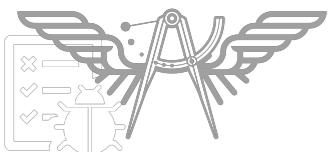


Microservices workshop





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...
Timeline About Friends 3,138 Photos More

When did you work at Opendream?
... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro
Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

@somkiat.cc

Home

Posts

Videos

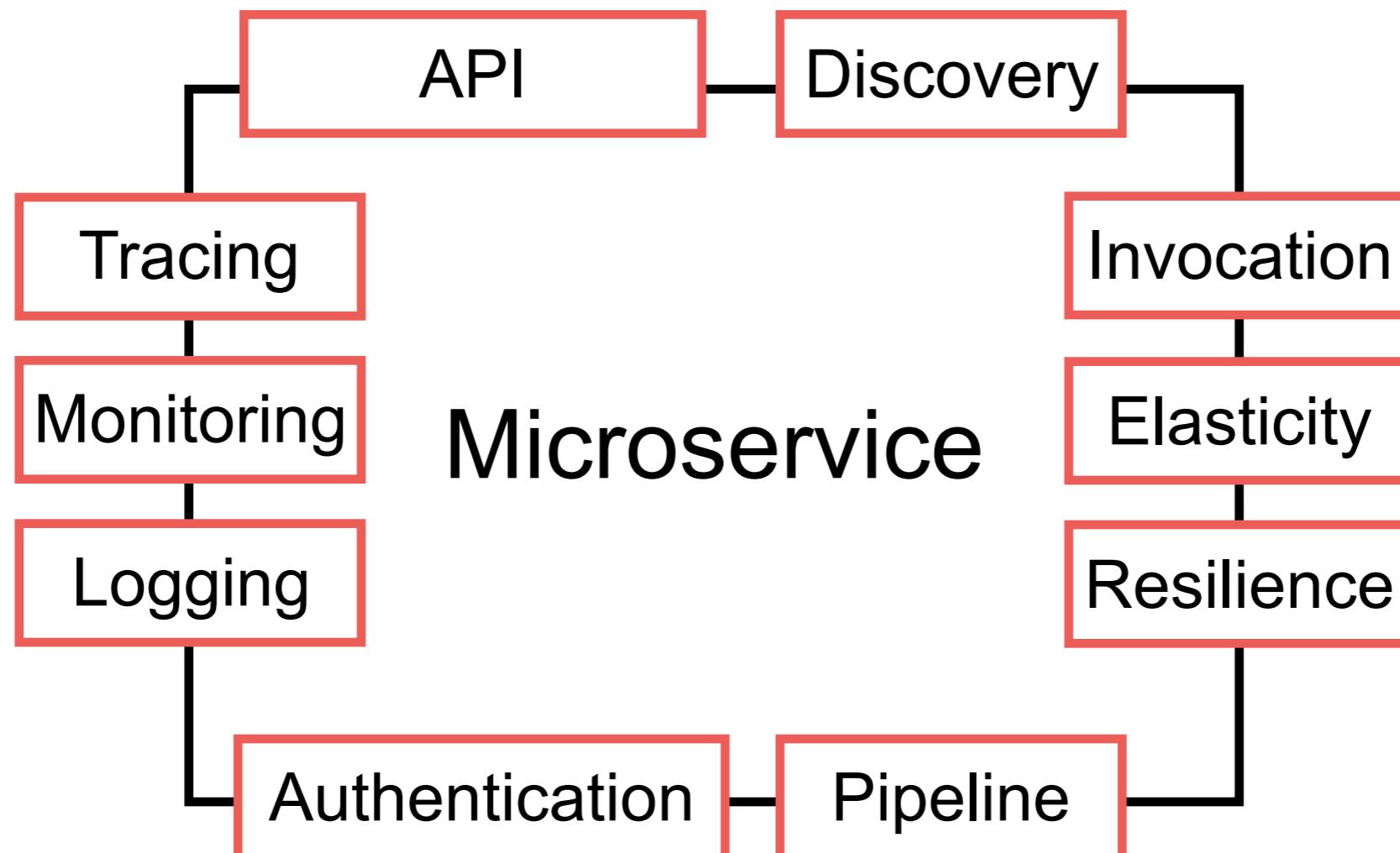
Photos



Workshop :: Develop service



Properties of Microservice



<https://github.com/up1/workshop-microservice-with-java>



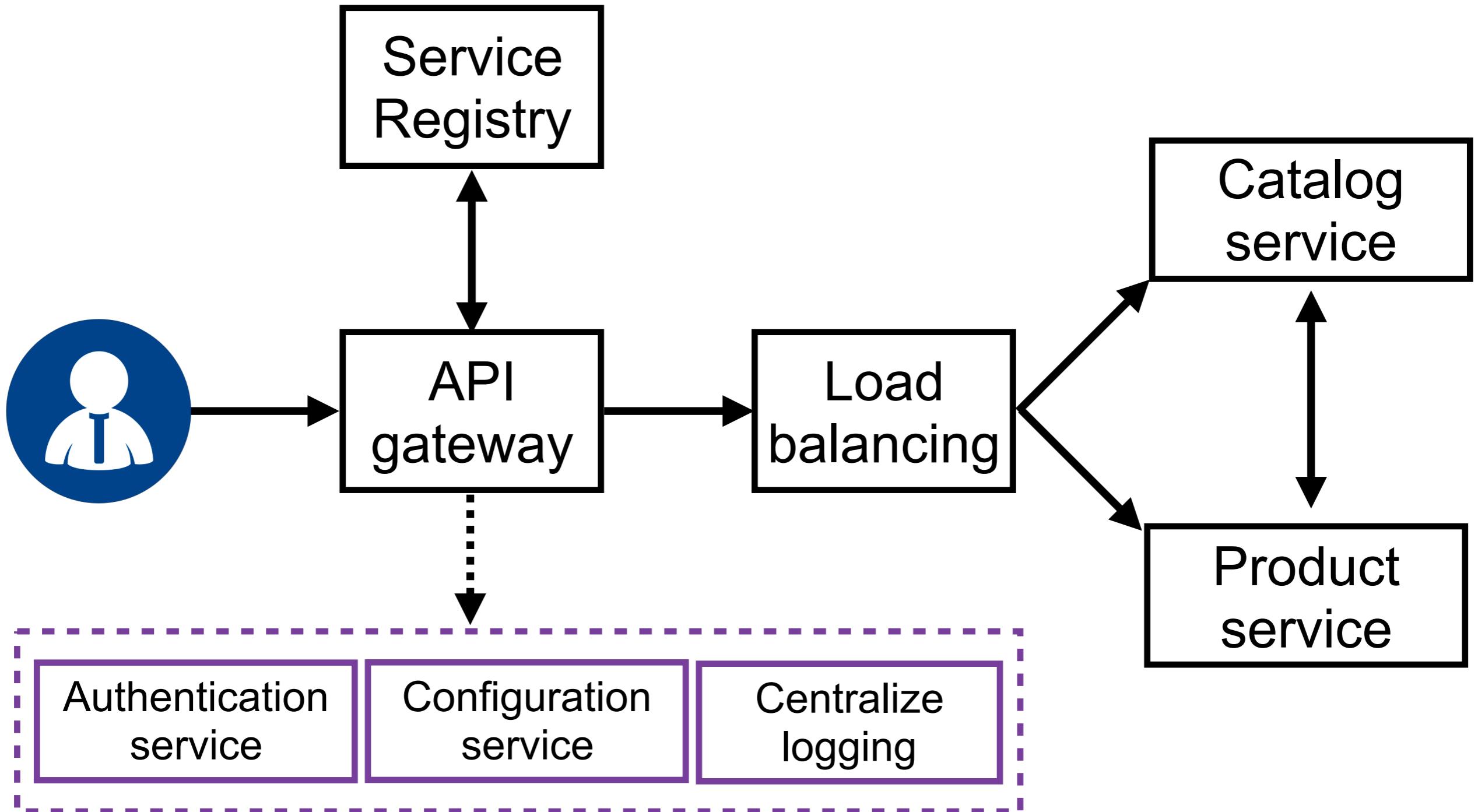
Communication between services



Synchronous communication



Goals

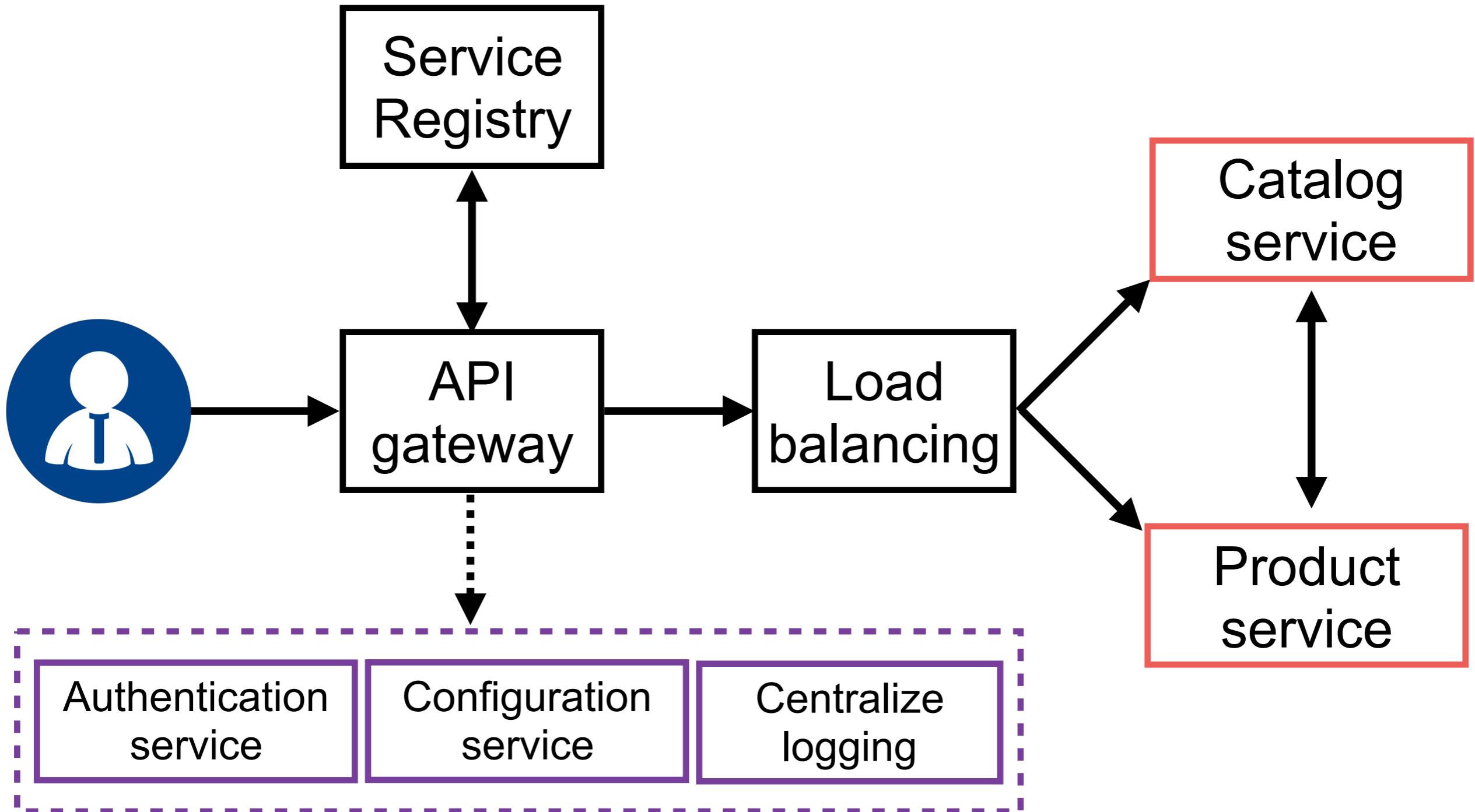


Step 1

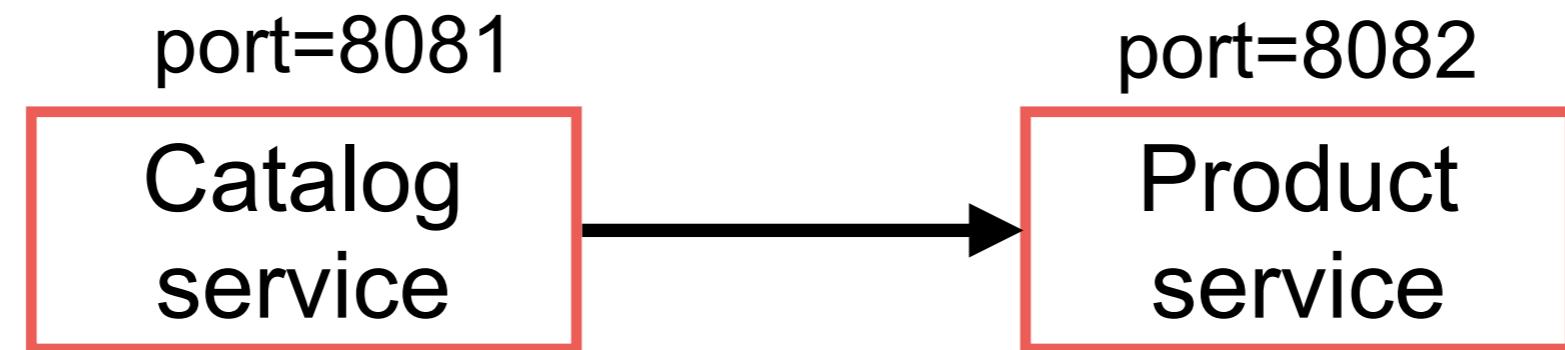
Point-to-Point communication



Goals



Point-to-Point communication



Point-to-Point communication

```
$git clone https://github.com/up1/workshop-microservice-with-java.git  
-b step01
```



Product service

List all products

Run on port 8082

\$mvnw clean package

\$java -jar target/product.jar



Product service

GET /products

```
← → ⌂ ⓘ localhost:8082/products
[
  - {
    id: 1,
    name: "Product 1",
    description: "Description 1",
    imageUrl: "URL of product 1"
  },
  - {
    id: 2,
    name: "Product 2",
