

# Microservices workshop





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...  
Timeline About Friends 3,138 Photos More

When did you work at Opendream?  
... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro  
Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

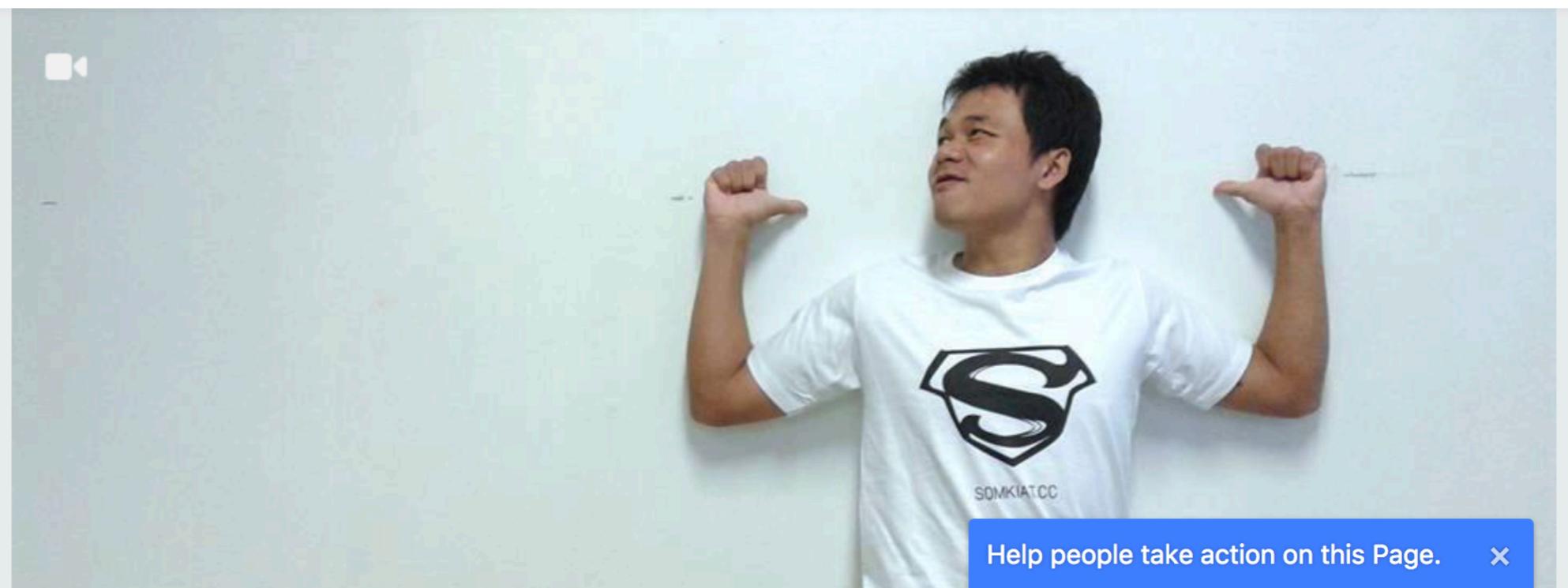
@somkiat.cc

Home

Posts

Videos

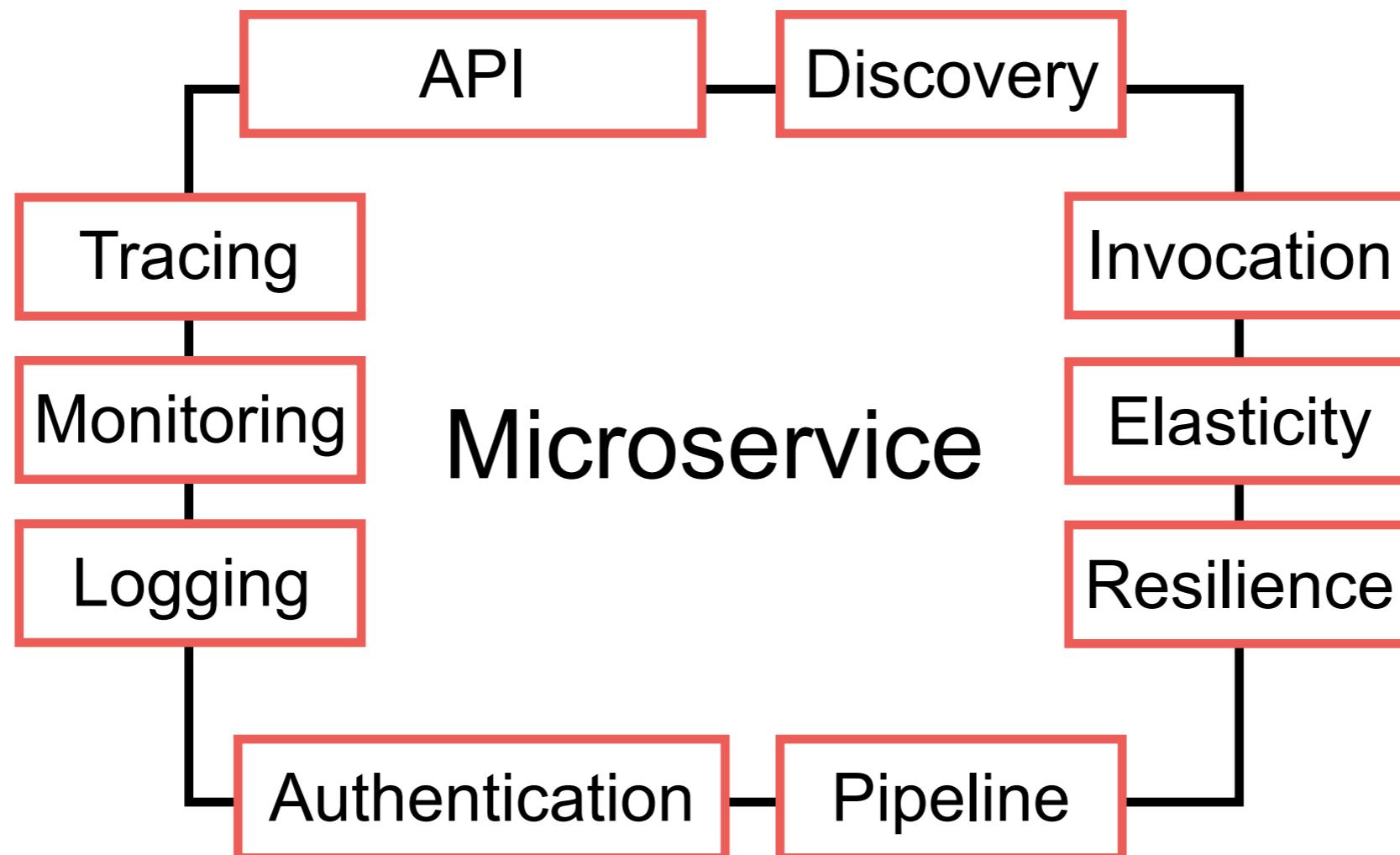
Photos



# Develop Microservice



# Properties of Microservice



# **Workshop**

# **Microservice with NodeJS**



# **API/service document**

## **00-api-document**



# Swagger



Why Swagger > Tools > Resources >

Sign In

Try Free

## Swagger

The Best APIs are Built with Swagger Tools

Try SwaggerHub

Explore Swagger Tools



### Design

Design and model APIs according to specification-based standards



### Build

Build stable, reusable code for your API in almost any language



### Document

Improve developer experience with interactive API documentation



### Test

Perform simple functional tests on your APIs without overhead



### Standardize

Set and enforce API style guidelines across your API architecture

<https://swagger.io/>



Microservices

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

# API document with Swagger

swagger

/api-docs

Explore

## Swagger sample app 0.0.1

[ Base url: localhost:3001/ ]

[/api-docs](#)

Microservices training Swagger Example

Schemes

HTTP

**sample** Sample API

GET

/sample

POST

/sample

GET

/sample/{id}

PUT

/sample/{id}

DELETE

/sample/{id}



Microservices

© 2017 - 2018 Siam Chamnankit Company Limited. All rights reserved.

