* Created product services
* A simple spring boot application
* Created service-registry application

Dependency added

* spring-cloud-starter
* spring-cloud-starter-netflix-eureka-serve

Enable eureka server by annotation in main class

* *@EnableEurekaServer*

Added below configuration to the application.yaml file

server:  
 port: 8761  
  
eureka:  
 instance:  
 hostname: localhost  
  
 client:  
 register-with-eureka: false  
 fetch-registry: false

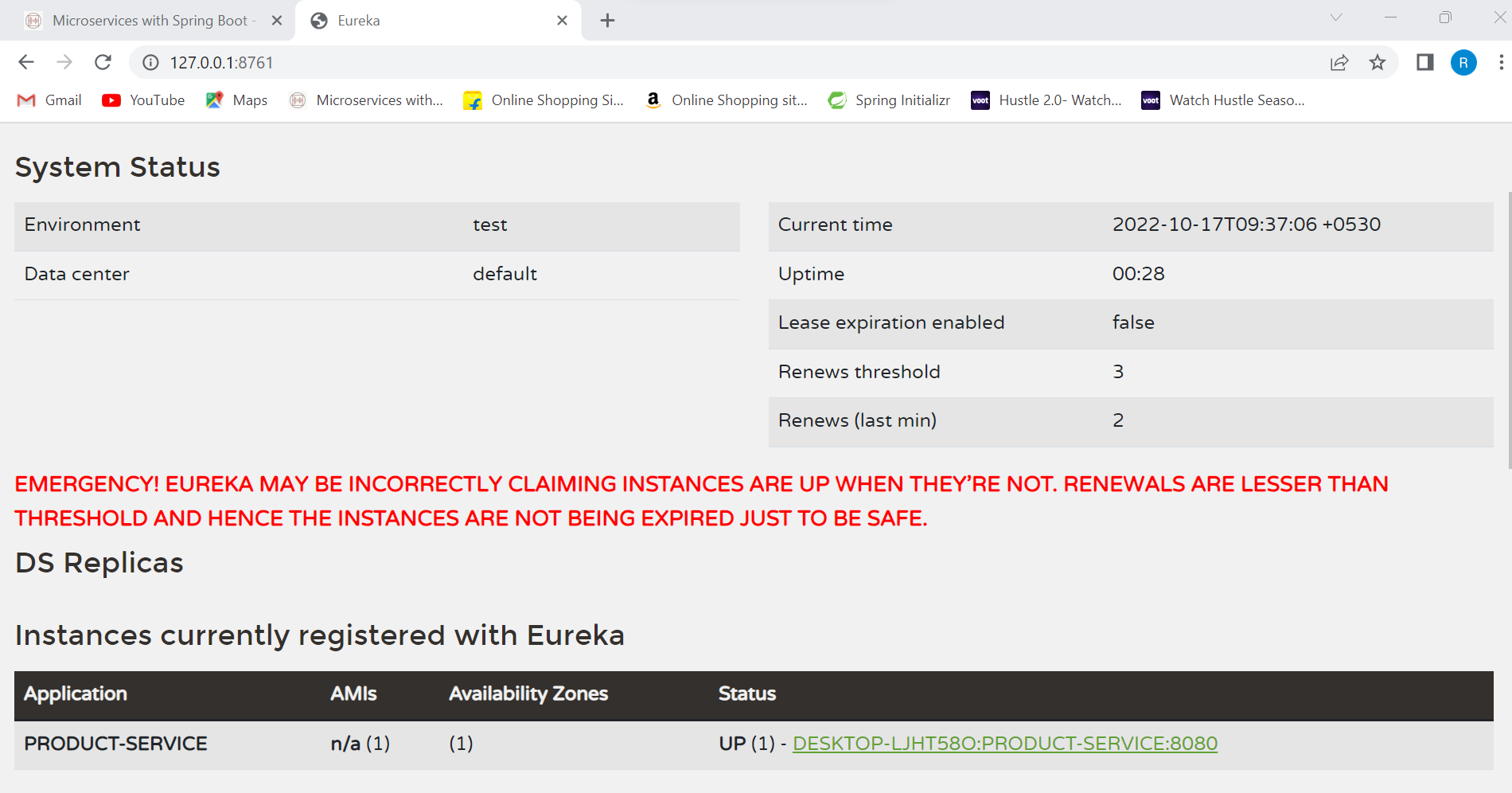
Run the service-registry

And on this port 127.0.0.1: 8761 port eureka server is up and running