    description: "Description 2",
    imageUrl: "URL of product 2"
  },
  - {
    id: 3,
    name: "Product 3",
    description: "Description 3",
    imageUrl: "URL of product 3"
  }
]
```



Catalog service

List of product by category
Run on port 8081



\$mvnw clean package
\$java -jar target/product.jar



Catalog service

GET /catalog/1

```
← → ⌂ ⓘ localhost:8081/catalog/1
{
  id: 1,
  - products: [
    - {
      id: 1,
      name: "Product 1",
      description: "Description 1",
      imageUrl: "URL of product 1"
    },
    - {
      id: 2,
      name: "Product 2",
      description: "Description 2",
      imageUrl: "URL of product 2"
    },
    - {
      id: 3,
      name: "Product 3",
      description: "Description 3",
      imageUrl: "URL of product 3"
    }
  ]
}
```



Catalog service

Open file CatalogController.java

```
@GetMapping("/catalog/{id}")
public Catalog getCatalog(@PathVariable int id) {
    Catalog catalog = new Catalog();
    catalog.setId(id);

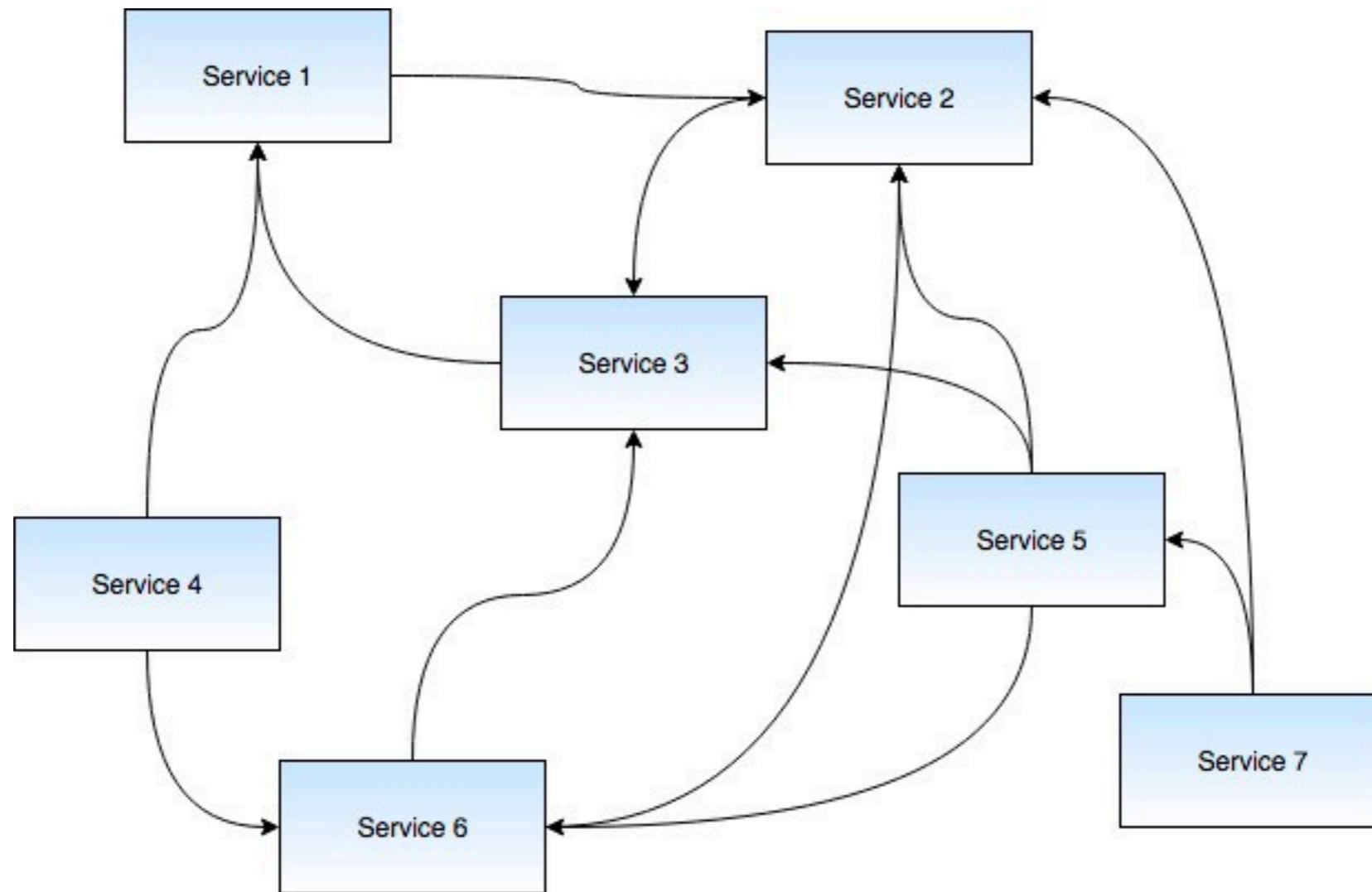
    // Get all products from product service
    List<Object> products
        = restTemplate.getForObject("http://localhost:8082/products",
                                    List.class);
    catalog.setProducts(products);

    return catalog;
}
```

