Observability for Modern JVM Applications

Jonatan Ivanov

2024-02-23

About Me

- Spring Team
  - Micrometer
  - Spring Cloud, Spring Boot
  - Spring Observability Team
- Seattle Java User Group
- develotters.com
- **@jonatan_ivanov**

spring

Gauge the audience

How many people are using:

- Observability in production?
- Spring Boot 3?
- Micrometer?
- Prometheus
- OpenTelemetry?

What is Observability?

What is Observability?

How well we can understand the internals of a system based on its outputs

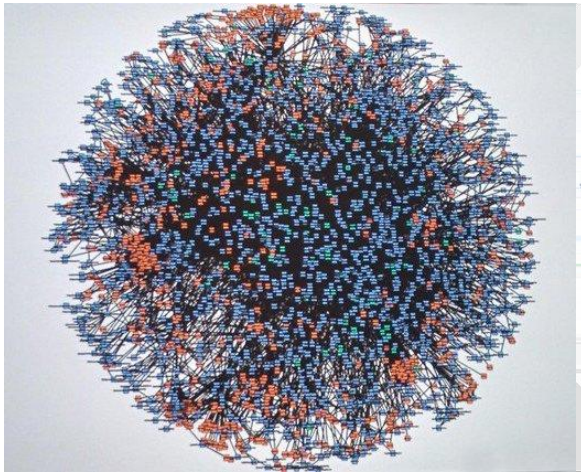
(Providing ***meaningful*** information about what happens inside)
(Data about your app)

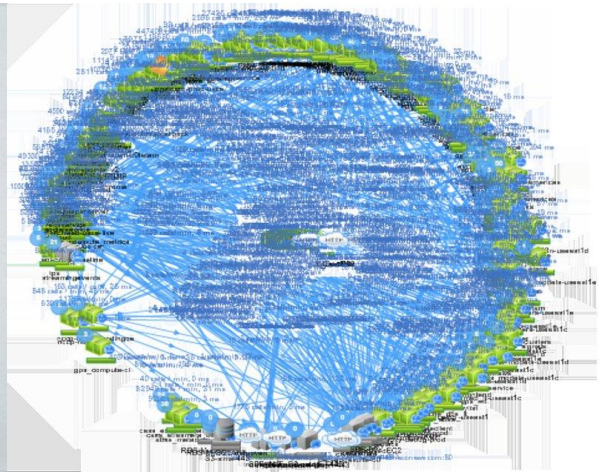
Why do we need Observability?

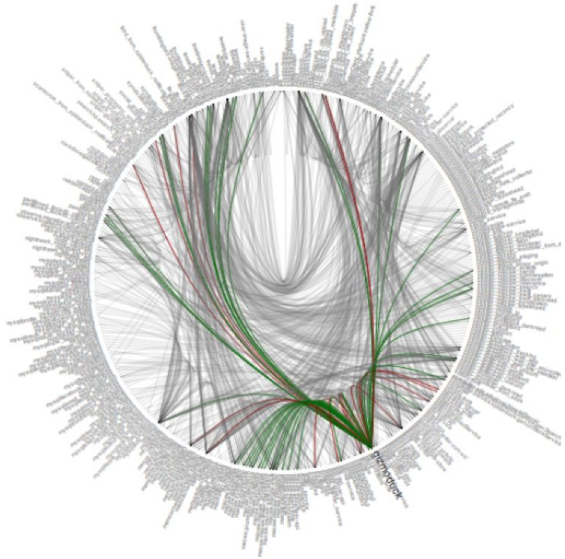
Why do we need Observability?

Today's systems are increasingly complex (cloud)

(Death Star Architecture, Big Ball of Mud)







spring

Why do we need Observability?