10

# Create Simple Services

## 01-starter



# Create Simple services

port=3002

Service1

port=3003

Service2



# Communication between services

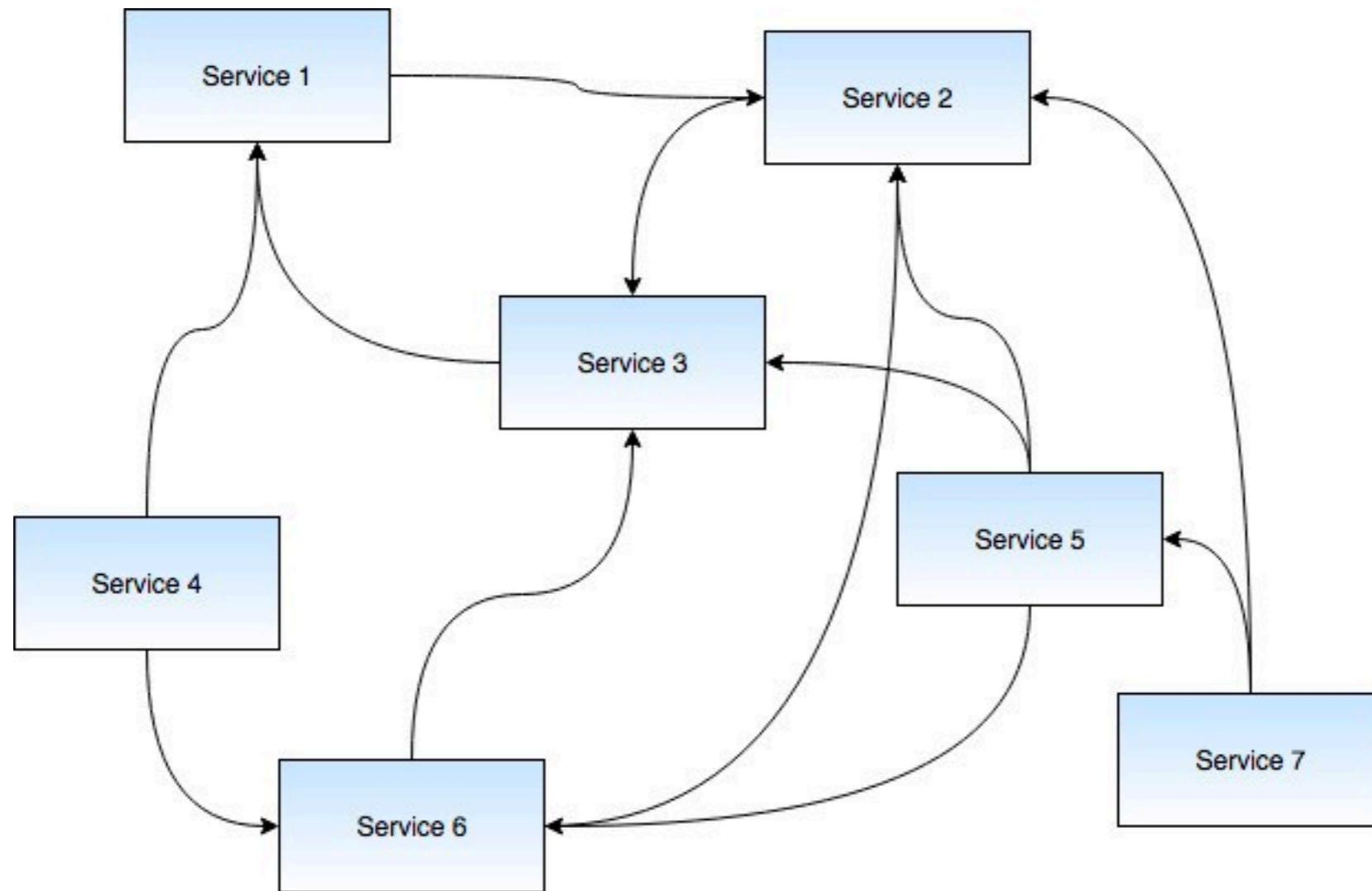
## 02-connect



# Communication between services



# Many services !!



# Using service discovery

## service-discovery-with-consul



# Communication between services



# Service Registry

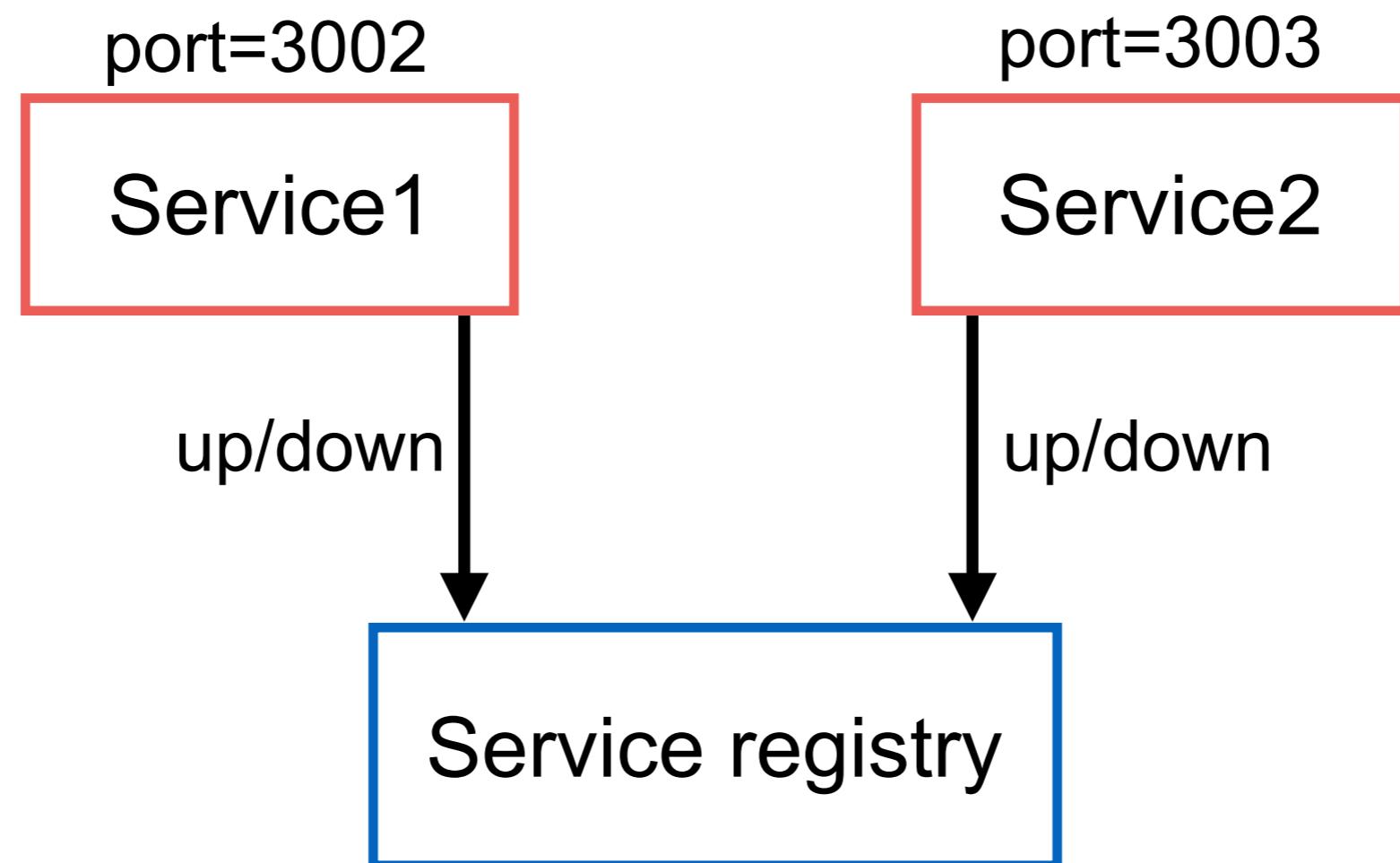
Database to keep data of services

How to register services ?

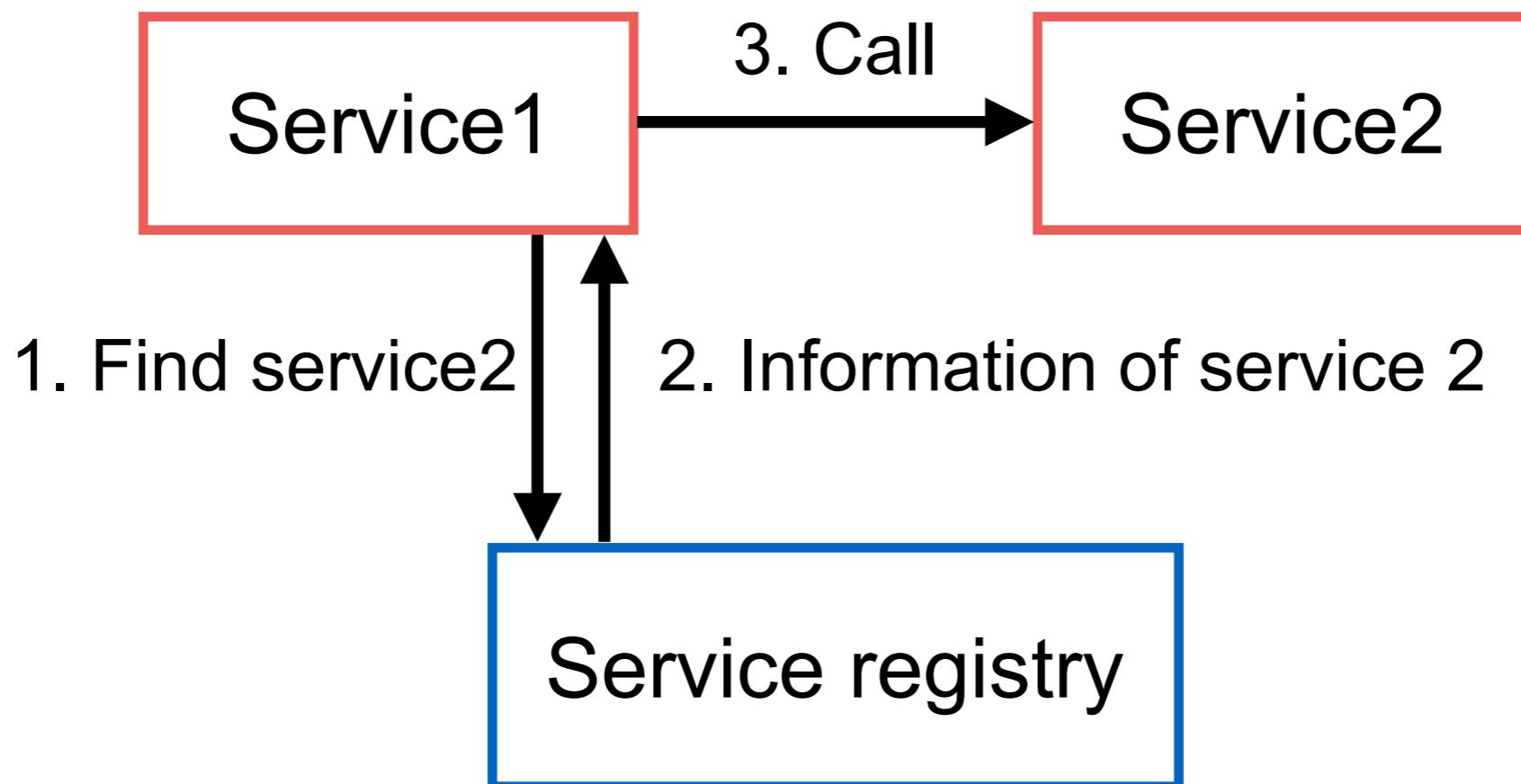
How to discover services ?



