

Asegurando nuestras APIs - Spring Cloud

Elaborado por:

- José Amadeo Martín Díaz Díaz

Source Code:

TABLA DE CONTENIDOS

Asegurando nuestras APIs	2
Proteger la comunicación externa con HTTPs	2
Configurar el certificado en el proyecto Gateway	4
Asegurar el acceso al Discovery Service	5
Cambios en el servidor de Eureka	5
Cambios en los clientes Eureka	8
Testeando el protegido servidor Eureka	9

Asegurando nuestras APIs

Proteger la comunicación externa con HTTPS

En esta sección, aprenderemos cómo evitar la escucha de las comunicaciones externas, por ejemplo desde Internet, a través de las API públicas expuestas por el servidor gateway.

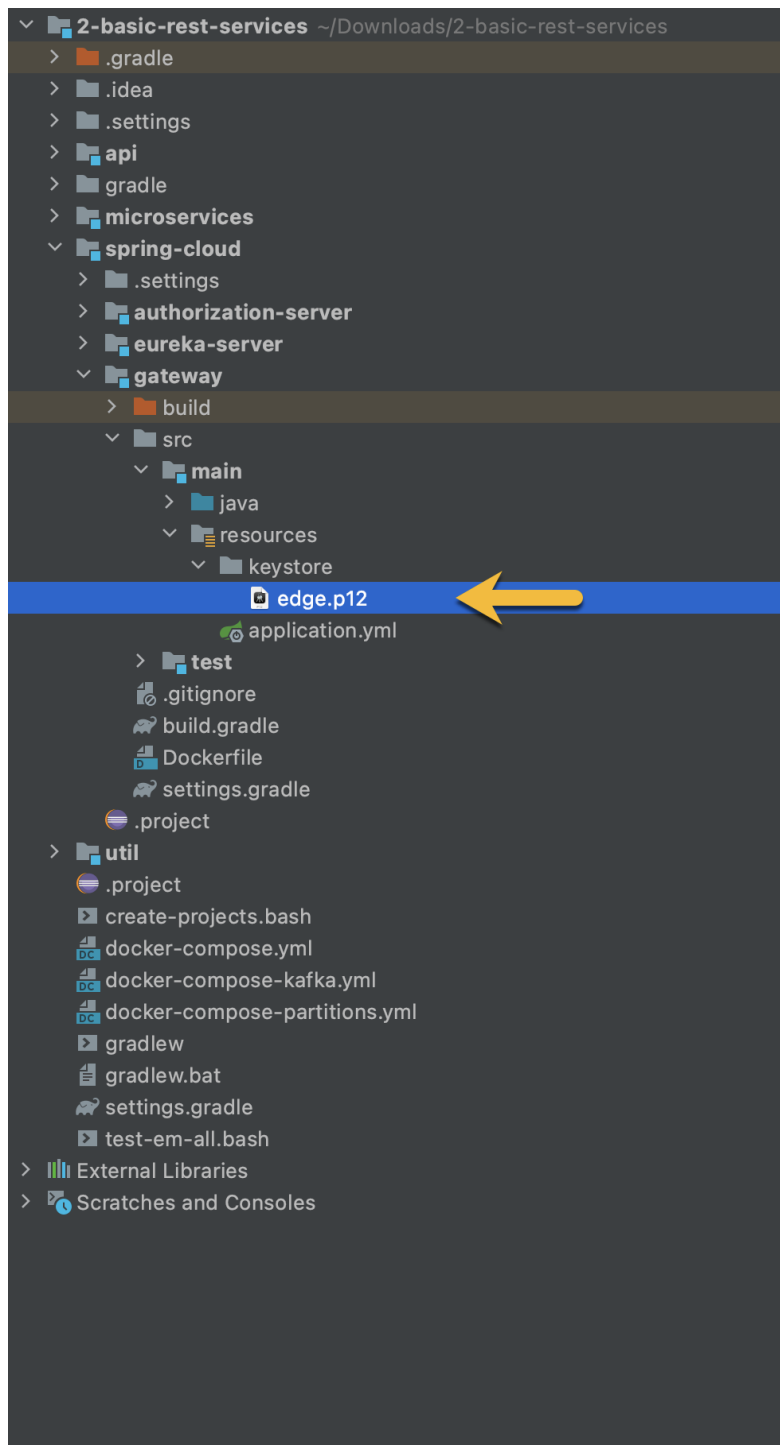
Usaremos HTTPS para cifrar la comunicación. Para usar HTTPS, debemos hacer lo siguiente:

- **Crear un certificado:** Crearemos nuestro propio certificado autofirmado, suficiente para fines de desarrollo.
- **Configurar el servidor gateway:** debe configurarse para aceptar solo tráfico externo basado en HTTPS mediante el certificado.