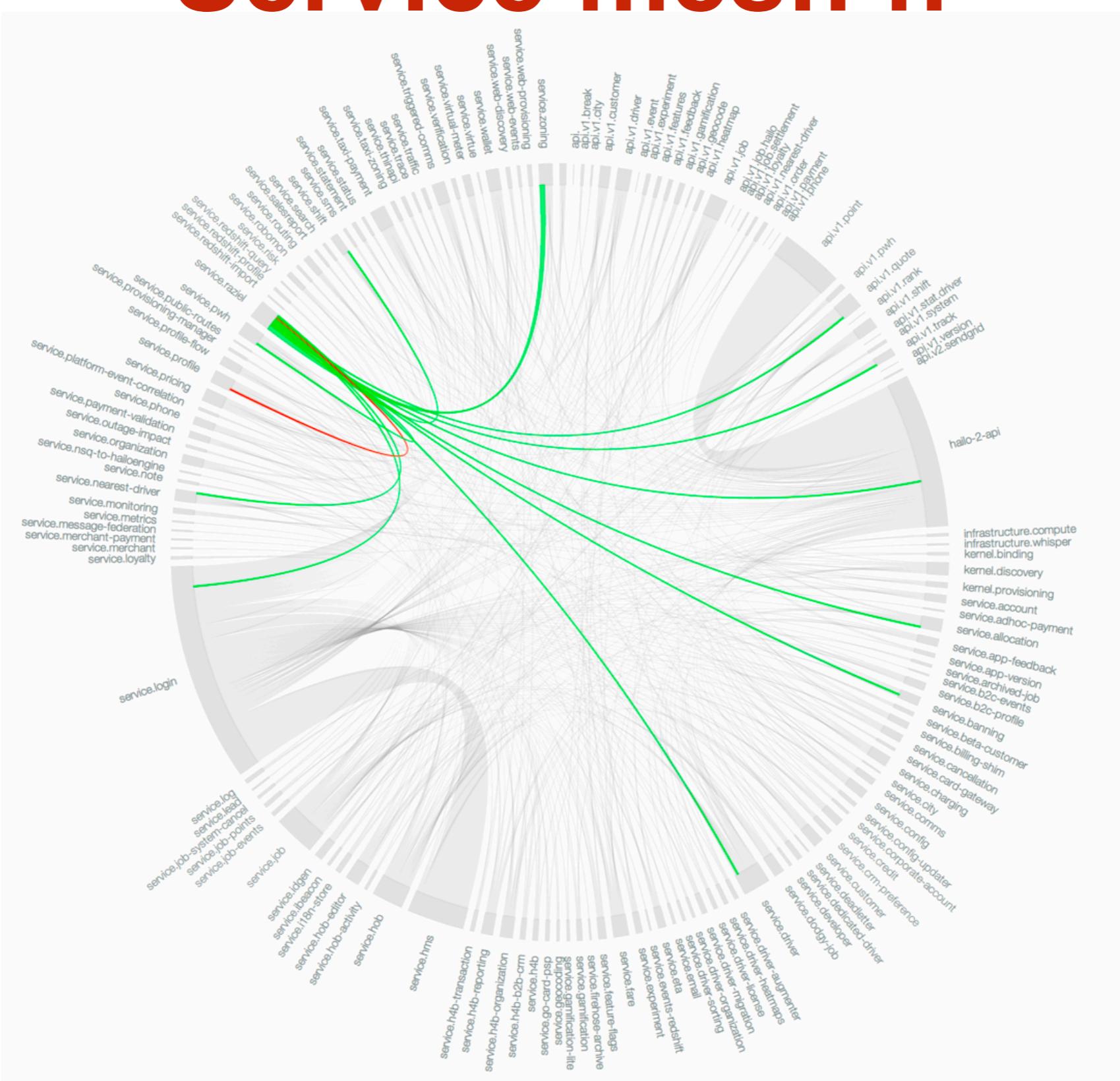
Hard code url of service !!
Change url !!
Many services !!



Many services !!



Service mesh !!