**Product-service Add as an Eureka Client :**

* Added the eureka client dependency to product service
* Now we have to add configuration for this product service to which particular eureka server is goes and connect to

Service-registry is running on port <http://127.0.0.1:8761/>



**Feign Client** **-**: Feign client is used to call a microservices from another microservices.

* first we have to add feign client dependency in the our microservices application.
* Annotate main class of services as
* @EnableFeignClients

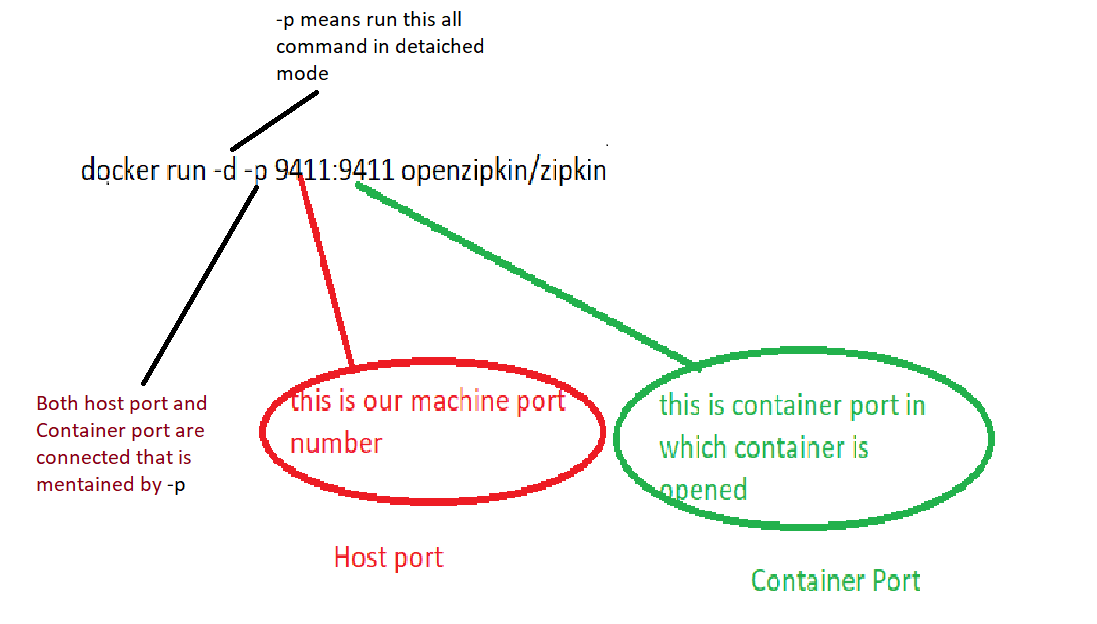
**Tracing the logs ::**

To trace the logs of all the microservices in proper way we need to install zipkin and sleuth.

