Microservices with Spring Cloud

Leverage the power of Spring to create resilient microservices

About Me

Marcin Grzejszczak

(@mgrzejszczak)

Spring Cloud developer at Pivotal.

- Spring Cloud Sleuth
- Spring Cloud Contract
- CI / CD

Course:

https://tinyurl.com/mg-contracts

Blog:

https://toomuchcoding.com

Hands-On Guide to Spring Cloud Contract: Creating Consumer-Driven Contracts to Leverage Contract Tests and Improve Your Code

by Marcin Grzejszczak

Publisher: Addison-Wesley Professional

Release Date: March 2019

ISBN: 9780135598436

View table of contents

PART 1

What is Spring Cloud and what does Spring Boot have to do with it?

SEGMENT 1

Agenda

12-factor application

Beyond the 12-factor application

What is Spring Boot?

What is Spring Cloud?

12-FACTOR APP

https://12factor.net/

Manifesto for cloud native applications from Heroku

Set of rules and guidelines

Declarative format for automation and setup

Beyond the 12factor app

The problem:

- How to build cloud native applications in practice?
- Apps not always running in Cloud
- Rules and guidelines specific for Heroku

Beyond the 12factor app

1. One codebase, one application
2. API first
3. Dependency management
4. Design, build, release, and run
5. Configuration, credentials, and code
6. Logs
7. Disposability
8. Backing services
9. Environment parity
10. Administrative processes
11. Port binding
12. Stateless processes
13. Concurrency
14. Telemetry
15. Authentication and authorization

https://content.pivotal.io/ebooks/beyond-the-12-factor-app

Beyond the 12factor app

1. One codebase, one application

Single codebase for an application

Multiple codebases suggest multiple applications

Difficult to deploy and test

Beyond the 12factor app

2. API first

Define contracts for your API

Define abstractions before implementing the details

Define how your interactions look like

Automate generation of the API documentation

Beyond the 12factor app

3. Dependency management

Cloud native applications bundled with all dependencies

Don’t assume that a dependency will be provided

Beyond the 12factor app

4. Design, build, release, and run

Hours of design can save weeks of coding

Design

- Architecture
- Bundled dependencies

Beyond the 12factor app

5. Configuration, credentials, and code

Store configuration in an environment

What about security?

    Assume externalization as if you pushed it to GitHub

    Have the credentials injected

    Ask for encrypted credentials and decrypt them at runtime

Beyond the 12factor app

6. Logs

Logs as event streams

Logs piped to an output stream

The application not concerned about stream storage

Beyond the 12factor app

7. Disposability

Treat an application as if it was disposable

It can be started or stopped at any time

Treat communication with other services the same way

Beyond the 12factor app

8. Backing services

Filesystem ephemeral

Instead use a backing service

    Caches, messaging systems, databases

Treat filesystem as a backing service

Beyond the 12factor app

9. Environment parity

Only production environment is production

Your application set up as if on production

Each commit a candidate for deployment to production

Beyond the 12factor app

10. Administrative processes

Factor 12 of the original 12 factor app - Run admin/management tasks as one-off processes

Issues with solutions like Cron

- Multiple instances in various zones

Create an application

- That runs batch jobs
- REST endpoints to run administrative jobs

Beyond the 12factor app

11. Port binding

Multiple instances, different ports

Push the problem to the platform

Manages network, Scaling, Routing etc.

Beyond the 12factor app

12. Stateless processes

State stored in a backing service

State not maintained in your application

Beyond the 12factor app

13. Concurrency

scale out horizontally

    No need to invest in more memory or cpu for your larger process

Multiple instances of your application do more work

Beyond the 12factor app

14. Telemetry

Application treated like a space probe

    When in space, can’t interact with it too much

Metrics

    Technical (health checks, load) - Application Performance Management

    Domain Specific (business domain) - Business Key Performance Indicators (KPI)

Beyond the 12factor app

15. Authentication and authorization

“We’ll talk about security if we don’t run out of time”

Security should be in core of your application’s development

It shouldn’t be added in the post-production phase

What is Spring Boot?

Opinionated view on Spring

Sets up third-party libraries if on classpath

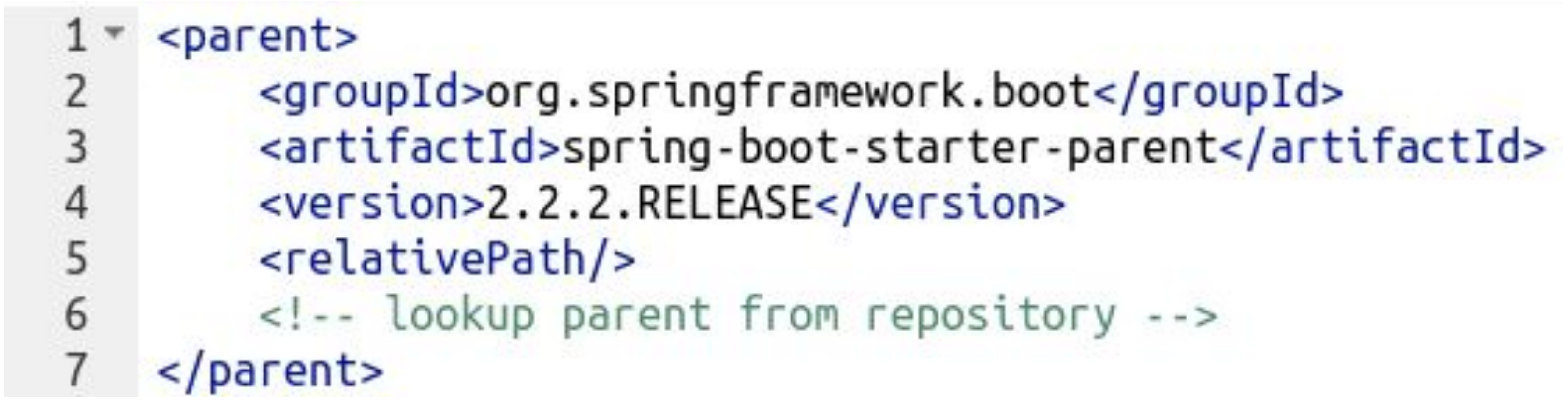
Fully extensible

Takes care of managing versions

Production-ready features e.g. metrics, health-checks

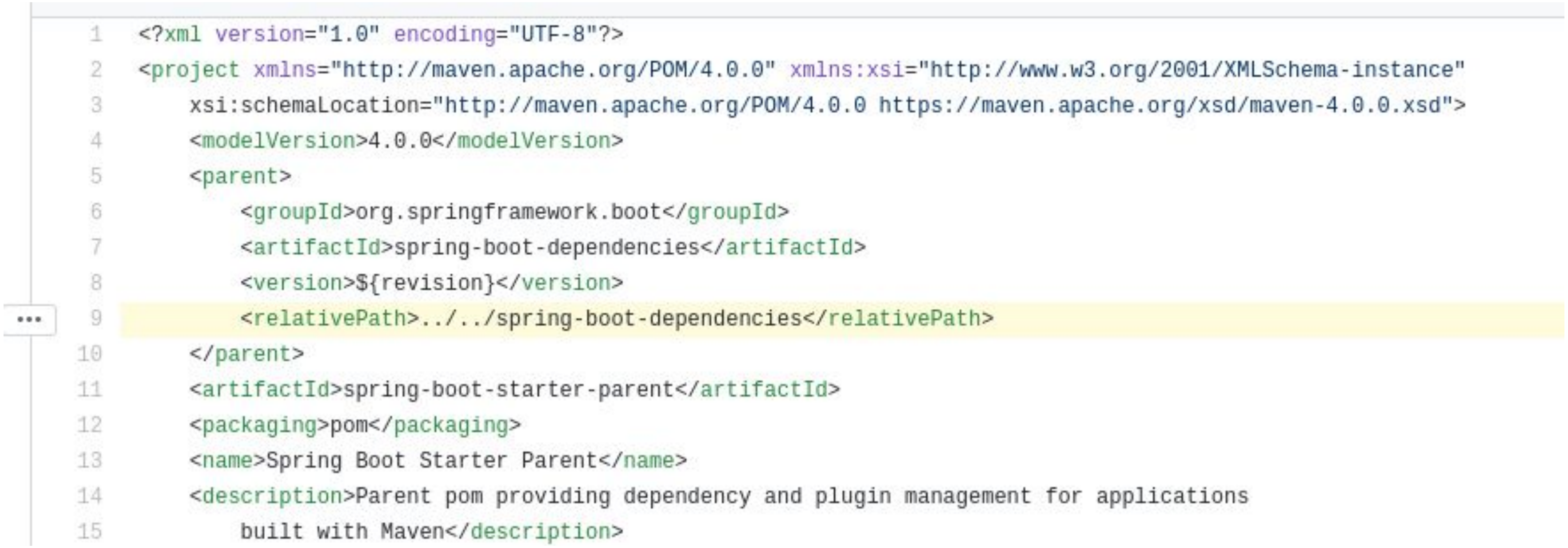
https://spring.io/projects/spring-boot

What is Spring Boot? - Version Management



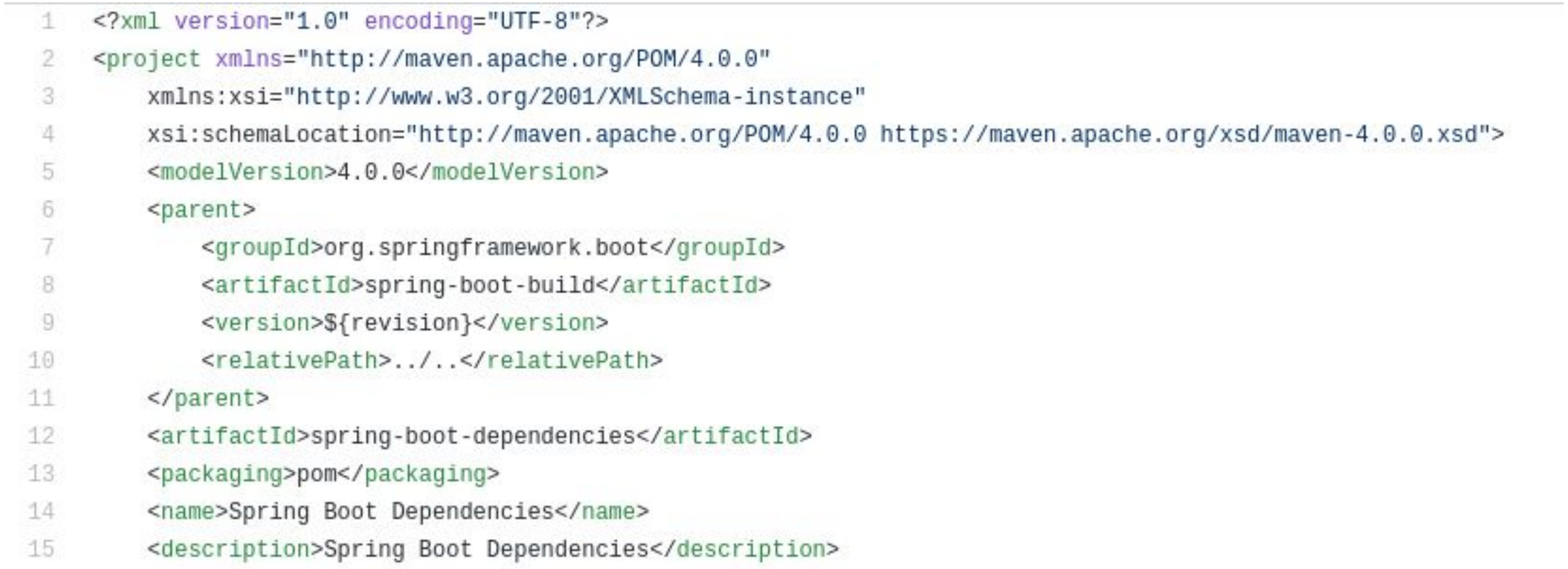
```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.2.RELEASE</version>
    <relativePath/>
    <!-- lookup parent from repository -->
</parent>
```

What is Spring Boot? - Version Management



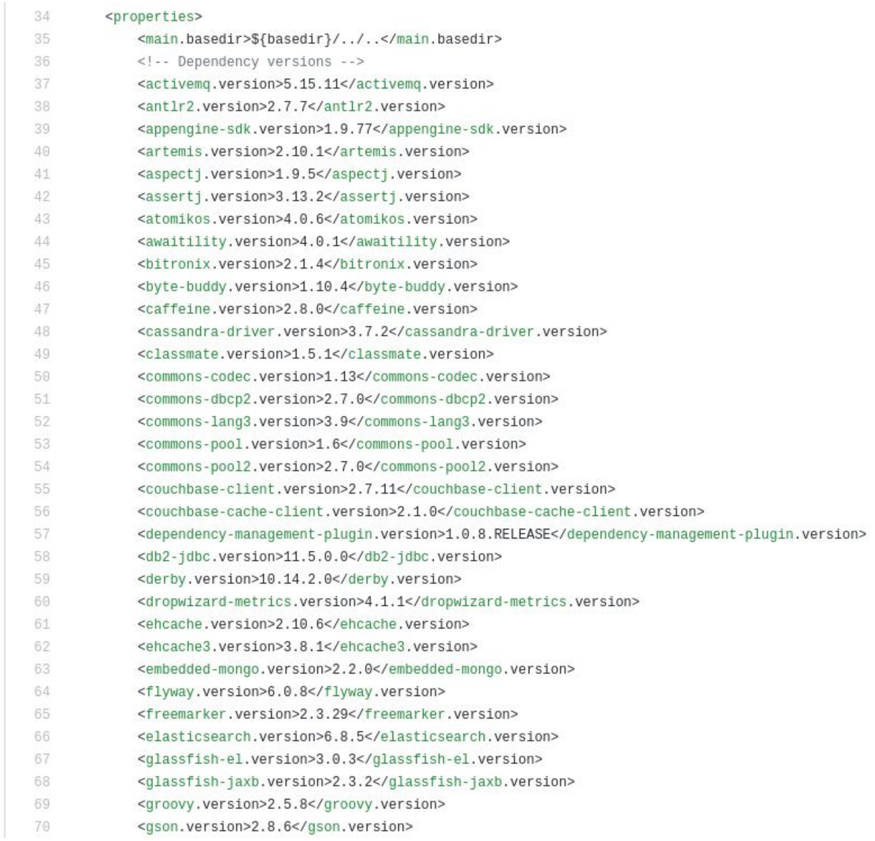
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-dependencies</artifactId>
        <version>${revision}</version>
        <relativePath>../../spring-boot-dependencies</relativePath>
    </parent>
    <artifactId>spring-boot-starter-parent</artifactId>
    <packaging>pom</packaging>
    <name>Spring Boot Starter Parent</name>
    <description>Parent pom providing dependency and plugin management for applications
        built with Maven</description>
```

What is Spring Boot? - Version Management



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-build</artifactId>
        <version>${revision}</version>
        <relativePath>../..</relativePath>
    </parent>
    <artifactId>spring-boot-dependencies</artifactId>
    <packaging>pom</packaging>
    <name>Spring Boot Dependencies</name>
    <description>Spring Boot Dependencies</description>
```

What is Spring Boot? - Version Management



```xml
<properties>
    <main.basedir>${basedir}/../..</main.basedir>
    <!-- Dependency versions -->
    <activemq.version>5.15.11</activemq.version>
    <antlr2.version>2.7.7</antlr2.version>
    <appengine-sdk.version>1.9.77</appengine-sdk.version>
    <artemis.version>2.10.1</artemis.version>
    <aspectj.version>1.9.5</aspectj.version>
    <assertj.version>3.13.2</assertj.version>
    <atomikos.version>4.0.6</atomikos.version>
    <awaitility.version>4.0.1</awaitility.version>
    <bitronix.version>2.1.4</bitronix.version>
    <byte-buddy.version>1.10.4</byte-buddy.version>
    <caffeine.version>2.8.0</caffeine.version>
    <cassandra-driver.version>3.7.2</cassandra-driver.version>
    <classmate.version>1.5.1</classmate.version>
    <commons-codec.version>1.13</commons-codec.version>
    <commons-dbcp2.version>2.7.0</commons-dbcp2.version>
    <commons-lang3.version>3.9</commons-lang3.version>
    <commons-pool.version>1.6</commons-pool.version>
    <commons-pool2.version>2.7.0</commons-pool2.version>
    <couchbase-client.version>2.7.11</couchbase-client.version>
    <couchbase-cache-client.version>2.1.0</couchbase-cache-client.version>
    <dependency-management-plugin.version>1.0.8.RELEASE</dependency-management-plugin.version>
    <db2-jdbc.version>11.5.0.0</db2-jdbc.version>
    <derby.version>10.14.2.0</derby.version>
    <dropwizard-metrics.version>4.1.1</dropwizard-metrics.version>
    <ehcache.version>2.10.6</ehcache.version>
    <ehcache3.version>3.8.1</ehcache3.version>
    <embedded-mongo.version>2.2.0</embedded-mongo.version>
    <flyway.version>6.0.8</flyway.version>
    <freemarker.version>2.3.29</freemarker.version>
    <elasticsearch.version>6.8.5</elasticsearch.version>
    <glassfish-el.version>3.0.3</glassfish-el.version>
    <glassfish-jaxb.version>2.3.2</glassfish-jaxb.version>
    <groovy.version>2.5.8</groovy.version>
    <gson.version>2.8.6</gson.version>
```

What is Spring Cloud?

Spring Boot based tools for developers with patterns in distributed systems

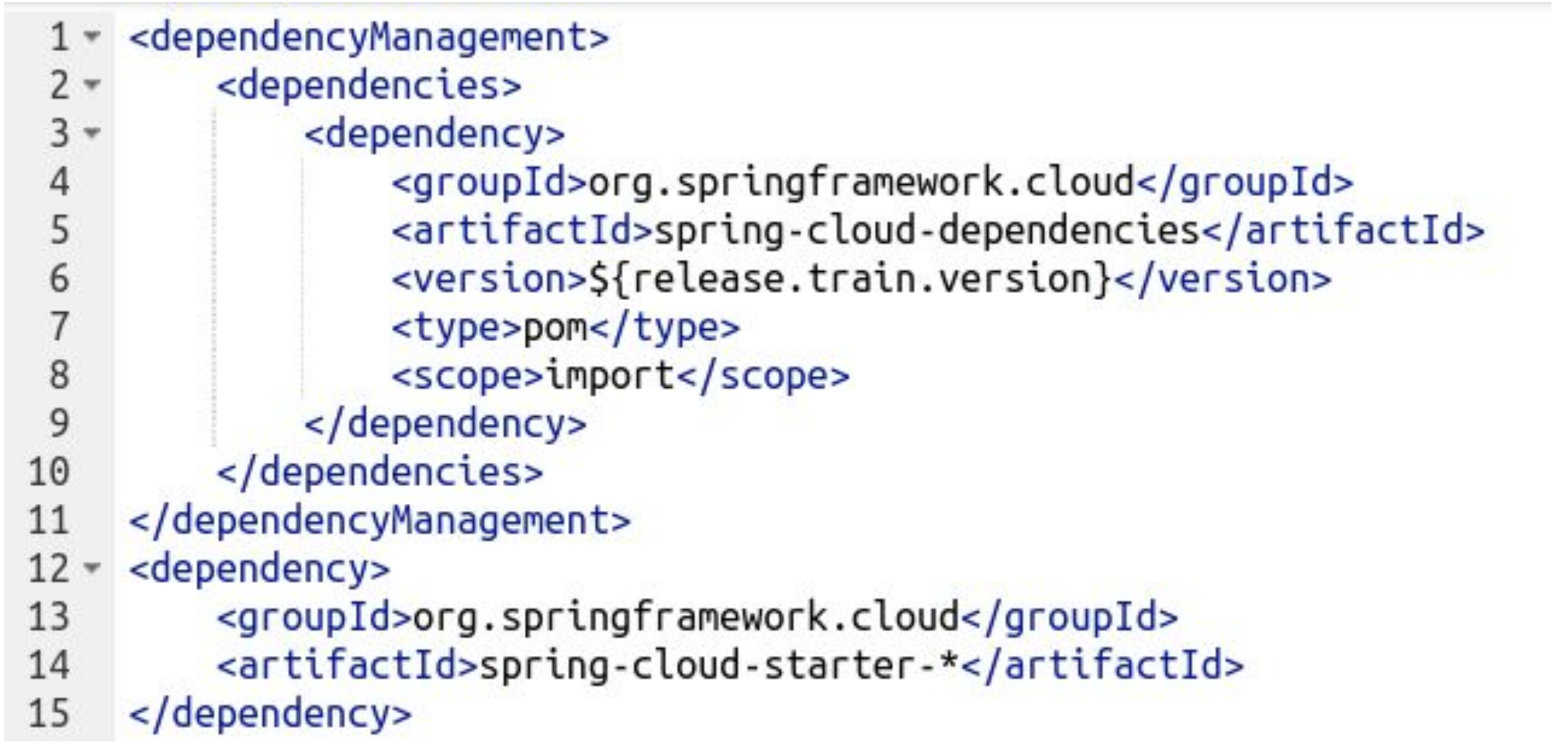
- Configuration management, service discovery etc.

Will work in any distributed environment

- Developer’s own laptop
- Bare metal data centres
- Managed platforms such as Cloud Foundry or Kubernetes

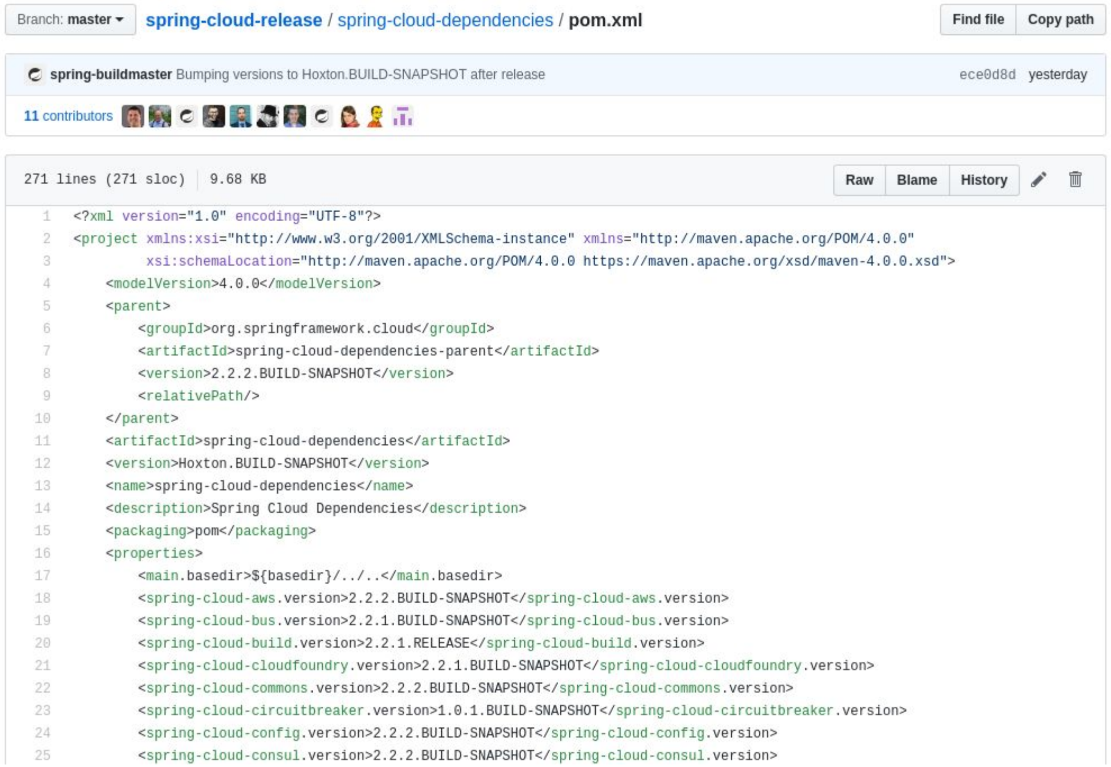
https://spring.io/projects/spring-cloud

What is Spring Cloud



```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>${release.train.version}</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-*</artifactId>
</dependency>
```

HTTPS://GITHUB.COM/SPRING-CLOUD/SPRING-CLOUD-RELEASE/

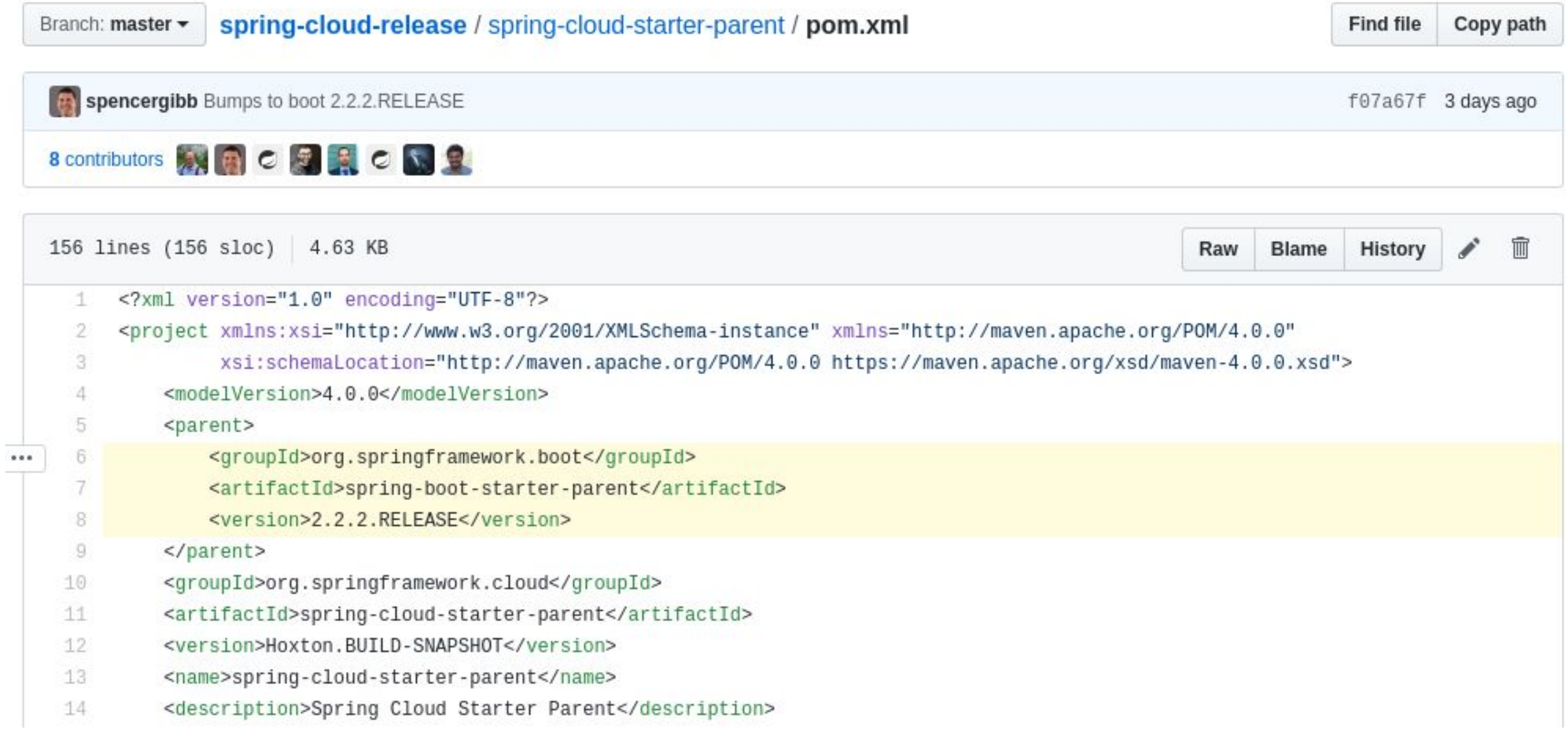


spring-buildmaster Bumping versions to Hoxton.BUILD-SNAPSHOT after release ece0d8d yesterday

271 lines (271 sloc) 9.68 KB

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-dependencies-parent</artifactId>
        <version>2.2.2.BUILD-SNAPSHOT</version>
        <relativePath/>
    </parent>
    <artifactId>spring-cloud-dependencies</artifactId>
    <version>Hoxton.BUILD-SNAPSHOT</version>
    <name>spring-cloud-dependencies</name>
    <description>Spring Cloud Dependencies</description>
    <packaging>pom</packaging>
    <properties>
        <main.basedir>${basedir}/../..</main.basedir>
        <spring-cloud-aws.version>2.2.2.BUILD-SNAPSHOT</spring-cloud-aws.version>
        <spring-cloud-bus.version>2.2.1.BUILD-SNAPSHOT</spring-cloud-bus.version>
        <spring-cloud-build.version>2.2.1.RELEASE</spring-cloud-build.version>
        <spring-cloud-cloudfoundry.version>2.2.1.BUILD-SNAPSHOT</spring-cloud-cloudfoundry.version>
        <spring-cloud-commons.version>2.2.2.BUILD-SNAPSHOT</spring-cloud-commons.version>
        <spring-cloud-circuitbreaker.version>1.0.1.BUILD-SNAPSHOT</spring-cloud-circuitbreaker.version>
        <spring-cloud-config.version>2.2.2.BUILD-SNAPSHOT</spring-cloud-config.version>
        <spring-cloud-consul.version>2.2.2.BUILD-SNAPSHOT</spring-cloud-consul.version>
```

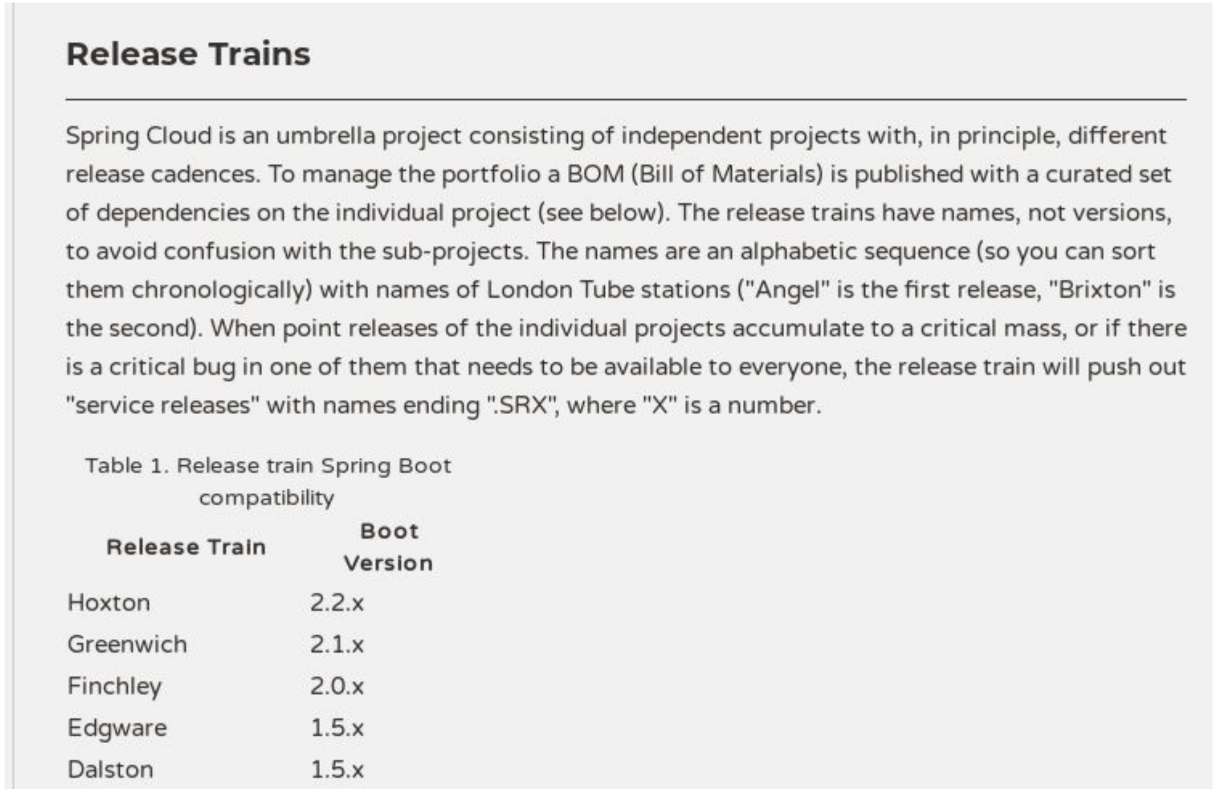
HTTPS://GITHUB.COM/SPRING-CLOUD/SPRING-CLOUD-RELEASE/



Branch: master
spring-cloud-release / spring-cloud-starter-parent / pom.xml
Find file
Copy path
spencergibb Bumps to boot 2.2.2.RELEASE
f07a67f 3 days ago
8 contributors
156 lines (156 sloc) 4.63 KB
Raw
Blame
History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.2.RELEASE</version>
    </parent>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-parent</artifactId>
    <version>Hoxton.BUILD-SNAPSHOT</version>
    <name>spring-cloud-starter-parent</name>
    <description>Spring Cloud Starter Parent</description>
```

What is Spring Cloud



Release Trains

Spring Cloud is an umbrella project consisting of independent projects with, in principle, different release cadences. To manage the portfolio a BOM (Bill of Materials) is published with a curated set of dependencies on the individual project (see below). The release trains have names, not versions, to avoid confusion with the sub-projects. The names are an alphabetic sequence (so you can sort them chronologically) with names of London Tube stations ("Angel" is the first release, "Brixton" is the second). When point releases of the individual projects accumulate to a critical mass, or if there is a critical bug in one of them that needs to be available to everyone, the release train will push out "service releases" with names ending ".SRX", where "X" is a number.