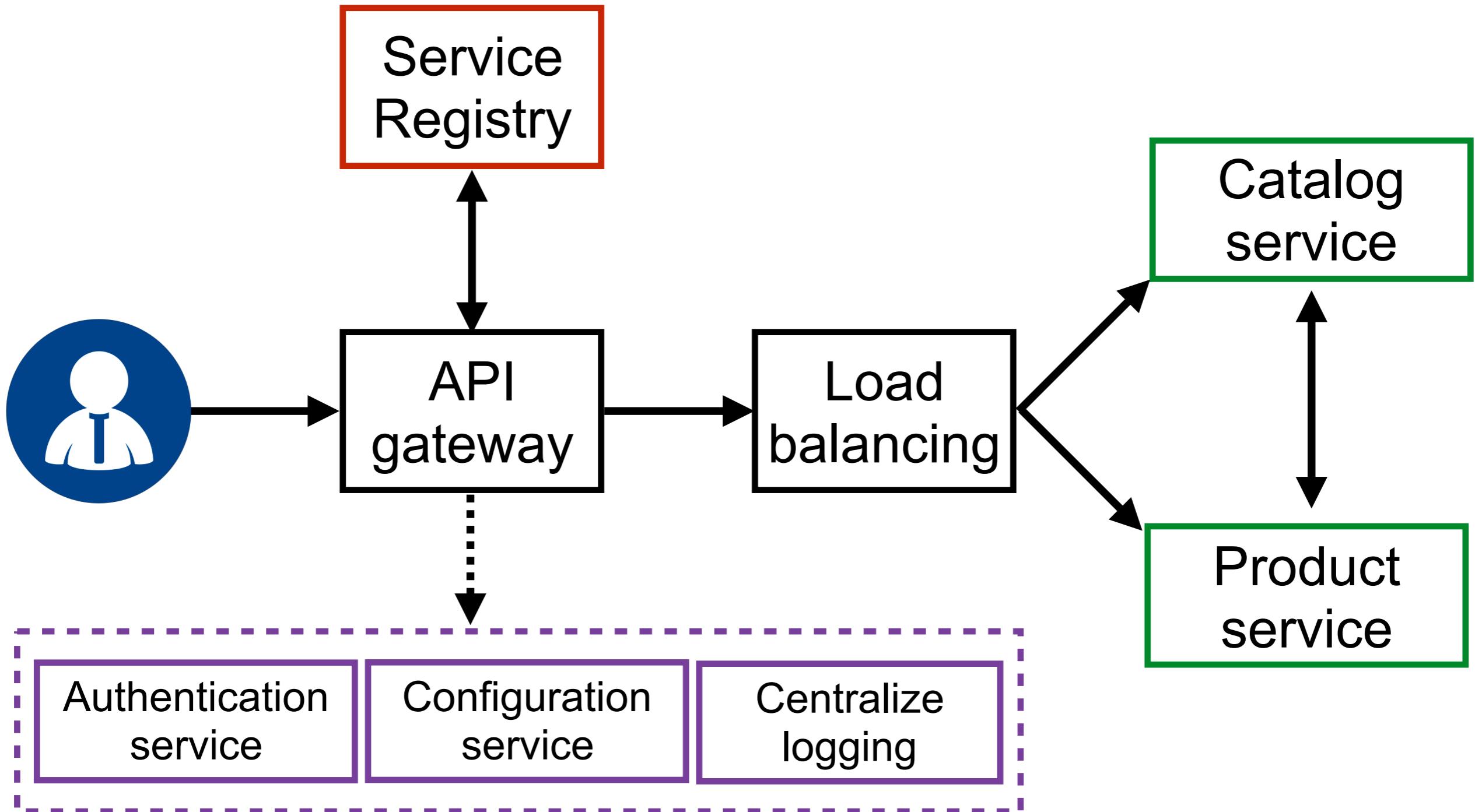


Step 2

Using service registry



Goals



Service Registry

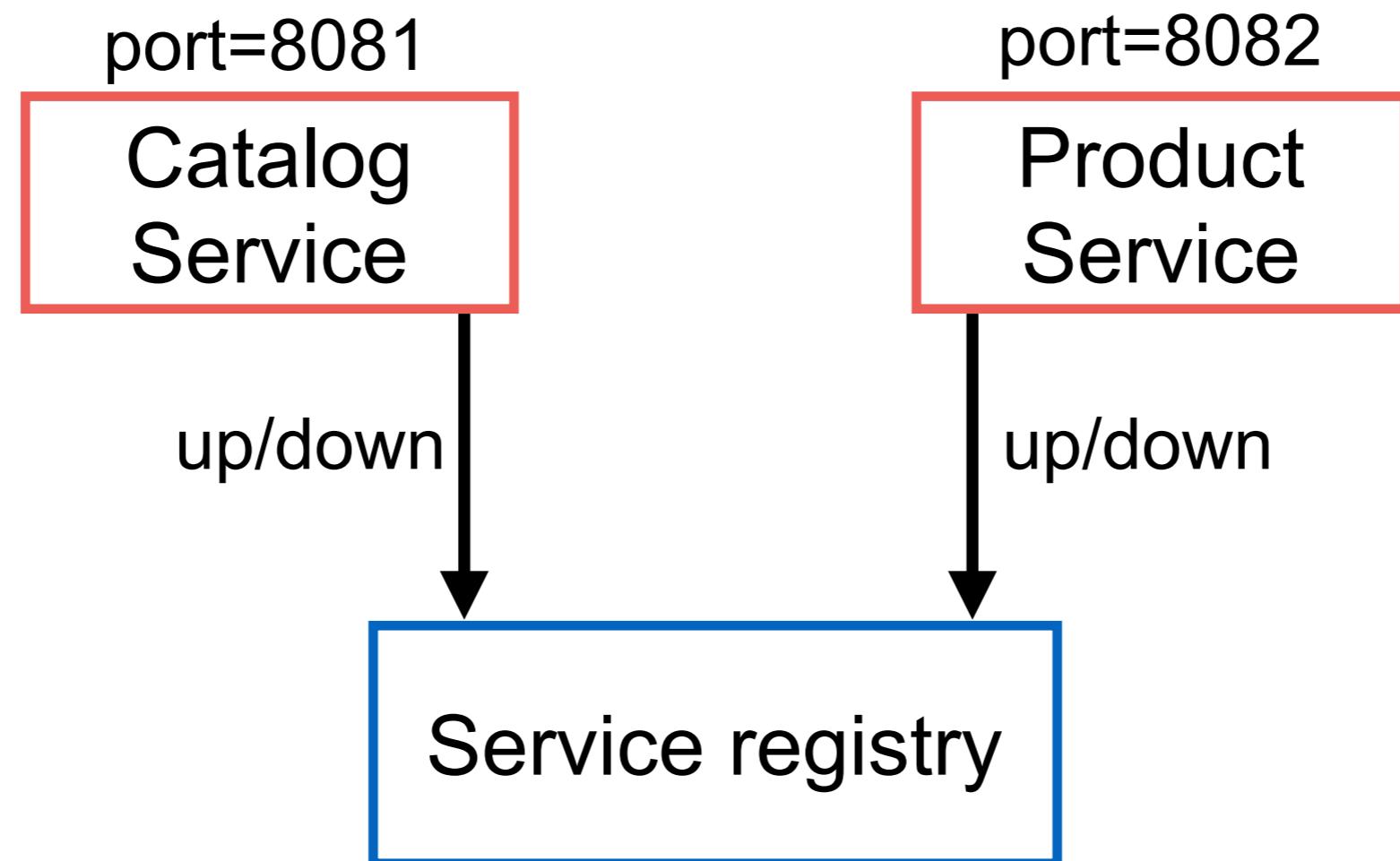
Database to keep data of services

How to register services ?

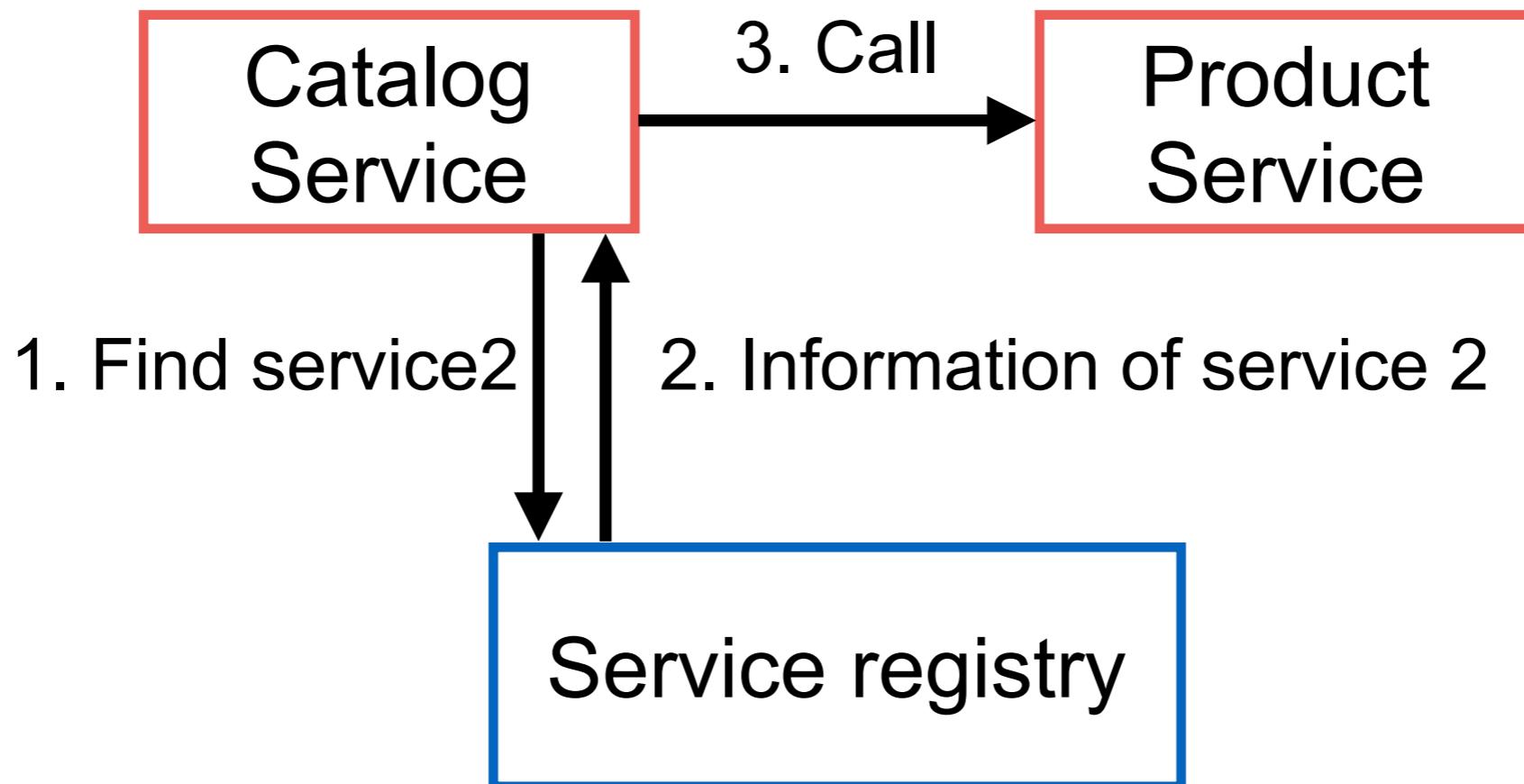
How to discover services ?