Environments can be chaotic

You turn a knob here a little and apps are going down there

We need to deal with unknown unknowns

We can’t know everything

Things can be perceived differently by observers

Everything is broken for the users but seems ok to you

Why do we need Observability? (business perspective)

Reduce lost revenue from production incidents

Lower mean time to recovery (MTTR)

Require less specialized knowledge

Shared method of investigating across system

Quantify user experience

Don't guess, measure!

Logging
Metrics
Distributed Tracing

Logging - Metrics - Distributed Tracing

Logging

What happened (why)?

Emitting events

Metrics

What is the context?

Aggregating data

Distributed Tracing

Why happened?

Recording causal ordering of events

Examples

Latency

Logging

Processing took 140ms

Metrics

P99.999: 140ms

Max: 150 ms

Distributed Tracing

DB was slow

(lot of data was requested)

Error

Logging

Processing failed (stacktrace?)

Metrics

The error rate is 0.001/sec

2 errors in the last 30 minutes

Distributed Tracing

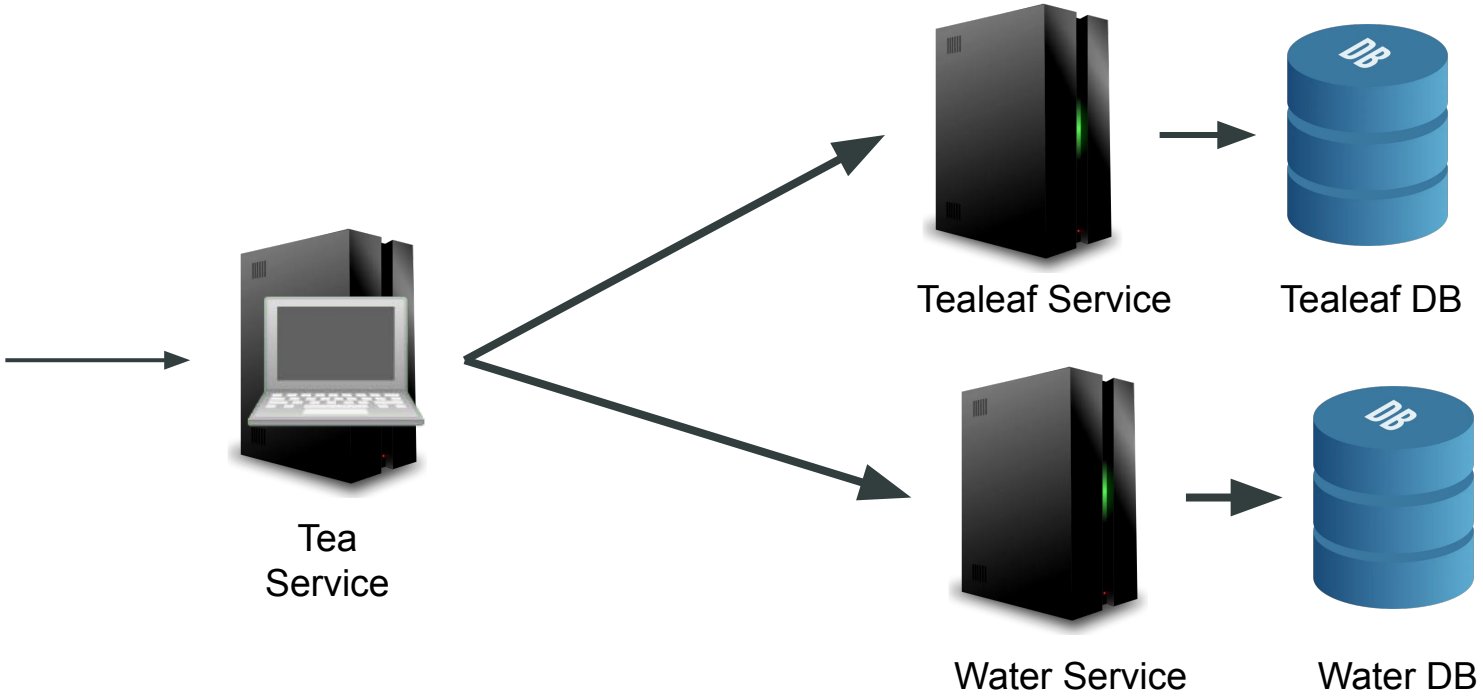
DB call failed

(invalid input)

