Microservices with Spring Boot & Spring Cloud

Eberhard Wolff

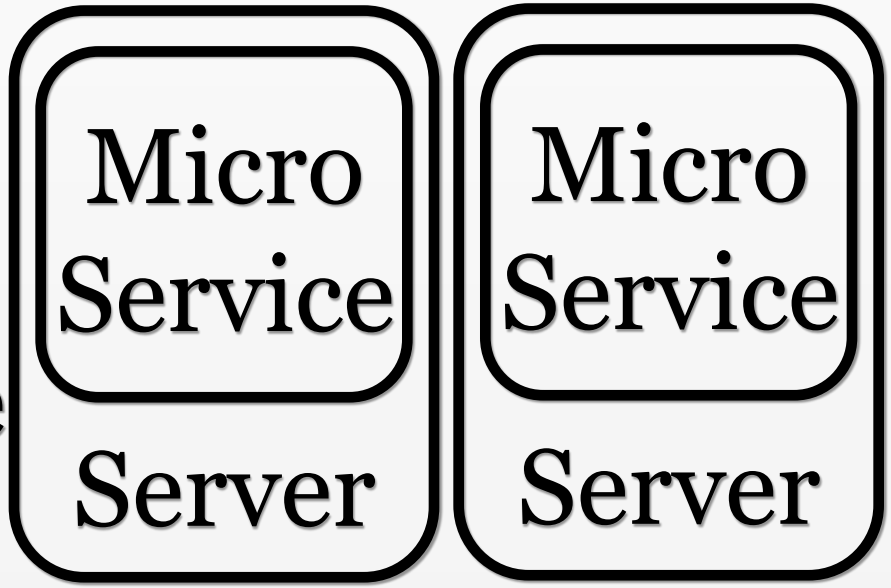
Freelancer / Trainer

What are Microservices?

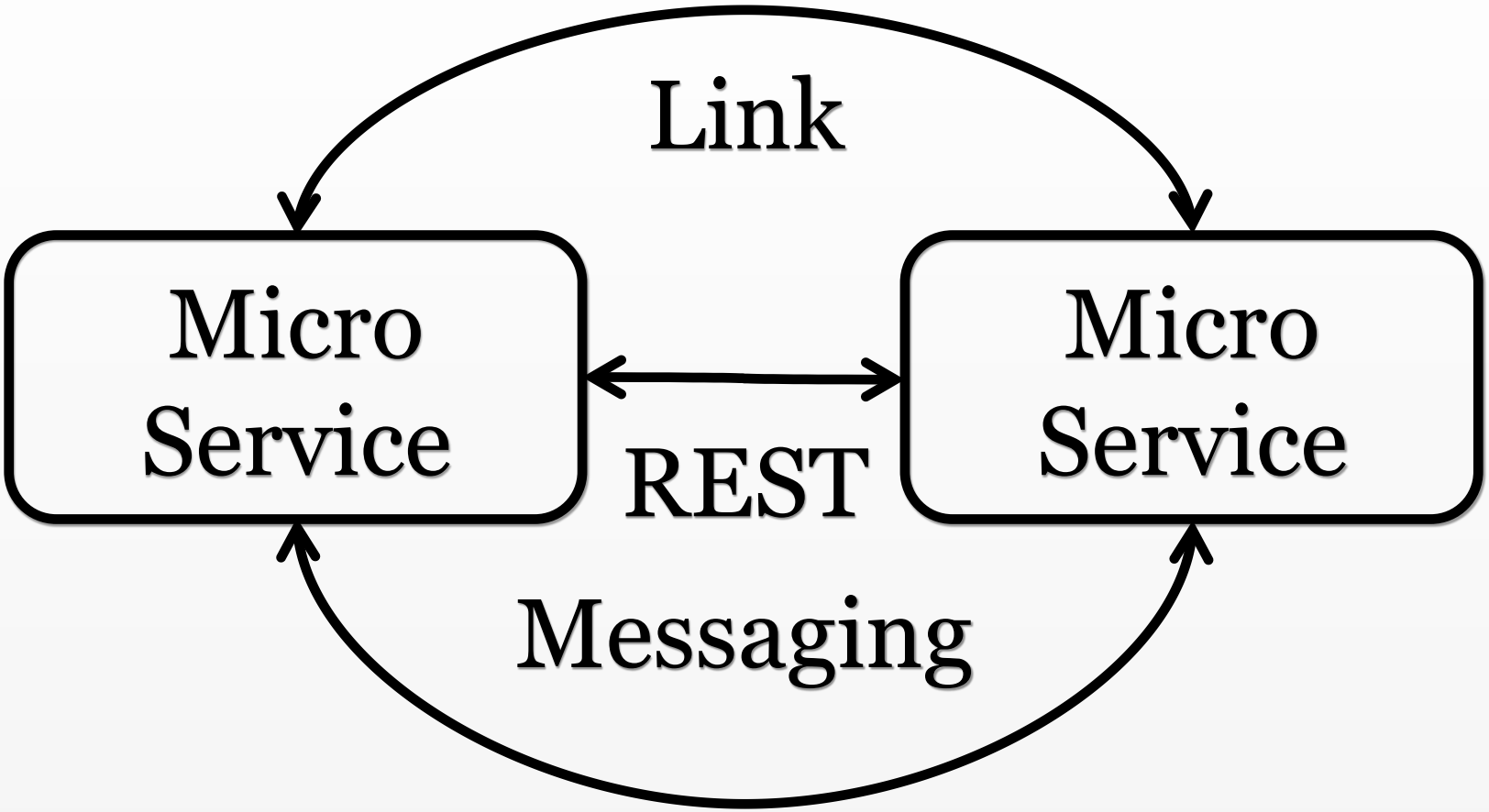
Micro Services: Definition

- Small
- Independent deployment units
- i.e. processes or VMs

- Any technology
- Any infrastructure
- Include GUI



Components Collaborate



Infrastructure for Microservices

- Lots of services

- Need infrastructure

Easy to create a new project

REST integrated

Messaging supported

Uniform operations

Spring Boot Demo

Easy to Create New Project

- One pom.xml
- …Gradle / Ant

- Very few dependencies
- One plug in
- Versions defined for you

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 can do everything plain Spring / Java can do

Infrastructure for Microservices

- More services

- Need infrastructure

Easy to create a new project ✓

REST integrated ✓

Messaging supported ✓

Simple deployment

Uniform operations

Spring Boot Deploy Demo

Deploy

- Just package everything in an executable JAR
- …or a WAR
- Based on Maven, Ant or Gradle
- Build in configuration (YAML, properties etc.)