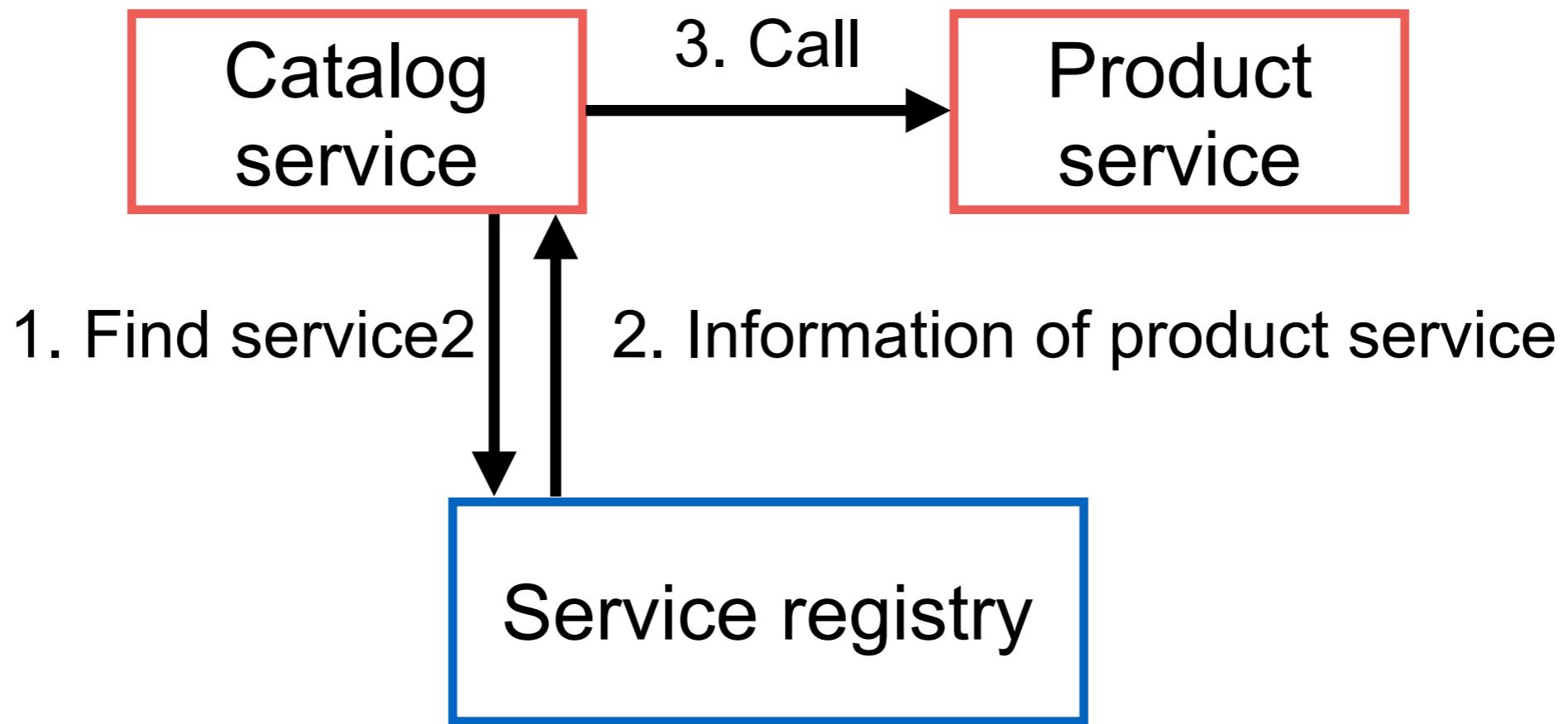
1. Service register



2. Service Discovery



Our project



NETFLIX
OSS
Eureka

HashiCorp
Consul

etcd



Service Registry

```
$git clone https://github.com/up1/workshop-microservice-with-java.git  
-b step02
```



Step

1. Create service registry with Eureka
2. Register product service to registry
3. Register catalog service to registry
4. Catalog service call product service



1. Create service registry

Project name = eureka-server

Default port = 8761

\$mvnw clean package

\$java -jar target/eureka-server.jar



Eureka server

<http://localhost:8761>



HOME

LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2019-04-02T00:10:11 +0700
Data center	default	Uptime	00:01
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value



Microservices

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

2. Register product service

Add Eureka client library to service
See detail in file **pom.xml**

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-browser</artifactId>
</dependency>
```



2. Register product service

Add url of Eureka server
See detail in file **application.properties**

```
spring.application.name=product-service
server.port=8082
```

```
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```



2. Register product service

Add Eureka client to service
See detail in file **ProductApplication.java**

```
@SpringBootApplication
@EnableEurekaClient
public class ProductApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProductApplication.class, args);
    }

}
```



2. Register product service

Build and run service

```
$mvnw clean package  
$java -jar target/product.jar
```

```
ication  : Started ProductApplication in 6.429 seconds (JVM running for 6.878)  
2019-04-02 00:20:08.084  INFO [product-service,,,] 28662 --- [nfoReplicator-0] com.netflix.discovery.Disco  
Client    : DiscoveryClient_PRODUCT-SERVICE/192.168.1.107:product-service:8082 - registration status: 204
```



Eureka server

<http://localhost:8761>

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:product-service:8082



3. Register catalog service

Add Eureka client library to service
See detail in file **pom.xml**

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-rest-hal-browser</artifactId>
</dependency>
```



3. Register catalog service

Add url of Eureka server
See detail in file **application.properties**

```
spring.application.name=catalog-service
server.port=8081
```

```
eureka.client.service-url.default-zone=http://localhost:8761/eureka
```



3. Register catalog service

Add Eureka client to service
See detail in file **CatalogApplication.java**

```
@SpringBootApplication
@EnableEurekaClient
public class CatalogApplication {

    public static void main(String[] args)
}

}
```



3. Register catalog service

Build and run service

```
$mvnw clean package  
$java -jar target/catalog.jar
```



Eureka server

<http://localhost:8761>

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CATALOG-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:catalog-service:8081
PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:product-service:8082



4. Call product service

Call product service from catalog service
See detail in file **CatalogController.java**

```
@GetMapping("/catalog/{id}")
public Catalog getCatalog(@PathVariable int id) {
    Catalog catalog = new Catalog();
    catalog.setId(id);
      

    // Get all products from product service
    List<Object> products
        = restTemplate.getForObject("http://product-service/products",
                                    List.class);
    catalog.setProducts(products);
      

    return catalog;
}
```

Using service name
from Service Registry



Catalog service

GET /catalog/1

