated...
2019-04-19 16:19:46.018 INFO 7972 --- [ Thread-9] com.zaxxer.hikari.HikariDataSource : Hik
ariPool-2 - Shutdown completed.
  
```

- Invoke the client’s “employees” endpoint directly by executing the following in Postman.

GET http://localhost:8082/emps

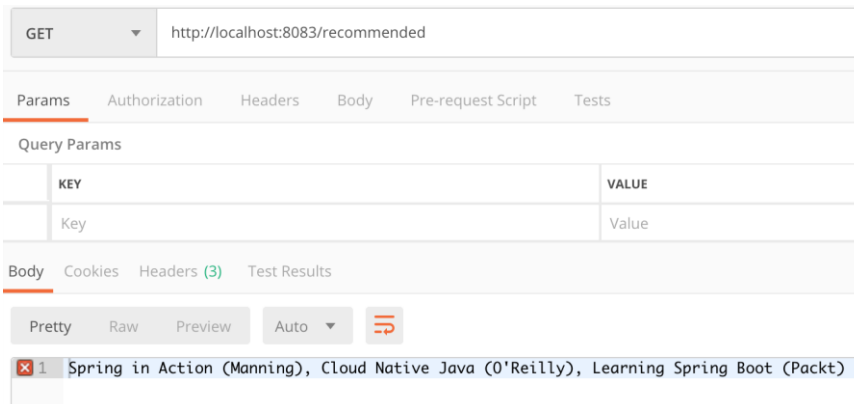
You should observe the following failure.



- Invoke the bookstore microservice “recommended” endpoint directly by executing the following in Postman.

GET <http://localhost:8083/recommended>

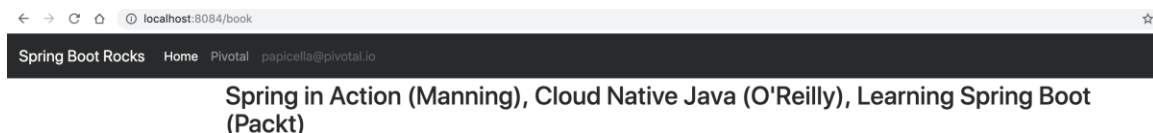
You should observe the following if it is running successfully.



- Invoke the bookstore web UI which will consume the bookstore microservice. Visit the following URL in the browser:

<http://localhost:8084/book>

You should see the following page.



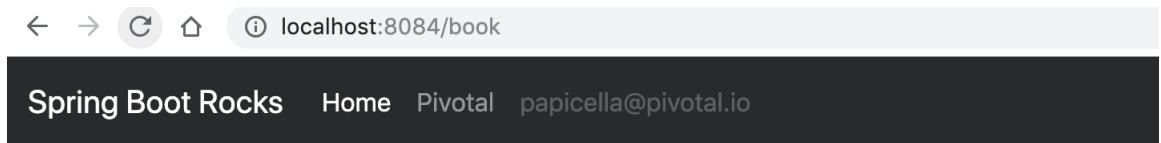
- Now turn off the bookstore microservice to observe the clients response when the backing microservice is unavailable.  
Turn off the bookstore microservice by issuing the keyboard “Control+C” keystrokes on the terminal running the bookstore microservice.

