Microservices

Are your Frameworks ready?

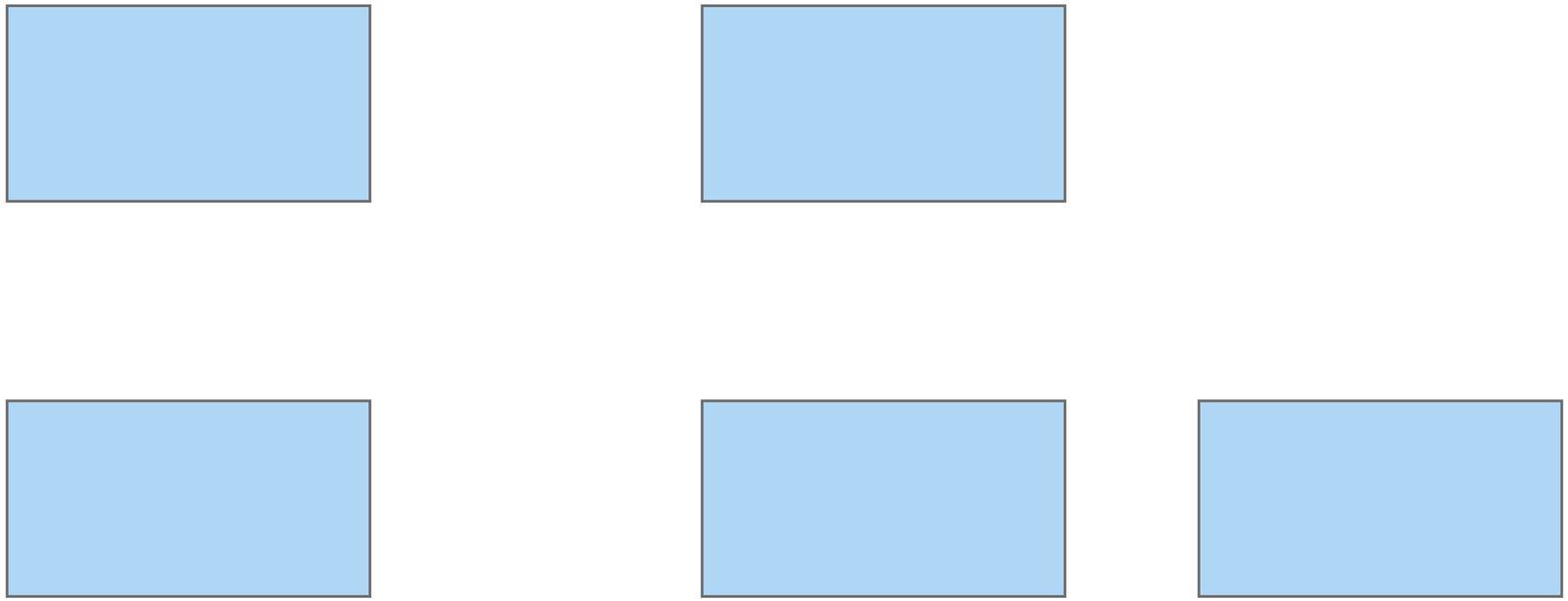
Martin Eigenbrodt | martin.eigenbrodt@innoq.com

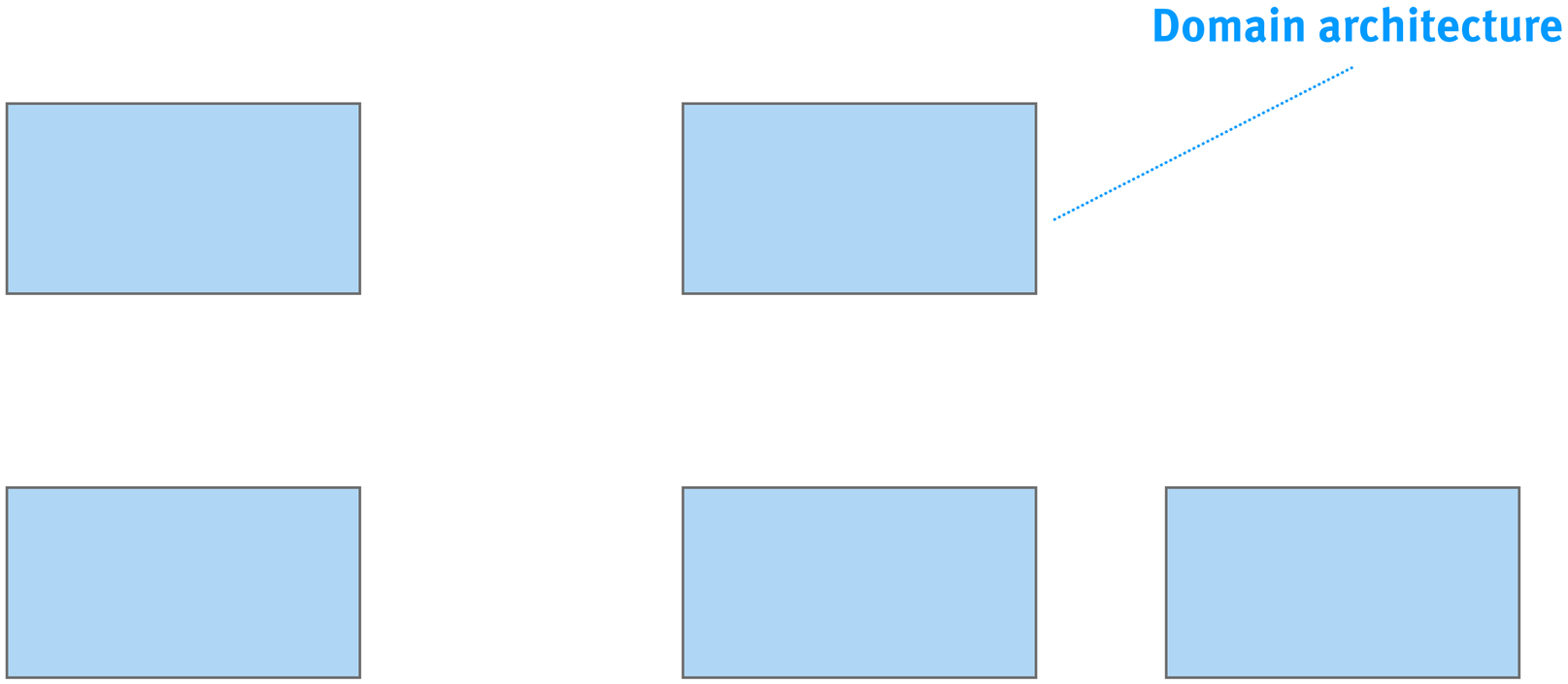
Alexander Heusingfeld | alexander.heusingfeld@innoq.com