Deploy

- Install a basic machine
- Install Java
- Copy over .jar
- Optional: Create application.properties

Infrastructure for Microservices

- 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
- Can be evaluated by monitoring tools etc.
- Another alternative approach to monitoring

Spring Boot Actuator Demo

Infrastructure for Microservices

- 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

Deploy

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

Spring Cloud

Based on Spring Boot

Spring Cloud

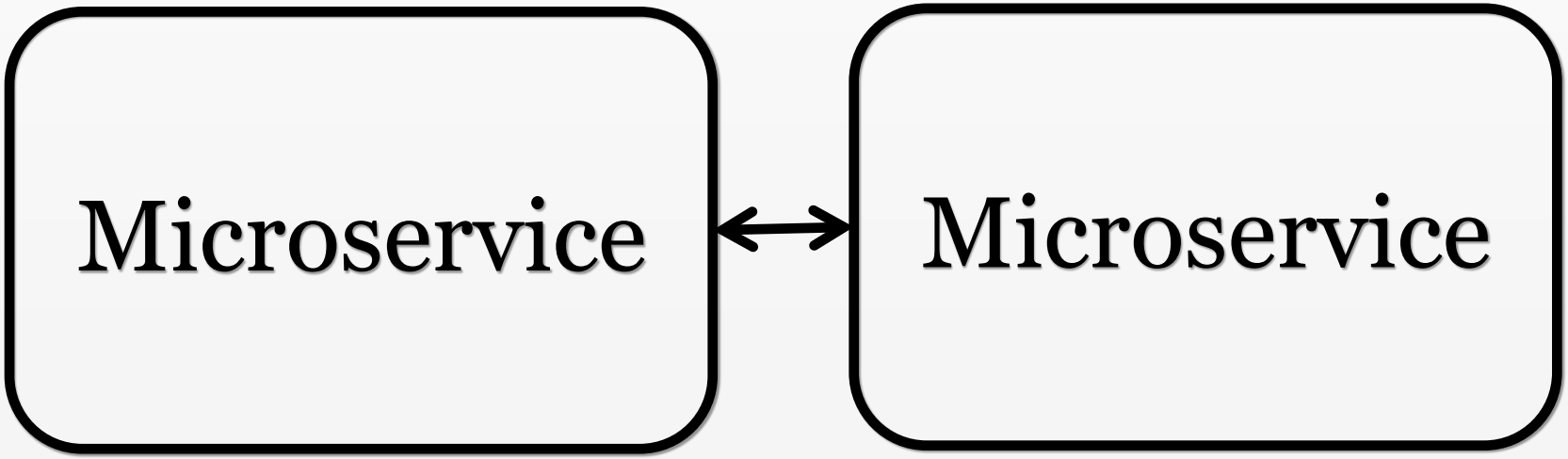
- Spring support for Amazon Web Services
- Connector for Heroku PaaS
- ...and Cloud Foundry PaaS

- The rest of Spring Cloud is for Microservices

Coordinating Microservices

Eberhard Wolff - @ewolff

Must find each other



Service Discovery Eureka

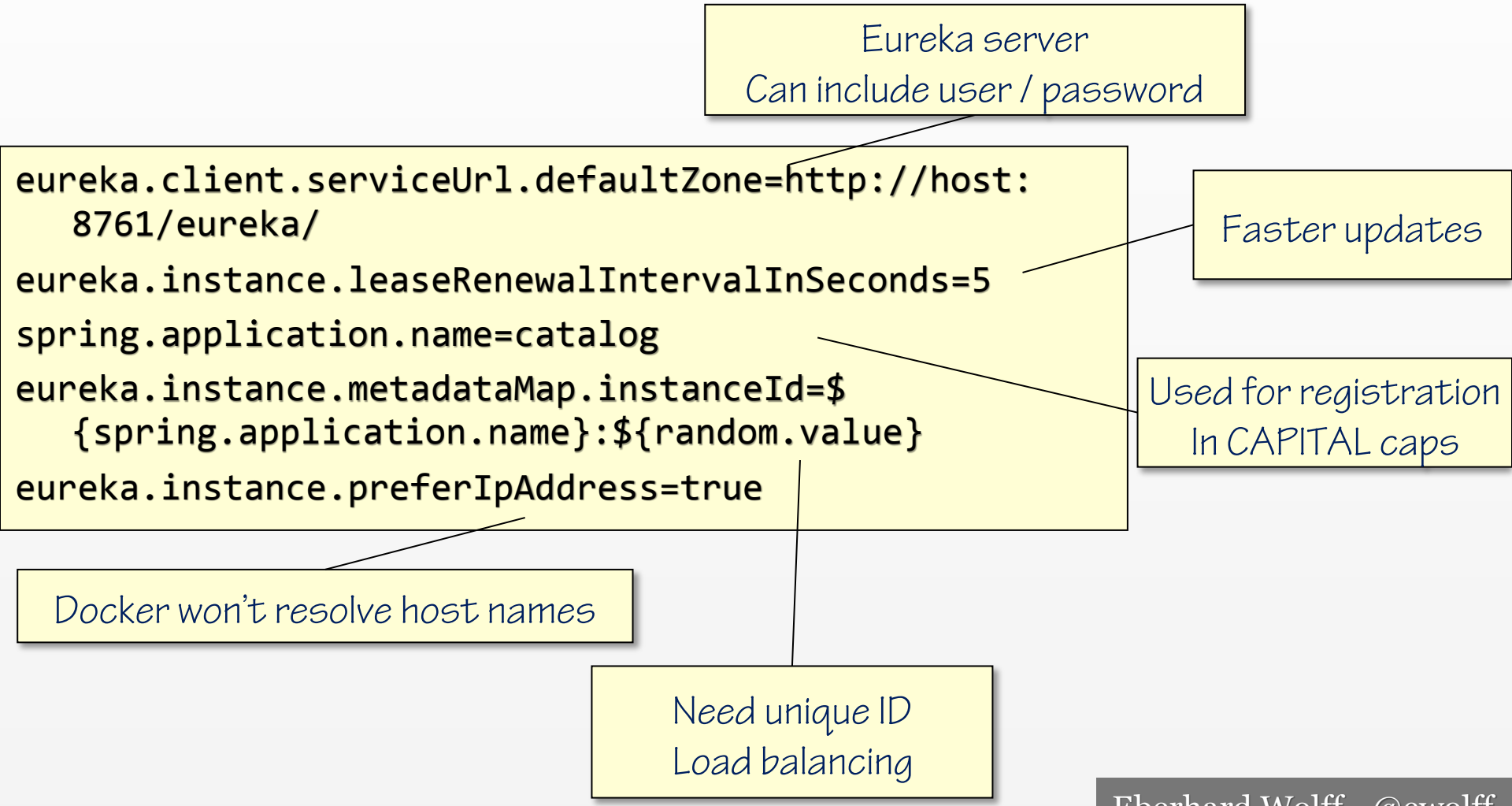
Why Eureka?