```
2019-04-19 16:22:12.508 WARN 7994 --- [Thread-9] .s.c.a.CommonAnnotationBeanPostProcessor : Destroy method on bean with name 'scopedTarget.eurekaClient' threw an exception: org.springframework.beans.factory.BeanCreationNotAllowedException: Error creating bean with name 'eurekaInstanceConfigBean': Singleton bean creation not allowed while singletons of this factory are in destruction (Do not request a bean from a BeanFactory in a destroy method implementation!)
[INFO] jerrick-Chuas-MacBook-Pro:bookstore-service tmchua$
```

15. Invoke the bookstore web UI which will consume the bookstore microservice. Visit the following URL in the browser:

<http://localhost:8084/book>

You should see the following page.



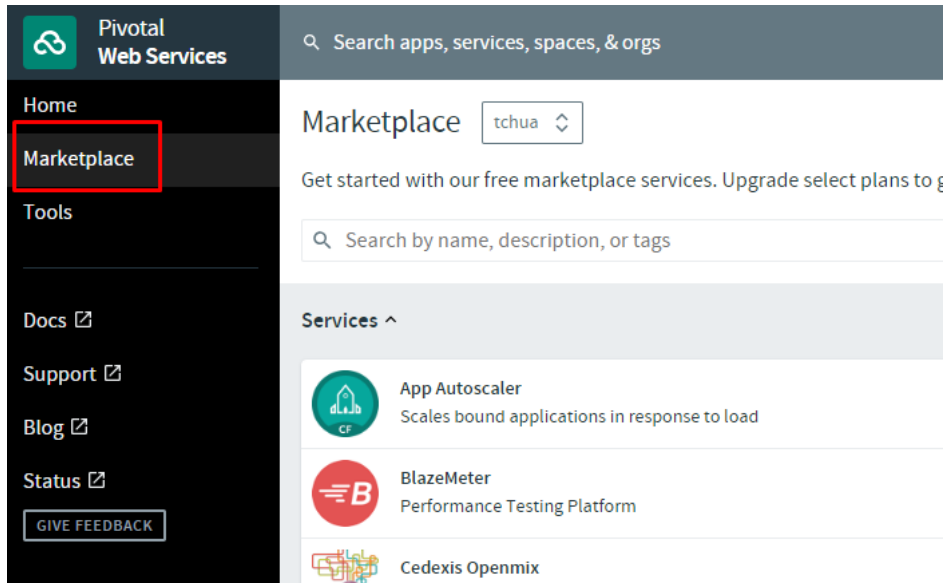
## Cloud Native Java (O'Reilly)

## 5. Create Spring Cloud Services on PAS

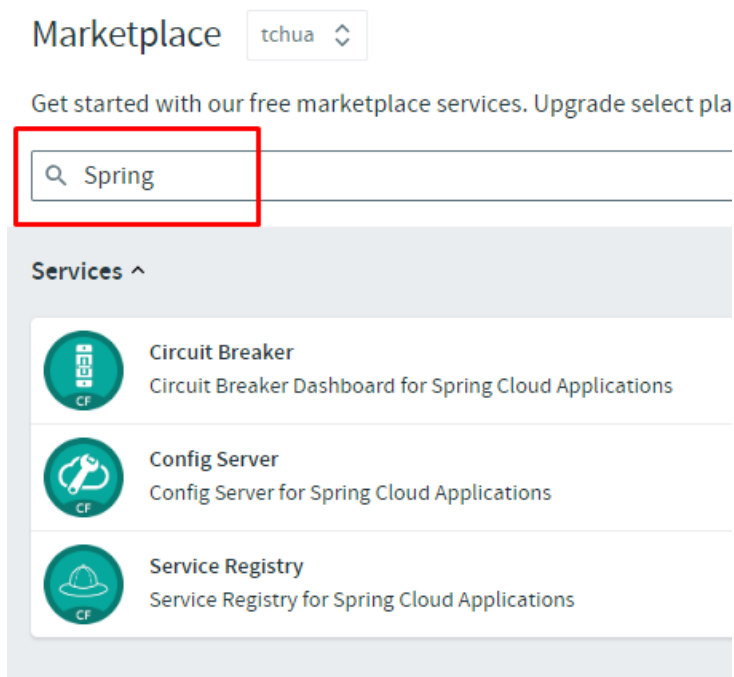
1. Navigate your browser to the following URL:

<https://console.run.pivotal.io>

Click on the “Marketplace” link.



2. Search using the keyword “Spring” and you should find the list of services narrowing to just the three Spring Cloud Services.



- Click on “Config Server”. In the following screen, select the “trial” plan and then click on the “Select This Plan” button.

**SERVICE**  
**Config Server**  
Config Server for Spring Cloud Applications  
[Docs](#) [Support](#)

**ABOUT THIS SERVICE**  
Provides server and client-side support for externalized configuration in a distributed system deployed to Pivotal Cloud Foundry.

**COMPANY**  
Pivotal

**trial free**

**standard**  
\$39.99/MONTH

**trial free**

- Single-tenant
- Backed by user-provided Git or Vault
- deleted automatically after 7 days

**SELECT THIS PLAN**

- The instance configuration screen appears. Use the following values to create the service.

**Instance name:** config-server

**Enter JSON:** Enable

**JSON value:**

```
{
  "git": {
    "uri": "https://github.com/derrick81/scs-config-store.git"
  }
}
```



**trial**  
free
 

- Single-tenant
- Backed by user-provided Git or Vault
- deleted automatically after 7 days

### Instance Configuration

**Instance Name**

**Add To Space**

**Bind To App (Optional)**

**Add Parameters (Optional)** Enter JSON ☒

**JSON**

```

1 {
2   "git": {
3     "uri": "https://github.com/derrick81/scs-config-store.git"
4   }
5 }

```

**Advanced Configuration**  
[HIDE ADVANCED OPTIONS](#)

CANCEL CREATE


- You should get redirected to the following screen.

[Home](#) / [tchua](#) / development


SPACE      RUNNING   STOPPED   CRASHED  
 development   ● 0   ● 0   ● 0

Apps   **Service (1)**   Routes   Member (1)   Settings

**Services**
ADD A SERVICE

Service	Name	Bound Apps	Plan	Last Operation
 Config Server Creating service instance.	config-server	0	free - trial	create in progress

- Go back to the Marketplace to create a Service Registry service. Similar to the Config Server, select the “trial” plan.



**SERVICE**  
**Service Registry**  
Service Registry for Spring Cloud Applications

[Docs](#) [Support](#)

**ABOUT THIS SERVICE**  
Provides application service registration and discovery in a distributed system deployed to Pivotal Cloud Foundry.

**COMPANY**  
Pivotal

**trial**  
free

**standard**  
\$39.99/MONTH

**trial free**

- Single-tenant
- Netflix OSS Eureka
- deleted automatically after 7 days

**SELECT THIS PLAN**

7. The instance configuration screen appears. Use the following values to create the service.

**Instance name:** service-registry

**trial**  
free

- Single-tenant
- Netflix OSS Eureka
- deleted automatically after 7 days

### Instance Configuration

**Instance Name**

service-registry

**Add To Space**

development

**Bind To App (Optional)**

[do not bind]

**Advanced Configuration** ⓘ  
Configuration parameters for creating a Spring Cloud Service Registry

**Count (Optional)**

*Minimum of 1*

**Peers (Optional)**



**CANCEL** **CREATE**

8. You will get redirected back to the “Services” tab of your space.

[Home](#) / [tchua](#) / development

SPACE      RUNNING   STOPPED   CRASHED  
development   ● 0   ● 0   ● 0

[Apps](#)   [Services \(2\)](#)   [Routes](#)   [Member \(1\)](#)   [Settings](#)



Services <span>ADD A SERVICE</span>				
Service	Name	Bound Apps	Plan	Last Operation
 Service Registry Creating service instance.	service-registry	0	free - trial	create in progress
 Config Server	config-server	0	free - trial	create succeeded

9. Wait for a couple of minutes and you should see both the Config Server and Service Registry created successfully.

[Home](#) / [tchua](#) / development

SPACE      RUNNING   STOPPED   CRASHED  
development   ● 0   ● 0   ● 0

[Apps](#)   [Services \(2\)](#)   [Routes](#)   [Member \(1\)](#)   [Settings](#)

Services <span>ADD A SERVICE</span>				
Service	Name	Bound Apps	Plan	Last Operation
 Service Registry	service-registry	0	free - trial	create succeeded
 Config Server	config-server	0	free - trial	create succeeded

## 6. Push microservices and clients to PAS

1. Log in to the cf cli using the following command, selecting the right Org and Space in the process.

