INDEX

SAN FRANCISCO

Discover. Collaborate. Deploy.

What is a Cloud Native application, anyway?

Erin Schnabel

@ebullientworks

Cloud Native

An **application architecture** designed to leverage the **strengths** and accommodate the **challenges** of a **standardized** cloud environment, including concepts such as **elastic** scaling, **immutable** deployment, **disposable** instances, and **less predictable infrastructure**.

Cloud Native applications

CLOUD NATIVE
COMPUTING FOUNDATION

- Container packaged
  - Resource isolation
  - Simplified operations
- Dynamically managed / orchestrated
- Microservice oriented
  - Loosely coupled
  - Declared external dependencies

Twelve Factor Applications

- Methodology for building SaaS applications

- The Twelve Factors can be applied to applications
  - In any programming language
  - With any backing services (or cloud provider..)

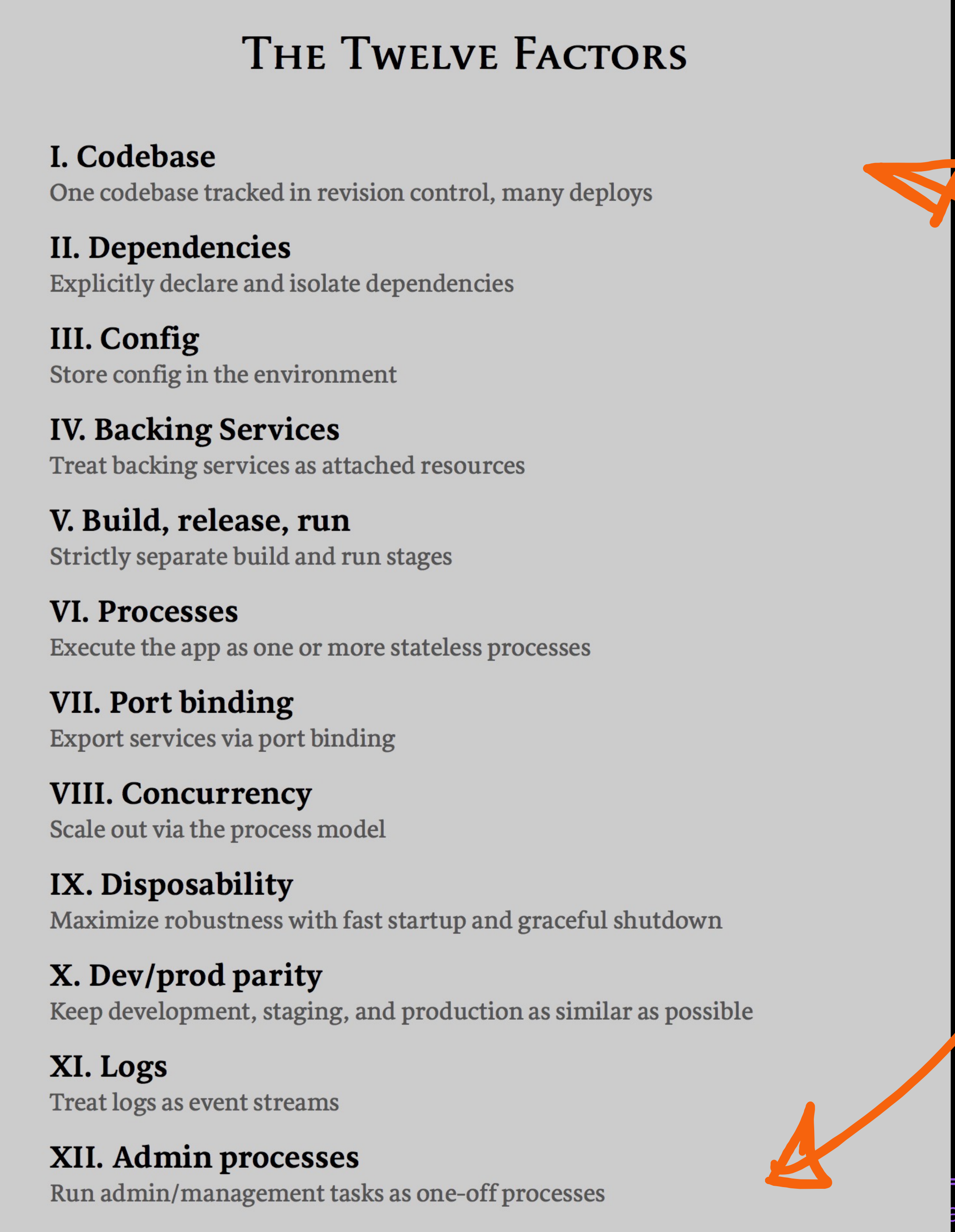
- http://12factor.net/

Key characteristics of 12 Factor apps

- Use **declarative formats** for setup automation
- Have a **clean contract** with the underlying OS

- **Minimum divergence** between **development and test** environments

- **Can scale up** without significant work
- Maintained in a **continuous delivery** pipeline



The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

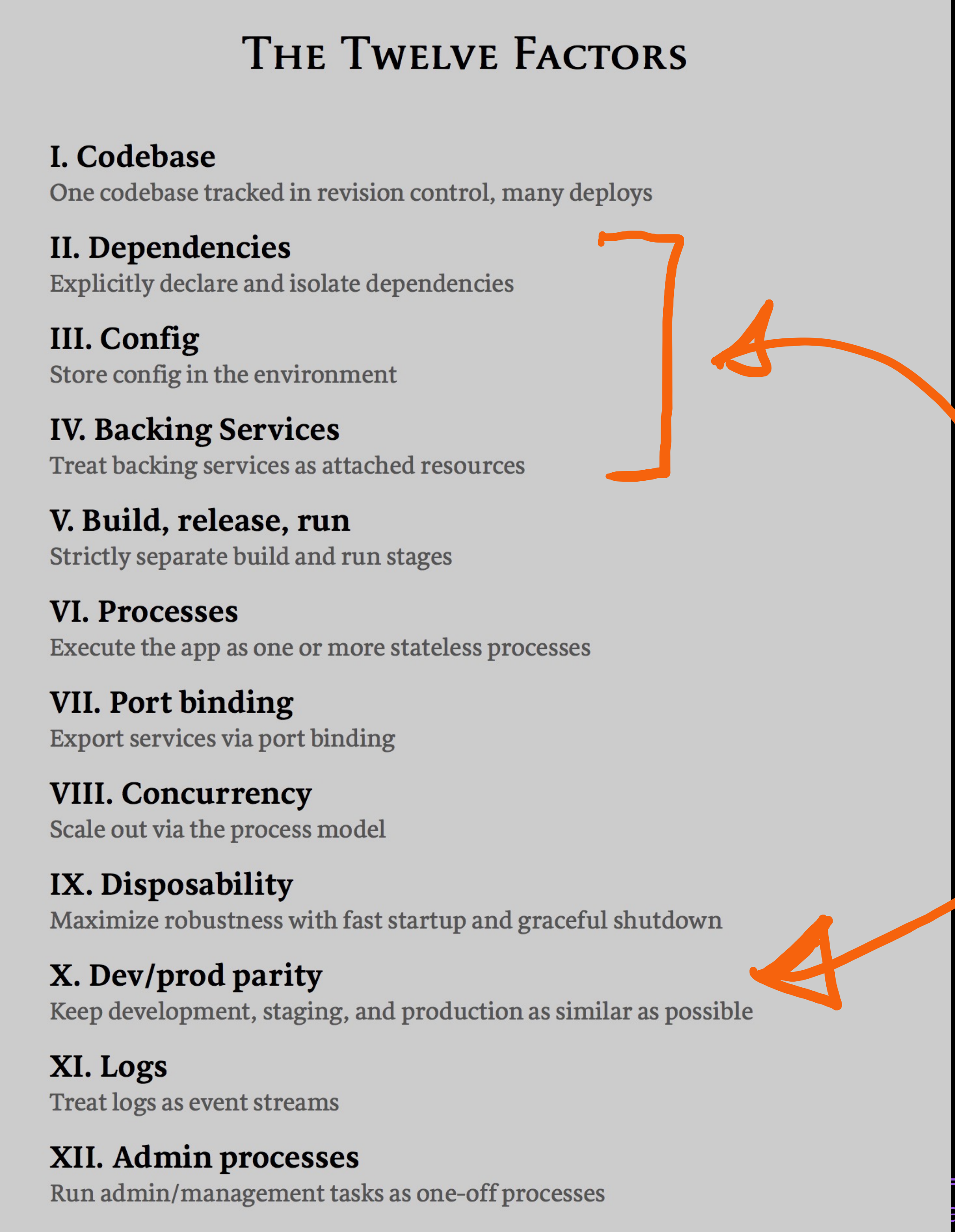
X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

