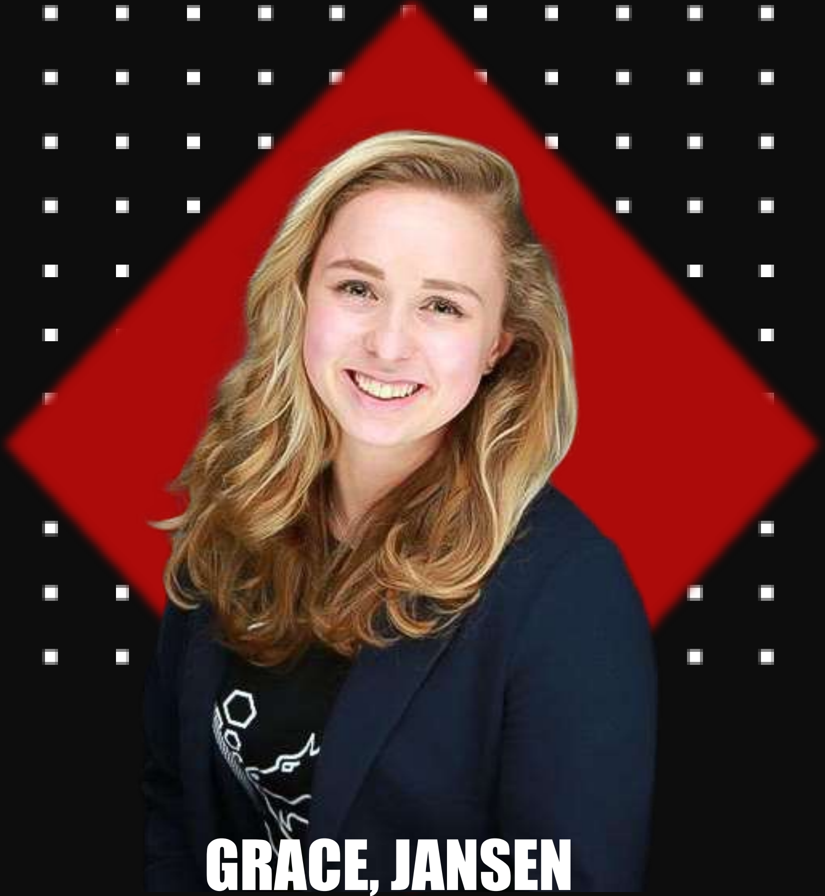
JAVA
USER
GROUP
CH

Through The Looking Glass:

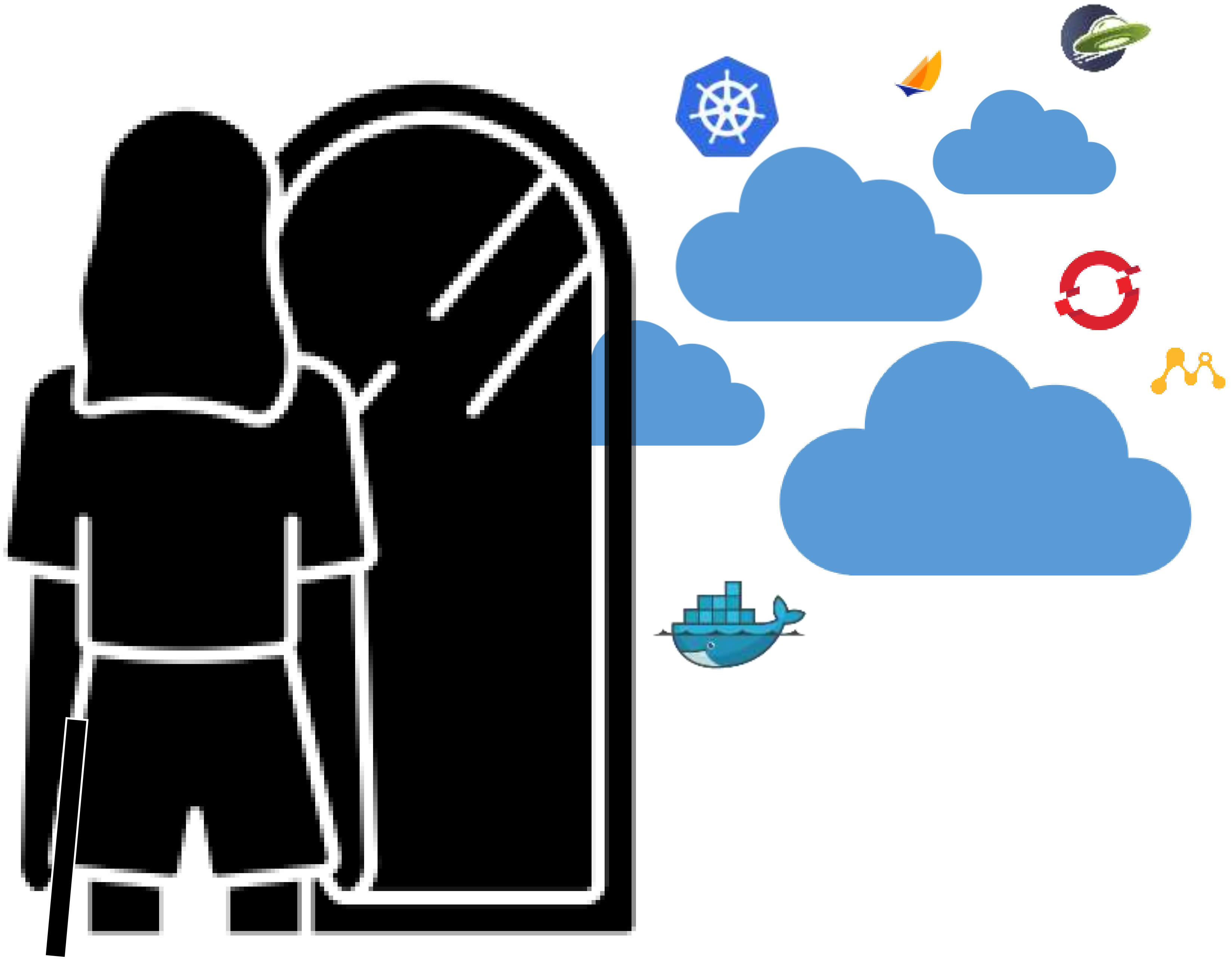
Effective observability for cloud native applications



GRACE, JANSEN
DEVELOPER ADVOCATE, IBM
@gracejansen27

Through the Looking-Glass...

- Phrase by Lewis Carroll
- The sequel to Alice's Adventures in Wonderland
- Alice passes through a mirror over a fireplace and finds herself (once more) in an enchanted land

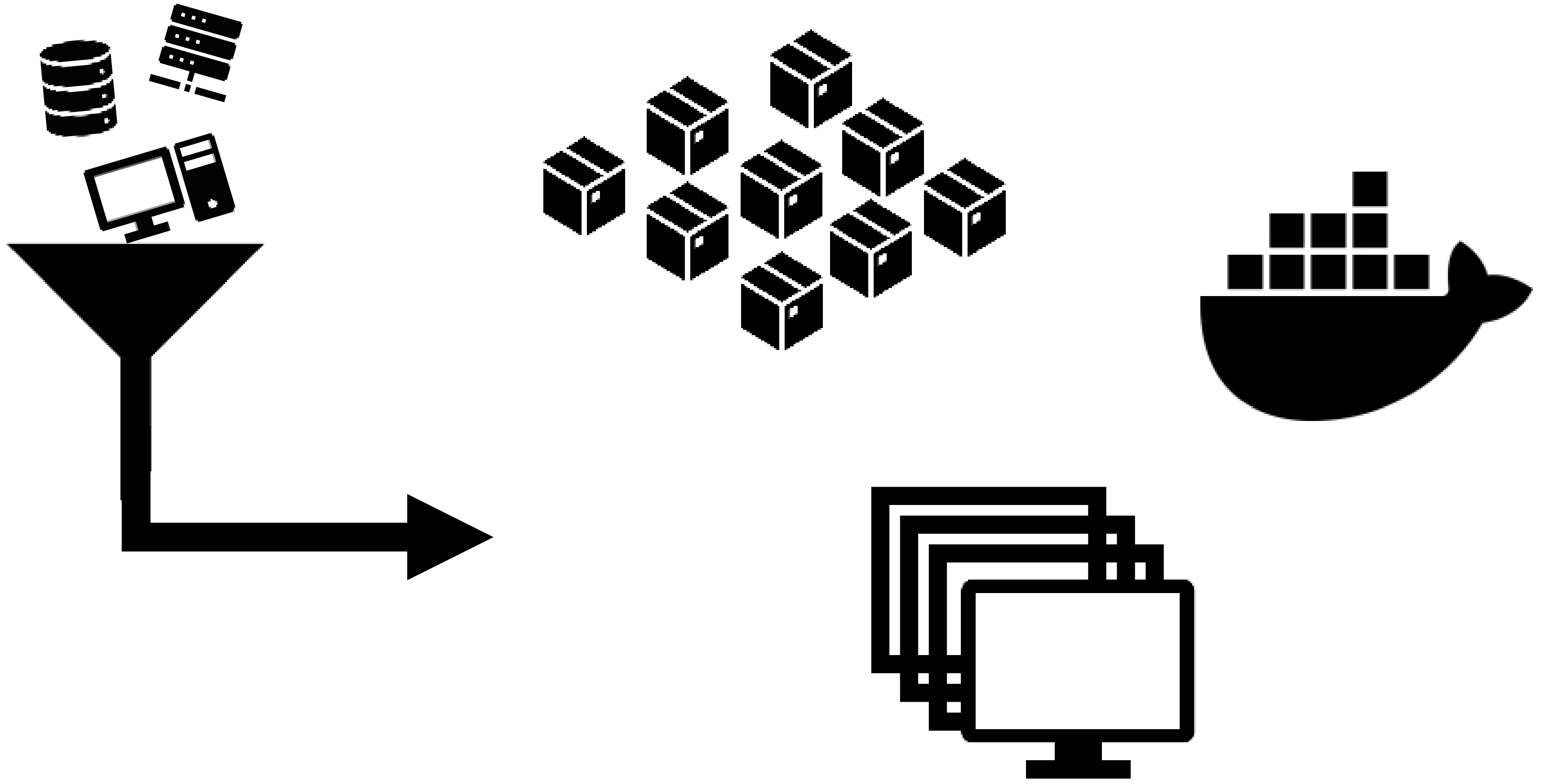


Agenda:

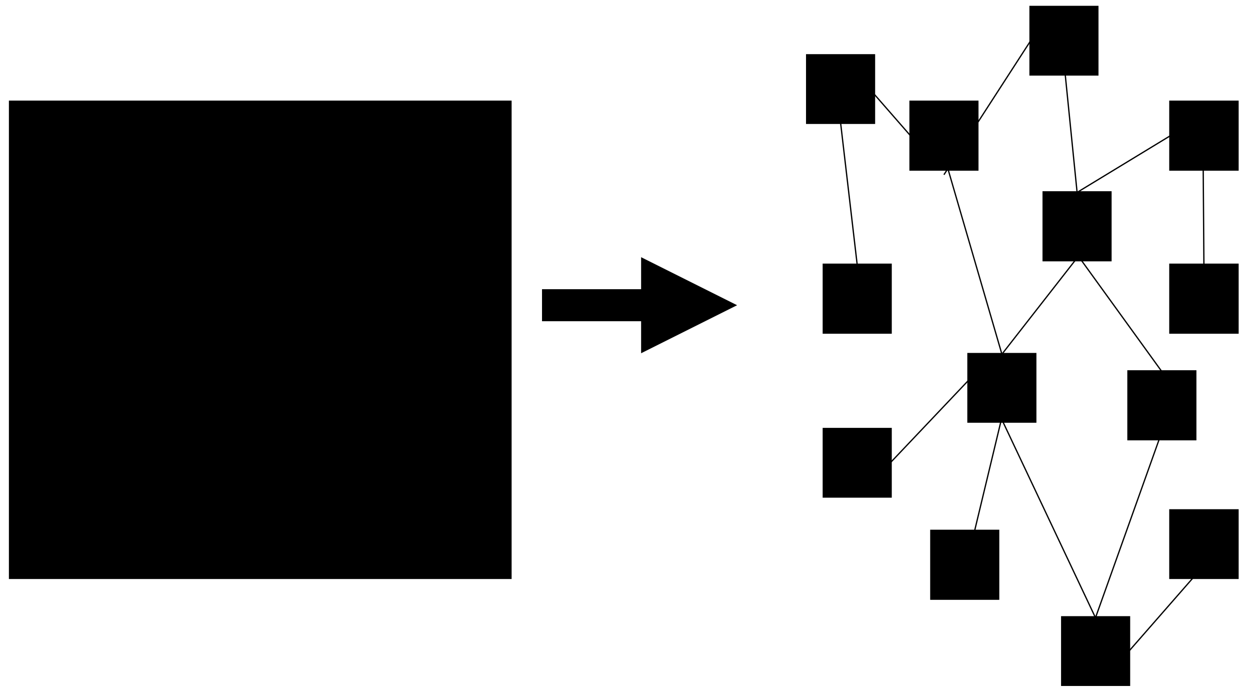
- Why do we need observability?
- What do we mean by “Observability”?
- How can we do this in our own apps?
- OpenTelemetry
- MicroProfile Telemetry 1.0
- Demo
- Summary and Resources