- 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



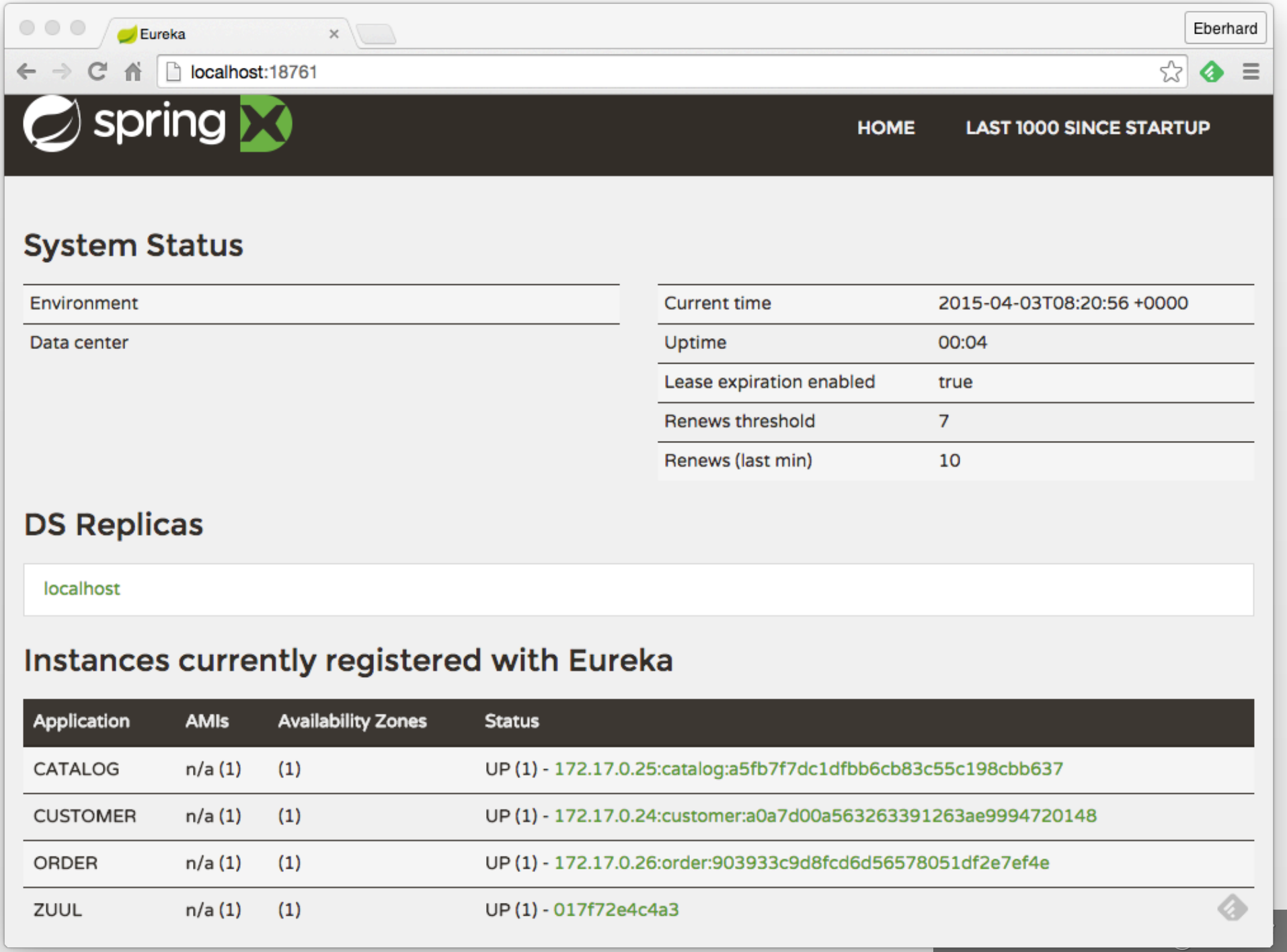
Eureka Server

```
@EnableEurekaServer
@EnableAutoConfiguration
public class EurekaApplication {

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

}
```

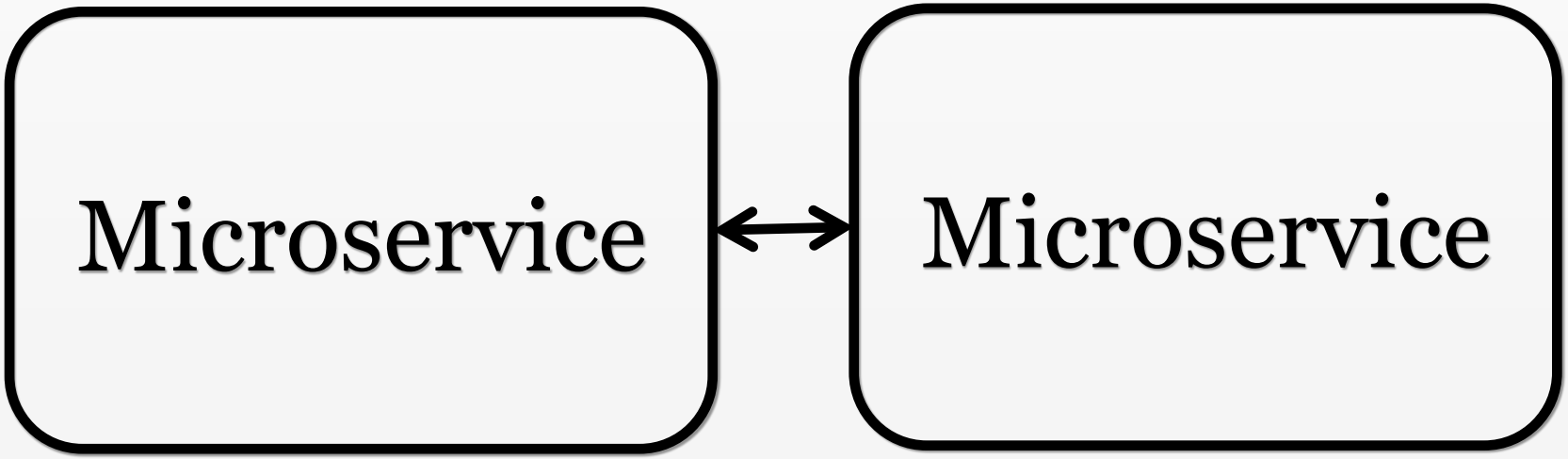
Add dependency to
spring-cloud-starter-eureka-server


Eureka
Eberhard
localhost:18761
spring X
HOME
LAST 1000 SINCE STARTUP
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