① no code living on someone's laptop

② Infrastructure as code



The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

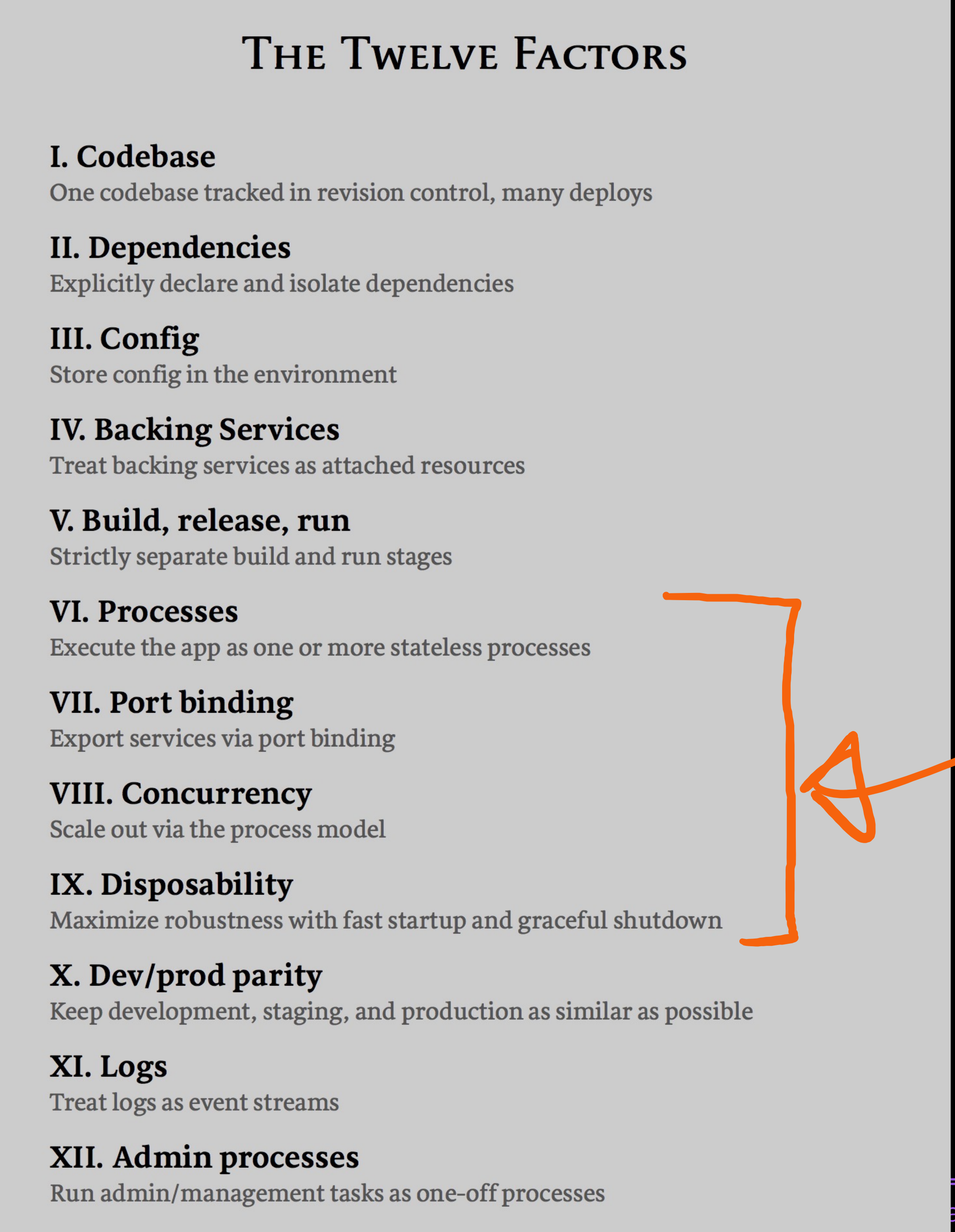
IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