# 1. Service register



## 2. Service Discovery

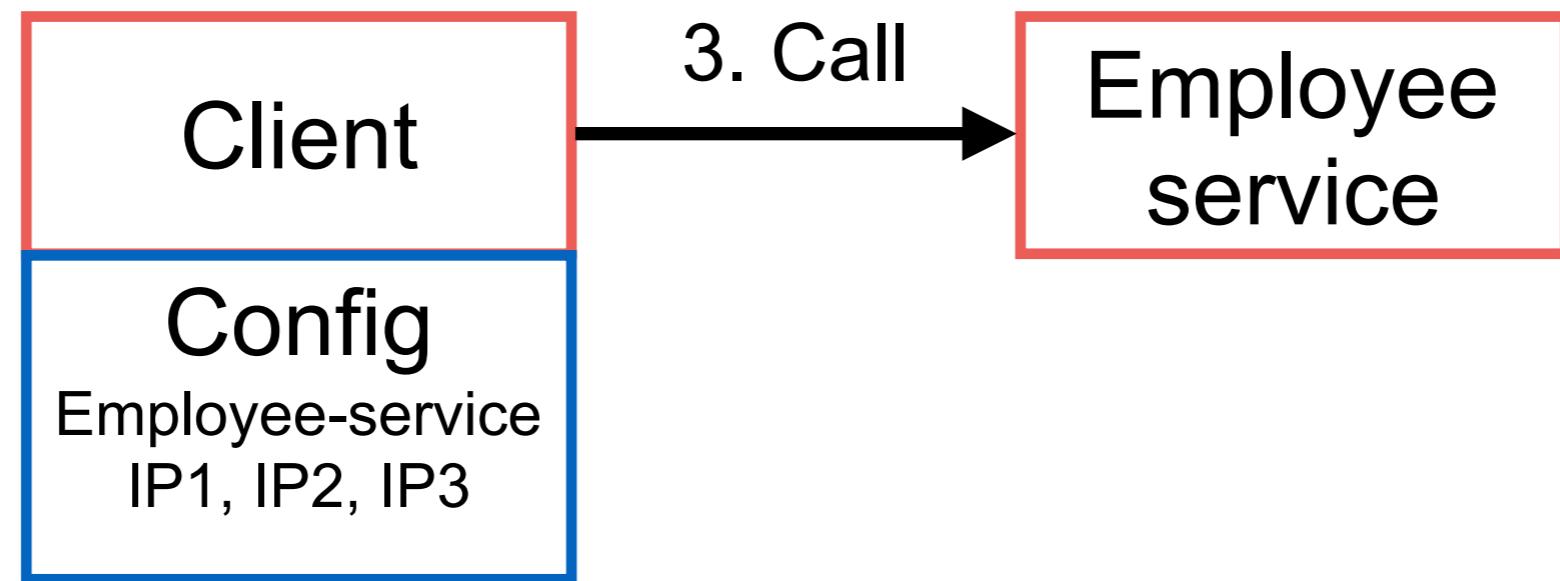


# Service Registry

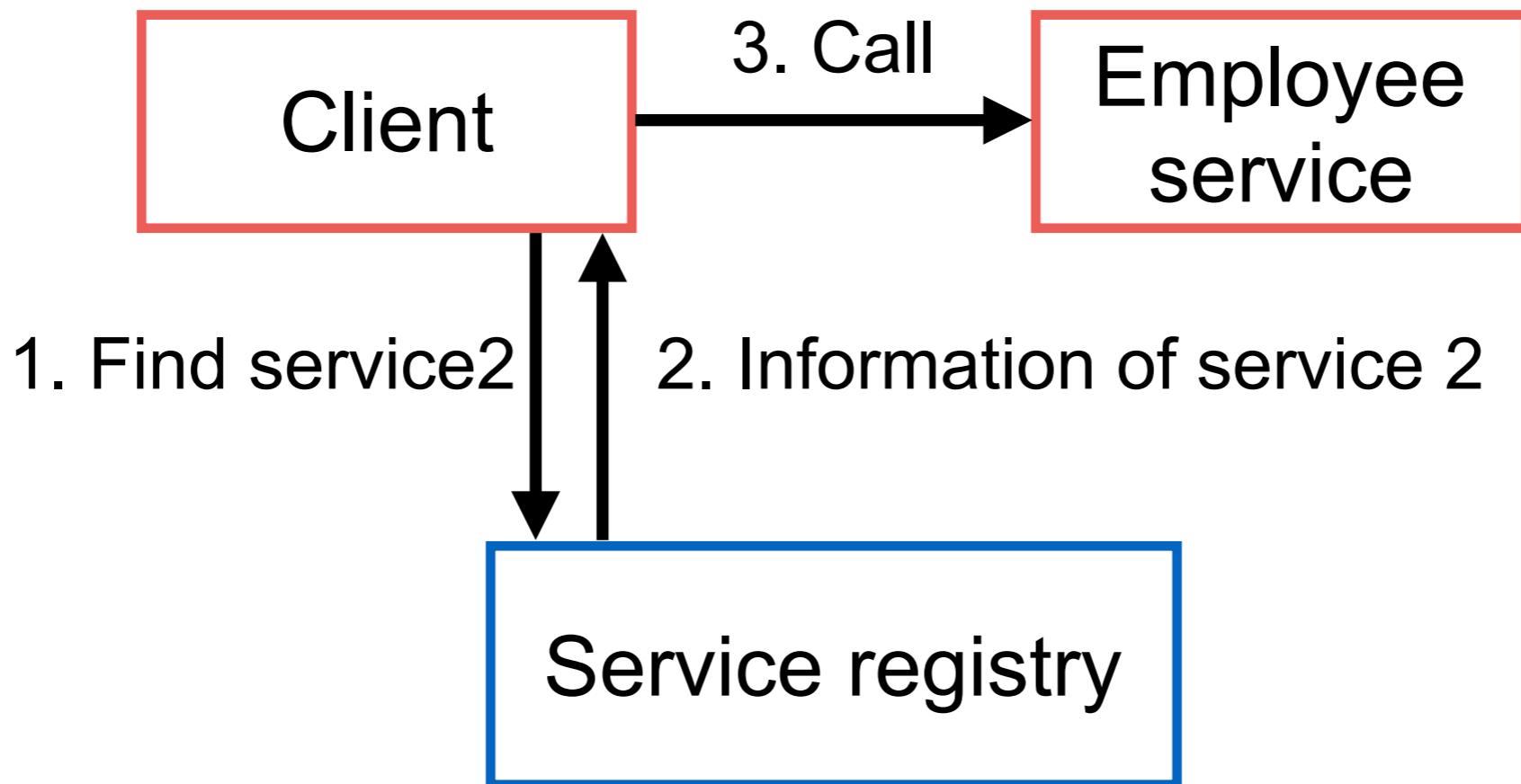
Client-side  
Server-side



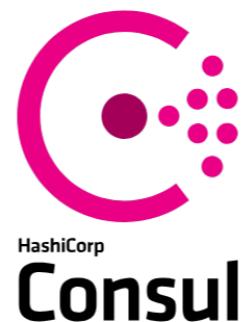
# Client-side



# Server-side



Spring Cloud  
Eureka

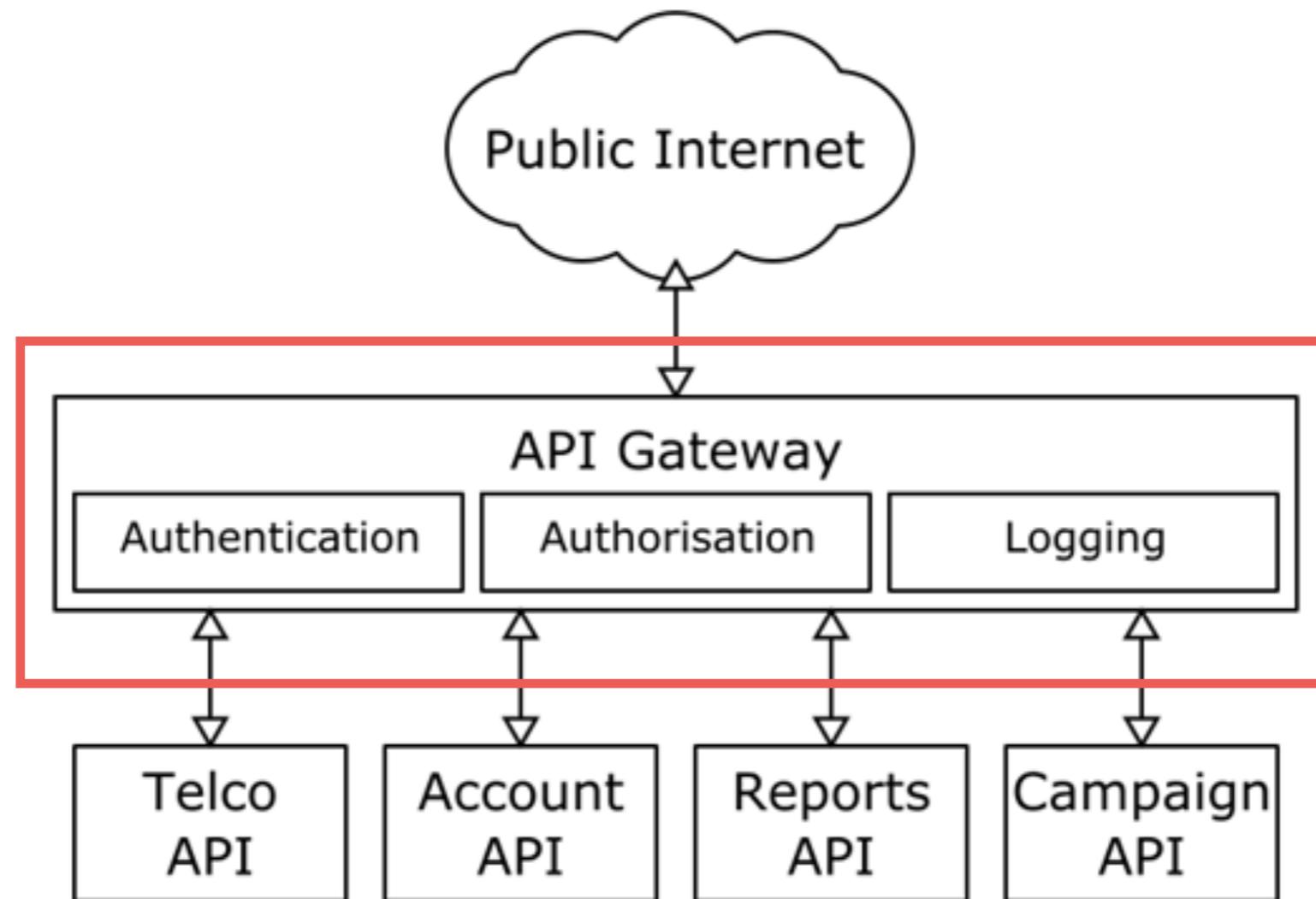


# API gateway

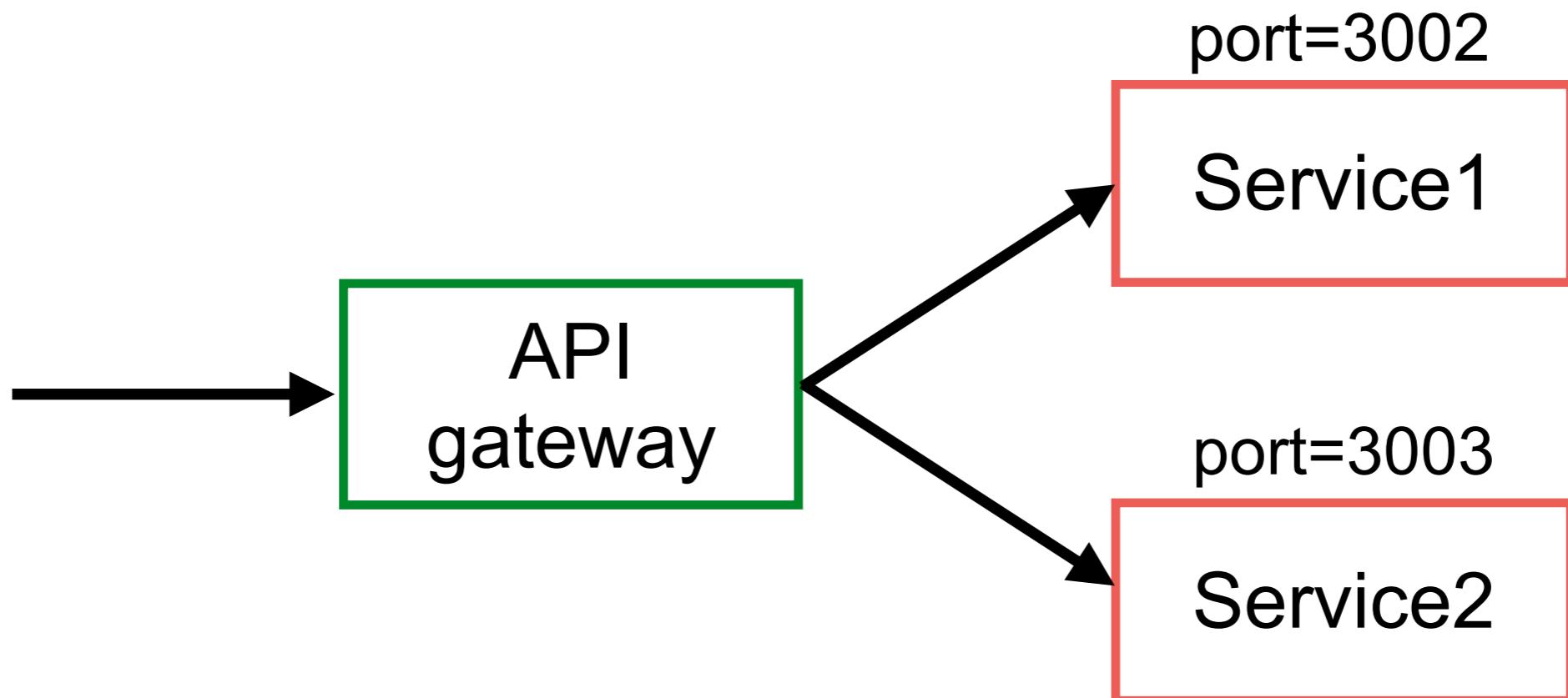
## 03-api-gateway



# API gateway



# API gateway

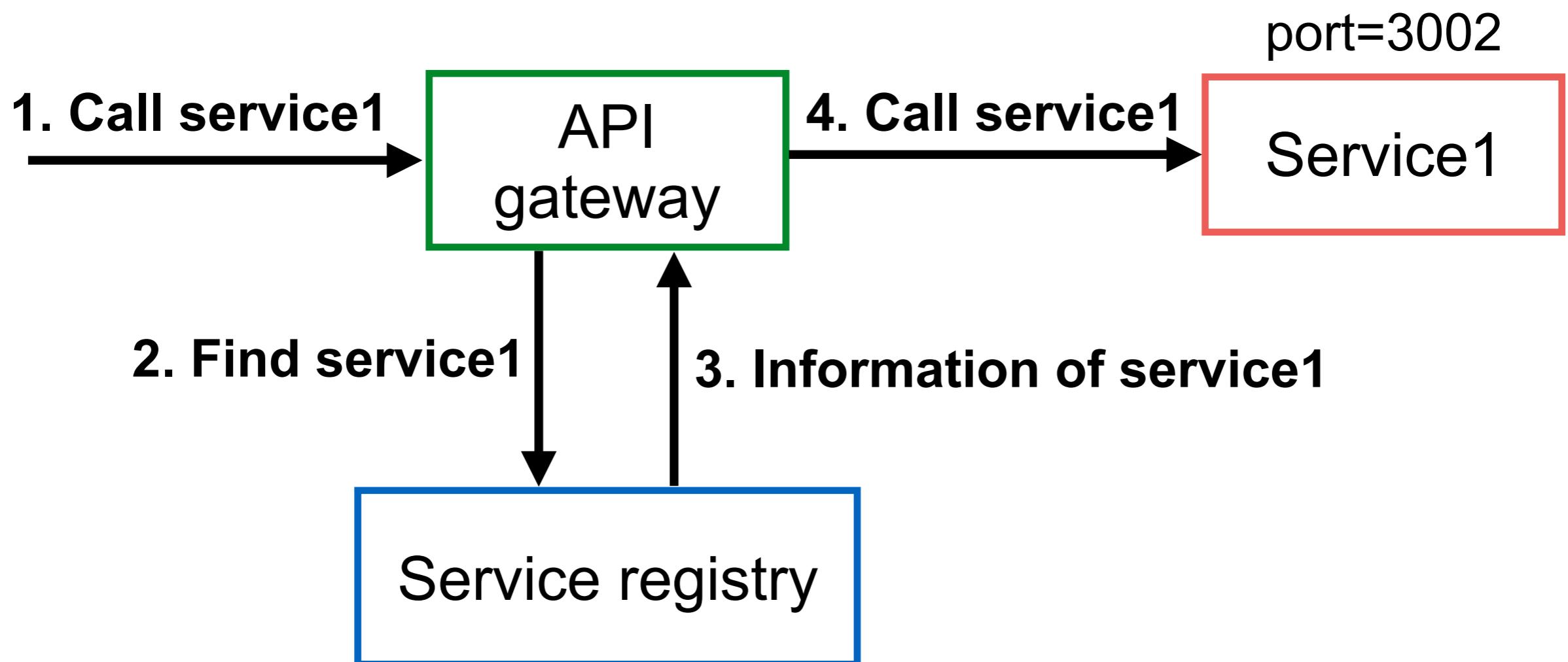


