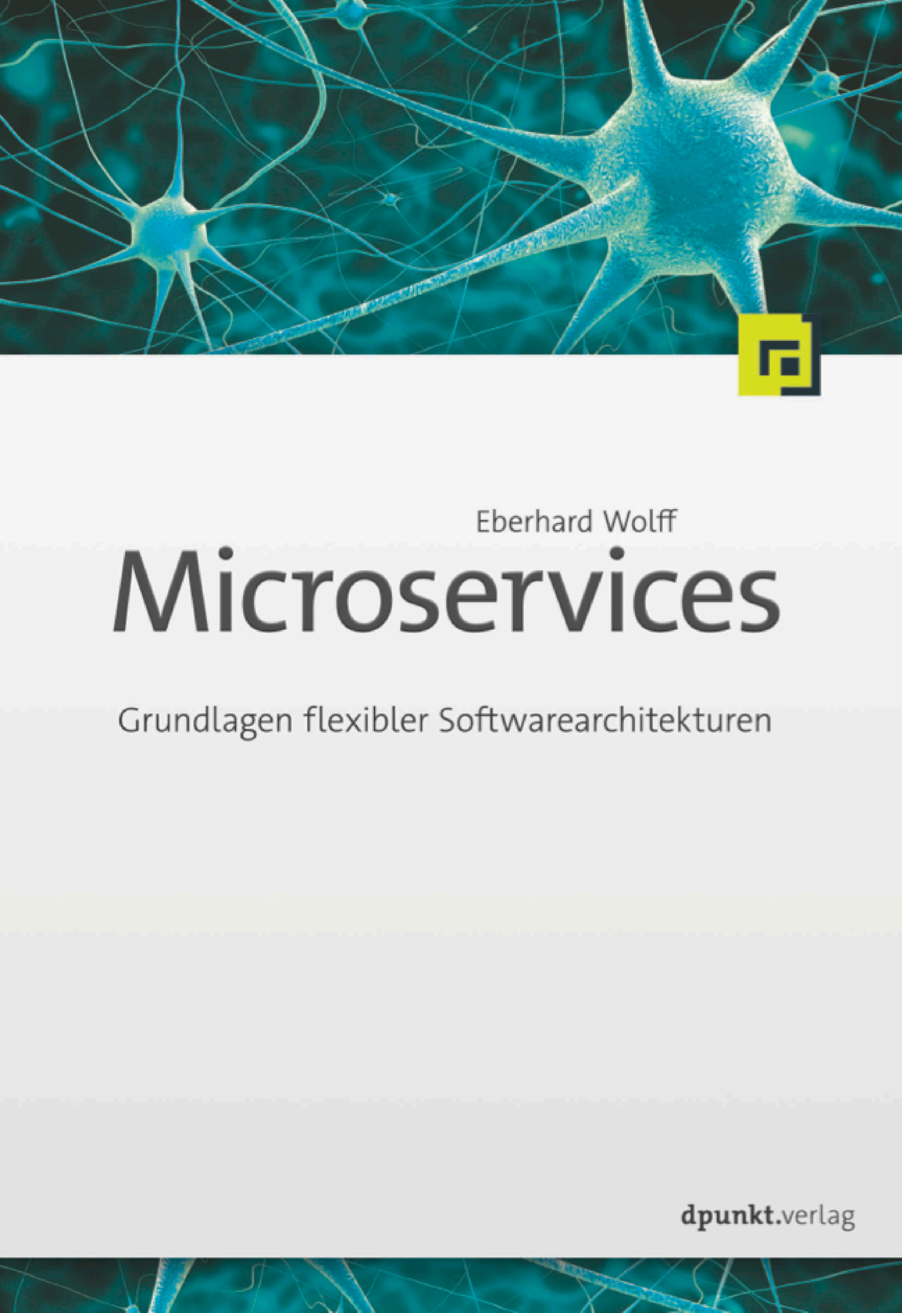
Microservices with Java, Spring Boot & Spring Cloud