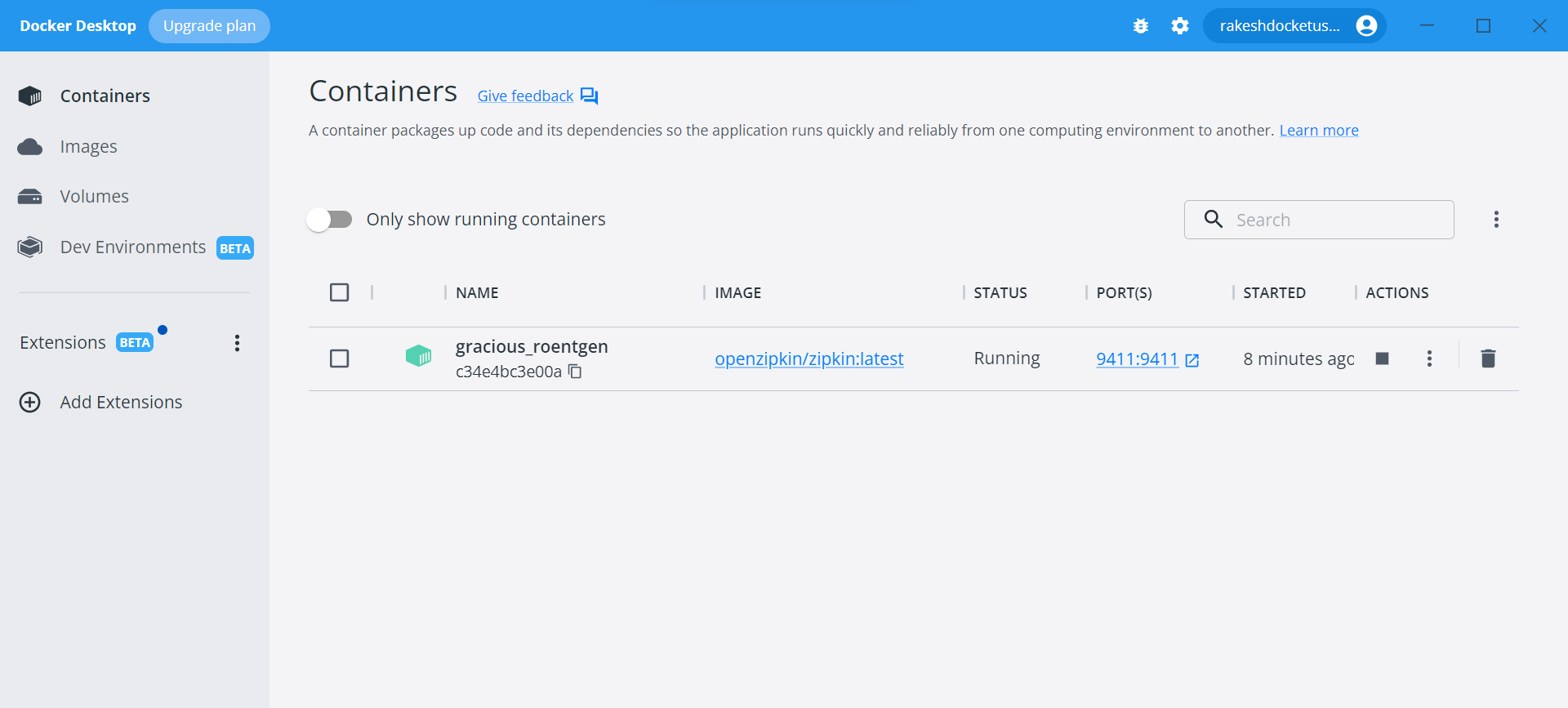
Paste this below command to windows PowerShell

docker run -d -p 9411:9411 openzipkin/zipkin

means of this above command is that – : run a docker image that is openzipkin/zipkin, we have to run a image on your machine at port number 9411



* Now after running this above command we could see that the open zipkin image would be running in docker



* Now we’ll add the zipkin and sleuth dependency in each microservices application.

**ALL DONE NOW WE COULD SEE ALL THE LOGS IS BEING CAPTURED IN ZIPKIN**

**Now we are going to created payment service-:**

* Created a Payment Service.
* Added the following dependencies
* Spring web
* Spring data jpa
* MySQL driver
* Lombok
* Eureka discovery client
* Config Client
* Zipkin client
* Sleuth
* Cloud bootstrap

**API GATEWAY**

CREATING API GATEWAY:

1. Create spring boot application from spring initializer

Add the following dependencies:

* **Gateway** spring cloud routing
* Cloud bootstrap
* Spring Reactive Web
* Zipkin
* Sleuth
* Eureka discovery client
* Lombok
* Spring boot actuator
* Config client