# API gateway + Service registry

## 05-server-side-service-discovery



# Server-side discovery

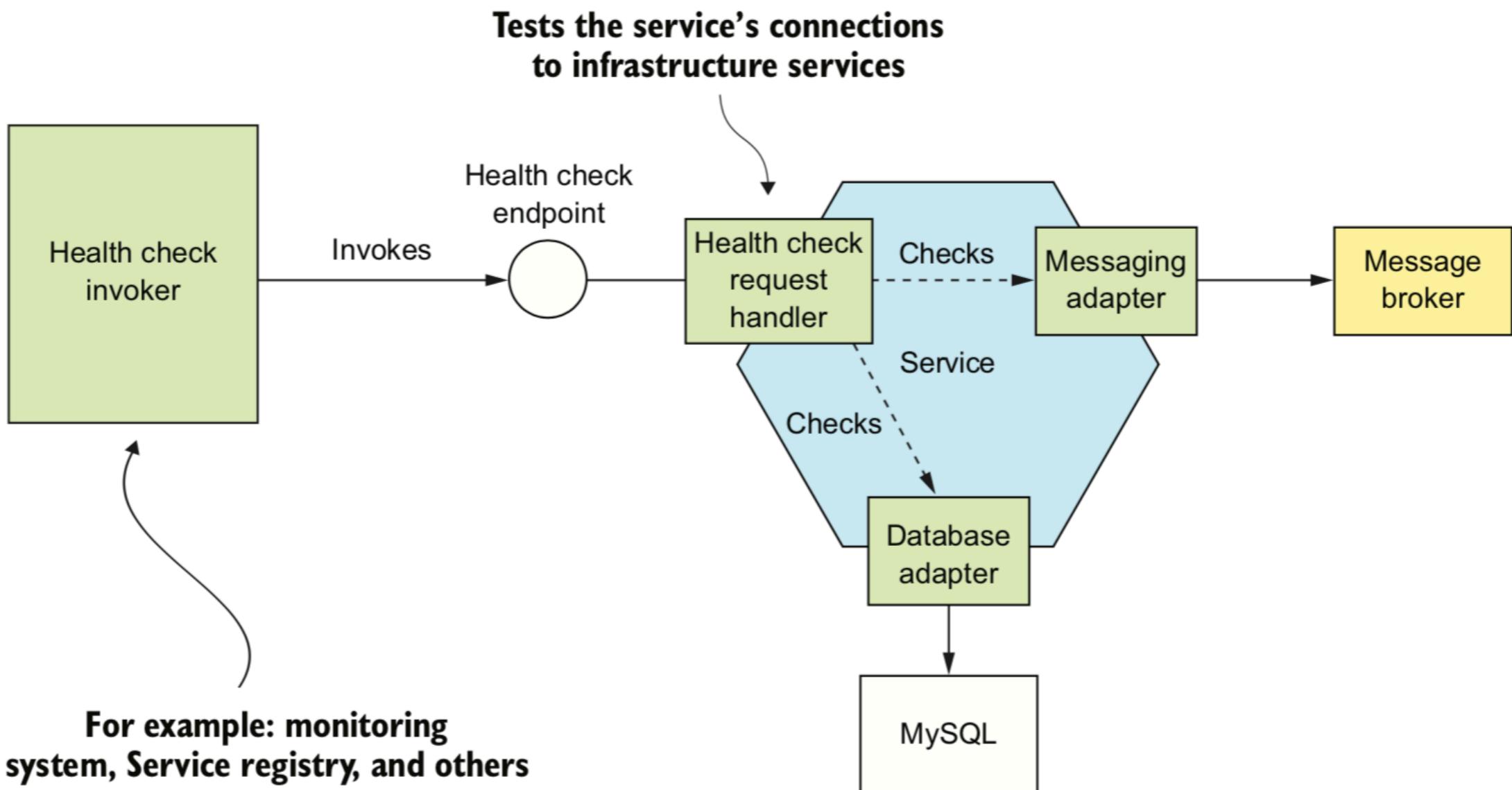


# Health check API

## 06-health-check-api



# Health check API

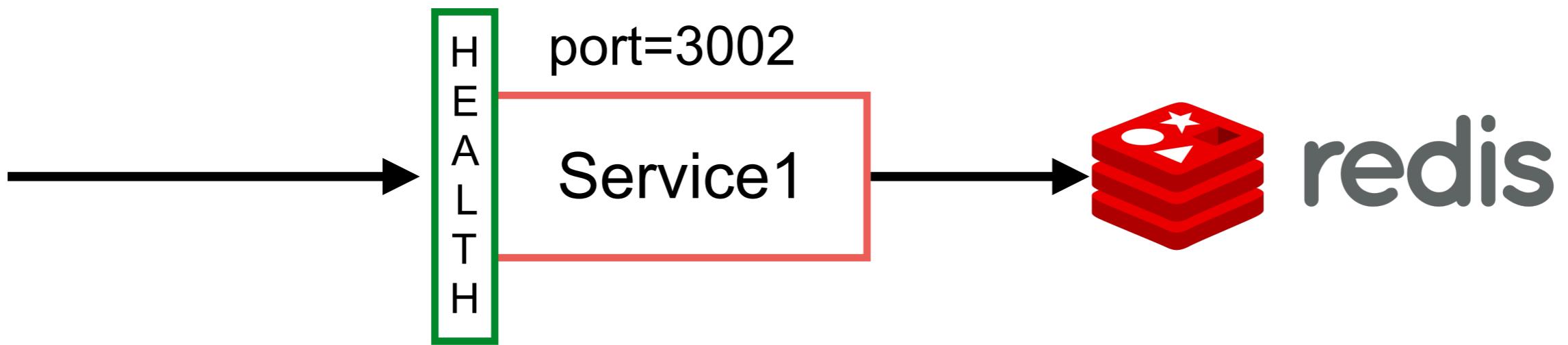


# Health check API

Liveness  
Readiness



# Health check API

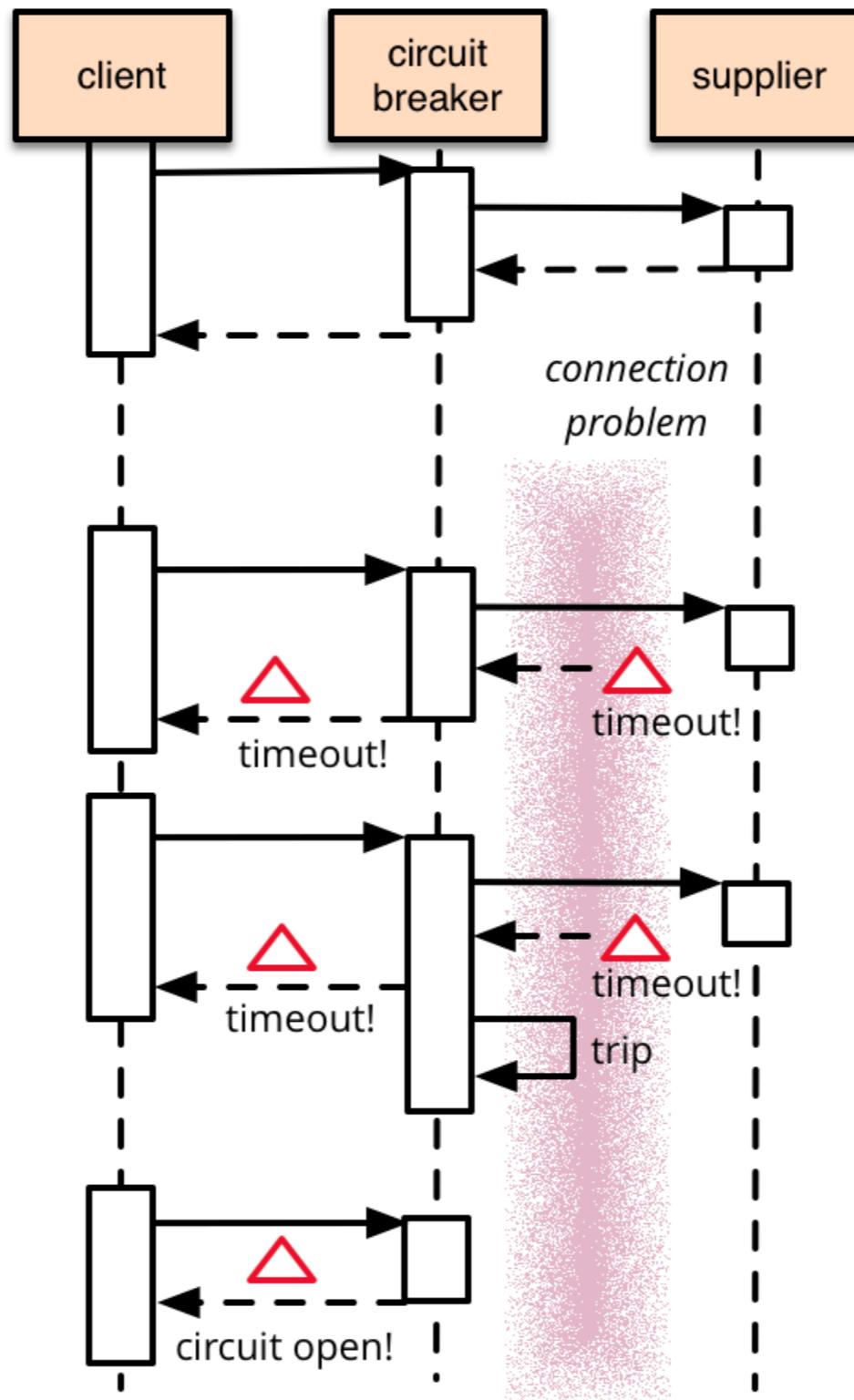


# Circuit breaker

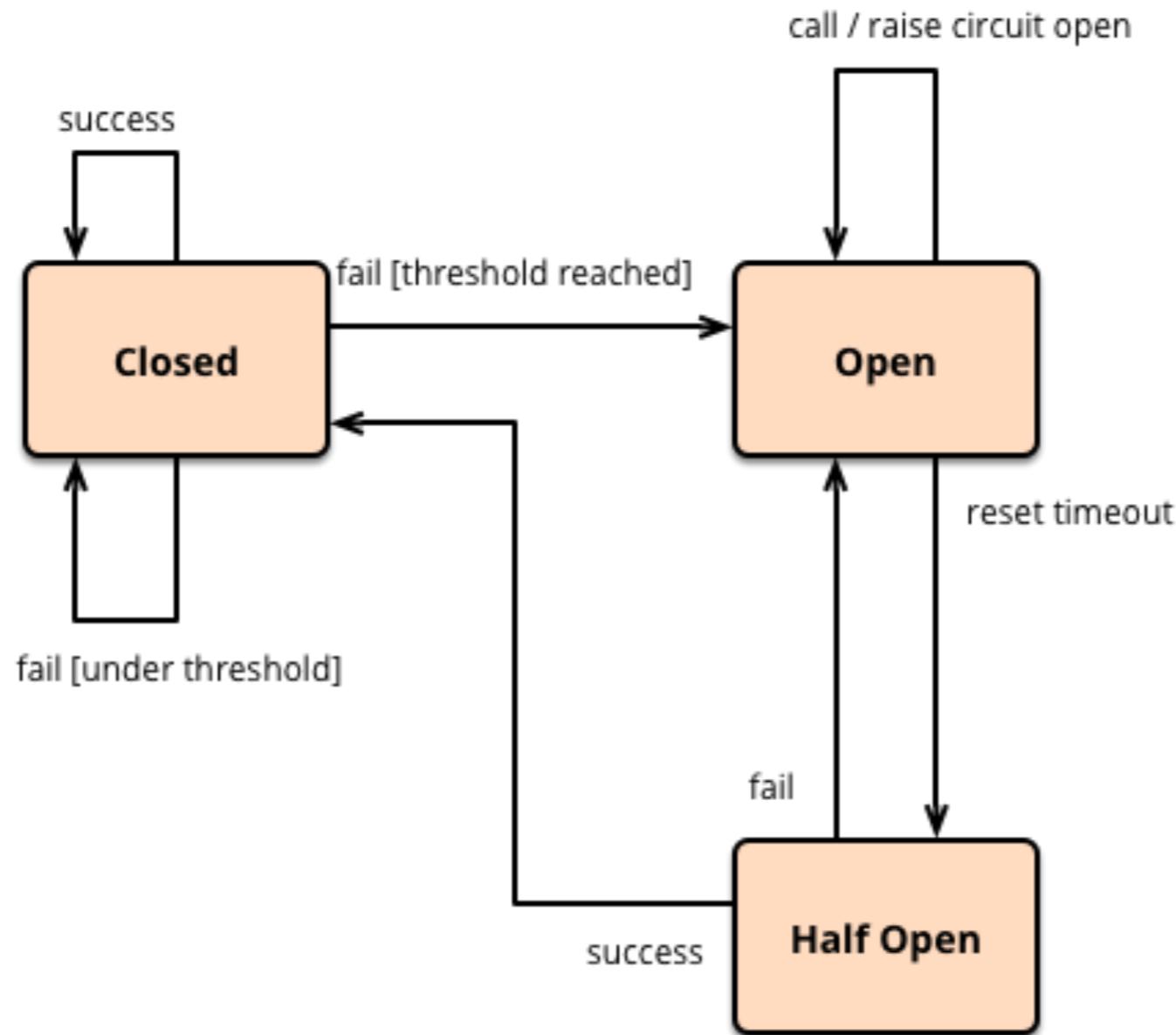