portability



The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

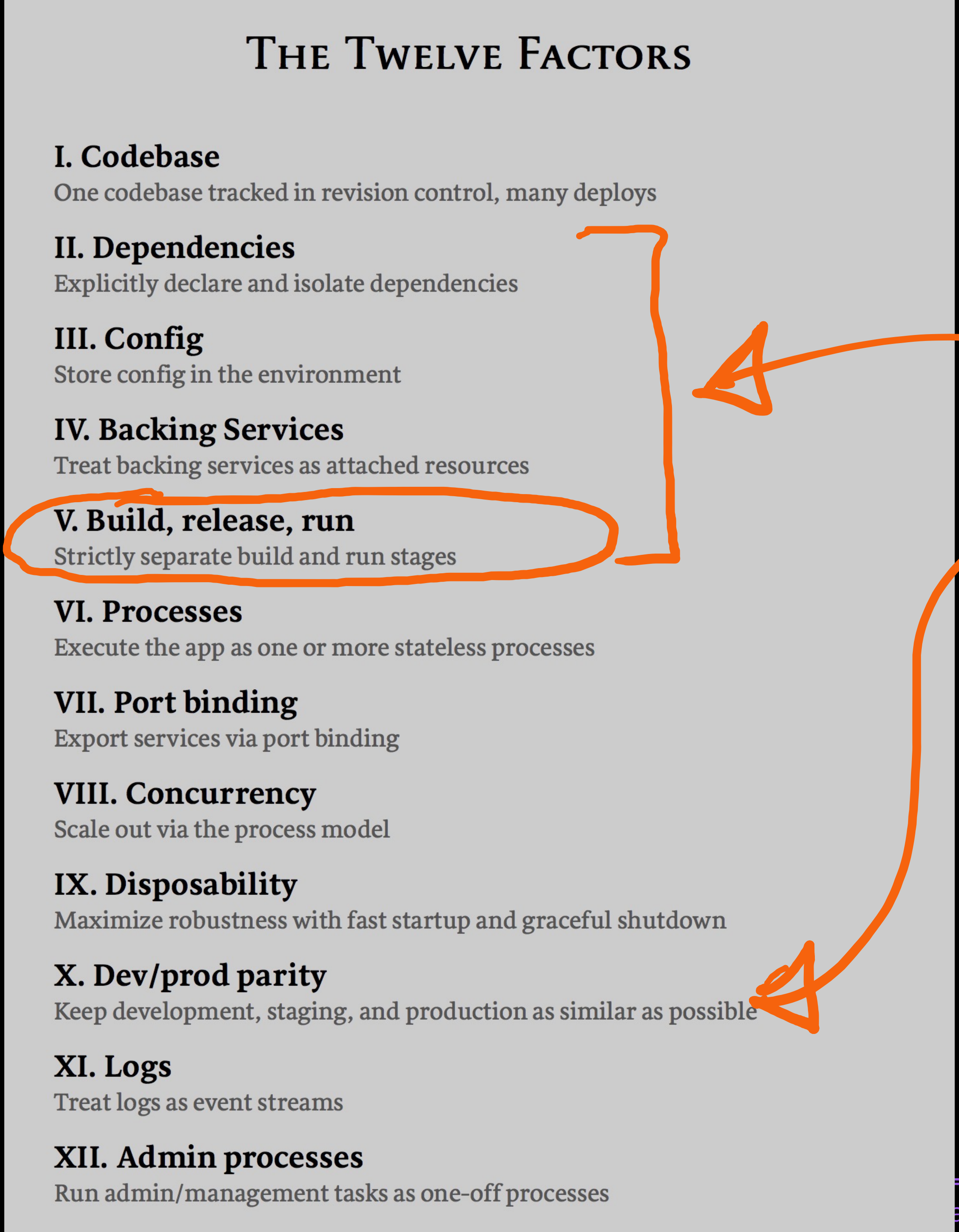
IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

scalability



The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

devOps

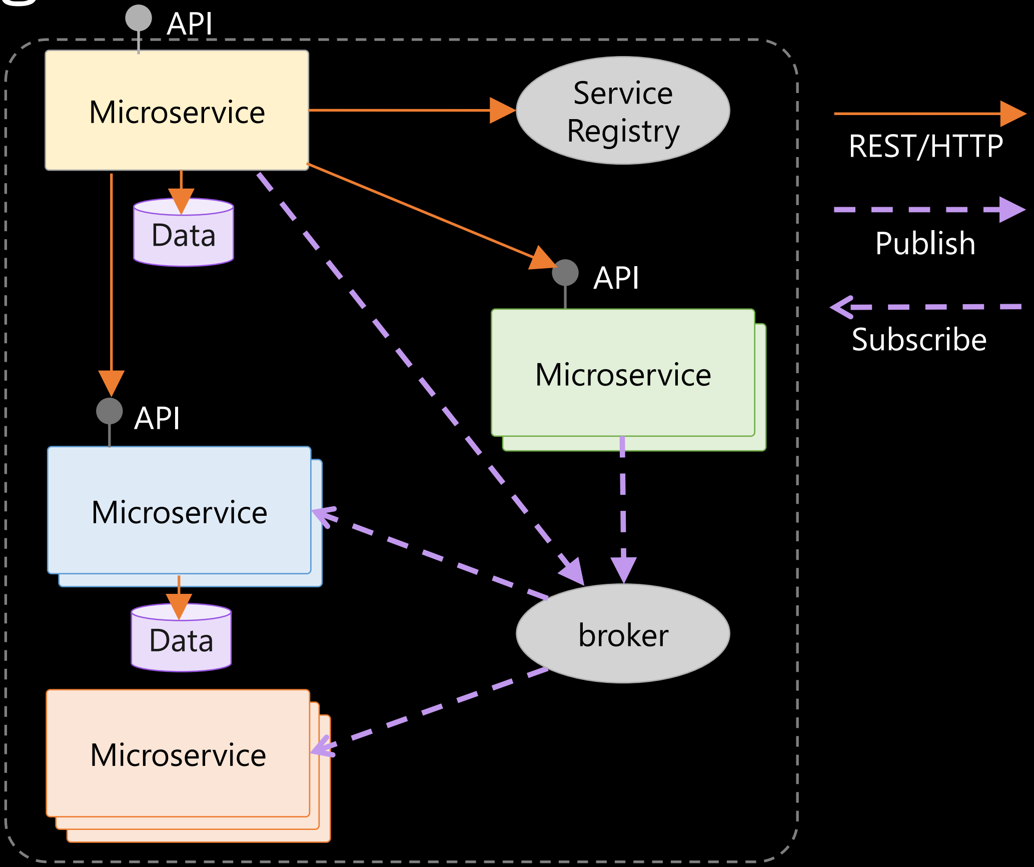
- immutable artifacts

- reproducible builds

Microservices are used to…

- compose a complex application using
  - “small”
  - independent (autonomous)
  - replaceable
  - processes

- that communicate via
  - language-agnostic APIs



Essential characteristics - Services

Autonomy
&
Independence

- Encapsulated by API
  - Language-agnostic protocols
  - Replaceable
- Decentralized
  - Data (eventual consistency)
  - Security (zones)

Fallacies of distributed computing

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn’t change
- There is one administrator
- Transport cost is zero
- The network is homogenous

-- L Peter Deutsch, 1994

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Essential characteristics - Services

Resilient

- Fault tolerant
  - Fail *fast*, gracefully
  - Expect rubbish
  - Fallback: retry vs. cached data
- Prevent cascading failures
  - Timeouts, Fallbacks
  - Circuit Breakers / Bulkheads

Essential characteristics - System

Automated

- Provisioning / Deployment
  - Zero-downtime upgrades
- Load balancing / Scaling
- Health Management
  - Cattle not Pets
- Real-time Monitoring
  - Logging / Metrics

“Twitter microservices map looks just like the Netflix one. “We called this the ‘Death Star’ diagram”