Eureka Demo

Must find each other

Route calls to a service



Zuul Routing

Routing

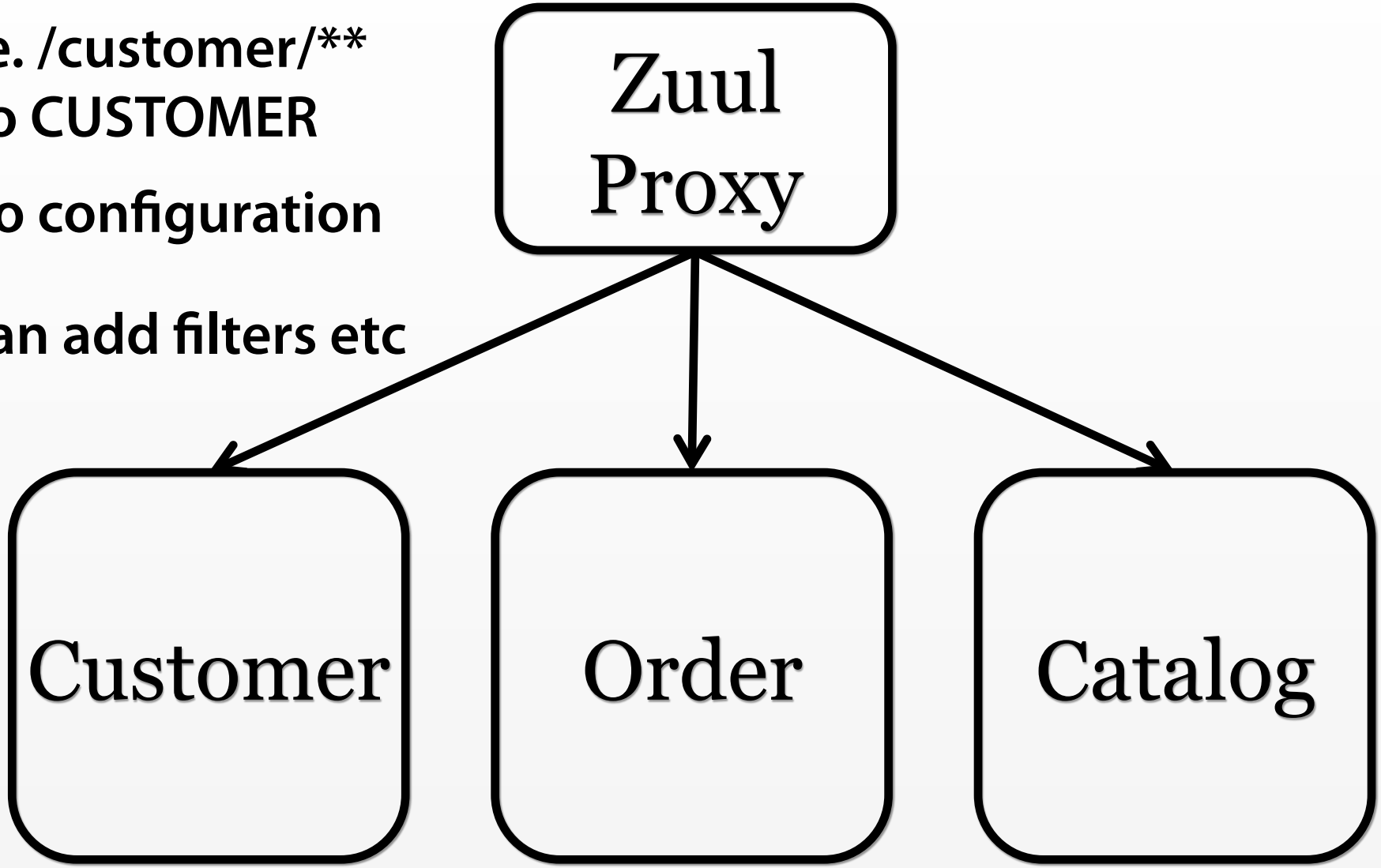
- One URL to the outside
- Internal: Many Microservices
- REST
- Or HTML GUI

Automatically maps route to server registered on Eureka

i.e. /customer/**
to CUSTOMER

No configuration

Can add filters etc



Zuul Proxy

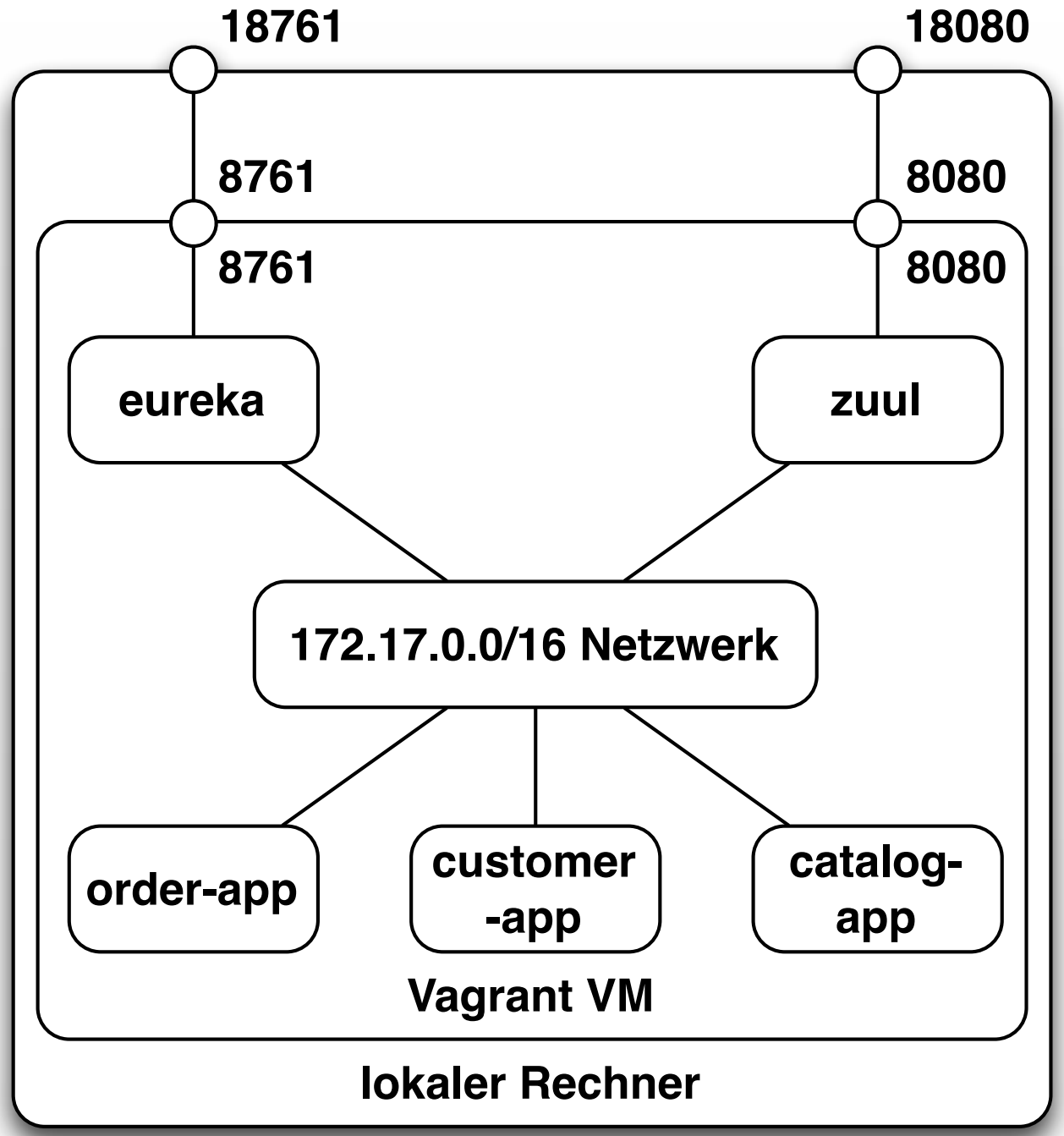
Enable Zuul Proxy

```
@SpringBootApplication
@EnableZuulProxy
public class ZuulApplication {

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

}
```