## 07-circuit-breaker



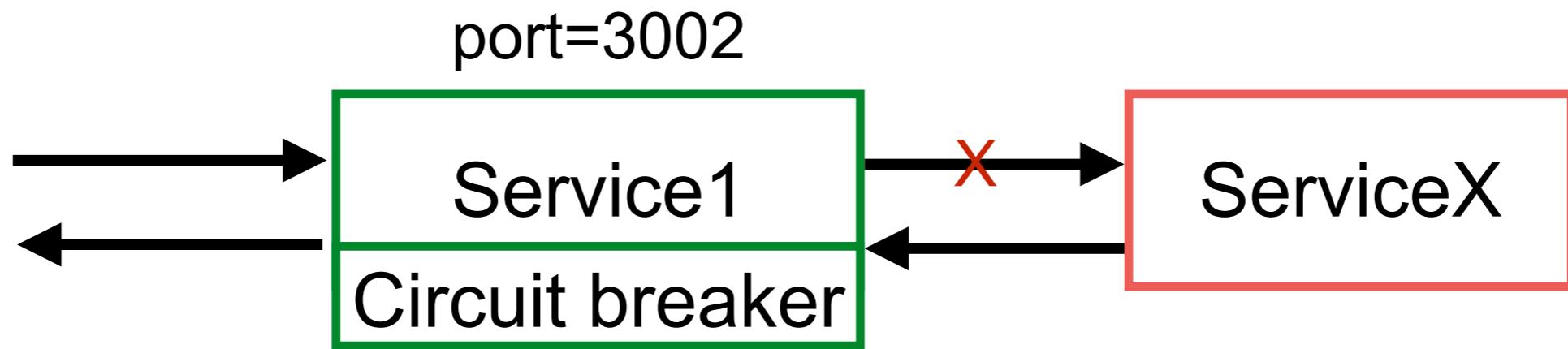
# Circuit breaker



# Circuit breaker



# Circuit breaker

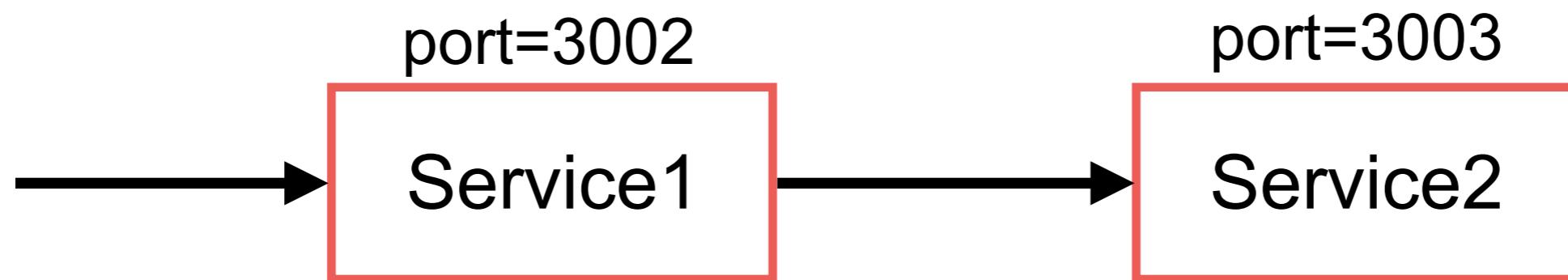


# Distributed tracing

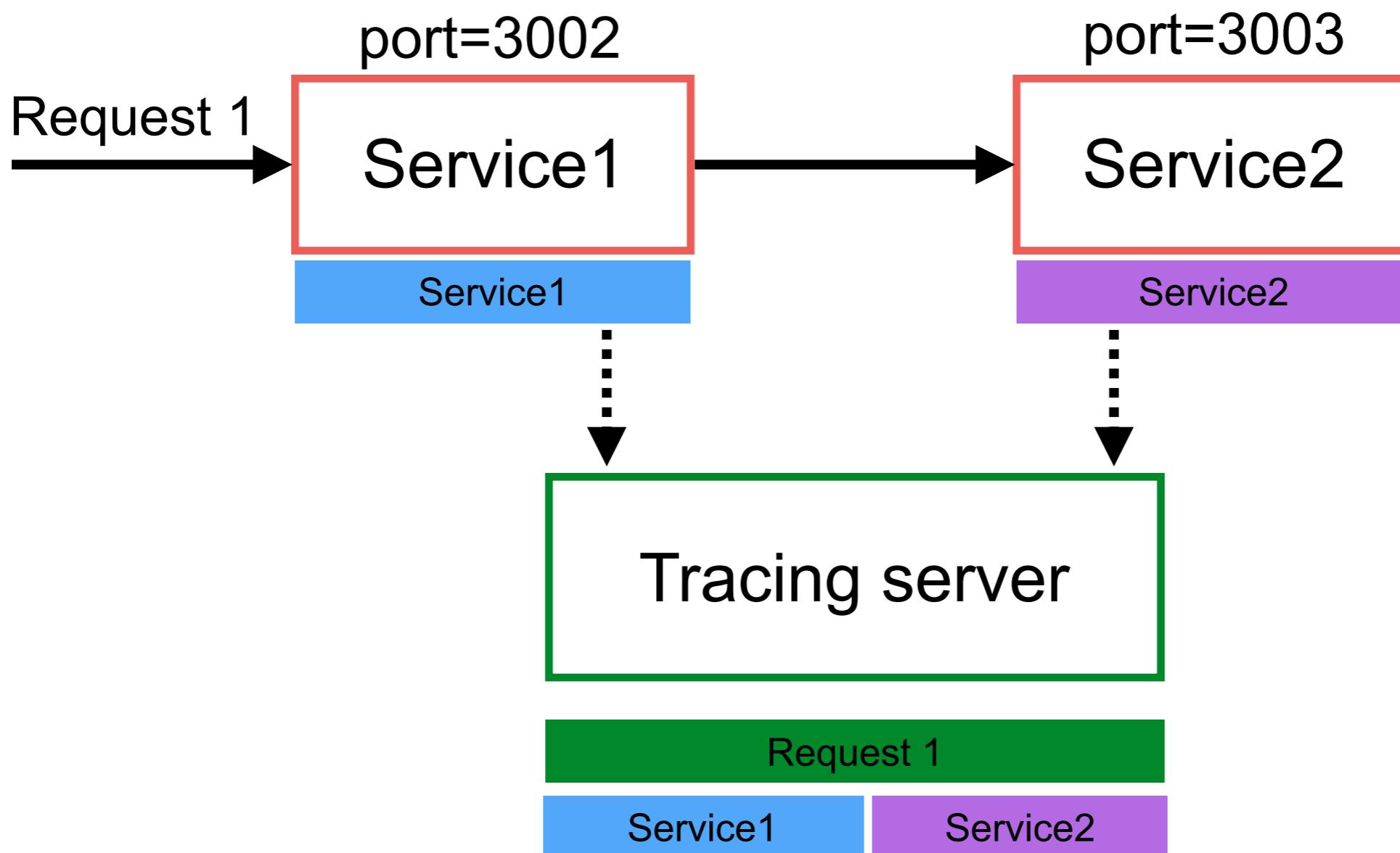
## 08-tracing



# Communication between services



# Distributed tracing



# Jaeger dashboard

http://localhost:16686

Jaeger UI    [Lookup by Trace ID...](#)    [Search](#)    [Compare](#)    [Dependencies](#)    [About Jaeger ▾](#)