Eberhard Wolff
Fellow
@ewolff

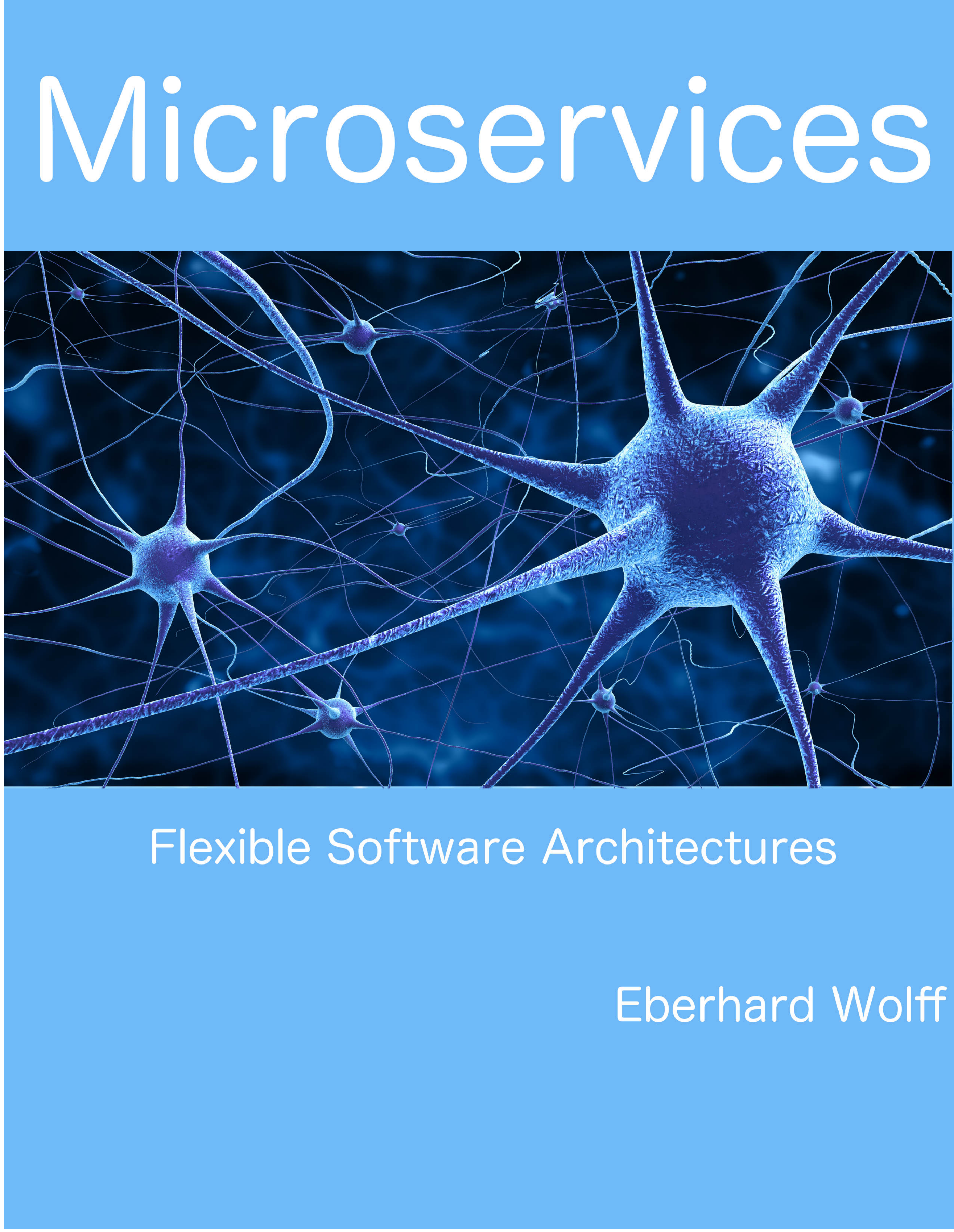


Eberhard Wolff

Microservices

Grundlagen flexibler Softwarearchitekturen

dpunkt.verlag



Microservices

Flexible Software Architectures

Eberhard Wolff

http://microservices-buch.de/

http://microservices-book.com/

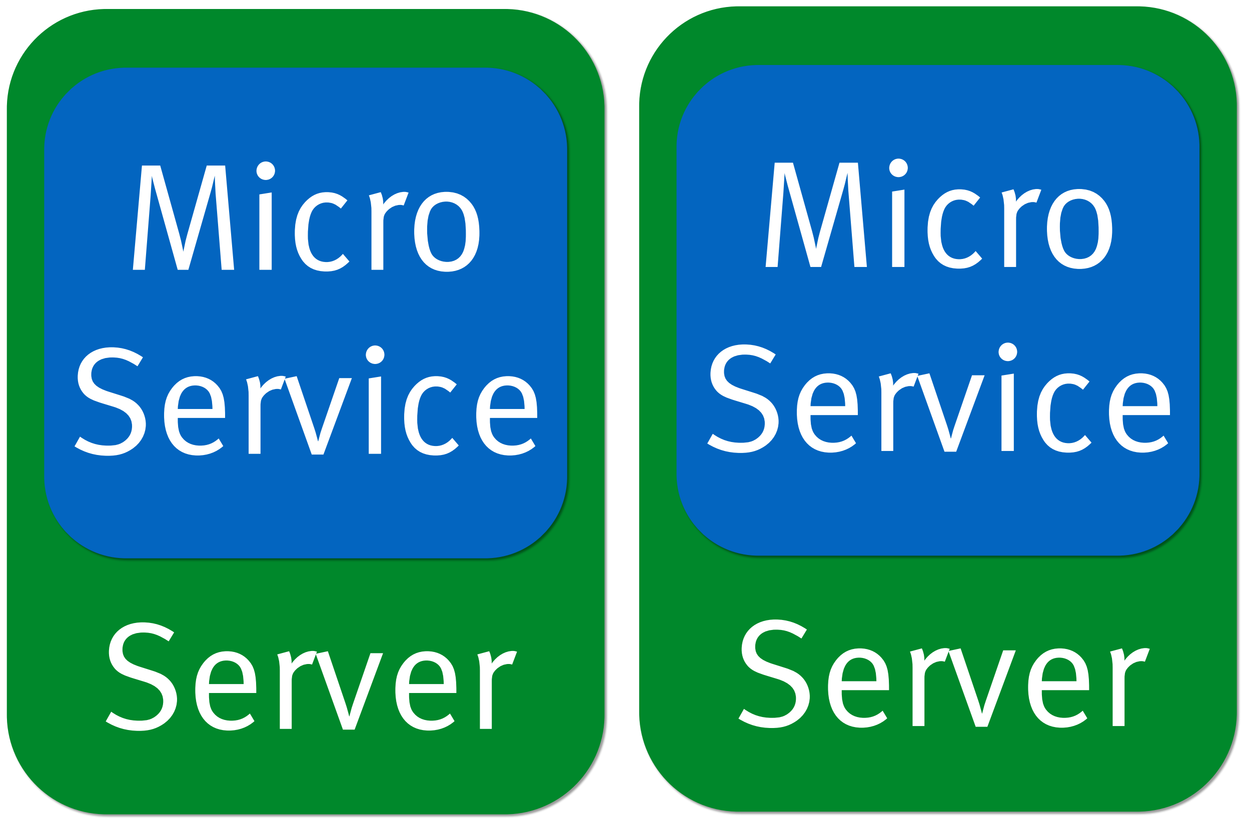
Microservice Definition

innoQ

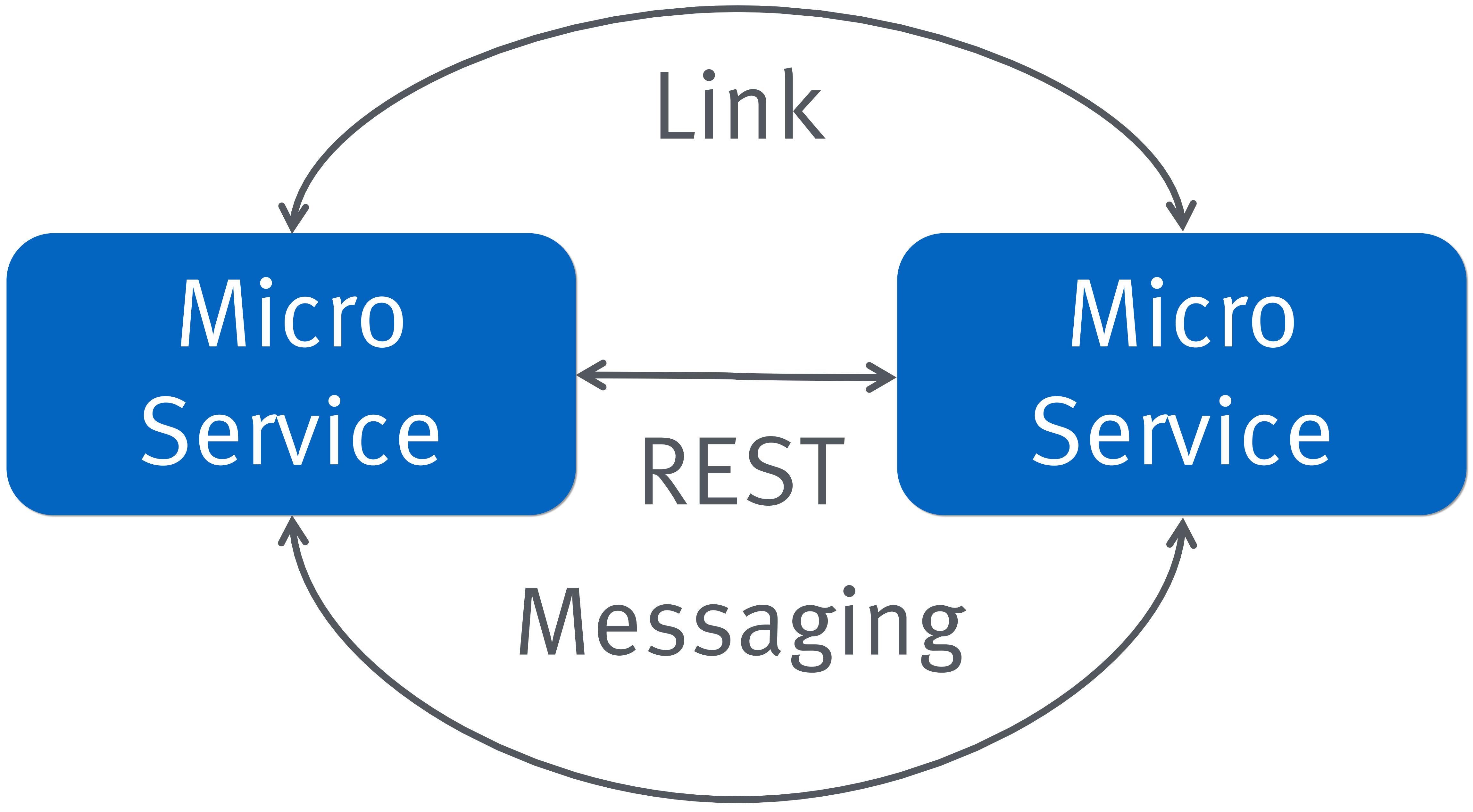
Microservices: Definition

- Small
- Independent deployment units

- Any technology
- Any infrastructure
- UI + Logic



Components Collaborate



Data Replication

Infrastructure

- Lots of services
- Need infrastructure
  - Easy to create a new project
  - REST integrated
  - Messaging supported
  - Uniform operations

Spring Boot Demo

Simple Infrastructure

- One pom.xml
- ...Gradle / Ant

- Very few dependencies
- One plug in
- Versions defined

REST Integrated

- Support in Spring MVC
- As we have seen

- Also support for JAX-RS
- Jersey

Messaging Support

- Numerous Spring Boot Starter

- AMQP (RabbitMQ)
- HornetQ (JMS)
- ActiveMQ (JMS, no starter)

Messaging Support

- Spring JMS abstraction
- Message driven POJOs
- Scalable
- Simplify sending JMS

- Can use other libs, too!
- Boot: everything Spring / Java can do

Infrastructure

- More services
- Need infrastructure
  - Easy to create a new project ✓
  - REST integrated ✓
  - Messaging supported ✓
  - Simple deployment
  - Uniform operations

Deploy

- Just package everything in an executable JAR
- ...or a WAR

- Based on Maven, Ant or Gradle

- Add configuration

Spring Boot Deploy Demo

Deploy

- Install a basic machine
- Install Java
- Copy over JAR
- Optional: Make it a Linux Service (1.3)
- Optional: Create application.properties

Infrastructure

- More services
- Need infrastructure
  - Easy to create a new project ✓
  - REST integrated ✓
  - Messaging supported ✓
  - Simple deployment ✓
  - Uniform operations

Spring Boot Actuator

- Provide information about the application

- Via HTTP / JSON
- …or Metrics

- Can be evaluated by monitoring tools etc.