```
← → ⌂ ⓘ localhost:8081/catalog/1
{
  id: 1,
  - products: [
    - {
      id: 1,
      name: "Product 1",
      description: "Description 1",
      imageUrl: "URL of product 1"
    },
    - {
      id: 2,
      name: "Product 2",
      description: "Description 2",
      imageUrl: "URL of product 2"
    },
    - {
      id: 3,
      name: "Product 3",
      description: "Description 3",
      imageUrl: "URL of product 3"
    }
  ]
}
```

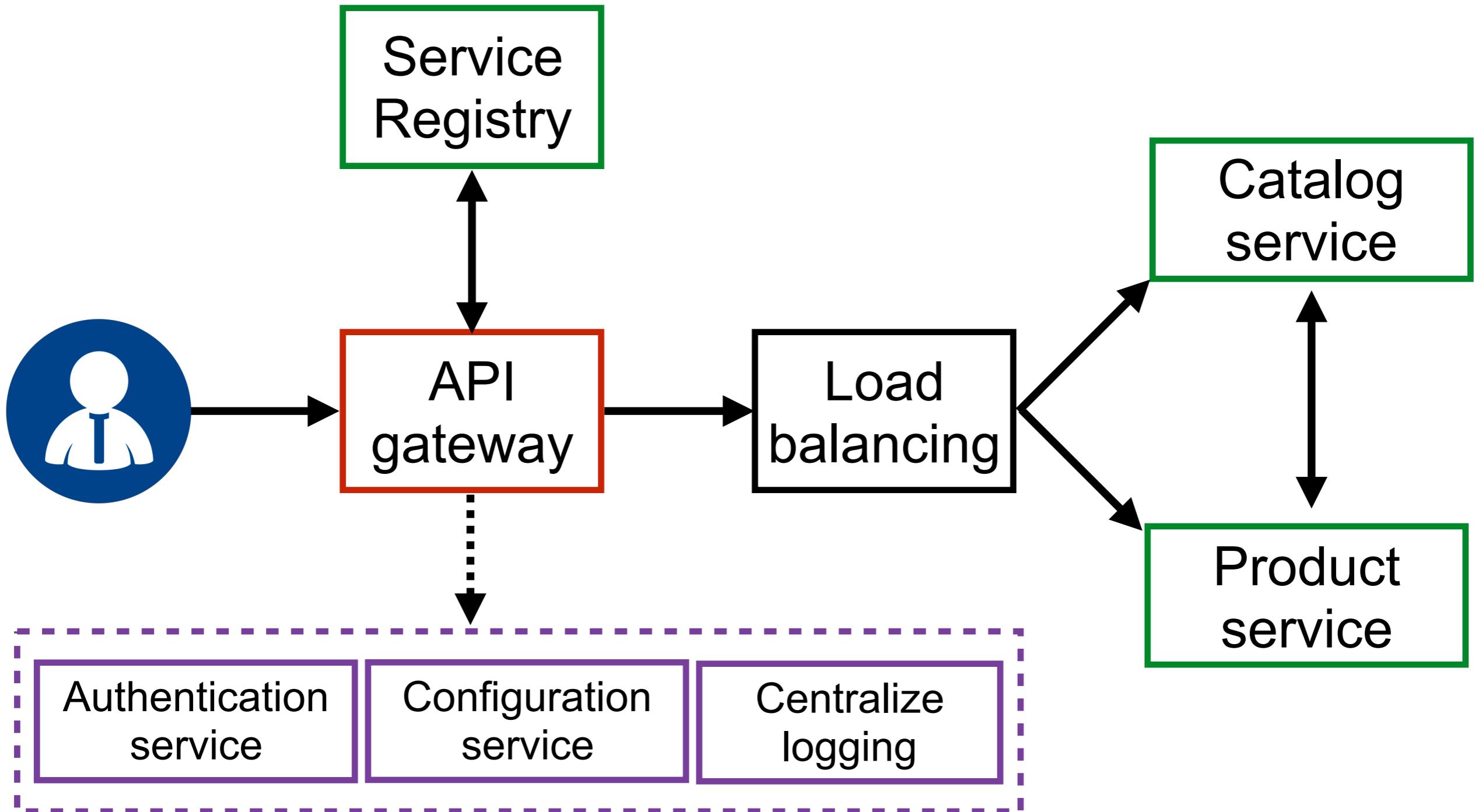


Step 3

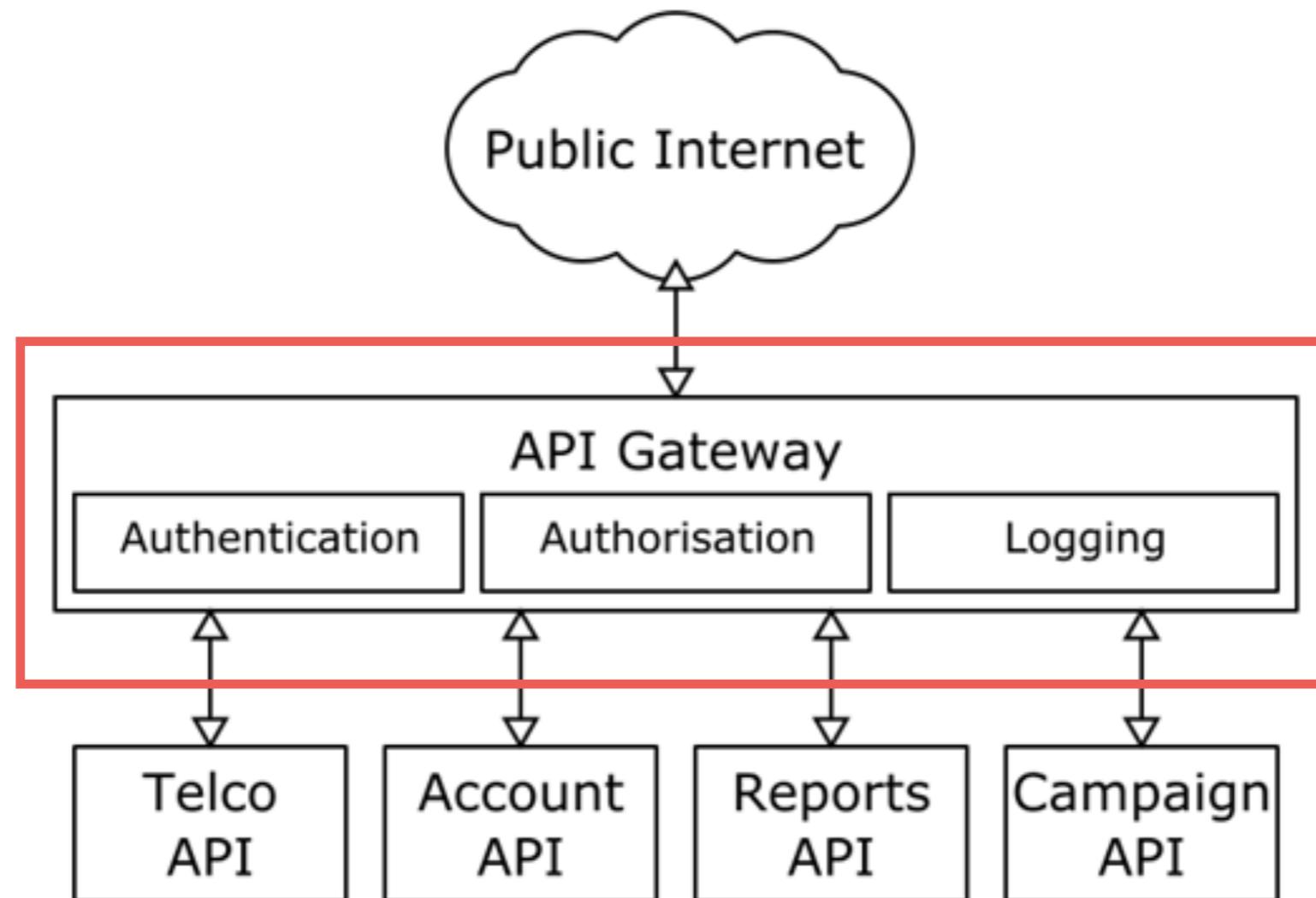
API gateway



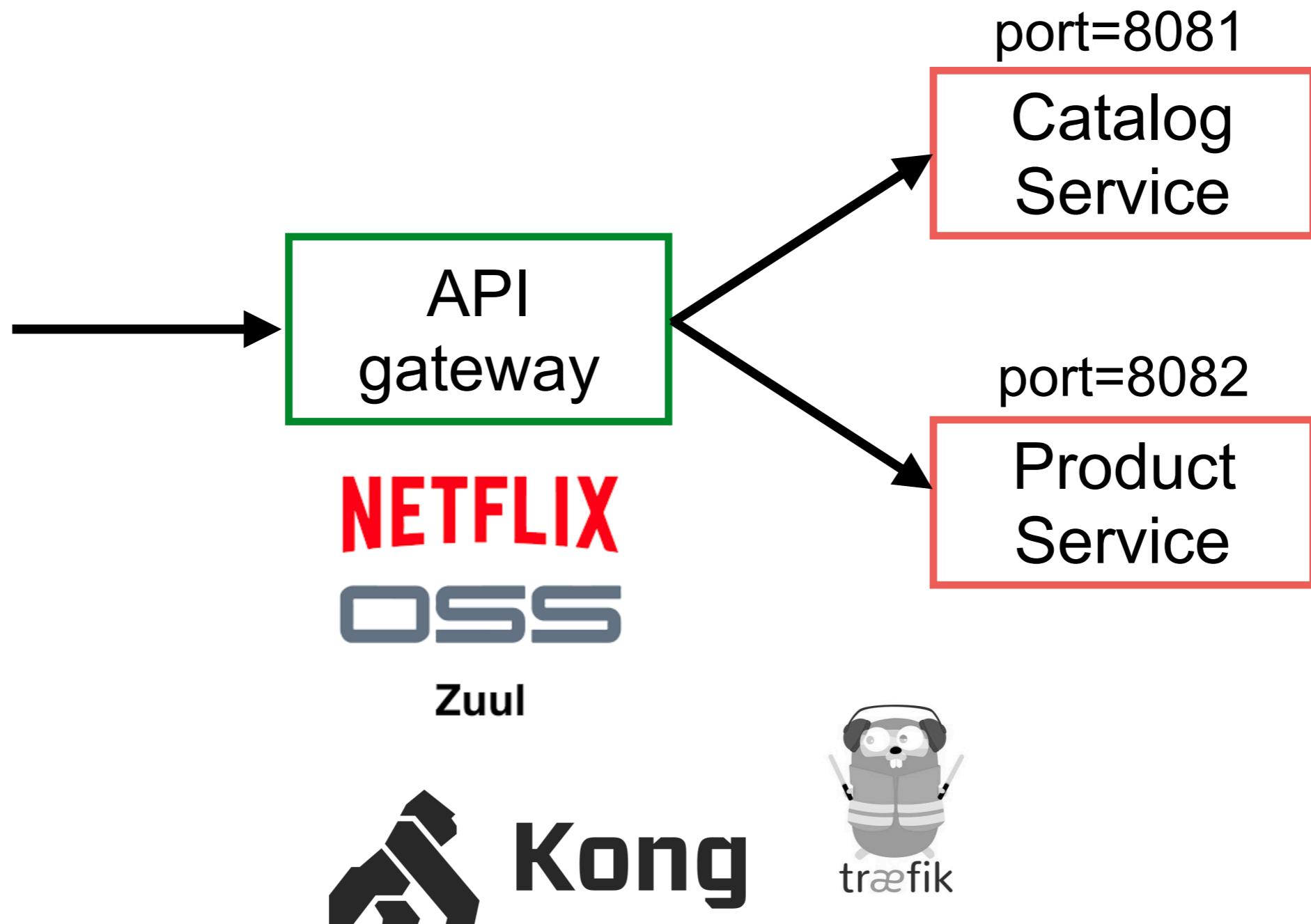
Goals



API gateway



API gateway



Service Registry