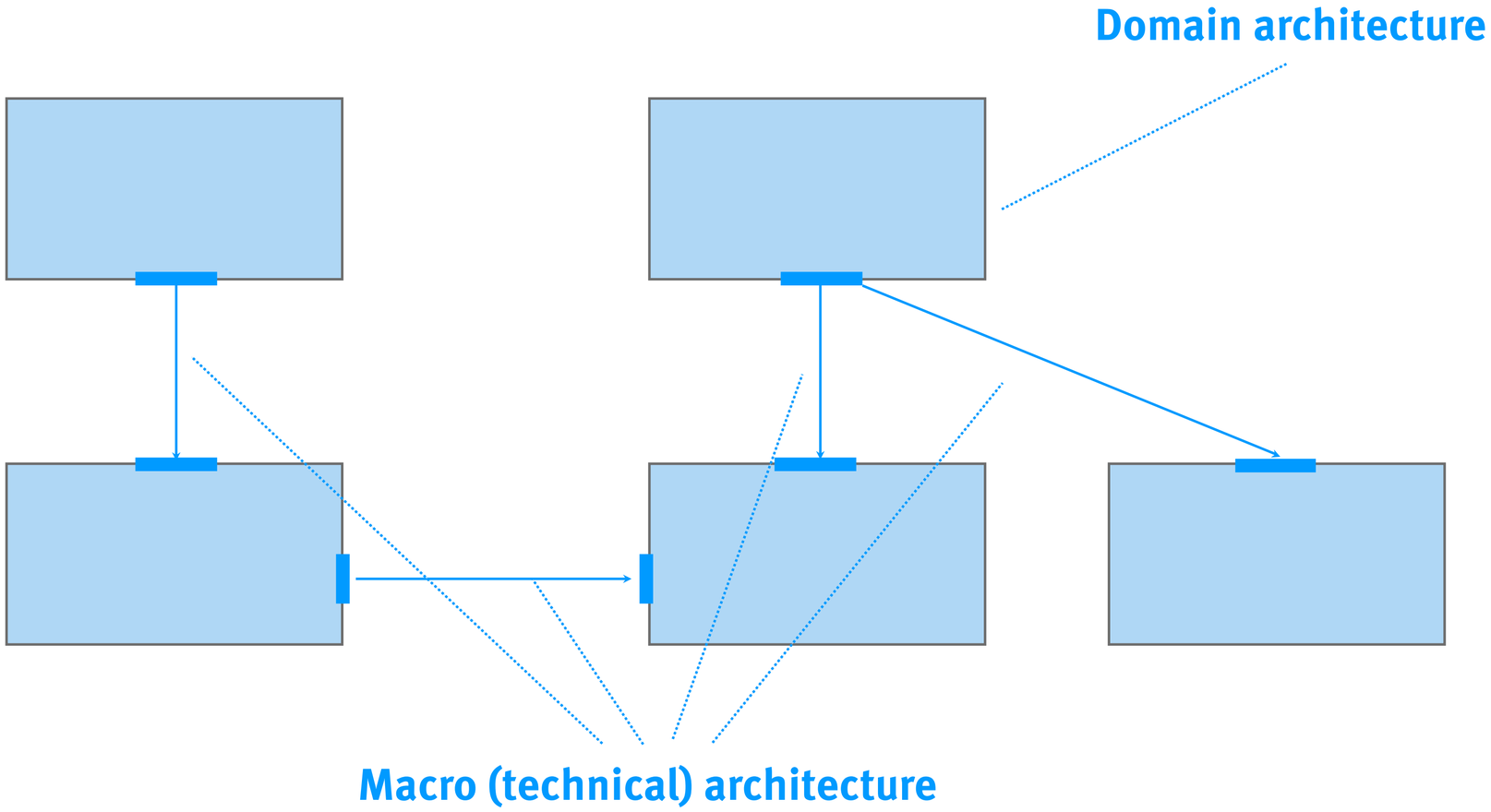
innoQ

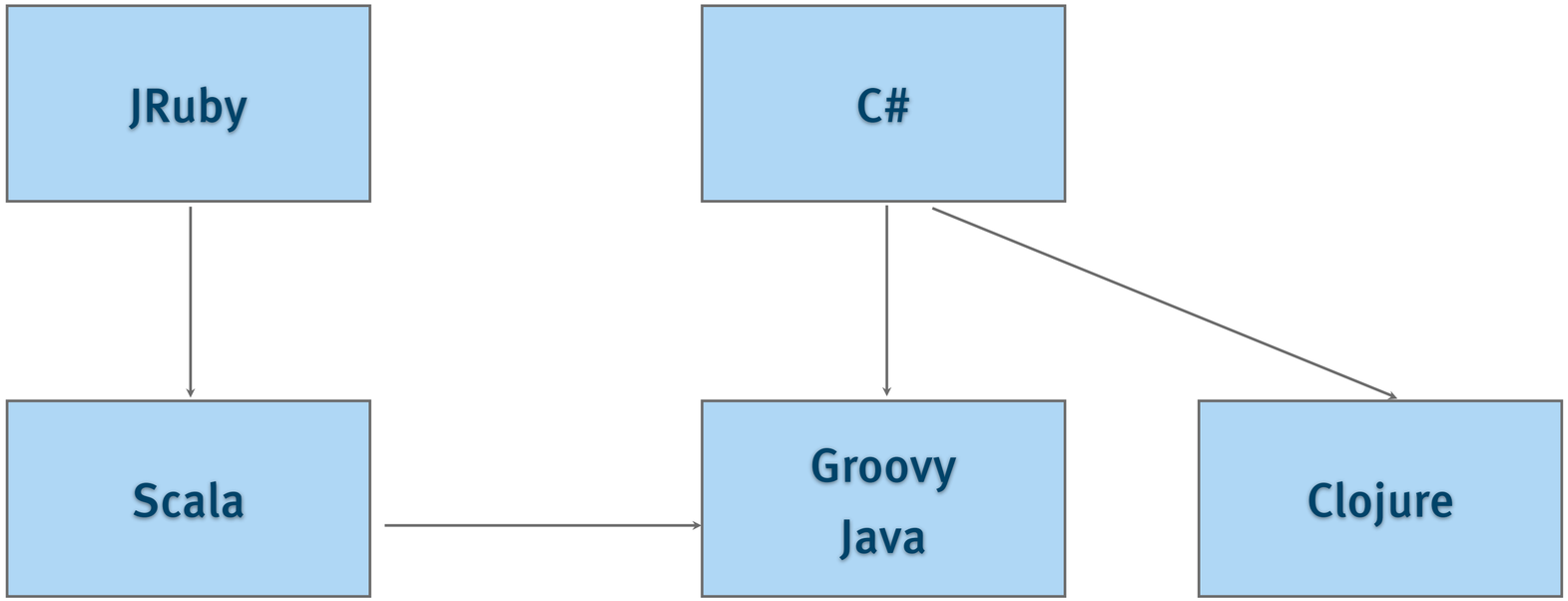
Microservices?