- Another alternative approach to monitoring

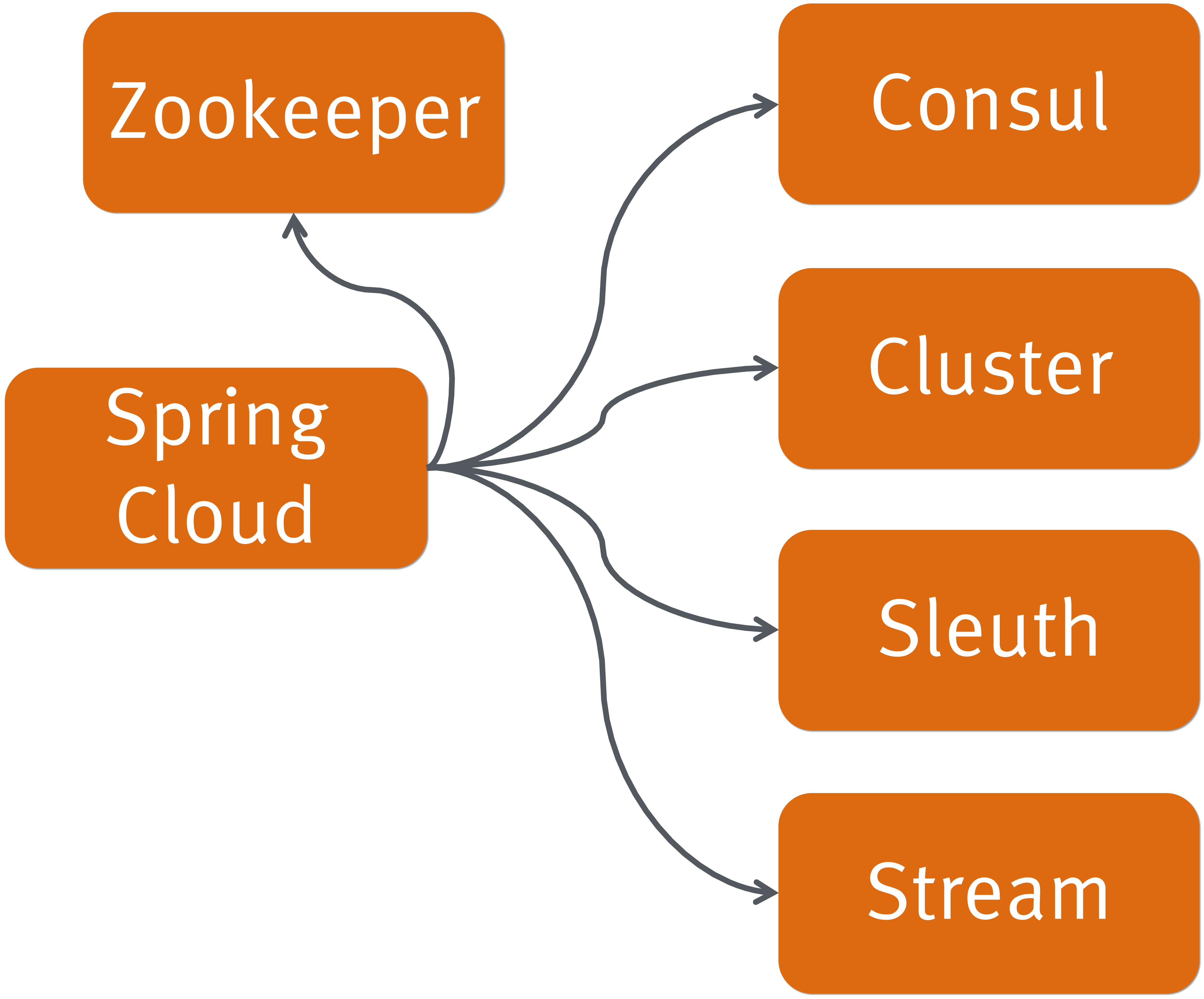
Spring Boot Actuator Demo

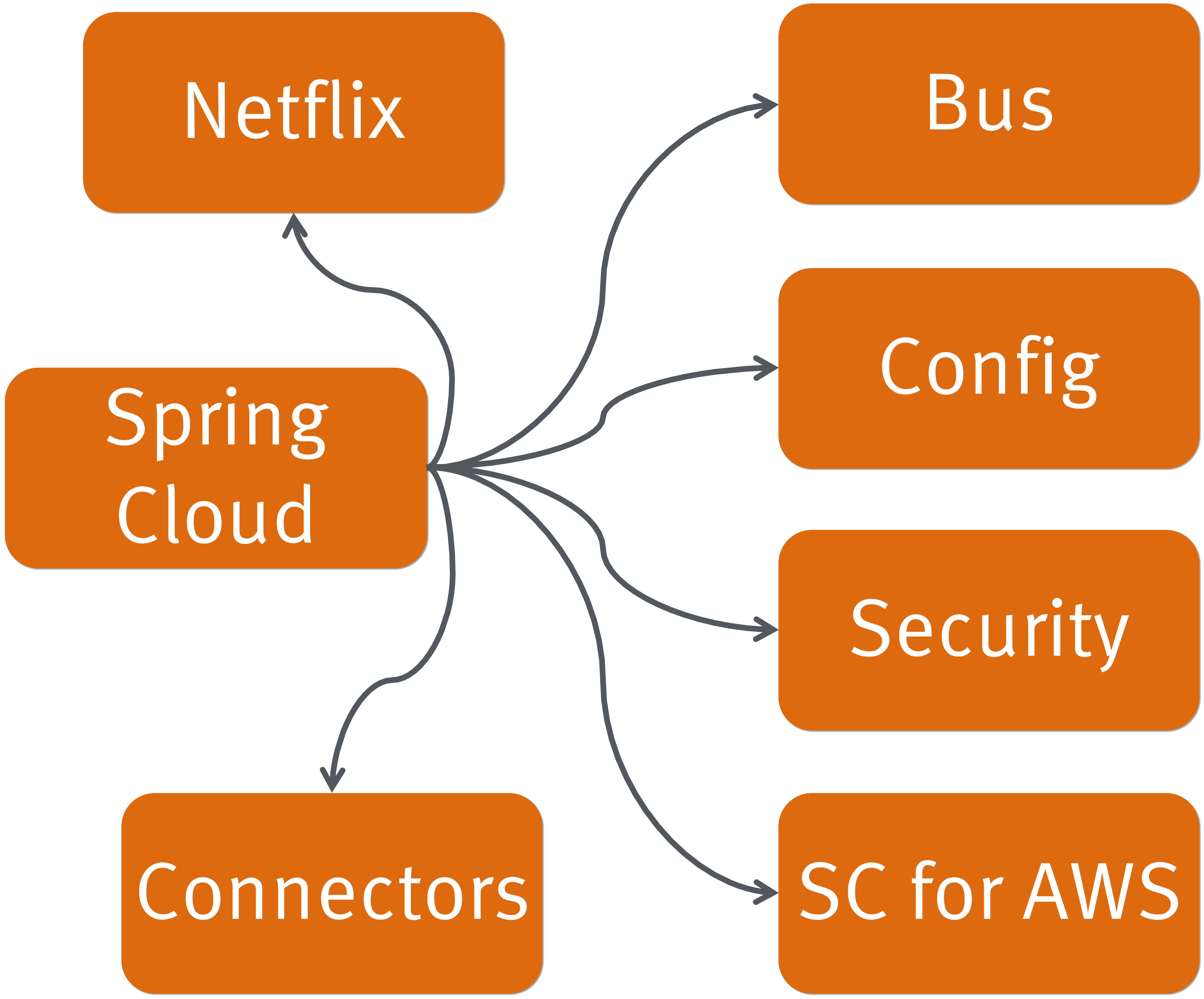
Infrastructure

- More services
- Need infrastructure
  - Easy to create a new project ✓
  - REST integrated ✓
  - Messaging supported ✓
  - Simple deployment ✓
  - Uniform operations ✓

Spring Cloud

Based on Spring Boot


Zookeeper
Consul
Cluster
Spring Cloud
Sleuth
Stream


Netflix
Bus
Config
Spring Cloud
Security
Connectors
SC for AWS

Spring Cloud Netflix

Zuul
Routing

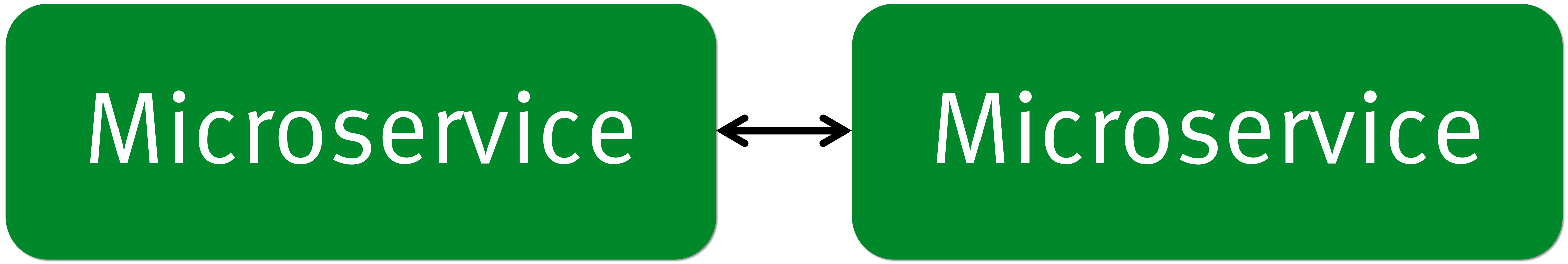
Ribbon
Client Side
Load Balancing

Eureka
Service Discovery

Hystrix
Resilience

Coordinating Microservices

- Must find each other



Service Discovery
Eureka

NETFLIX

Why Eureka?

- REST based service registry
- Supports replication
- Caches on the client
- Resilient
- Fast
- ...but not consistent
- Foundation for other services

Eureka Client in Spring Cloud

- @EnableDiscoveryClient: generic
- @EnableEurekaClient: more specific
- Dependency to spring-cloud-starter-eureka
- Automatically registers application

application.properties

Eureka server
Can include user / password

```
eureka.client.serviceUrl.defaultZone=http://host:8761/eureka/

eureka.instance.leaseRenewalIntervalInSeconds=5

spring.application.name=catalog

eureka.instance.metadataMap.instanceId=${spring.application.name}:${r
    andom.value}

eureka.instance.preferIpAddress=true
```