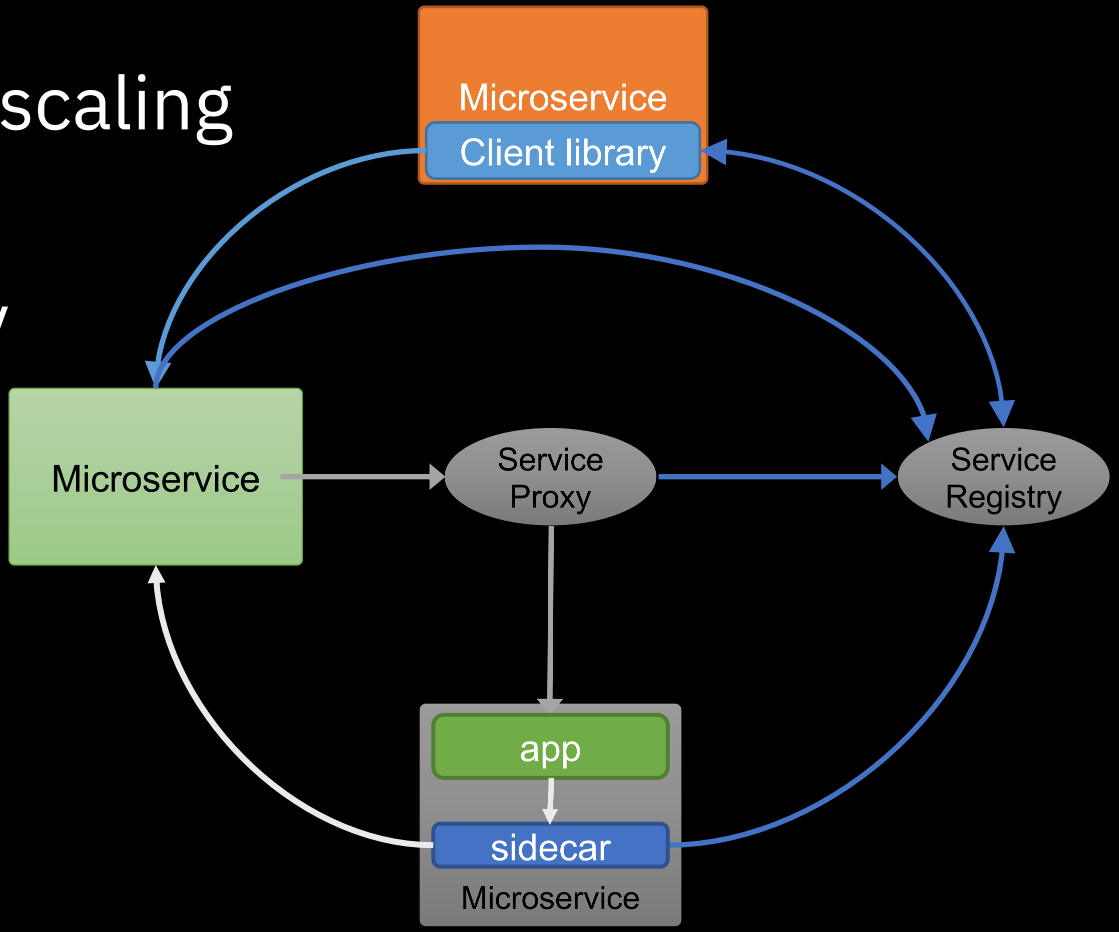
— Adrian Cockcroft via twitter

https://twitter.com/adrianco/status/441883572618948608

Service registration and discovery

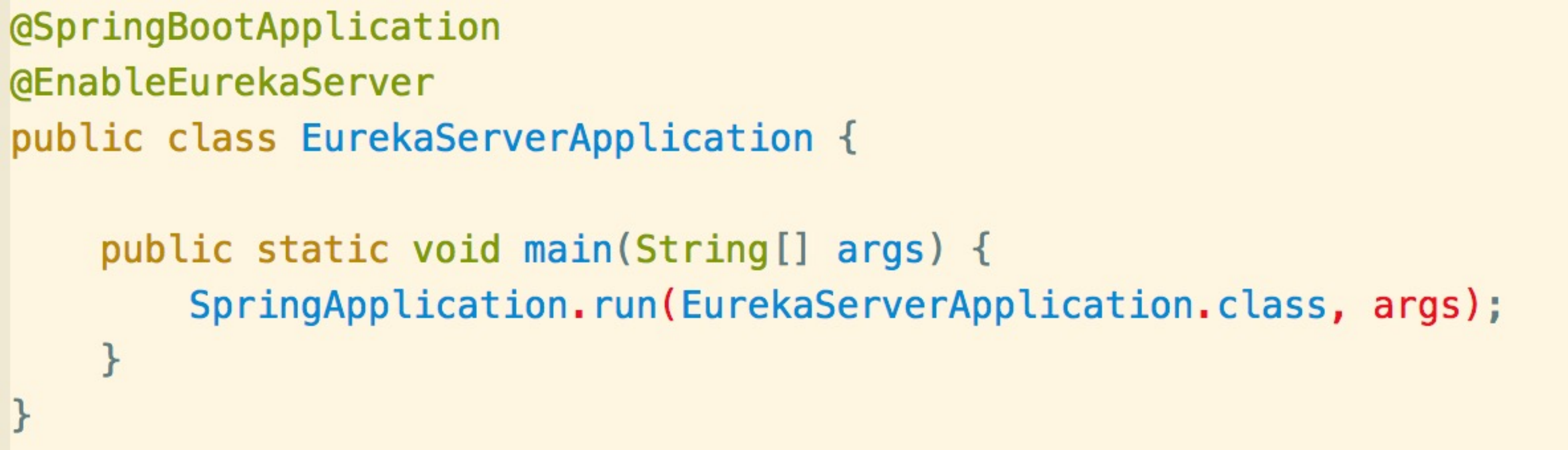
- Required for load balancing and scaling
- Services need to find each other
- Environment changes constantly

- Client-side or server-side?
- Client library, sidecar, or proxy?