Can change route
Also routing to external services possible

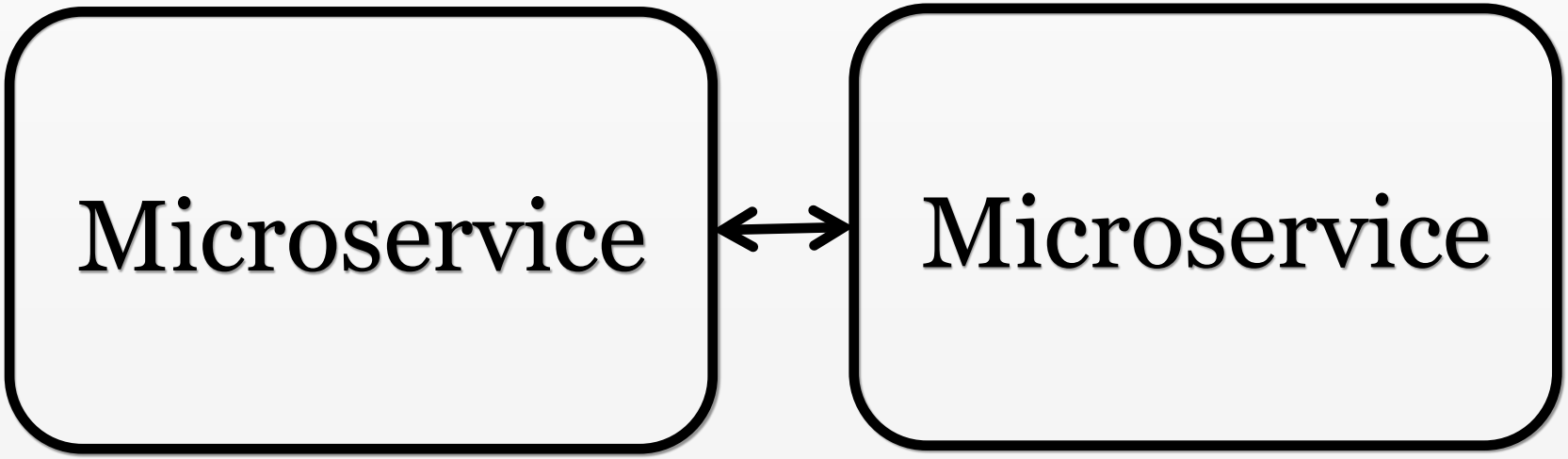

18761
18080
8761
8080
8761
8080
eureka
zuul
172.17.0.0/16 Netzwerk
order-app
customer
-app
catalog-
app
Vagrant VM
lokaler Rechner

Zuul Demo

Must find each other

Route calls to a service

Configuration



Configuration

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

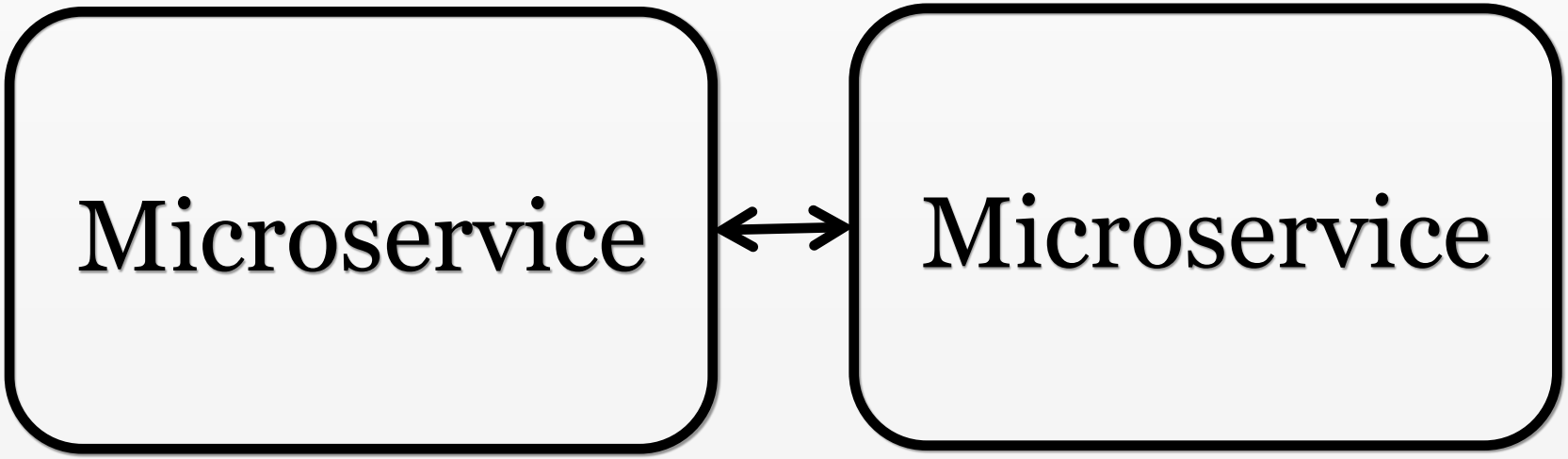
- I prefer immutable server
- & DevOps tools (Docher, Chef…)