Table 1. Release train Spring Boot compatibility

Release Train	Boot Version
Hoxton	2.2.x
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

SEGMENT 2

Agenda

What is Spring Initlizr?

How to use it?

What problem are we trying to solve?

Manual version setting

Version mismatch

Class / Method not found

Dependency management hell

0

I want to use spring cloud consul for Service discovery, but getting `java.lang.ClassNotFoundException: org.springframework.cloud.client.loadbalancer.RestTemplateCustomizer` , what am i missing here?

my pom.xml

```
    <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-actuator</artifactId>
     </dependency>

     <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-commons</artifactId>
         <version>1.0.0.RELEASE</version>
     </dependency>

     <!-- <dependency> <groupId>org.springframework.cloud</groupId> <artifactId>spring-
         <version>1.0.0.M1</version> </dependency> -->


     <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-consul-discovery</artifactId>
         <version>1.0.0.M1</version>
     </dependency>
     <dependency>
         <groupId>org.springframework.cloud</groupId>
         <artifactId>spring-cloud-consul-bus</artifactId>
         <version>1.0.0.M1</version>
     </dependency>
     <!-- <dependency> <groupId>org.springframework.cloud</groupId> <artifactId>spring-
         <version>1.0.0.M1</version> </dependency> -->
```

do I have to add any extra dependencies next to:

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-security</artifactId>
  <version>1.0.1.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-eureka</artifactId>
  <version>${spring.cloud.version}</version>
</dependency>
```

1 Answer

active oldest votes

4

You have a mixture of Spring 4.2.1.RELEASE and 4.1.7.RELEASE on the classpath. You need the former rather than the latter. Specifically, you have the wrong version of `spring-core` on the classpath:

```
[INFO] |  +- org.springframework:spring-core:jar:4.1.7.RELEASE:compile
```

I would guess you have a dependency on `spring-core` with an explicit version declared on it. If you remove the `<version>`, the dependency management provided by `spring-cloud-starter-parent` will give you the right version (4.2.1.RELEASE).

share edit flag

answered Sep 19 '15 at 1:31
Andy Wilkinson
69.6k ● 13 ● 172 ● 173

you are right. Some nasty person added that to one of the poms. Once i removed it things were fine.
– EvilJinious1 Sep 20 '15 at 21:36

add a comment

What is Spring Initializr?

Extensible API to generate JVM-based projects

- Basic language generation for Java, Kotlin and Groovy
- Build system abstraction for Apache Maven and Gradle
- .gitignore support
- Several hook-points for custom resources generations

https://github.com/spring-io/initializr

What is Start Spring IO?

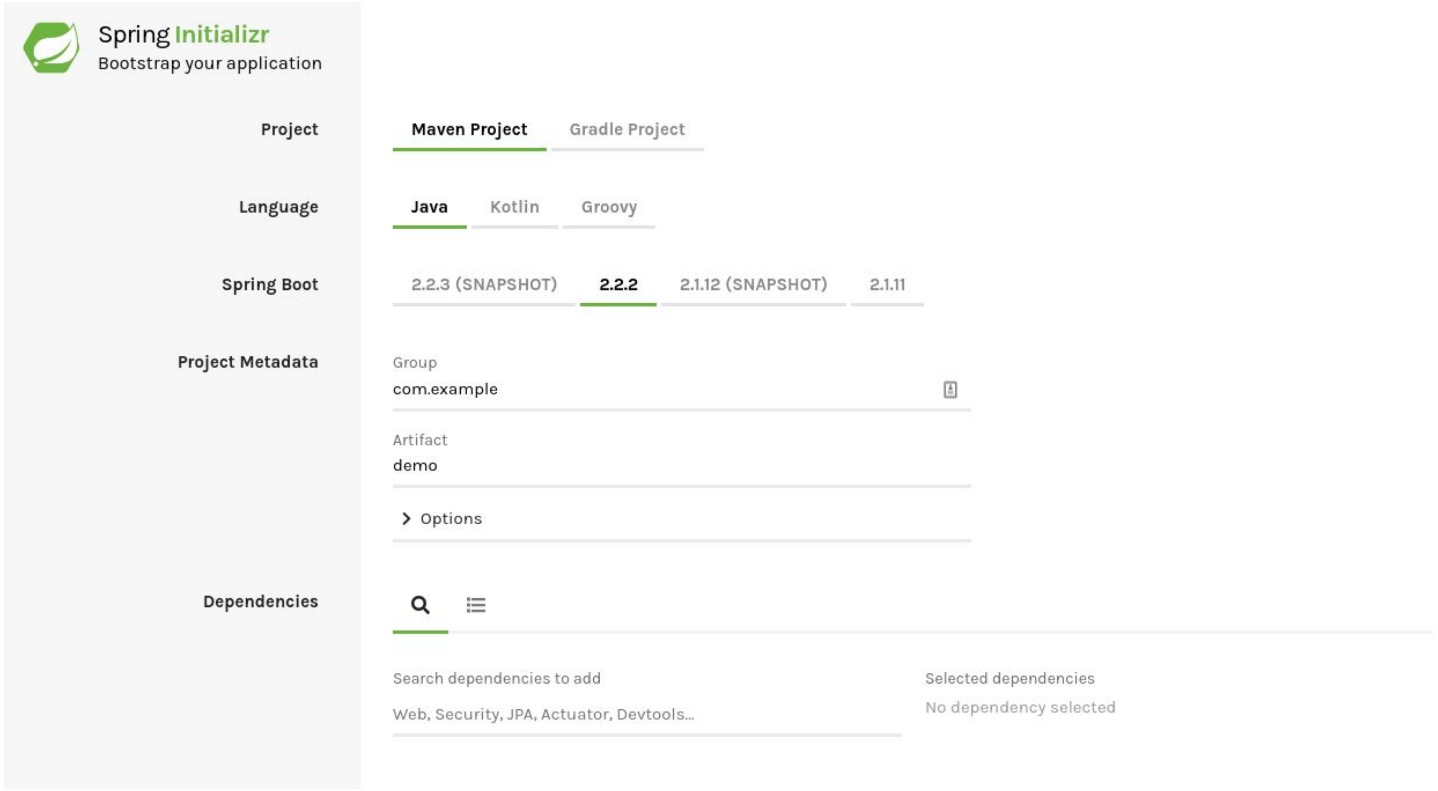
A GitHub project
https://github.com/spring-io/start.spring.io

Set up for the https://start.spring.io site

    Configuration
https://github.com/spring-io/start.spring.io/blob/master/start-site/src/main/resources/application.yml

Contains a custom UI

What is Start Spring IO?



Spring Initializr
Bootstrap your application

Project: Maven Project [✓] | Gradle Project

Language: Java [✓] | Kotlin | Groovy

Spring Boot: 2.2.3 (SNAPSHOT) | 2.2.2 [✓] | 2.1.12 (SNAPSHOT) | 2.1.11

Project Metadata

Group
com.example

Artifact
demo

Options

Dependencies

Search dependencies to add
Web, Security, JPA, Actuator, Devtools...

Selected dependencies
No dependency selected

How to use Start Spring IO?

Supported interfaces
https://github.com/spring-io/initializr#supported-interfaces

- Command line
- IDE
- Custom Web UI

DEMO

SEGMENT 3

Agenda

Configuration, credentials, and code

What is Spring Cloud Config server?

Beyond the 12factor app

5. Configuration, credentials, and code

Factor 3 of the original 12 factor app - store configuration in an environment

What about security?

Assume externalization as if you pushed it to GitHub

Have the credentials injected

Ask for encrypted credentials and decrypt them at runtime

Spring Cloud Config

GitHub project
https://spring.io/projects/spring-cloud-config

Server and client-side support for externalizing configuration

The default backend implementation is git

Spring Cloud Config Server

Different backends

    Git, File System, Vault, JDBC, Redis, AWS S3, CredHub

Security

    Spring Boot-configured HTTP Basic security with Spring Security

    symmetric (shared) key

    asymmetric (RSA key pair)

Spring Cloud Config Server Security

Encryption

```
    $ curl localhost:8888/encrypt -d mysecret
```

Decryption

```
    $ curl localhost:8888/decrypt -d
682bc583f4641835fa2db009355293665d2647dade3375c0ee201de2a49f
7bda
```

DEMO

Spring Cloud Config Server

Spring Cloud Config has an HTTP service:

```
/{application}/{profile}[/{label}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties
```

DEMO

Spring Cloud Config Client

Binds to the Config Server (default localhost:8888)

    you can override it by `bootstrap.properties`

Spring Environment with remote property sources

Add the **spring-cloud-starter-config** dependency

You need to refresh the application to see the changed prop

Whitelist the refresh endpoint
management.endpoints.web.exposure.include=*

DEMO

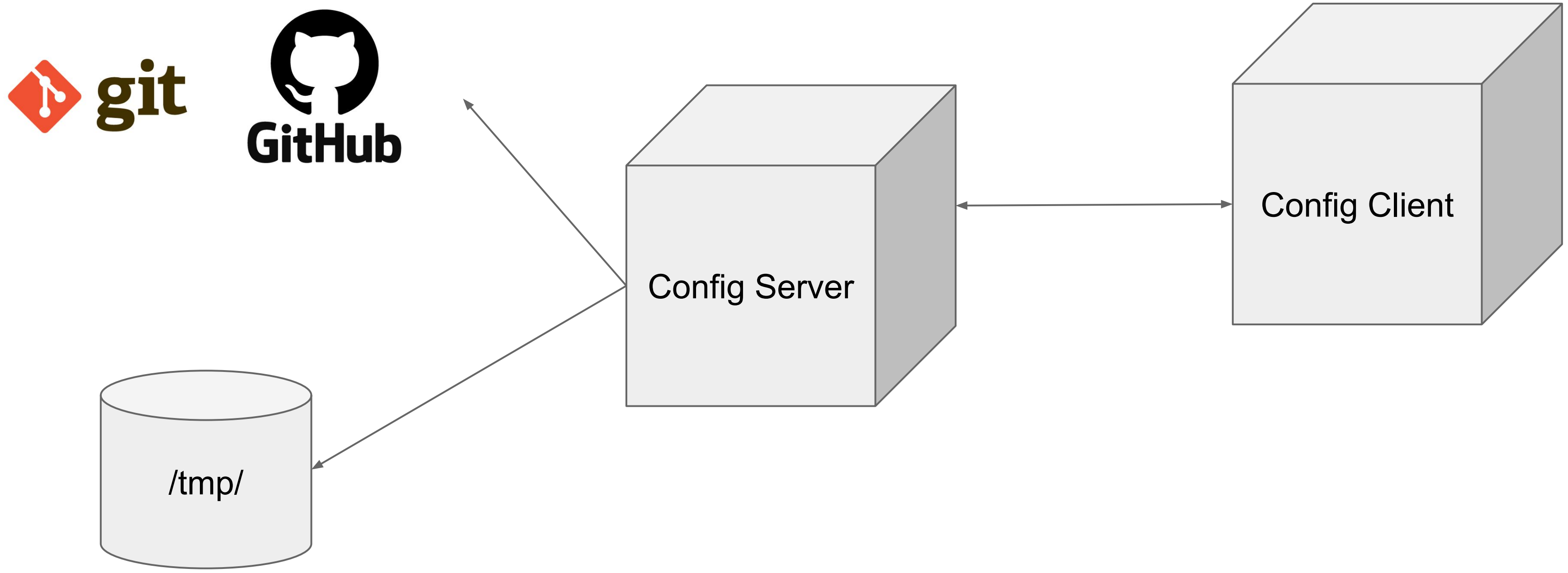
PART 1
Assignment

Part 1 - Assignment

Externalizing configuration via Spring Cloud Config. In this lab, students will use the Project Initializr (start.spring.io) to generate two projects - a Spring Cloud Config server and a Spring Cloud Config client application. Students will create a simple Git repository where the externalized configuration will be stored. During the exercise students will be able to fetch those properties and refresh them at runtime.

Assignment time (15 min)

Part 1 - Assignment



Part 1 - Assignment

https://tinyurl.com/spring-cloud-workshops#assignment-1

https://gist.github.com/marcingrzejszczak/82a0e46f65c9ba3280dd14f395bfbf5d#assignment-1

PART 2

How can microservices communicate and what is service discovery?

SEGMENT 4

Agenda

How can microservices communicate?

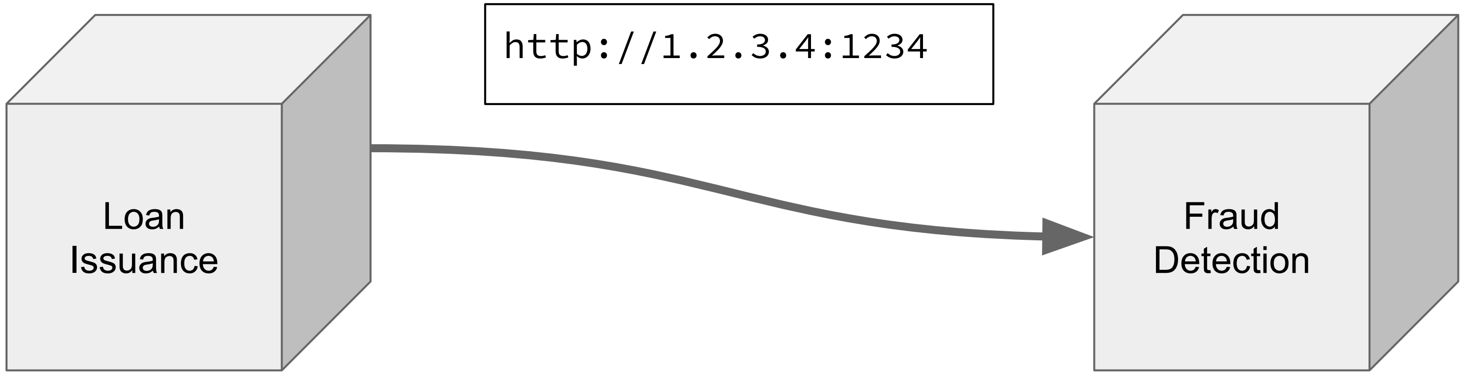
What is service discovery?

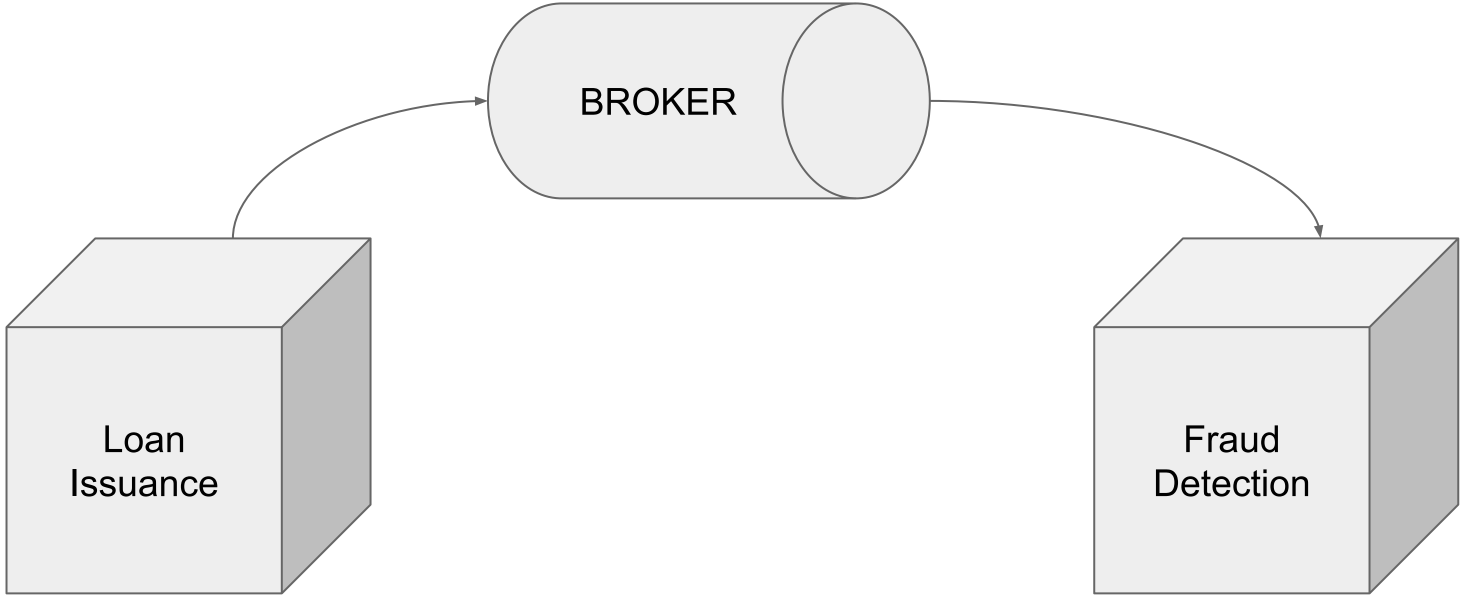
What is Netflix Eureka?

How can microservices communicate?

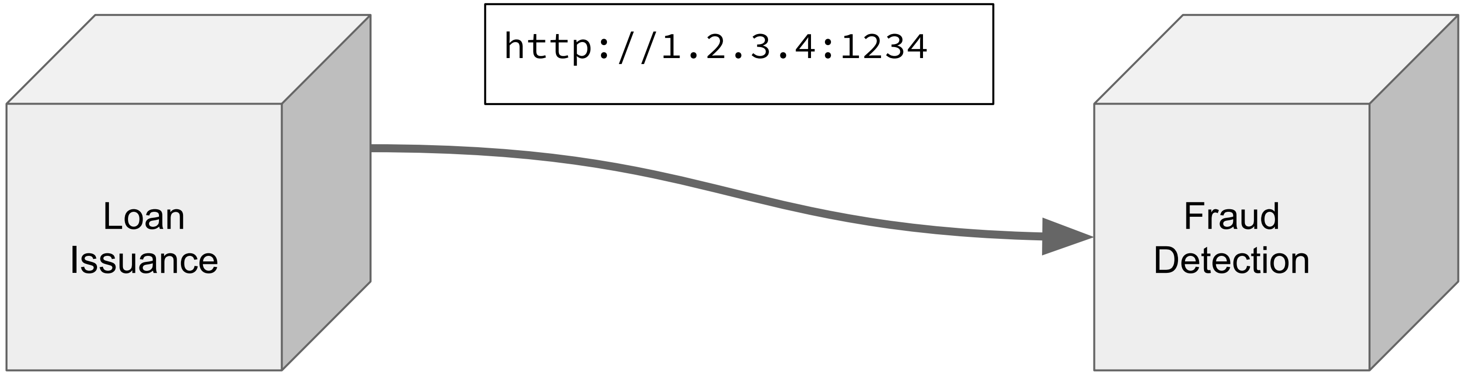
HTTP

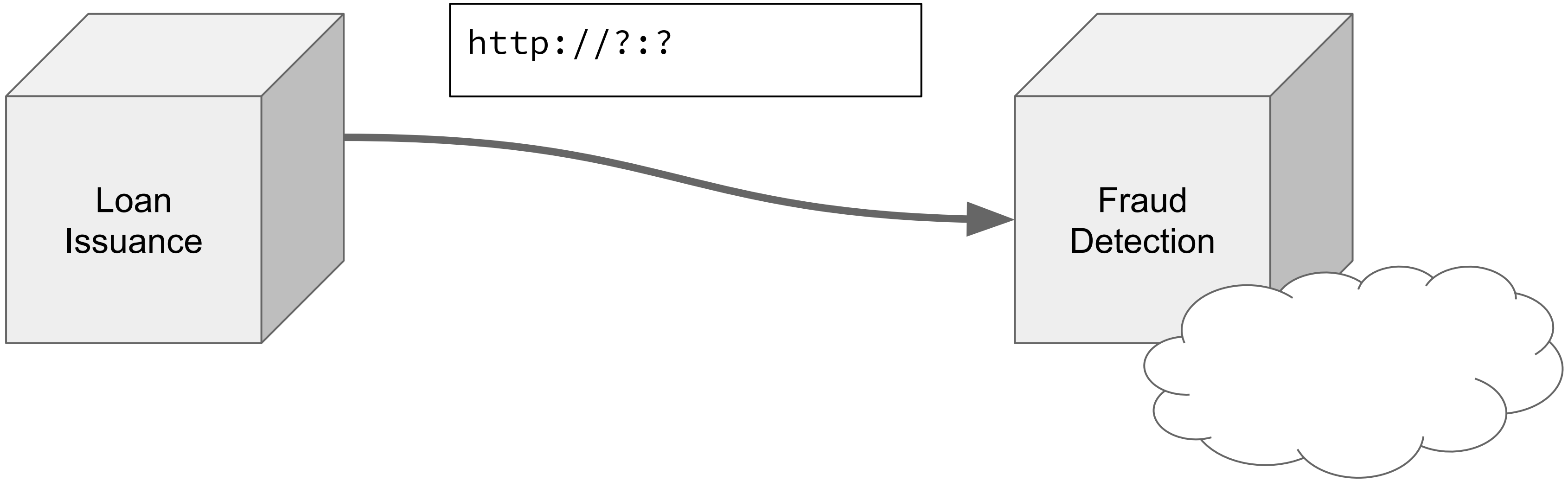
Messaging

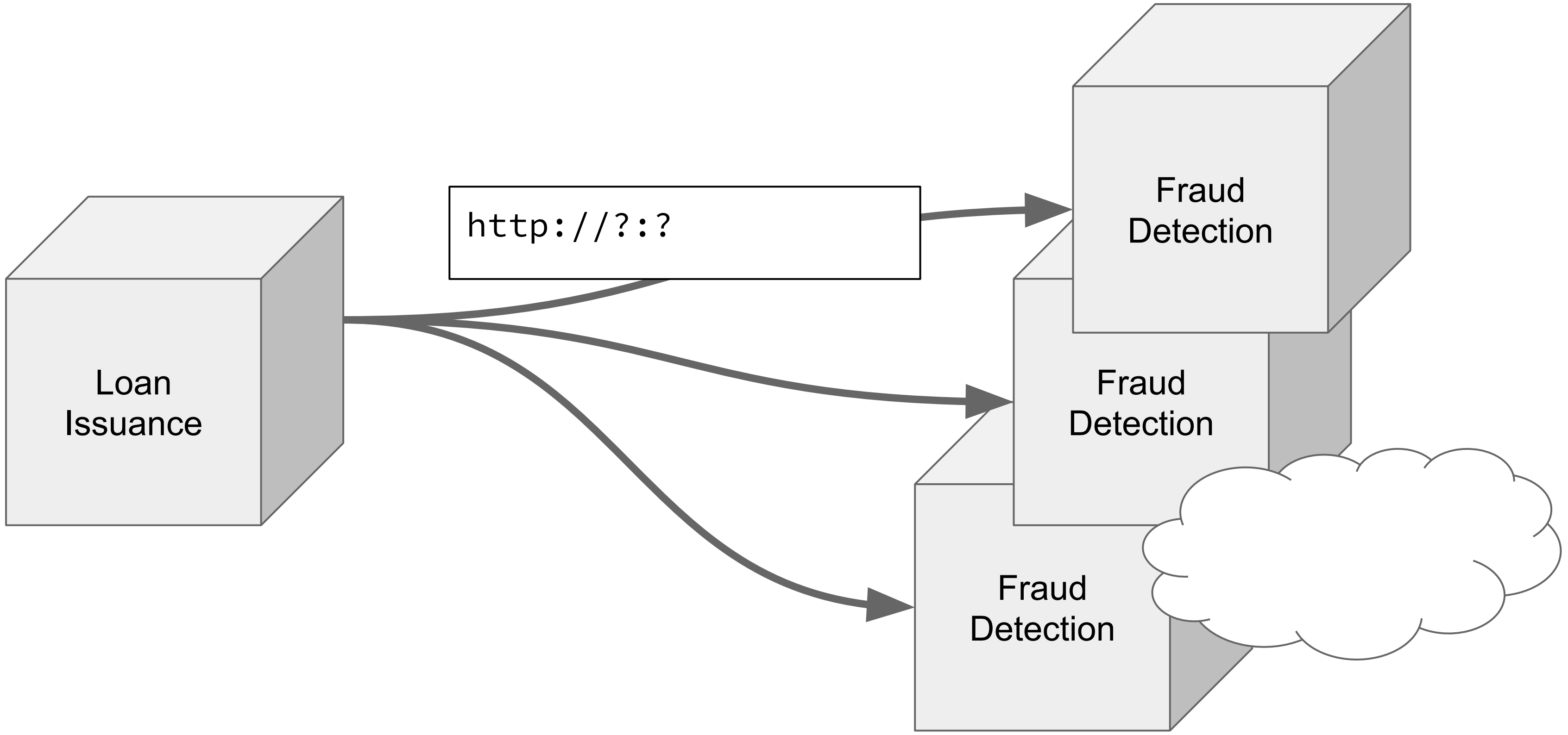

Loan Issuance
http://1.2.3.4:1234
Fraud Detection

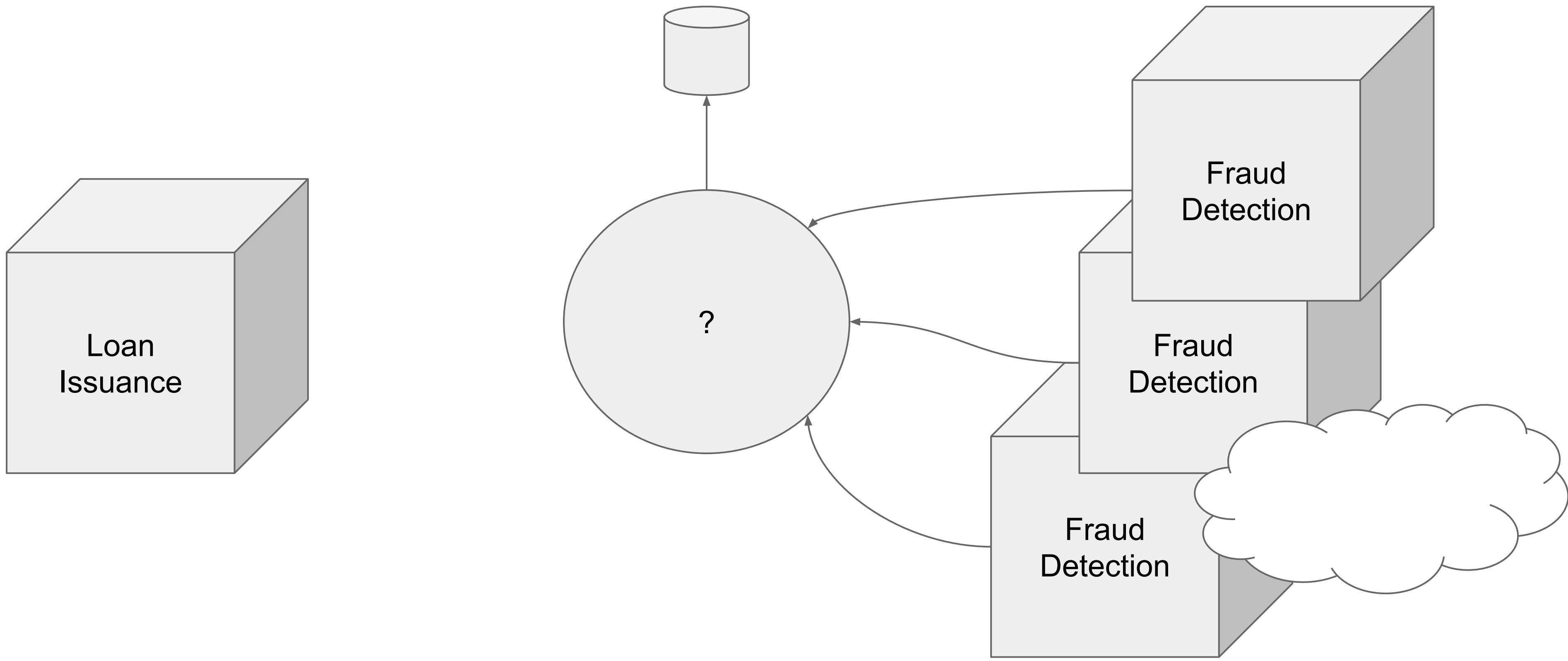

BROKER
Loan Issuance
Fraud Detection

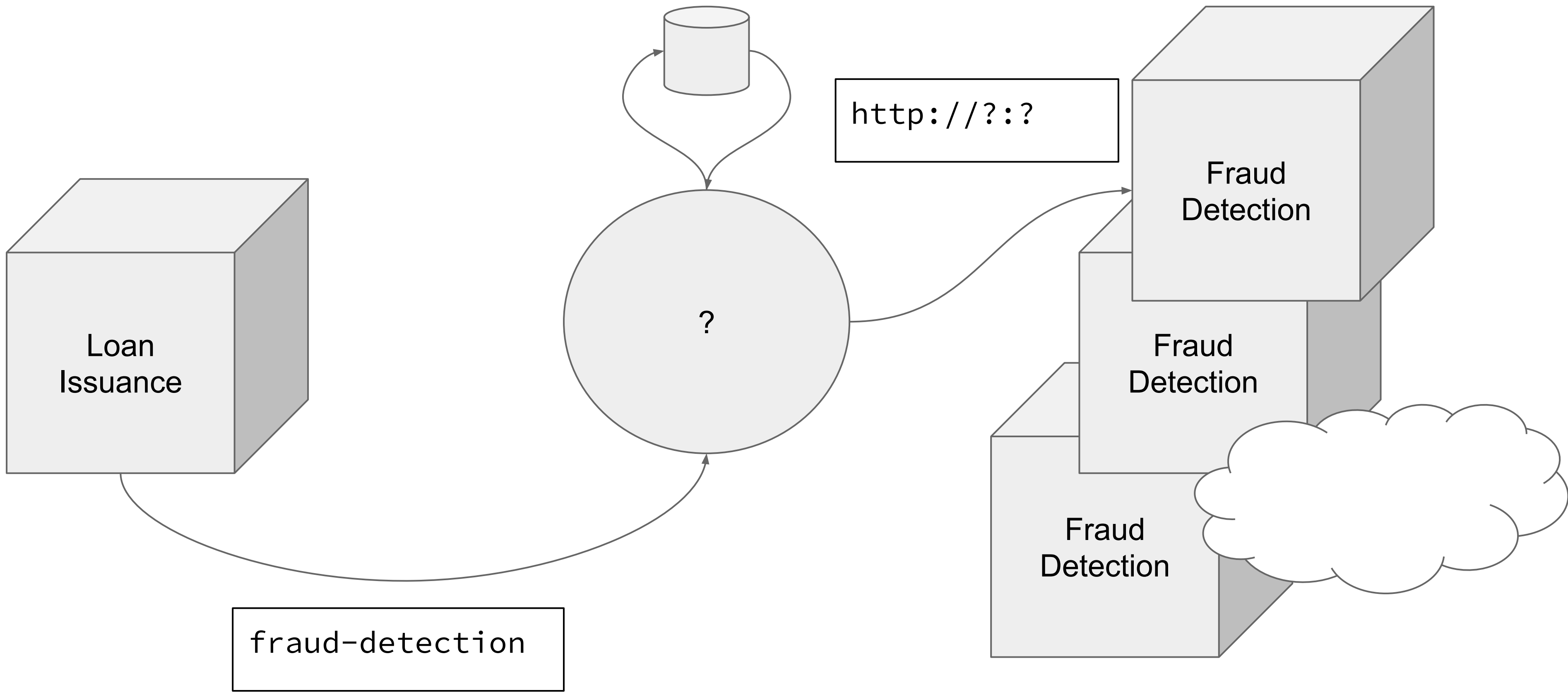
Service Discovery


Loan Issuance
http://1.2.3.4:1234
Fraud Detection


Loan Issuance
http://?:?
Fraud Detection


Loan Issuance
http://?:?
Fraud Detection
Fraud Detection
Fraud Detection


Loan Issuance
?
Fraud Detection
Fraud Detection
Fraud Detection

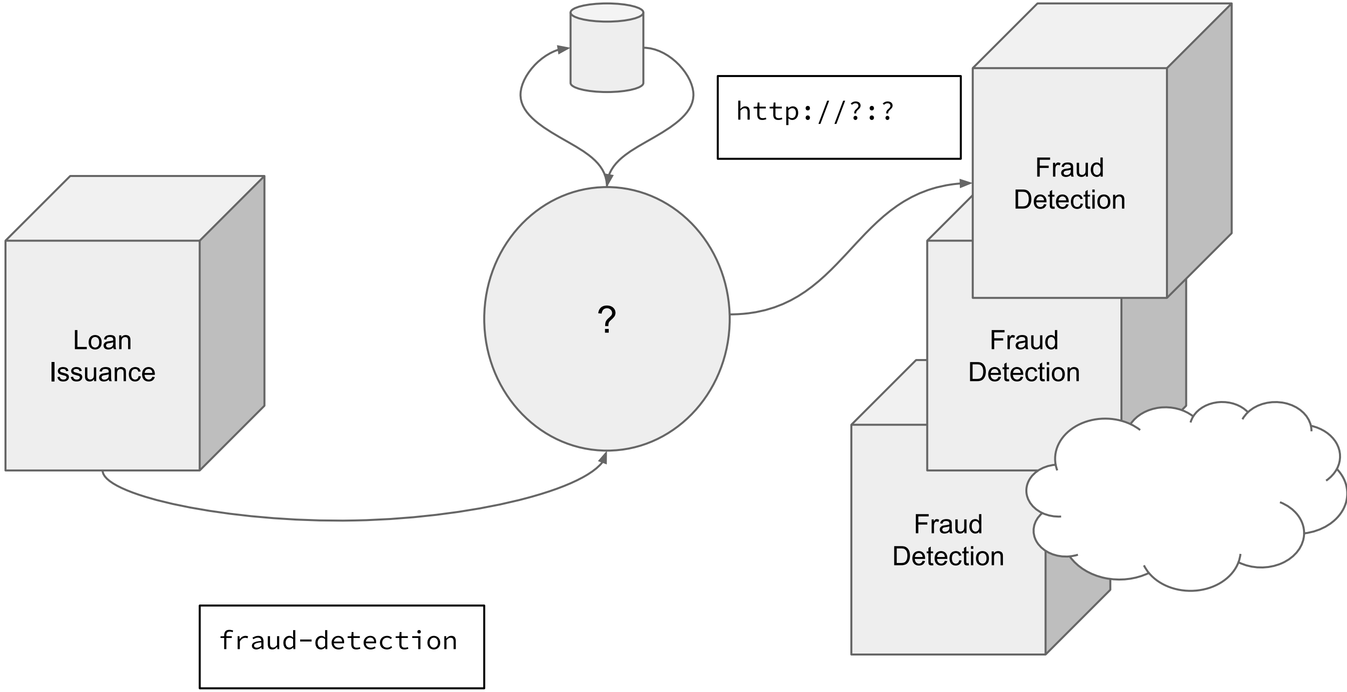

http://?:?
Fraud Detection
Fraud Detection
Fraud Detection
Loan Issuance
?
fraud-detection

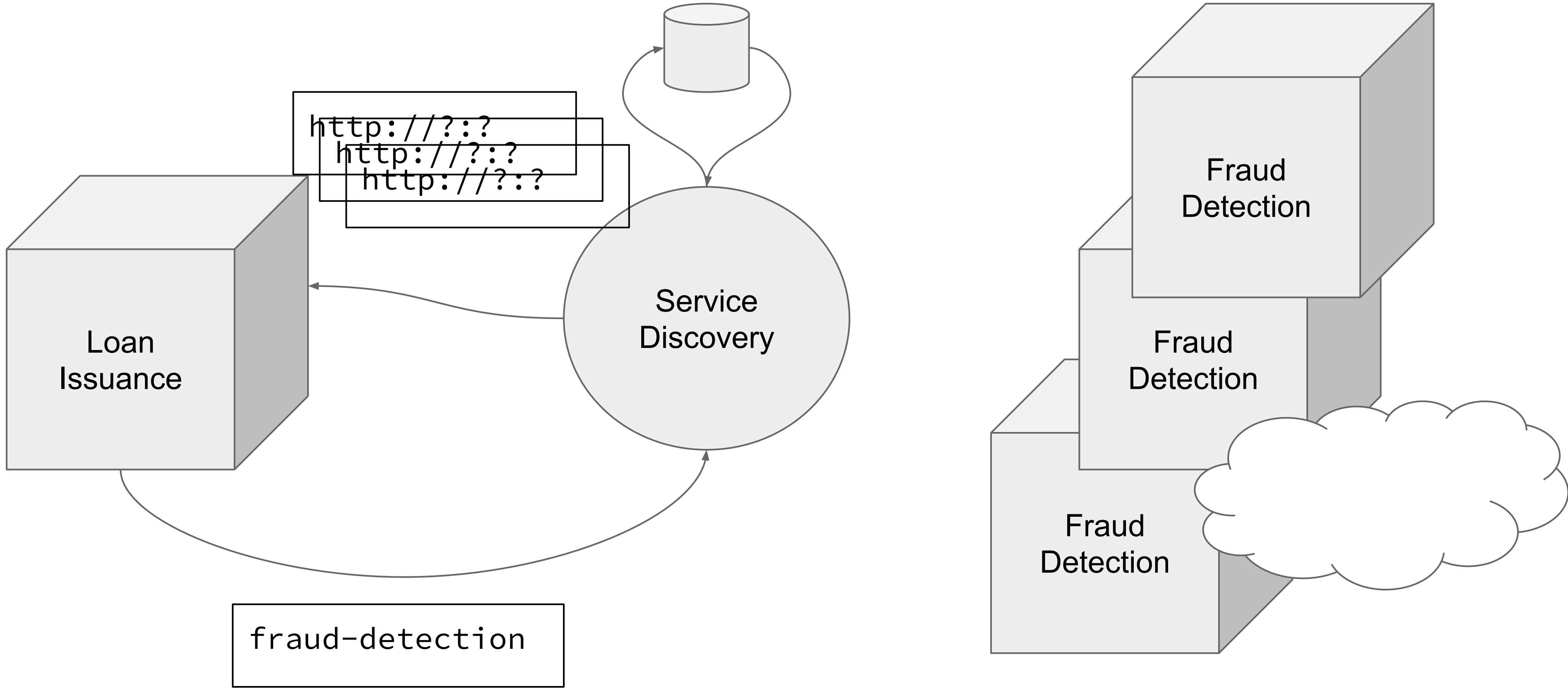
Load Balancing

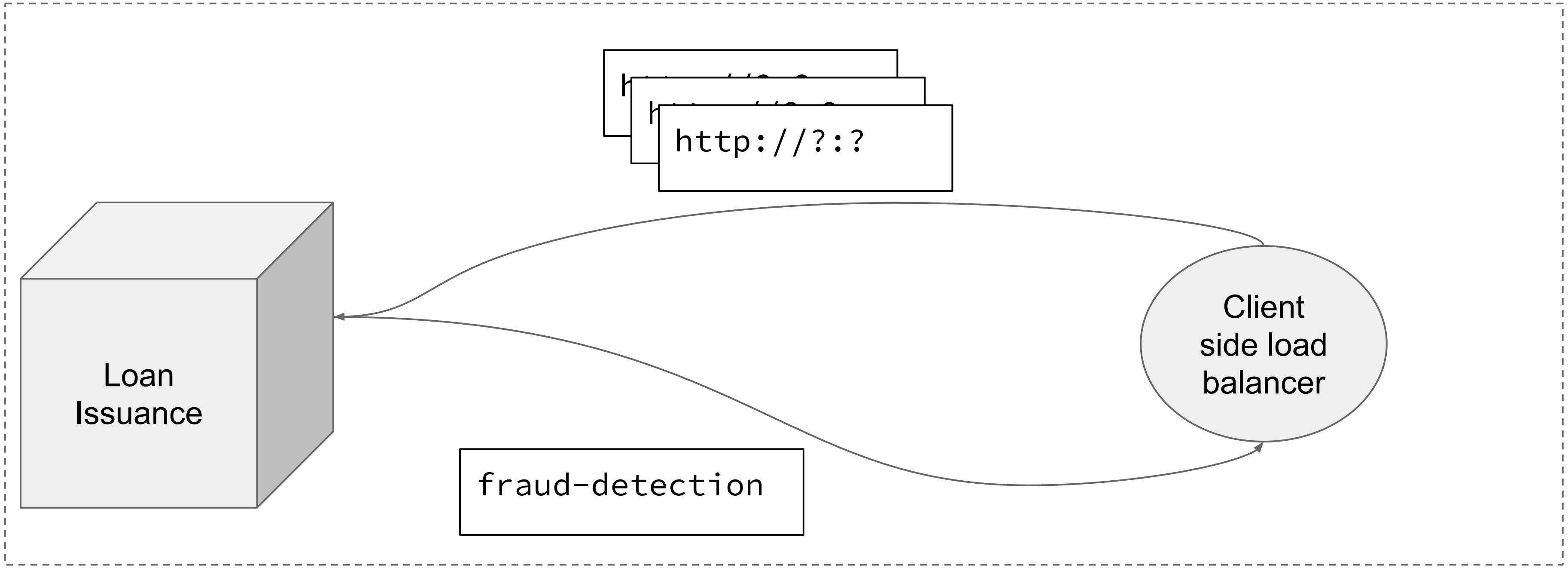
DNS

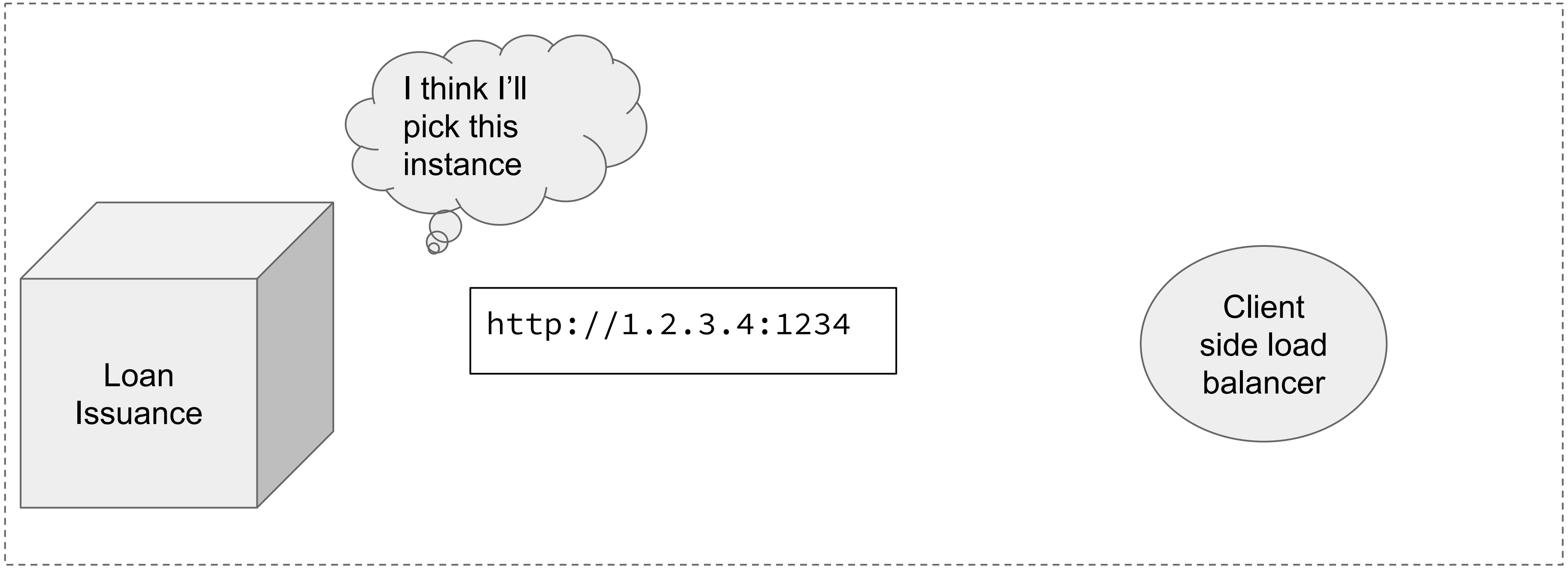
Client side

Server side

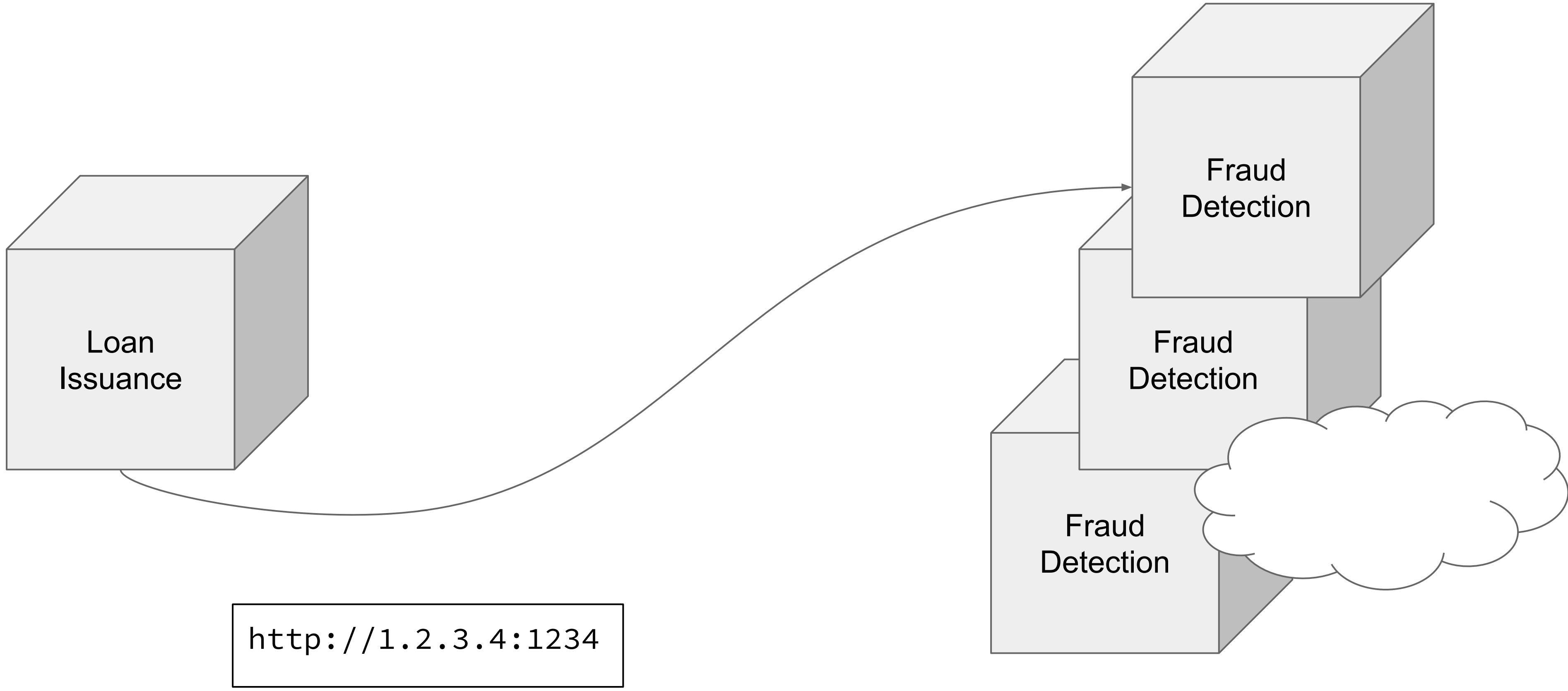



http://?:?
http://?:?
http://?:?
Loan Issuance
Service Discovery
fraud-detection
Fraud Detection
Fraud Detection
Fraud Detection


http://?:?
Loan Issuance
Client side load balancer
fraud-detection



```
http://1.2.3.4:1234
```


Loan Issuance
Fraud Detection
Fraud Detection
Fraud Detection
http://1.2.3.4:1234

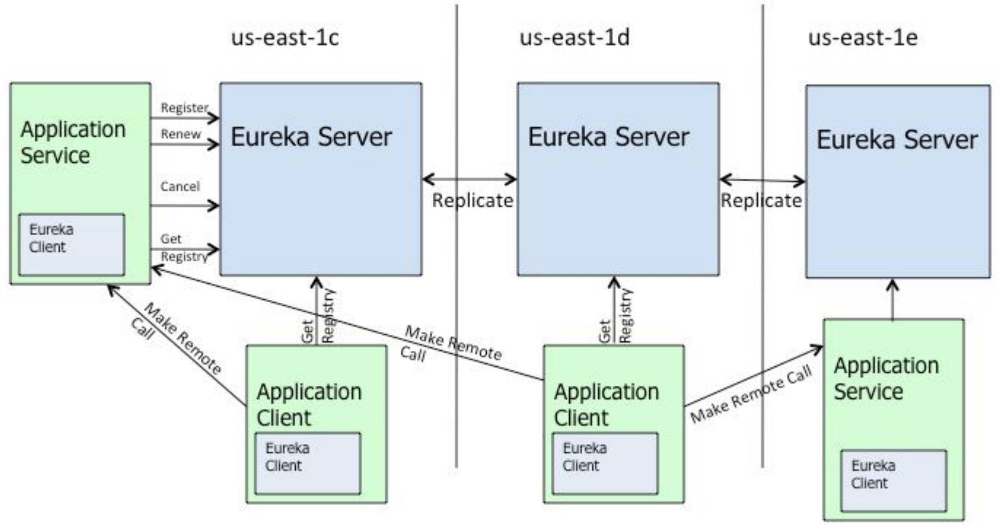
What is Netflix Eureka?

Service discovery from Netflix

Was used for AWS cloud by Netflix

Locating service, load balancing and failover

What is Netflix Eureka?



https://github.com/Netflix/eureka/wiki/Eureka-at-a-glance

What is Netflix Eureka?

Eureka Clients cache state

If server is down you will use the cache

If server down in your zone, failover to another

What is Netflix Eureka?

Eureka Server is a Eureka Client

Replicates data from peers

No peers = plenty of error log messages

SEGMENT 5

Agenda

What is Spring Cloud Netflix?

How to use Spring Cloud Netflix to perform service to service calls?

What is Spring Cloud Netflix?

Netflix OSS integrations for Spring Boot apps

integrates with

- Service Discovery (Eureka)
- Circuit Breaker (Hystrix)
- Intelligent Routing (Zuul)
- Client Side Load Balancing (Ribbon)

Neftlix Oss Deprecations

Ribbon, 2016

Hystrix Dashboard → Atlas

Zuul 1 → backward incompatible Zuul 2

Archaius 1 → backward incompatible Archaius 2

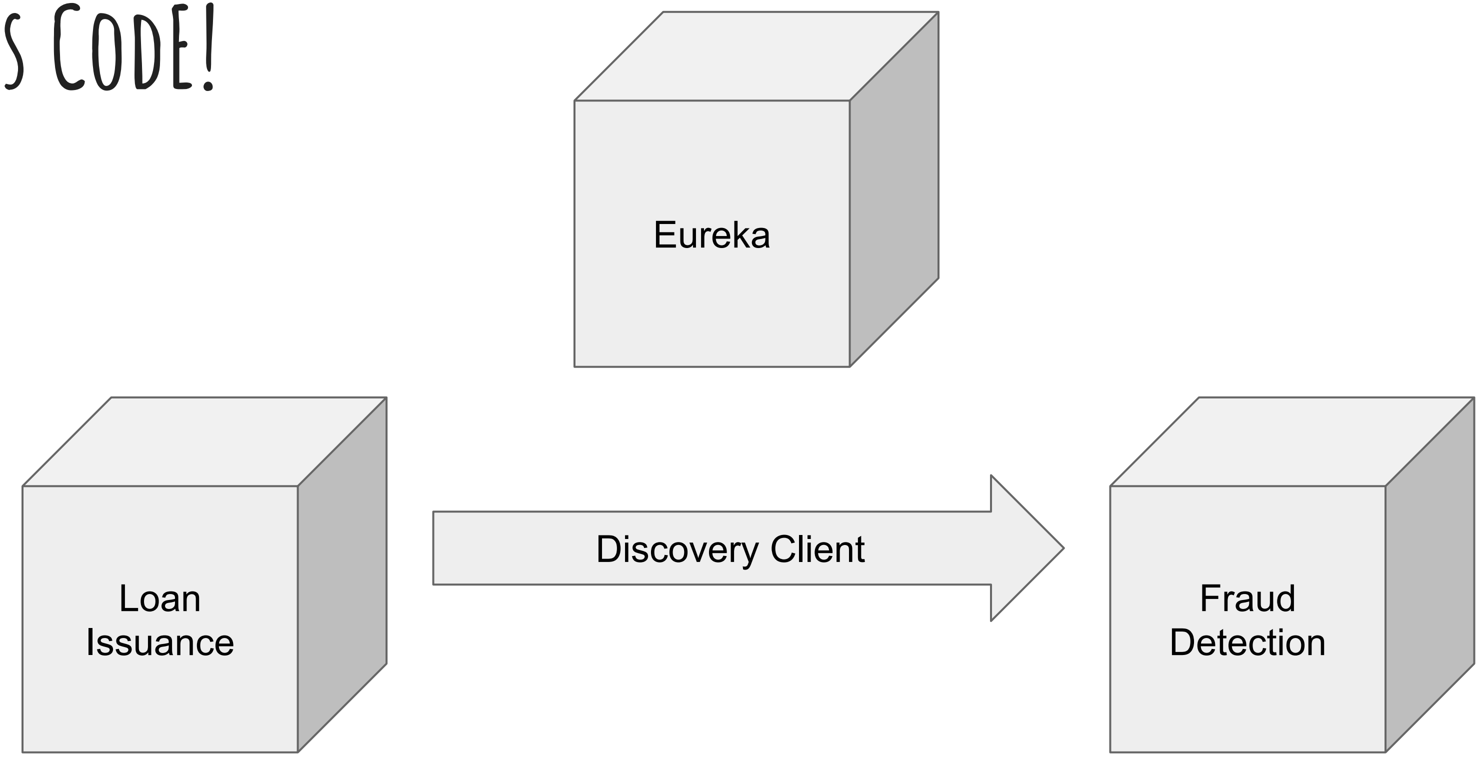
Hystrix, 2018

What is Spring Cloud Netflix?

- Spring Cloud Netflix Eureka Client
- Spring Cloud Netflix Eureka Server
- Spring Cloud Netflix Archaius
- Spring Cloud Netflix Ribbon
- Spring Cloud Netflix Zuul
- Spring Cloud Netflix Hystrix
- Spring Cloud Netflix Hystrix Dashboard
- Spring Cloud Netflix Turbine
- Spring Cloud Netflix Hystrix Stream
- Spring Cloud Netflix Turbine Stream

DEPRECATED!

Let's Code!



DEMO

SEGMENT 6

Agenda

What is Spring Cloud LoadBalancer?

What is LoadBalanced RestTemplate?

How to use different RestTemplates at the same time?

What is OpenFeign?

What is Spring Cloud LoadBalancer?

Client-side load-balancer abstraction and implementation

- `ReactiveLoadBalancer` interface
- Default is Round-Robin-based implementation

What is LoadBalanced RestTemplate?

```
new RestTemplate().getForObject(instance.getUri().toString() + "/frauds", List.class);
```

What is LoadBalanced RestTemplate?

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


@Autowired
RestTemplate restTemplate;
restTemplate.getForObject("http://fraud-detection/frauds", List.class);
```

What is OpenFeign?