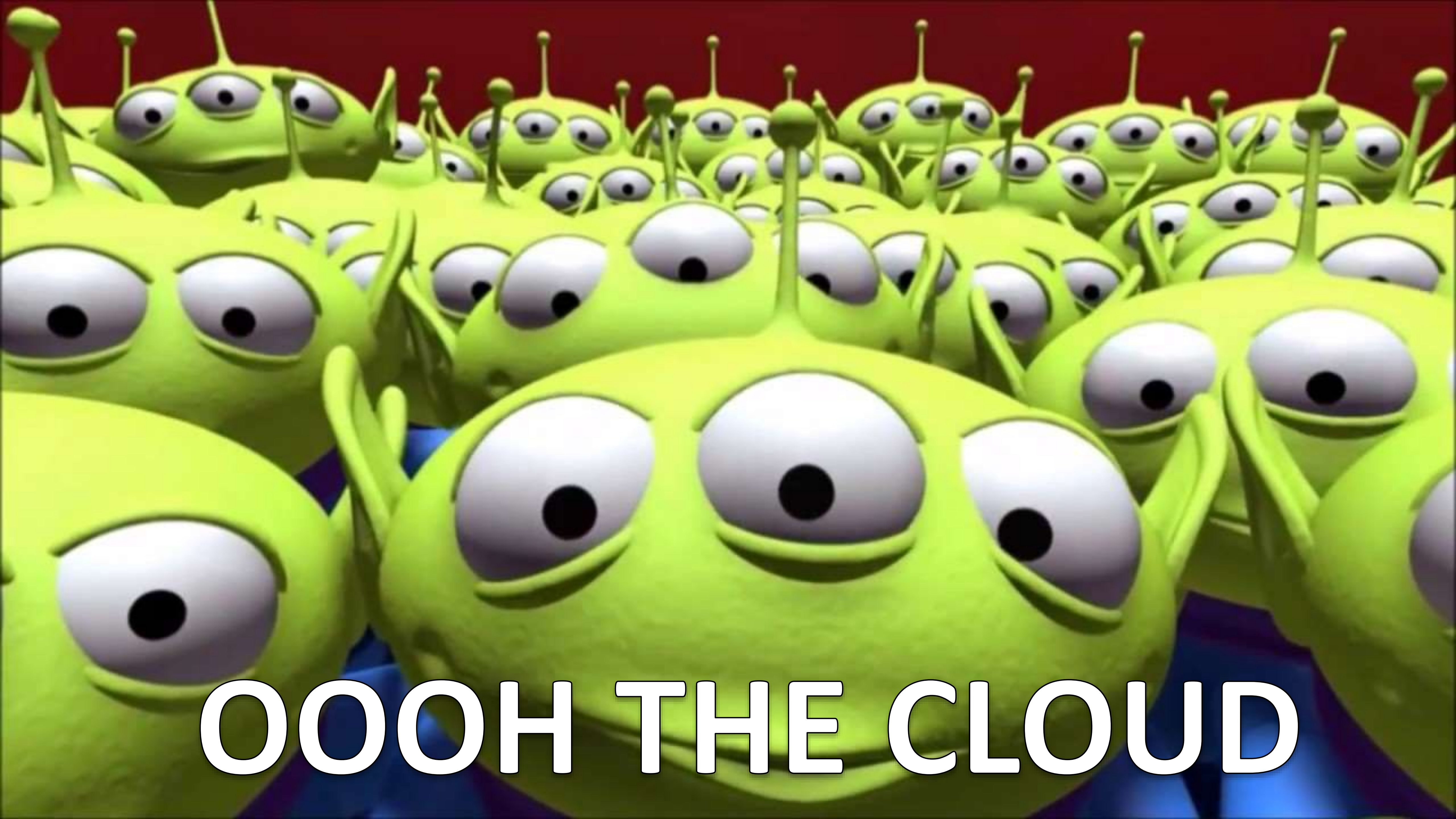
Why do we need observability?

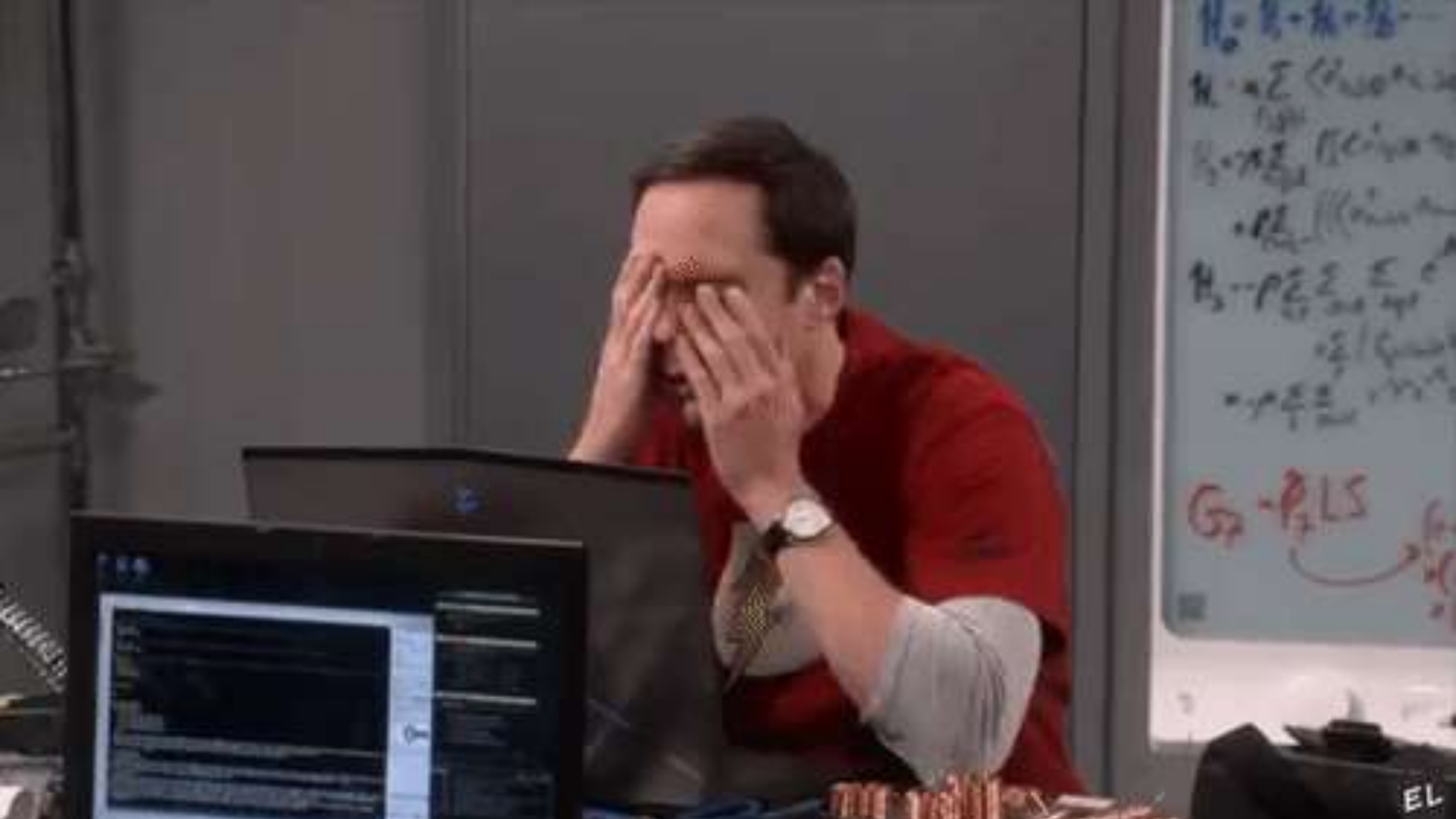
Evolution of Modern Infrastructure



Evolution of Applications




OOOH THE CLOUD

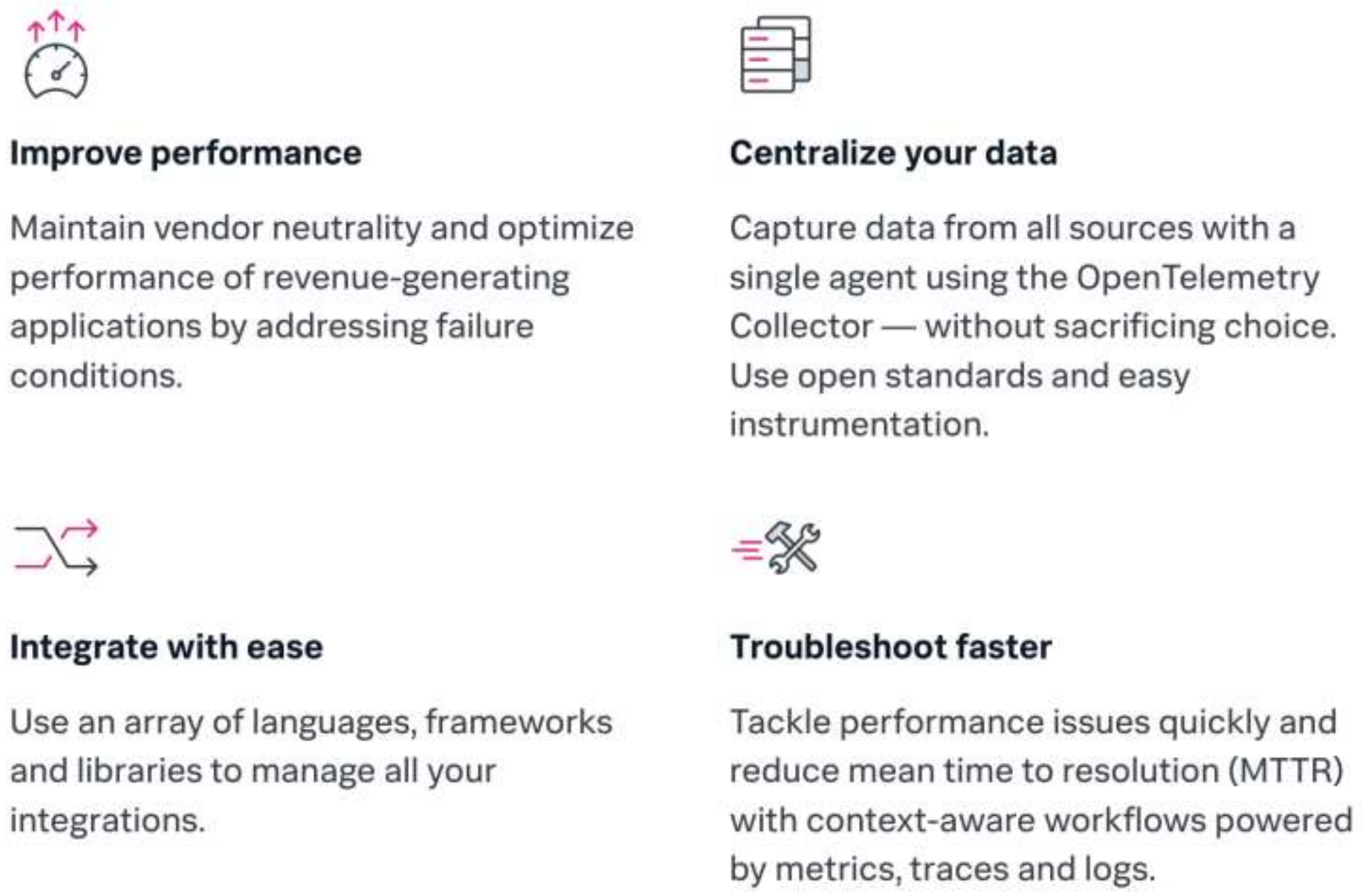


15 Factor Applications (revised from 12 factors)

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://developer.ibm.com/articles/15-factor-applications/

Why it matters



Improve performance

Maintain vendor neutrality and optimize performance of revenue-generating applications by addressing failure conditions.

Centralize your data

Capture data from all sources with a single agent using the OpenTelemetry Collector — without sacrificing choice. Use open standards and easy instrumentation.

Integrate with ease

Use an array of languages, frameworks and libraries to manage all your integrations.