### Find Traces

Service (0)

Select A Service

Operation (0)

all

Tags ⓘ

http.status\_code=200 error=true

Lookback

Last Hour

Min Duration

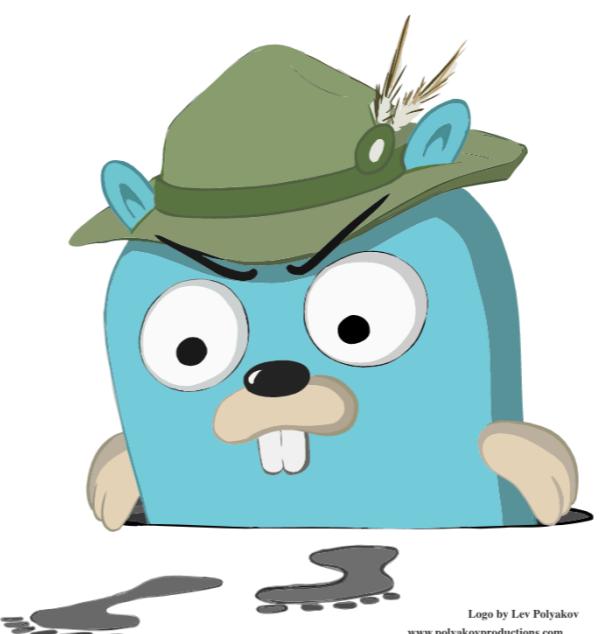
e.g. 1.2s, 100ms, 500us

Max Duration

e.g. 1.1s

Limit Results

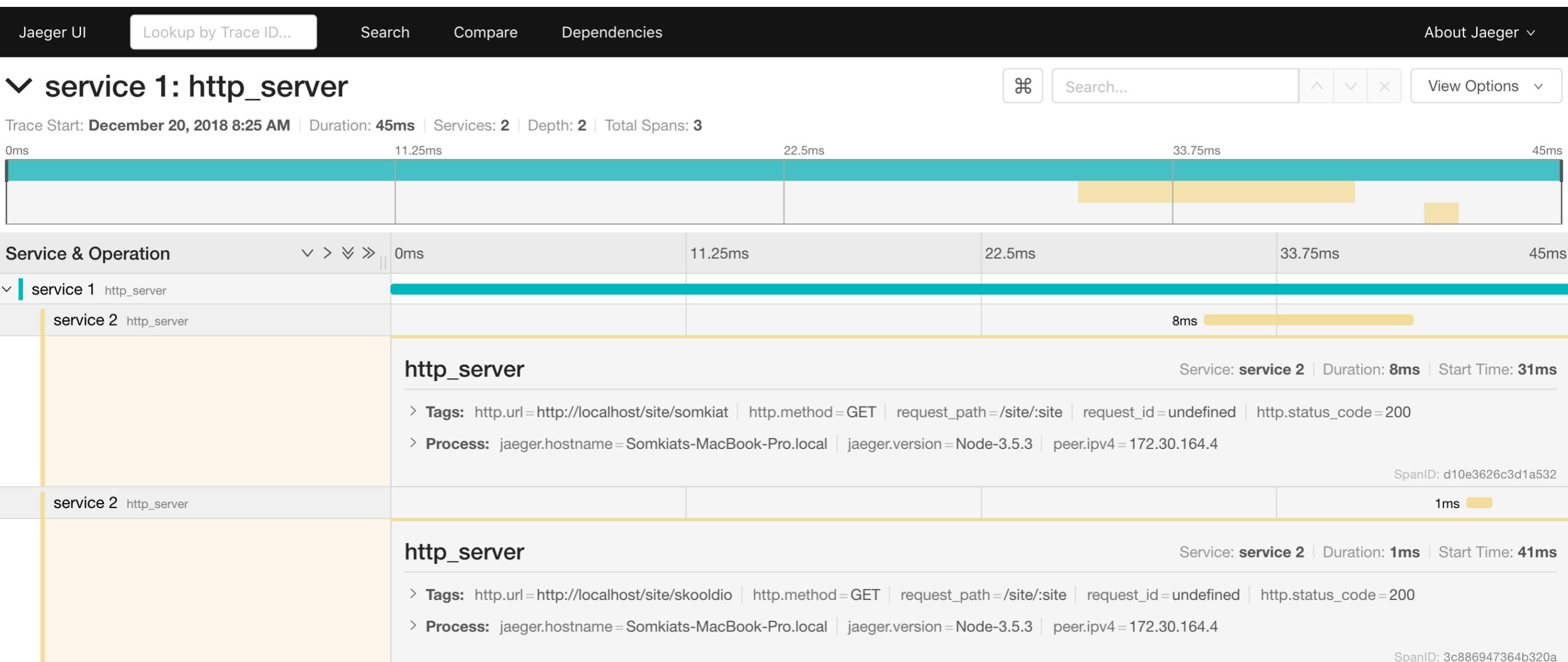
20



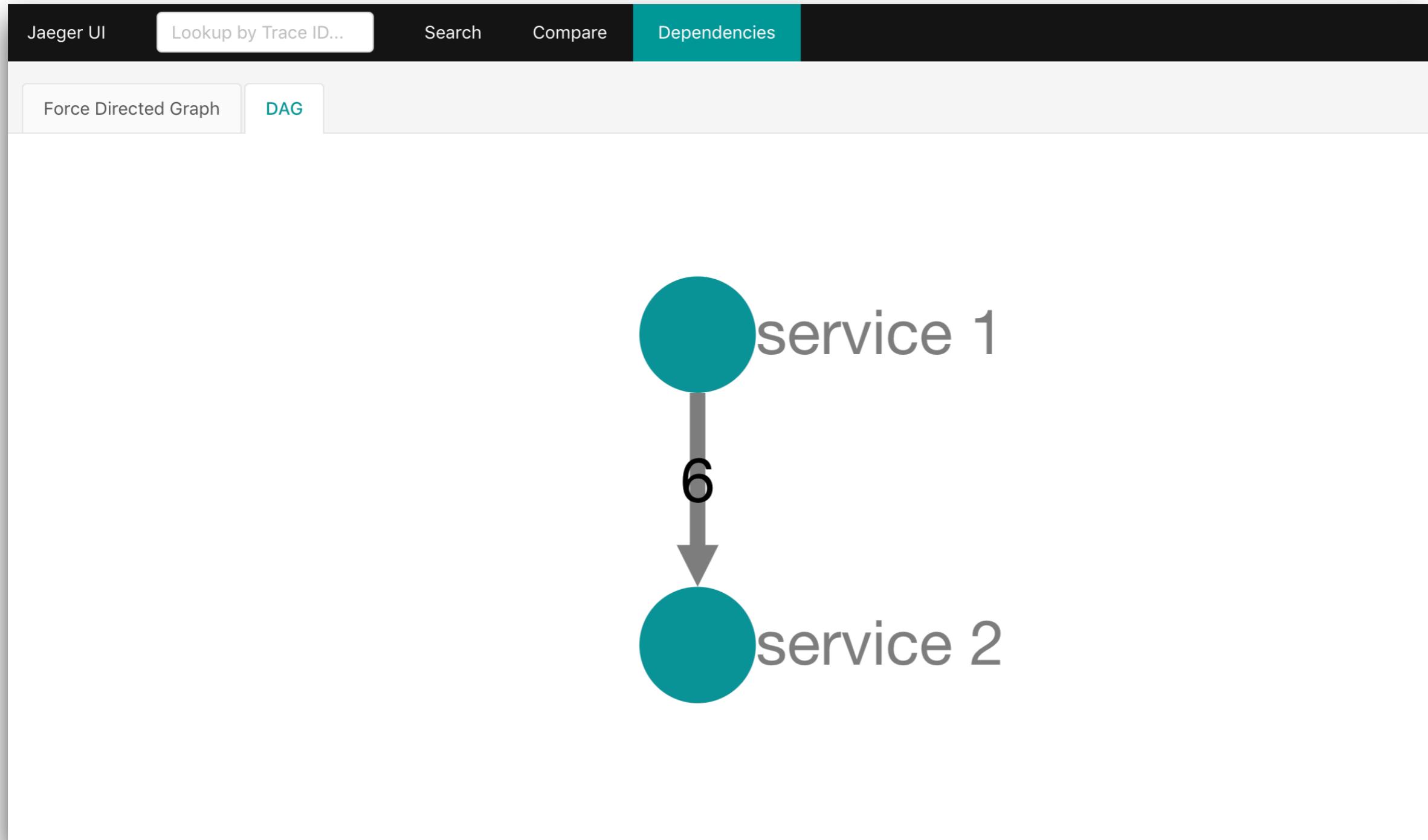
Logo by Lev Polyakov  
www.polyakovproductions.com



