Bipin Kumar

shael.dhn88@gmail.com

Eureka

Eureka setup docs

Eureka server is used as Discovery server.

Step 1: Create a spring boot application using below configuration:

* Use below parent for spring boot

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.4.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

* For eureka we have added below dependency.

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

* In pom if you look we have defined no version in dependency the version dependency is handled by parent pom. For eureka we have mentioned a parent using dependencimanagement tag in which we defined parent pom and its version.
* To enable eureka we need to used @EnableEurekaServer, it will enable autoconfiguration of eureka server configuration. And in application.yml file we have defined below properties

server:

port: 5001

If you open EurekaServerAutoConfiguration.java you can find @ConditionalOnBean(EurekaServerMarkerConfiguration.Marker.**class**) It says that if any class have this marker enable auto configuration.

If you now open @EnableEurekaServer annotation you can find out below code.

@Target(ElementType.***TYPE***)

@Retention(RetentionPolicy.***RUNTIME***)

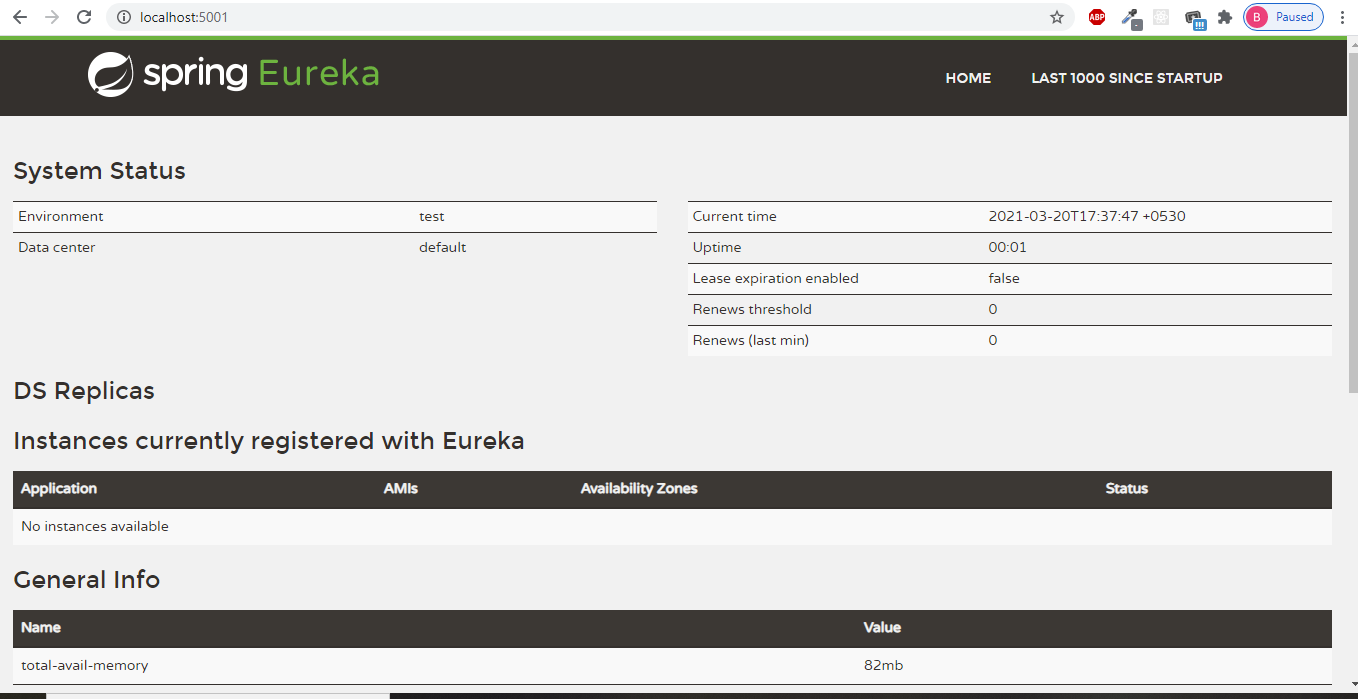
@Documented

@Import(EurekaServerMarkerConfiguration.**class**)

**public** **@interface** EnableEurekaServer {

}

If you start this project you will get exception but you eureka server will get started on port 5001.