Troubleshoot faster

Tackle performance issues quickly and reduce mean time to resolution (MTTR) with context-aware workflows powered by metrics, traces and logs.

What do we mean by “Observability”?

What is observability?

- **In general…**
  - observability is the extent to which you can understand the internal state or condition of a complex system based only on knowledge of its external outputs

What is observability?

- **In general…**
  - observability is the extent to which you can understand the internal state or condition of a complex system based only on knowledge of its external outputs

- **In IT and cloud computing…**
  - observability also refers to software tools and practices for aggregating, correlating and analyzing a steady stream of performance data from a distributed application along with the hardware and network it runs on, in order to more effectively monitor, troubleshoot and debug the application and the network

Observability vs Monitoring

- **Monitoring** consists in using tools/techniques that highlight that an issue occurred. A monitoring system could raise a warning when:
    - average response time is getting slower and slower;
    - a growing number of requests result in HTTP 500 – internal server error;
    - application crashes;

- **Observability** is the ability to measure the internal states of a system by examining its outputs (Control theory definition).

- An application is “observable” when it provides detailed visibility into its behavior and always allows identifying the root cause of an issue.

How can we do this in our own apps?

Implementing Observability

1. Instrument systems and applications to collect relevant data (e.g. metrics, traces, and logs).

Implementing Observability

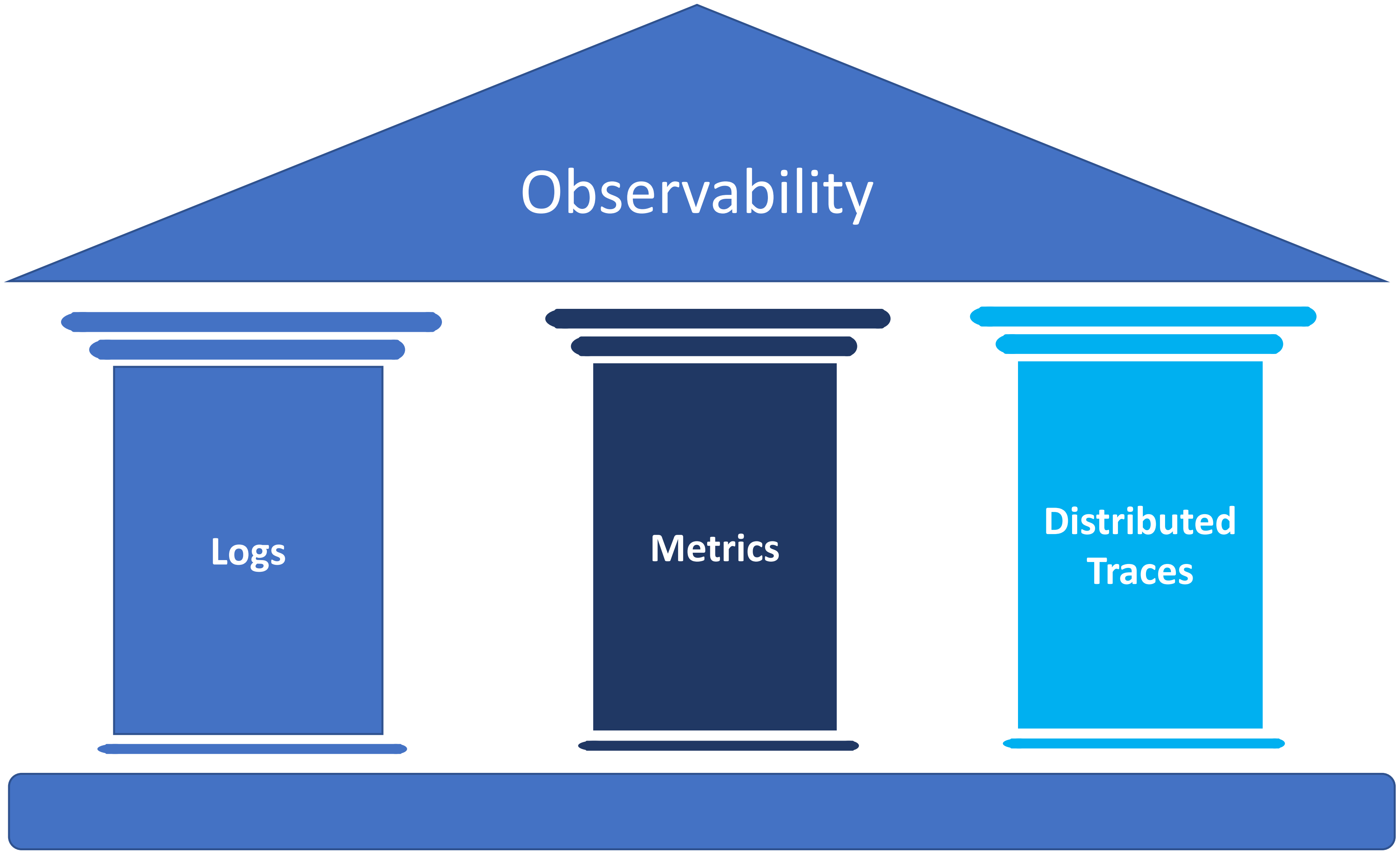
1. Instrument systems and applications to collect relevant data (e.g. metrics, traces, and logs).
2. Send this data to a separate external system that can store and analyze it.

Implementing Observability

1. Instrument systems and applications to collect relevant data (e.g. metrics, traces, and logs).
2. Send this data to a separate external system that can store and analyze it.
3. Provide visualizations and insights into systems as a whole (including query capability for end users).

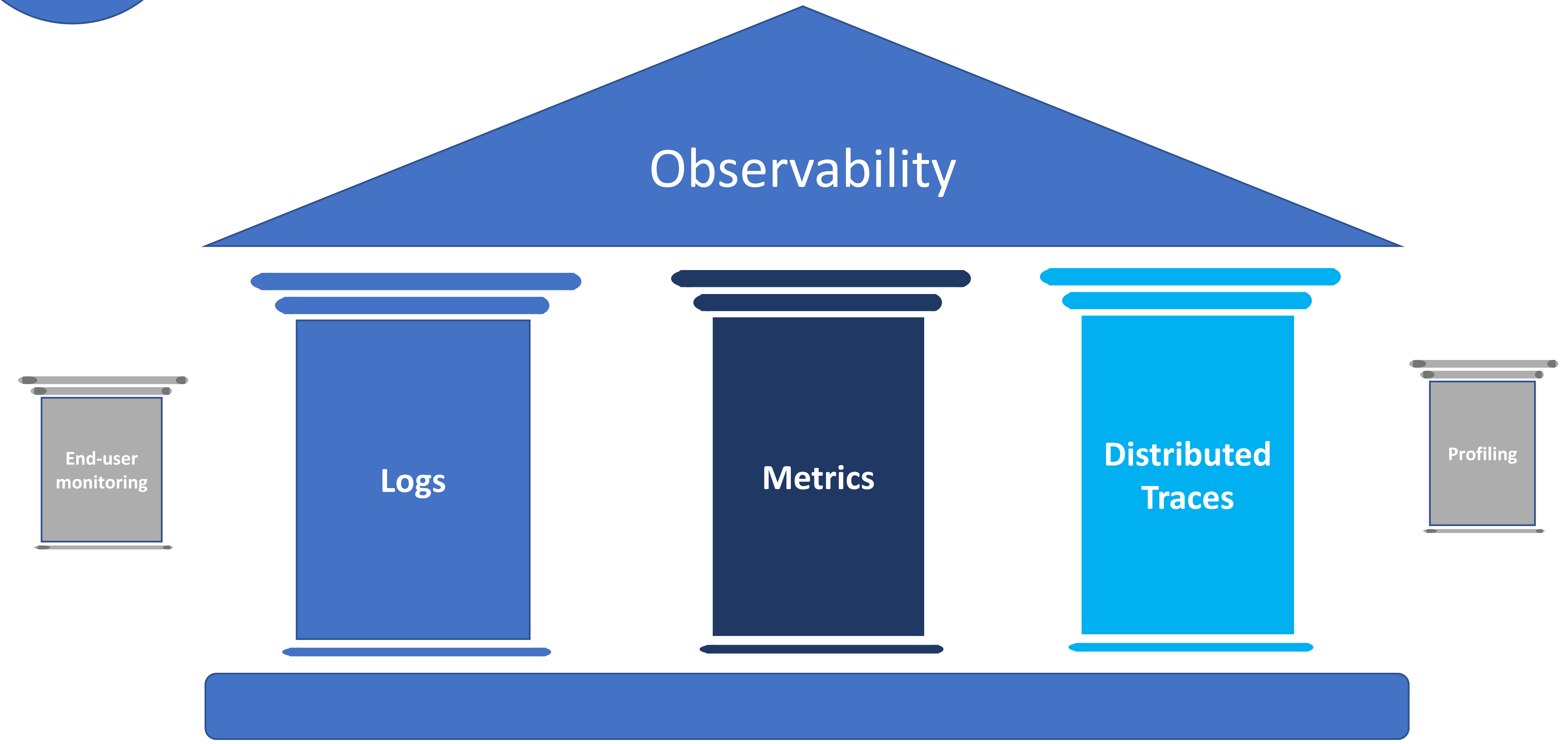
1

Instrumentation: The Three Pillars



1

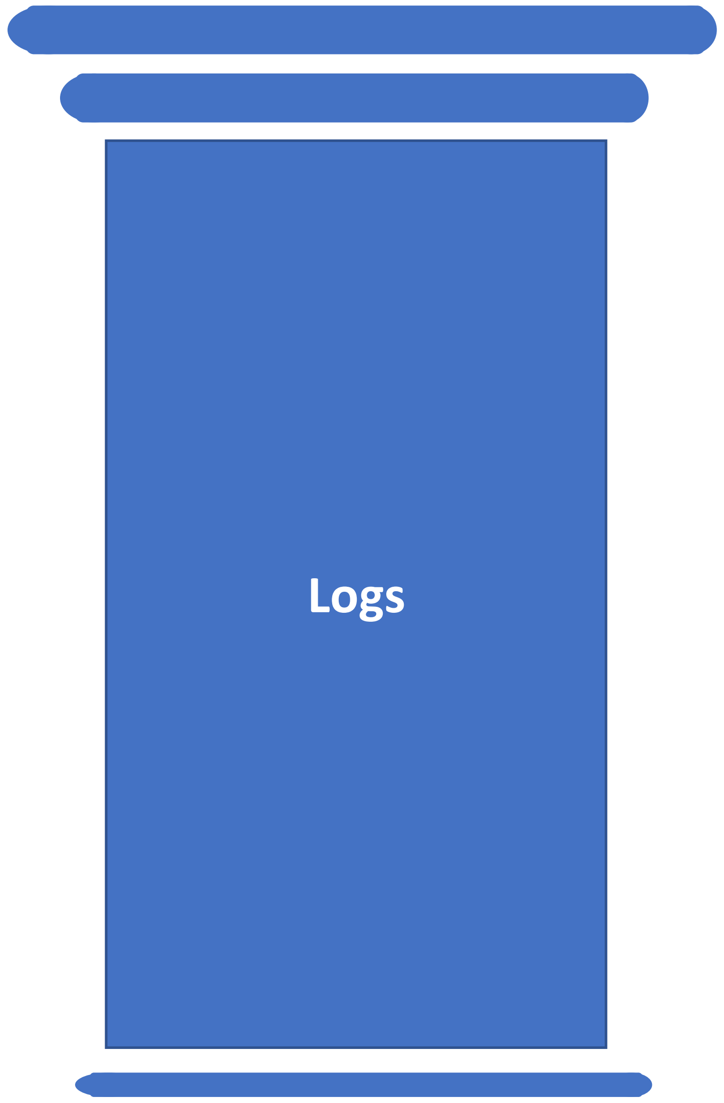
Instrumentation: The Three Pillars



Instrumentation: Logs

Logs

- a timestamped message emitted by services or other application components, providing coarser-grained or higher-level information about system behaviors (like errors, warnings, etc) and typically will be stored in a set of log files.
- not necessarily associated with any particular user request or transaction



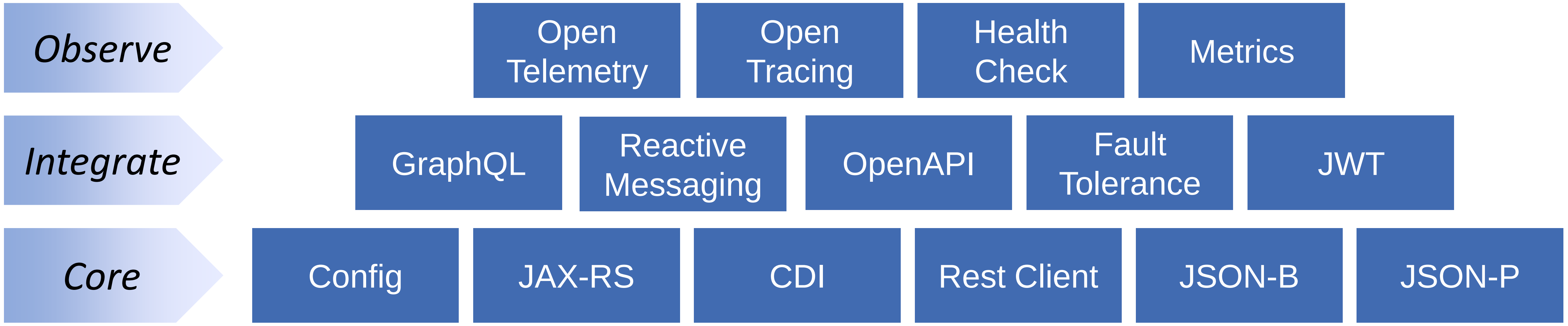
Instrumentation: Metrics

Metrics

- aggregations of numeric data about infrastructure or an application over a period of time. Examples include system error rates, CPU utilization, and request rates for a given service.