Feign is a declarative web service client

Create an interface and annotate it

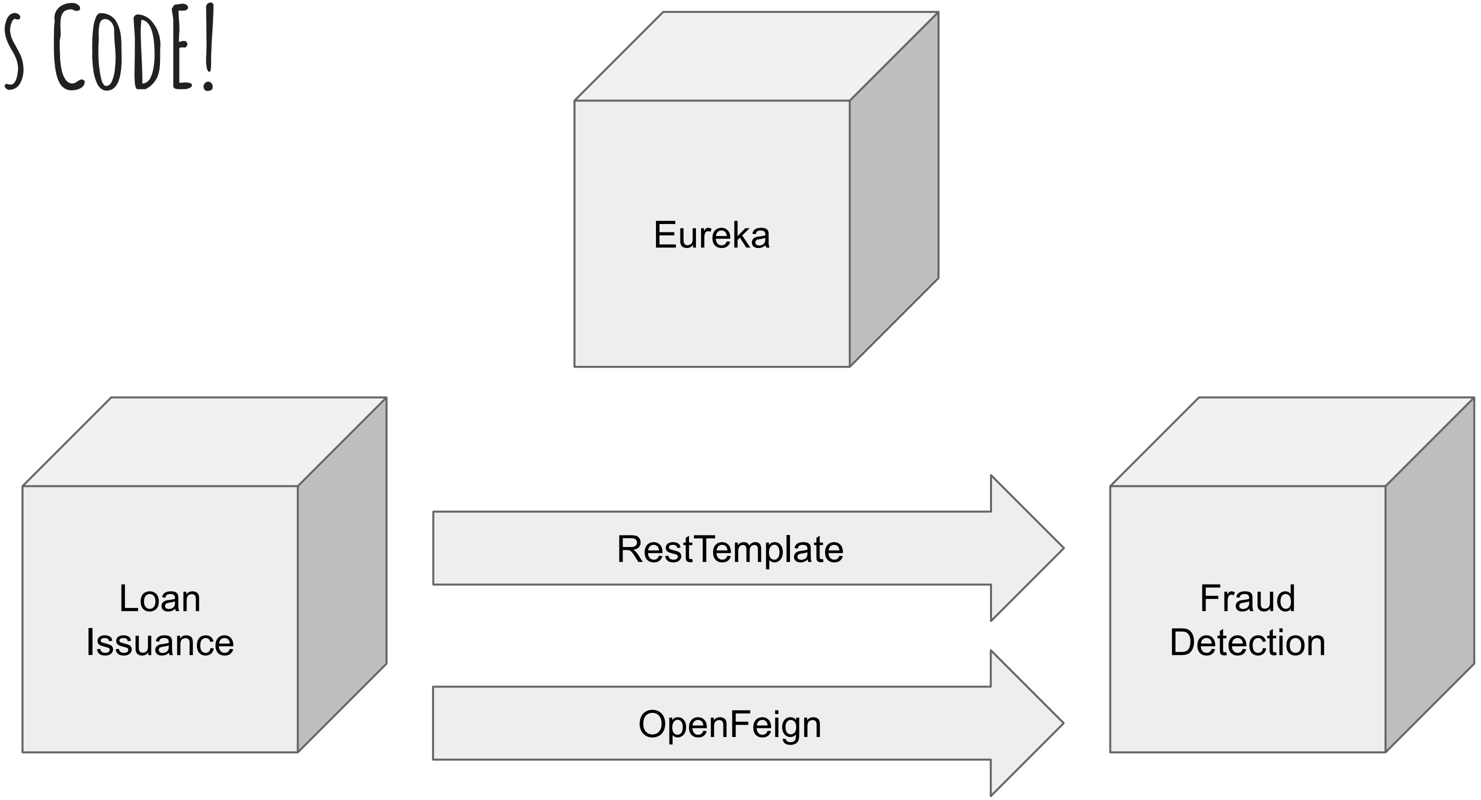
Spring Cloud adds support for Spring MVC annotations

Integrates Spring Cloud LoadBalancer

What is OpenFeign?

```java
@FeignClient("stores")
public interface StoreClient {

    @RequestMapping(method = RequestMethod.GET, value = "/stores")
    List<Store> getStores();

    @RequestMapping(method = RequestMethod.POST,
     value = "/stores/{storeId}", consumes = "application/json")
    Store update(@PathVariable("storeId") Long storeId, Store store);

}
```

Let's Code!



DEMO

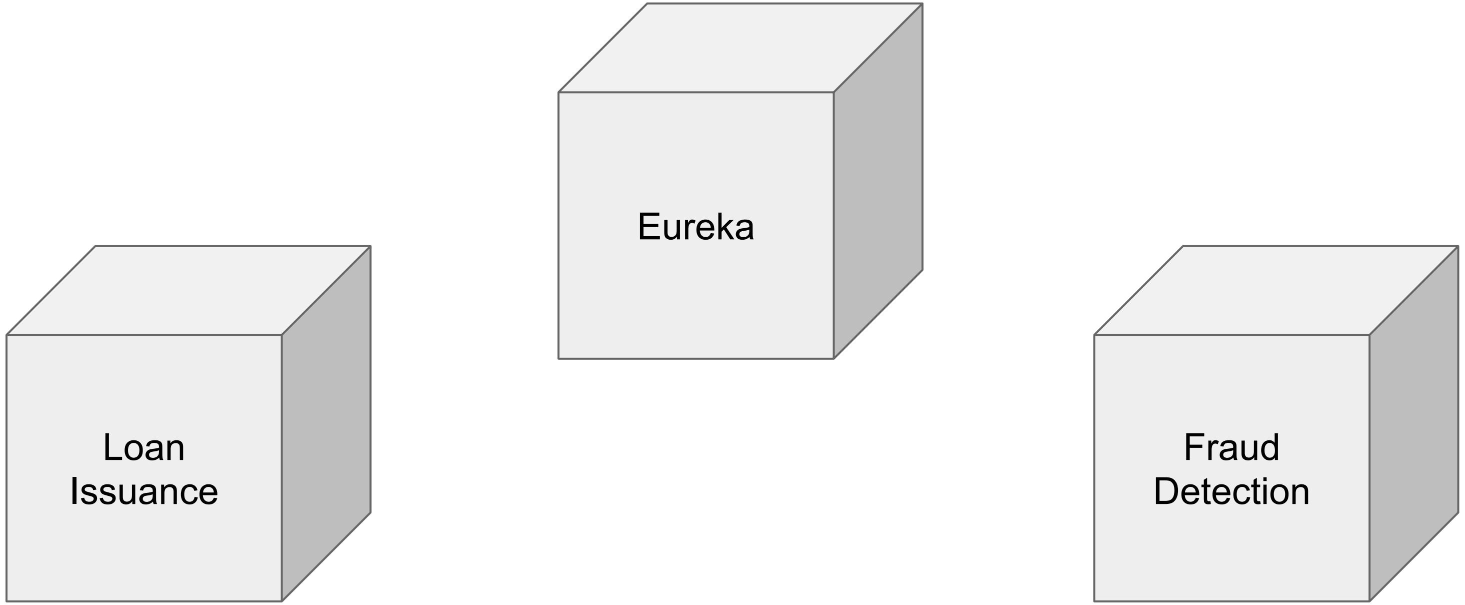
PART 2
Assignment

Part 2 - Assignment

Assignment:

Service to service communication with Spring Cloud. In this lab, students will use the Project Initializr(start.spring.io) to generate a Spring Cloud Eureka server and two client applications. Students will need to implement a REST API, make the applications register in Eureka and make the applications communicate with each other via RestTemplate and Feign.

Assignment time (15 min)


Eureka
Loan Issuance
Fraud Detection

Part 2 - Assignment

https://tinyurl.com/spring-cloud-workshops#assignment-2

https://gist.github.com/marcingrzejszczak/82a0e46f65c9ba3280dd14f395bfbf5d#assignment-2

PART 3

How can microservices gather metrics? How can microservices not cascade failure?

SEGMENT 7

Agenda

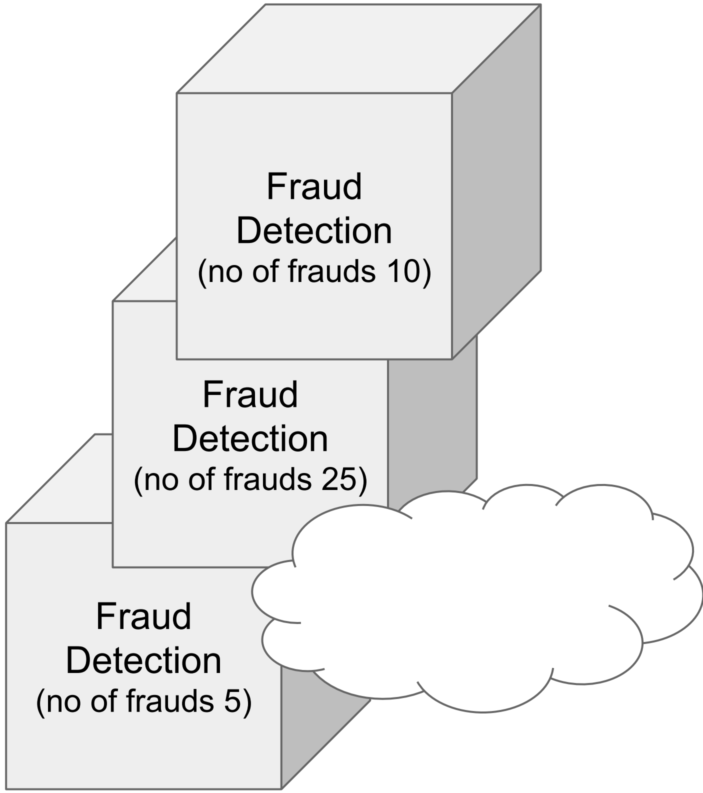
What are the issues related to microservice metrics gathering and aggregation?

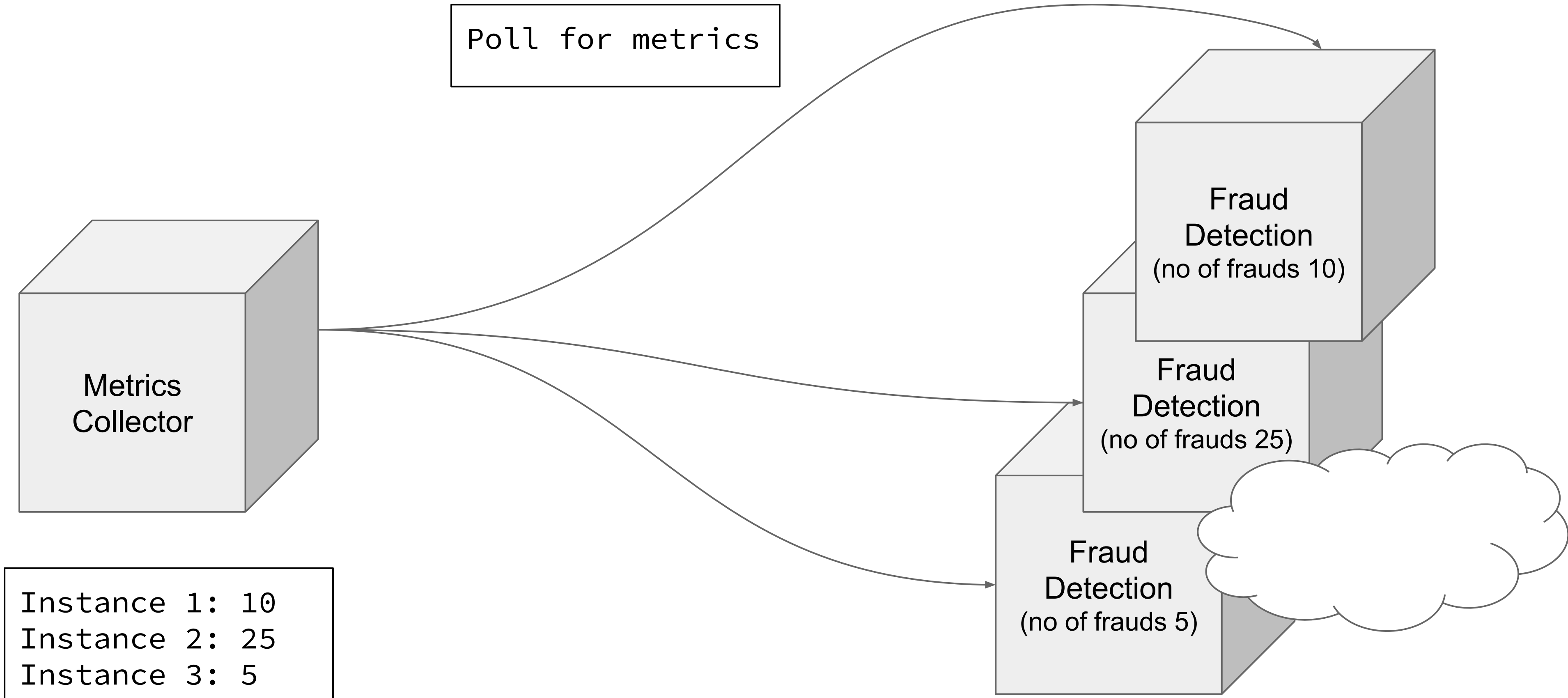
How does Actuator help in that?

What is project Micrometer?

What is Prometheus and Grafana?

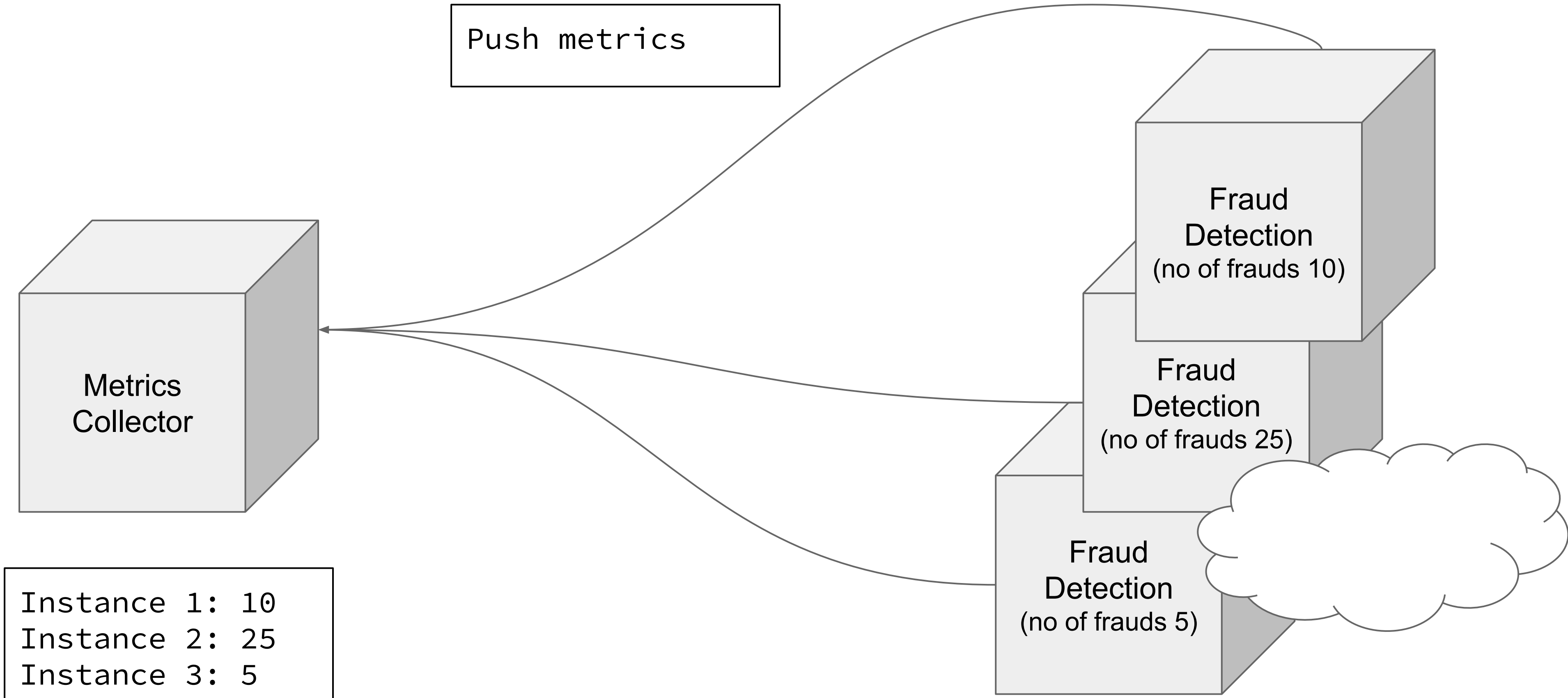
```
Total: 40
```


Fraud Detection (no of frauds 10)
Fraud Detection (no of frauds 25)
Fraud Detection (no of frauds 5)

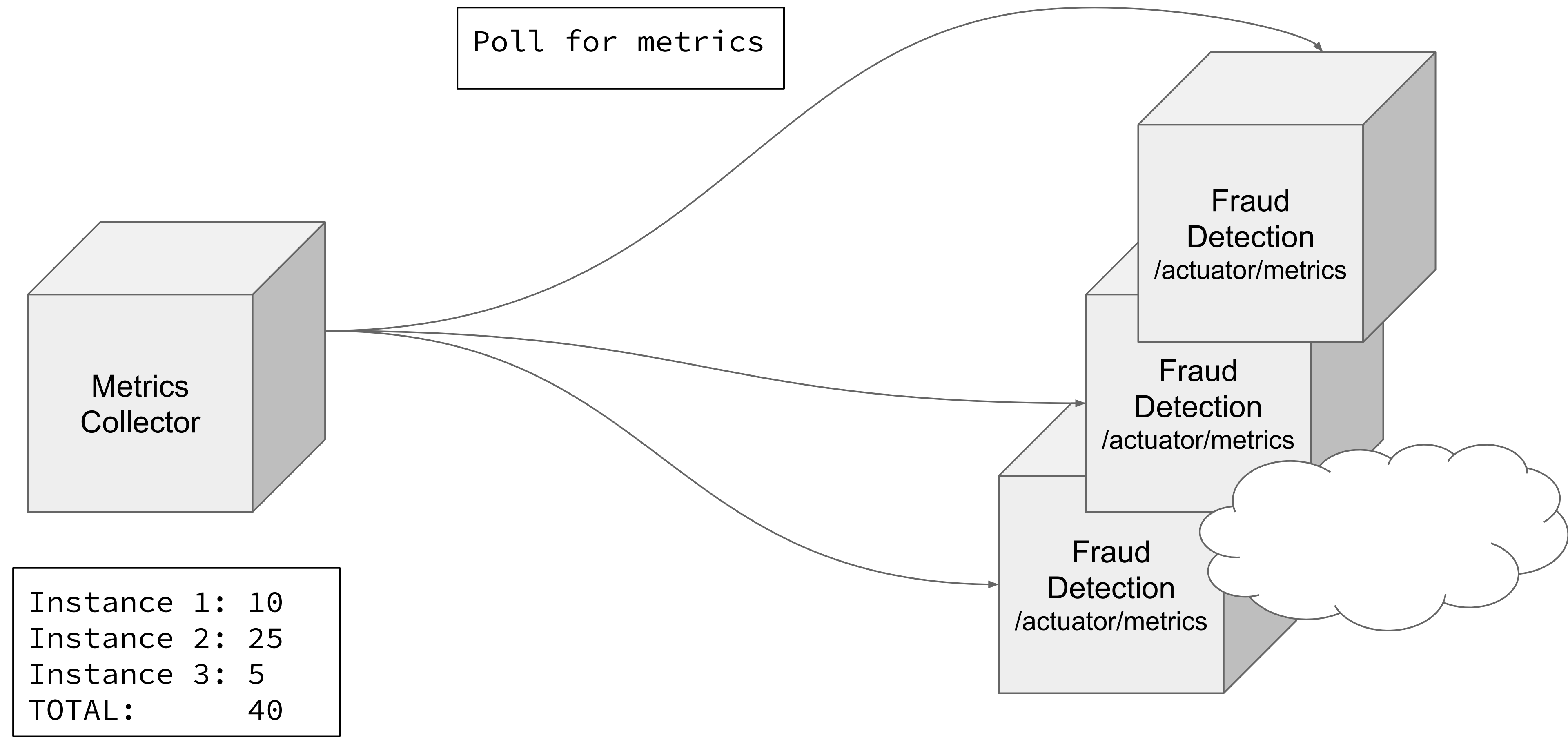

Poll for metrics
Metrics Collector
Fraud Detection (no of frauds 10)
Fraud Detection (no of frauds 25)
Fraud Detection (no of frauds 5)

```
Instance 1: 10
Instance 2: 25
Instance 3: 5
TOTAL:      40
```

```
Push metrics
```


Metrics Collector
Fraud Detection (no of frauds 10)
Fraud Detection (no of frauds 25)
Fraud Detection (no of frauds 5)

```
Instance 1: 10
Instance 2: 25
Instance 3: 5
TOTAL:      40
```


Poll for metrics
Metrics Collector
Fraud Detection /actuator/metrics
Fraud Detection /actuator/metrics
Fraud Detection /actuator/metrics

```
Instance 1: 10
Instance 2: 25
Instance 3: 5
TOTAL:      40
```

What is Spring Boot Actuator?

Beyond 12 factor app: 14. Telemetry - Application treated like a space probe

Monitor and manage your application when you push it to production

    HTTP endpoints

    JMX

Auditing, health, and metrics

What is Spring Boot Actuator?

Metrics endpoint

- Not available by default
- Must be exposed
  - **management.endpoints.web.exposure.include=metrics**

/actuator/metrics

- List of available meter names
- **/actuator/metrics/{metric-name}** e.g. jvm.memory.max

Integrates with Micrometer

What is Micrometer?

Vendor-neutral application metrics facade

Instrument your app without vendor lock-in

Micrometer as an API over your monitoring system

What is Micrometer?

Brings dimensional metrics

Basing on the classpath ships metrics to a given backend

Support for AppOptics, Netflix Atlas, Prometheus...

What is Prometheus?

Open-source systems monitoring and alerting toolkit

Originally built at SoundCloud

What is Grafana?

The open observability platform

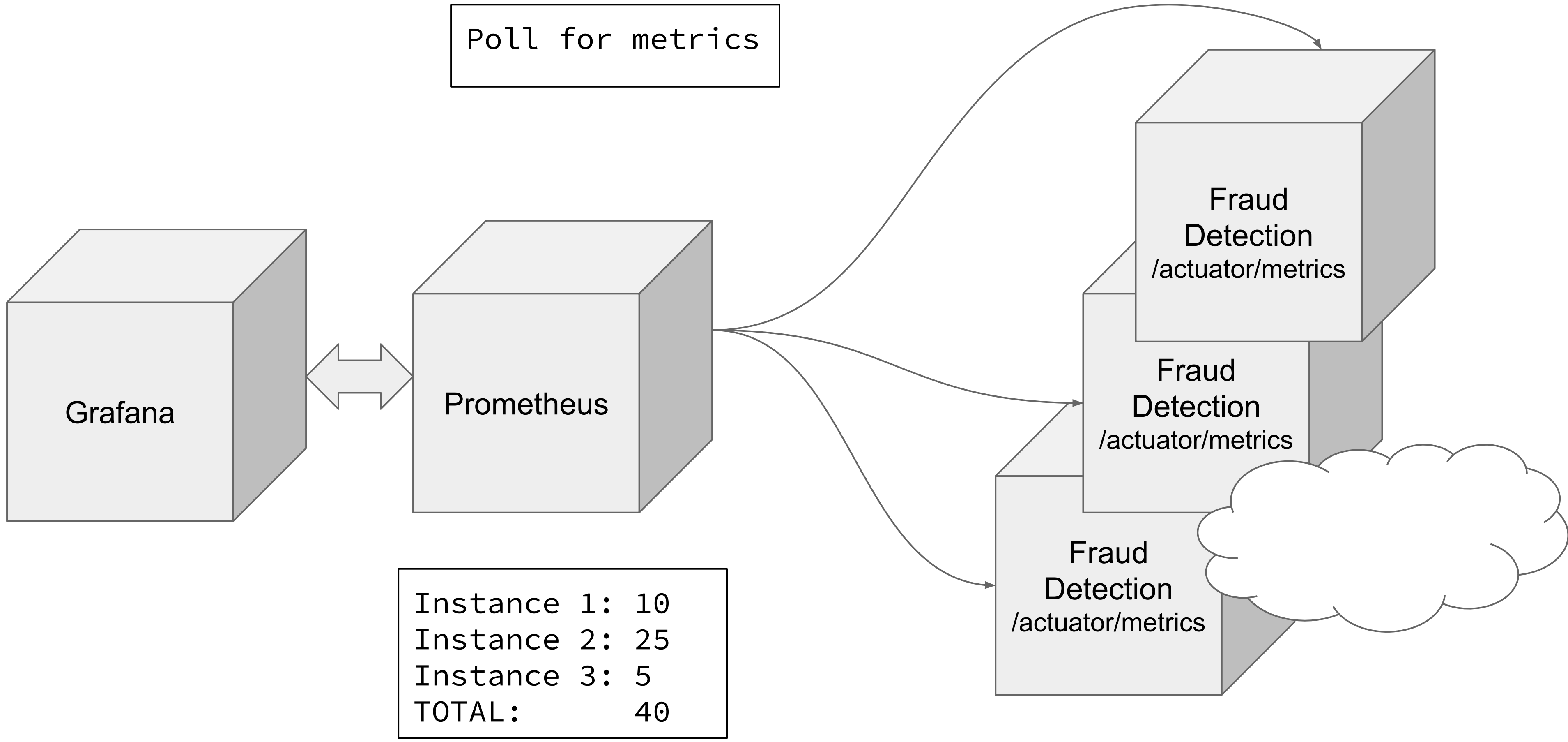
Query, visualize, alert on and understand your metrics

Consumes metrics from various databases

Allows to create, explore, and share dashboards

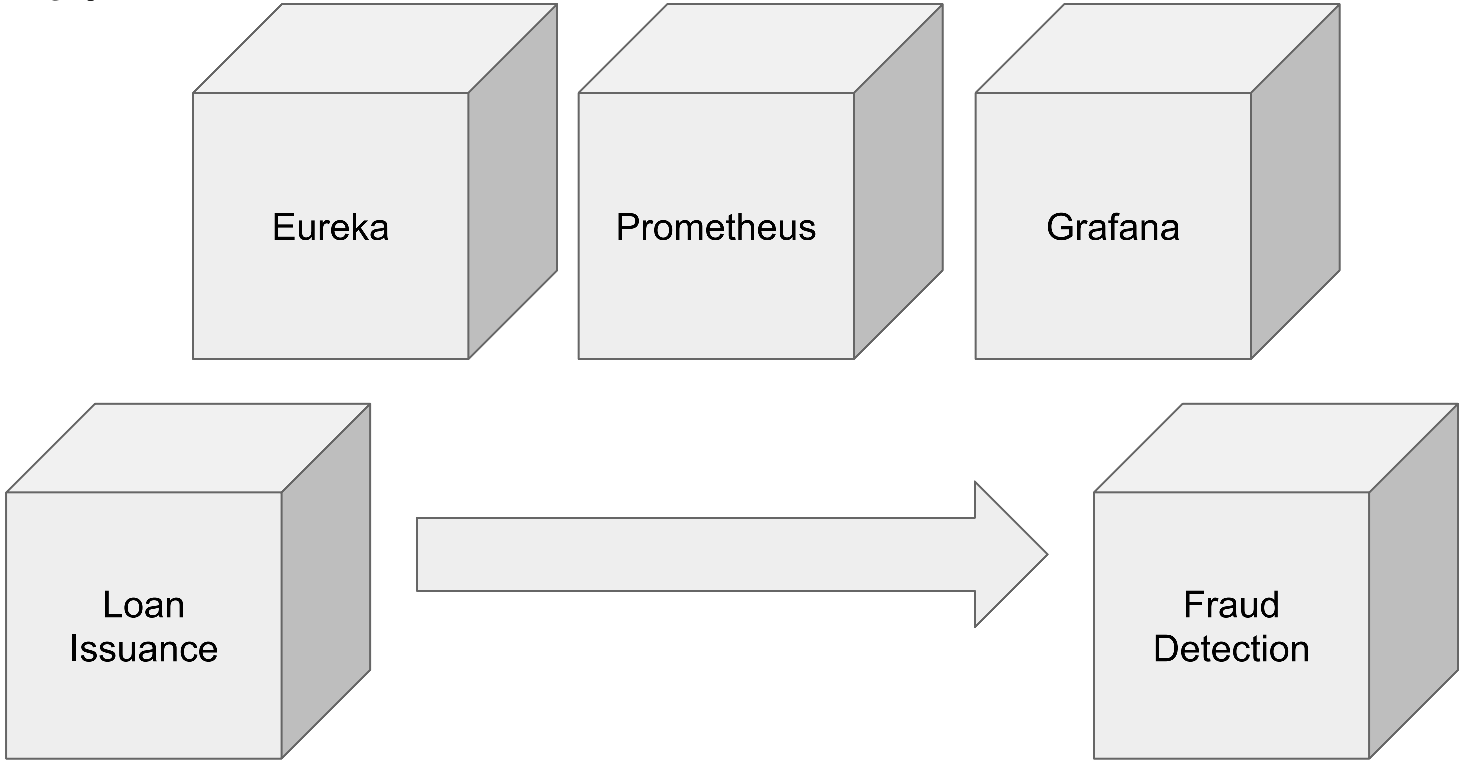
Micrometer & Prometheus & Grafana

```xml
<dependency>
     <groupId>io.micrometer</groupId>
     <artifactId>micrometer-registry-prometheus</artifactId>
 </dependency>
```


Poll for metrics
Grafana
Prometheus
Fraud Detection /actuator/metrics
Fraud Detection /actuator/metrics
Fraud Detection /actuator/metrics

```
Instance 1: 10
Instance 2: 25
Instance 3: 5
TOTAL:      40
```