# See detail in each request



# Dependencies of services



# Monitoring

## 09-monitoring

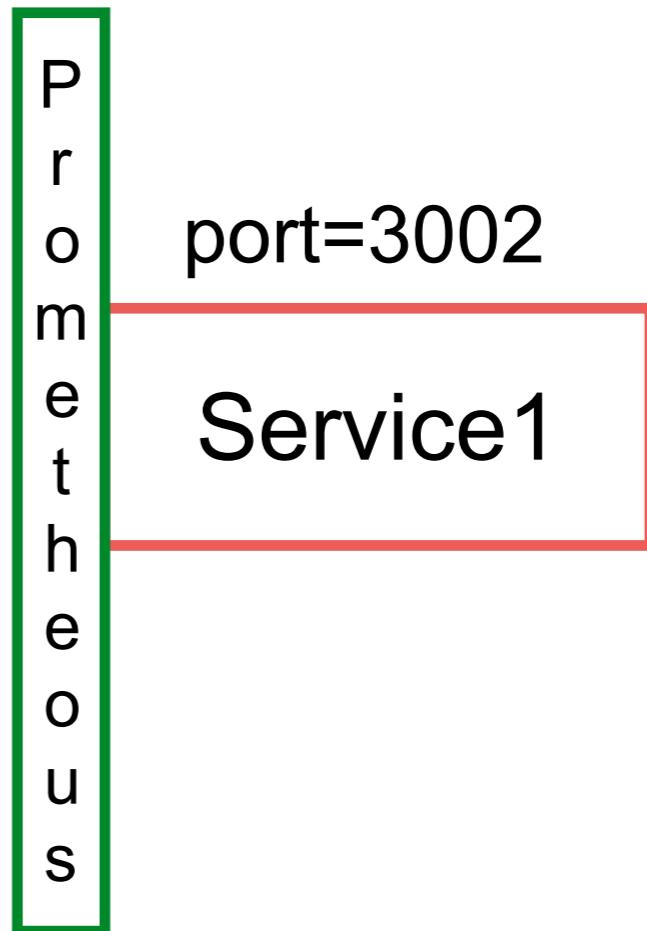


# Monitoring with Prometheus

Keep data from service



Prometheus



Visualize data



# Logging

## 10-logging



# Logging

Timestamps

Logging format

Log destinations (standard output/error)

Log levels



# Logging in distributed system

Timestamps

Logging format

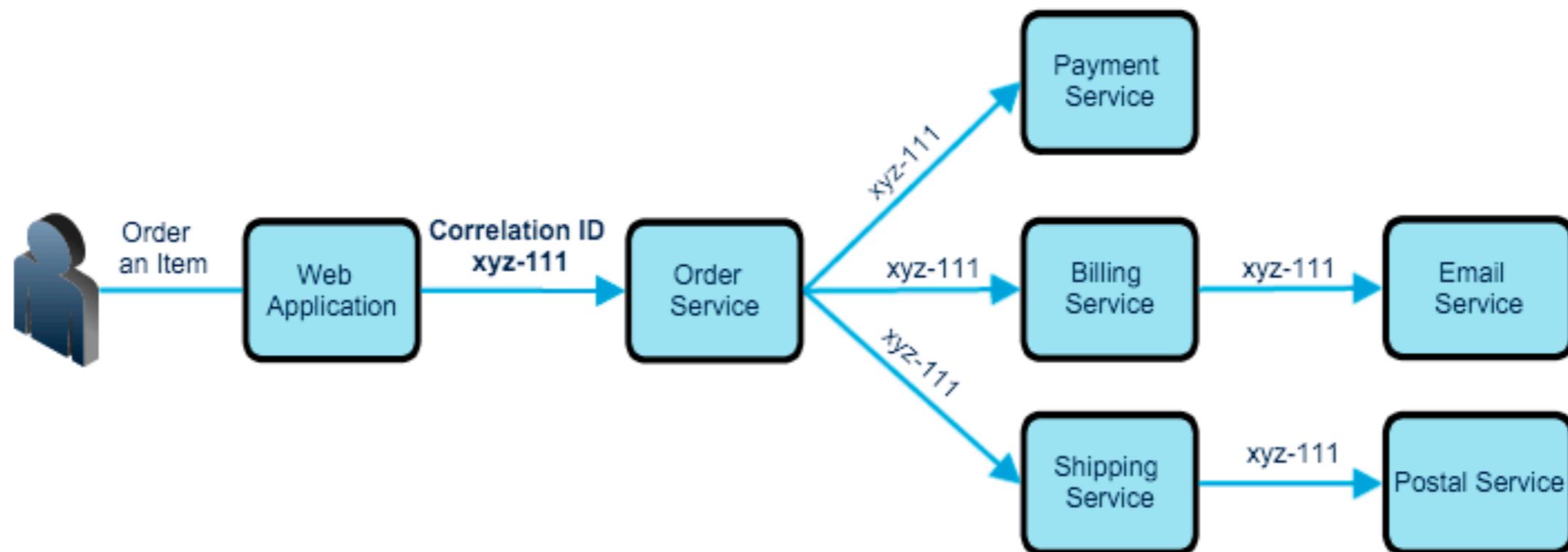
Log destinations (standard output/error)

Log levels

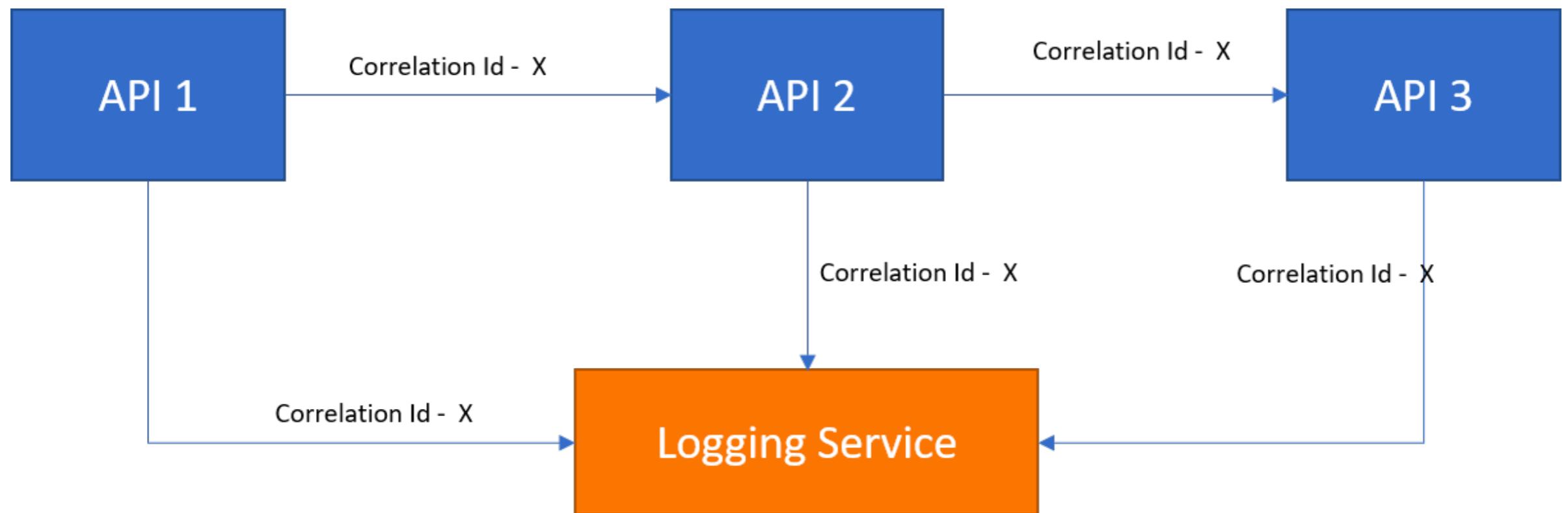
Correlation ID (UUID)