```
cf login -a https://api.run.pivotal.io
```

```
Derrick-Chuas-MacBook-Pro:employee-service tmchua$ cf login -a https://api.run.pivotal.io
API endpoint: https://api.run.pivotal.io
```

2. Execute the following chained commands in your terminal to push the Employee microservice to PAS:




```
cd ~/scs-ws/scs-clients-master/employee-service && \
cf push
```

```
1. cf
Derrick-Chuas-MacBook-Pro:employee-service tmchua$ cd ~/scs-ws/scs-clients-master/employee-service && \
> cf push
Pushing from manifest to org tchua / space ws-test as tchua@pivotal.io...
Using manifest file /Users/tmchua/scs-ws/scs-clients-master/employee-service/manifest.yml
Getting app info...
Creating app with these attributes...
+ name: employee-service
+ path: /Users/tmchua/scs-ws/scs-clients-master/employee-service/target/employee-service-0.0.1-SNAPSHOT.jar
+ instances: 2
+ memory: 1G
+ services:
+   config-server
+   service-registry
+ env:
+   JAVA_OPTS
+   TRUST_CERTS
+ routes:
+   employee-service-thankful-genet.cfapps.io
```

3. Verify in the Apps Manager that the Employee microservice has been deployed successfully.

Home / tchua / ws-test / employee-service

APP

employee-service    Running

Overview Services (2) Route (1) Logs Tasks Trace Threads Settings

Events

- Started app  
tchua@pivotal.io 04/19/2019 at 11:02:54 PM
- Mapped route to app  
tchua@pivotal.io 04/19/2019 at 11:02:26 PM
- Created app  
tchua@pivotal.io 04/19/2019 at 11:02:26 PM

Last Push: 11:02 PM 04/19/19 App Summary

Instances / Allocated	Memory / Allocated	Disk / Allocated
2 / 2	0.64 / 2.00 GB	0.33 / 2.00 GB

Processes and Instances

web

Instances	2	Memory Allocated	1 GB	Disk Allocated	1 GB
-----------	---	------------------	------	----------------	------

☐ Autoscaling

#	App Health	CPU	Memory	Disk	Uptime
> 0	↑ Up	6%	331.43 MB	169.18 MB	3 min
> 1	↑ Up	1%	319.08 MB	169.18 MB	3 min

- Execute the following chained commands in your terminal to push the Employee feign client to PAS:





```
cd ~/scs-ws/scs-clients-master/employee-feign-client && \
cf push
```

```
Derrick-Chuas-MacBook-Pro:employee-service tmchua$ cd ~/scs-ws/scs-clients-master/employee-feign-client && \
> cf push
Pushing from manifest to org tchua / space ws-test as tchua@pivotal.io...
Using manifest file /Users/tmchua/scs-ws/scs-clients-master/employee-feign-client/manifest.yml
Getting app info...
Creating app with these attributes...
+ name: employee-feign-client
+ path: /Users/tmchua/scs-ws/scs-clients-master/employee-feign-client/target/employee-feign-client-0.0.1-SNAPSHOT.jar
+ instances: 1
+ memory: 1G
+ services:
+ config-server
+ service-registry
+ env:
+ JAVA_OPTS
+ TRUST_CERTS
+ routes:
+ employee-feign-client-wacky-chimpanzee.cfapps.io
```

- Verify in the Apps Manager that the Employee feign client has been deployed successfully.

Home / tchua / ws-test / employee-feign-client

APP

employee-feign-client     Running [VIEW APP](#)

Overview Services (2) Route (1) Logs Tasks Trace Threads Settings Buildpack: N/A

Events Last Push: 11:09 PM 04/19/19

- Started app tchua@pivotal.io 04/19/2019 at 11:09:08 PM
- Mapped route to app tchua@pivotal.io 04/19/2019 at 11:08:45 PM
- Created app tchua@pivotal.io 04/19/2019 at 11:08:44 PM

App Summary

Instances / Allocated	Memory / Allocated	Disk / Allocated
1 / 1	0.23 / 1.00 GB	0.15 / 1.00 GB

Processes and Instances [View in PCF Metrics](#)

web					
Instances	1	Memory Allocated	1 GB	Disk Allocated	1 GB
Autoscaling					
#	App Health	CPU	Memory	Disk	Uptime
> 0	↑ Up	0%	239.2 MB	157.49 MB	1 min

[SCALE](#)

- Execute the following chained commands in your terminal to push the Bookstore microservice to PAS:

```
cd ~/scs-ws/scs-clients-master/bookstore-service && \
cf push
```



```
Derrick-Chuas-MacBook-Pro:~ tmchua$ cd ~/scs-ws/scs-clients-master/bookstore-service && \
> cf push

Pushing from manifest to org tchua / space ws-test as tchua@pivotal.io...
Using manifest file /Users/tmchua/scs-ws/scs-clients-master/bookstore-service/manifest.yml
Getting app info...
Creating app with these attributes...
+ name: bookstore-service
+ path: /Users/tmchua/scs-ws/scs-clients-master/bookstore-service/target/bookstore-service-0.0.1-SNAPSHOT.jar
+ instances: 1
+ memory: 1G
+ services:
+   config-server
+   service-registry
+ env:
+   JAVA_OPTS
+   TRUST_CERTS
+ routes:
+   bookstore-service-unexpected-baboon.cfapps.io
```

7. Verify in the Apps Manager that the Bookstore microservice has been deployed successfully.

Home / tchua / ws-test / bookstore-service

APP

bookstore-service Running

Overview Services (2) Route (1) Logs Tasks Trace Threads Settings

Events Last Push: 11:09 PM 04/19/19 App Summary

Instances / Allocated	Memory / Allocated	Disk / Allocated
1 / 1	0.27 / 1.00 GB	0.15 / 1.00 GB

Processes and Instances

web					
Instances	1	Memory Allocated	1 GB	Disk Allocated	1 GB
<input type="checkbox"/> Autoscaling					
#	App Health	CPU	Memory	Disk	Uptime
> 0	↑ Up	0%	272.06 MB	157.7 MB	1 min

Events

- Started app  
tchua@pivotal.io 04/19/2019 at 11:09:22 PM
- Mapped route to app  
tchua@pivotal.io 04/19/2019 at 11:08:57 PM
- Created app  
tchua@pivotal.io 04/19/2019 at 11:08:57 PM

8. Execute the following chained commands in your terminal to push the Bookstore feign hystrix client to PAS:

```
cd ~/scs-ws/scs-clients-master/bookstore-ui-feign-hystrix && \
cf push
```





```
Derrick-Chuas-MacBook-Pro:~ tmchua$ cd ~/scs-ws/scs-clients-master/bookstore-ui-feign-hystrix && \
> cf push

Pushing from manifest to org tchua / space ws-test as tchua@pivotal.io...
Using manifest file /Users/tmchua/scs-ws/scs-clients-master/bookstore-ui-feign-hystrix/manifest.yml
Getting app info...
Creating app with these attributes...
+ name: bookstore-ui-feign-hystrix
+ path: /Users/tmchua/scs-ws/scs-clients-master/bookstore-ui-feign-hystrix/target/bookstore-ui-resttemplate-hystrix-0.0.1-SNAPSHOT.jar
+ instances: 1
+ memory: 1G
+ services:
+   config-server
+   service-registry
+ env:
+   JAVA_OPTS
+   TRUST_CERTS
+ routes:
+   bookstore-ui-feign-hystrix-noisy-gerenuk.cfapps.io
```

- Verify in the Apps Manager that the Bookstore feign hystrix client has been deployed successfully.

Home / tchua / ws-test / bookstore-ui-feign-hystrix

APP

 bookstore-ui-feign-hystrix    ● Running

Overview Services (2) Route (1) Logs Tasks Trace Threads Settings

Events Last Push: 11:09 PM 04/19/19

- Started app  
tchua@pivotal.io 04/19/2019 at 11:09:56 PM
- Mapped route to app  
tchua@pivotal.io 04/19/2019 at 11:09:21 PM
- Created app  
tchua@pivotal.io 04/19/2019 at 11:09:20 PM

App Summary

Instances / Allocated	Memory / Allocated	Disk / Allocated
1 / 1	0.26 / 1.00 GB	0.15 / 1.00 GB

Processes and Instances

web

Instances	1	Memory Allocated	1 GB	Disk Allocated	1 GB
-----------	---	------------------	------	----------------	------

☐ Autoscaling

#	App Health	CPU	Memory	Disk	Uptime
> 0	↑ Up	0%	263.34 MB	158.24 MB	1 min

## 7. Test and observe on Cloud

1. Take note of the route for each of the four applications that were pushed to PAS by going to the Apps tab in your space. Note that there will be five apps because of the Employee API from the previous workshop exercise.

SPACE RUNNING STOPPED CRASHED  
ws-test ● 5 ● 0 ● 0

Apps (5) Services (4) Routes (6) Member (1) Settings

Apps

Status	Name	Instances	Memory	Last Push	Route
Running	bookstore-service	1	1 GB	11 hours ago	<a href="https://bookstore-service-unexpected-baboon.cfapps.io">https://bookstore-service-unexpected-baboon.cfapps.io</a>
Running	bookstore-ui-feign-hystrix	1	1 GB	11 hours ago	<a href="https://bookstore-ui-feign-hystrix-noisy-gerenuk.cfapps.io">https://bookstore-ui-feign-hystrix-noisy-gerenuk.cfapps.io</a>
Running	employee-api	2	2 GB	4 days ago	<a href="https://employee-api-rested-puku.cfapps.io">https://employee-api-rested-puku.cfapps.io</a>
Running	employee-feign-client	1	1 GB	11 hours ago	<a href="https://employee-feign-client-wacky-chimpanzee.cfapps.io">https://employee-feign-client-wacky-chimpanzee.cfapps.io</a>
Running	employee-service	2	2 GB	11 hours ago	<a href="https://employee-service-thankful-genet.cfapps.io">https://employee-service-thankful-genet.cfapps.io</a>

2. Invoke the employee microservice “employees” endpoint directly by executing the following in Postman.

GET `https://{employee.microservice.route}/employees`

You should observe the following if it is running successfully.

GET `https://employee-service-thankful-genet.cfapps.io/employees`

Params Authorization Headers Body Pre-request Script Tests

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

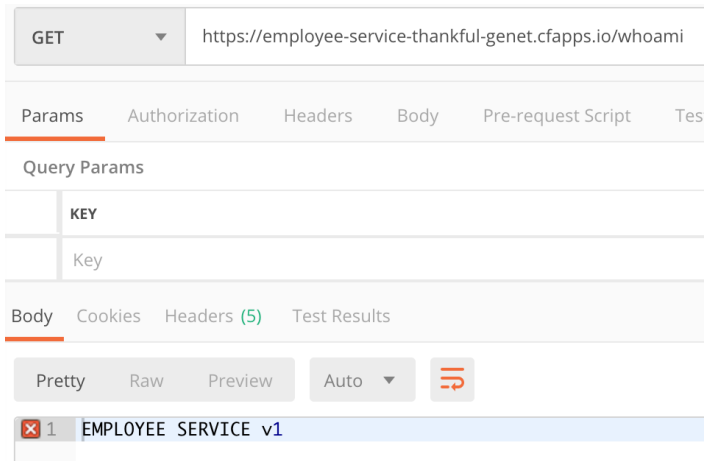
```