Se creará el certificado auto-firmado con el siguiente comando:

```
keytool -genkeypair -alias localhost -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore edge.p12 -validity 3650
```

El comando te preguntará por varios parámetros. Cuando se consulte por un password, por simplicidad, colocaremos: **password**. Para el resto de parámetros se aceptarán los valores por defecto. El archivo creado es **edge.p12** colocado en el proyecto **gateway** en el directorio **src/main/resources/keystore**.



Certificado auto-firmado

Configurar el certificado en el proyecto Gateway

Para configurar el servidor **gateway** y pueda usar el certificado y HTTPS, se agrega lo siguiente al **application.yml** en el proyecto de puerta de enlace:

```
server.port: 8443
```

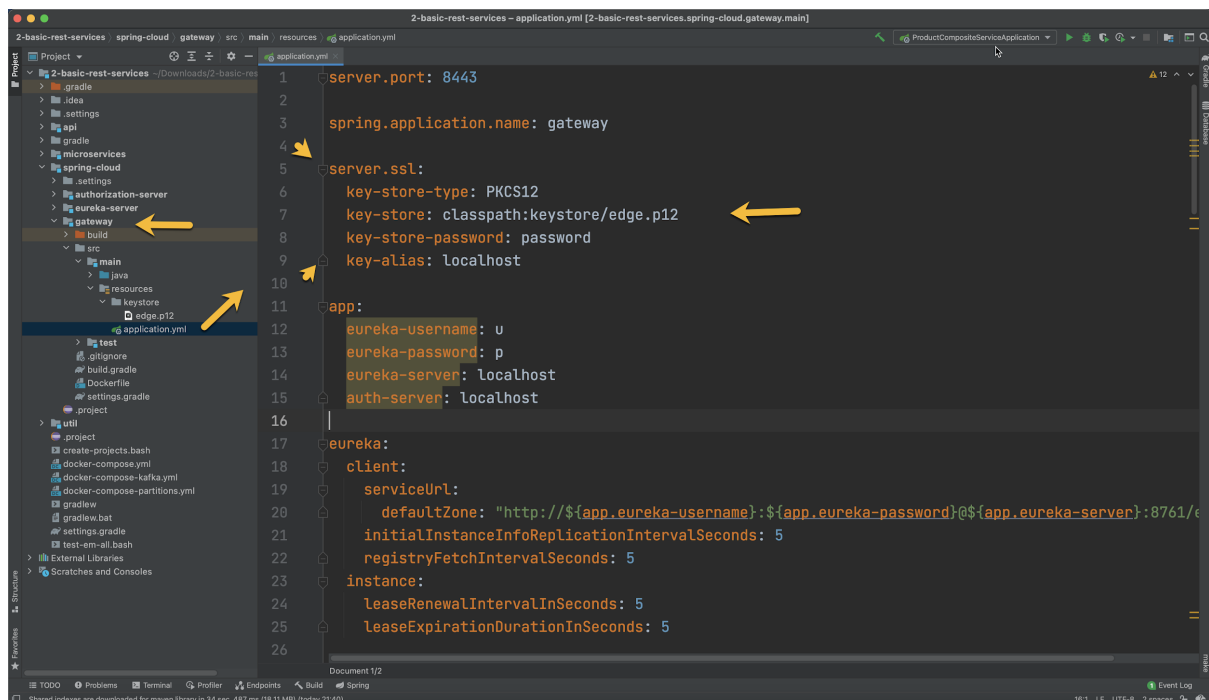
```
server.ssl:
```

```
key-store-type: PKCS12
```

```
key-store: classpath:keystore/edge.p12
```

```
key-store-password: password
```

```
key-alias: localhost
```



También se debe hacer los cambios en los archivos:

- **docker-compose*.yml**
- El test script, **test-em-all.bash**

Asegurar el acceso al Discovery Service

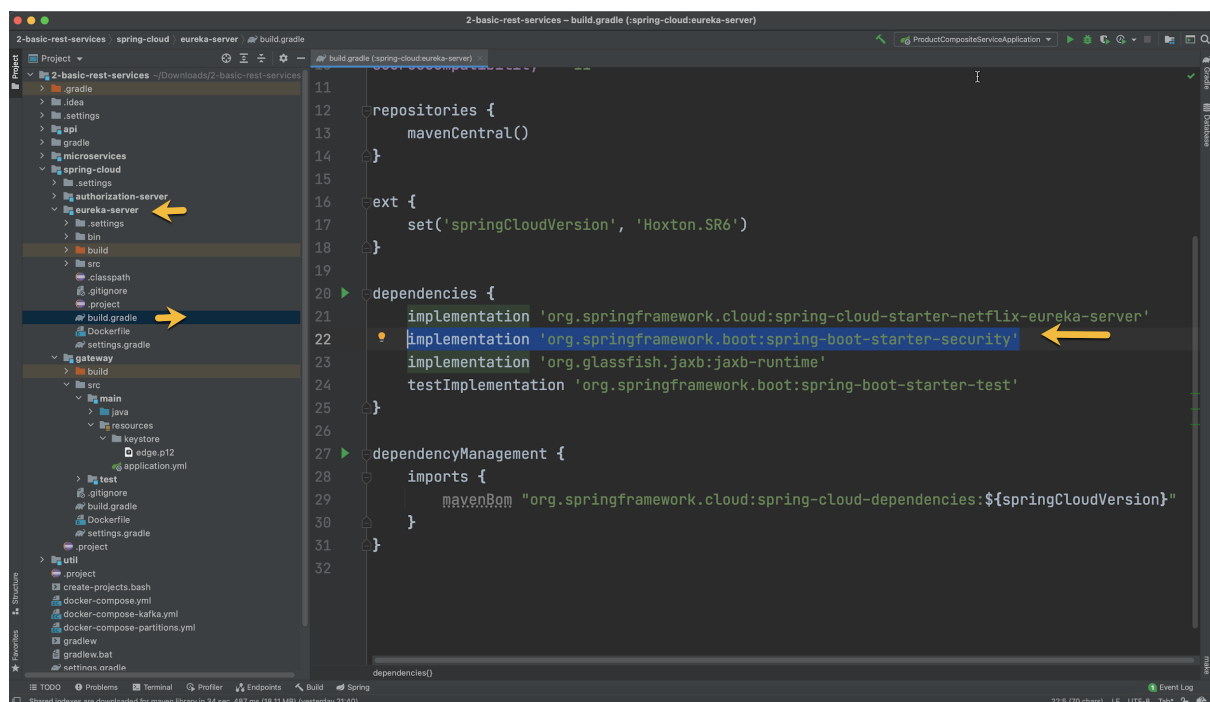
Ahora usaremos la autenticación básica HTTP para restringir el acceso a las API y páginas web en el discovery service, Netflix Eureka; es decir, le pediremos al usuario que proporcione un nombre de usuario y una contraseña para obtener acceso. Se requieren cambios tanto en el servidor de Eureka como en los clientes de Eureka que se describen a continuación.

Cambios en el servidor de Eureka

Para proteger los servidores de Eureka, los siguientes cambios deben ser aplicados:

1. Agregar la dependencia

implementation 'org.springframework.boot:spring-boot-starter-security'



2. La configuración de seguridad se agrega mediante la clase se.magnus.springcloud.eurekaserver.SecurityConfig:

```

11
12 @Configuration
13 public class SecurityConfig extends WebSecurityConfigurerAdapter{
14
15     private final String username;
16     private final String password;
17
18     @Autowired
19     public SecurityConfig(
20         @Value("${app.eureka-username}") String username,
21         @Value("${app.eureka-password}") String password
22     ) {
23         this.username = username;
24         this.password = password;
25     }
26
27     @Override
28     public void configure(AuthenticationManagerBuilder auth) throws Exception {
29         auth.inMemoryAuthentication()
30             .passwordEncoder(NoOpPasswordEncoder.getInstance())
31             .withUser(username).password(password)
32             .authorities("USER");
33     }

```

El usuario se define en **configure(AuthenticationManagerBuilder auth)** como se muestra en la figura anterior.