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

https://microprofile.io/



Open cloud-native Java APIs

https://microprofile.io/compatible

Compatible Runtimes

Compatible with MicroProfile APIs	2.x and 3.x	4.x	5.x	6.x
Open Liberty	x	x	x	x
WebSphere Liberty	x	x	x	x
Quarkus	x	x		
Payara Micro	x	x	x	
WildFly	x	x	x	
Payara Server	x	x	x	
TomEE	x		x	
KumuluzEE	x			
Thorntail	x			
JBoss EAP XP	x			
Helidon	x		x	
Apache Launcher			x	

MicroProfile Metrics

“This specification aims at providing a unified way for Microprofile servers to export Monitoring data ("Telemetry") to management agents and also a unified Java API, that all (application) programmers can use to expose their telemetry data.”

Instrumentation: Traces

- Distributed traces (i.e. Traces)
  - records the paths taken by requests (made by an application or end user) as they disseminate through multi-service architectures, like microservice, macroservice, and serverless applications.

Key Tracing Concepts

- **Traces**
  - Traces represent requests and consist of multiple spans.
- **Spans**
  - Spans are representative of single operations in a request. A span contains a name, time-related data, log messages, and metadata to give information about what occurs during a transaction.

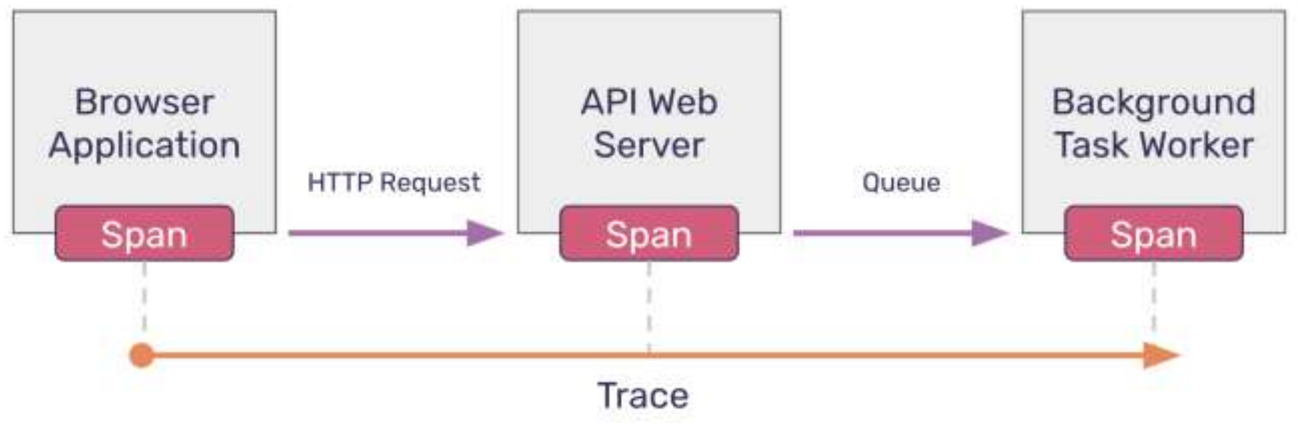


Image: https://blog.sentry.io/2021/08/12/distributed-tracing-101-for-full-stack-developers/

Key Tracing Concepts

- **Context**
  - Context is an immutable object contained in the span data to identify the unique request that each span is a part of. This data is required for moving trace information across service boundaries, allowing developers to follow a single request through a potentially complex distributed system.

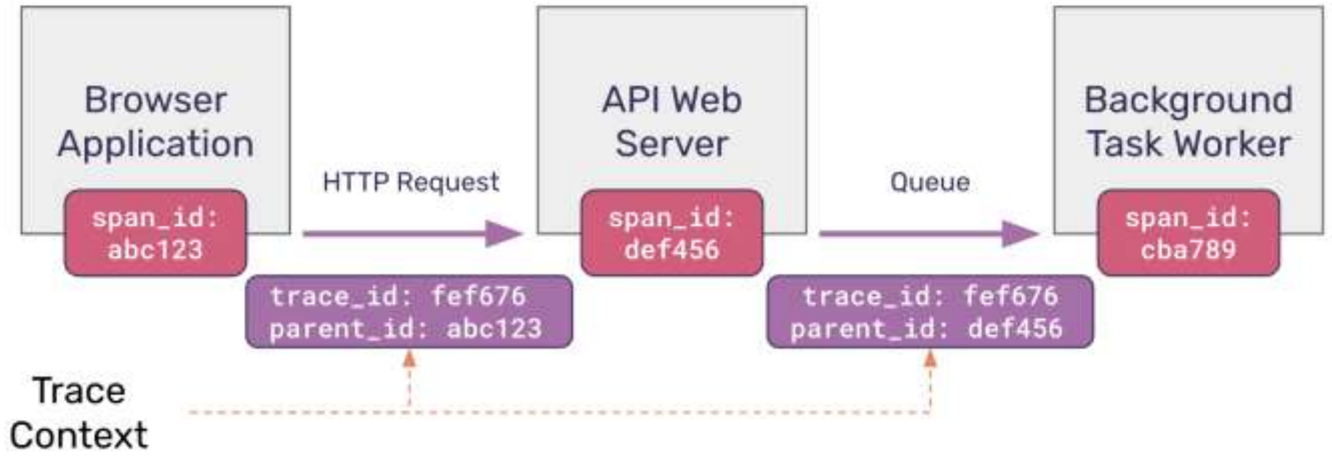


Image: https://blog.sentry.io/2021/08/12/distributed-tracing-101-for-full-stack-developers/

OpenTelemetry

Open Telemetry

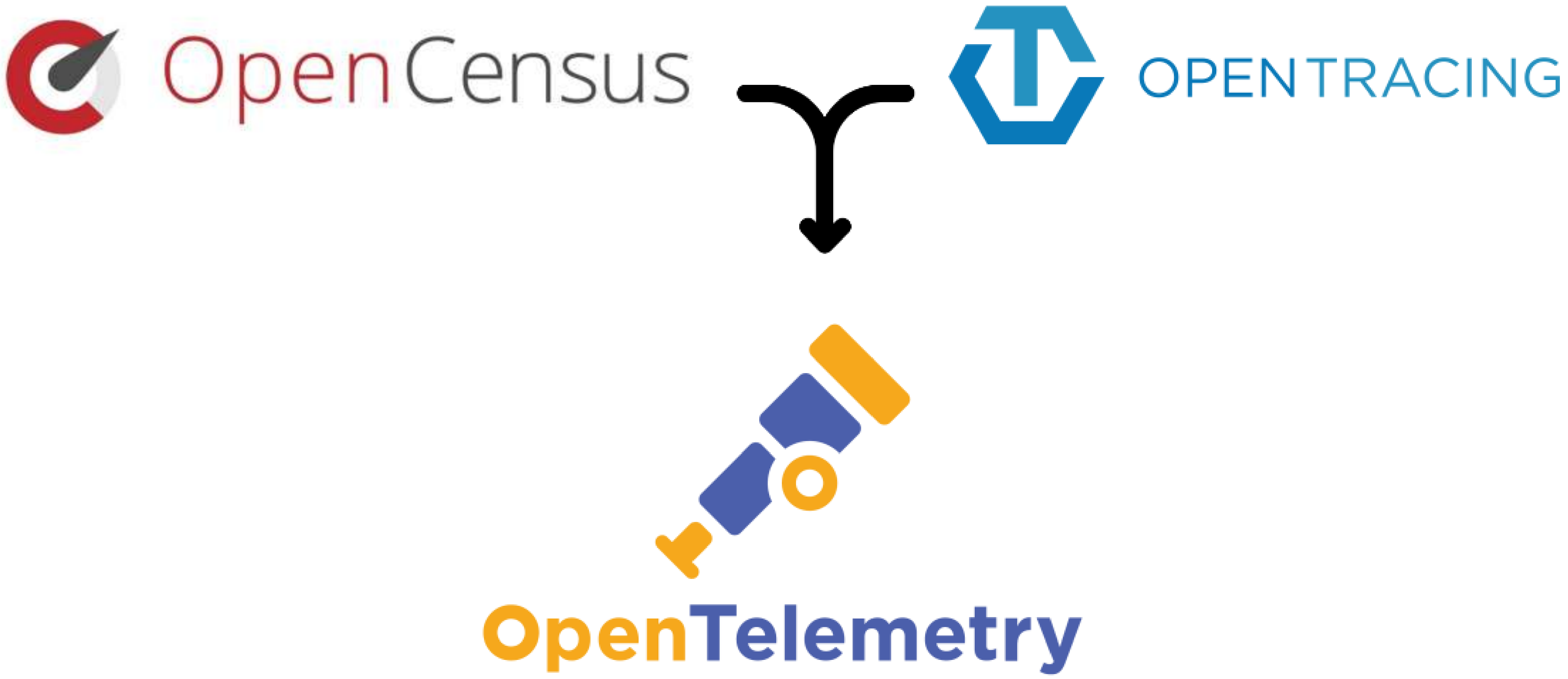
- High-quality, ubiquitous, and portable telemetry to enable effective observability

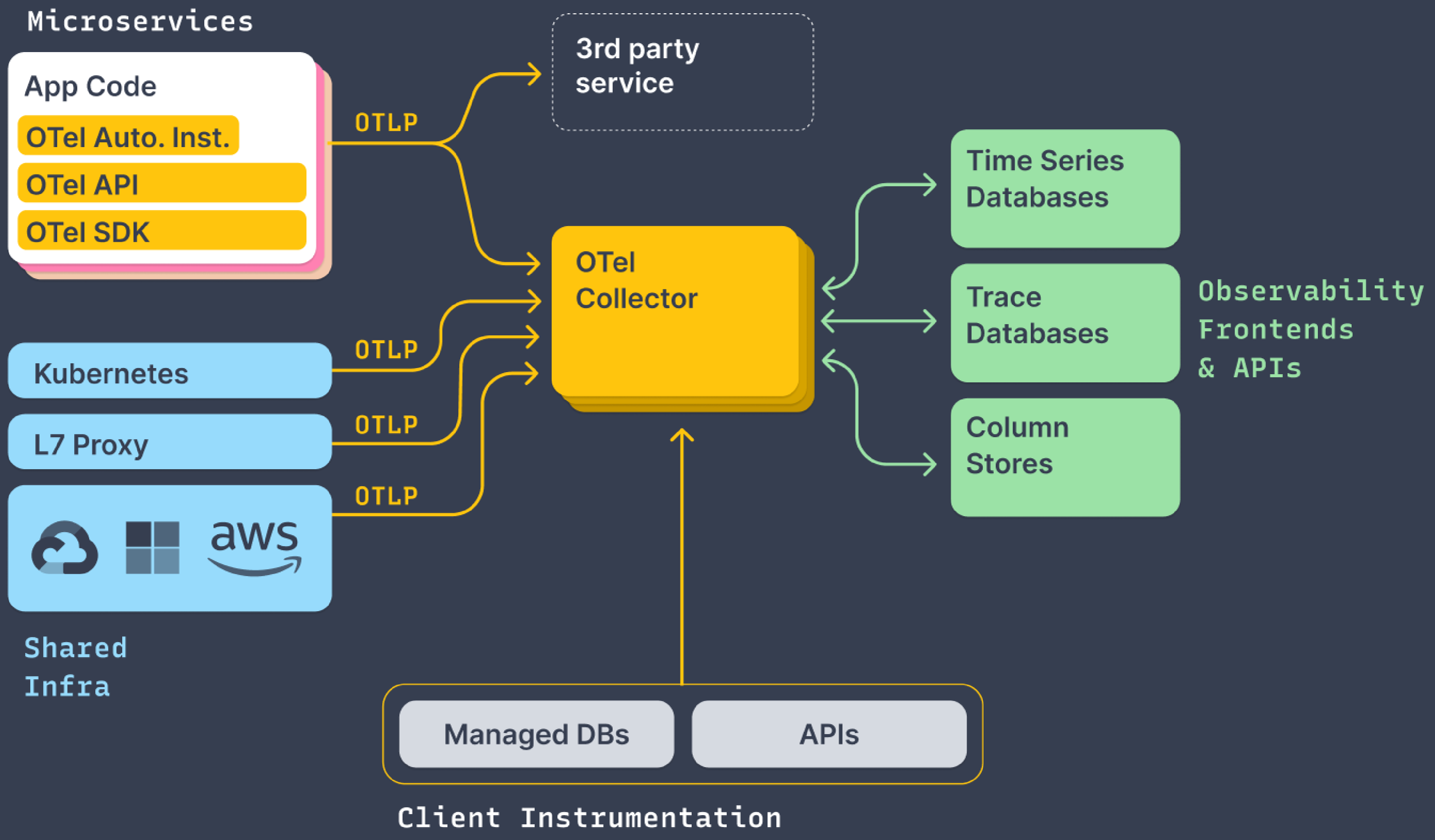
- OpenTelemetry is a collection of tools, APIs, and SDKs. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software’s performance and behaviour.