1 {
2   "_embedded": {
3     "employees": [
4       {
5         "name": "pas",
6         "_links": {
7           "self": {
8             "href": "https://employee-service-thankful-genet.cfapps.io/employees/1"
9           },
10          "employee": {
11            "href": "https://employee-service-thankful-genet.cfapps.io/employees/1"
12          }
13        }
14      },
15      {
16        "name": "lucia",
17        "_links": {
18          "self": {
19            "href": "https://employee-service-thankful-genet.cfapps.io/employees/2"
20          },
21          "employee": {
22            "href": "https://employee-service-thankful-genet.cfapps.io/employees/2"
23          }
24        }
25      }
26    ]
27  }
28 }
  
```

- Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET https://{employee.microservice.route}/whoami

You should observe the following if it is running successfully.



- Wait for the instructor to update employee microservice config in the github config store to reflect “v2”.

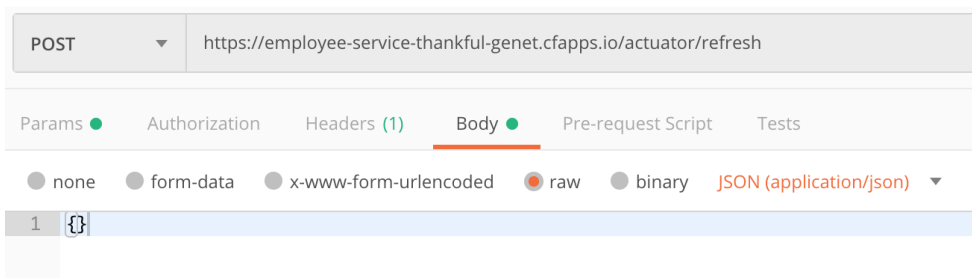


- Refresh the employee microservice’s copy of the configuration by executing the following command in Postman.

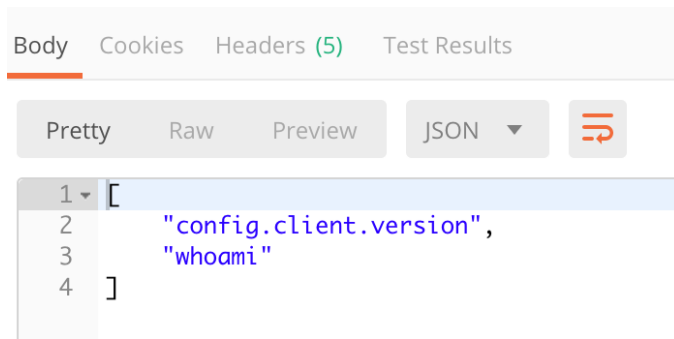
POST https://{employee.microservice.route}/actuator/refresh

**Content-Type:** application/json

**Body:** {}



You should observe the following if executed successfully.



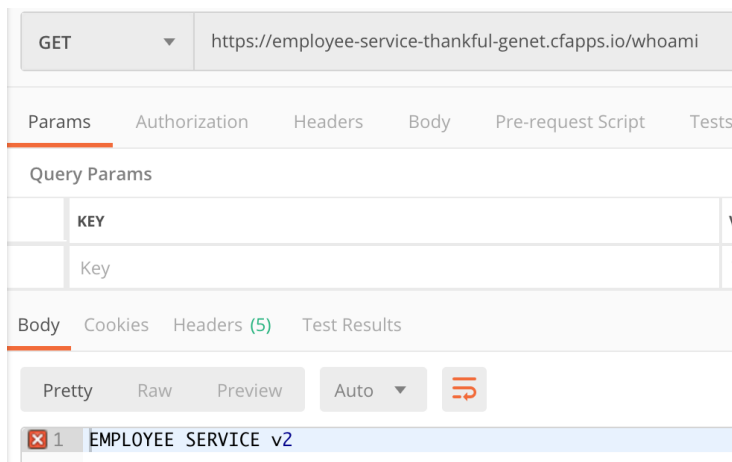
```

Body Cookies Headers (5) Test Results
Pretty Raw Preview JSON
1 [
2   "config.client.version",
3   "whoami"
4 ]
  
```

6. Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET `https://{employee.microservice.route}/whoami`

You should observe the following (a change in the version number) if it is running successfully.