Faster updates

Used for registration
In CAPITAL caps

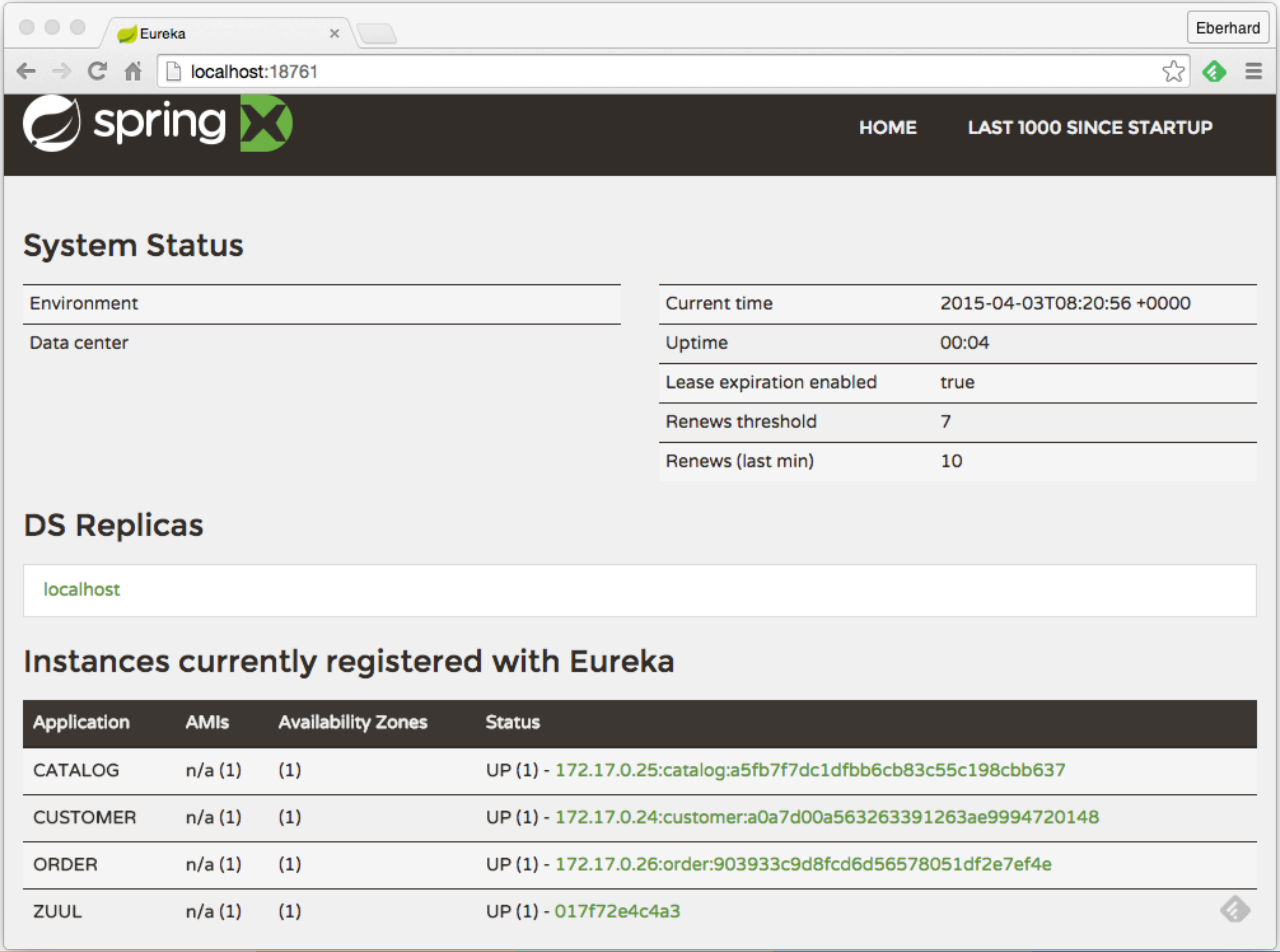
Docker won't resolve host names

Need unique ID
Load balancing

Eureka Server

```
@EnableEurekaServer

@EnableAutoConfiguration

public class EurekaApplication {



    public static void main(String[] args) {

      SpringApplication.run(EurekaApplication.class,
    args);

    }



}
```

Add dependency to
spring-cloud-starter-eureka-server



System Status

Environment	
Data center	

Current time	2015-04-03T08:20:56 +0000
Uptime	00:04
Lease expiration enabled	true
Renews threshold	7
Renews (last min)	10

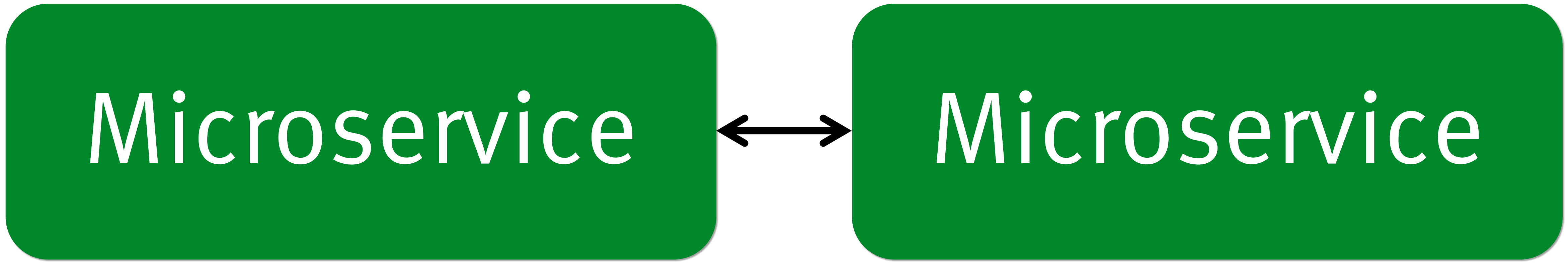
DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CATALOG	n/a (1)	(1)	UP (1) - 172.17.0.25:catalog:a5fb7f7dc1dfbb6cb83c55c198cbb637
CUSTOMER	n/a (1)	(1)	UP (1) - 172.17.0.24:customer:a0a7d00a563263391263ae9994720148
ORDER	n/a (1)	(1)	UP (1) - 172.17.0.26:order:903933c9d8fcd6d56578051df2e7ef4e
ZUUL	n/a (1)	(1)	UP (1) - 017f72e4c4a3