- *NB: OpenTelemetry ≠ observability back-end*

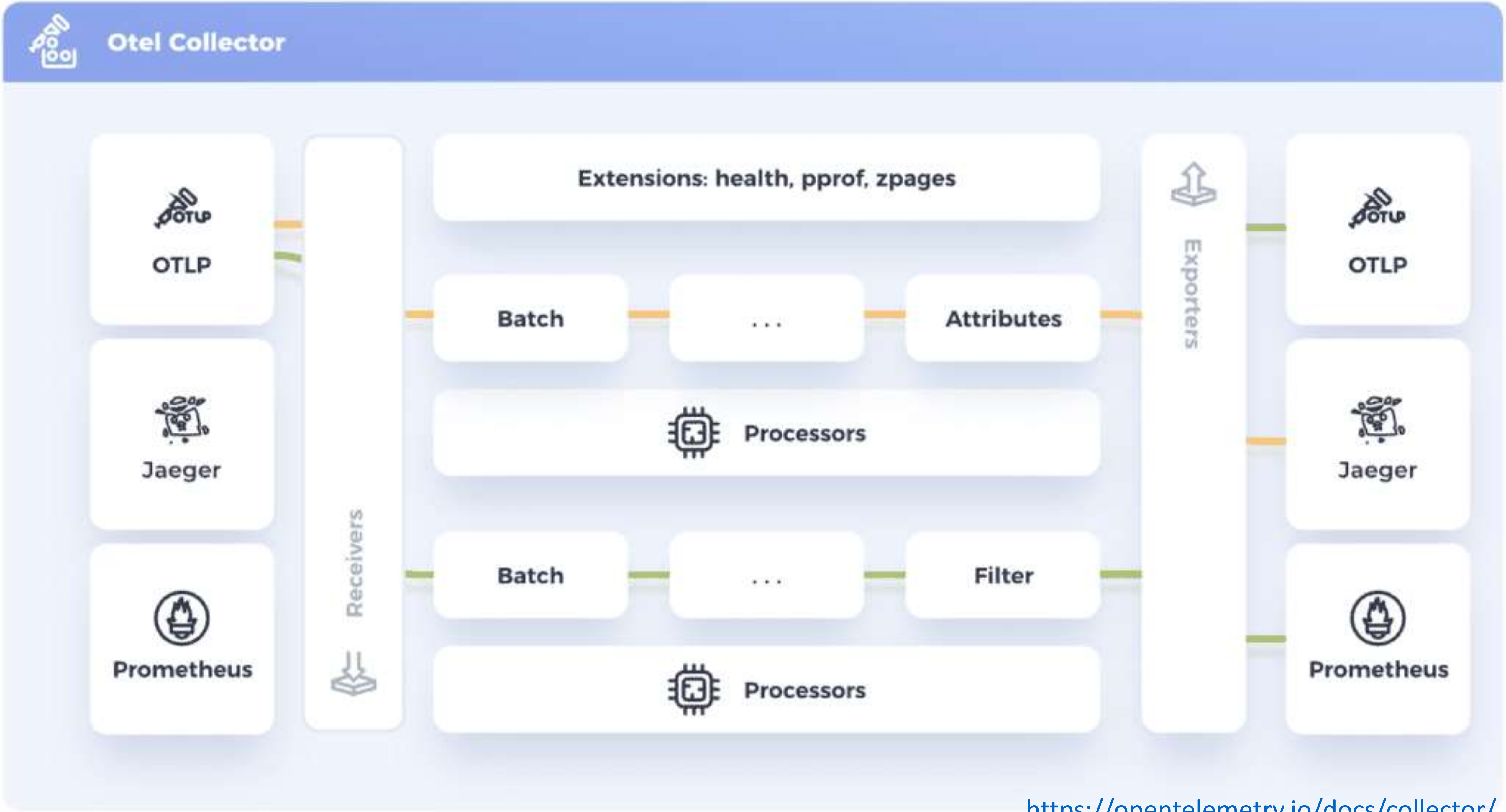
https://opentelemetry.io

Creating One Standard





https://opentelemetry.io/docs/

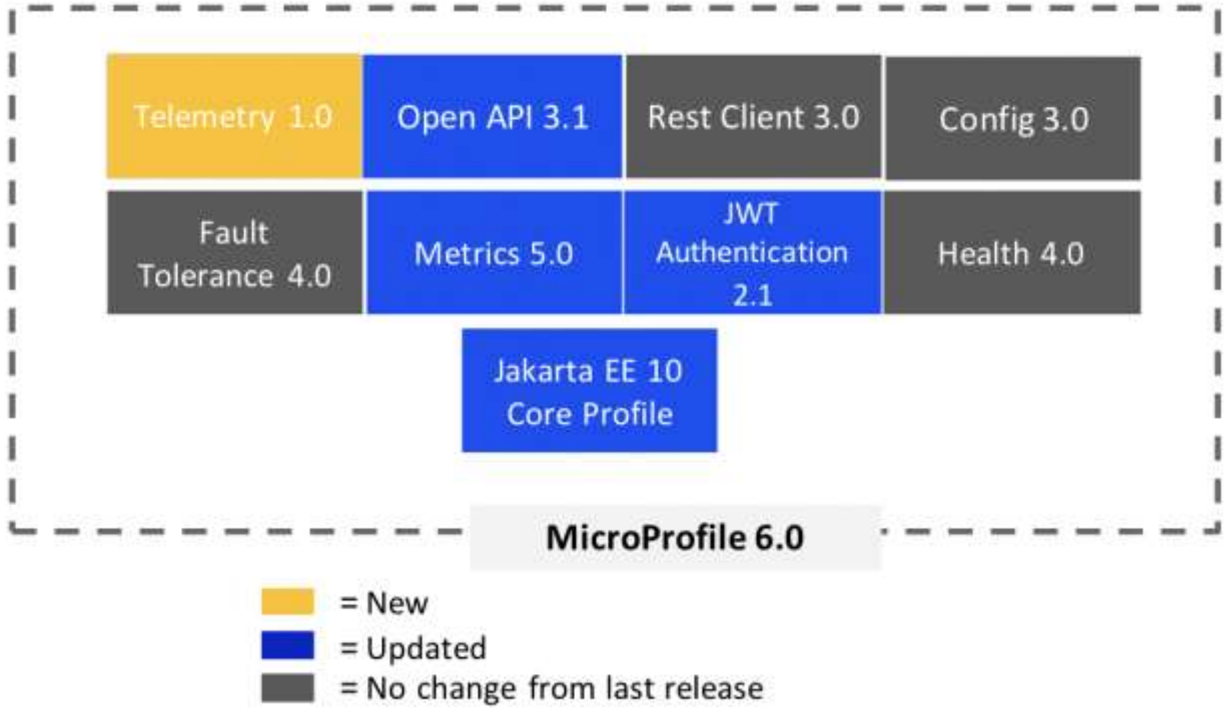


https://opentelemetry.io/docs/collector/

MicroProfile Telemetry 1.0

MicroProfile Telemetry 1.0

- Introduced in MicroProfile 6.0 release
- Adopts OpenTelemetry Tracing
- Set of APIs, SDKs, tooling and integrations
- Designed for the creation and management of telemetry data (traces)

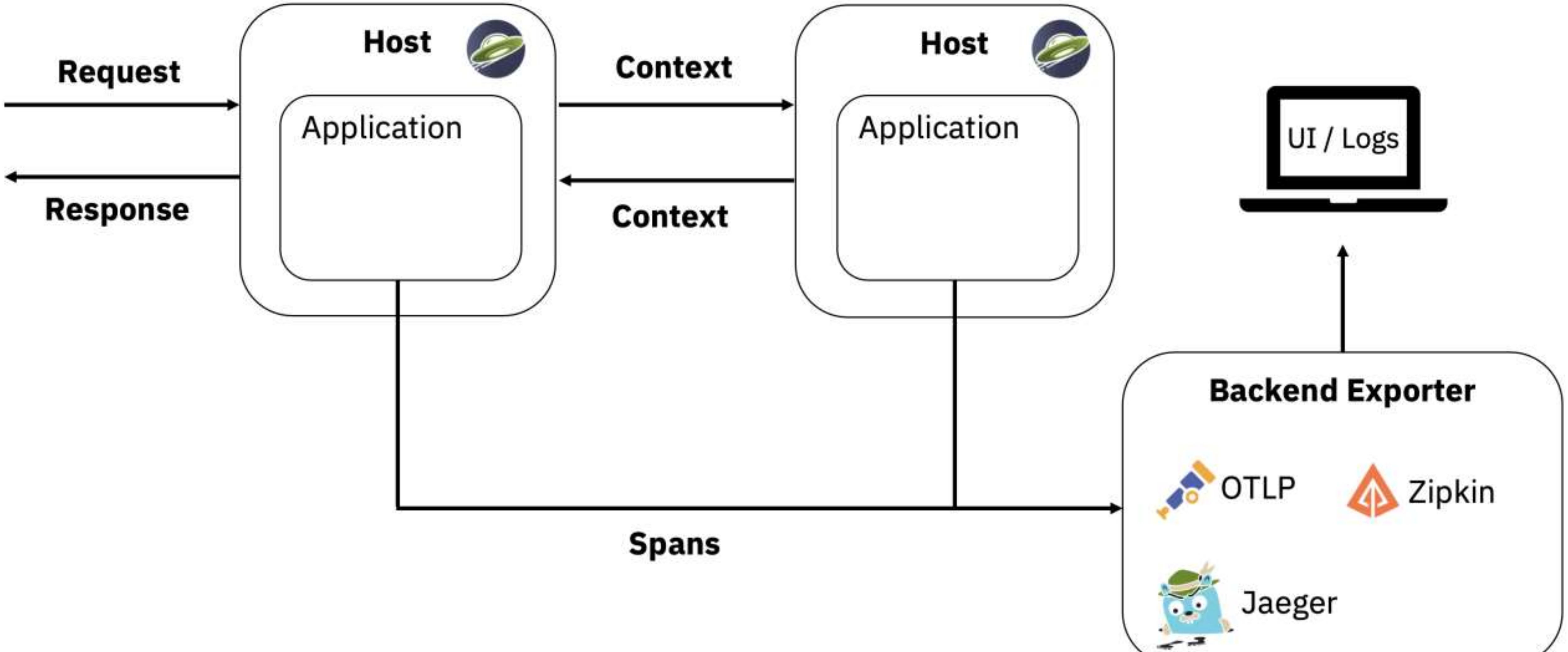


https://github.com/eclipse/microprofile-telemetry

MP Telemetry Instrumentation

- Automatic Instrumentation:
  - Jakarta RESTful Web Services and MicroProfile Rest Client automatically enlisted in distributed tracing
- Manual Instrumentation:
  - Manual instrumentation can be added via annotations `@WithSpan` or via CDI injection `@Inject Tracer` or `@Inject Span` or programmatic lookup `Span.current()`
- Agent Instrumentation:
  - Use OpenTelemetry Java Instrumentation project to gather telemetry data without any code modification