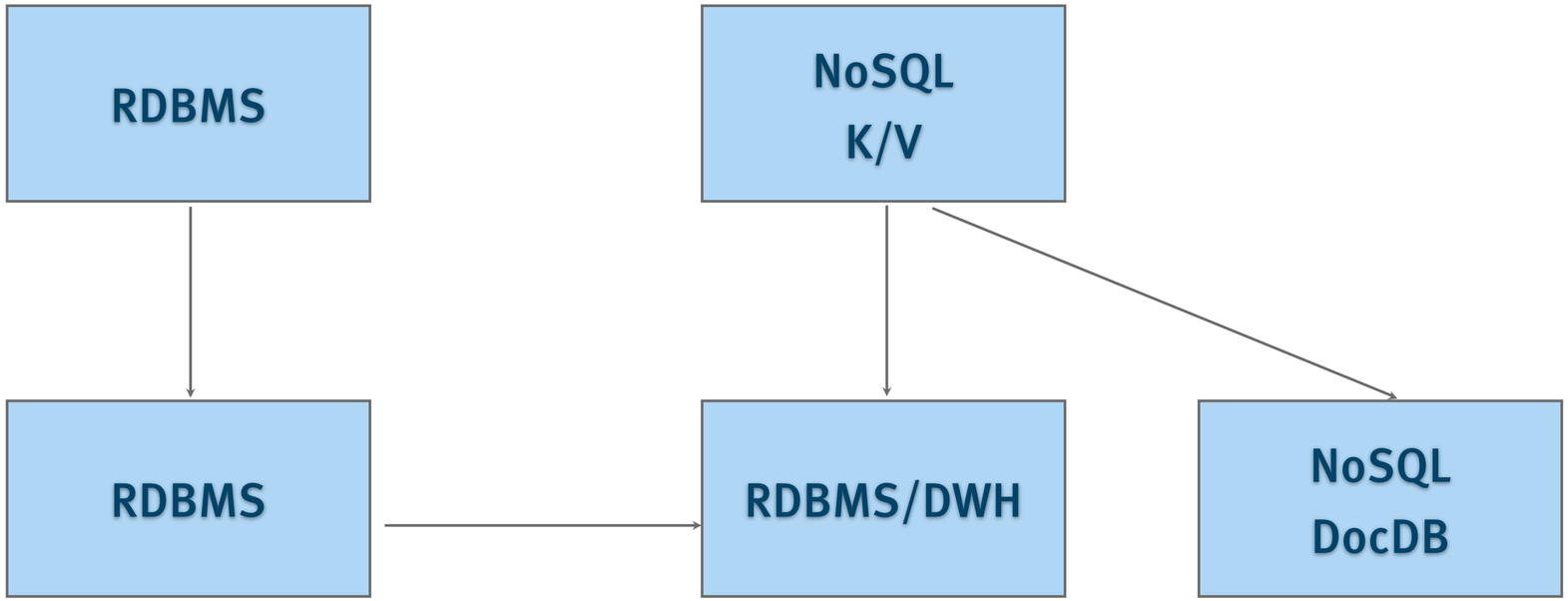
Levels of Architecture

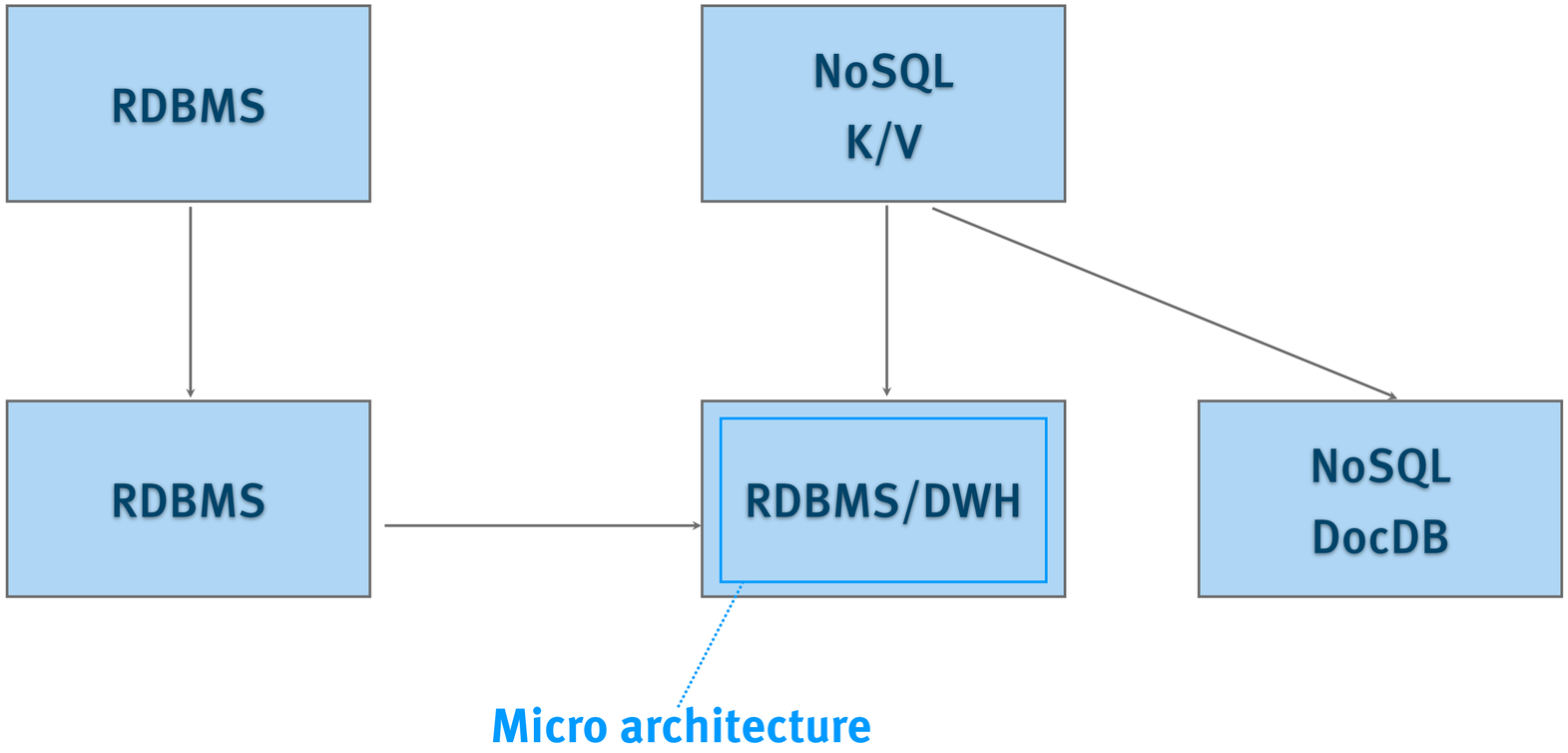



Domain architecture

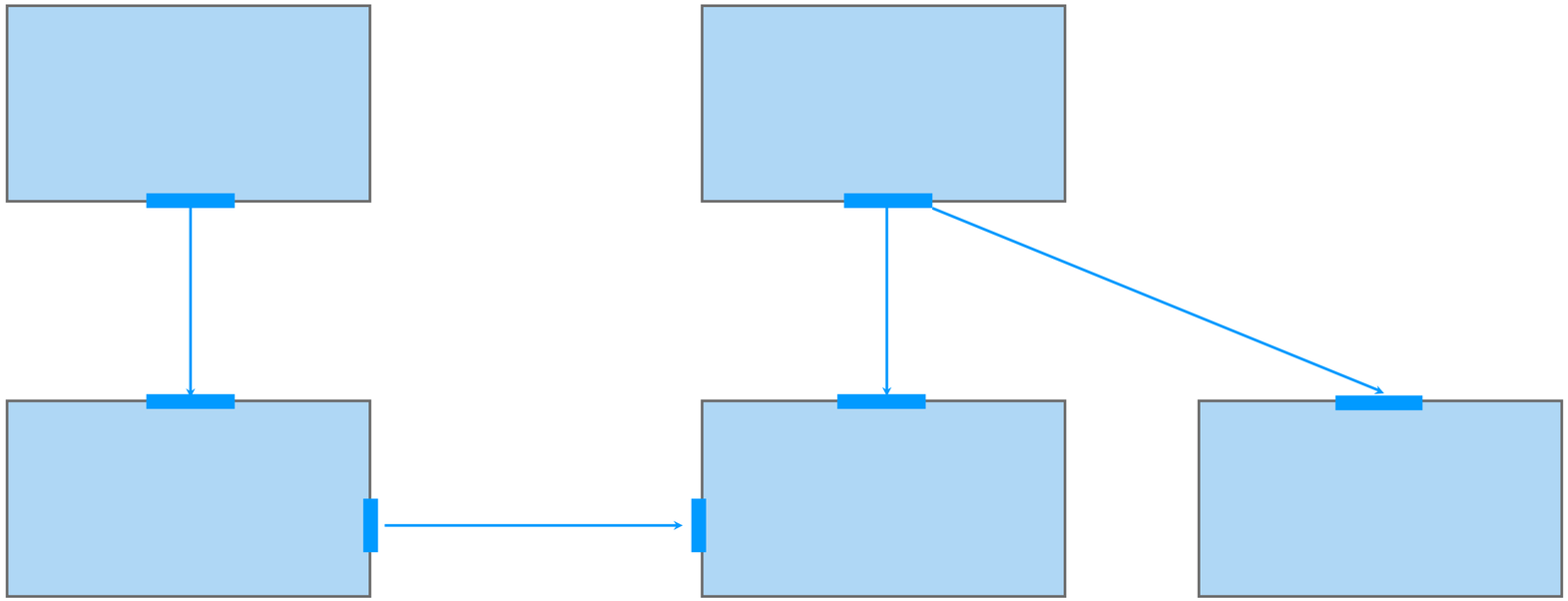

Domain architecture
Macro (technical) architecture


JRuby
C#
Scala
Groovy
Java
Clojure

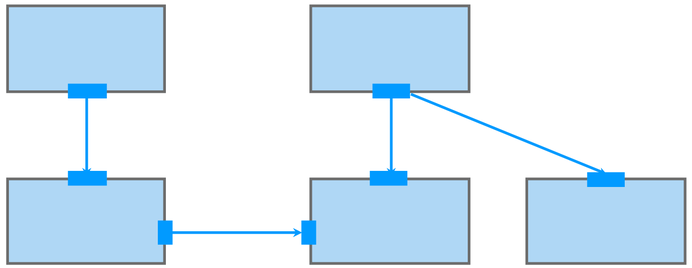

RDBMS
NoSQL
K/V
RDBMS
RDBMS/DWH
NoSQL
DocDB


RDBMS
NoSQL
K/V
RDBMS
RDBMS/DWH
NoSQL
DocDB
Micro architecture

Challenges

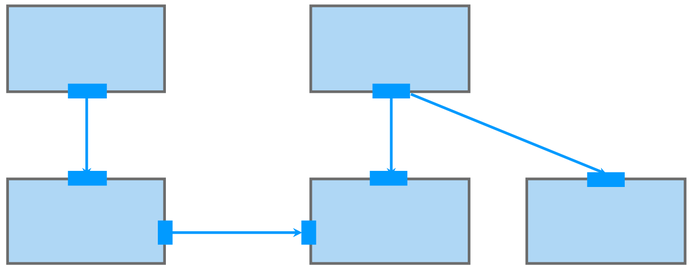


Challenges



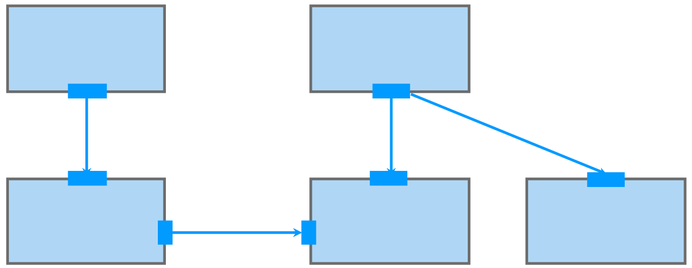
Challenges

- many services to take care of



Challenges

- many services to take care of
- distributed system



Challenges

- distributed configuration
- service registration & discovery
- resilience
- fast, automated deployment
- metrics

Macro- vs. Micro Architecture

Frameworks

Dropwizard

Dropwizard

- Glue Code for well known libraries
- Java

Dropwizard libraries

Dropwizard libraries

- Jetty

Dropwizard libraries

- Jetty
- Jersey

Dropwizard libraries

- Jetty
- Jersey
- Metrics

Dropwizard libraries

- Jetty
- Jersey
- Metrics
- Jackson
- Guava
- Logback
- Hibernate Validator
- Apache Http Client
- JDBI
- Liquibase
- Freemarker & Mustache
- Joda