El username y password es inyectado al constructor desde el archivo de configuración.

```

28 @ @ public void configure(AuthenticationManagerBuilder auth) throws Exception {
29     auth.inMemoryAuthentication()
30         .passwordEncoder(NoOpPasswordEncoder.getInstance())
31         .withUser(username).password(password)
32         .authorities("USER");
33 }
34
35 @Override
36 @ @ protected void configure(HttpSecurity http) throws Exception {
37     http
38         // Disable CSRF to allow services to register themselves with Eureka
39         .csrf() .csrfConfigurer<HttpSecurity>
40             .disable() HttpSecurity
41         .authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptU
42             .anyRequest().authenticated()
43             .and() HttpSecurity
44             .httpBasic();
45 }
46 }
47

```

Todas las APIs y páginas web son protegidos usando autenticación básica HTTP con la definición **configure(HttpSecurity http)**.

Las credenciales se encuentran en el application.yml:

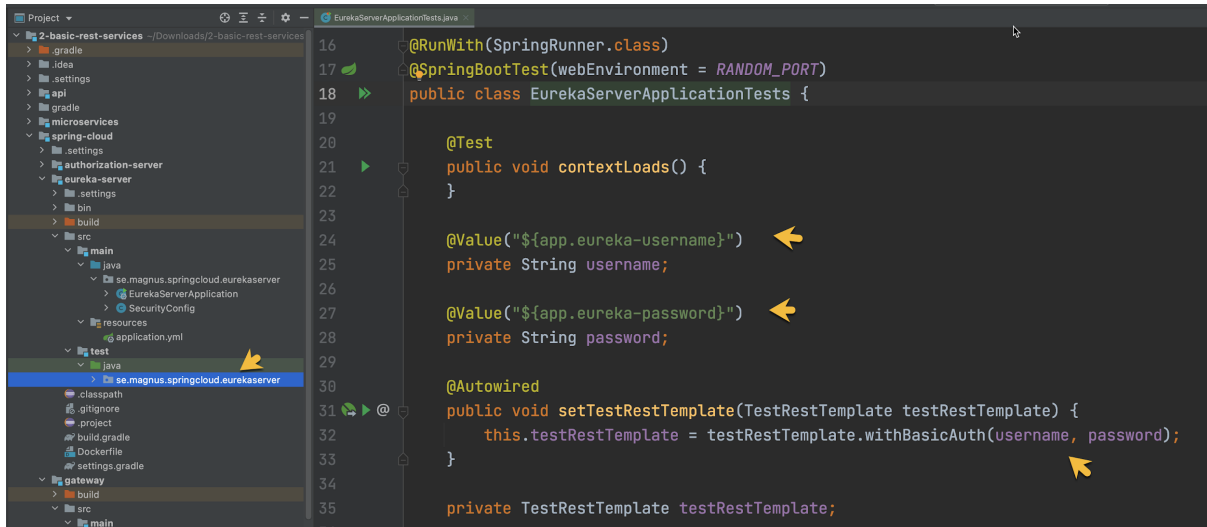
```

1 server:
2   port: 8761
3
4 app:
5   eureka-username: u
6   eureka-password: p
7
8 eureka:
9   instance:
10    hostname: localhost
11   client:
12    registerWithEureka: false
13    fetchRegistry: false
14    serviceUrl:

```

Finalmente, la clase de prueba,

se.magnus.springcloud.eureka.server.EurekaServerApplicationTests, usa las credenciales del archivo de configuración cuando prueba las API del servidor Eureka:



```
16 @RunWith(SpringRunner.class)
17 @SpringBootTest(webEnvironment = RANDOM_PORT)
18 public class EurekaServerApplicationTests {
19
20     @Test
21     public void contextLoads() {
22     }
23
24     @Value("${app.eureka-username}")
25     private String username;
26
27     @Value("${app.eureka-password}")
28     private String password;
29
30     @Autowired
31     public void setTestRestTemplate(TestRestTemplate testRestTemplate) {
32         this.testRestTemplate = testRestTemplate.withBasicAuth(username, password);
33     }
34
35     private TestRestTemplate testRestTemplate;
36 }
```

Los anteriores son los pasos necesarios para restringir el acceso a las API y las páginas web del servidor de Discovery Service, Netflix Eureka. Ahora utilizará la autenticación básica HTTP y requerirá que un usuario proporcione un nombre de usuario y una contraseña para obtener acceso. En la siguiente sección, aprenderemos cómo configurar los clientes de Netflix Eureka para que pasen credenciales al acceder al servidor de Netflix Eureka.