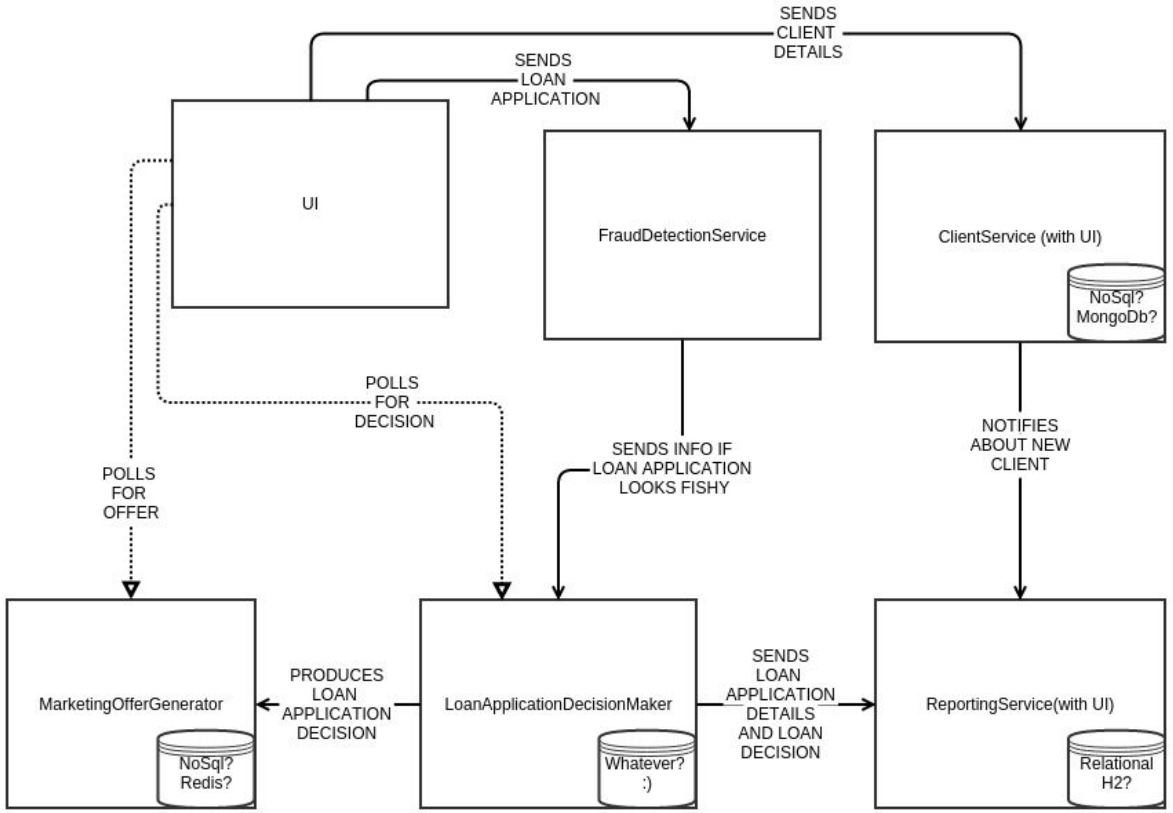
Let's Code



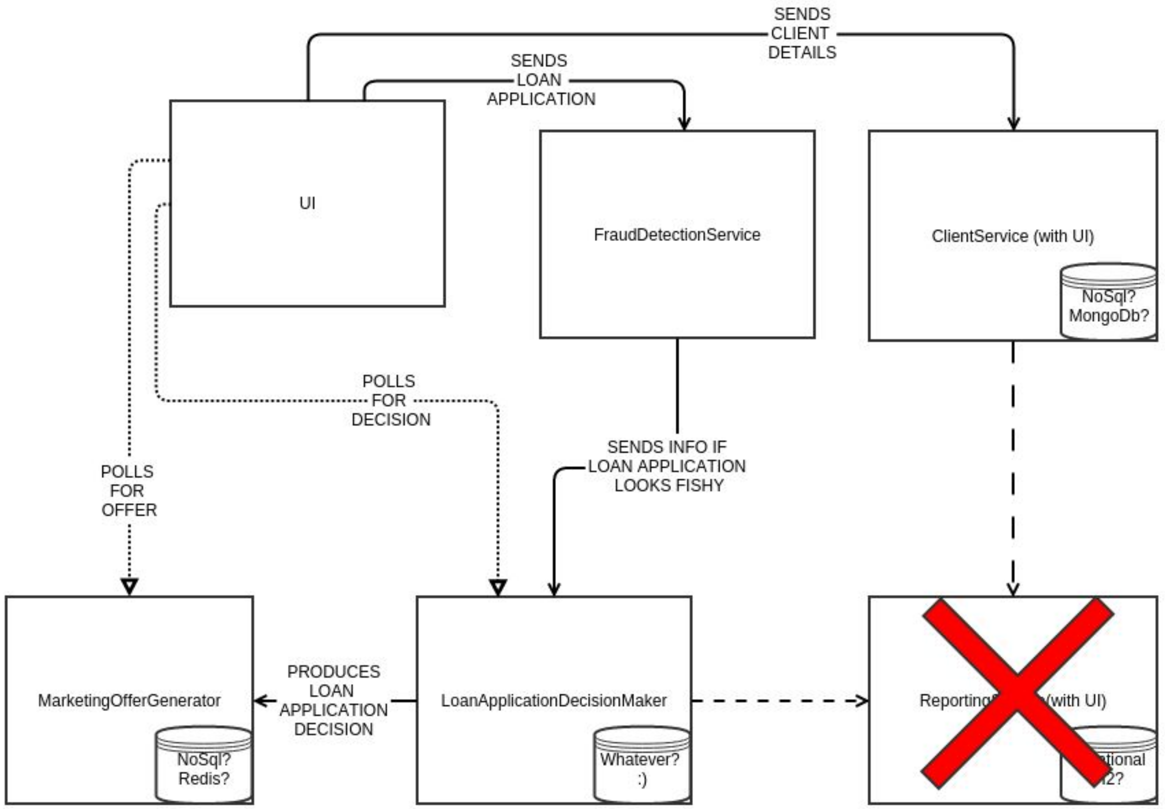
DEMO

SEGMENT 8

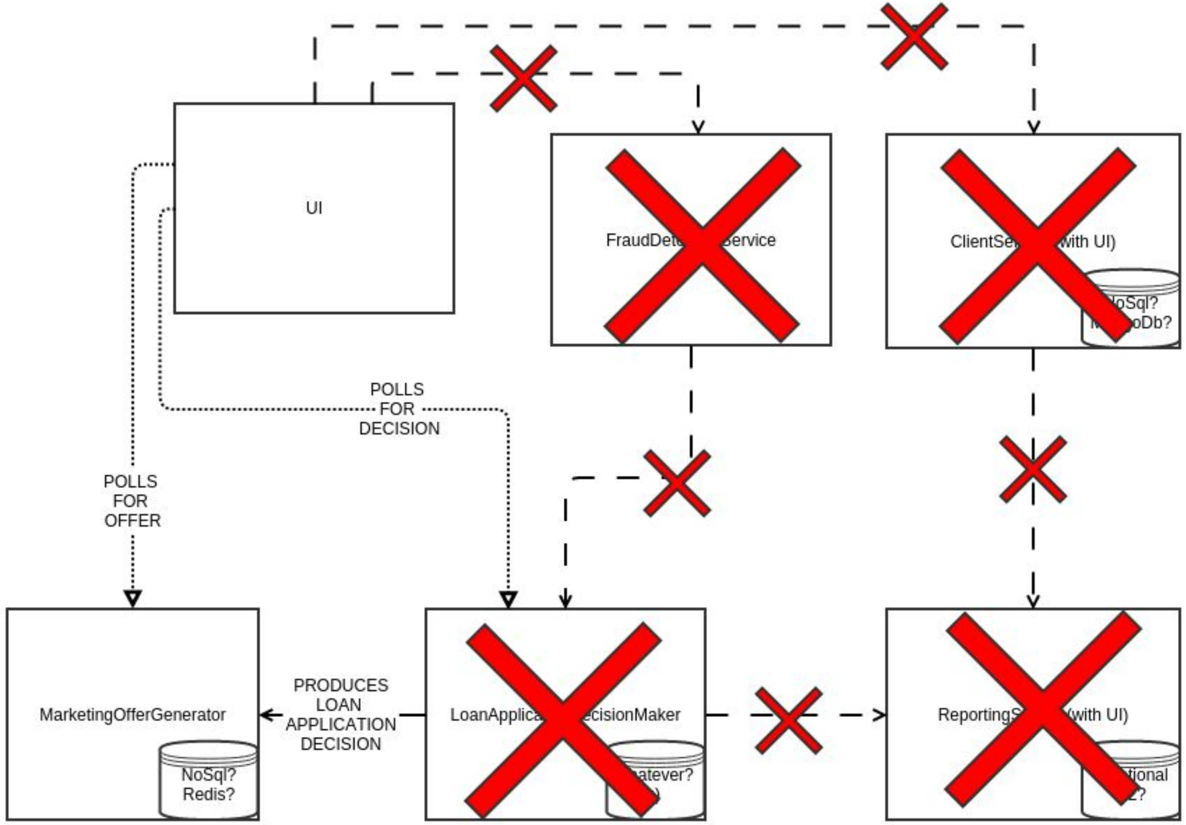
How can microservices not cascade failure?


SENDS CLIENT DETAILS
SENDS LOAN APPLICATION
UI
FraudDetectionService
ClientService (with UI)
NoSql? MongoDb?
POLLS FOR DECISION
POLLS FOR OFFER
SENDS INFO IF LOAN APPLICATION LOOKS FISHY
NOTIFIES ABOUT NEW CLIENT
MarketingOfferGenerator
NoSql? Redis?
PRODUCES LOAN APPLICATION DECISION
LoanApplicationDecisionMaker
Whatever? :)
SENDS LOAN APPLICATION DETAILS AND LOAN DECISION
ReportingService(with UI)
Relational H2?

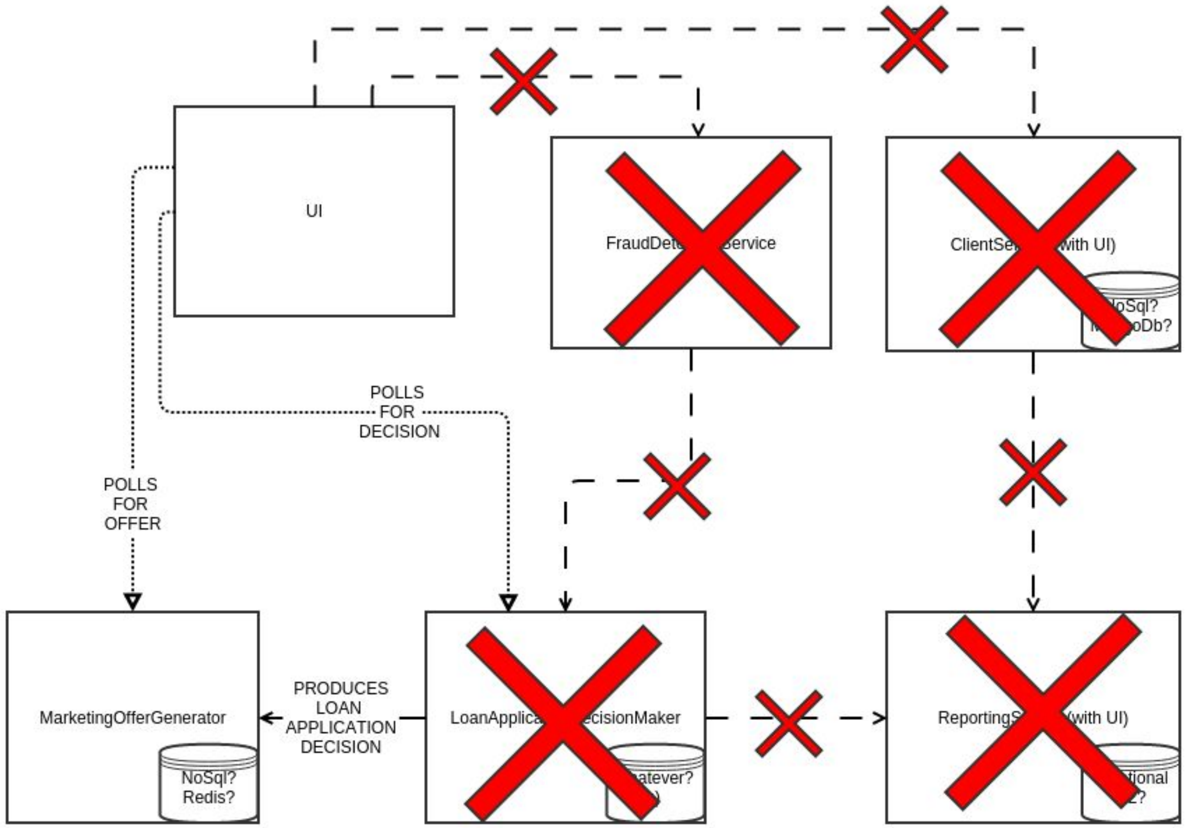
How can microservices not cascade failure?



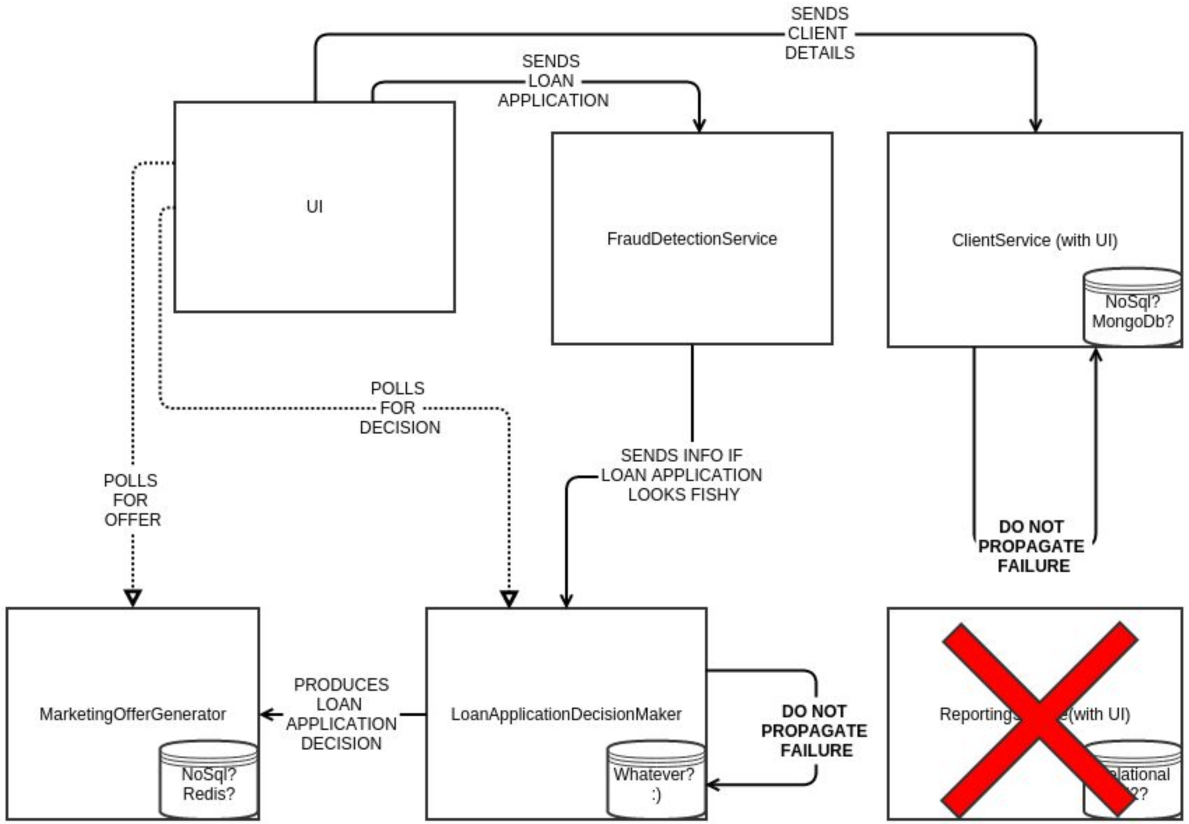
How can microservices not cascade failure?



How can microservices not cascade failure?



How can microservices not cascade failure?



What is a Circuit Breaker?

Design pattern

Detect failures

Prevent a failure from constantly recurring

What is a Circuit Breaker?

- Implementations
  - Netflix Hystrix (maintenance mode)
  - Resilience4J
- Abstractions
  - Spring Cloud CircuitBreaker

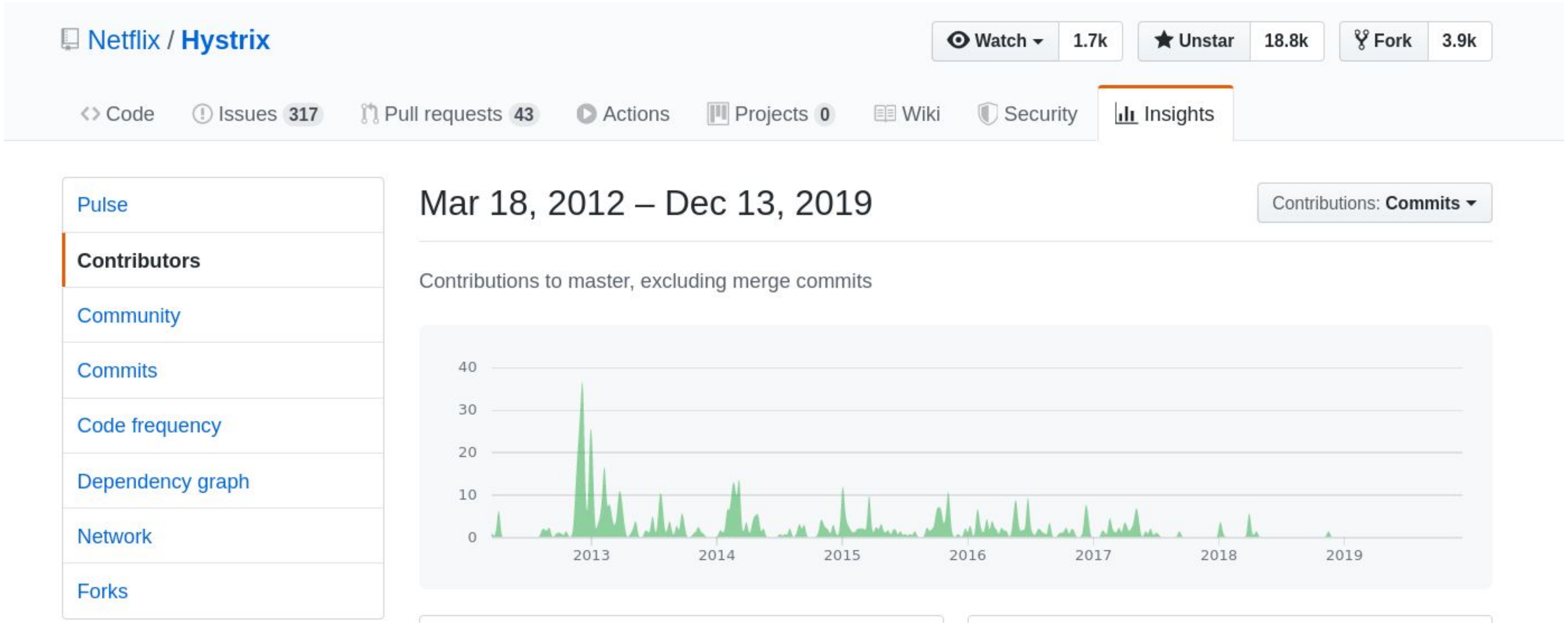
What is Netflix Hystrix?

Initial work began in 2011 in one of Netflix teams

In 2012 Netflix adopted it internally

In 2018 Hystrix put in maintenance mode

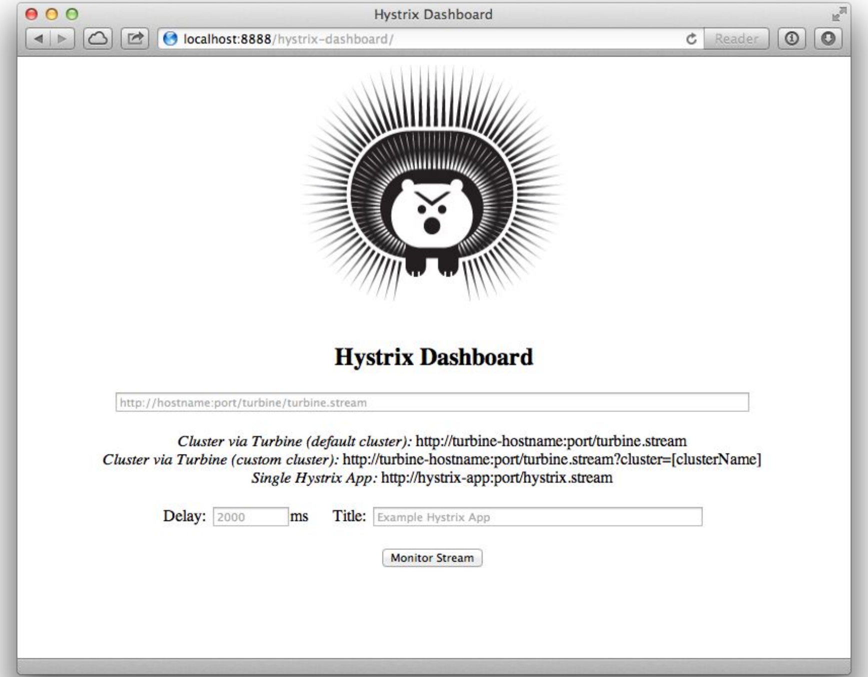
What is Netflix Hystrix?



Mar 18, 2012 – Dec 13, 2019

Contributions to master, excluding merge commits

What is Netflix Hystrix?



Hystrix Dashboard

http://hostname:port/turbine/turbine.stream

Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: http://hystrix-app:port/hystrix.stream

Delay: 2000 ms Title: Example Hystrix App

Monitor Stream

Resilience4J

Lightweight fault tolerance library inspired by Hystrix

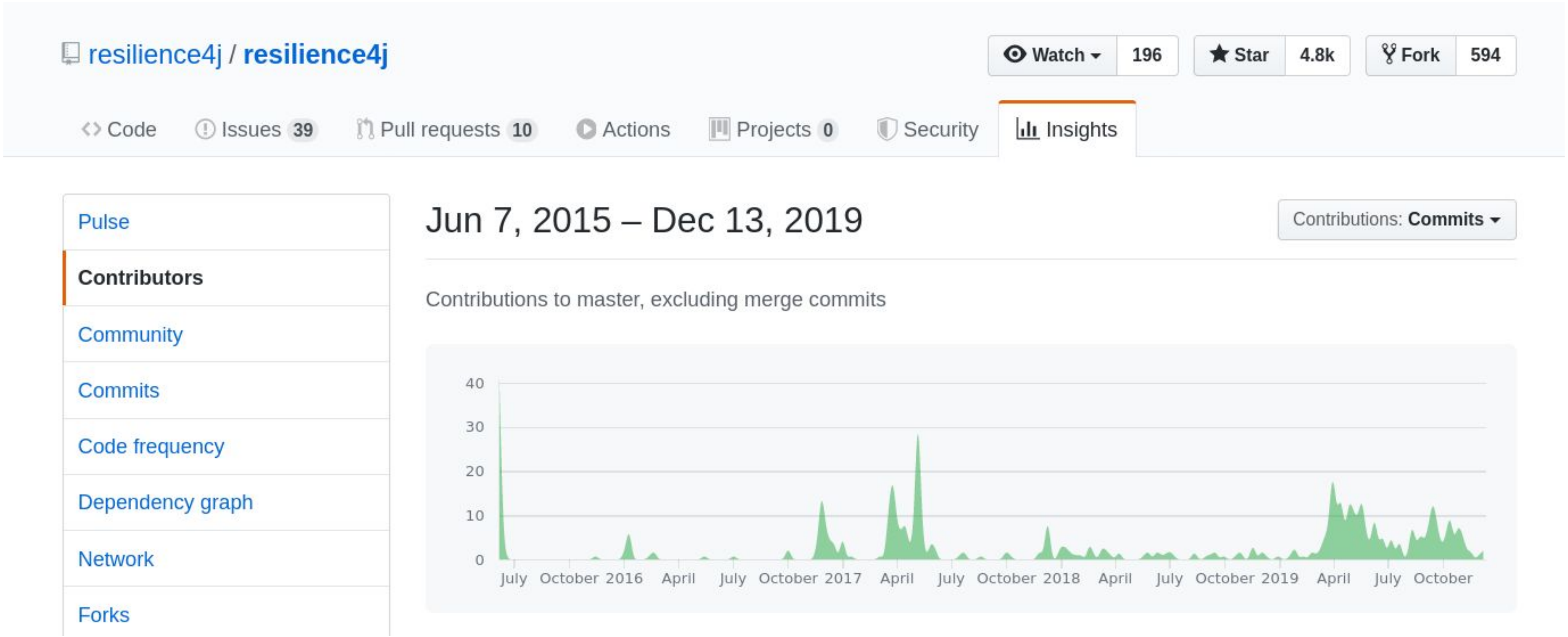
- Designed for Java 8 and functional programming

Uses Vavr as the only dependency

Resilience4j provides

- Circuit Breaker
- Rate Limiter
- Retry
- Bulkhead

Resilience4J



resilience4j / resilience4j

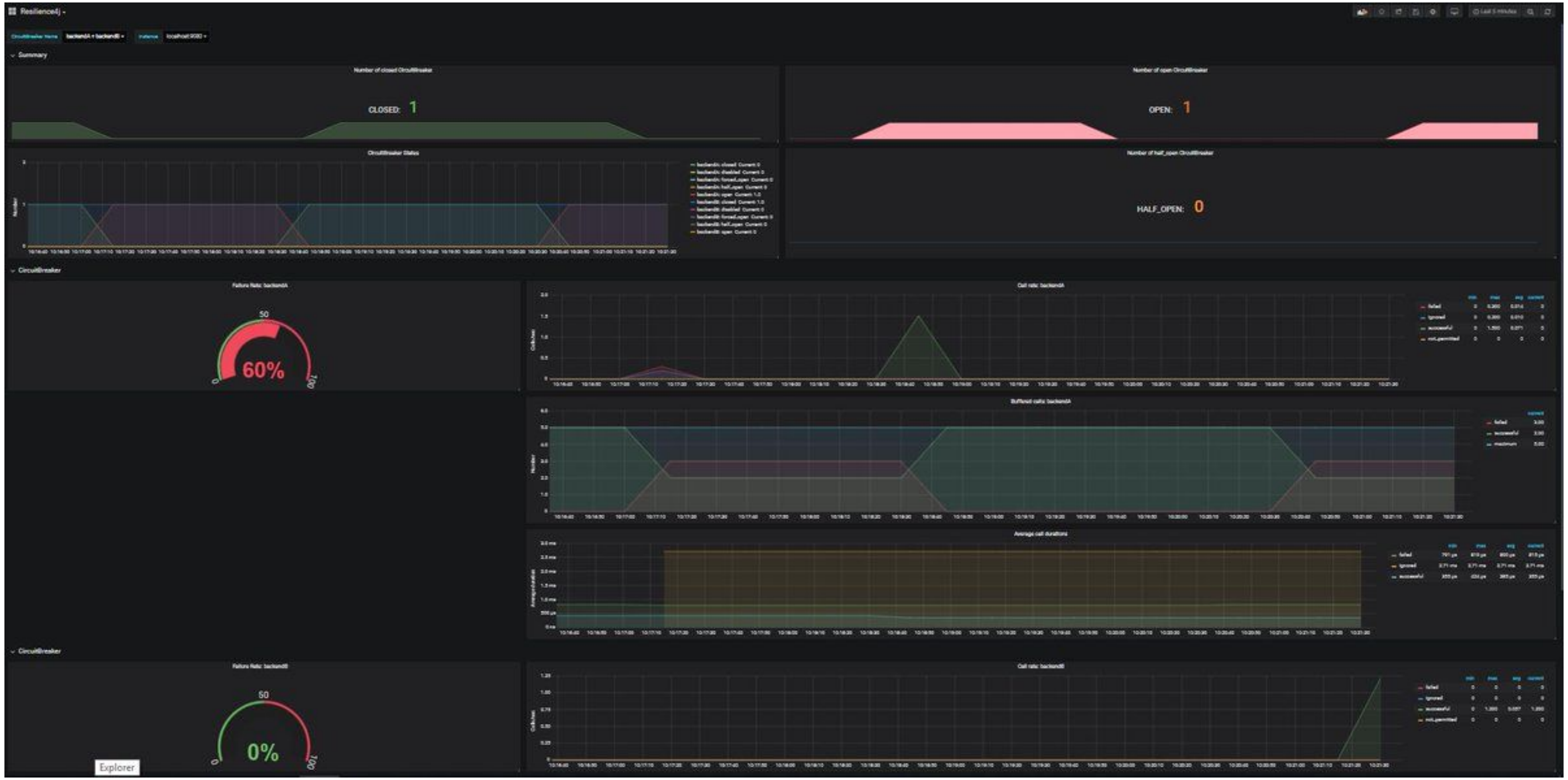
Jun 7, 2015 – Dec 13, 2019

Contributions to master, excluding merge commits

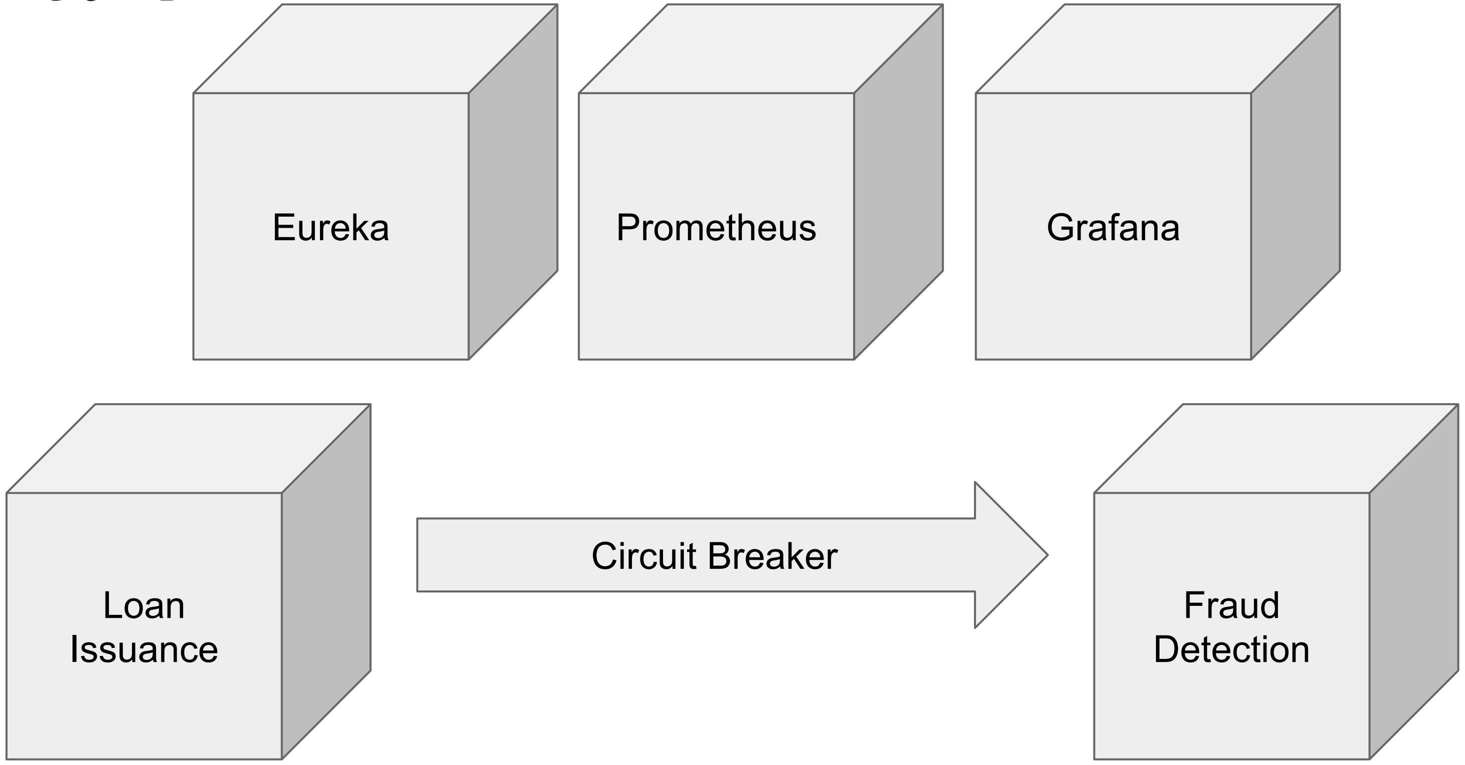
Resilience4J-Micrometer & Prometheus & Grafana