Spring Boot

and Spring Cloud

Spring Boot

Spring Boot

- convention over configuration approach

Spring Boot

- convention over configuration approach
- Java, Groovy or Scala

Spring Boot

- convention over configuration approach
- Java, Groovy or Scala
- self-contained jar or war

Spring Boot

- convention over configuration approach
- Java, Groovy or Scala
- self-contained jar or war
- tackles dependency-hell via pre-packaging

Spring Cloud

Spring Cloud

- umbrella project for cloud connectors

Spring Cloud

- umbrella project for cloud connectors
- On top of Spring Boot

Spring Cloud

- umbrella project for cloud connectors
- On top of Spring Boot
- config server for distributed configuration

Spring Cloud

- umbrella project for cloud connectors
- On top of Spring Boot
- config server for distributed configuration
- annotations for service-discovery & resilience

Play 2

Play 2

- Java or Scala
- based on Akka
- strong async support

Configuration

Play - Typesafe Config

Play - Typesafe Config

- Config Library used by akka, play and other

Play - Typesafe Config

- Config Library used by akka, play and other
- HOCON - JSON Data Model + syntactic sugar

Play - Typesafe Config

- Config Library used by akka, play and other
- HOCON - JSON Data Model + syntactic sugar
- override via system property

Play - Typesafe Config

- Config Library used by akka, play and other
- HOCON - JSON Data Model + syntactic sugar
- override via system property
- rich merge and include possibilities

Spring Boot

```java
@ComponentScan
@EnableAutoConfiguration
public class OrderApp {

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

Spring Boot

```
@ComponentScan
@EnableAutoConfiguration
public class OrderApp {

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

Spring Boot

```
@ComponentScan
@EnableAutoConfiguration
public class OrderApp {

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

- HTTP resource “/autoconfig” shows all properties

Spring Boot

```
@ComponentScan
@EnableAutoConfiguration
public class OrderApp {

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

- HTTP resource “/autoconfig” shows all properties
- overwrite via application.properties or CLI parameter

Spring Boot

```
@ComponentScan
@EnableAutoConfiguration
public class OrderApp {

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

- HTTP resource “/autoconfig” shows all properties
- overwrite via application.properties or CLI parameter
- configuration in git? -> Check spring-cloud configserver

Http Client

Dropwizard

```
public Product resolveProduct(String url) {
  Product product = client.resource(url)
        .accept(MediaType.APPLICATION_JSON).get(Product.class);
  return product;
}
```

Spring Boot

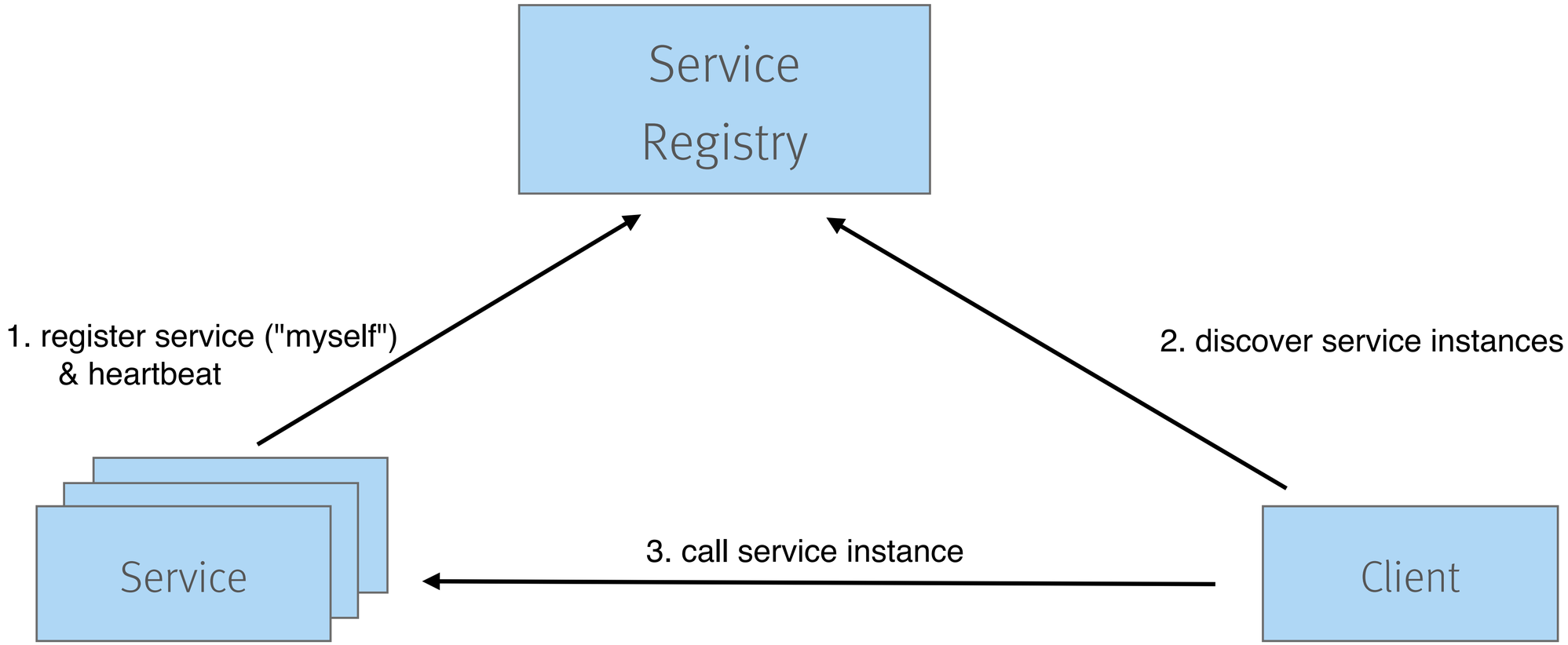
```
public Product resolveProduct(String url) {
 return restTemplate.getForEntity(url, Product.class);
}
```

Play

```
WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }.recover { case e => List() }
```

Service Discovery

Service Discovery



Spring Cloud

```
@ComponentScan
@EnableAutoConfiguration
@EnableDiscoveryClient
public class OrdersApp {

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

Spring Cloud

```
@ComponentScan
@EnableAutoConfiguration
@EnableDiscoveryClient
public class OrdersApp {

    public static void main(Strir
        SpringApplication.run(Orc
    }
}
```

spring X

HOME LAST 1000 SINCE STARTUP

System Status

Environment	
Data center	

Current time	2014-11-11T15:11:21 +0100
Uptime	00:15
Lease expiration enabled	true
Renews threshold	3
Renews (last min)	4