Must find each other

Route calls to a service

Configuration

Communication



Spring Cloud Bus

Spring Cloud Bus

- Pushed config updates
- …or individual message

- I prefer a messaging solution
- Independent from Spring

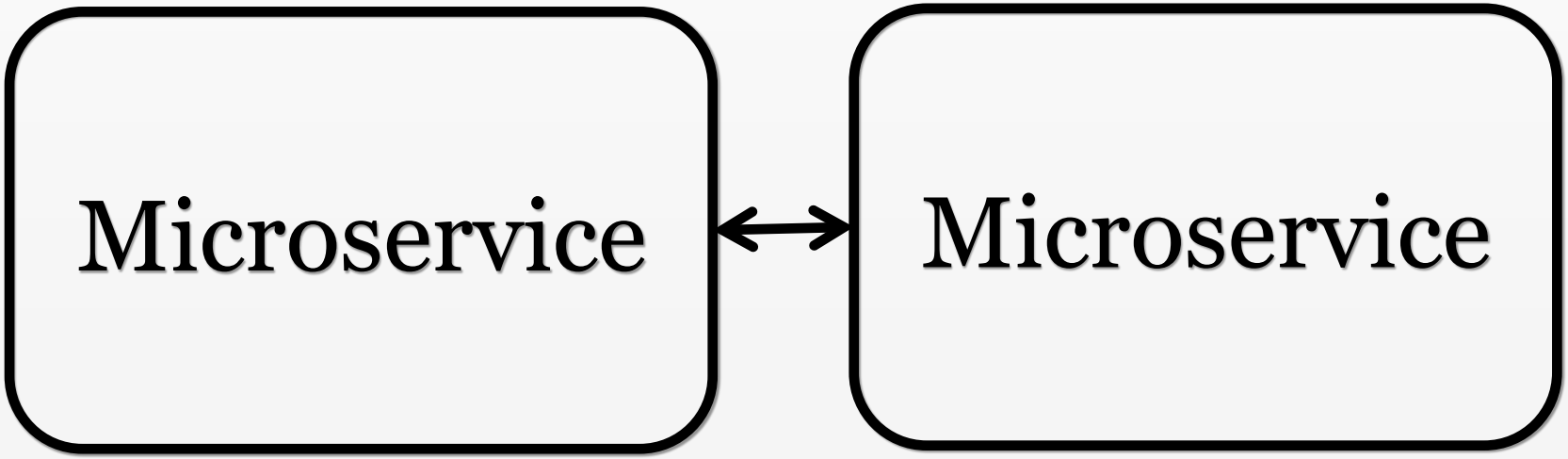
Must find each other

Route calls to a service

Configuration

Communication

Security



Spring Cloud Security

Eberhard Wolff - @ewolff

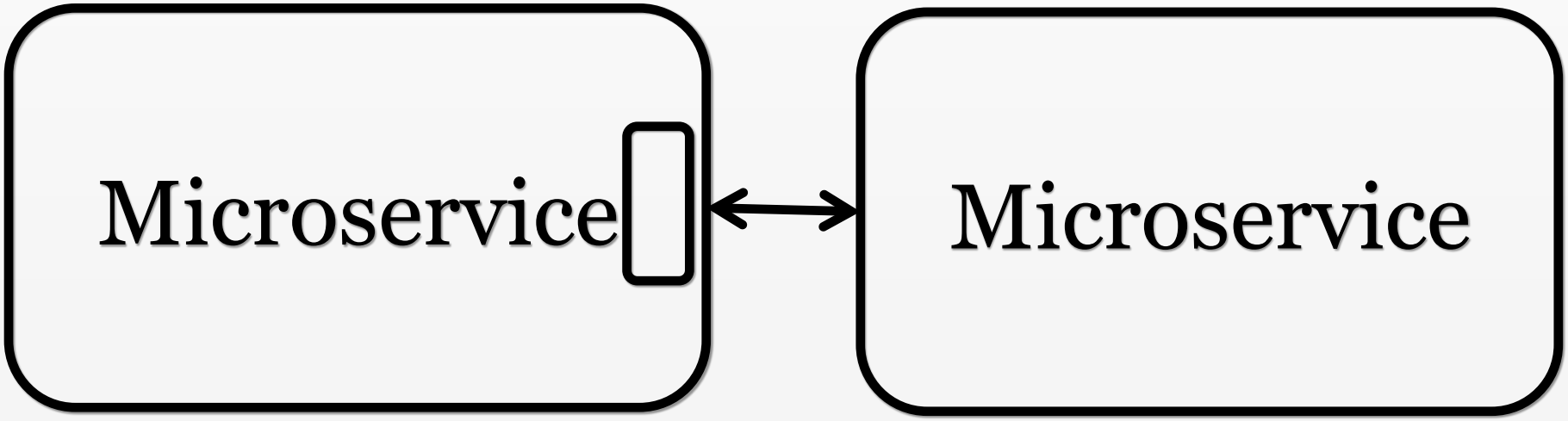
Spring Cloud Security

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

- Very valuable!

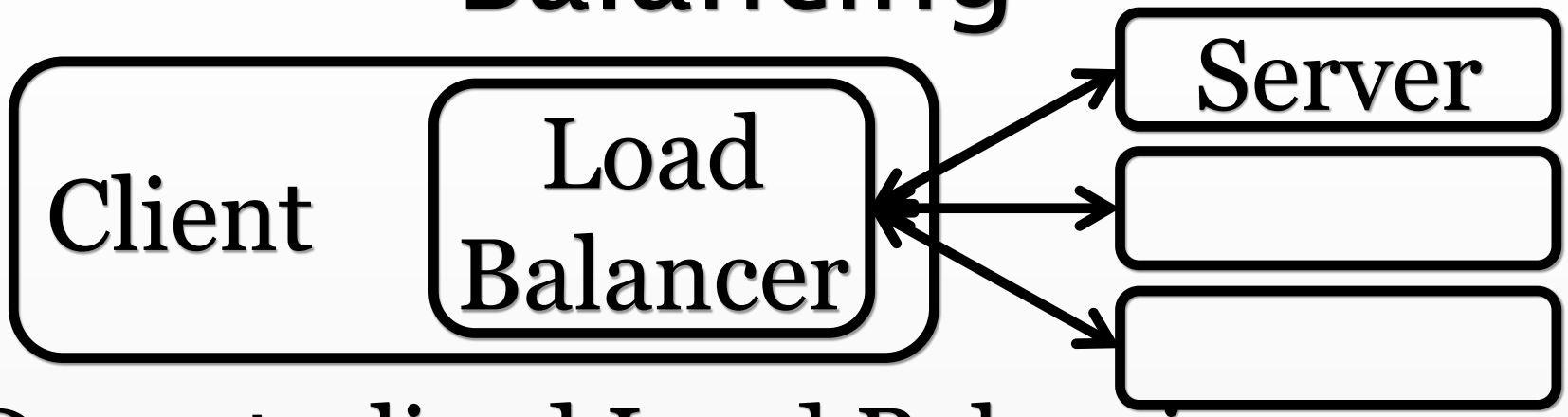
Implementing Microservices

Load Balancing



Load Balancing Ribbon

Ribbon: Client Side Load Balancing



- Decentralized Load Balancing
- No bottle neck
- Resilient
- Hard to consider metrics / health
- 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);
   }