```
$git clone https://github.com/up1/workshop-microservice-with-java.git  
-b step03
```



Step

1. Create api gateway with Zuul
2. Register api gateway to registry
3. Configuration api gateway
4. Call service from api gateway



Create api gateway with Zuul

Project name = api-gateway

Default port = 8762

\$mvnw clean package

\$java -jar target/api-gateway.jar



Create api gateway with Zuul

Add Zuul library to service
See detail in file **pom.xml**

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
</dependency>
```



Create api gateway with Zuul

Enable Zuul server

See detail in file **ApiGatewayApplication.java**

```
@SpringBootApplication
@EnableEurekaClient
@EnableZuulProxy
public class ApiGatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(ApiGatewayApplication.
    }

}
```



Eureka server

<http://localhost:8761>

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - 192.168.1.107:api-gateway:8762
CATALOG-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:catalog-service:8081
PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - 192.168.1.107:product-service:8082



Create api gateway with Zuul

Add url of Eureka server
See detail in file **application.properties**

```
spring.application.name=api-gateway
server.port=8762
eureka.client.service-url.default-zone=http://localhost:8761/eureka/

# Config of Zuul
zuul.prefix=/api
zuul.ignored-services=*

# Map paths to services
zuul.routes.catalog-service.path=/catalogservice/**

management.endpoints.web.exposure.include=*
```



Check Route of API gateway

<http://localhost:8762/actuator>

<http://localhost:8762/actuator/routes>



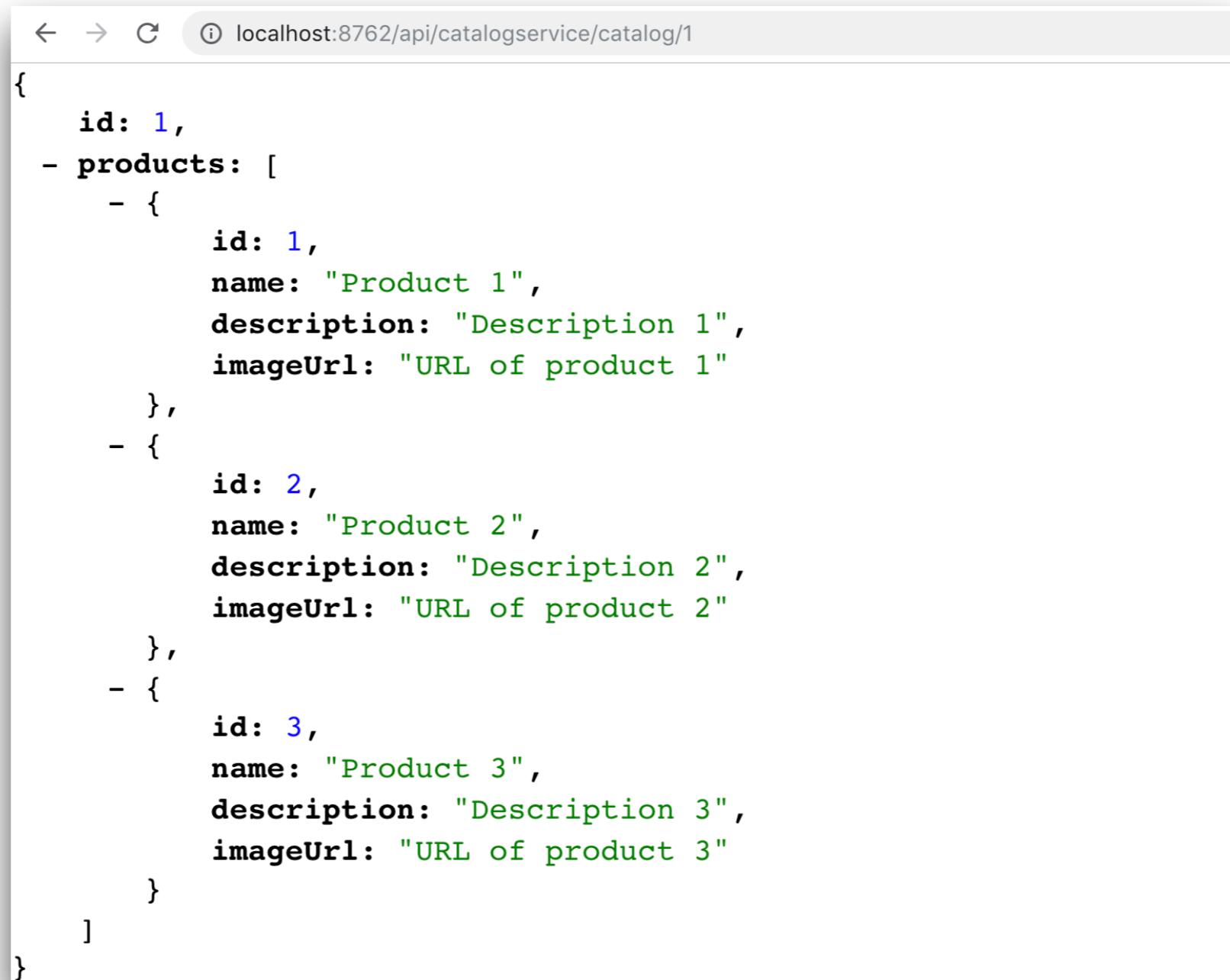
A screenshot of a web browser displaying the JSON output of the `/actuator/routes` endpoint. The URL in the address bar is `localhost:8762/actuator/routes`. The response body shows a single route entry:

```
{  
  "/api/catalogservice/**": "catalog-service"  
}
```