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
ORDERS	n/a (1)	(1)	UP (1) - ahembp15.monheim.office.innoq.com

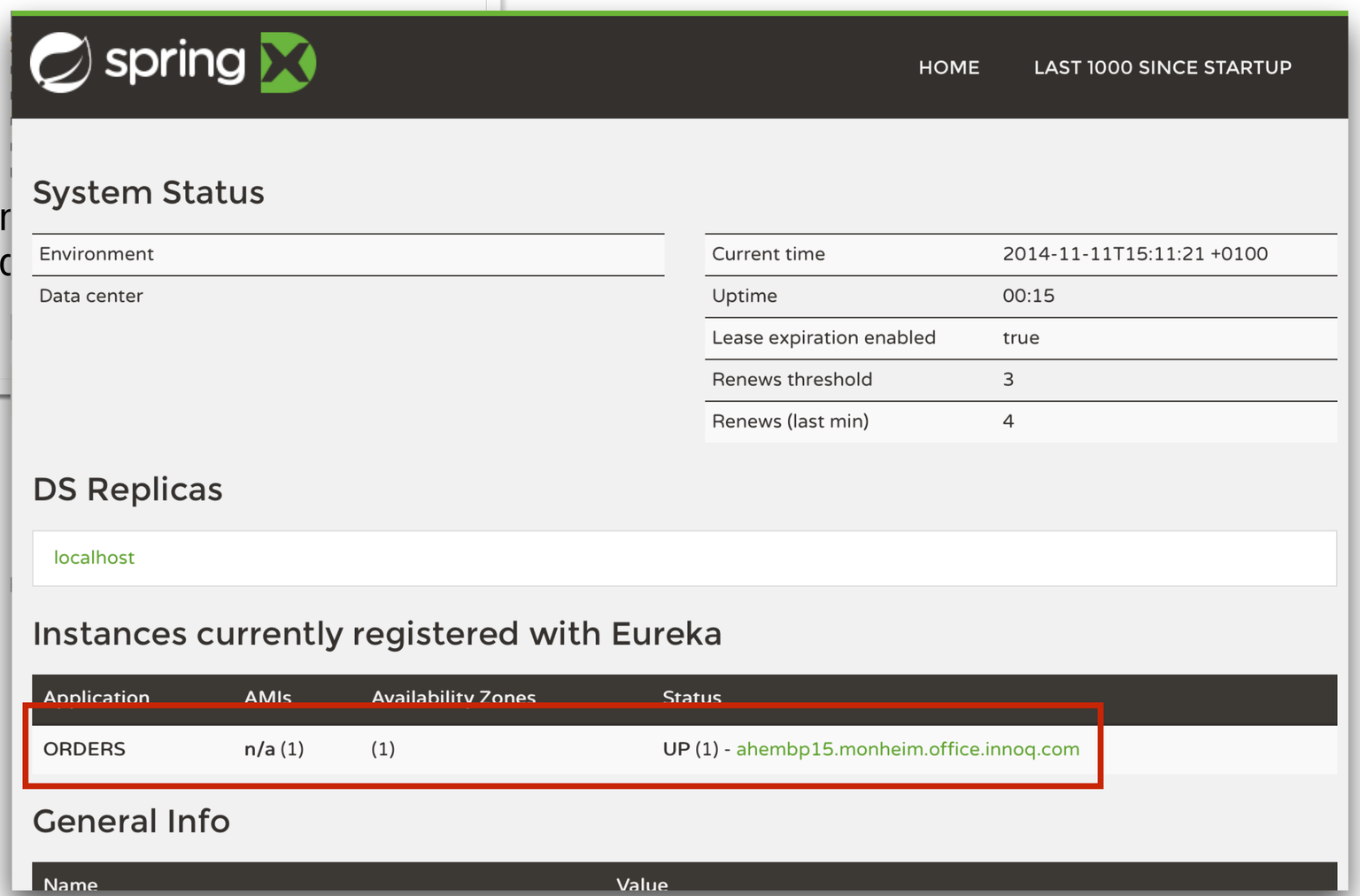
General Info

Name	Value

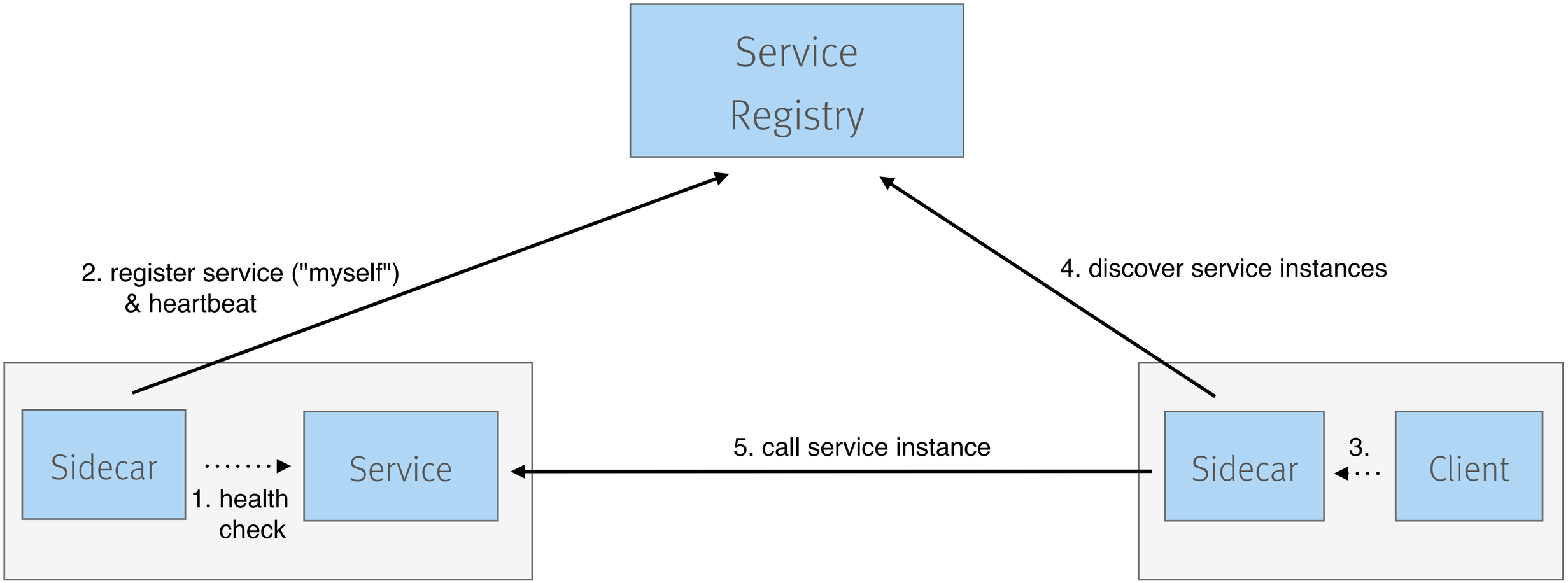
Spring Cloud

```
@ComponentScan
@EnableAutoConfiguration
@EnableDiscoveryClient
public class OrdersApp {

    public static void main(Strir
        SpringApplication.run(Orc
    }
}
```