```

GET https://employee-service-thankful-genet.cfapps.io/whoami
Params Authorization Headers Body Pre-request Script Tests
Query Params
KEY v
Key \
Body Cookies Headers (5) Test Results
Pretty Raw Preview Auto
1 EMPLOYEE SERVICE v2
  
```

7. Wait for the instructor to revert employee microservice config in the github config store to reflect “v1”.



```

Branch: master scs-config-store / employee-service.properties
derrick81 Update employee-service.properties
1 contributor
Executable File | 3 lines (1 sloc) | 28 Bytes
1 whoami=EMPLOYEE SERVICE v1
2
  
```



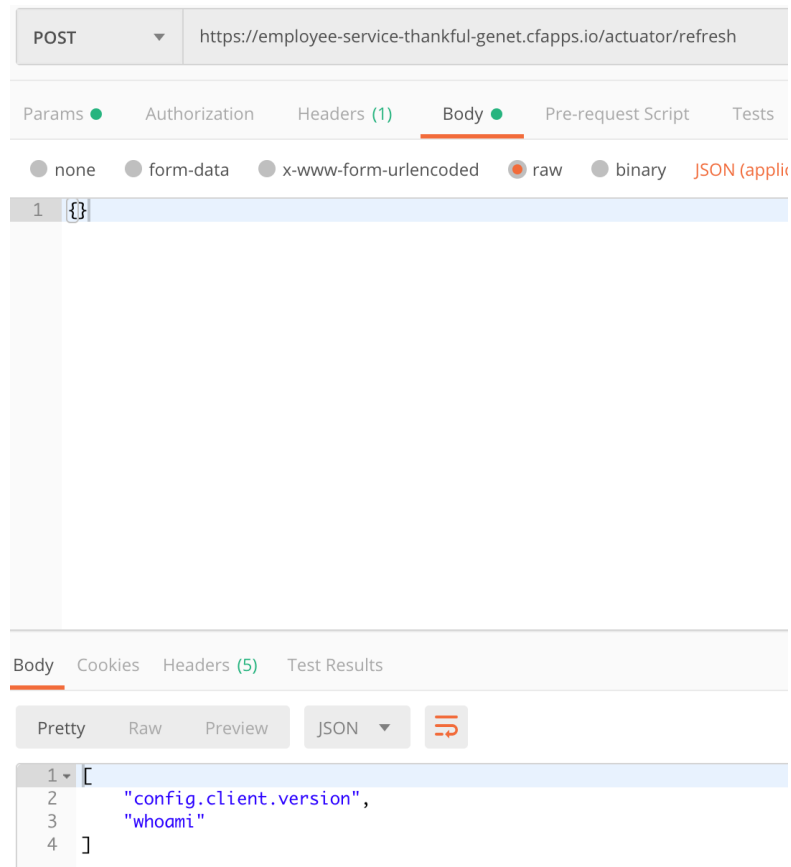
8. Refresh the employee microservice's copy of the configuration by executing the following command in Postman.

POST `https://{employee.microservice.route}/actuator/refresh`

**Content-Type:** application/json

**Body:** {}

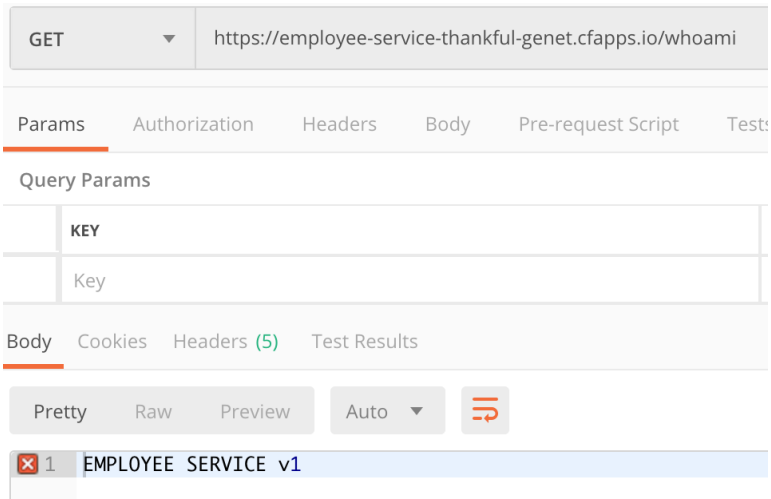
You should observe the following if executed successfully.



9. Invoke the employees microservice “whoami” endpoint directly by executing the following in Postman.