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
 </dependency>
 <dependency>
    <groupId>io.github.resilience4j</groupId>
    <artifactId>resilience4j-micrometer</artifactId>
 </dependency>
```

Resilience4J-Micrometer & Prometheus & Grafana



Let's Code



DEMO

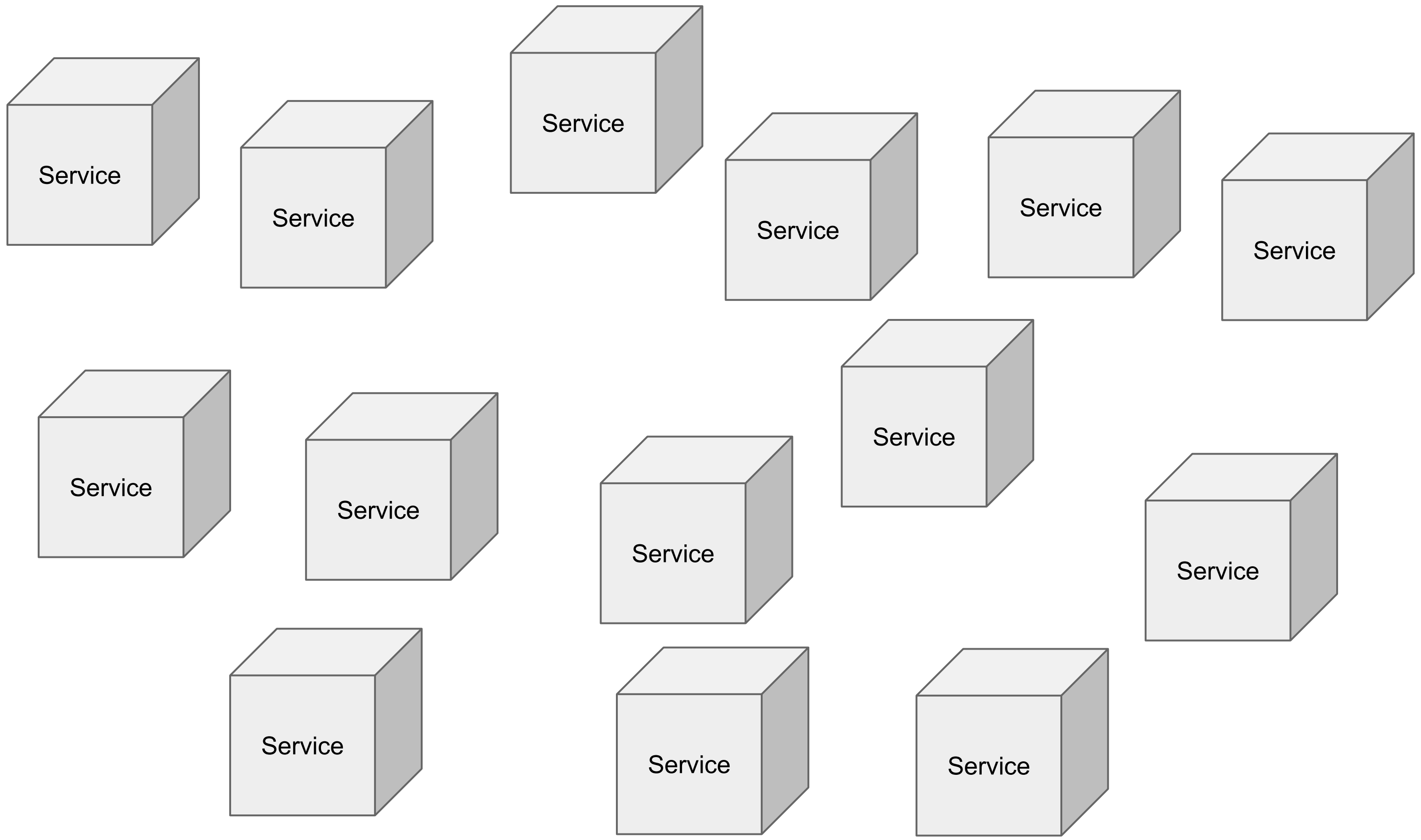
SEGMENT 9

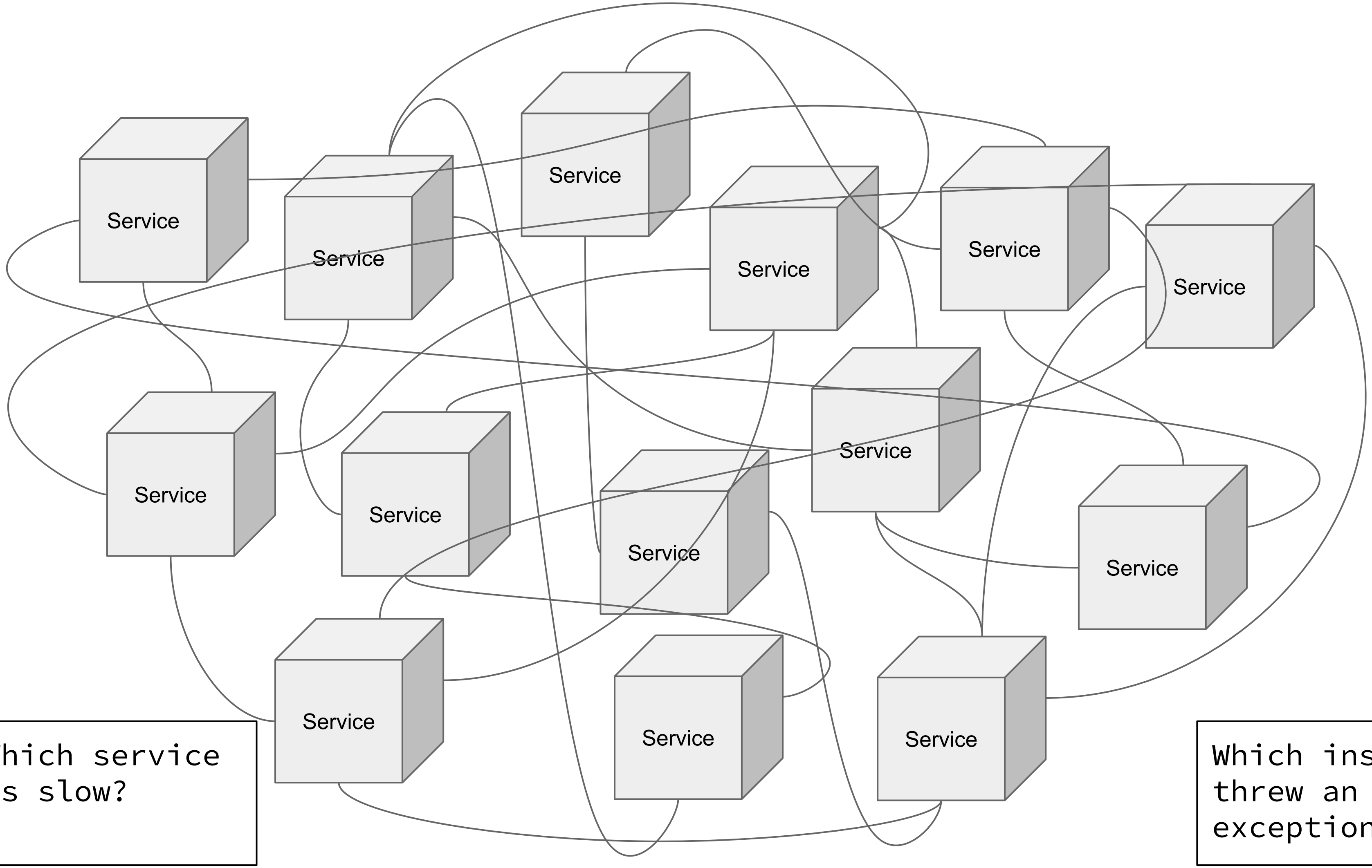
How can microservices Be Traced?

What is distributed tracing?

What is Spring Cloud Sleuth?

What is project Zipkin?


Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service
Service



Which service is slow?

Which instance threw an exception?

What is a Span?

The basic unit of work (e.g. sending RPC)

They keep track of their timing information

Once you create a span, you must stop it at some point in the future

Has a parent and can have multiple children

All spans have unique span ids

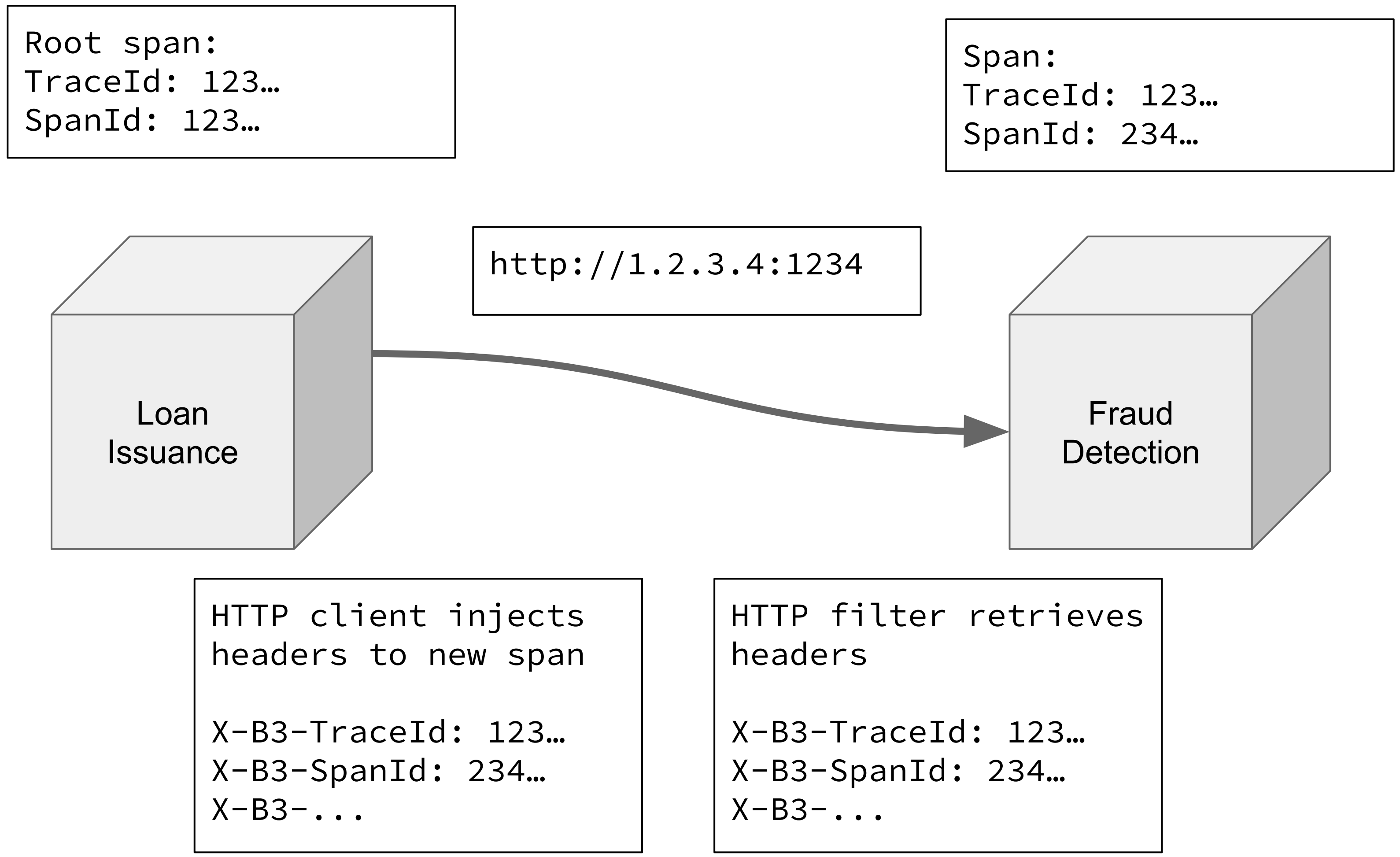
Spans in a single hierarchy share a trace id

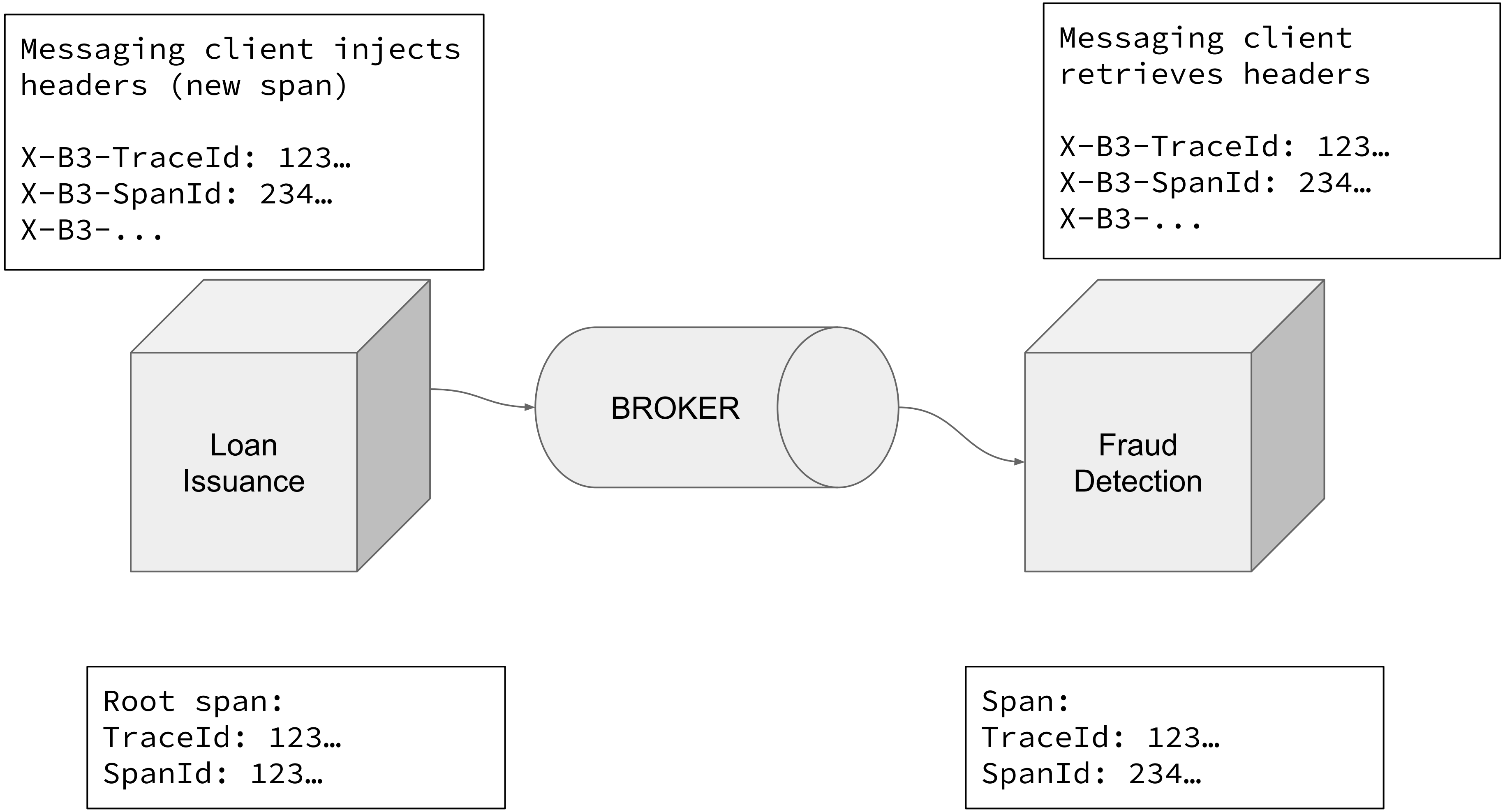
What is a Trace?

A set of spans forming a tree-like structure.

For example, if you are running a bookstore then

- Trace could be retrieving a list of available books
- Assuming that to retrieve the books you have to
  - Send 3 requests to 3 services
  - You could have at least 3 spans (1 for each hop)
  - Forming 1 trace


Root span:
TraceId: 123…
SpanId: 123…
Span:
TraceId: 123…
SpanId: 234…
Loan Issuance
http://1.2.3.4:1234
Fraud Detection
HTTP client injects headers to new span
X-B3-TraceId: 123…
X-B3-SpanId: 234…
X-B3-...
HTTP filter retrieves headers
X-B3-TraceId: 123…
X-B3-SpanId: 234…
X-B3-...



```
Messaging client injects
headers (new span)

X-B3-TraceId: 123…
X-B3-SpanId: 234…
X-B3-...
```

```
Messaging client
retrieves headers

X-B3-TraceId: 123…
X-B3-SpanId: 234…
X-B3-...
```

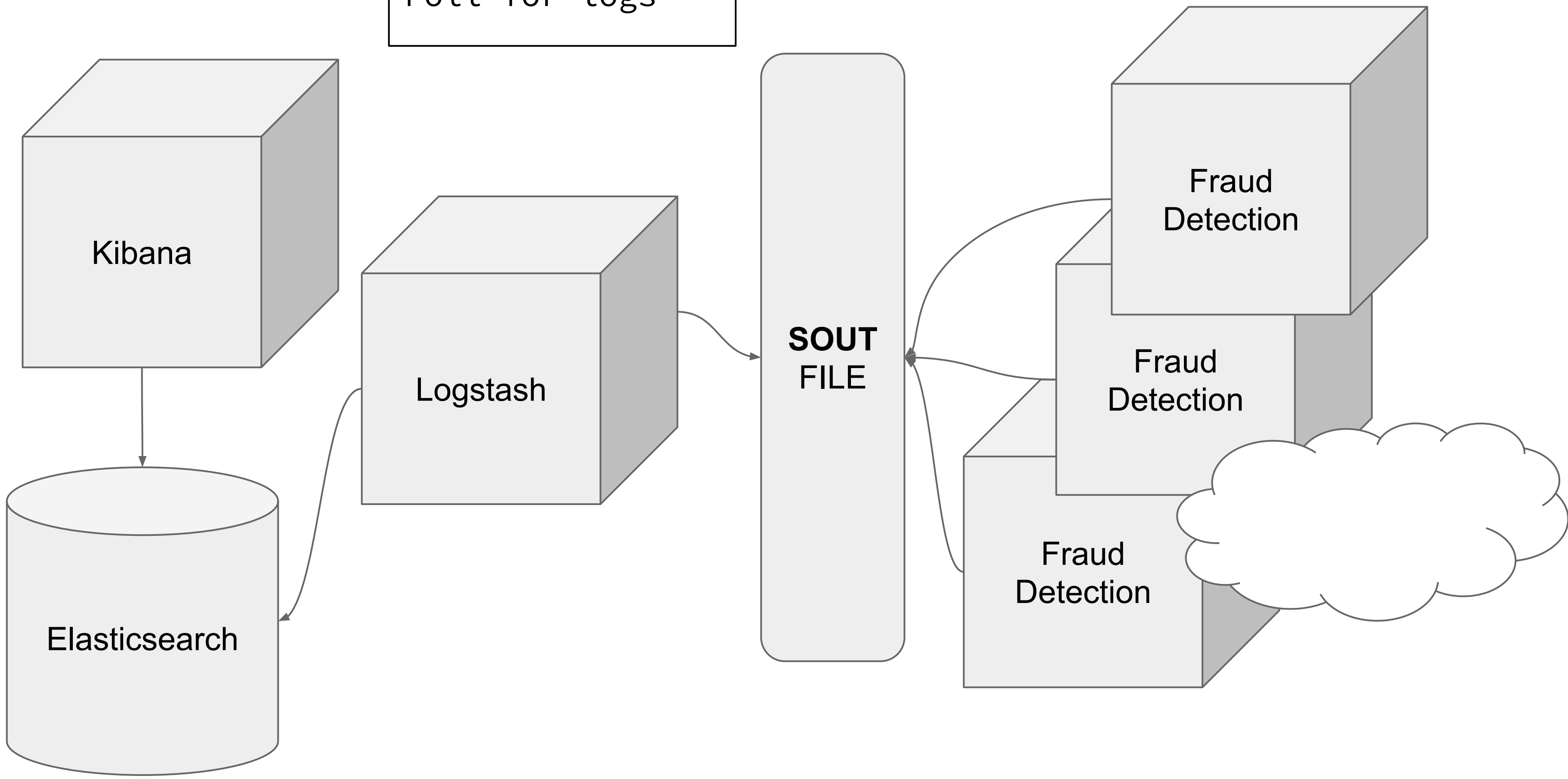
```
Root span:
TraceId: 123…
SpanId: 123…
```

```
Span:
TraceId: 123…
SpanId: 234…
```

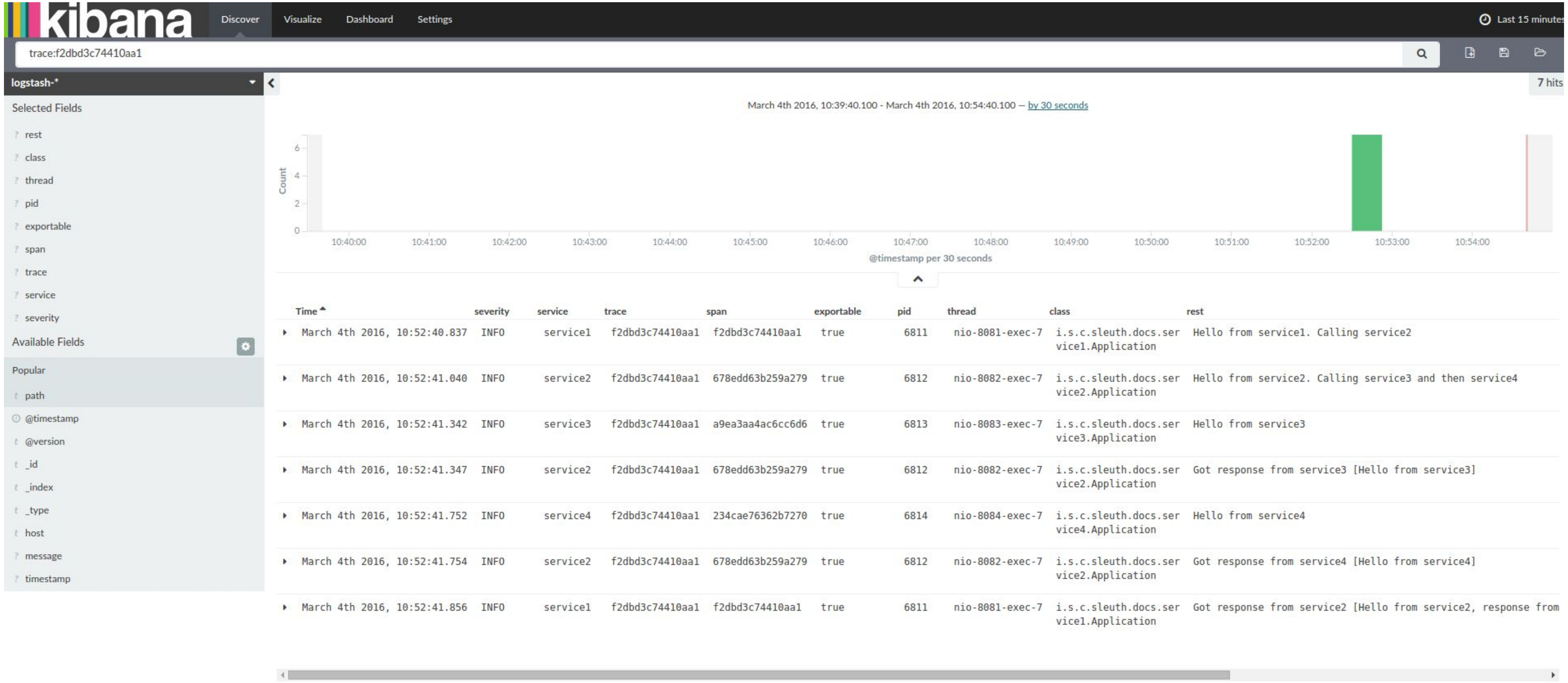
How can microservices Be Traced?

- Spring Cloud Sleuth
  - Log correlation
- Project Zipkin
  - Latency analysis