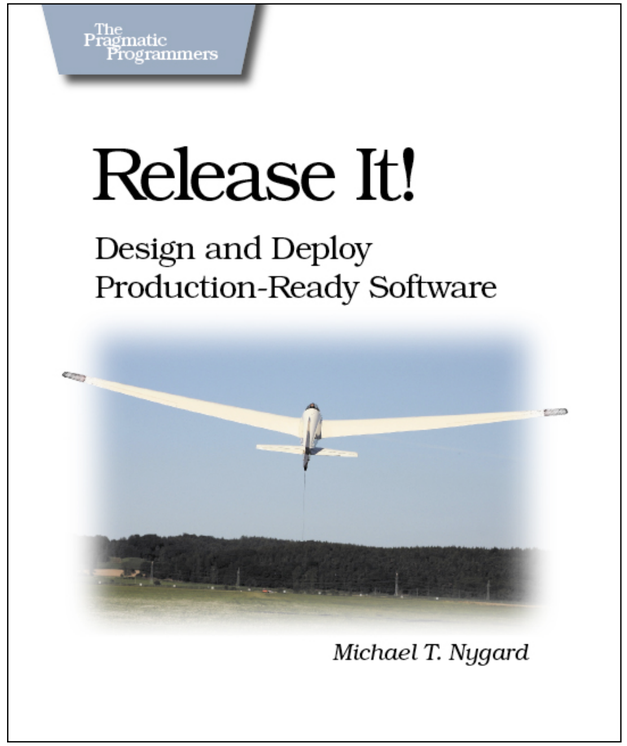
Service Discovery with Sidecar



Resilience

Resilience

- isolate Failure
- apply graceful degradation
- be responsive in case of failure



Request →  *closed* → service

http://en.wikipedia.org/wiki/Circuit_breaker

Request →  *closed* → service

Request → ←  *open* service

http://en.wikipedia.org/wiki/Circuit_breaker

Request →  *closed* → service

Request ⇄  *open* service

Request ⇄  *half-open* ⇢ service

http://en.wikipedia.org/wiki/Circuit_breaker

Hystrix

Defend Your App

- Provides Command-oriented Integration of Services
- Introduces Circuit Breaker, Bulkheads and Isolation
- Decouples from Service-dependencies
- Provides metrics-facility to protect from failures

Hystrix & Dropwizard

```
public class CommandInDropwizard extends TenacityCommand<Product> {


    @Override
    protected Product run() throws Exception {

      Product product = client.resource(url)
        .accept(MediaType.APPLICATION_JSON).get(Product.class);
     return product;
    }

    protected Product getFallback() {
        return FALLBACK_PRODUCT
    }
}
```

Hystrix & Dropwizard

```
public class CommandInDropwizard extends TenacityCommand<Product> {


    @Override
    protected Product run() throws Exception {

      Product product = client.resource(url)
        .accept(MediaType.APPLICATION_JSON).get(Product.class);
     return product;
    }

    protected Product getFallback() {
        return FALLBACK_PRODUCT
    }
}
```

Hystrix & Dropwizard

```
public class CommandInDropwizard extends TenacityCommand<Product> {


    @Override
    protected Product run() throws Exception {

      Product product = client.resource(url)
        .accept(MediaType.APPLICATION_JSON).get(Product.class);
     return product;
    }

    protected Product getFallback() {
        return FALLBACK_PRODUCT
    }
}
```

Hystrix & Dropwizard

```
public class CommandInDropwizard extends TenacityCommand<Product> {


    @Override
    protected Product run() throws Exception {

      Product product = client.resource(url)
        .accept(MediaType.APPLICATION_JSON).get(Product.class);
     return product;
    }

    protected Product getFallback() {
        return FALLBACK_PRODUCT
    }
}
```

Hystrix & Dropwizard

```
ResolveProductCommand command = new ResolveProductCommand(client, url);
Product product = command.execute();
```

Spring Cloud Hystrix

```
@HystrixCommand(fallbackMethod = “fallbackProduct")
private Pair<String, ResponseEntity<Product>> resolveProduct(String productUri) {
    final RestTemplate restTemplate = new RestTemplate();
    return new Pair(productUri, restTemplate.getForEntity(productUri, Product.class));
}


private Pair<String, ResponseEntity<Product>> fallbackProduct(String productUri) {
    final Product product = new Product(productUri, null, BigDecimal.ZERO);
    final ResponseEntity<Product> response = new ResponseEntity<Product>(product, PARTIAL_CONTENT);
    return new Pair(productUri, response);
}
```

Spring Cloud Hystrix

auto-wrapped with command!

```
@HystrixCommand(fallbackMethod = “fallbackProduct")
private Pair<String, ResponseEntity<Product>> resolveProduct(String productUri) {
    final RestTemplate restTemplate = new RestTemplate();
    return new Pair(productUri, restTemplate.getForEntity(productUri, Product.class));
}


private Pair<String, ResponseEntity<Product>> fallbackProduct(String productUri) {
    final Product product = new Product(productUri, null, BigDecimal.ZERO);
    final ResponseEntity<Product> response = new ResponseEntity<Product>(product, PARTIAL_CONTENT);
    return new Pair(productUri, response);
}
```

Spring Cloud Hystrix

```
@HystrixCommand(fallbackMethod = “fallbackProduct")
private Pair<String, ResponseEntity<Product>> resolveProduct(String productUri) {
    final RestTemplate restTemplate = new RestTemplate();
    return new Pair(productUri, restTemplate.getForEntity(productUri, Product.class));
}


private Pair<String, ResponseEntity<Product>> fallbackProduct(String productUri) {
    final Product product = new Product(productUri, null, BigDecimal.ZERO);
    final ResponseEntity<Product> response = new ResponseEntity<Product>(product, PARTIAL_CONTENT);
    return new Pair(productUri, response);
}
```