Cambios en los clientes Eureka

Para los clientes de Eureka, las credenciales deben especificarse en la URL de conexión para el servidor de Eureka. Esto se especifica en el archivo de configuración de cada cliente, **application.yml**, de la siguiente manera:


```

56  app:
57    eureka-username: u
58    eureka-password: p
59    eureka-server: localhost
60    auth-server: localhost
61
62  eureka:
63    client:
64      serviceUrl:
65        defaultZone: "http://${app.eureka-username}:${app.eureka-password}@${app.eureka-server}:8761/eureka/"
66        initialInstanceInfoReplicationIntervalSeconds: 5
67        registryFetchIntervalSeconds: 5
68    instance:
69      leaseRenewalIntervalInSeconds: 5
70      leaseExpirationDurationInSeconds: 5

```

Este cambio se ha aplicado a los proyectos: **product-composite-service**, **review-service**, **recommendation-service**, **product-service**.

Testeando el protegido servidor Eureka

Una vez que el servidor protegido de Eureka está en funcionamiento, debemos proporcionar credenciales válidas para poder acceder a sus API y páginas web.

Por ejemplo, solicitar instancias registradas al servidor de Eureka se puede realizar mediante el siguiente comando curl:

```
curl -H "accept:application/json" https://u:p@localhost:8443/eureka/api/apps -ks | jq -r '.applications.application[].instance[].instanceId'
```

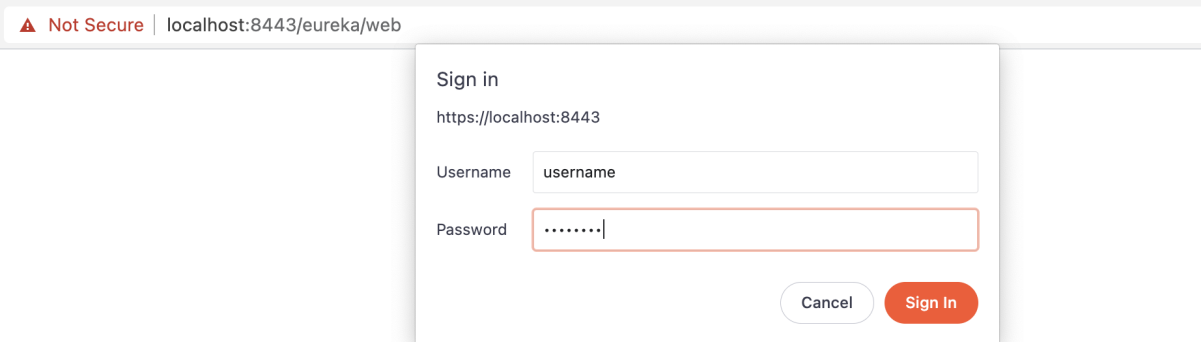
Para ver eso, primero correr la aplicación con: **./test-em-all.bash start**

```

eyJzdWl0Ij01JtYwduXMiLCJleHAiOjIyMTcxNjMzMzUsImF1dGhvcml0aWVzIjpbILJPTEVfVVFUJdLCJqdGkiOiJhVnBwa0taVjZLUGZpMzJRRUE4ZzJzaXE4OWc9IiwiaY
2xpZW50X2lkIjoicmVhZGVyIiwic2Nvc6UiOlsicHJvZHVjdDpyZWFKIL19.RJY6VW0gUDt-K3NgrMIIt-MZQTzLDh5U5uW5w-4vZzKzce8L9oTsiaL7uz6QG6Pps6kWWmVbtAN
nu1ux91prScD2XtFNFoTF0PB5f5Uga_4URY-BxLNk8o-W0dQ9JBcTuv6P084Tx07FqAyQc7KwytdZd5ais6QTj1h0SjH1FI5dB4VVPdxLYQvqvbzxVx_Pw0XL8LfX9NXz61
t_rD0pVN2Wb15jIZHDFDnLWjHtwyK7bzVMCBYrFBzL-tgovVJGAAU0Z8SedgaEUo38EXskcg2fGyi1NLtKLEmdWE01GTeVnNw2nvt29ALJm4m0dFsLIp4C28CfpULj01HSveS
RwQ" -X DELETE -s -w "%{http_code}"
- Response Body:
> curl -H "accept:application/json" https://u:p@localhost:8443/eureka/api/apps -ks | jq -r '.applications.application[].instance[].in
stanceId'
4124b38edbc3:gateway:8443
b015b5270ab4:auth-server:9999
51a7f267ee0e:product-composite:8080
3750019f162b:product:8080
29113af2b37e:review:8080
49173a3db209:recommendation:8080
~/Downloads/2-basic-rest-services