To understand the error let assume a scenario, suppose we have S1, S2, S3... S20 microservices having n no. of instance can we manage all this instance in one box. No because if box went down all our microservices will go down. So, we are deploying our services on cloud or using kubernate.

Let’s take example of cloud like AWS, every cloud has something called as Region and each region is divided into zones, let assume we have we have three zones and each zone have some microservices deployed and all are register with one eureka, here if eureka go down then it will stop synching of registry, also face some latency issue etc.

That’s why we use eureka cluster in prod environment in which in all zones one eureka is deployed. And here each eureka synch up it’s registry from another eureka server.

So here we are getting this error because here eureka is thinking that there is another eureka and trying to fetch its registry.

Now I have to resolve this issue I need to write some property. Which is given below: Also during startup it try to get host name from machine so it is better practice to write host name using below property. ( in unix etc/hosts/

eureka:

client:

fetch-registry: false

register-with-eureka: false

instance:

hostname: localhost

**Registering microservices to Eureka server**:

1. Here for demo I have taken cargo-routing application.
2. We have only one eureka server and I want to register routing service to register.

Property of microservice to register with eureka.

1. At startup register itself to eureka server
2. At regular interval send heart beat to eureka server.
3. Suppose if eureka server is down due to crash, and it restarted, then its responsibility of eureka client to register again it to eureka server.
4. eureka also maintain local registry of each eureka in cache.
5. To register microservice to eureka we need to add below dependency in pom.xml of microservice.

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

Annotate your main class with either @EnableEurekaClient

Put name of application with which it registers to eureka

spring:

application:

name: routing-service

Now for eureka put below property:

eureka:

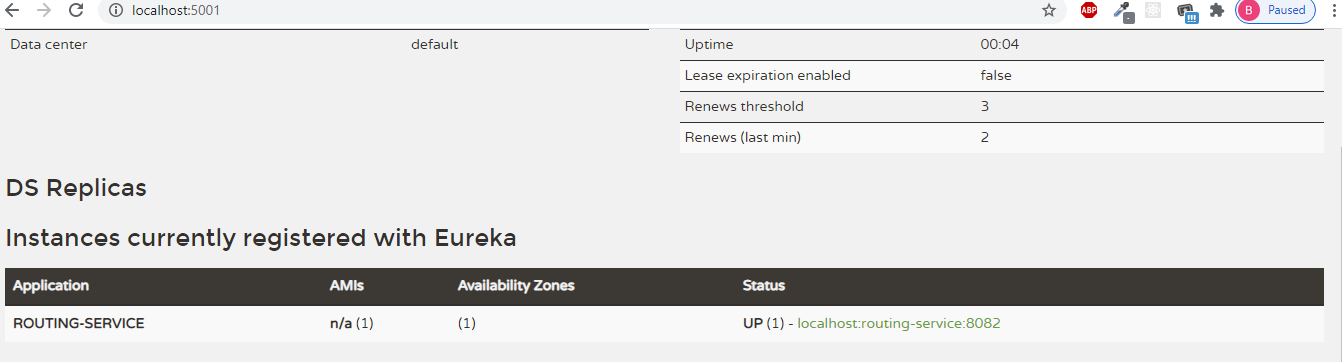
client:

service-url:

defaultZone: <http://localhost:5001/eureka>

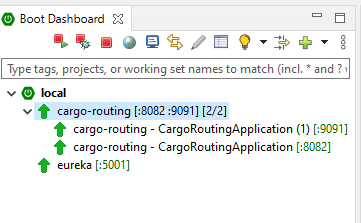
Here service-url is of type of map so you can pass multiple zone on your microservice needs to be register to eureka.

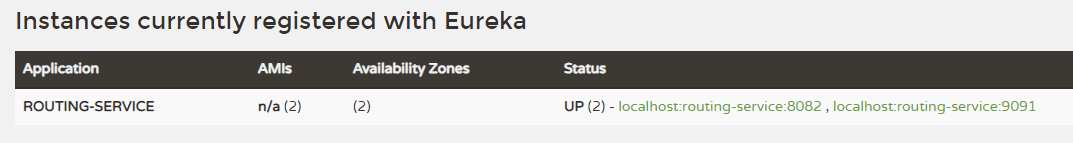
First start eureka and then routing-service, In routing service log you can see it is registering the instance to eureka server. Now if you go to eureka page and refresh you should find microservice to be registered.