GET `https://{employee.microservice.route}/whoami`

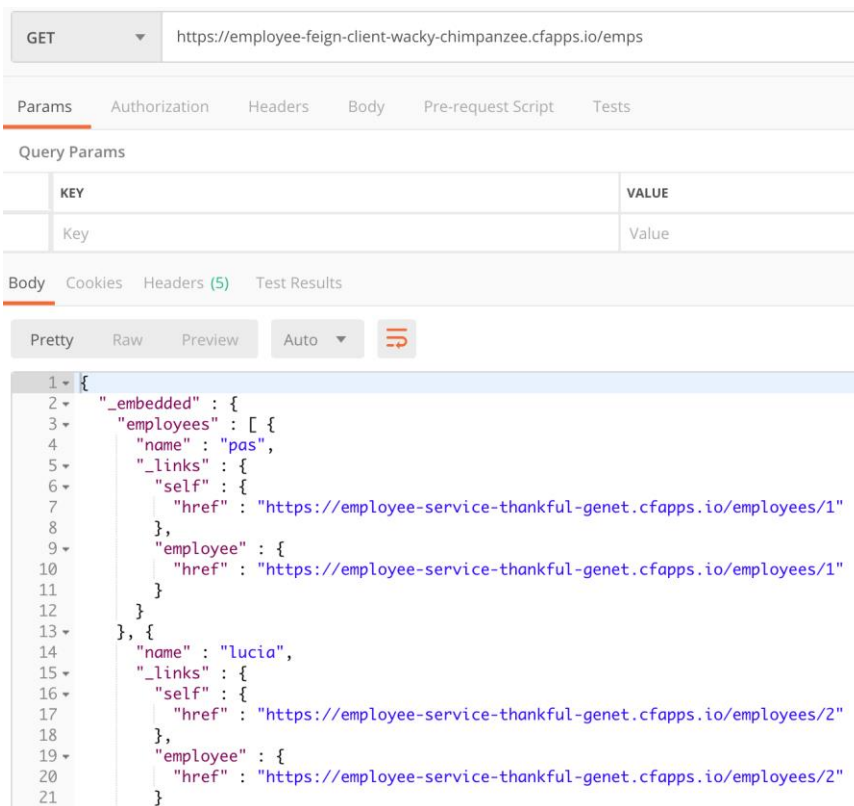
You should observe the following (a change in the version number) if it is running successfully.



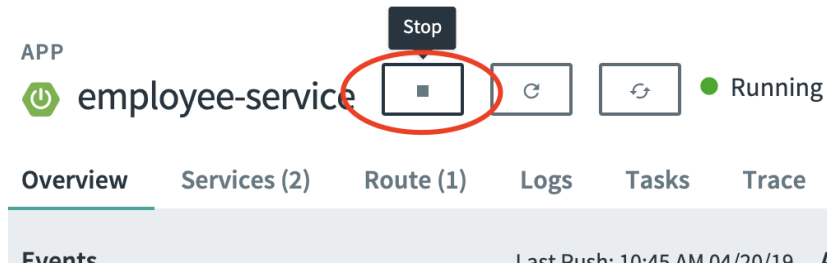
- Now consume the employee microservice through a feign client. Invoke the client's "employees" endpoint directly by executing the following in Postman.

GET <https://{employee.feign.client.route}/emps>

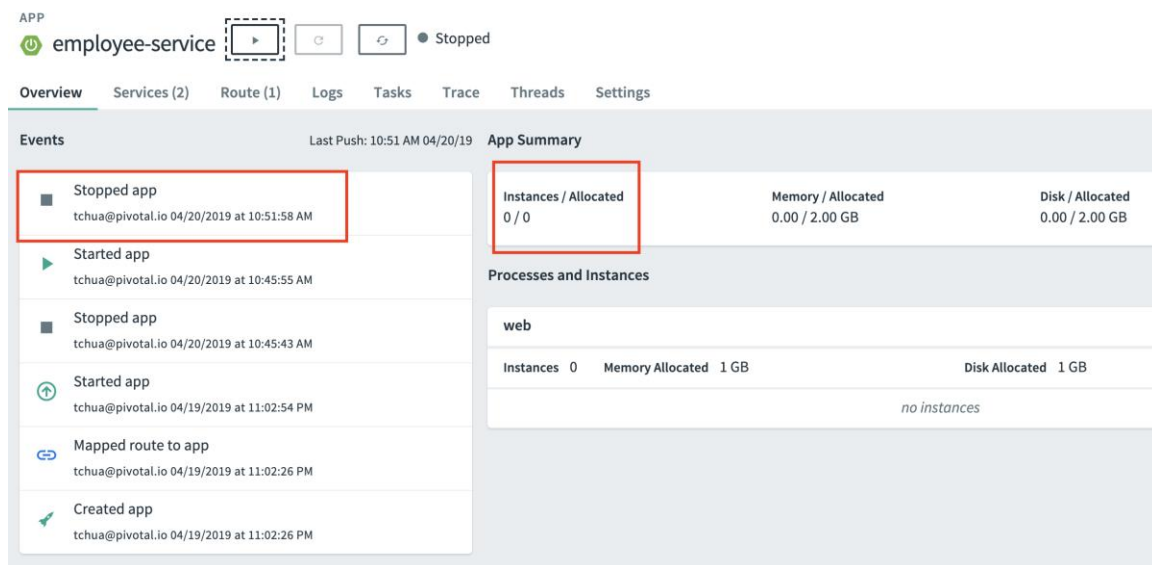
You should observe the following if it is running successfully.



- Now turn off the employee microservice to observe the clients response when the backing microservice is unavailable.  
Turn off the employee microservice by clicking on the “Stop” button on the employee microservice’s app page.



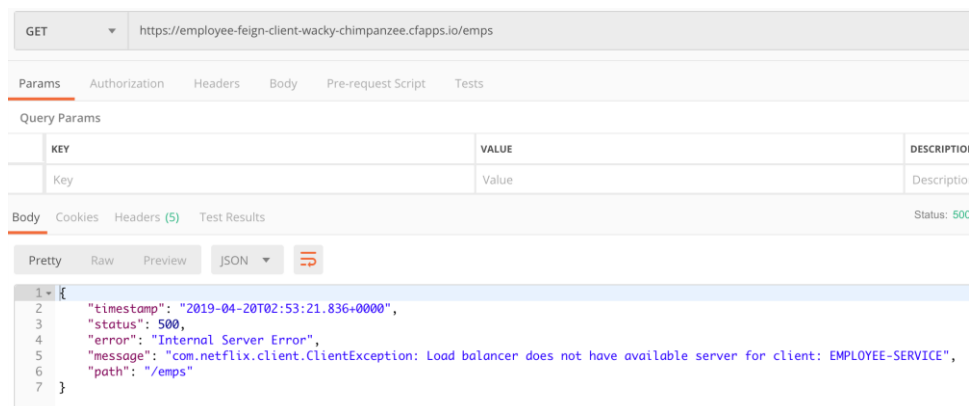
After confirming the stop, you should see that the app has indeed stopped running.



- Invoke the client’s “employees” endpoint directly by executing the following in Postman.

GET <https://employee-feign-client.wacky-chimpanzee.cfapps.io/emps>