How MP Telemetry works



2 Backend Exporter

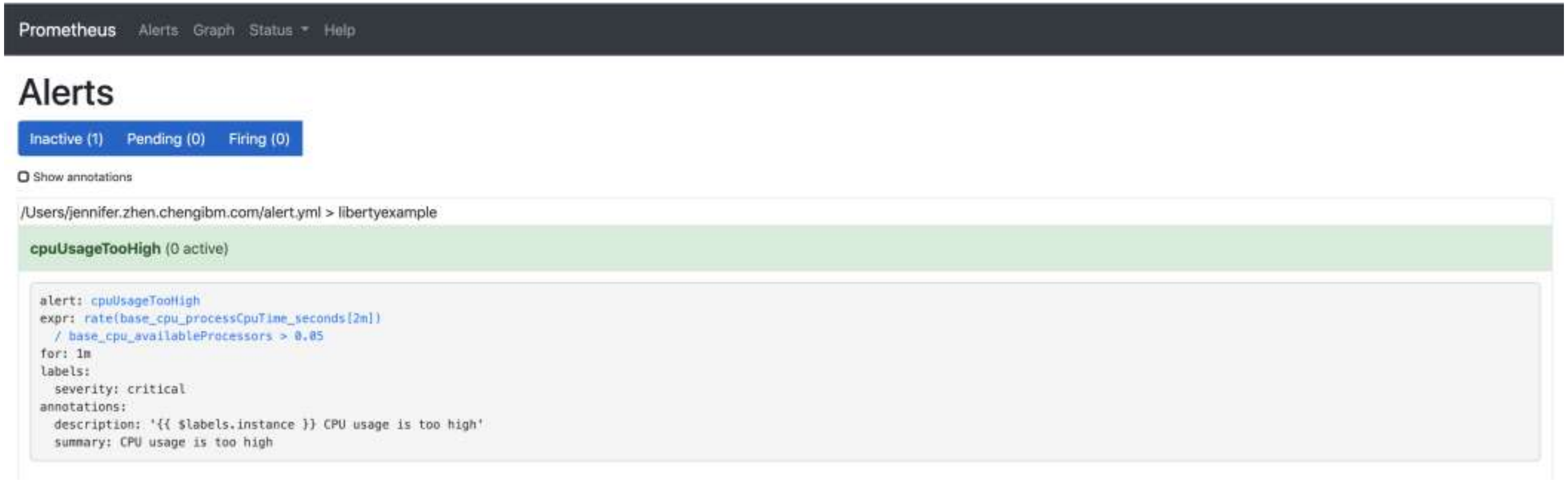
- You can export the data that MicroProfile Telemetry collects to multiple exporters.
- E.g.:
  - Jaeger
  - Zipkin
  - Otel Collector

3

Visualization

- Prometheus
  - Systems monitoring and alerting toolkit
- Grafana
  - An open source analytics and interactivee visualization
- Kibana
  - provides users with a tool for exploring, visualizing, and building dashboards on top of the log data stored in Elasticsearch clusters.

https://blog.sebastian-daschner.com/entries/openliberty-monitoring-prometheus-grafana



Prometheus Alerts Graph Status Help

Alerts

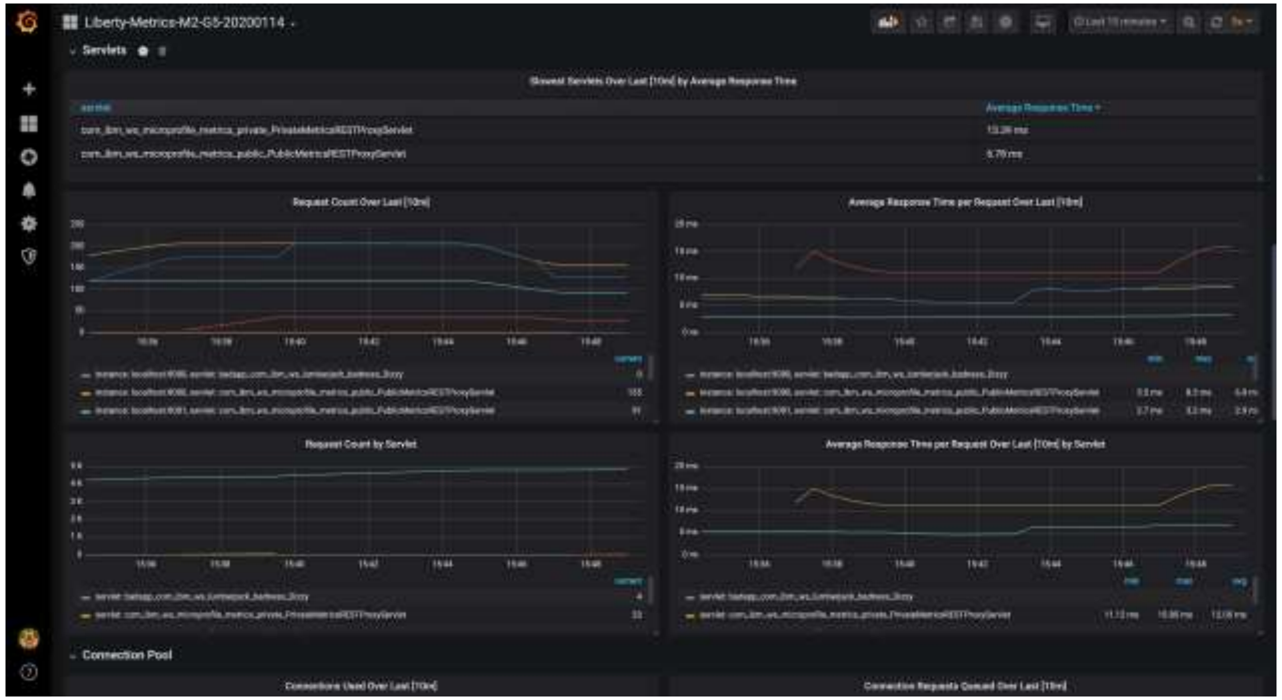
Inactive (1) Pending (0) Firing (0)

Show annotations

/Users/jennifer.zhen.chengibm.com/alert.yml > libertyexample

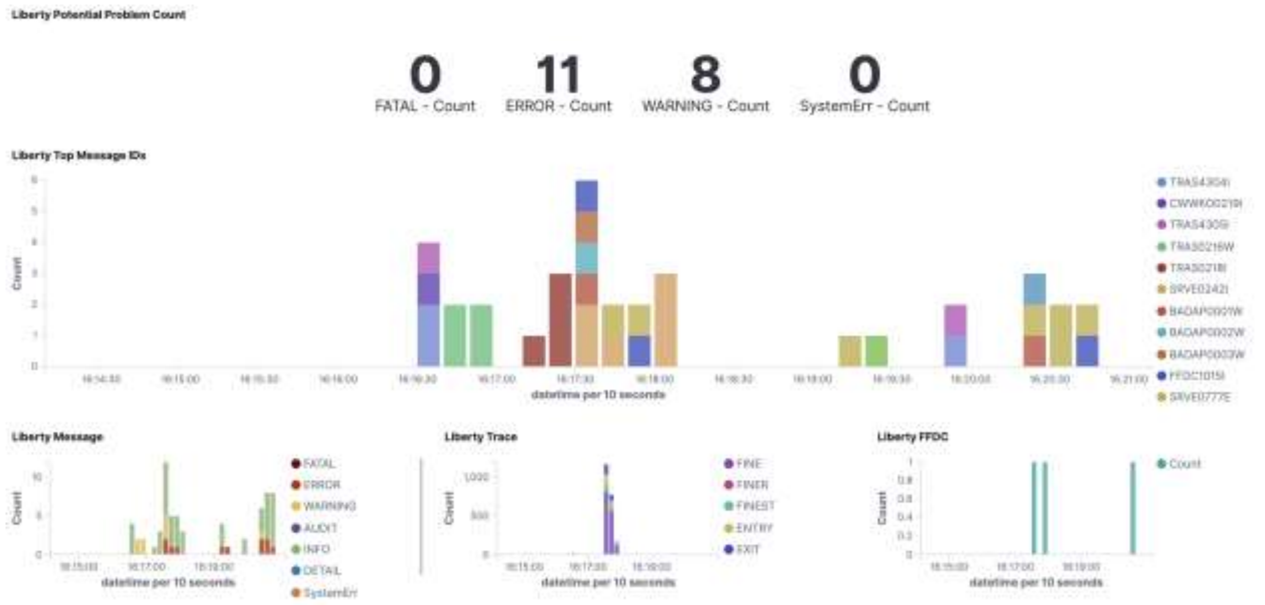
cpuUsageTooHigh (0 active)

```
alert: cpuUsageTooHigh
expr: rate(base_cpu_processCpuTime_seconds[2m])
  / base_cpu_availableProcessors > 0.05
for: 1m
labels:
  severity: critical
annotations:
  description: '{{ $labels.instance }} CPU usage is too high'
  summary: CPU usage is too high
```



Liberty-Metrics-M2-G5-20200114
Servlets
Slowest Servlets Over Last [10m] by Average Response Time
Request Count Over Last [10m]
Average Response Time per Request Over Last [10m]
Request Count by Servlet
Average Response Time per Request Over Last [10m] by Servlet
Connection Pool
Connections Used Over Last [10m]
Connection Requests Queued Over Last [10m]

servlet	Average Response Time
com_ibm_ws_microprofile_metrics_private_PrivateMetricsRESTProxyServlet	13.28 ms
com_ibm_ws_microprofile_metrics_public_PublicMetricsRESTProxyServlet	6.79 ms



Demo Time

Open Liberty

Focus on code

Easy to make fast and iterative changes

Easy to write tests

True-to-production testing (as much as possible)

Ready for containers

Not-in-your-way tools and flexibility

Jump on board and work at lightspeed

Build cloud-native apps and microservices while running only what you need. Open Liberty is the most flexible server runtime available to Java™ developers in this solar system. Here's why.

https://developer.ibm.com/articles/why-cloud-native-java-developers-love-liberty/