Spring Cloud Hystrix

```
@HystrixCommand(fallbackMethod = “fallbackProduct")
private Pair<String, ResponseEntity<Product>> resolveProduct(String productUri) {
    final RestTemplate restTemplate = new RestTemplate();
    return new Pair(productUri, restTemplate.getForEntity(productUri, Product.class));
}


private Pair<String, ResponseEntity<Product>> fallbackProduct(String productUri) {
    final Product product = new Product(productUri, null, BigDecimal.ZERO);
    final ResponseEntity<Product> response = new ResponseEntity<Product>(product, PARTIAL_CONTENT);
    return new Pair(productUri, response);
}
```

Spring Cloud Hystrix

method reference!

```
@HystrixCommand(fallbackMethod = "fallbackProduct")
private Pair<String, ResponseEntity<Product>> resolveProduct(String productUri) {
    final RestTemplate restTemplate = new RestTemplate();
    return new Pair(productUri, restTemplate.getForEntity(productUri, Product.class));
}


private Pair<String, ResponseEntity<Product>> fallbackProduct(String productUri) {
    final Product product = new Product(productUri, null, BigDecimal.ZERO);
    final ResponseEntity<Product> response = new ResponseEntity<Product>(product, PARTIAL_CONTENT);
    return new Pair(productUri, response);
}
```

Play - Circuit Breaker

```
val apiUrl = "..."
val breaker =
  CircuitBreaker(Akka.system().scheduler,
    maxFailures = 5,
    callTimeout = 2.seconds,
    resetTimeout = 1.minute)

def getBestseller : Future[List[Bestseller]] = {
  breaker.withCircuitBreaker(
    WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }).recover { case e => List() }
}
```

Play - Circuit Breaker

```
val apiUrl = "..."
val breaker =
  CircuitBreaker(Akka.system().scheduler,
    maxFailures = 5,
    callTimeout = 2.seconds,
    resetTimeout = 1.minute)

def getBestseller : Future[List[Bestseller]] = {
  breaker.withCircuitBreaker(
    WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }).recover { case e => List() }
}
```

Play - Circuit Breaker

```
val apiUrl = "..."
val breaker =
  CircuitBreaker(Akka.system().scheduler,
    maxFailures = 5,
    callTimeout = 2.seconds,
    resetTimeout = 1.minute)

def getBestseller : Future[List[Bestseller]] = {
  breaker.withCircuitBreaker(
    WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }) recover { case e => List() }
}
```

Play - Circuit Breaker

```
val apiUrl = "..."
val breaker =
  CircuitBreaker(Akka.system().scheduler,
    maxFailures = 5,
    callTimeout = 2.seconds,
    resetTimeout = 1.minute)

def getBestseller : Future[List[Bestseller]] = {
  breaker.withCircuitBreaker(
    WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }).recover { case e => List() }
}
```

Play - Circuit Breaker

```
val apiUrl = "..."
val breaker =
  CircuitBreaker(Akka.system().scheduler,
    maxFailures = 5,
    callTimeout = 2.seconds,
    resetTimeout = 1.minute)

def getBestseller : Future[List[Bestseller]] = {
  breaker.withCircuitBreaker(
    WS.url(apiUrl).get.map {
      response => response.json.as[List[Bestseller]]
    }).recover { case e => List() }
}
```

Deployment

Spring Boot - Packaging

```
./gradlew build
./gradlew distZip
```

Spring Boot - Packaging

executable JAR

```
./gradlew build
./gradlew distZip
```

Spring Boot - Packaging

executable JAR

```
./gradlew build
./gradlew distZip
```

ZIP + shell-script

Play - Packaging

```
sbt dist
sbt debian:packageBin
sbt rpm:packageBin
```

Metrics

Dropwizard

- “Metrics” Integrated with Dropwizard
- @Timed on Resources
- HTTP Client is already instrumented
- JVM Data

```
"org.apache.http.client.HttpClient.cart.get-requests": {
    "count": 11,
    "max": 0.062107,
    "mean": 0.013355909090909092,
    "min": 0.005750000000000001,
    "p50": 0.009454,
    "p75": 0.010427,
    "p95": 0.062107,
    "p98": 0.062107,
    "p99": 0.062107,
    "p999": 0.062107,
    "stddev": 0.016285873488729705,
    "m15_rate": 0,
    "m1_rate": 0,
    "m5_rate": 0,
    "mean_rate": 2.9714422786532126,
    "duration_units": "seconds",
    "rate_units": "calls/second"
}
```

```
"cart.resources.ShoppingCartResource.shoppingCart": {
    "count": 22,
    "max": 0.136162,
    "mean": 0.01208109090909091,
    "min": 0.00093,
    "p50": 0.008174500000000001,
    "p75": 0.011782250000000001,
    "p95": 0.11783499999999976,
    "p98": 0.136162,
    "p99": 0.136162,
    "p999": 0.136162,
    "stddev": 0.02813530239821426,
    "m15_rate": 1.8524577712890011,
    "m1_rate": 0.18057796798879996,
    "m5_rate": 1.315746847992022,
    "mean_rate": 0.133050618509084,
    "duration_units": "seconds",
    "rate_units": "calls/second"
}
```

Dropwizard Metrics

- Exposed over HTTP (as Json)
- Exposed as jmx
- Others available: stdout, csv, slf4j, ganglia, graphite

Spring Boot Metrics

- Prepackaged Spring Boot starter module
- enables HTTP resources for metrics
- configurable via application.properties

http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#production-ready

Spring Boot Metrics

Using a counter metric in your Java code...

```
counterService.increment("checkouts.withproducts." + productUris.size());
```

...will display it in the /metrics JSON

```
GET /metrics HTTP/1.1
{
    "counter.checkouts.withproducts.3": 4,
    ...
}
```

Summary

Summary

Modularize into independent, self-contained systems

Summary

Modularize into independent, self-contained systems

Separate micro and macro architectures

Summary

Modularize into independent, self-contained systems

Separate micro and macro architectures

Strike a balance between control and decentralization

Summary

Modularize into independent, self-contained systems

Separate micro and macro architectures

Strike a balance between control and decentralization

MicroServices aren’t micro!

Summary

Modularize into independent, self-contained systems

Separate micro and macro architectures

Strike a balance between control and decentralization

MicroServices aren’t micro!

frameworks can’t solve all your problems

Thank you!

Questions?

Comments?

Martin Eigenbrodt | eigenbrodtm
martin.eigenbrodt@innoq.com

Alexander Heusingfeld | goldstift
alexander.heusingfeld@innoq.com

https://www.innoq.com/en/talks/2015/02/microservices-jvm-applications-talk

innoQ
www.innoq.com

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Robert-Bosch-Straße 7
64293 Darmstadt
Germany
Phone: +49 2173 3366-0

Radlkoferstraße 2
D-81373 München
Germany
Telefon +49 (0) 89 741185-270

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116