```

Al acceder a la página web en **https://localhost:8443/eureka/web**, primero tenemos que aceptar una conexión insegura, ya que nuestro certificado es autofirmado, y luego tenemos que proporcionar credenciales válidas, como se especifica en la configuración anterior archivos:



Sign in

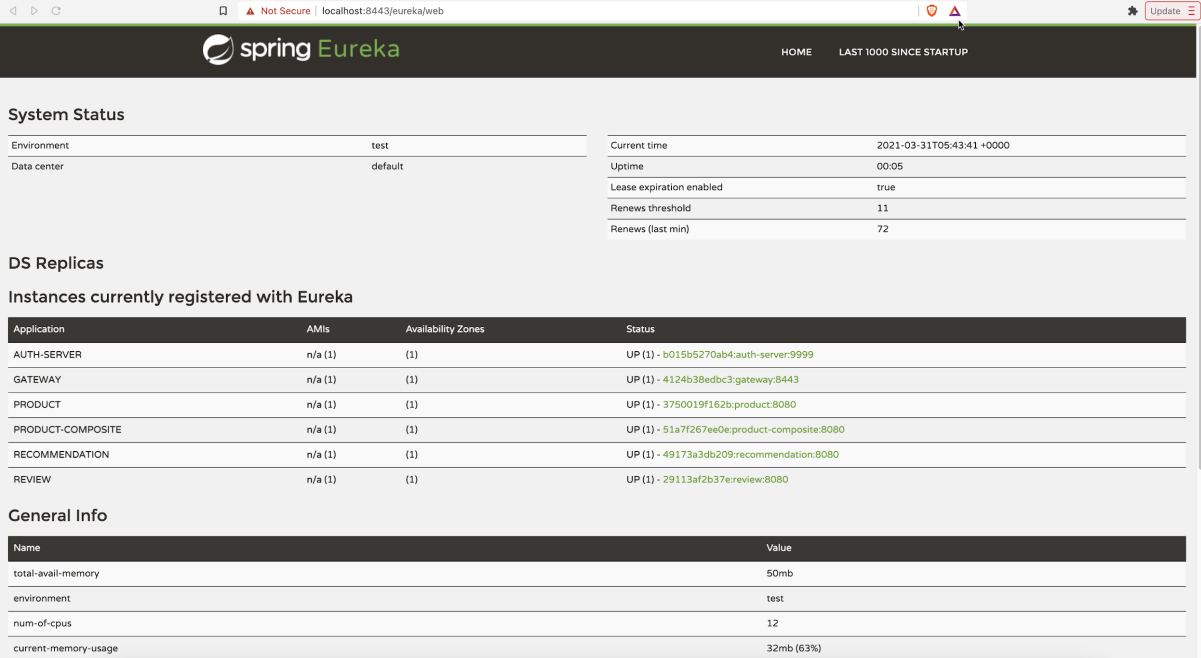
https://localhost:8443

Username

Password

Cancel Sign In

usuario: u
password: p



spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2021-03-31T05:43:41 +0000
Data center	default	Uptime	00:05
		Lease expiration enabled	true
		Renews threshold	11
		Renews (last min)	72

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTH-SERVER	n/a (1)	(1)	UP (1) - b015b5270ab4-auth-server:9999
GATEWAY	n/a (1)	(1)	UP (1) - 4124b38edbc3-gateway:8443
PRODUCT	n/a (1)	(1)	UP (1) - 3750019f162b-product:8080
PRODUCT-COMPOSITE	n/a (1)	(1)	UP (1) - 51a7f267ee0e-product-composite:8080
RECOMMENDATION	n/a (1)	(1)	UP (1) - 49173a3db209-recommendation:8080
REVIEW	n/a (1)	(1)	UP (1) - 29113af2b37e-review:8080

General Info

Name	Value
total-avail-memory	50mb
environment	test
num-of-cpus	12
current-memory-usage	32mb (63%)

Con esto concluye la sección sobre cómo restringir el acceso al servidor Netflix Eureka. En la siguiente sección, aprenderemos cómo usar OAuth 2.0 y OpenID Connect para autenticar y autorizar el acceso a las API.