Call service from API gateway

<http://localhost:8762/api/catalogservice/catalog/1>

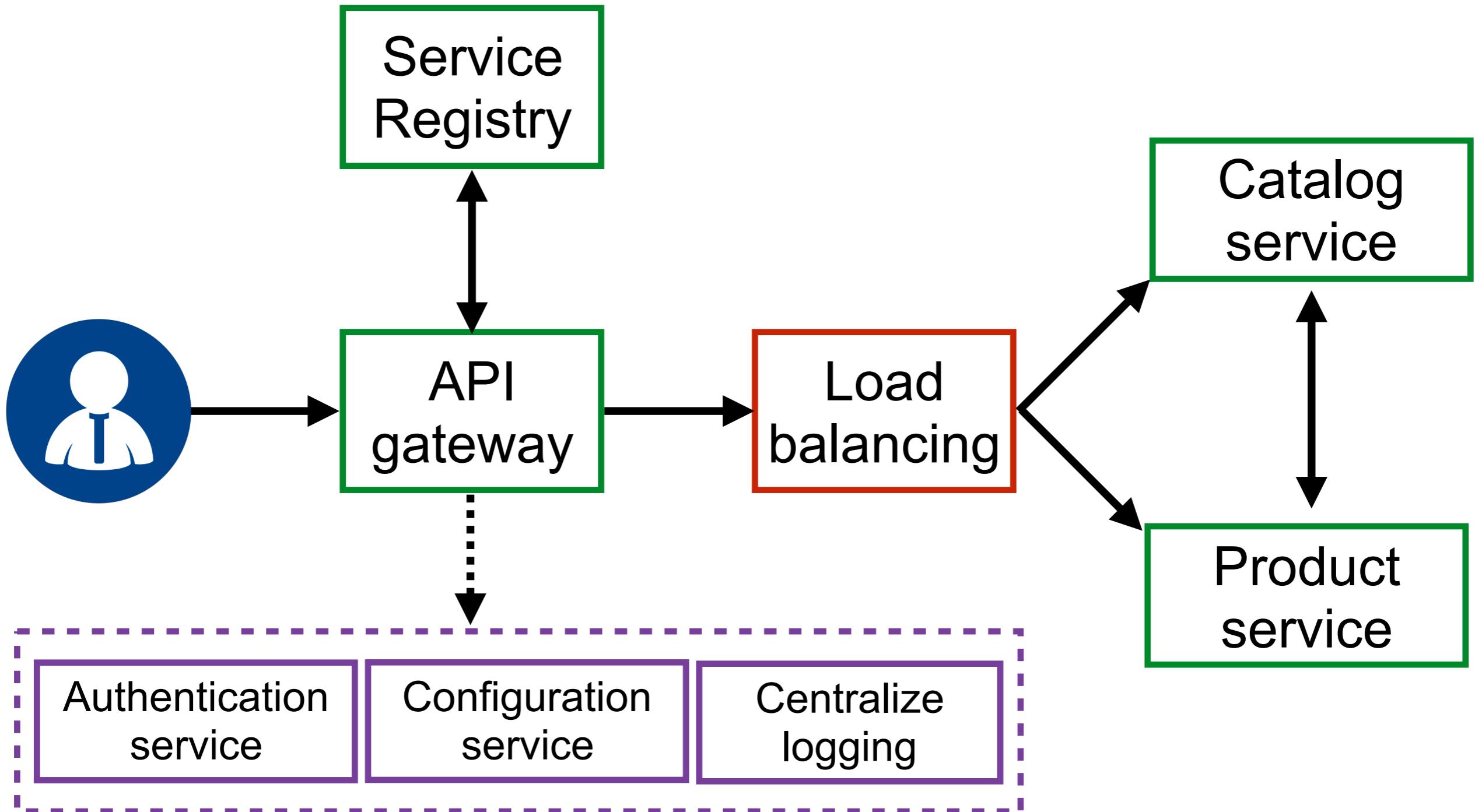


A screenshot of a web browser window displaying a JSON response. The address bar shows the URL: localhost:8762/api/catalogservice/catalog/1. The JSON object contains an 'id' field with the value 1, and a 'products' array containing three product objects. Each product has fields: id, name, description, and imageUrl.

```
{
  "id": 1,
  "products": [
    {
      "id": 1,
      "name": "Product 1",
      "description": "Description 1",
      "imageUrl": "URL of product 1"
    },
    {
      "id": 2,
      "name": "Product 2",
      "description": "Description 2",
      "imageUrl": "URL of product 2"
    },
    {
      "id": 3,
      "name": "Product 3",
      "description": "Description 3",
      "imageUrl": "URL of product 3"
    }
  ]
}
```



Goals

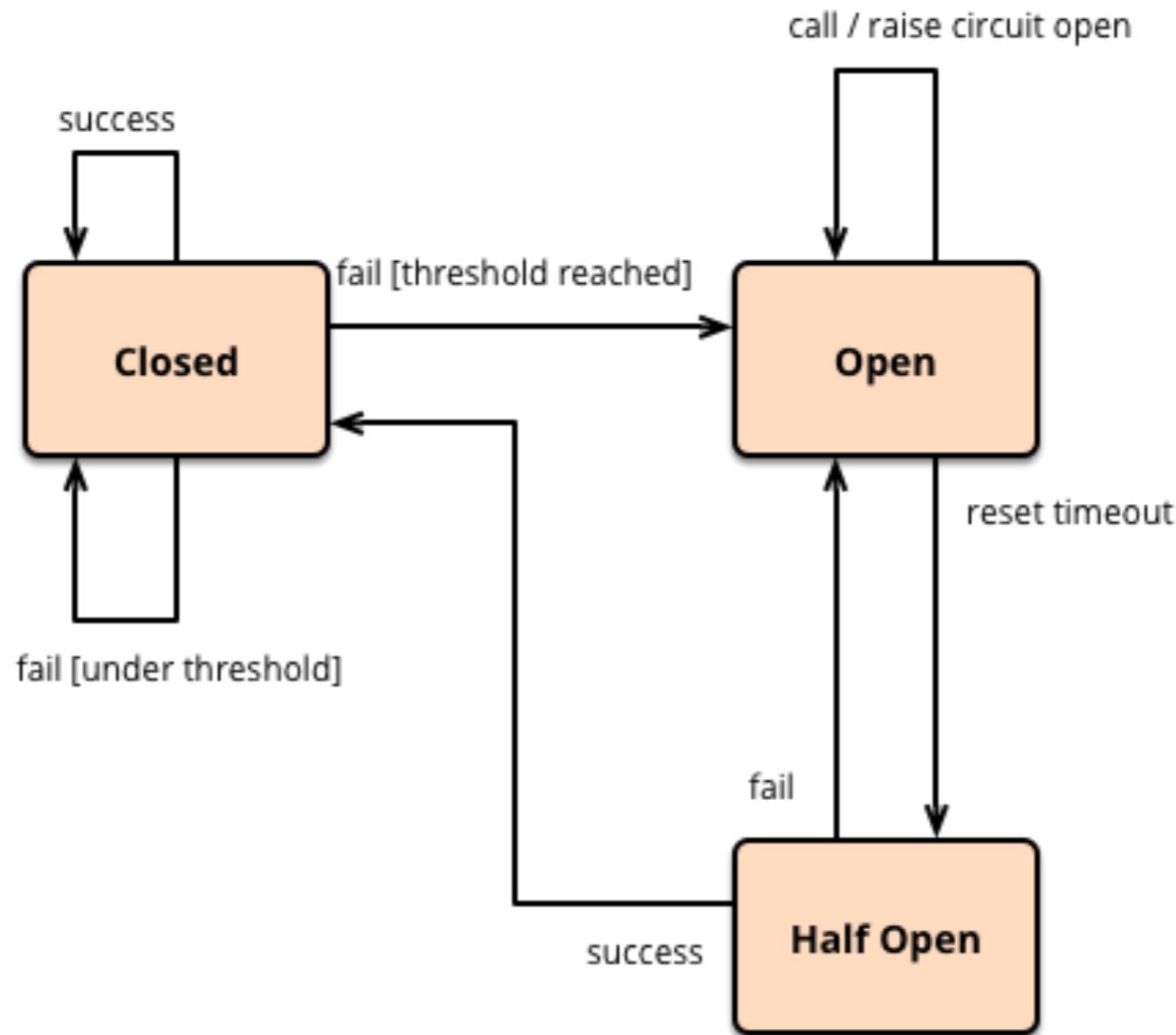


Step 4

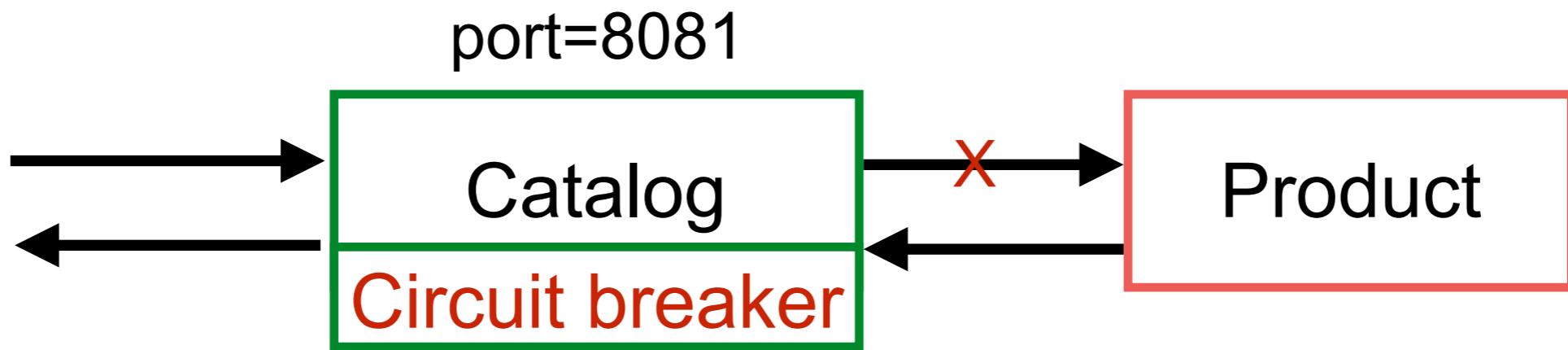
Circuite Breaker



Circuit breaker



Circuit breaker



How to handling problem ?

??



Circuit breaker

```
$git clone https://github.com/up1/workshop-microservice-with-java.git  
-b step04
```



Step

1. Enable circuit breaker in catalog service
2. Add hystrix library
3. Create fallback method for operation



Enable circuit breaker

See detail in file **CatalogApplication.java**

```
@SpringBootApplication
@EnableEurekaClient
@EnableCircuitBreaker
public class CatalogApplication {

    public static void main(String[] args) {
        SpringApplication.run(CatalogApplication.
    }

}
```



Add Hystrix library to service

See detail in file **pom.xml**

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-hystrix</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
```



Create fallback method with Hystrix

Try to handle error with default value
See detail in file **CatalogController.java**

```
@HystrixCommand(fallbackMethod = "fallback")
@GetMapping("/catalog/{id}")
public Catalog getCatalog(@PathVariable int id)
{
```

```
}
```

```
    ...
}

public Catalog fallback(int catalogId, Throwable hystrixCommand) {
    Catalog catalog = new Catalog();
    catalog.setId(catalogId);
    return catalog;
}
```



Step 5

??