Now suppose we shut down the service, so we know that till 90 sec it will keep registry but if you shutdown service and go to eureka page you can see it got registered, it is because in eureka client there is shutdown hook which make API call to deregister the service. You can see this in logs.

Now to check two instances go to eclipse, in boot dashboard right click on service and click on duplicate config, expand and right click one of configuration and in argument put –sever.port=9091 apply and close.





Suppose you don’t want to show name of service like that then you can put below properties.

eureka:

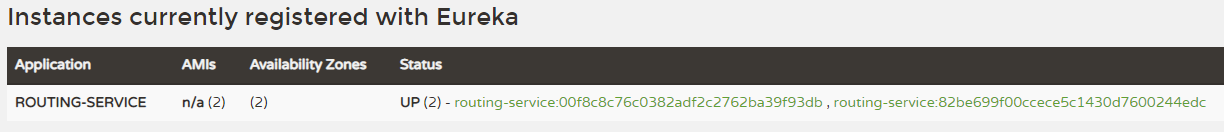
client:

service-url:

defaultZone: http://localhost:5001/eureka

instance:

instance-id: ${spring.application.name}:${random.value}



Internal API calls:

1. In eclipse go to window -> go to show view -> search for TCP/IP monitoring select and open.
2. Right click and select properties on TCP/IP view.
3. Click on Add button and enter below details in pop up.

Local monitoring port= 1234

Host name = localhost

Port = 5001 (actual port on which your eureka running

1. Select the property and click on start
2. Go to application.yml file of microservice and change the properties of eureka server url as mentioned below:

eureka:

client:

service-url:

# defaultZone: http://localhost:5001/eureka

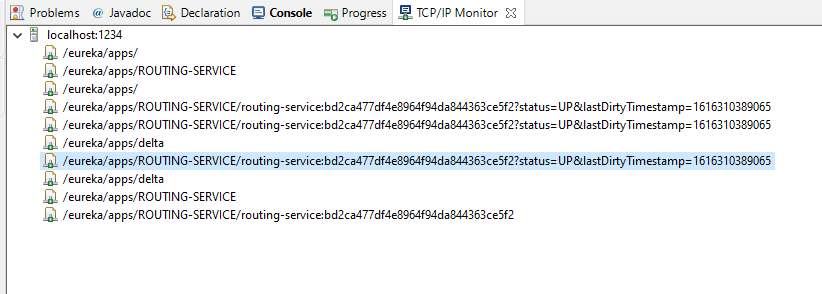
#TCP/IP monitoring

defaultZone: http://localhost:1234/eureka

# z1: http://localhost:5001/eureka

# z2: <http://localhost:5001/eureka>

1. Now start eureka server and then microservice let it run for sometime and you can see different url which is called from client as shown in below screenshot.



Here you can see API for registering service, heartbeat, delta and also when You shut down your application then un-registering API call.

Similarly using property, we can configure renewal interval and expiration time.

eureka:

client:

service-url:

defaultZone: http://localhost:5001/eureka

#TCP/IP monitoring

# defaultZone: http://localhost:1234/eureka

# z1: http://localhost:5001/eureka

# z2: http://localhost:5001/eureka

instance:

instance-id: ${spring.application.name}:${random.value}

lease-renewal-interval-in-seconds: 30

lease-expiration-duration-in-seconds: 60

We have seen one warning in eureka server page.

Self-preservation mode:

Suppose there are 500 servers registered with eureka and suddenly 15% service go down due to some reason in that case eureka automatically go to self-preservation mode and don’t mark those registries for eviction.

eureka:

server:

enable-self-preservation: **false**

Eureka clustering:

Generally, we can have eureka cluster in production environment in which each zone instance of microservice will register to respective zone eureka and each eureka call other eureka to fetch its registry. To simulate it in windows, we need to go C:\Windows\System32\drivers\etc open hosts file as admin and add below lines.

127.0.0.1 eureka-primary

127.0.0.1 eureka-secondary

127.0.0.1 eureka-ternary

Now add properties in eureka file with profiles and create running configuration based on profiles.