}
```

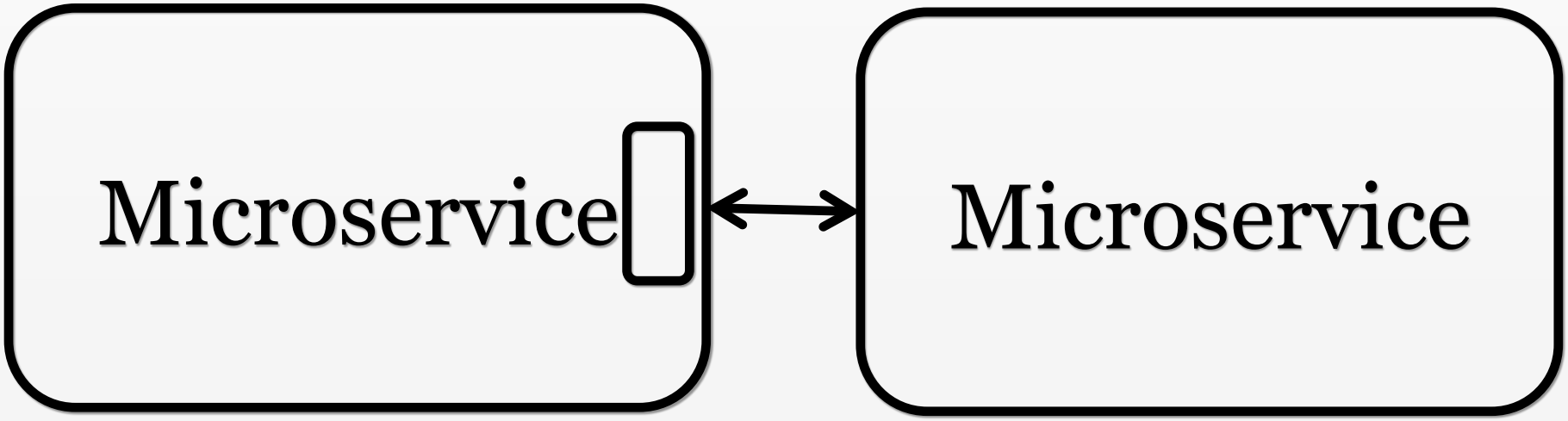
Standard Spring REST client

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

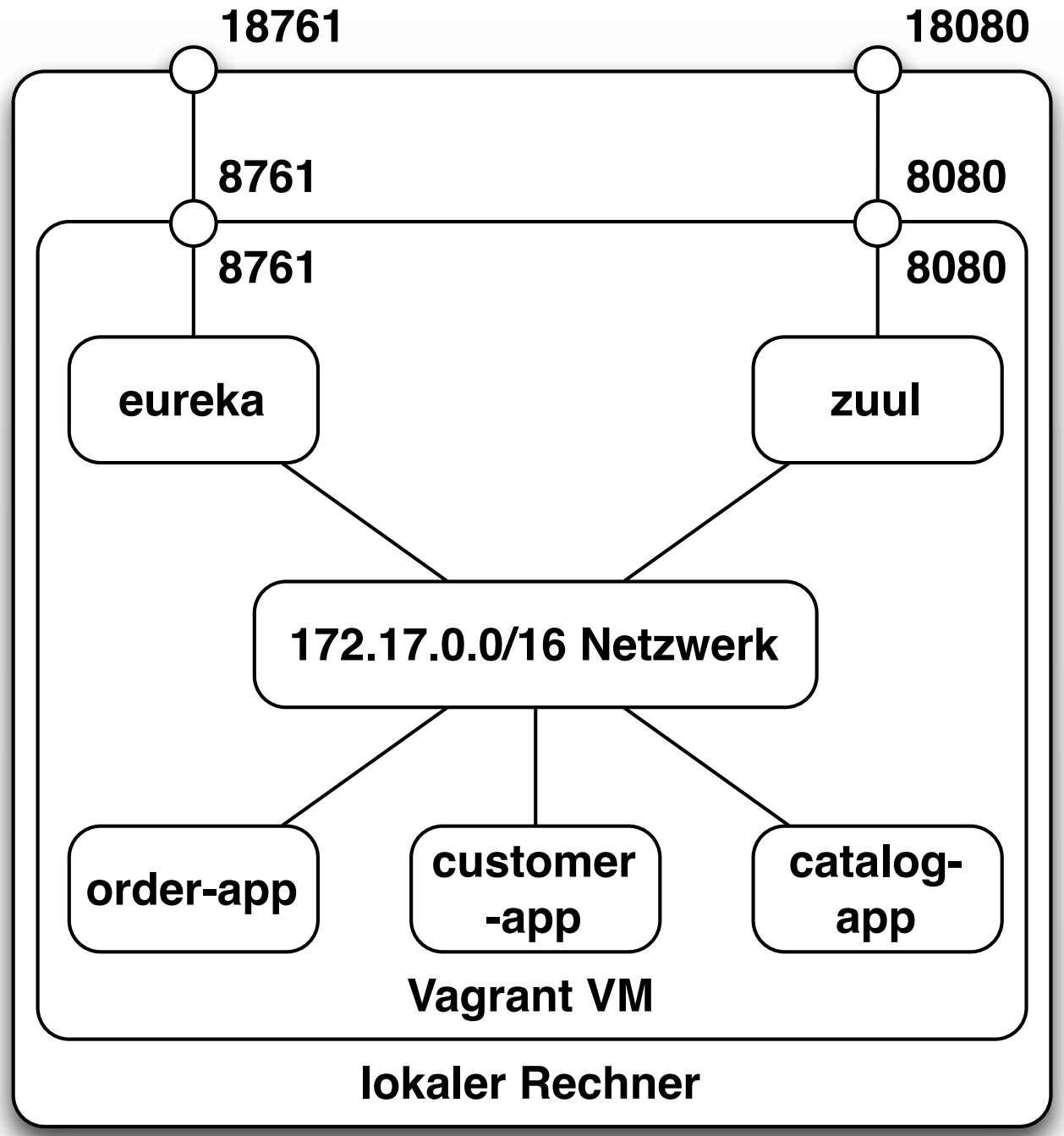
- Annotation based approach
- Java Proxies automatically created
- Annotations of javanica libraries

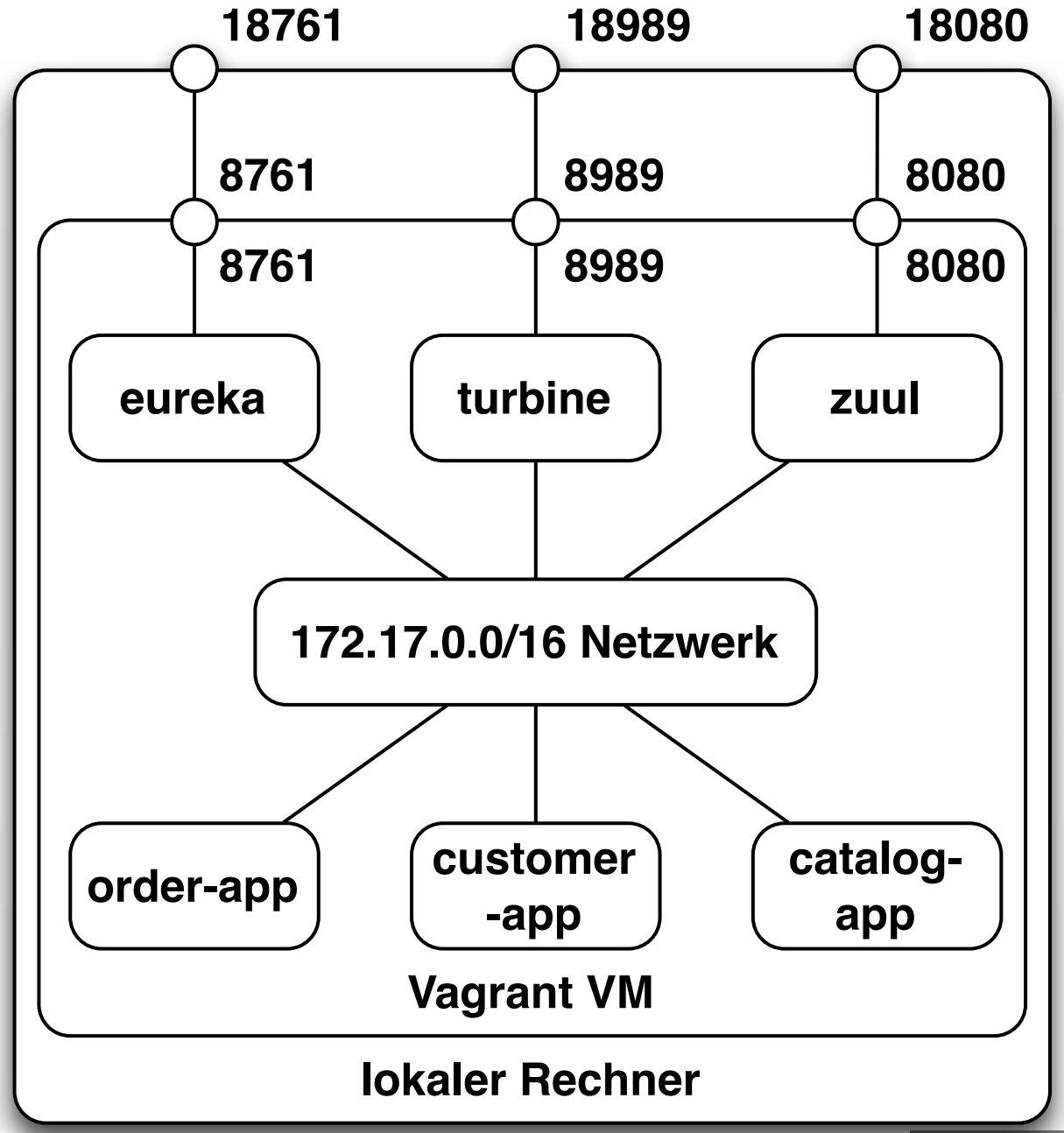
- Simplifies Hystrix dramatically
- No commands etc

Fallback

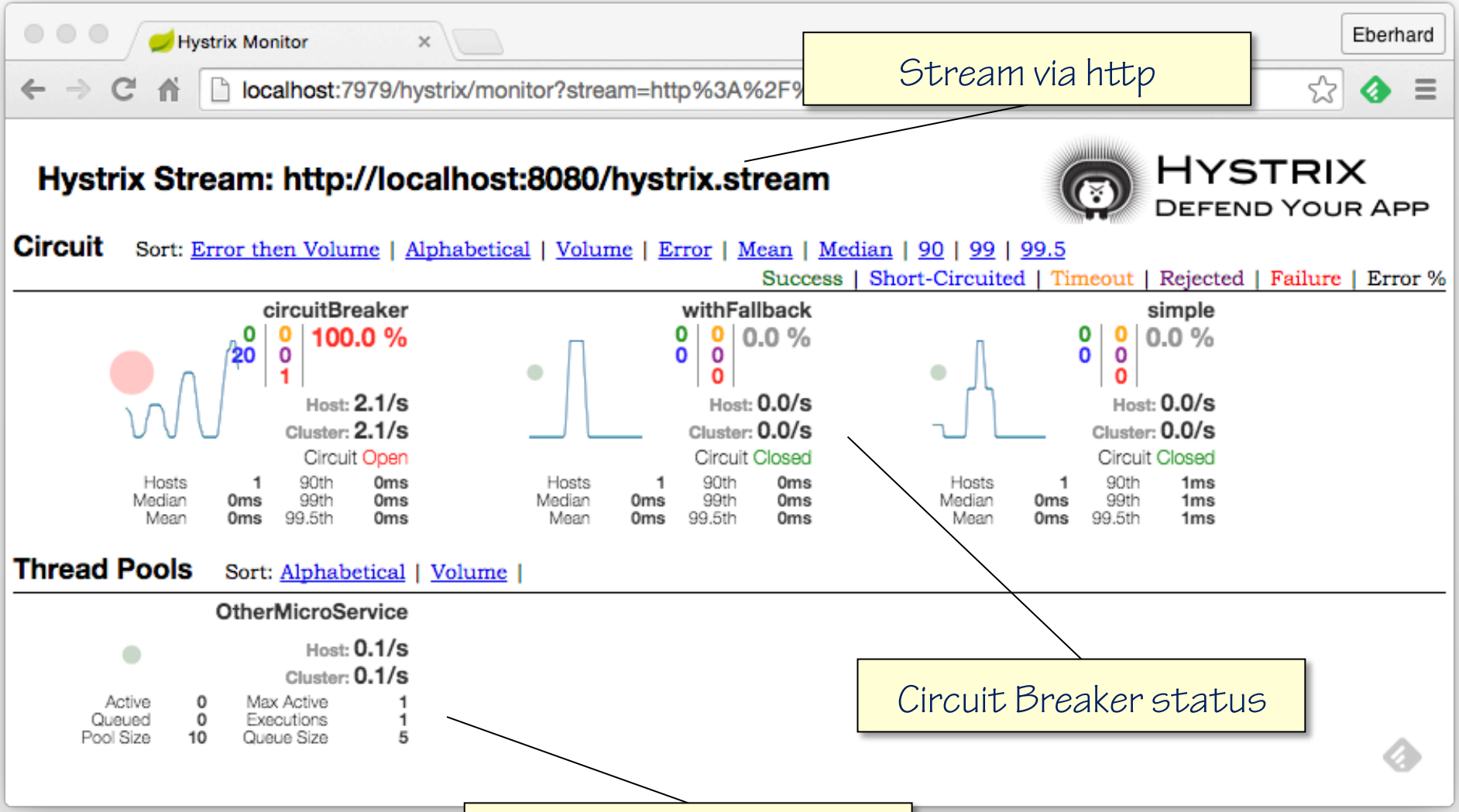
```
@HystrixCommand(fallbackMethod = "getItemsCache")
public Collection<Item> findAll() {
   …
   this.itemsCache = pagedResources.getContent();
   return itemsCache;
}

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


18761
18080
8761
8080
8761
8080
eureka
zuul
172.17.0.0/16 Netzwerk
order-app
customer
-app
catalog-
app
Vagrant VM
lokaler Rechner


18761
18989
18080
8761
8989
8080
8761
8989
8080
eureka
turbine
zuul
172.17.0.0/16 Netzwerk
order-app
customer
-app
catalog-
app
Vagrant VM
lokaler Rechner

Hystrix Dashboard



Conclusion

Spring Boot for Microservices

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

Hystrix Demo

Spring Cloud for Microservices

Must find each other: Service Discovery

Route calls to a service

Configuration

Communication

Load Balancing

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!!