- Must find each other
- Route calls to a service



Zuul
Routing

NETFLIX

Routing

- One URL to the outside

- Internal: Many Microservices

- REST
- Or HTML GUI

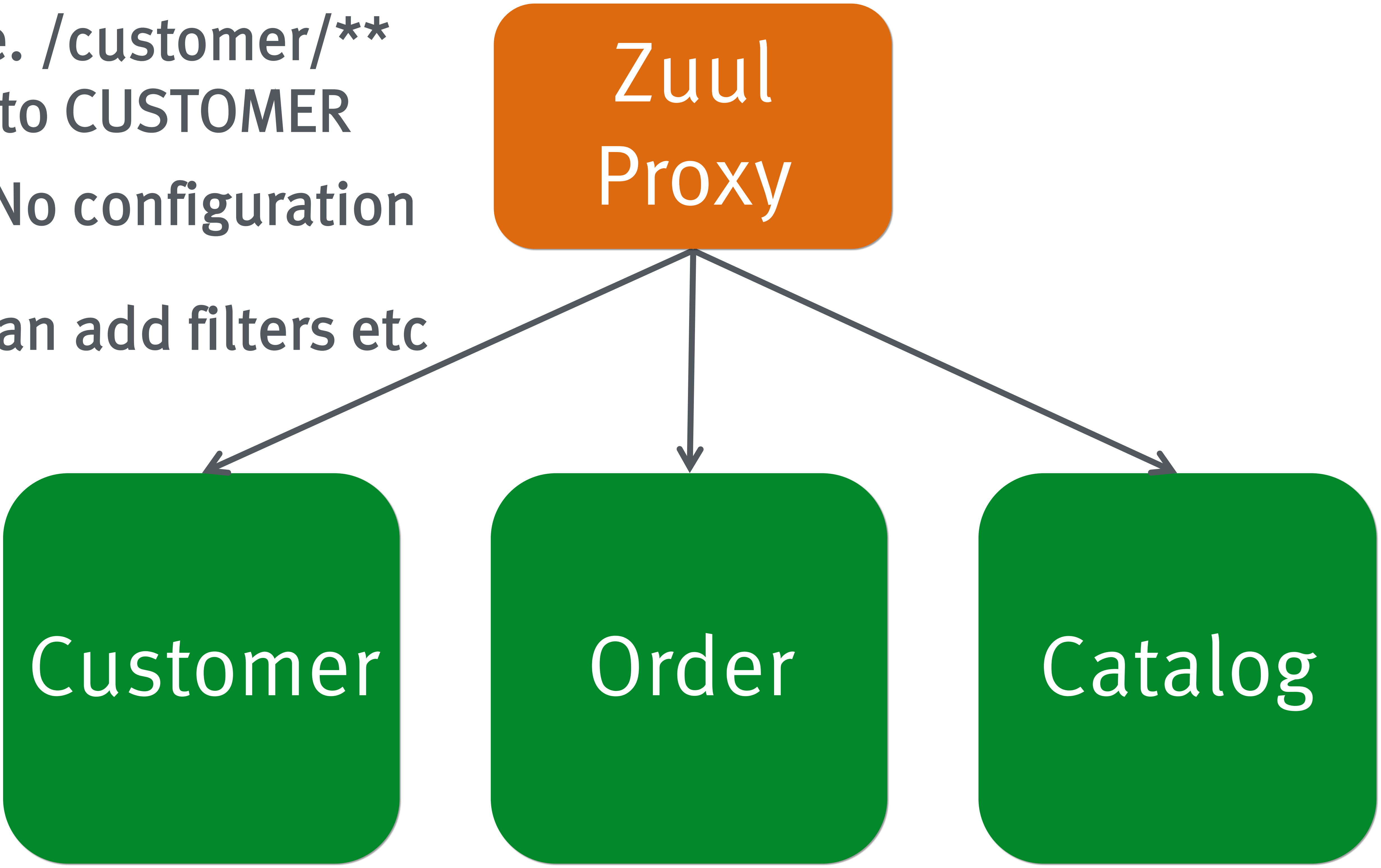
- Power through filters

Automatically maps route to server registered on Eureka

i.e. /customer/** to CUSTOMER

No configuration

Can add filters etc



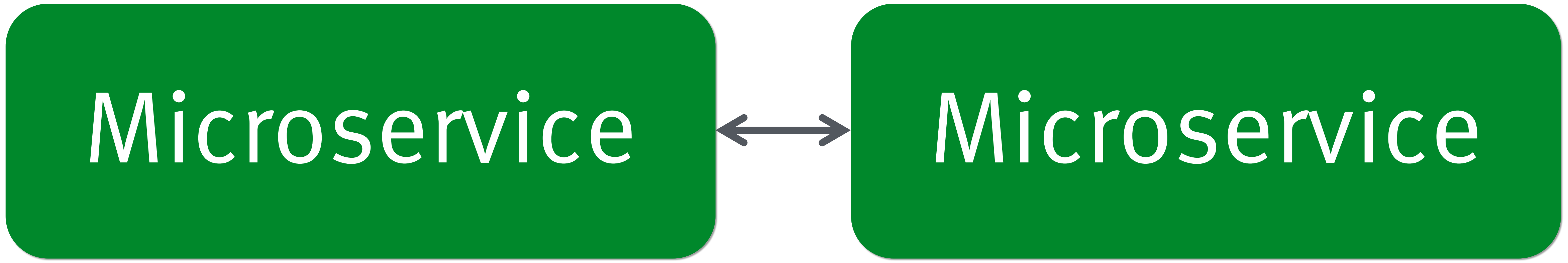
Zuul Proxy

```
@SpringBootApplication
@EnableZuulProxy
public class ZuulApplication {

   public static void main(String[] args) {
     new SpringApplicationBuilder(ZuulApplication.class).
      web(true).run(args);
   }

}
```

Enable Zuul Proxy

Can change route
Also routing to external services possible

- Must find each other
- Route calls to a service
- Configuration



Spring Cloud Config

Configuration

- Spring Cloud Config
- Central configuration
- Dynamic updates
- Can use git backend

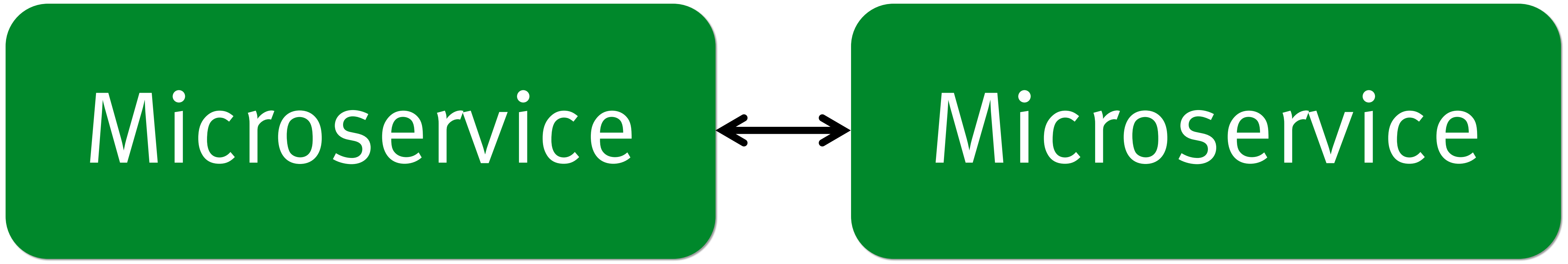
- I prefer immutable server
- & DevOps tools (Docker, Chef…)

Spring Cloud Bus

- Pushed config updates
- …or individual message

- I prefer a messaging solution
- Independent from Spring

- Must find each other
- Route calls to a service
- Configuration
- Security



Spring Cloud Security

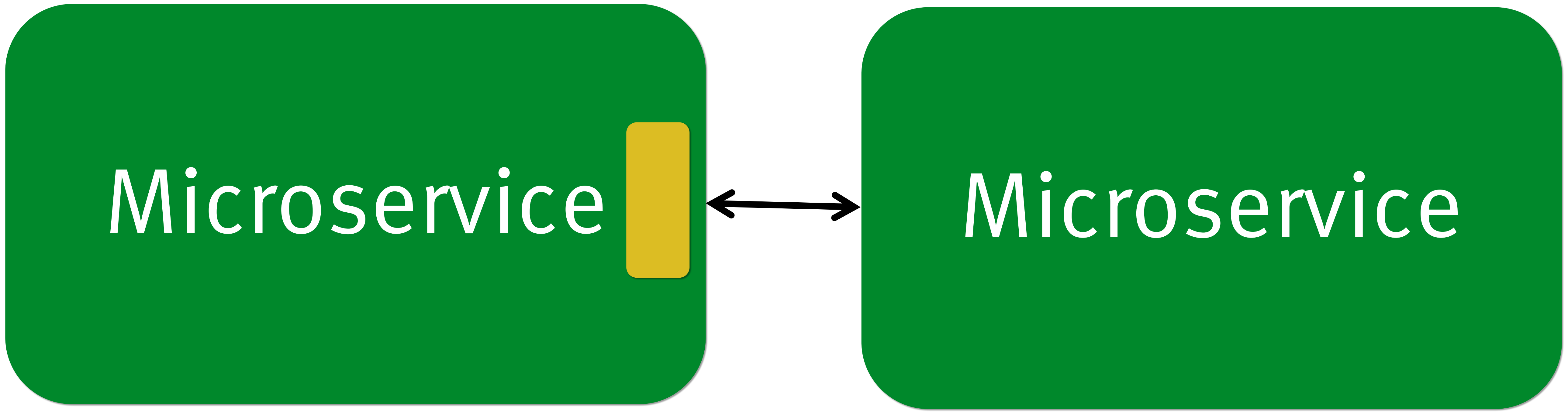
Spring Cloud Security

- Single Sign On via OAuth2
- Forward token e.g. via RestTemplate
- Support for Zuul

- Very valuable!