---

spring:

profiles: eurekaone

server:

port: 4002

eureka:

instance:

hostname: eureka-primary

client:

fetch-registry: true

register-with-eureka: false

serviceUrl:

defaultZone: http://eureka-secondary:4003/eureka,http://eureka-ternary:4004/eureka

---

spring:

profiles: eurekatwo

server:

port: 4003

eureka:

instance:

hostname: eureka-secondary

client:

fetch-registry: true

register-with-eureka: false

serviceUrl:

defaultZone: http://eureka-primary:4002/eureka,http://eureka-ternary:4004/eureka

---

spring:

profiles: eurekatwo

server:

port: 4004

eureka:

instance:

hostname: eureka-ternary

client:

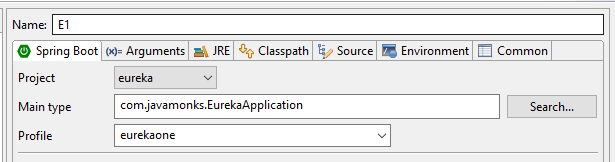
fetch-registry: true

register-with-eureka: false

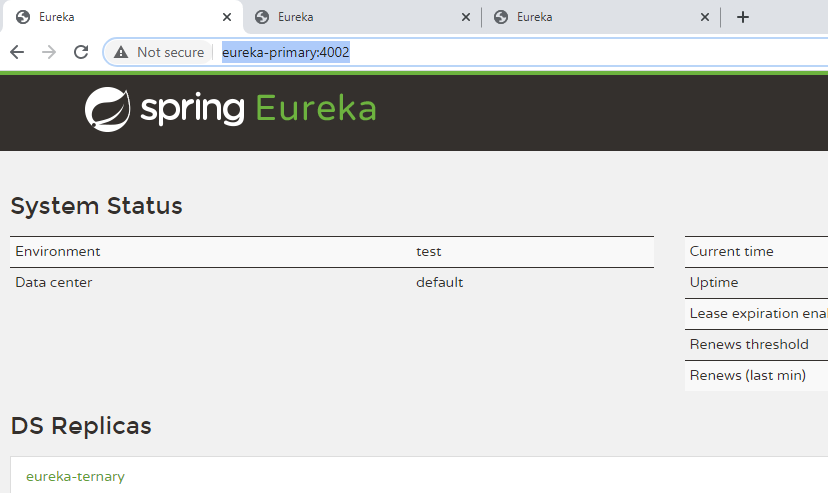
serviceUrl:

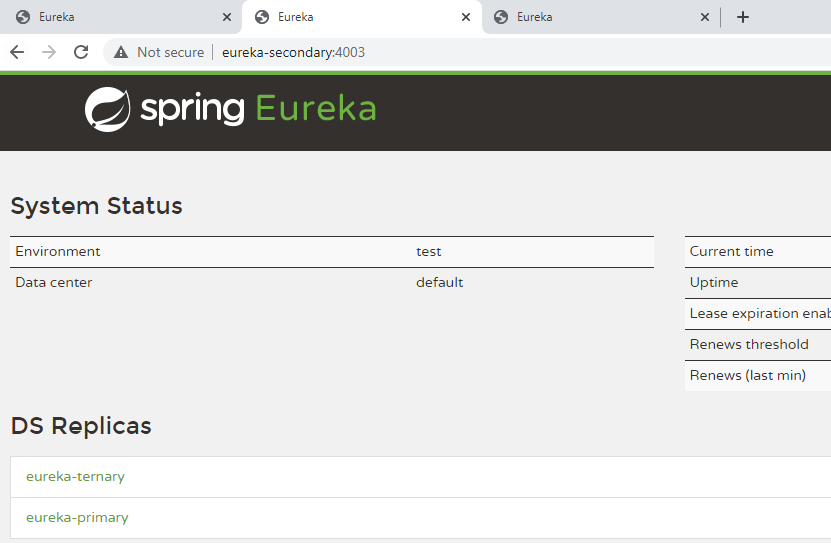
defaultZone: http://eureka-primary:4002/eureka,http://eureka-secondary:4003/eureka