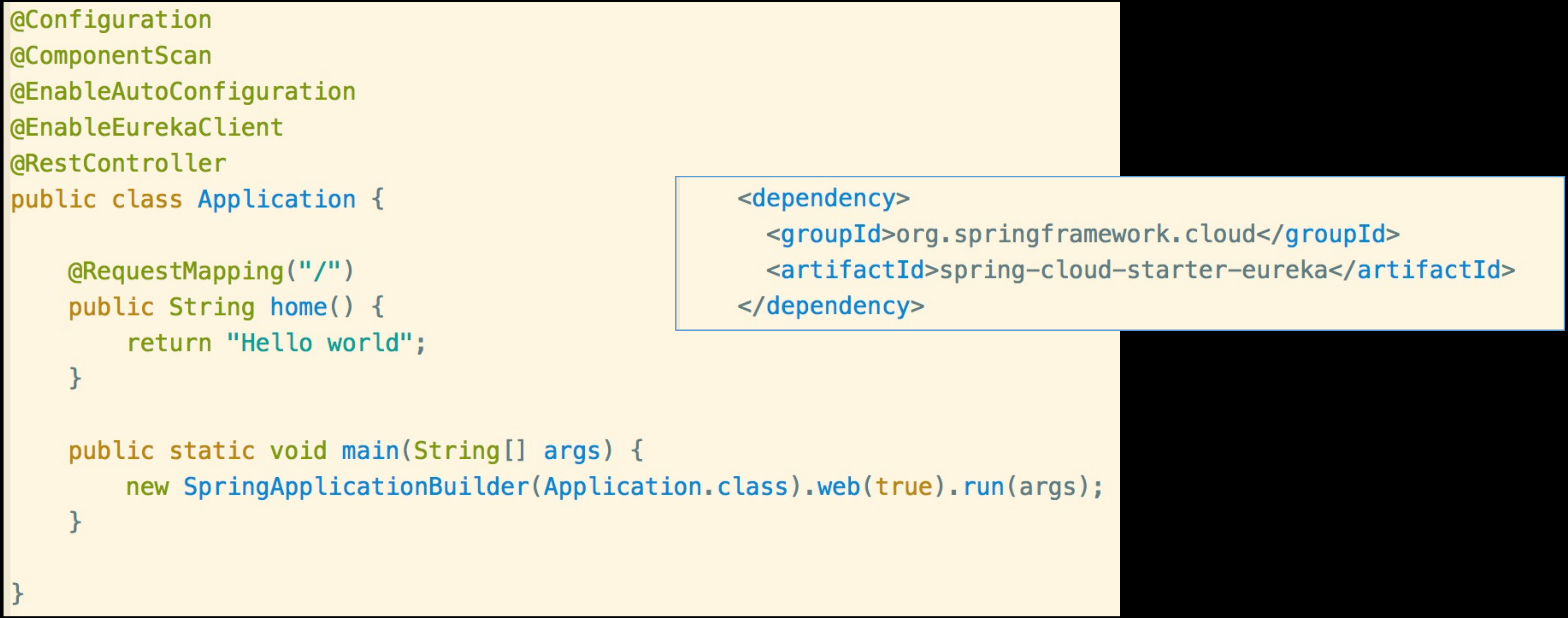
Service Registry and Discovery: Eureka

- Netflix Eureka
  - Stand-alone / self-contained service registry
  - HA, zones, regions… or stand-alone



```java
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApplication {

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

Spring Cloud: Service Registration



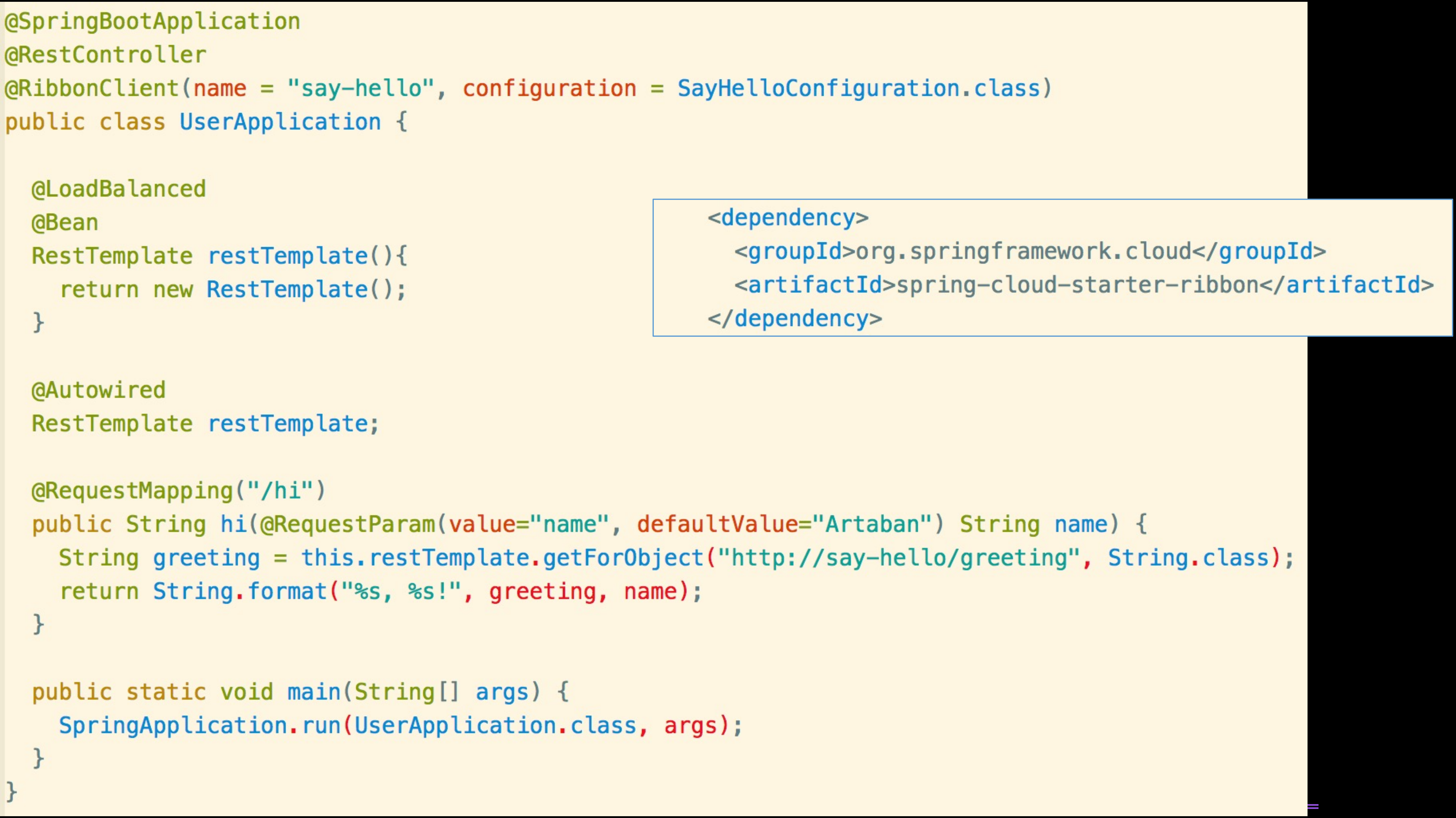
```java
@Configuration
@ComponentScan
@EnableAutoConfiguration
@EnableEurekaClient
@RestController
public class Application {

    @RequestMapping("/")
    public String home() {
        return "Hello world";
    }

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

}
```

```xml
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-eureka</artifactId>
    </dependency>
```

Netflix Ribbon

- Client-side: load balancer integrated in the client
- Rule-based load balancing
  - round robin, response time weighted, random load balancing
  - More via plugins
- ribbon-eureka uses Netflix Eureka for service discovery
- Integrated with Spring Cloud

- Maintenance mode



```java
@SpringBootApplication
@RestController
@RibbonClient(name = "say-hello", configuration = SayHelloConfiguration.class)
public class UserApplication {

  @LoadBalanced
  @Bean
  RestTemplate restTemplate(){
    return new RestTemplate();
  }

  @Autowired
  RestTemplate restTemplate;

  @RequestMapping("/hi")
  public String hi(@RequestParam(value="name", defaultValue="Artaban") String name) {
    String greeting = this.restTemplate.getForObject("http://say-hello/greeting", String.class);
    return String.format("%s, %s!", greeting, name);
  }

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

```xml
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-ribbon</artifactId>
    </dependency>
```

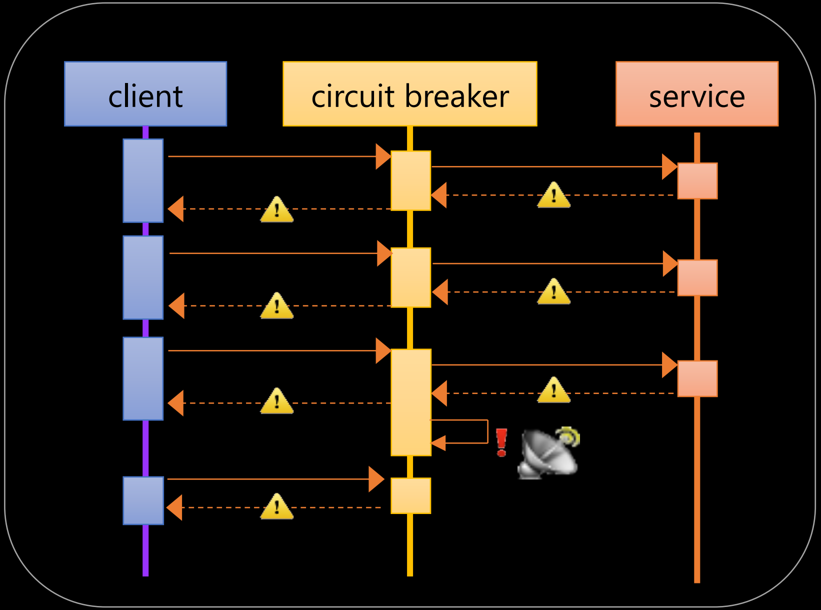
Service Registry and Discovery

- Discovery via DNS
  - Kubernetes DNS
  - Docker DNS
  - Consul – services must register
- Sidecar load balancers
  - Istio: Envoy proxy within Kubernetes pod