```
Poll for logs
```



How can microservices Be Traced?

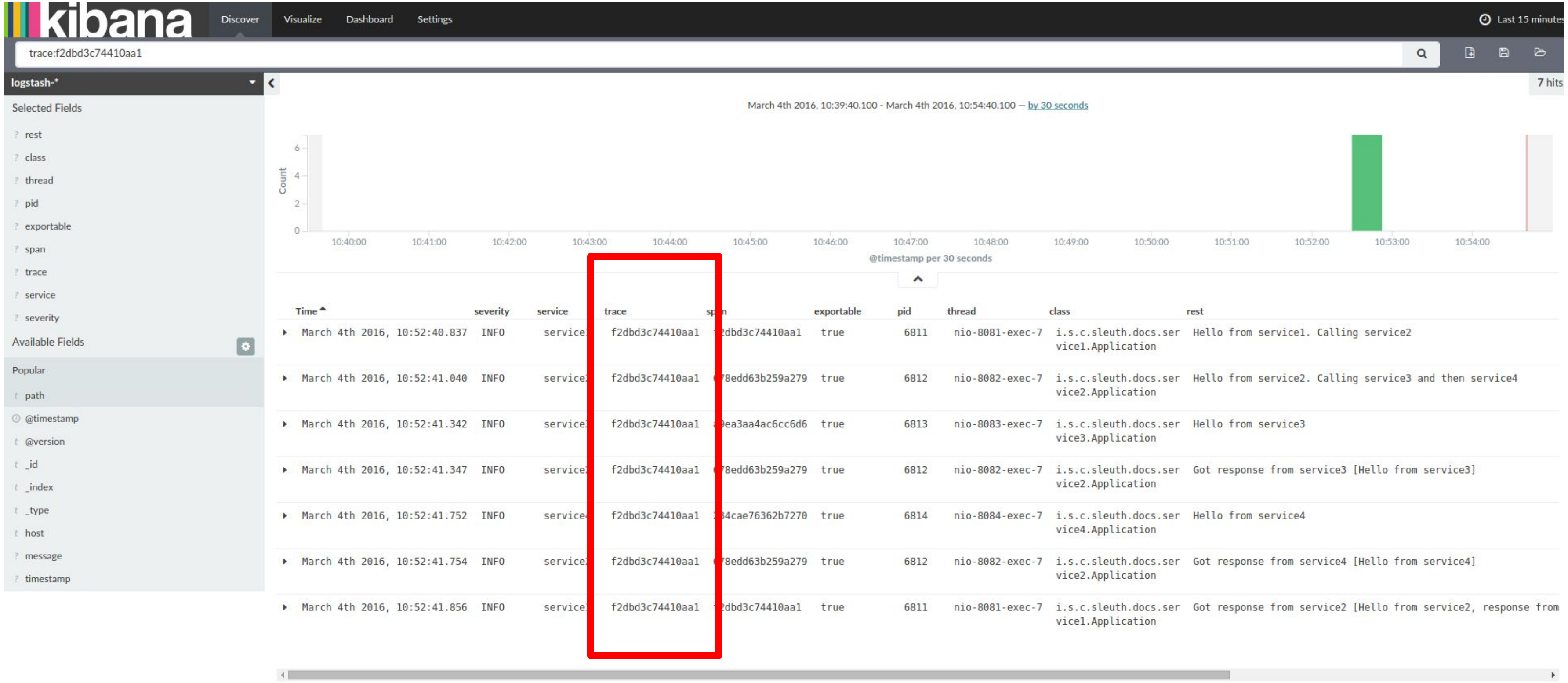


trace:f2dbd3c74410aa1

March 4th 2016, 10:39:40.100 - March 4th 2016, 10:54:40.100 — by 30 seconds

Time	severity	service	trace	span	exportable	pid	thread	class	rest
March 4th 2016, 10:52:40.837	INFO	service1	f2dbd3c74410aa1	f2dbd3c74410aa1	true	6811	nio-8081-exec-7	i.s.c.sleuth.docs.service1.Application	Hello from service1. Calling service2
March 4th 2016, 10:52:41.040	INFO	service2	f2dbd3c74410aa1	678edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Hello from service2. Calling service3 and then service4
March 4th 2016, 10:52:41.342	INFO	service3	f2dbd3c74410aa1	a9ea3aa4ac6cc6d6	true	6813	nio-8083-exec-7	i.s.c.sleuth.docs.service3.Application	Hello from service3
March 4th 2016, 10:52:41.347	INFO	service2	f2dbd3c74410aa1	678edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Got response from service3 [Hello from service3]
March 4th 2016, 10:52:41.752	INFO	service4	f2dbd3c74410aa1	234cae76362b7270	true	6814	nio-8084-exec-7	i.s.c.sleuth.docs.service4.Application	Hello from service4
March 4th 2016, 10:52:41.754	INFO	service2	f2dbd3c74410aa1	678edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Got response from service4 [Hello from service4]
March 4th 2016, 10:52:41.856	INFO	service1	f2dbd3c74410aa1	f2dbd3c74410aa1	true	6811	nio-8081-exec-7	i.s.c.sleuth.docs.service1.Application	Got response from service2 [Hello from service2, response from

How can microservices Be Traced?



trace:f2dbd3c74410aa1

March 4th 2016, 10:39:40.100 - March 4th 2016, 10:54:40.100 — by 30 seconds

Selected Fields: rest, class, thread, pid, exportable, span, trace, service, severity

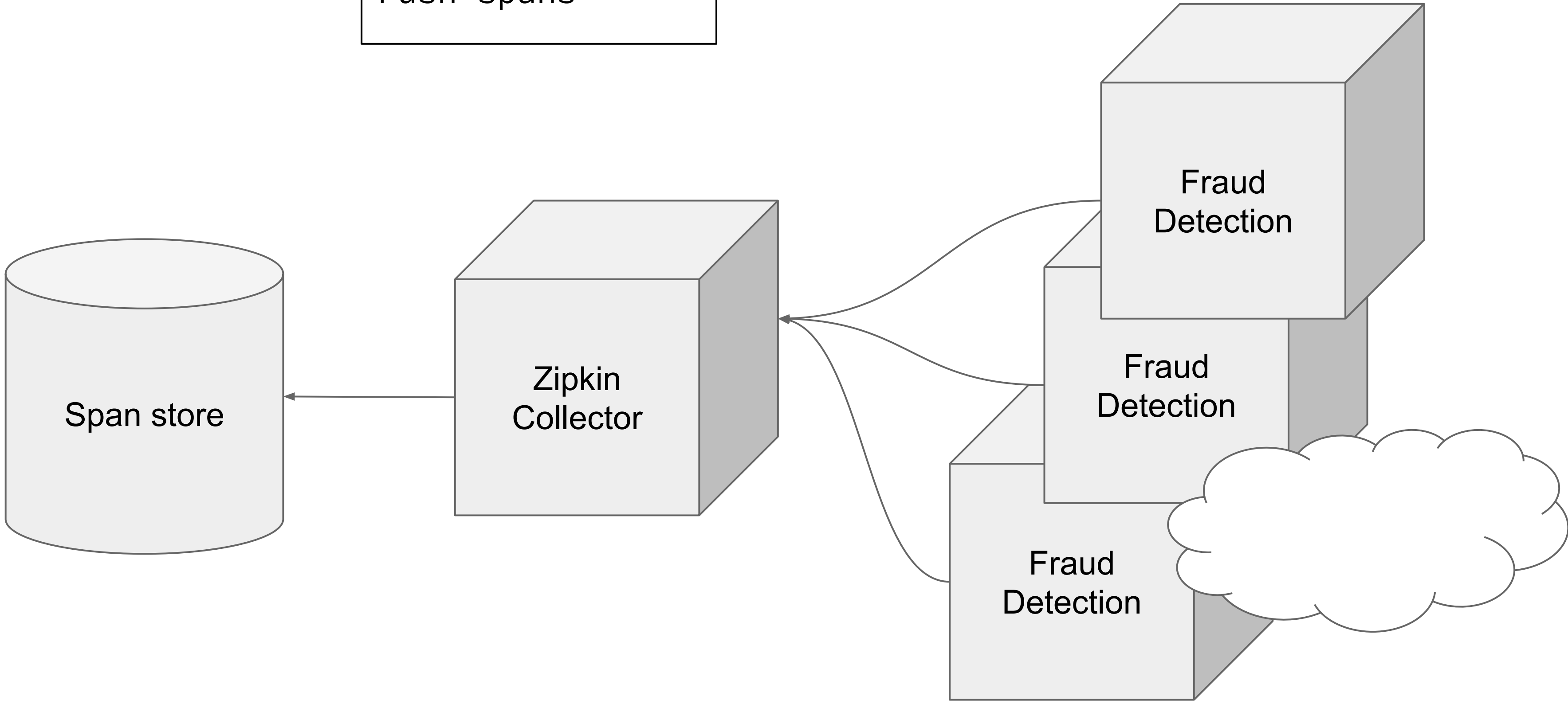
Available Fields

Popular: path

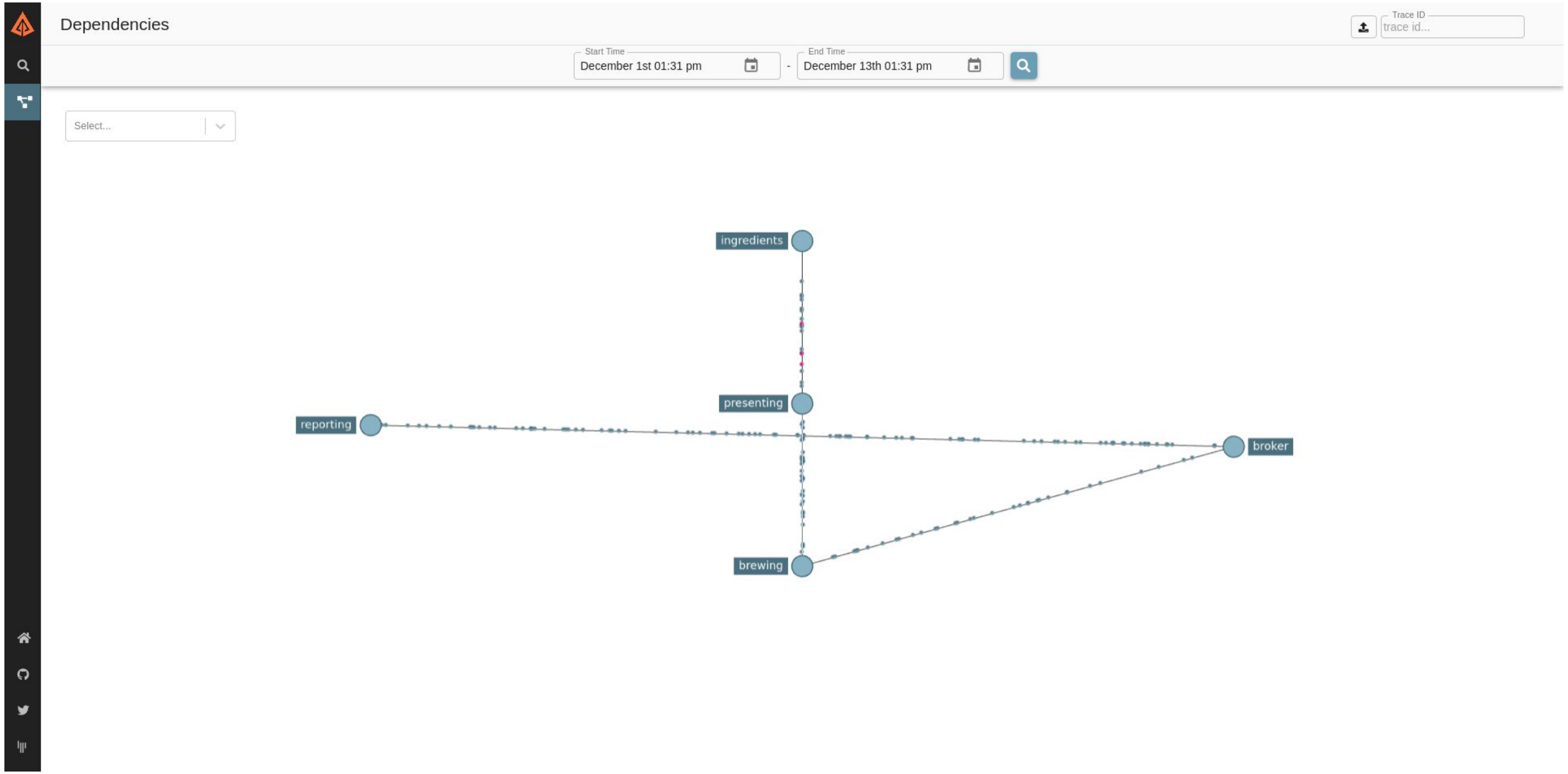
@timestamp, @version, _id, _index, _type, host, message, timestamp

Time	severity	service	trace	sp n	exportable	pid	thread	class	rest
March 4th 2016, 10:52:40.837	INFO	service	f2dbd3c74410aa1	2dbd3c74410aa1	true	6811	nio-8081-exec-7	i.s.c.sleuth.docs.service1.Application	Hello from service1. Calling service2
March 4th 2016, 10:52:41.040	INFO	service	f2dbd3c74410aa1	78edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Hello from service2. Calling service3 and then service4
March 4th 2016, 10:52:41.342	INFO	service	f2dbd3c74410aa1	ea3aa4ac6cc6d6	true	6813	nio-8083-exec-7	i.s.c.sleuth.docs.service3.Application	Hello from service3
March 4th 2016, 10:52:41.347	INFO	service	f2dbd3c74410aa1	78edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Got response from service3 [Hello from service3]
March 4th 2016, 10:52:41.752	INFO	service	f2dbd3c74410aa1	4cae76362b7270	true	6814	nio-8084-exec-7	i.s.c.sleuth.docs.service4.Application	Hello from service4
March 4th 2016, 10:52:41.754	INFO	service	f2dbd3c74410aa1	78edd63b259a279	true	6812	nio-8082-exec-7	i.s.c.sleuth.docs.service2.Application	Got response from service4 [Hello from service4]
March 4th 2016, 10:52:41.856	INFO	service	f2dbd3c74410aa1	2dbd3c74410aa1	true	6811	nio-8081-exec-7	i.s.c.sleuth.docs.service1.Application	Got response from service2 [Hello from service2, response from

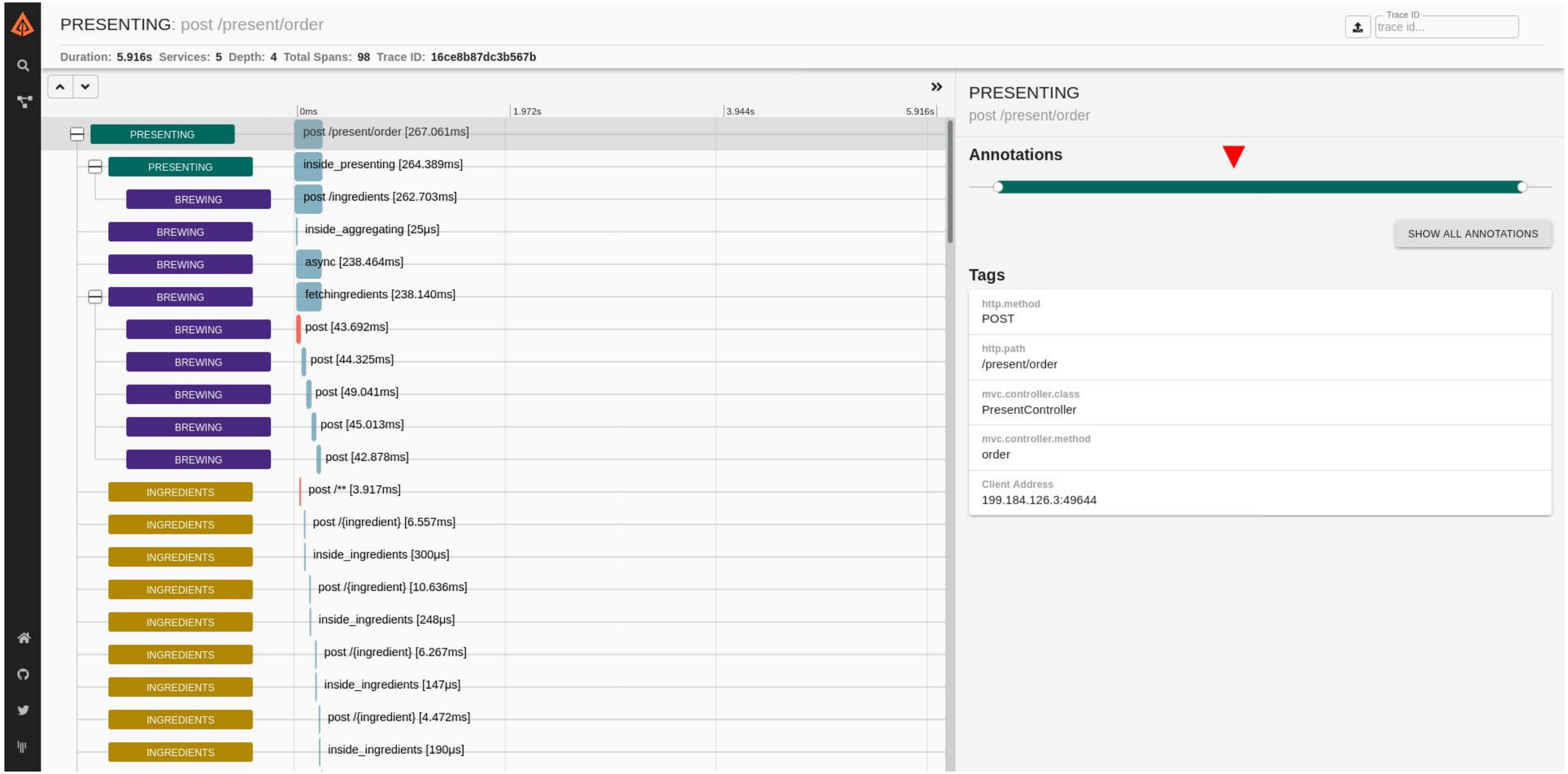
```
Push spans
```


Span store
Zipkin Collector
Fraud Detection
Fraud Detection
Fraud Detection

Zipkin with Lens UI



Zipkin with Lens UI



PRESENTING: post /present/order

Duration: **5.916s** Services: **5** Depth: **4** Total Spans: **98** Trace ID: **16ce8b87dc3b567b**

Service	Span
PRESENTING	post /present/order [267.061ms]
PRESENTING	inside_presenting [264.389ms]
BREWING	post /ingredients [262.703ms]
BREWING	inside_aggregating [25µs]
BREWING	async [238.464ms]
BREWING	fetchingredients [238.140ms]
BREWING	post [43.692ms]
BREWING	post [44.325ms]
BREWING	post [49.041ms]
BREWING	post [45.013ms]
BREWING	post [42.878ms]
INGREDIENTS	post /** [3.917ms]
INGREDIENTS	post /{ingredient} [6.557ms]
INGREDIENTS	inside_ingredients [300µs]
INGREDIENTS	post /{ingredient} [10.636ms]
INGREDIENTS	inside_ingredients [248µs]
INGREDIENTS	post /{ingredient} [6.267ms]
INGREDIENTS	inside_ingredients [147µs]
INGREDIENTS	post /{ingredient} [4.472ms]
INGREDIENTS	inside_ingredients [190µs]

PRESENTING
post /present/order

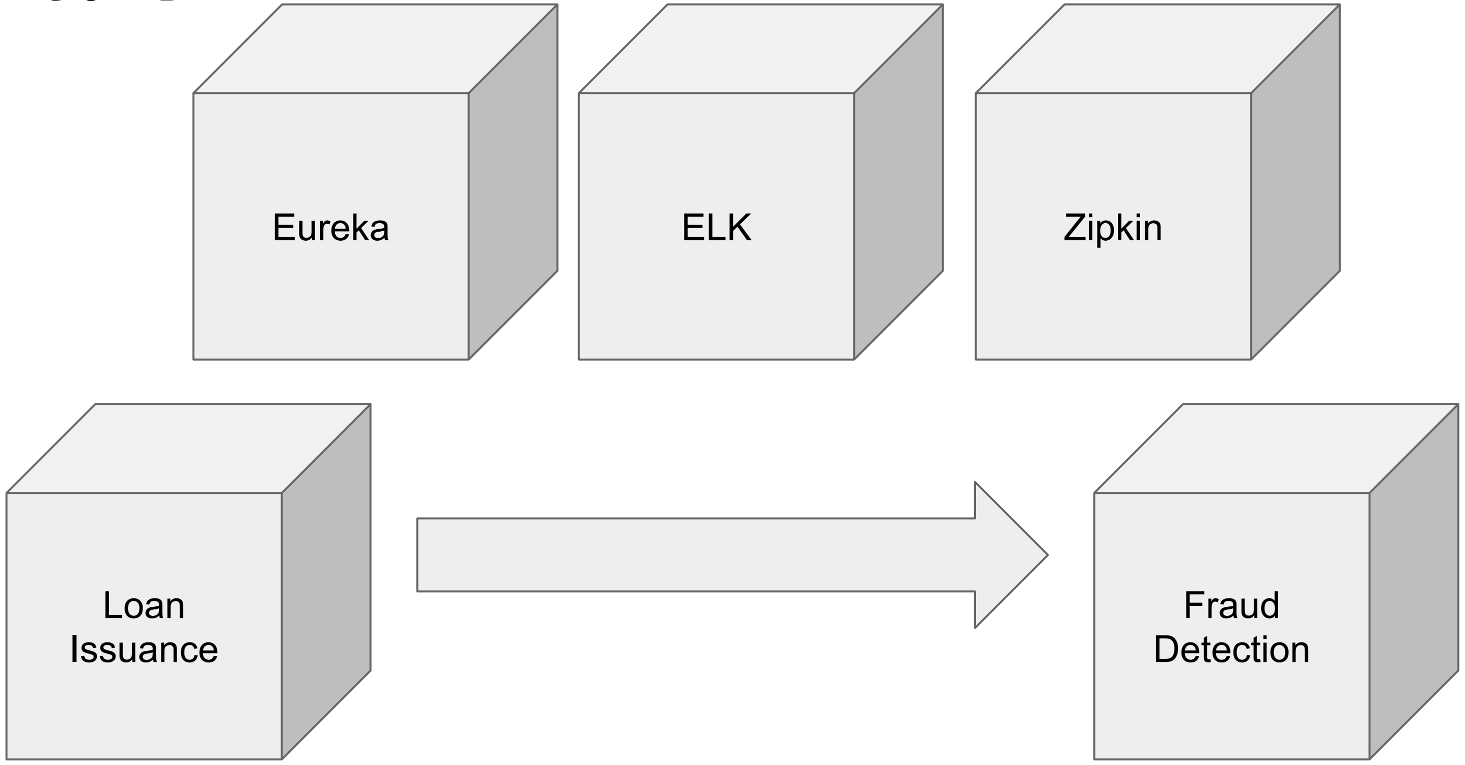
Annotations

SHOW ALL ANNOTATIONS

Tags

http.method	POST
http.path	/present/order
mvc.controller.class	PresentController
mvc.controller.method	order
Client Address	199.184.126.3:49644

Let's Code



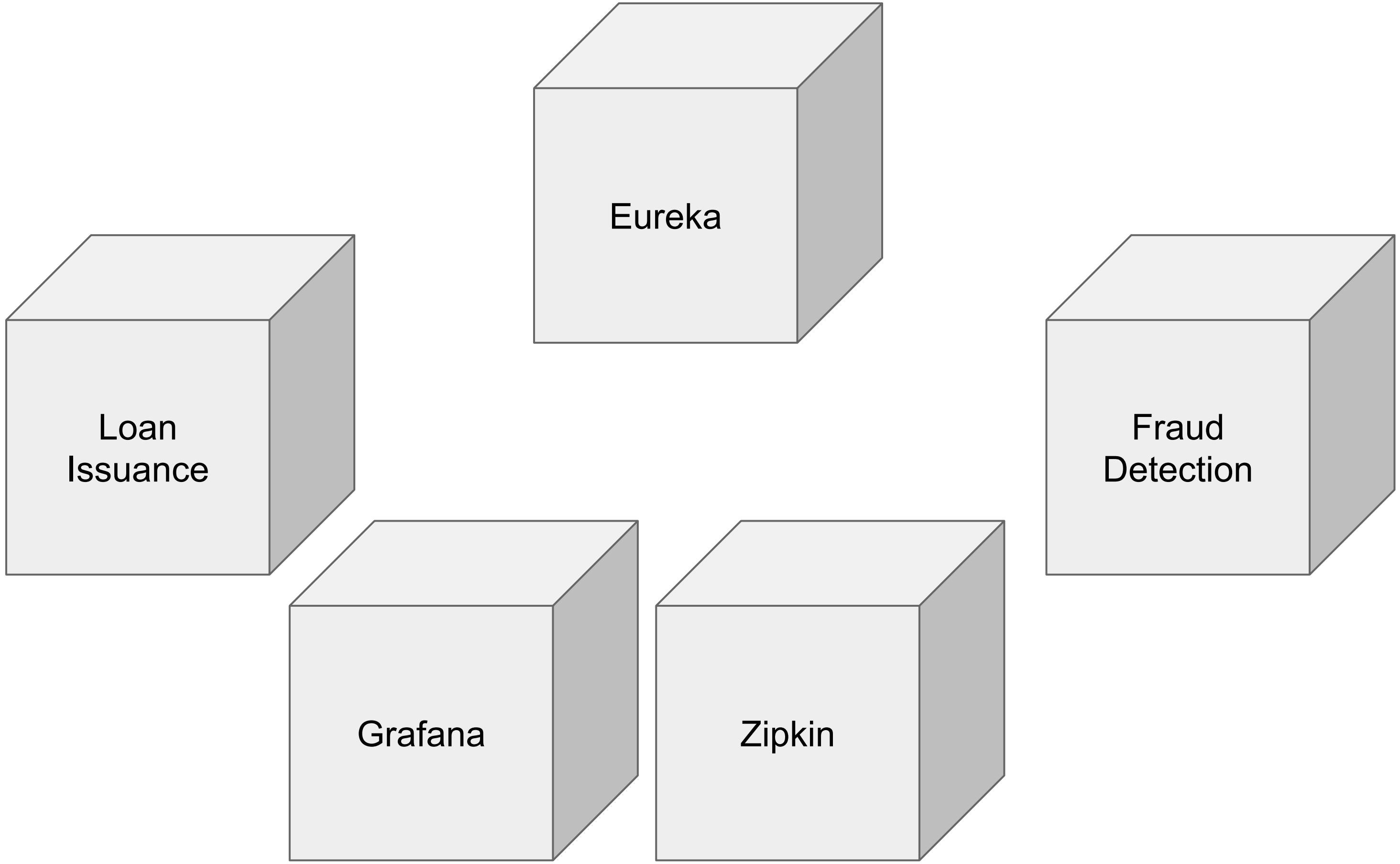
DEMO

Part 3 - Assignment

Assignment:

Circuit breaking, metrics aggregation and latency analysis with Spring Cloud. In this lab, students will use the Micrometer project to create custom metrics in their Spring Cloud based applications, generated via Spring Initialzr. Spring Cloud Circuit Breaker will be used to wrap the calls from one application to another. Via Resilience4j and Micrometer Prometheus integration we will be able to gather metrics in Prometheus and display them in Grafana. Afterwards, thanks to adding Spring Cloud Sleuth to the classpath we will be able to see the latency analysis in the Zipkin project.

Assignment time (15 min)


Eureka
Loan Issuance
Fraud Detection
Grafana
Zipkin

Part 3 - Assignment

https://tinyurl.com/spring-cloud-workshops#assignment-3

https://gist.github.com/marcingrzejszczak/82a0e46f65c9ba3280dd14f395bfbf5d#assignment-3

PART 4

How can we implement an API gateway in a distributed system?

How can we work with messaging when dealing with microservices?

SEGMENT 10

How can we implement an API gateway in a distributed system?

What is an API Gateway?

What is Spring Cloud Gateway?

API Gateway - the benefits

Separation of system clients from

- Services API
- Services location

Might lower the communication chattiness (lower number of calls)

Moves the API complexity from the client to the gateway

Abstracts the internal protocols by using a common API

API Gateway - the drawbacks

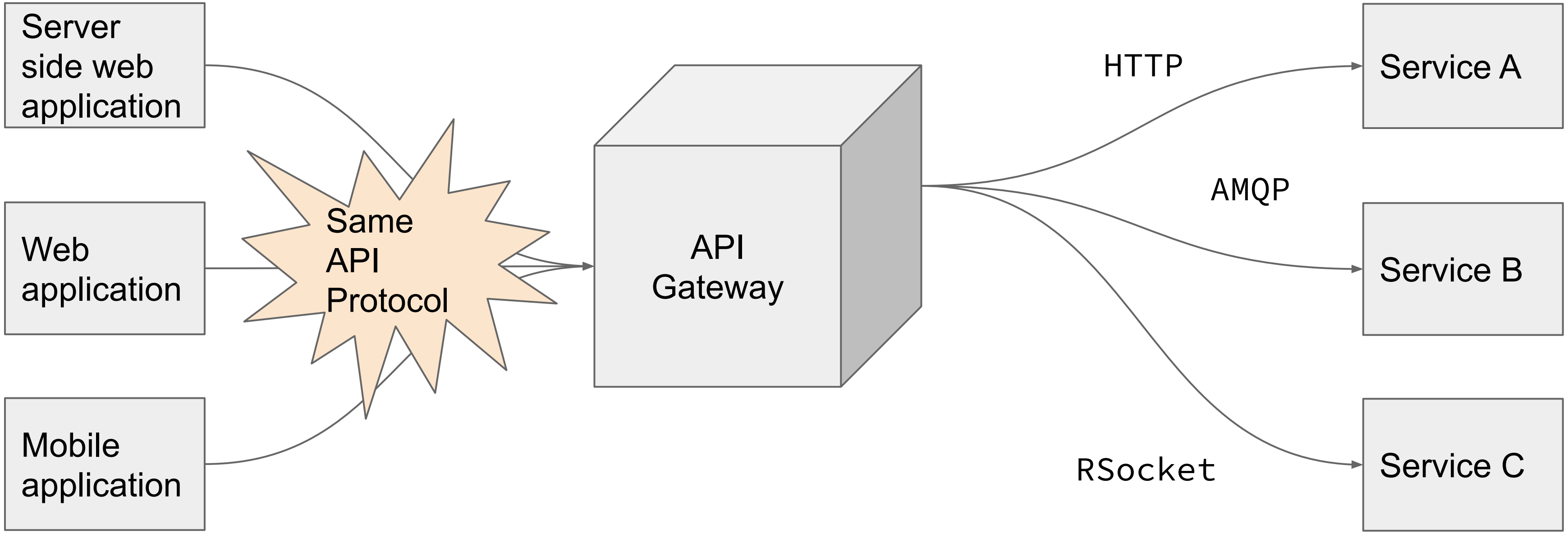
Complexity

    The gateway has to be deployed and managed

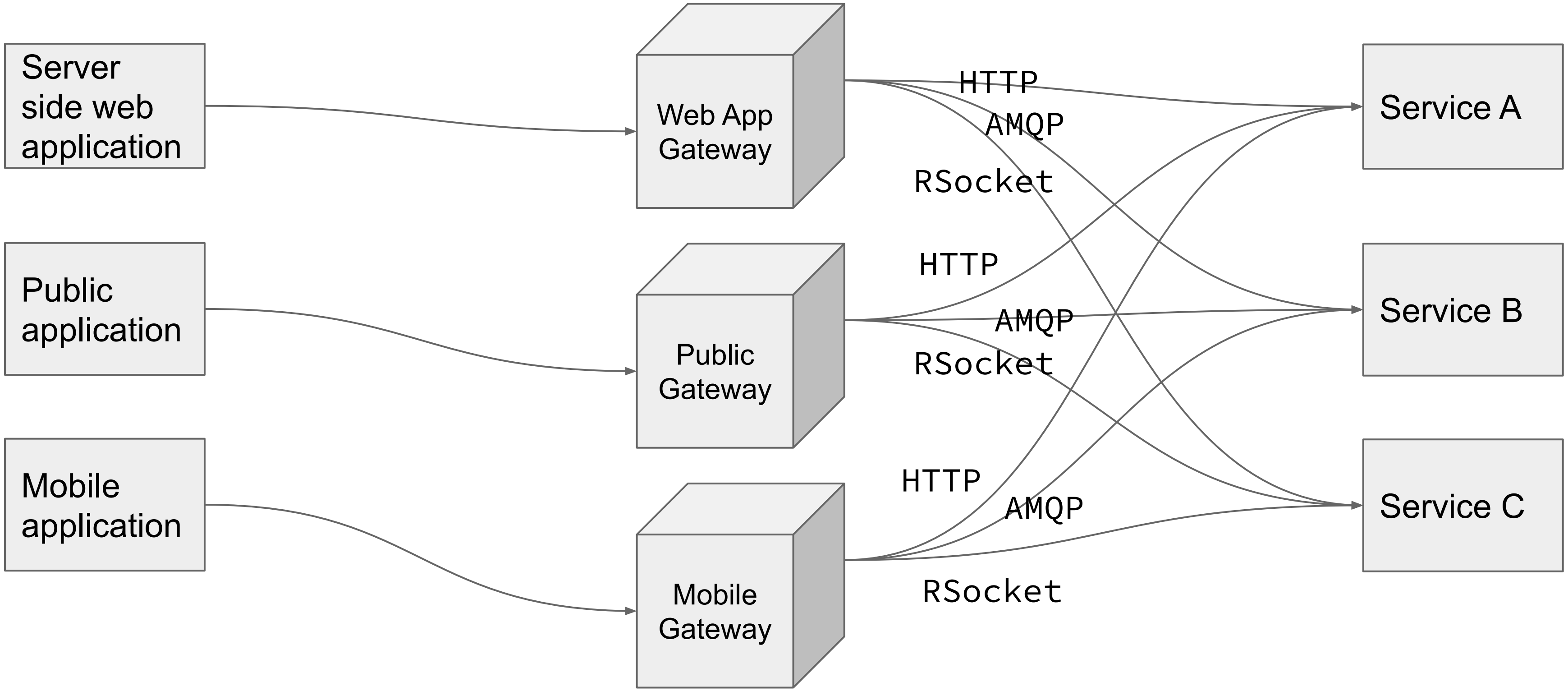
Latency

    Additional hop to perform a business request

What is an API Gateway?



What is Backend for Frontend?



What is Spring Cloud Gateway

Built on top of Spring 5, Spring Boot 2 and Project Reactor

Route to APIs

Provide cross cutting concerns

- Security
- Monitoring/metrics
- Resiliency

What is Spring Cloud Gateway

Route

- ID
- Destination URI
- Predicates
- Filters
- Route matched if the aggregate predicate is true

What is Spring Cloud Gateway

Predicate

- Java 8 Function `Predicate`
- Input type is a Spring Framework `ServerWebExchange`
- Match on anything from the HTTP request

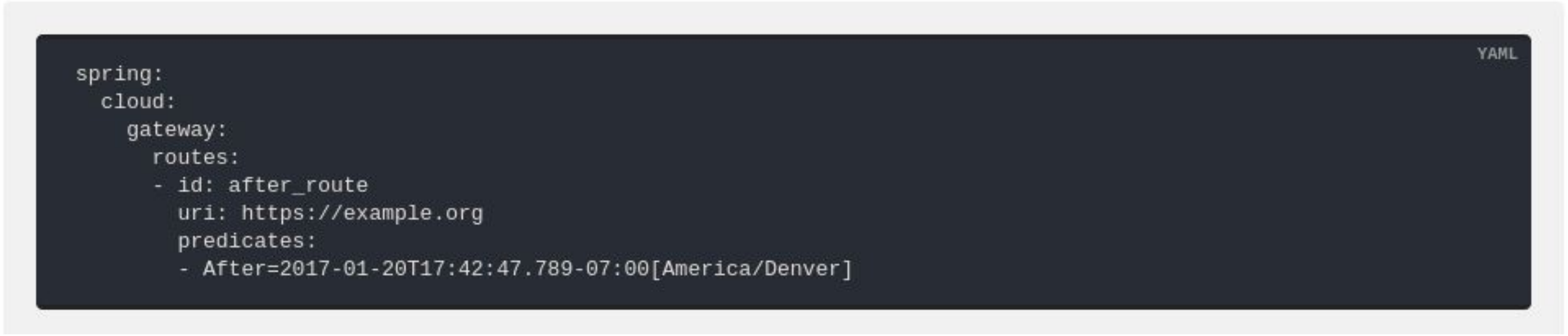
What is Spring Cloud Gateway

Filter

- Spring Framework `GatewayFilter`
- Constructed with a specific factory
- You can modify requests and responses
  - Before sending the downstream request
  - After sending the downstream request

What is Spring Cloud Gateway

The After Route Predicate Factory

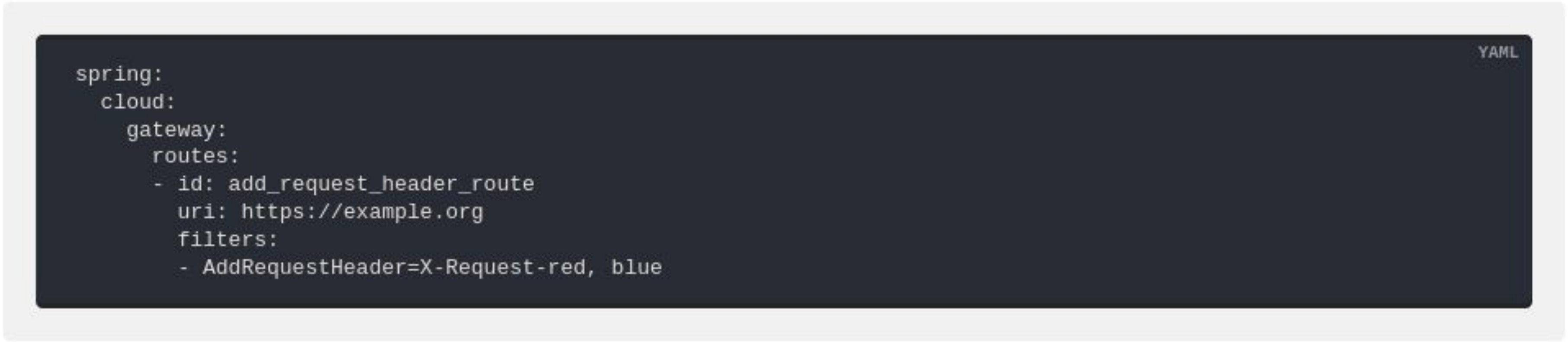

YAML

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: after_route
        uri: https://example.org
        predicates:
        - After=2017-01-20T17:42:47.789-07:00[America/Denver]
```