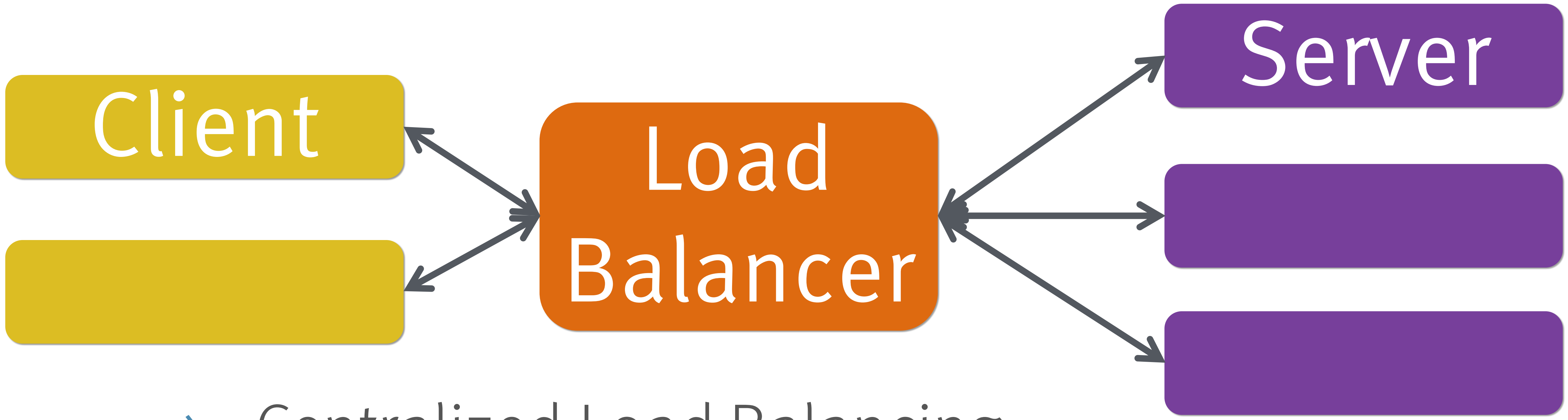
Implementing Microservices

- Load Balancing



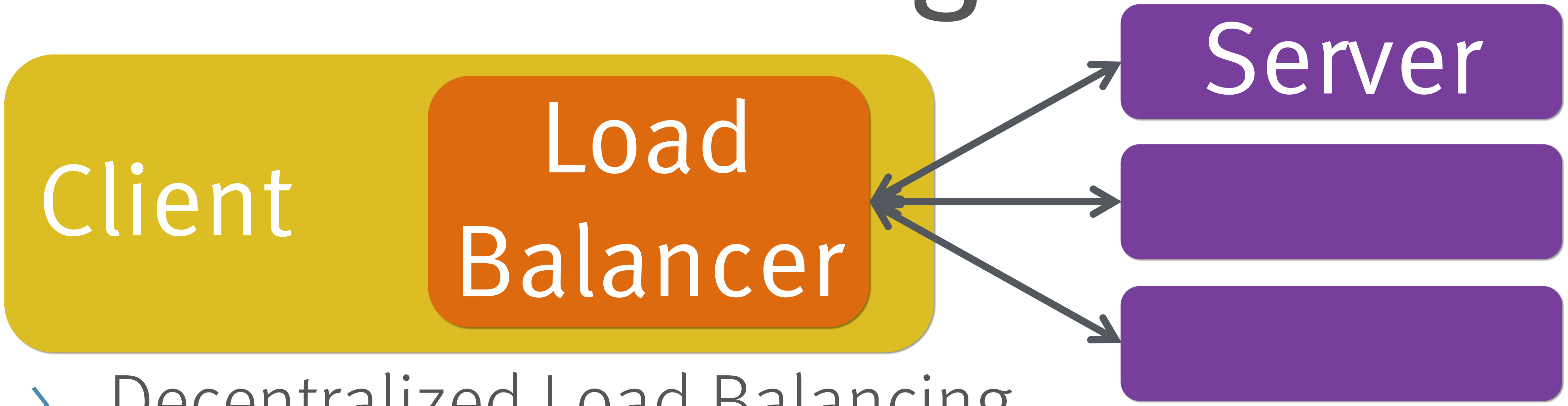
Load Balancing Ribbon

Proxy Load Balancing



- Centralized Load Balancing
- Can become bottle neck
- Single point of failure
- Configuration complex

Ribbon: Client Side Load Balancing



- Decentralized Load Balancing
- No bottle neck
- Resilient
- Can consider response time
- Data might be inconsistent

RestTemplate & Load Balancing

Enable Ribbon

Left out other annotations

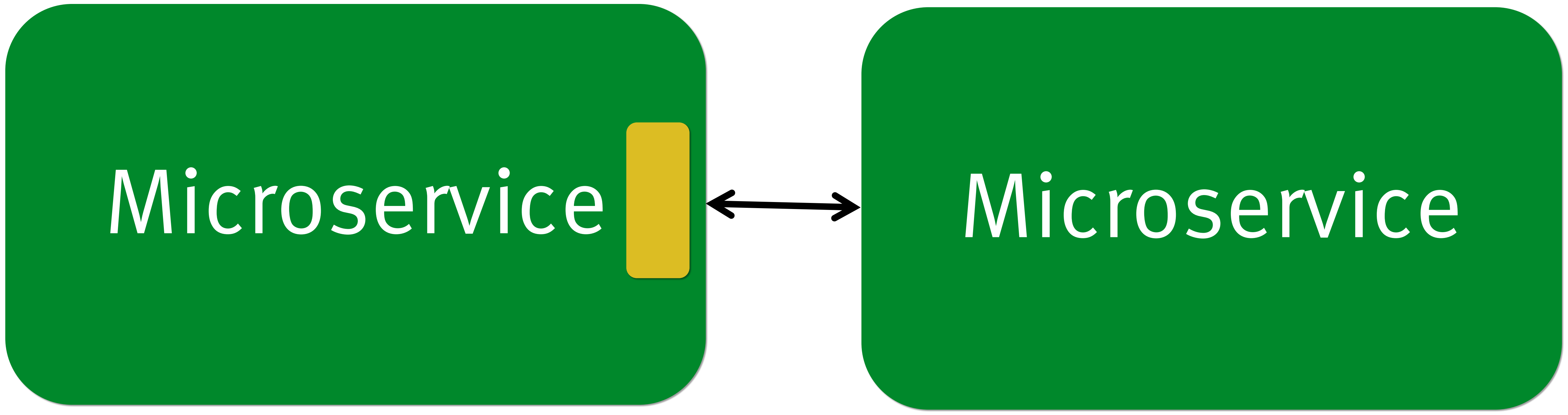
```
@RibbonClient(name = "ribbonApp")

…

public class RibbonApp {

   @Autowired

   private RestTemplate restTemplate;

   public void callMicroService() {

     Store store = restTemplate.


   getForObject("http://stores/store/1",
     Store.class);

   }

}
```