Fault Tolerance

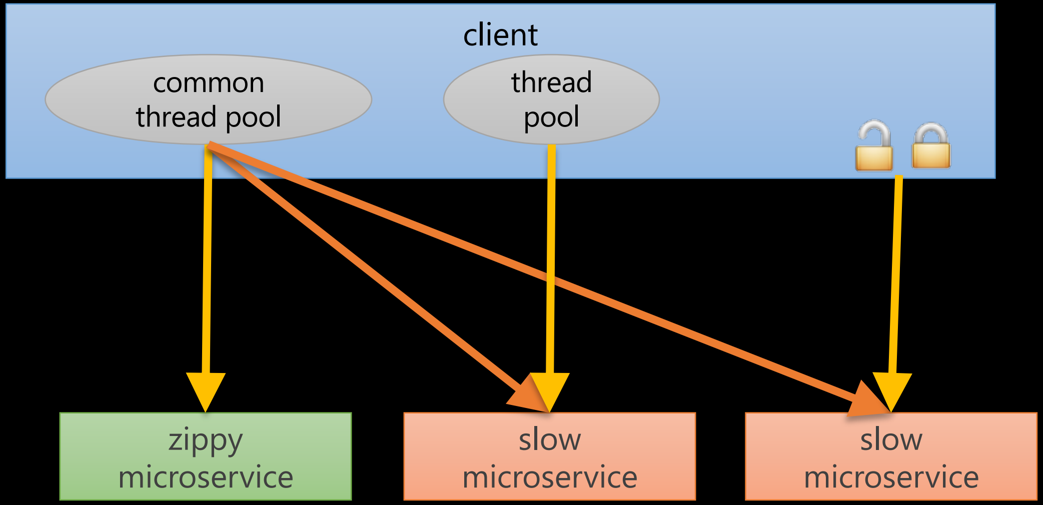
Circuit Breakers

- Wrap remote calls
- Monitor for failures
- Notify when circuit is tripped
- Retry or Fallback?
- When is circuit reset?



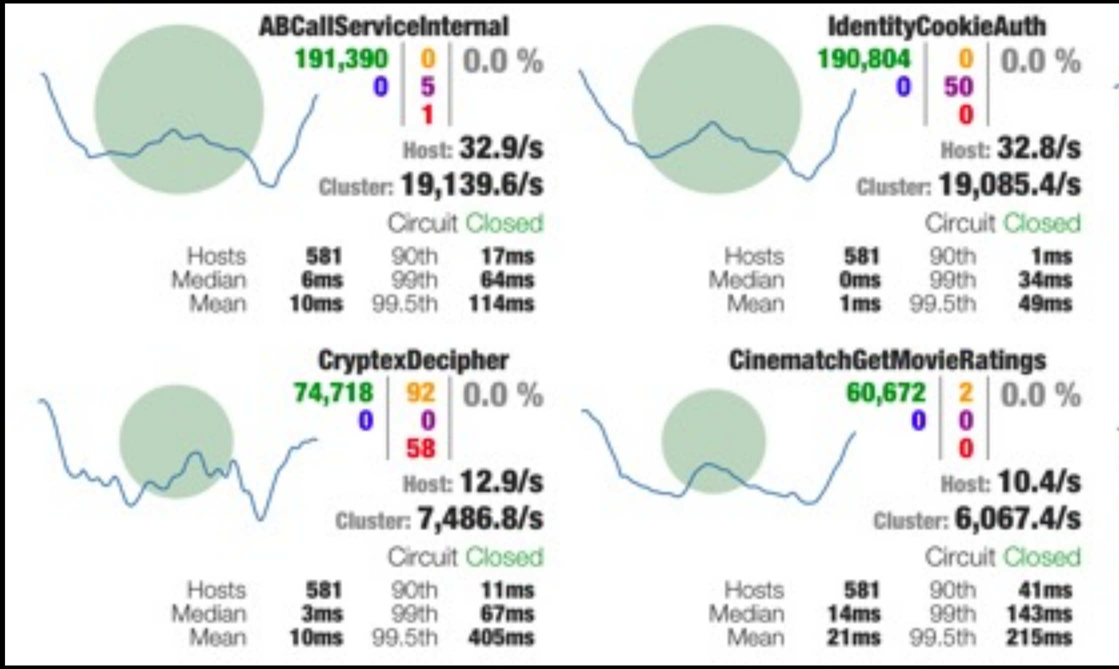
Bulkheads

- Ensure at most ‘n’ threads waiting for a slow resource
  - Thread isolation
    - With or without a queue
    - Timeout / fallback
  - Semaphore isolation
    - Request sent if lock obtained