DEMO

github.com/jonatan-ivanov/teahouse

Architecture



spring

spring-boot-starter-**web**

spring-boot-starter-data-**jpa**

spring-cloud-starter-**openfeign**

spring-boot-starter-**actuator** (micrometer-observation)

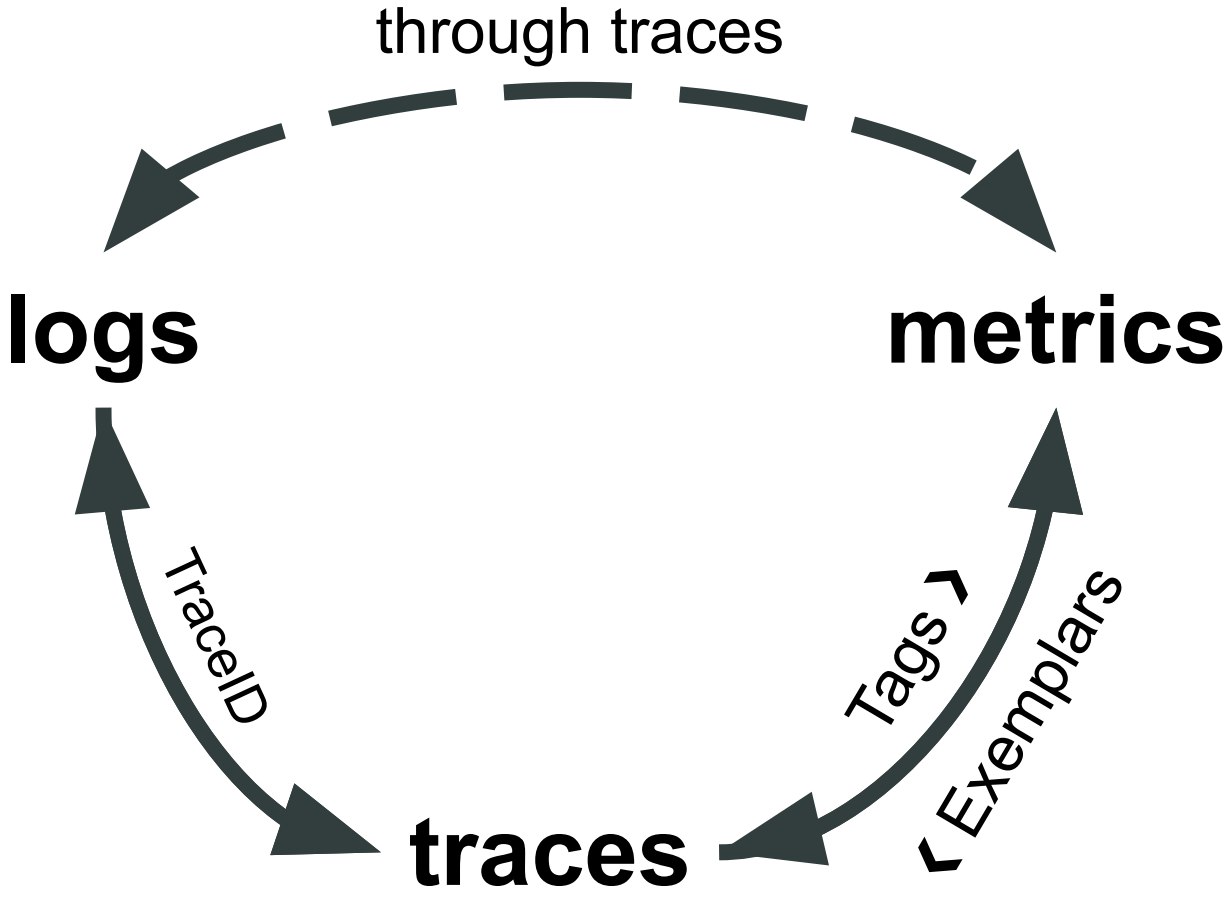
micrometer-registry-**prometheus**

micrometer-tracing-bridge-**brave** + **zipkin-reporter**-brave

net.ttddyy.observation:**datasource-micrometer-spring-boot**

Let’s make some tea!