Developer productivity

IDEs

Dev Mode

Repositories

The Central Repository

docker hub

Build

Maven

Gradle

APIs

MICROPROFILE
OPTIMIZING ENTERPRISE JAVA

Java EE

JAKARTA EE

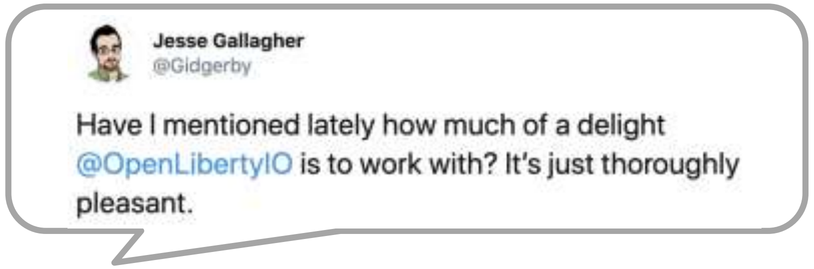
Spring Boot

Testing

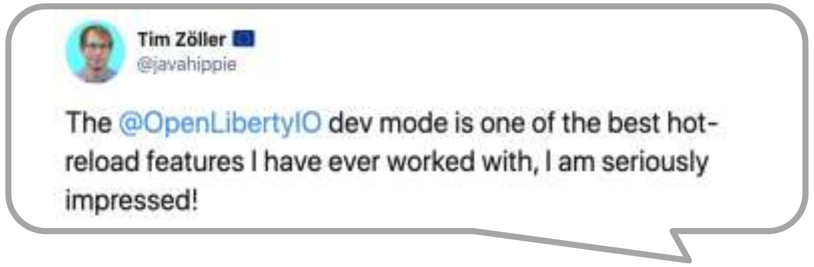
microshed testing

JUnit

arquillian


Jesse Gallagher
@Gidgerby

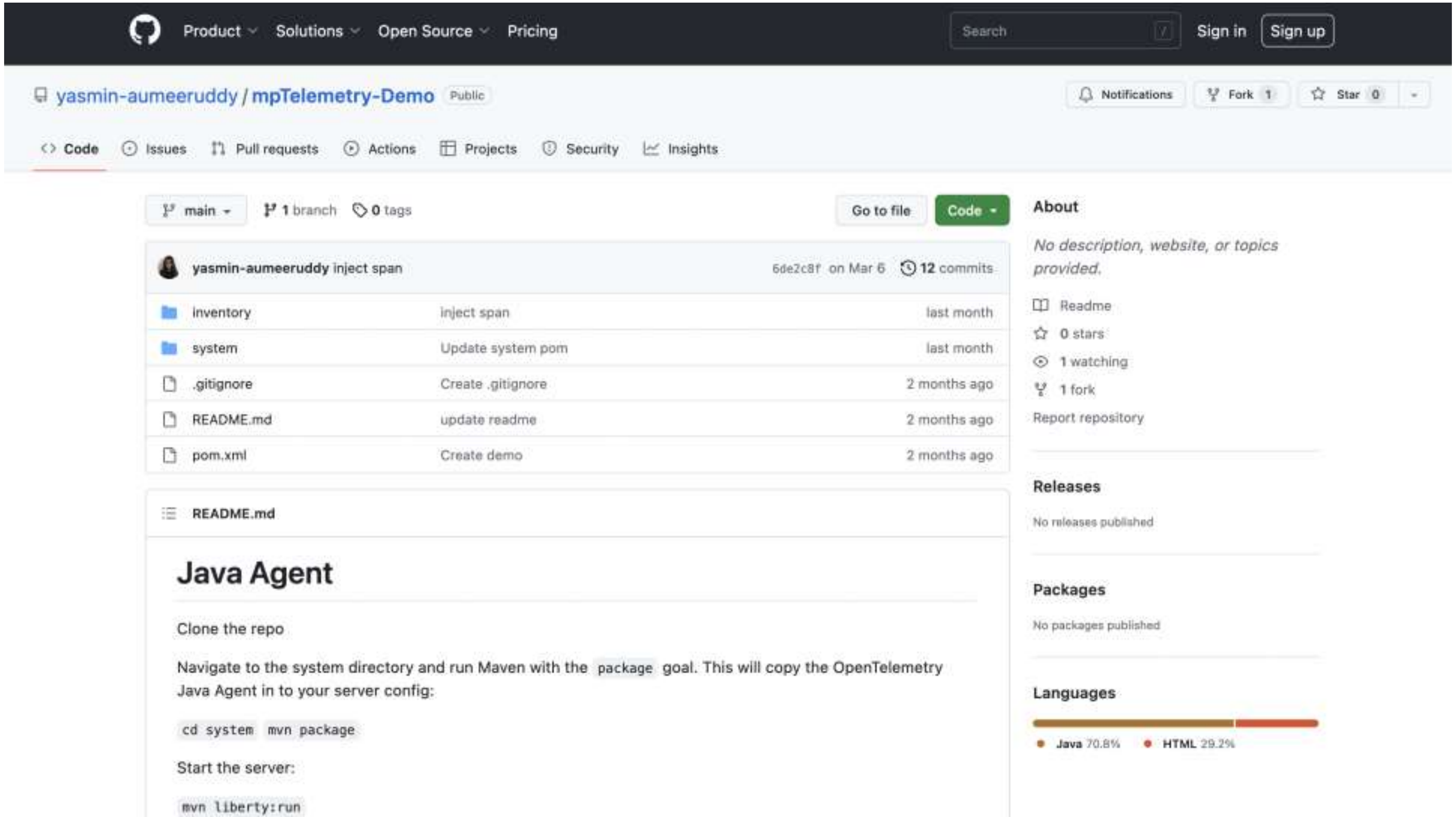
Have I mentioned lately how much of a delight @OpenLibertyIO is to work with? It's just thoroughly pleasant.


Tim Zöller
@javahippie

The @OpenLibertyIO dev mode is one of the best hot-reload features I have ever worked with, I am seriously impressed!

https://github.com/yasmin-aumeeruddy/mpTelemetry-Demo

MP Telemetry Demo



yasmin-aumeeruddy / mpTelemetry-Demo Public

1 branch 0 tags

yasmin-aumeeruddy inject span 6de2c8f on Mar 6 12 commits

inventory	inject span	last month
system	Update system pom	last month
.gitignore	Create .gitignore	2 months ago
README.md	update readme	2 months ago
pom.xml	Create demo	2 months ago

About

No description, website, or topics provided.

Readme
0 stars
1 watching
1 fork
Report repository

Releases

No releases published

Packages

No packages published

Languages

● Java 70.8% ● HTML 29.2%

README.md

Java Agent

Clone the repo

Navigate to the system directory and run Maven with the `package` goal. This will copy the OpenTelemetry Java Agent in to your server config:

```
cd system mvn package
```

Start the server:

```
mvn liberty:run
```

Summary:

- Entering a world of increased complexity

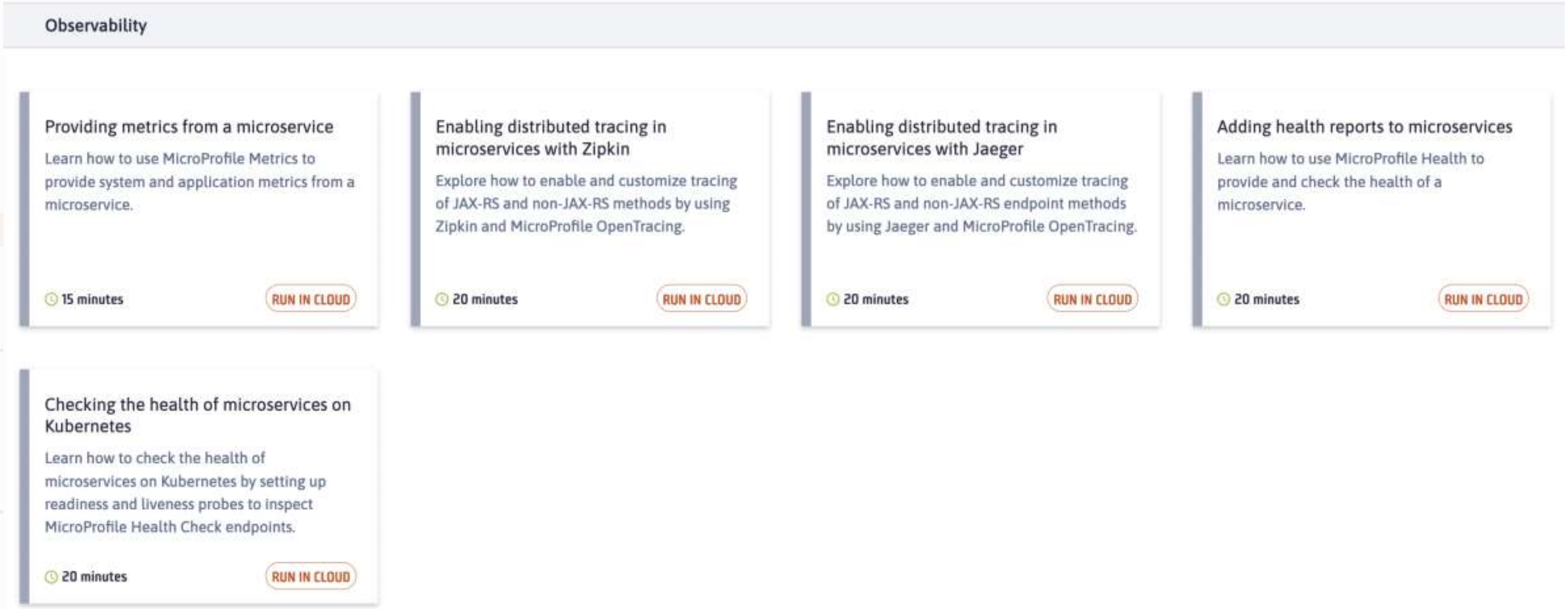
- Effective observability is critical to monitor and understand how our applications are behaving and performing in this complex environment

- Many open source tools available to help us look through the looking glass, including new standards like Open Telemetry
    - OSS Java tools like MicroProfile enable us to make use of this in our own applications

Resources:

- What is observability? - https://www.ibm.com/uk-en/topics/observability
- OpenTelemetry and MicroProfile: Enabling effective observability for your cloud-native Java applications - https://developer.ibm.com/articles/opentelemetry-effective-observability-for-your-cloud-native-java-apps/
- Tracing your microservices made easy with MicroProfile Telemetry 1.0 - https://openliberty.io/blog/2023/03/10/tracing-with-microprofile-telemetry.html

Open Liberty Interactive Guides



Observability

Providing metrics from a microservice

Learn how to use MicroProfile Metrics to provide system and application metrics from a microservice.

15 minutes RUN IN CLOUD

Enabling distributed tracing in microservices with Zipkin

Explore how to enable and customize tracing of JAX-RS and non-JAX-RS methods by using Zipkin and MicroProfile OpenTracing.

20 minutes RUN IN CLOUD

Enabling distributed tracing in microservices with Jaeger

Explore how to enable and customize tracing of JAX-RS and non-JAX-RS endpoint methods by using Jaeger and MicroProfile OpenTracing.

20 minutes RUN IN CLOUD

Adding health reports to microservices

Learn how to use MicroProfile Health to provide and check the health of a microservice.

20 minutes RUN IN CLOUD

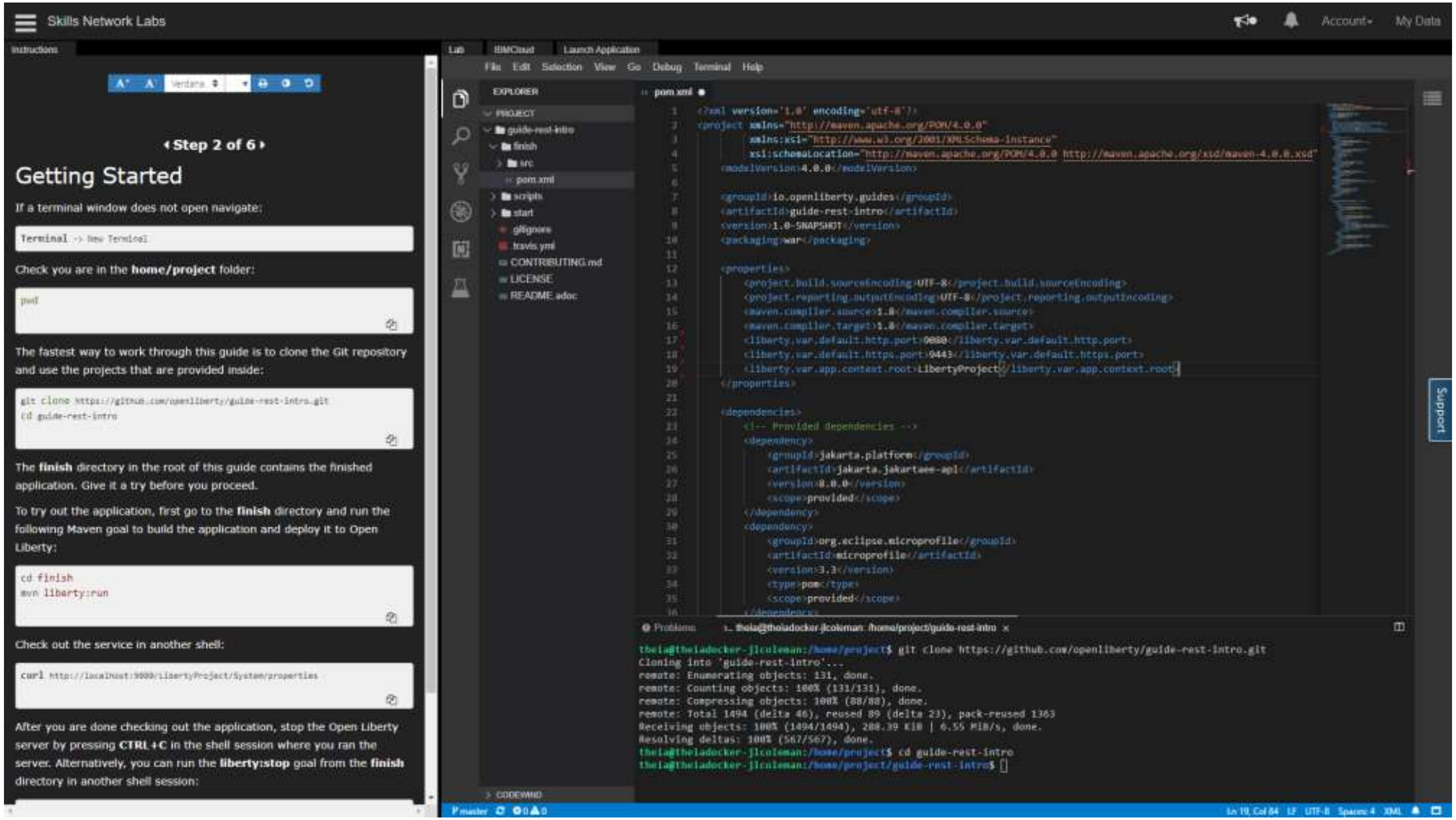
Checking the health of microservices on Kubernetes

Learn how to check the health of microservices on Kubernetes by setting up readiness and liveness probes to inspect MicroProfile Health Check endpoints.

20 minutes RUN IN CLOUD

https://openliberty.io/guides/#observability

Interactive cloud-native labs



Getting Started

If a terminal window does not open navigate:

```
Terminal -> New Terminal
```

Check you are in the **home/project** folder:

```
pwd
```

The fastest way to work through this guide is to clone the Git repository and use the projects that are provided inside:

```
git clone https://github.com/openliberty/guide-rest-intro.git
cd guide-rest-intro
```

The **finish** directory in the root of this guide contains the finished application. Give it a try before you proceed.