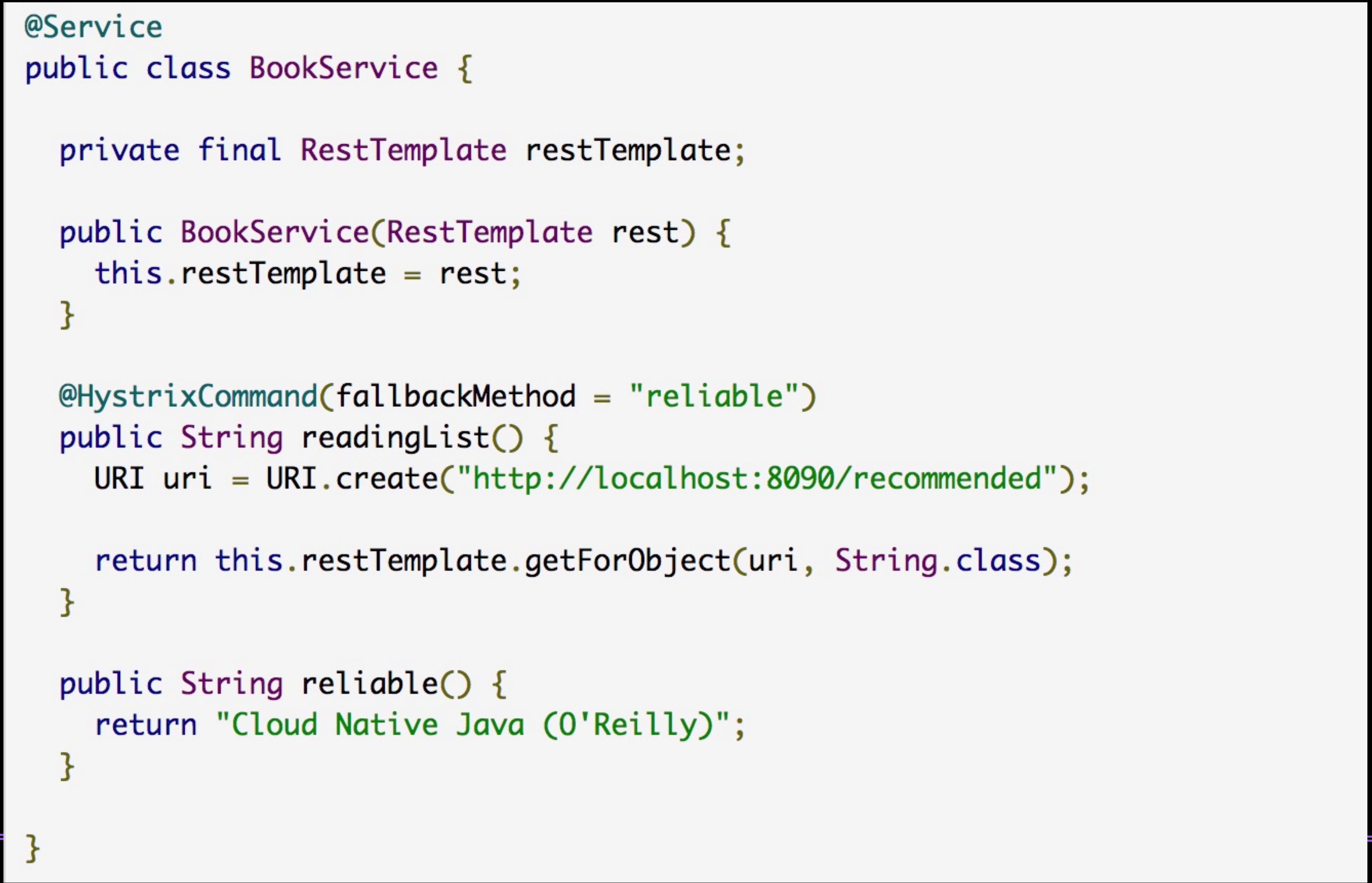


Fault Tolerance Library: Hystrix

- Circuit Breaker
- Bulk Head
  - Thread & Semaphore
- Fallbacks
- Request batching
- Real time monitoring:
  - Client-side metrics gathering



Spring Cloud with Hystrix



```java
@Service
public class BookService {

  private final RestTemplate restTemplate;

  public BookService(RestTemplate rest) {
    this.restTemplate = rest;
  }

  @HystrixCommand(fallbackMethod = "reliable")
  public String readingList() {
    URI uri = URI.create("http://localhost:8090/recommended");

    return this.restTemplate.getForObject(uri, String.class);
  }

  public String reliable() {
    return "Cloud Native Java (O'Reilly)";
  }

}
```

Fault Tolerance with Istio

- Circuit breaker
- Retries
- Timeouts
- Can not specify fallbacks
    - Lose some context

Configuration

- Environment Variables
  - VCAP_SERVICES?
  - Kubernetes Config Maps?

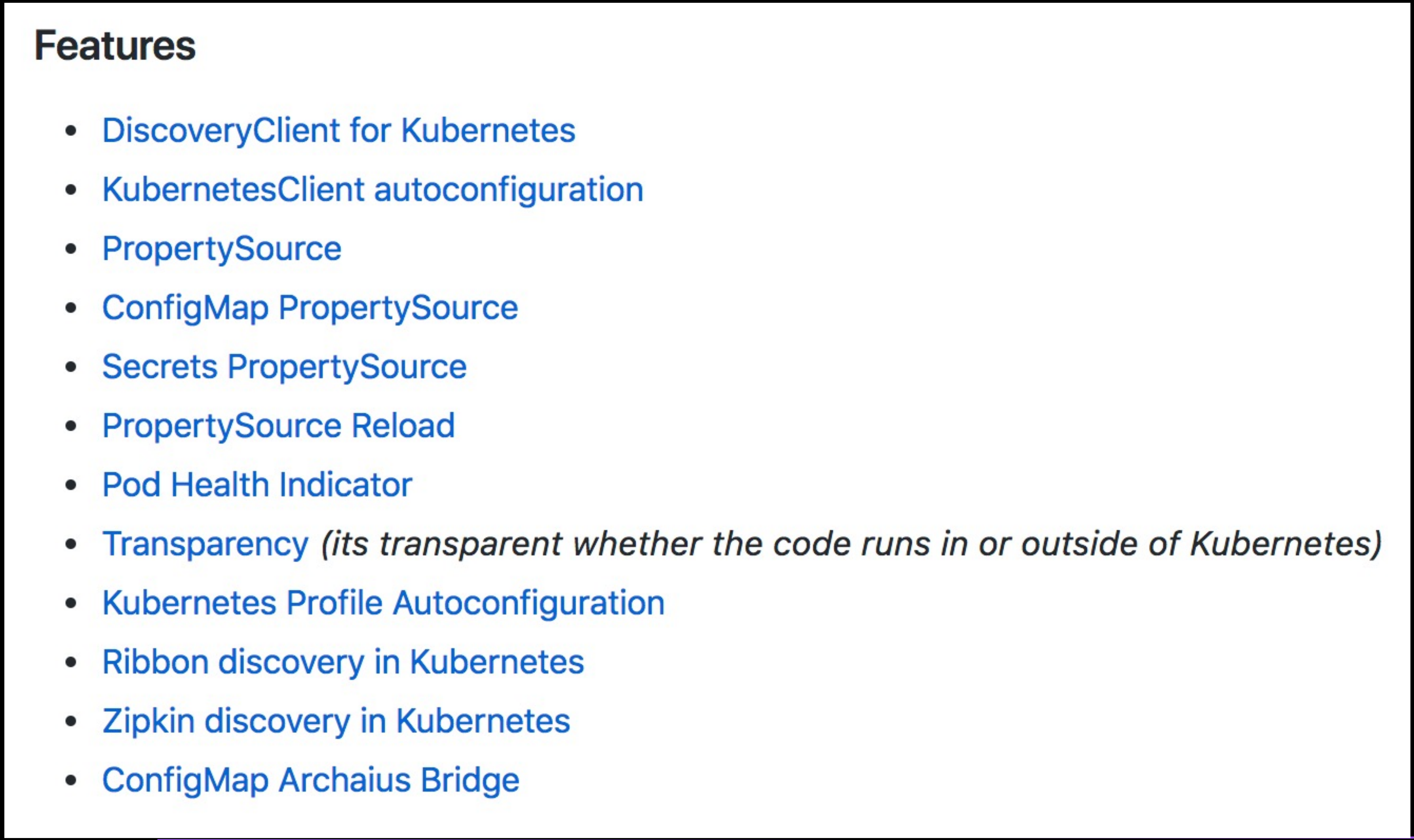
- How do you maintain configuration across environments?
- JSON structures with service bindings.. Are they the same?