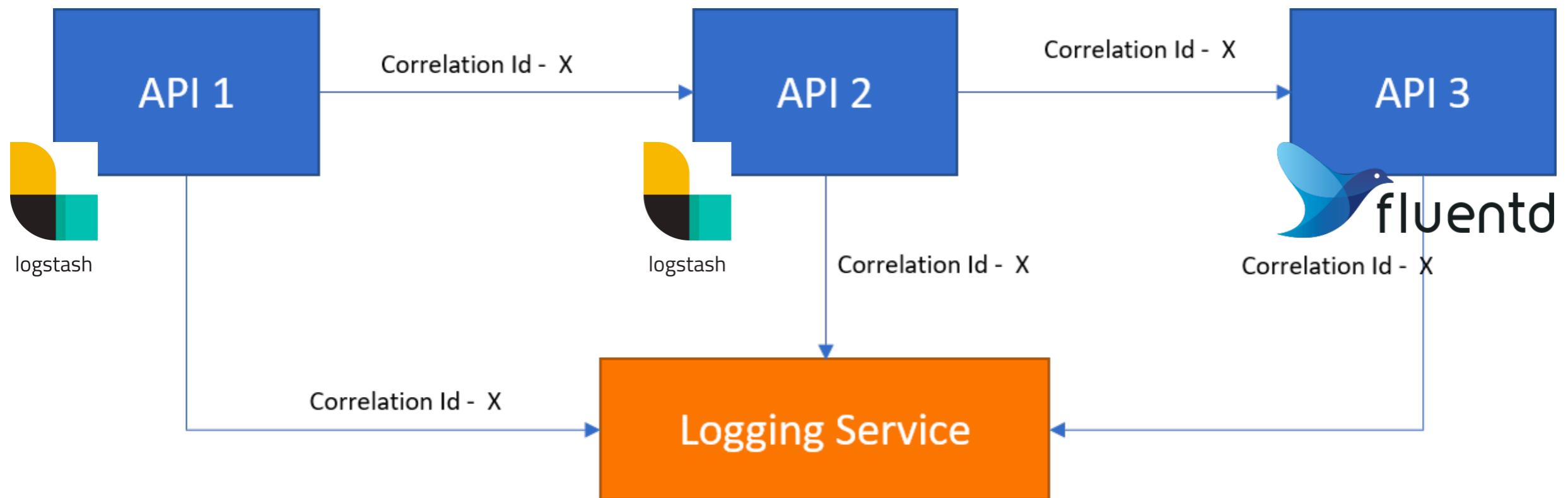
# Logging in distributed system



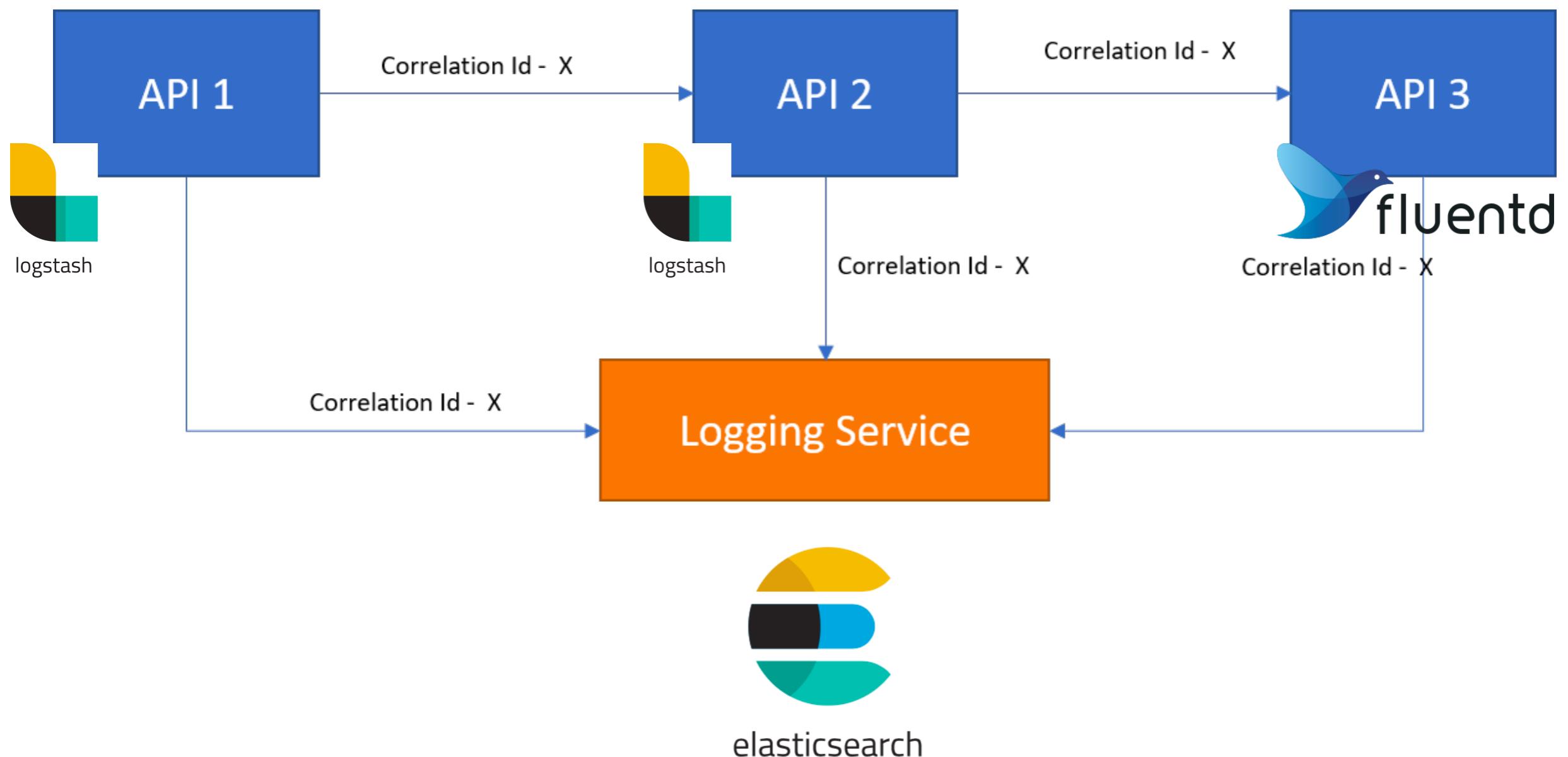
# Logging in distributed system



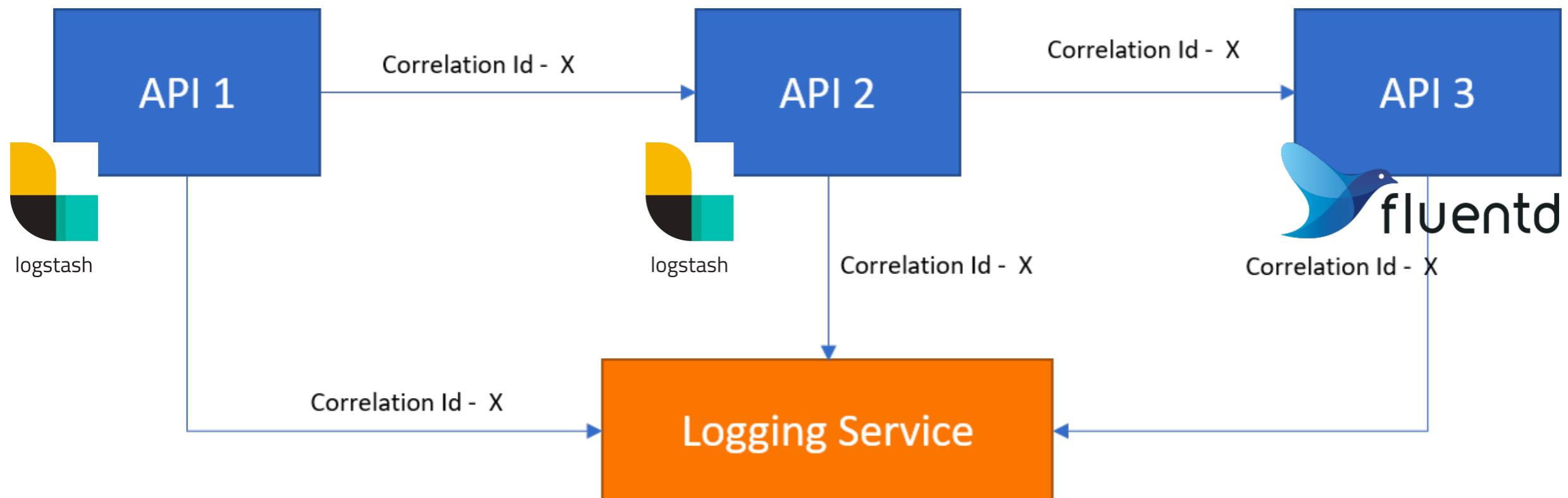
# Logging in distributed system



# Logging in distributed system



# Logging in distributed system

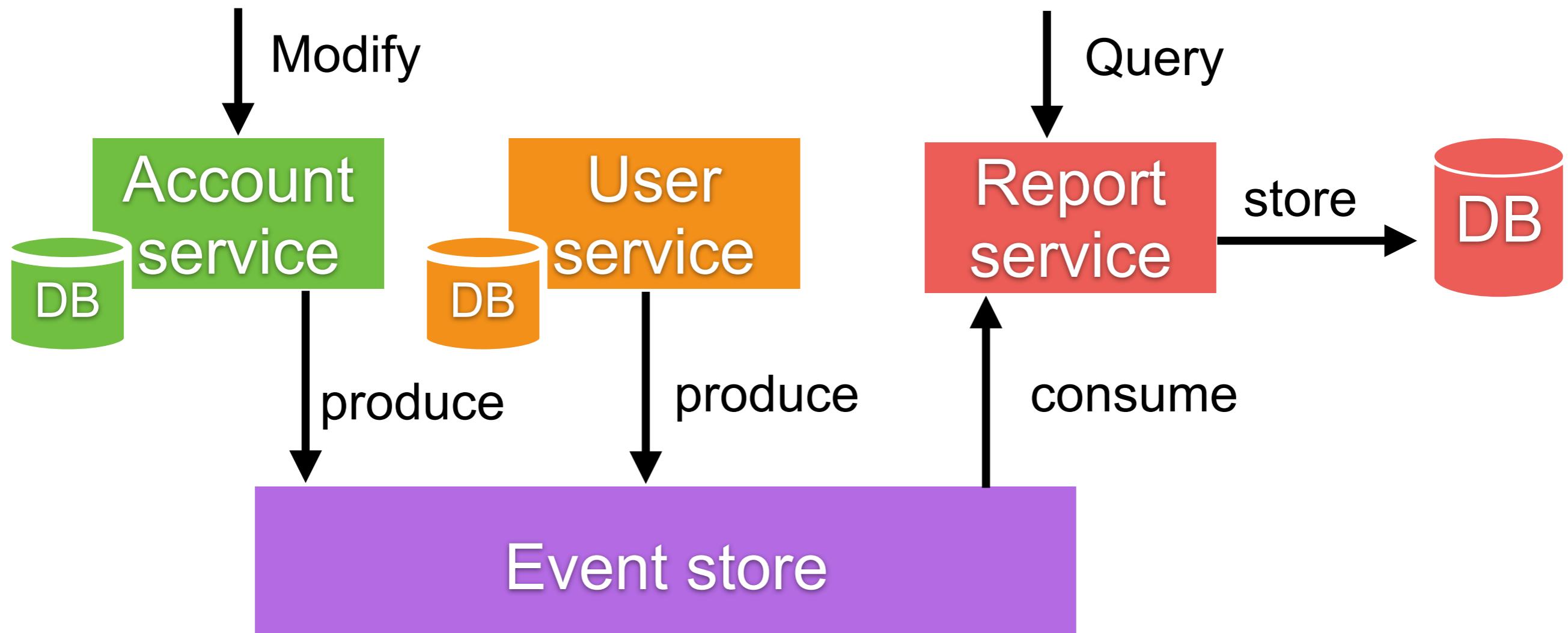


# CQRS

## 11-cqrs



# CQRS



 RabbitMQ