To try out the application, first go to the **finish** directory and run the following Maven goal to build the application and deploy it to Open Liberty:

```
cd finish
mvn liberty:run
```

Check out the service in another shell:

```
curl http://localhost:9080/LibertyProject/System/properties
```

After you are done checking out the application, stop the Open Liberty server by pressing **CTRL+C** in the shell session where you ran the server. Alternatively, you can run the **liberty:stop** goal from the **finish** directory in another shell session:

pom.xml

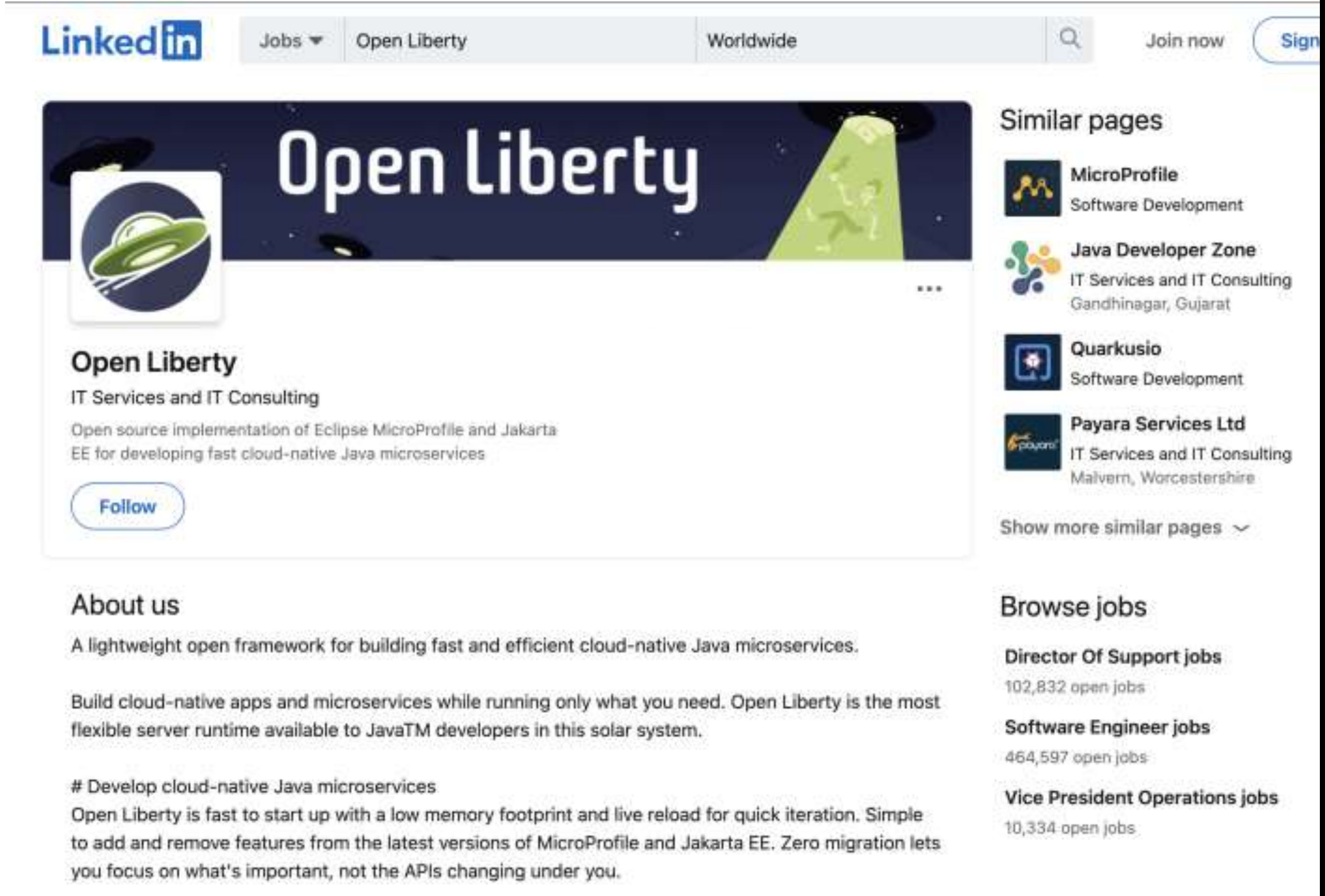
```xml
<?xml version='1.0' encoding='utf-8'?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>io.openliberty.guides</groupId>
    <artifactId>guide-rest-intro</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <liberty.var.default.http.port>9080</liberty.var.default.http.port>
        <liberty.var.default.https.port>9443</liberty.var.default.https.port>
        <liberty.var.app.context.root>LibertyProject</liberty.var.app.context.root>
    </properties>

    <dependencies>
        <!-- Provided dependencies -->
        <dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-api</artifactId>
            <version>8.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.eclipse.microprofile</groupId>
            <artifactId>microprofile</artifactId>
            <version>3.3</version>
            <type>pom</type>
            <scope>provided</scope>
        </dependency>
```

Problems theia@theiadocker-jlcoleman: /home/project/guide-rest-intro

```
theia@theiadocker-jlcoleman:/home/project$ git clone https://github.com/openliberty/guide-rest-intro.git
Cloning into 'guide-rest-intro'...
remote: Enumerating objects: 131, done.
remote: Counting objects: 100% (131/131), done.
remote: Compressing objects: 100% (88/88), done.
remote: Total 1494 (delta 46), reused 89 (delta 23), pack-reused 1363
Receiving objects: 100% (1494/1494), 288.39 KiB | 6.55 MiB/s, done.
Resolving deltas: 100% (567/567), done.
theia@theiadocker-jlcoleman:/home/project$ cd guide-rest-intro
theia@theiadocker-jlcoleman:/home/project/guide-rest-intro$ 
```

Ln 19, Col 84 LF UTF-8 Spaces: 4 XML

Connect with us



Open Liberty

IT Services and IT Consulting

Open source implementation of Eclipse MicroProfile and Jakarta EE for developing fast cloud-native Java microservices

About us

A lightweight open framework for building fast and efficient cloud-native Java microservices.

Build cloud-native apps and microservices while running only what you need. Open Liberty is the most flexible server runtime available to JavaTM developers in this solar system.

Develop cloud-native Java microservices
Open Liberty is fast to start up with a low memory footprint and live reload for quick iteration. Simple to add and remove features from the latest versions of MicroProfile and Jakarta EE. Zero migration lets you focus on what's important, not the APIs changing under you.

Similar pages

MicroProfile
Software Development

Java Developer Zone
IT Services and IT Consulting
Gandhinagar, Gujarat

Quarkusio
Software Development

Payara Services Ltd
IT Services and IT Consulting
Malvern, Worcestershire

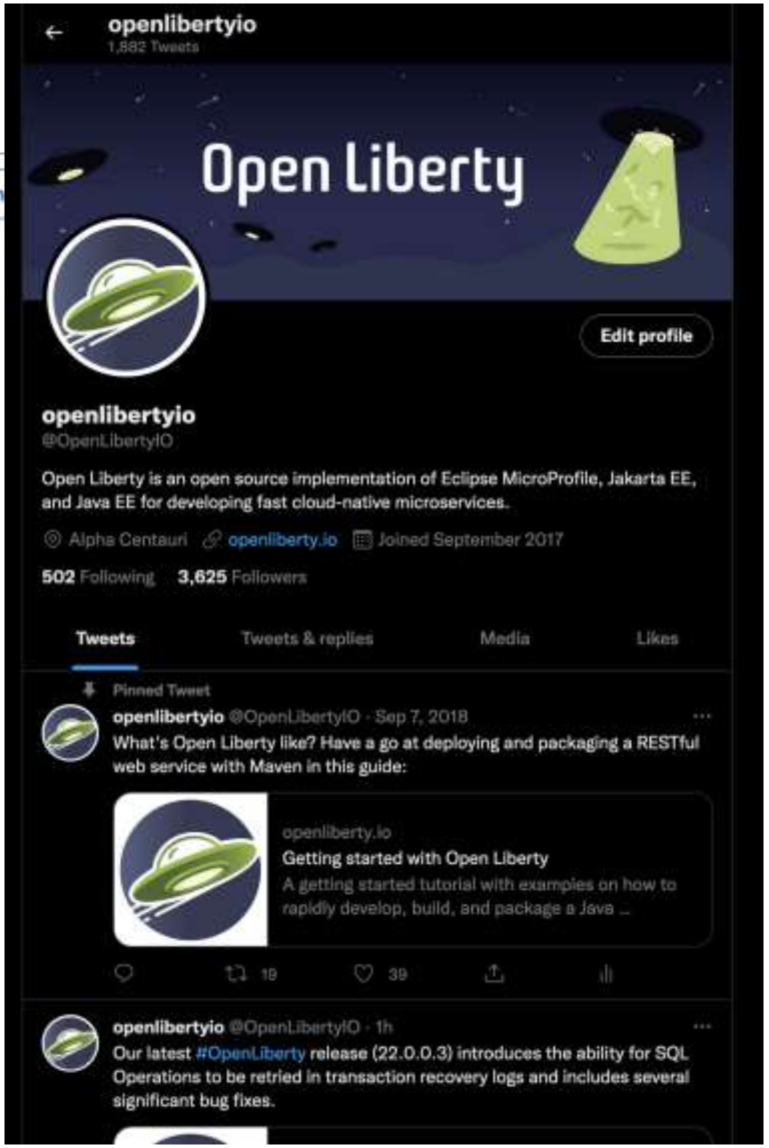
Show more similar pages

Browse jobs

Director Of Support jobs
102,832 open jobs

Software Engineer jobs
464,597 open jobs

Vice President Operations jobs
10,334 open jobs



openlibertyio
@OpenLibertyIO

Open Liberty is an open source implementation of Eclipse MicroProfile, Jakarta EE, and Java EE for developing fast cloud-native microservices.

Alpha Centauri openliberty.io Joined September 2017

502 Following 3,625 Followers

Pinned Tweet
openlibertyio @OpenLibertyIO · Sep 7, 2018
What's Open Liberty like? Have a go at deploying and packaging a RESTful web service with Maven in this guide:

openliberty.io
Getting started with Open Liberty
A getting started tutorial with examples on how to rapidly develop, build, and package a Java ...

19 39

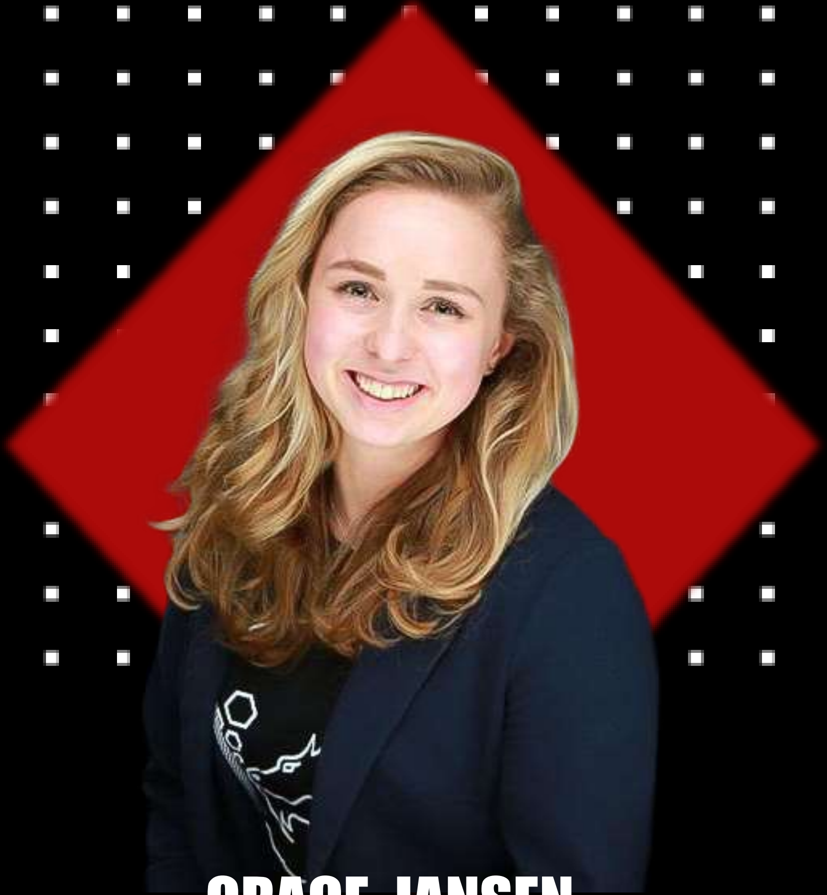
openlibertyio @OpenLibertyIO · 1h
Our latest #OpenLiberty release (22.0.0.3) introduces the ability for SQL Operations to be retried in transaction recovery logs and includes several significant bug fixes.

https://www.linkedin.com/company/openlibertyio/

https://twitter.com/OpenLibertyIO

THANK YOU

JAVA USER GROUP CH



GRACE, JANSEN

@gracejansen27