Configuration

- Spring Cloud Config
  - External source for Spring configuration
  - Environment specific configuration
  - Handles secrets
  - Supports dynamic reconfiguration without restart

- Kubernetes ConfigMaps and Secrets

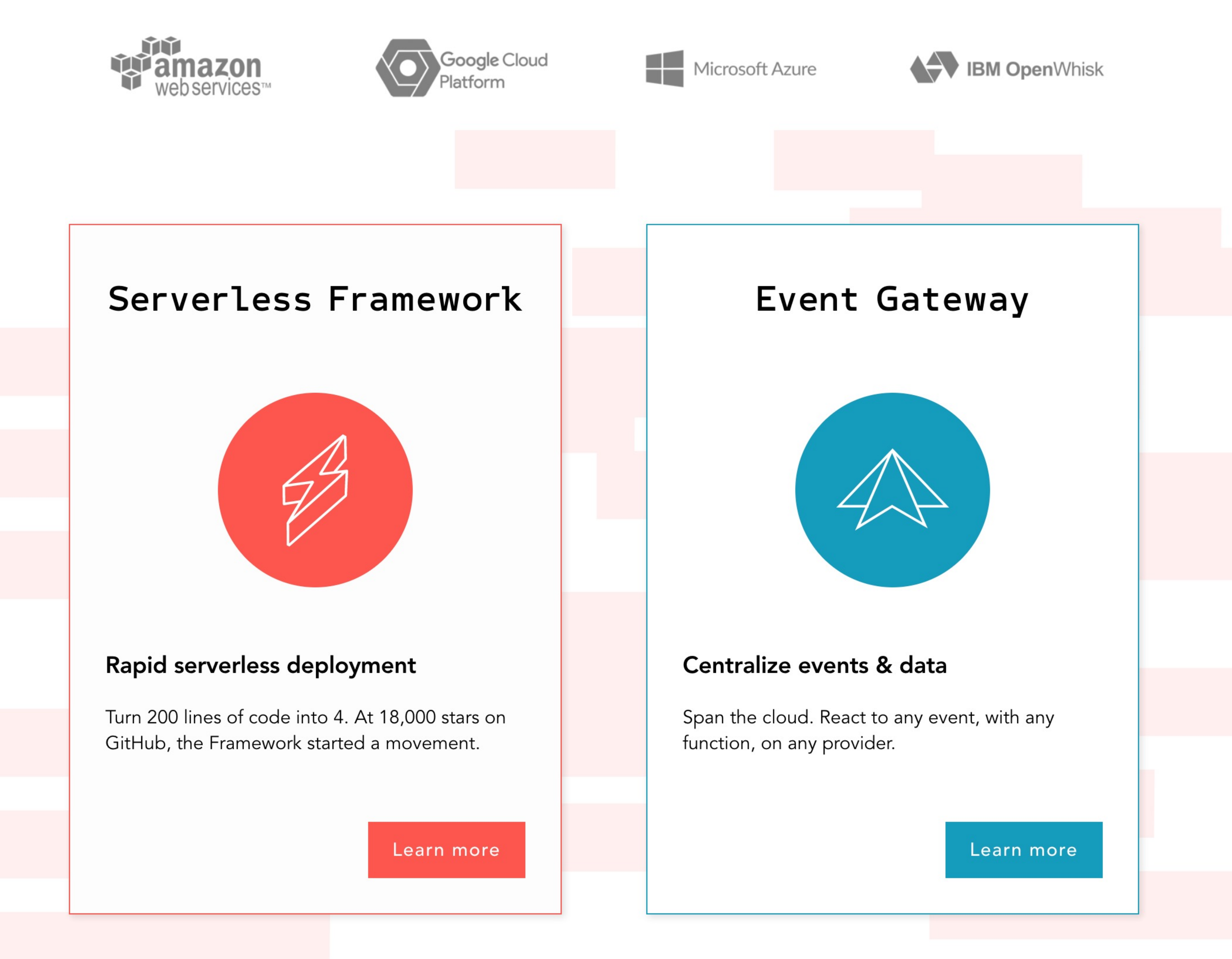
spring-cloud-kubernetes



Features

- DiscoveryClient for Kubernetes
- KubernetesClient autoconfiguration
- PropertySource
- ConfigMap PropertySource
- Secrets PropertySource
- PropertySource Reload
- Pod Health Indicator
- Transparency *(its transparent whether the code runs in or outside of Kubernetes)*
- Kubernetes Profile Autoconfiguration
- Ribbon discovery in Kubernetes
- Zipkin discovery in Kubernetes
- ConfigMap Archaius Bridge

So what is a cloud native application?



Serverless Framework

Rapid serverless deployment

Turn 200 lines of code into 4. At 18,000 stars on GitHub, the Framework started a movement.

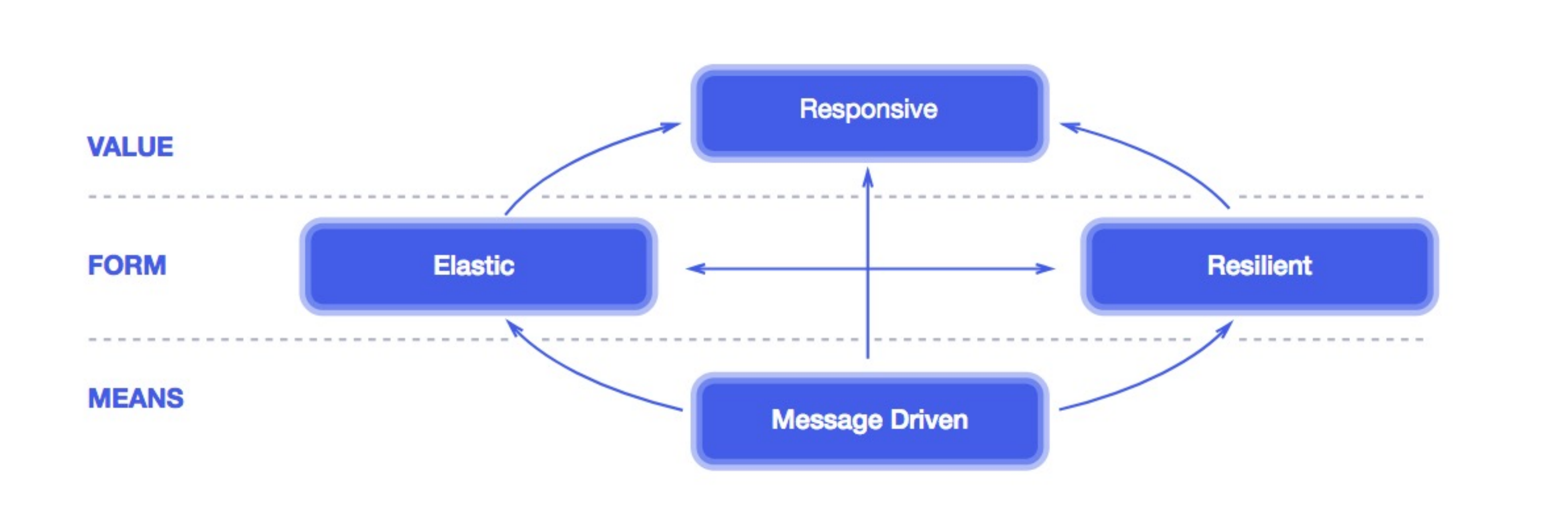
Event Gateway

Centralize events & data

Span the cloud. React to any event, with any function, on any provider.

Reactive Systems

- Asynchronous
- Non-blocking
- Event-driven (non-directed)



So what is a cloud native application?

what do you think?

Notices and disclaimers

- © 2018 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.
- **U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**
- Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
- IBM products are manufactured from new parts or new and used parts.
  In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
- **Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**
- Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those
- customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
- References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
- Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
- It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.