1. Added the following config in application.yaml

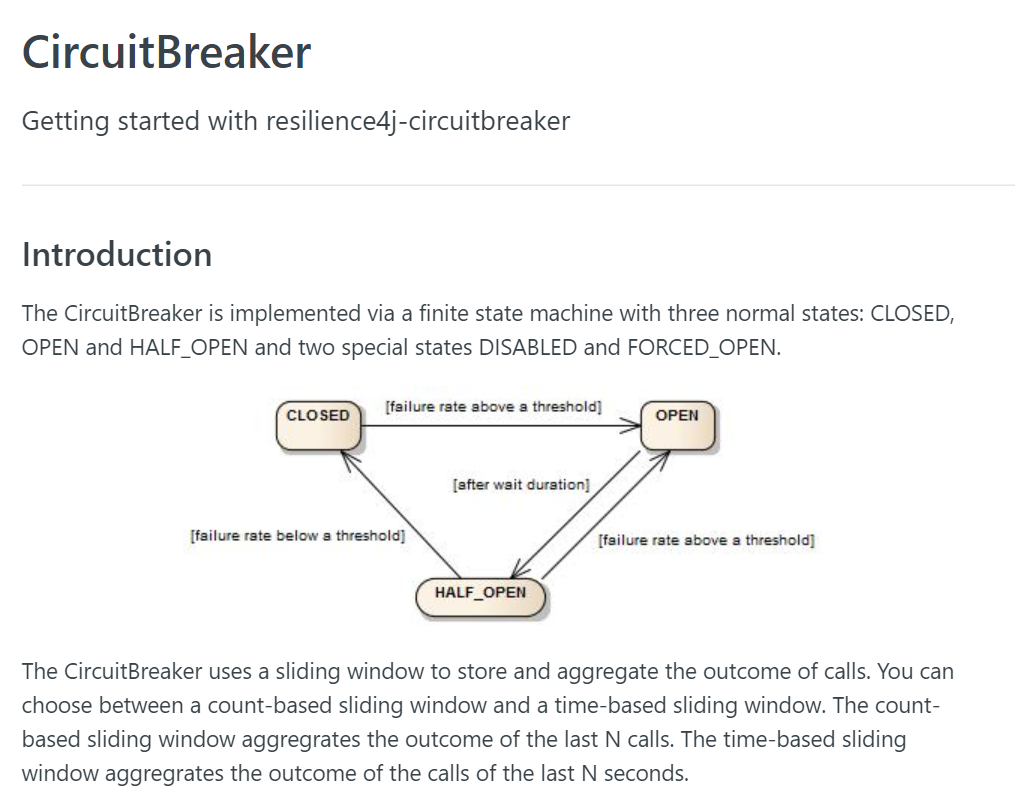
server:  
 port: 9090  
  
spring:  
 application:  
 name: API-GATEWAY  
 config:  
 import: configserver:http://localhost:9296  
  
 cloud:  
 gateway:  
 routes:  
 - id : ORDER-SERVICE  
 uri : lb://ORDER-SERVICE  
 predicates:  
 - Path=/order/\*\*  
 - id: PAYMENT-SERVICE  
 uri: lb://PAYMENT-SERVICE  
 predicates:  
 - Path=/payment/\*\*  
 - id: PRODUCT-SERVICE  
 uri: lb://PRODUCT-SERVICE  
 predicates:  
 - Path=/product/\*\*

**This is all about API Gateway**

**: CIRCUIT BREAKER :**

What is circuit breaker in Spring Boot microservices?

Spring Cloud's Circuit Breaker library provides an implementation of the Circuit Breaker pattern: when we wrap a method call in a circuit breaker, Spring Cloud Circuit Breaker watches for failing calls to that method, and if failures build up to a threshold, Spring Cloud Circuit Breaker opens the circuit so that ...



1. **Add the Resilience4J circuit breaker dependency to the cloud API gateway service.**