[1] - https://cloud.spring.io/spring-cloud-gateway/reference/html/

What is Spring Cloud Gateway

The AddRequestHeader GatewayFilter Factory

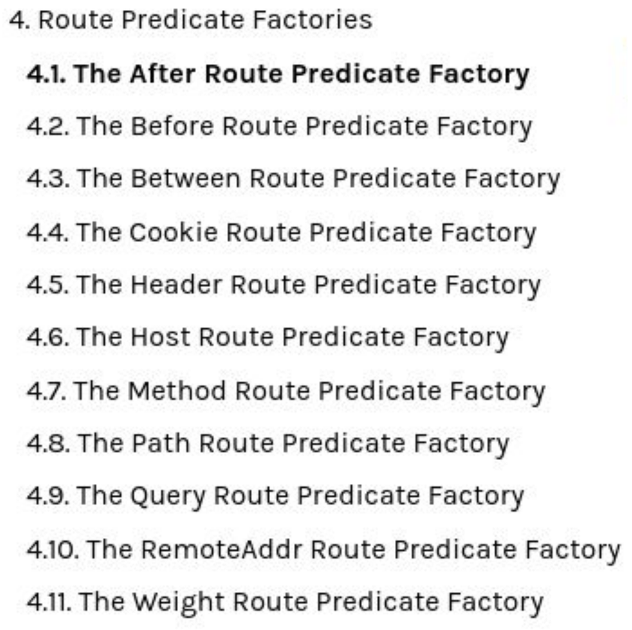


YAML

```yaml
spring:
  cloud:
    gateway:
      routes:
      - id: add_request_header_route
        uri: https://example.org
        filters:
        - AddRequestHeader=X-Request-red, blue
```

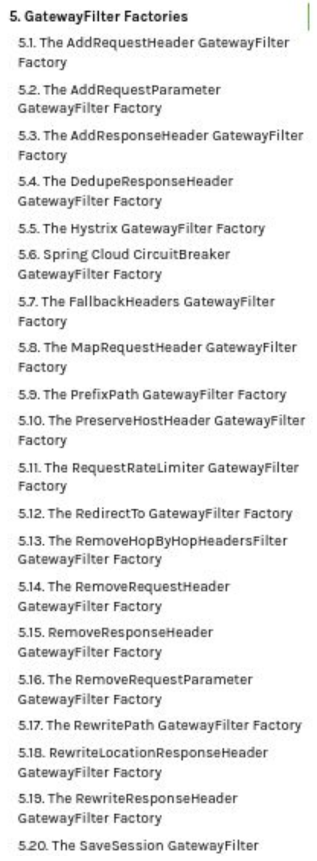
[1] - https://cloud.spring.io/spring-cloud-gateway/reference/html/

What is Spring Cloud Gateway



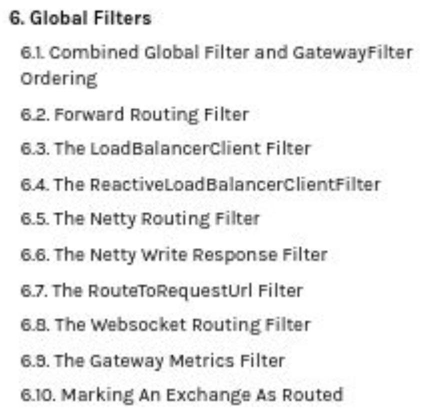
4. Route Predicate Factories

- **4.1. The After Route Predicate Factory**
- 4.2. The Before Route Predicate Factory
- 4.3. The Between Route Predicate Factory
- 4.4. The Cookie Route Predicate Factory
- 4.5. The Header Route Predicate Factory
- 4.6. The Host Route Predicate Factory
- 4.7. The Method Route Predicate Factory
- 4.8. The Path Route Predicate Factory
- 4.9. The Query Route Predicate Factory
- 4.10. The RemoteAddr Route Predicate Factory
- 4.11. The Weight Route Predicate Factory



5. GatewayFilter Factories

- 5.1. The AddRequestHeader GatewayFilter Factory
- 5.2. The AddRequestParameter GatewayFilter Factory
- 5.3. The AddResponseHeader GatewayFilter Factory
- 5.4. The DedupeResponseHeader GatewayFilter Factory
- 5.5. The Hystrix GatewayFilter Factory
- 5.6. Spring Cloud CircuitBreaker GatewayFilter Factory
- 5.7. The FallbackHeaders GatewayFilter Factory
- 5.8. The MapRequestHeader GatewayFilter Factory
- 5.9. The PrefixPath GatewayFilter Factory
- 5.10. The PreserveHostHeader GatewayFilter Factory
- 5.11. The RequestRateLimiter GatewayFilter Factory
- 5.12. The RedirectTo GatewayFilter Factory
- 5.13. The RemoveHopByHopHeadersFilter GatewayFilter Factory
- 5.14. The RemoveRequestHeader GatewayFilter Factory
- 5.15. RemoveResponseHeader GatewayFilter Factory
- 5.16. The RemoveRequestParameter GatewayFilter Factory
- 5.17. The RewritePath GatewayFilter Factory
- 5.18. RewriteLocationResponseHeader GatewayFilter Factory
- 5.19. The RewriteResponseHeader GatewayFilter Factory
- 5.20. The SaveSession GatewayFilter

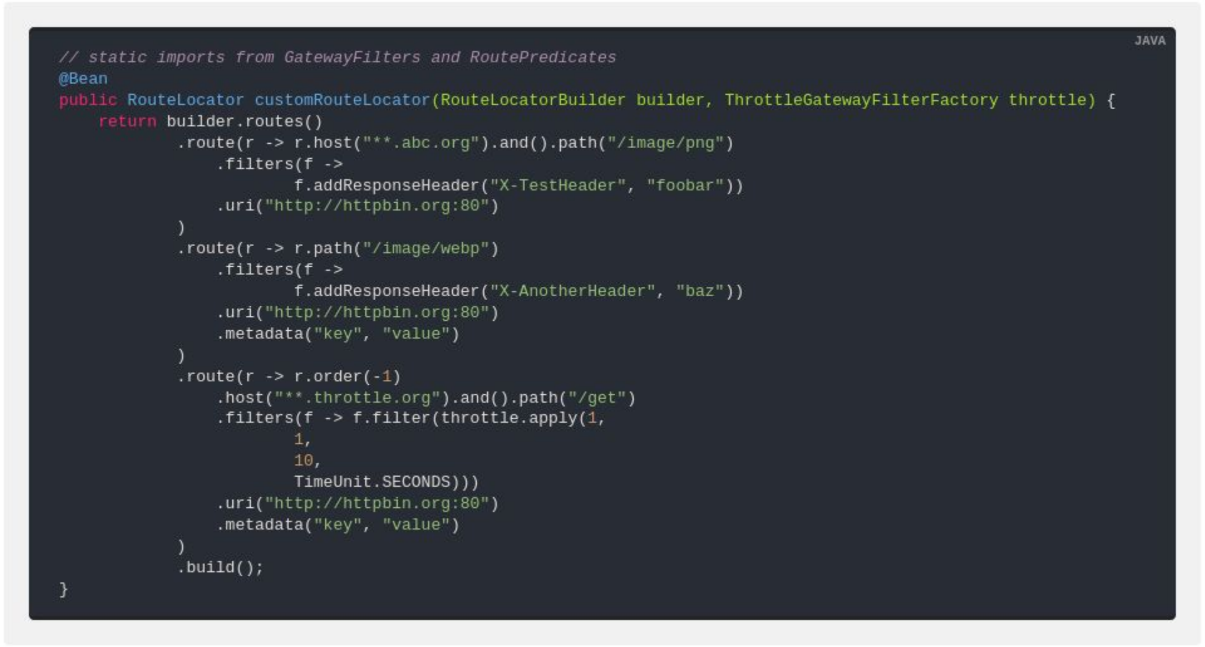


6. Global Filters

- 6.1. Combined Global Filter and GatewayFilter Ordering
- 6.2. Forward Routing Filter
- 6.3. The LoadBalancerClient Filter
- 6.4. The ReactiveLoadBalancerClientFilter
- 6.5. The Netty Routing Filter
- 6.6. The Netty Write Response Filter
- 6.7. The RouteToRequestUrl Filter
- 6.8. The Websocket Routing Filter
- 6.9. The Gateway Metrics Filter
- 6.10. Marking An Exchange As Routed

[1] - https://cloud.spring.io/spring-cloud-gateway/reference/html/

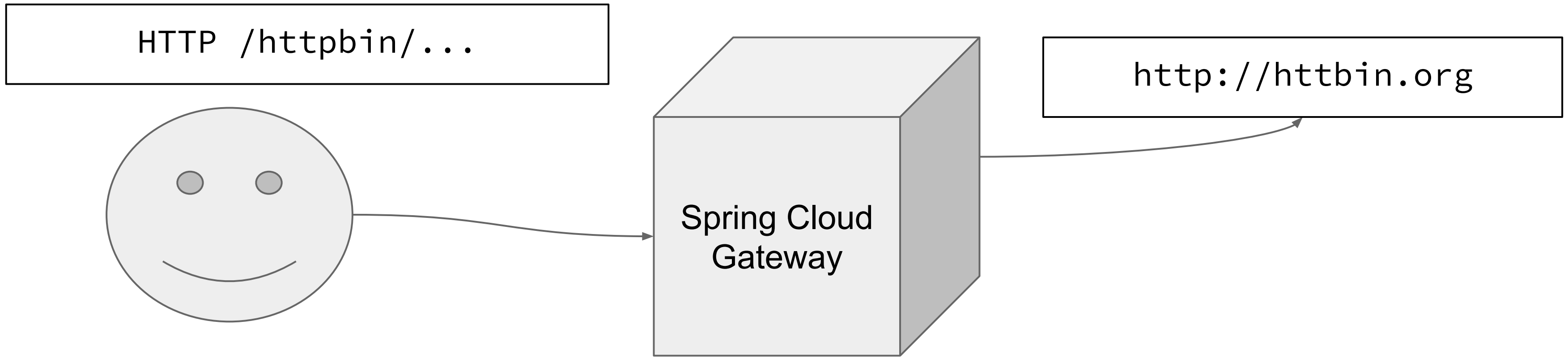
What is Spring Cloud Gateway


JAVA

```java
// static imports from GatewayFilters and RoutePredicates
@Bean
public RouteLocator customRouteLocator(RouteLocatorBuilder builder, ThrottleGatewayFilterFactory throttle) {
    return builder.routes()
            .route(r -> r.host("**.abc.org").and().path("/image/png")
                .filters(f ->
                        f.addResponseHeader("X-TestHeader", "foobar"))
                .uri("http://httpbin.org:80")
            )
            .route(r -> r.path("/image/webp")
                .filters(f ->
                        f.addResponseHeader("X-AnotherHeader", "baz"))
                .uri("http://httpbin.org:80")
                .metadata("key", "value")
            )
            .route(r -> r.order(-1)
                .host("**.throttle.org").and().path("/get")
                .filters(f -> f.filter(throttle.apply(1,
                        1,
                        10,
                        TimeUnit.SECONDS)))
                .uri("http://httpbin.org:80")
                .metadata("key", "value")
            )
            .build();
}
```

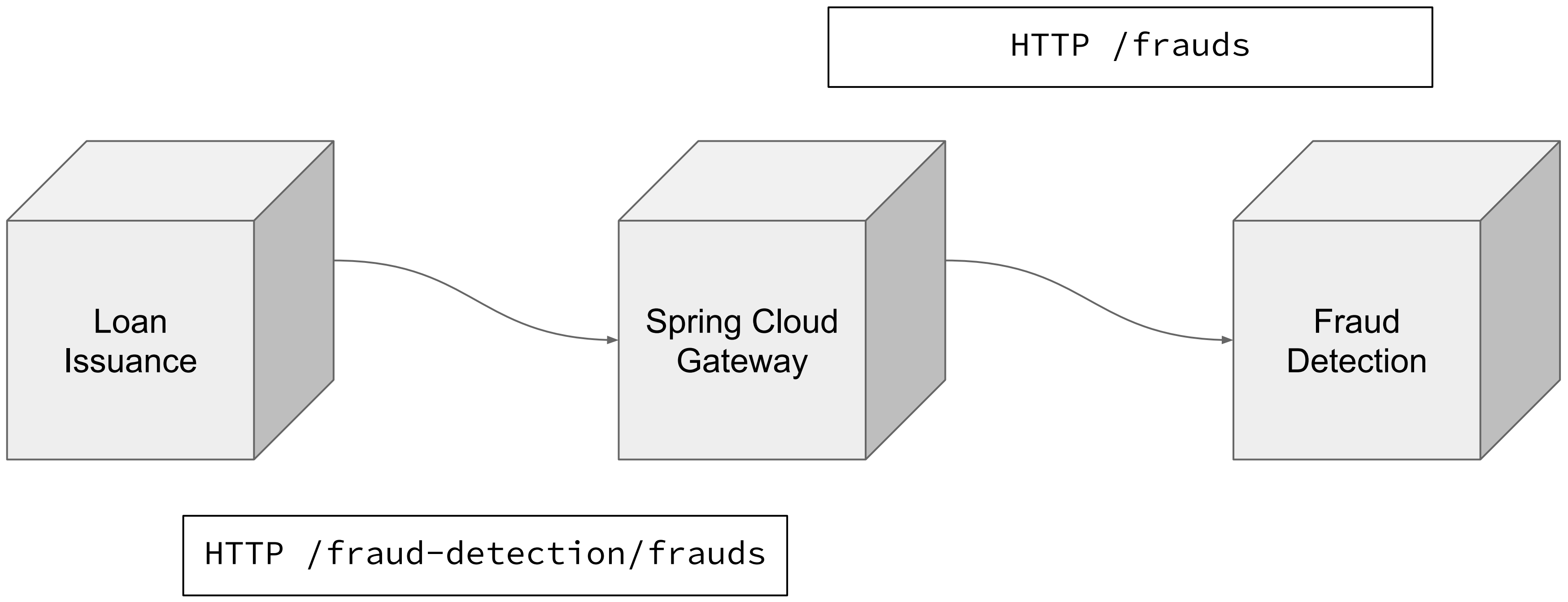
[1] - https://cloud.spring.io/spring-cloud-gateway/reference/html/

Let's Code



DEMO

Let's Code



DEMO

SEGMENT 11

How can microservices talk to each other over messaging?

Why should you consider using messaging in microservice environment?

What is Spring Cloud Stream?

Why should you consider using messaging in microservice environment?

Microservice = individual, separately deployed application

Immediate reply required = request-response

Events and asynchronous messaging

- Follow real life cases
- Scalability
- Resiliency
- Loose physical coupling

Why should you consider using messaging in microservice environment?

Microservices = smart endpoints, dumb pipes

No central integration messaging bus

Each microservice picks their messaging broker

Lower coordination required

New version of the service = new topic / queue

Asynchronous Messaging Patterns

Event Firehose

- Events from different sources (many-to-many)
- Highly scalable
- Consumers decide how they want to process the messages
- Broker example - Kafka
- Flow orchestrator example - Spring Cloud Data Flow

Asynchronous Messaging Patterns

Asynchronous Command Calls

- Exchanging messages with a guaranteed delivery
- Often point-to-point (queues not topics) yet asynchronous
- No stream of events
- Scalable
- Example of a Broker - RabbitMQ

Asynchronous Messaging Patterns

- Data Events Exchange
  - React to a data store change
  - When system reacts to data updates
  - Tool examples (listeners sending out messages)
    - Pivotal GemFire
    - Apache Geode
    - Debezium

What is Spring Cloud Stream?

Framework for message-driven microservice applications

Build on top of Spring Boot and Spring Integration

Provides

- Abstraction over message brokers
- Opinionated configuration for middleware

Setup depending on classpath

Functional requirement based on `java.util.function` package

What is Spring Cloud Stream?

Treat data-centric applications as microservices

  Independently built, tested and deployed

Map business to events on top of message brokers

Push the configuration and content type resolution to the framework

What is Spring Cloud Stream?

JAVA

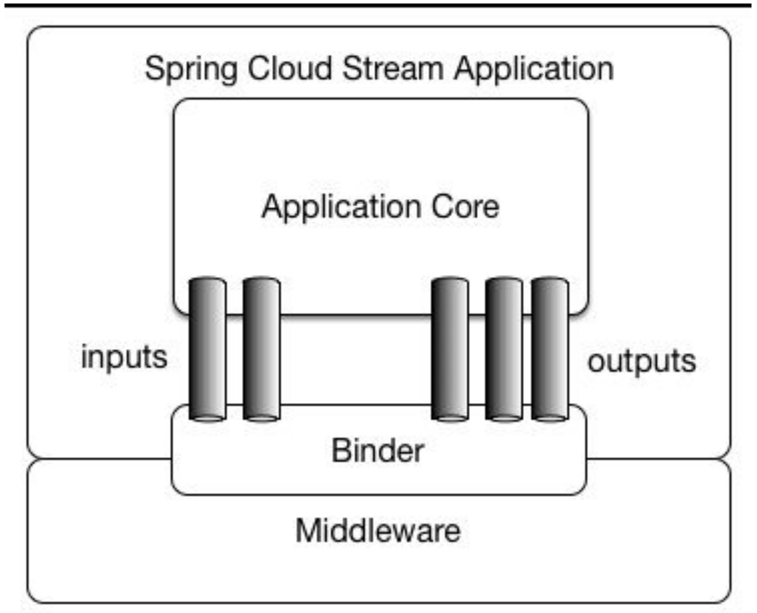
```java
@SpringBootApplication
public class SampleApplication {

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

    @Bean
        public Function<String, String> uppercase() {
            return value -> value.toUpperCase();
        }
}
```

and corresponding test

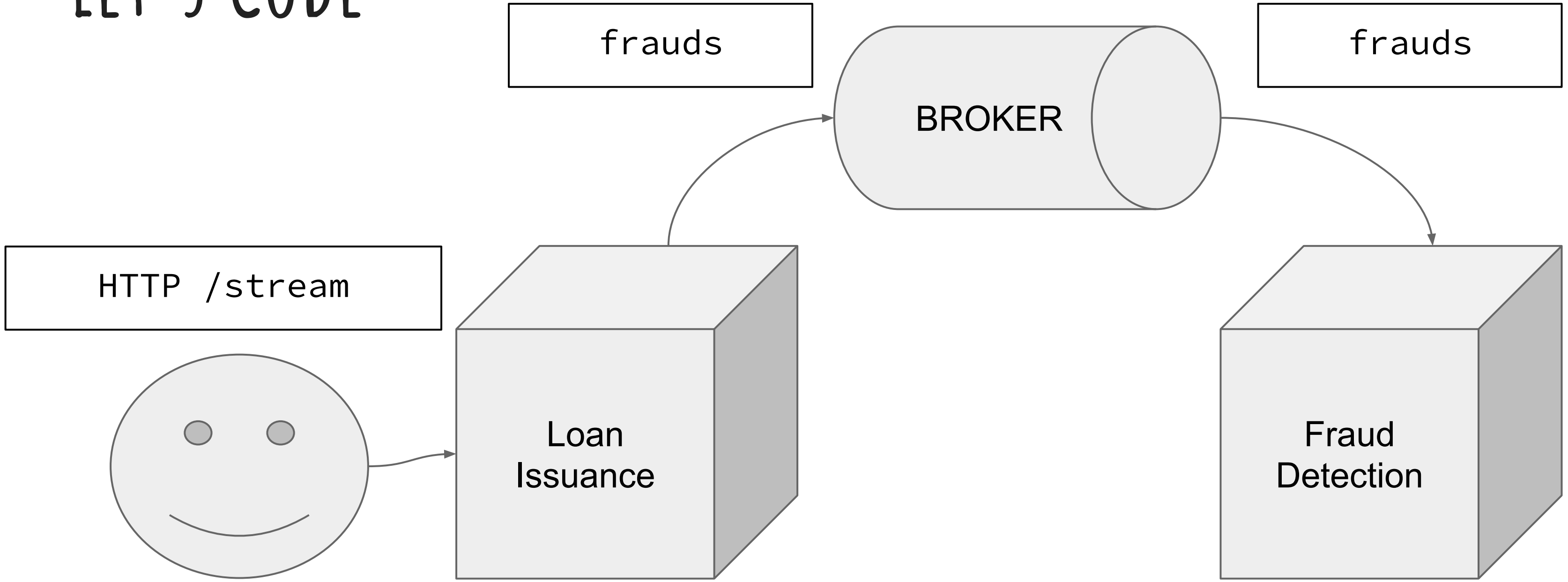
JAVA

```java
@SpringBootTest(classes =  SampleApplication.class)
@Import({TestChannelBinderConfiguration.class})
class BootTestStreamApplicationTests {

        @Autowired
        private InputDestination input;

        @Autowired
        private OutputDestination output;

        @Test
        void contextLoads() {
                input.send(new GenericMessage<byte[]>("hello".getBytes()));
                assertThat(output.receive().getPayload()).isEqualTo("HELLO".getBytes());
        }
}
```

What is Spring Cloud Stream?



Let's Code



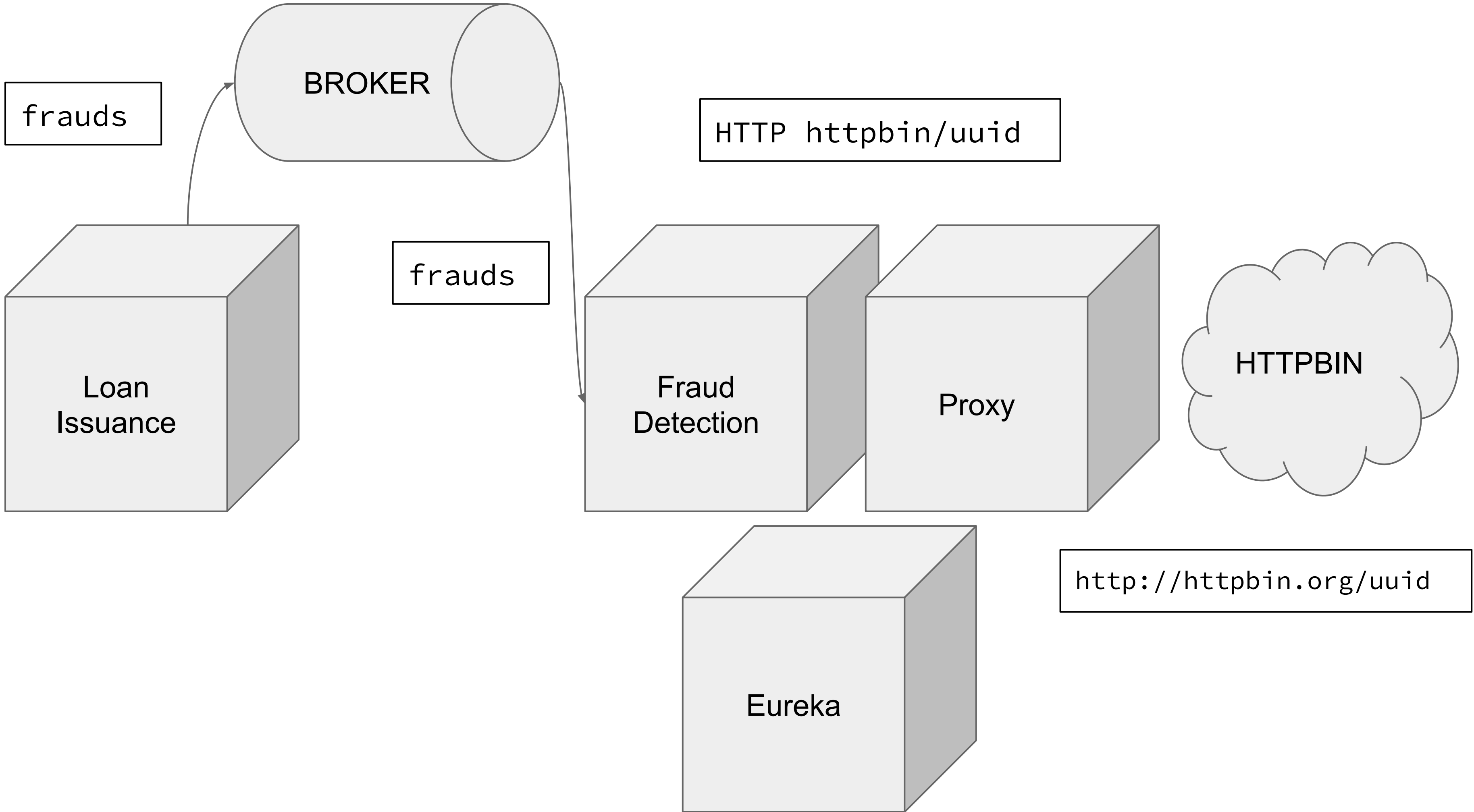
DEMO

Part 4 - Assignment

Assignment:

API gateway and messaging. In this lab, students will generate two Spring Cloud Stream applications that will talk to each other over Spring Cloud Stream with RabbitMQ. The first one will also have an HTTP API to trigger it via the command line. The other upon receiving the message from RabbitMQ will call the Spring Cloud Gateway application to route the traffic to an external website.

Assignment time (20 min)


BROKER
frauds
HTTP httpbin/uuid
frauds
Loan Issuance
Fraud Detection
Proxy
HTTPBIN
Eureka
http://httpbin.org/uuid

Part 4 - Assignment

https://tinyurl.com/spring-cloud-workshops#assignment-4

https://gist.github.com/marcingrzejszczak/82a0e46f65c9ba3280dd14f395bfbf5d#assignment-4