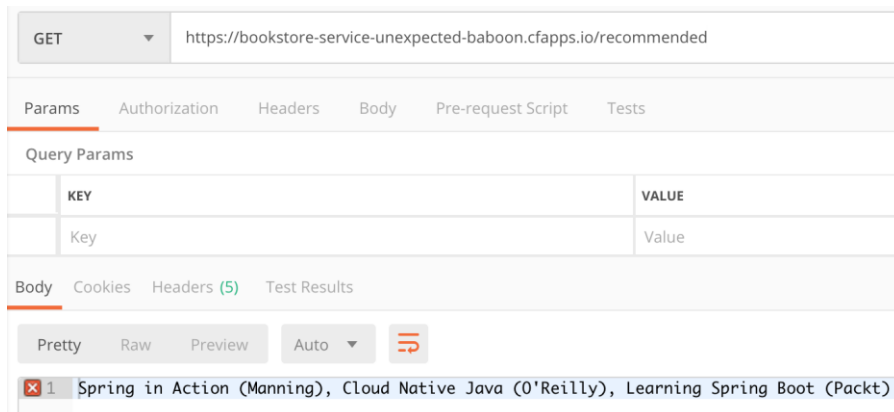
You should observe the following failure.



- Invoke the bookstore microservice “recommended” endpoint directly by executing the following in Postman.

GET <https://{bookstore.microservice.route}/recommended>

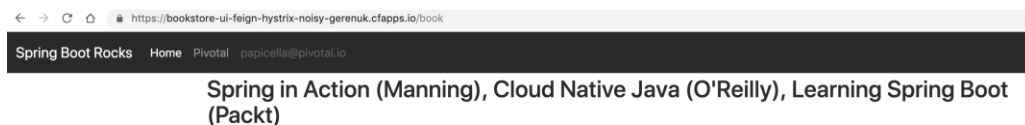
You should observe the following if it is running successfully.



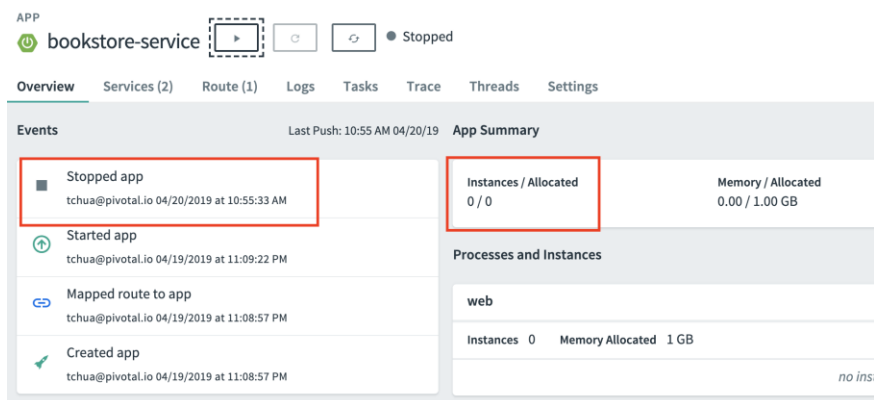
- Invoke the bookstore web UI which will consume the bookstore microservice. Visit the following URL in the browser:

<https://{bookstore.ui.feign.hystrix.route}/book>

You should see the following page.



- Now turn off the bookstore microservice to observe the clients response when the backing microservice is unavailable. Turn off the employee microservice by clicking on the “Stop” button on the bookstore microservice’s app page.



16. Invoke the bookstore web UI which will consume the bookstore microservice. Visit the following URL in the browser:

`https://{bookstore.ui.feign.hystrix.route}/book`

You should see the following page.