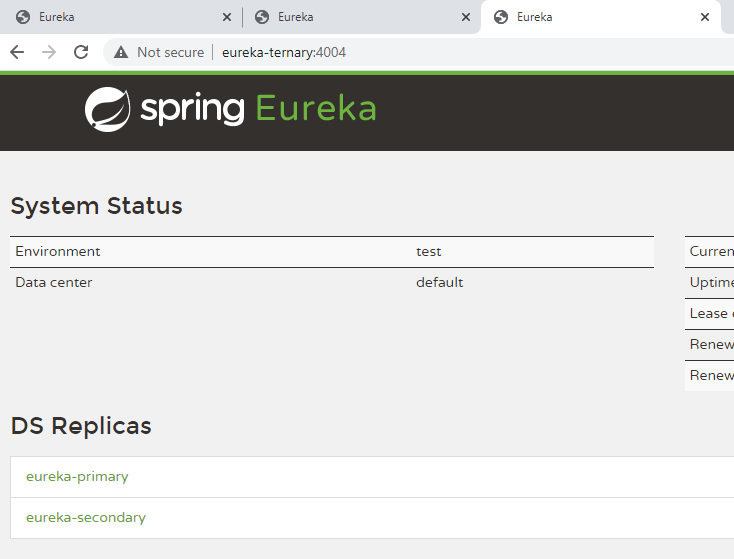
Running configuration based on profiles



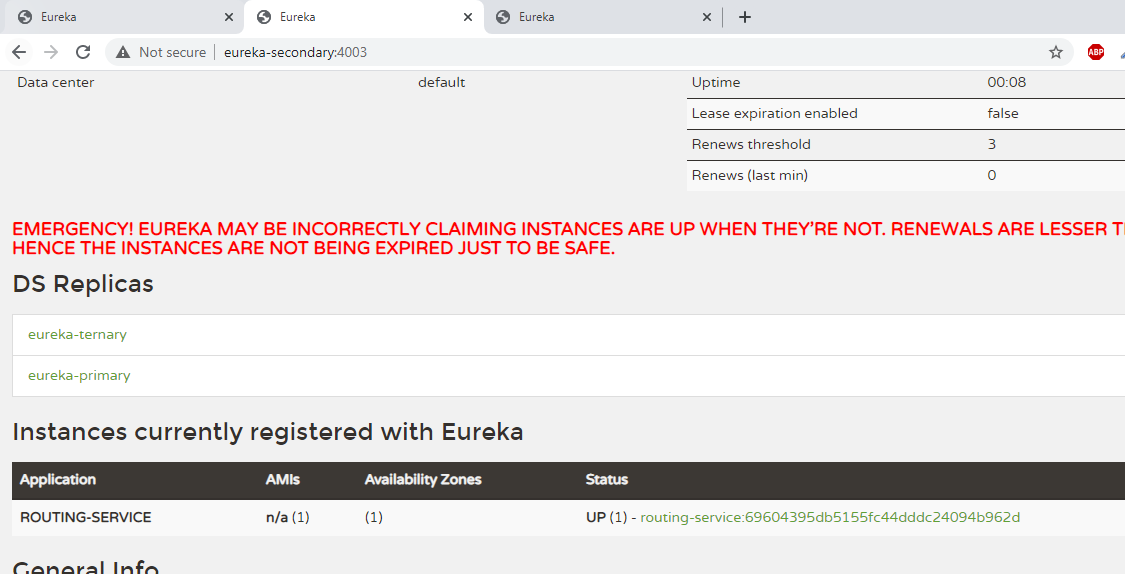
Now start all the eureka servers you may get errors at startup because during startup it try to fetch registry from another eureka which yet to be started.







Now let’s start microservice which register one of the eureka server and if you refresh page you should see registry available with each eureka server.



In Cargo-booking app do the same configuration as routing app. Now open ExternalCargoRoutingService.java class and do below changes.

**import** org.springframework.cloud.client.discovery.DiscoveryClient;

@Autowired

**private** DiscoveryClient discoveryClient;

here we have autowired discoveryclient because suppose in future you have changed your discovery server then you don’t need to change code only you need to change your property in yml files.

Similarly, in main class use @EnableDiscoveryClient instead of @EnableEurekaClient as it will be generic one.

Now start your application and open swagger html and use it see how code executed. You can see in console every time same URL got printed this because we currently didn’t configure client side load balancing.

{

"bookingAmount": 100,

"originLocation": "BLR",

"destLocation" : "DEL",

"destArrivalDeadline" : "2020-01-28"

}