Standard Spring REST client

Can also use Ribbon API

Eureka name or server list

- Load Balancing
- Resilience



Hystrix
Resilience

Hystrix

- Enable resilient applications

- Do call in other thread pool
- Won’t block request handler
- Can implement timeout

Hystrix

- Circuit Breaker
- If call system fail open
- If open do not forward call
- Forward calls after a time window

- System won’t be swamped with requests

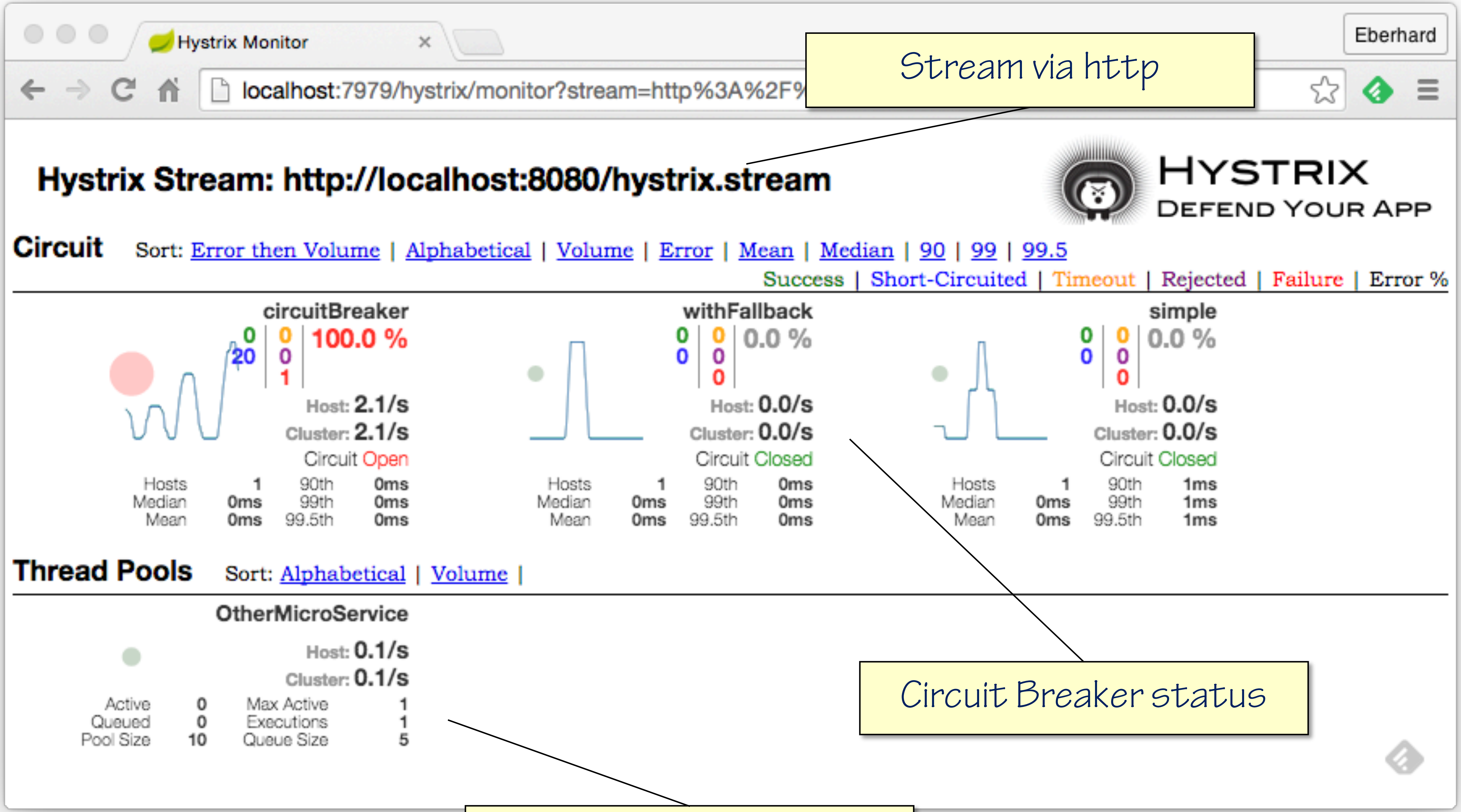
Hystrix / Spring Cloud

- Annotation based approach
- Annotations of javanica libraries
- Java Proxies automatically created

- Simplifies Hystrix dramatically
- No commands etc

Fallback

```
@HystrixCommand(fallbackMethod = "getItemsCache")
public Collection<Item> findAll() {
   …
    this.itemsCache = pagedResources.getContent();
    return itemsCache;
}

private Collection<Item> getItemsCache() {
    return itemsCache;
}
```



Hystrix Monitor

localhost:7979/hystrix/monitor?stream=http%3A%2F%

Eberhard

Hystrix Stream: http://localhost:8080/hystrix.stream

Circuit Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

Success | Short-Circuited | Timeout | Rejected | Failure | Error %

```
circuitBreaker
0   0 | 100.0 %
20  0
    1
Host: 2.1/s
Cluster: 2.1/s
Circuit Open
Hosts      1   90th    0ms
Median   0ms   99th    0ms
Mean     0ms   99.5th  0ms

withFallback
0   0 | 0.0 %
0   0
    0
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts      1   90th    0ms
Median   0ms   99th    0ms
Mean     0ms   99.5th  0ms

simple
0   0 | 0.0 %
0   0
    0
Host: 0.0/s
Cluster: 0.0/s
Circuit Closed
Hosts      1   90th    1ms
Median   0ms   99th    1ms
Mean     0ms   99.5th  1ms
```

Thread Pools Sort: Alphabetical | Volume |

```
OtherMicroService
Host: 0.1/s
Cluster: 0.1/s
Active      0   Max Active   1
Queued      0   Executions   1
Pool Size  10   Queue Size   5
```

Stream via http

Circuit Breaker status

Thread Pool status

Conclusion

Infrastructure

- Easy to create a new project ✓
- REST integrated ✓
- Messaging supported ✓
- Simple deployment ✓
- Uniform operations ✓

Spring Cloud

- Eureka: Service Discovery
- Zuul: Route calls to a service
- Spring Cloud Config: Configuration
- Ribbon: Load Balancing
- Hystrix: Resilience

Links

- http://projects.spring.io/spring-boot/
- http://projects.spring.io/spring-cloud
- https://github.com/ewolff/spring-boot-demos
- https://github.com/ewolff/microservices
- https://spring.io/guides/

Thank You!!
@ewolff