spring


through traces
logs
metrics
TraceID
Tags
Exemplars
traces

spring

Logging With JVM/Spring

Logging with JVM/Spring: SLF4J + Logback

SLF4J with Logback comes pre-configured

SLF4J (Simple Logging Façade for Java)

Simple API for logging libraries

Logback

Natively implements the SLF4J API

If you want Log4j2 instead of Logback:

```
- spring-boot-starter-logging
+ spring-boot-starter-log4j2
```

Metrics
With JVM/Spring

Metrics with JVM/Spring: Micrometer

Dimensional Metrics library on the JVM

Like SLF4J, but for metrics

API is independent of the configured metrics backend

Supports many backends

Comes with `spring-boot-actuator`

Spring projects are instrumented using Micrometer

Many third-party libraries use Micrometer

Supported metrics backends/formats/protocols

AppOptics

Atlas

Azure Monitor

CloudWatch (AWS)

Datadog

Dynatrace

Elastic

Ganglia

Graphite

Humio

InfluxDB

JMX

KairosDB

New Relic

(/actuator/metrics)

OpenTSDB

OTLP

Prometheus

SignalFx

Stackdriver (GCP)

StatsD

Wavefront (VMware)

Tracing
With JVM/Spring

Distributed Tracing with JVM/Spring

Boot 2.x: `Spring Cloud Sleuth`

Boot 3.x: `Micrometer Tracing`
(Sleuth w/o Spring dependencies)

Provide an abstraction layer on top of tracing libraries

- Brave (OpenZipkin), default
- OpenTelemetry (CNCF), experimental

Instrumentation for Spring Projects, 3rd party libraries, your app

Support for various backends

Observation API

You want to instrument your application...

- Add logs
  (application logs)
- Add metrics
  - Increment Counters
  - Start/Stop Timers
- Add Distributed Tracing
  - Start/Stop Spans
  - Log Correlation
  - Context Propagation

Observation API (since Micrometer 1.10)

```
Observation observation = Observation.start("talk",registry);
try { // TODO: scope
    Thread.sleep(1000);
}
catch (Exception exception) {
    observation.error(exception);
    throw exception;
}
finally { // TODO: attach tags (key-value)
    observation.stop();
}
```

spring

Observation API (since Micrometer 1.10)

```
ObservationRegistry registry = ObservationRegistry.create();

registry.observationConfig()
    .observationHandler(new MeterHandler(...))
    .observationHandler(new TracingHandler(...))
    .observationHandler(new LoggingHandler(...))
    .observationHandler(new AuditEventHandler(...));
```

Observation API (since Micrometer 1.10)

```
Observation.createNotStarted("talk",registry)
    .lowCardinalityKeyValue("event", "ConFoo")
    .highCardinalityKeyValue("uid", userId)
    .observe(this::talk);
```

```
@Observed
```

spring

Recent changes, What's next?

- Improved Exemplars support
- Observability improvements in the Spring portfolio
  - Better Context Propagation
  - Automatic Log Correlation
  - Additional instrumentations (@Scheduled, R2DBC, JMS, etc.)
- New Docs site
- *Investigating Virtual Treads / Project Loom (?)*
- *Investigating Coordinated Restore at Checkpoint (?)*

Thank you!

@jonatan_ivanov

develotters.com

github.com/jonatan-ivanov/teahouse

slack.micrometer.io