Observabilité avec Spring Boot

À quoi sert l’observabilité ?

« Il est 3h du matin, un client vous écrit : “Le service plante”. Que faites-vous ? Où regarder ? Quelles informations avez-vous réellement ? »

Pourquoi l’observabilité ?

L’observabilité est la capacité à comprendre l’état interne d’un système à partir de ses signaux externes, afin d’expliquer ce qui se passe, pourquoi ça se passe et où intervenir, sans avoir prévu le problème à l’avance.

Observabilité ≠ Monitoring

Monitoring	Observabilité
Vérifie ce qu’on connaît	Permet d'expliquer ce qu’on ne connaît pas
Dashboards prévus	Exploration
Alertes fixes	Investigation

Le triptyque

- Logs
- Métriques
- Traces

Xavier Bouclet

- ~19 d’expérience en IT
- CTO / Technical Leader
- Co-organisateur de la conférence /dev/mtl
- Blogger https://xavierbouclet.com
- Youtube @XavierBouclet

Les logs - la base

Événements textuels ou structurés

Exemples : erreurs, warnings, événements métiers

→ Que s’est-il passé ?

🔥 Bonnes pratiques

- Logs structurés (JSON).
- Niveaux de logs cohérents.
- Corrélation ID : traceId, spanId

🚫 Mauvaises pratiques

- Logguer trop → bruit.
- Logguer pas assez → aveugle.
- Logguer des données personnelles.

Les métriques - comprendre l’état du système

Valeurs numériques agrégées dans le temps

Exemples : latence, taux d’erreur, CPU, mémoire

→ Que se passe-t-il ?

Compteur

```
meterRegistry.counter("total_order_placed",
              "method", method,
       "status", String.valueOf(status),
       "route", routeTemplate
       )
.increment();
```

Gauges

```
import java.util.concurrent.atomic.AtomicInteger;
import io.micrometer.core.instrument.MeterRegistry;

public class Gauge {
 private final AtomicInteger queueSize = new
AtomicInteger(0);

 public Gauge(MeterRegistry registry) {
   registry.gauge("queue_size", queueSize);
 }

 public void onEnqueue() { queueSize.incrementAndGet(); }
 public void onDequeue() { queueSize.decrementAndGet(); }
}
```

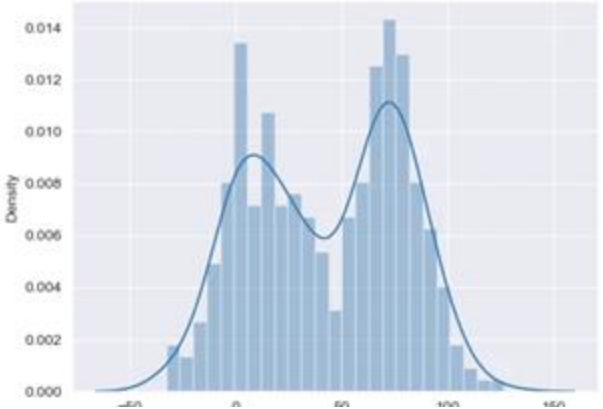
Timer

```
Timer.Sample sample = Timer.start(registry);
try {
       doWork();
} finally {
       sample.stop(registry.timer("job_duration_seconds", "job", "import"));
}
```

Distribution summary

```
DistributionSummary items =
DistributionSummary.builder("search_results_items")
        .description("Number of items returned by search")
        .baseUnit("items")
        .register(registry);


items.record(resultCount);
```



Les traces - comprendre les appels entre services

Traces (distributed tracing)

Suivi d’une requête à travers plusieurs composants

Exemples : appel API → service → DB → cache

→ Où et pourquoi ça ralentit / échoue ?

Pourquoi OpenTelemetry

Collecte unifiée des traces, métriques et logs
Standardisation du contexte distribué (W3C Trace Context)
Instrumentation automatique et manuelle

Indépendant des outils (Prometheus, Grafana, Jaeger, Tempo, Datadog…)

Les piliers de l’observabilité dans Spring Boot

Micrometer

- Façade d’instrumentation → abstrait du backend.
- Différents backends possibles
- Intégré nativement dans Spring Boot.

Actuator

- Actuator expose : health, metrics, info, env…
- Configuration minimale.
- Exposer uniquement ce qui doit l’être (sécurité).

Les métriques fournit par Spring Boot

- http.server.requests
- jdbc.connections
- jvm.*
- process.*

L’instrumentation par Spring Boot

- Créée et propage les traces et spans
- Attache automatiquement aux appels courants (web, jdbc, …)
- Expose le contexte de trace aux logs
- Délègue l’export à des outils externes

Collecte & monitoring

DATADOG

splunk>
a CISCO company

Grafana

Les différentes façons de collecter les informations

Façon	Avantages	Inconvénients
Ajouter un agent au runtime (Datadog, New Relic, Open Telemetry)	Simple à mettre en place (zéro code)	Moins contrôlable Overhead sur le code
Starter Spring Boot Open Telemetry Tiers	Intégré	Plus de configuration Pas supporté directement par Spring
Starter Spring Boot Open Telemetry Tiers	Intégré Supporté par Spring	Plus de configuration

Démonstration

Conclusion

- Observabilité = Logs, métriques et traces
- Spring Boot: Actuator, Micrometer, Tracing intégré
- Outils: Prometheus+Grafana ou Datadog
- Pensez l'observabilité dès le début

Questions ?

Merci

Ressources

- Observability - Wikipedia
- Spring Boot 4 OpenTelemetry: From Zero to Full Observability in Minutes
- GitHub - danvega/ot
- OpenTelemetry with Spring Boot : r/java
- Spring Office Hours: S4E34 - OpenTelemetry with Spring Boot
- OpenTelemetry with Spring Boot
- Tracing :: Spring Boot
- Documentation | OpenTelemetry
- https://micrometer.io/docs/tracing
- https://opentelemetry.io/docs/zero-code/java/agent/performance/
- https://www.graphapp.ai/blog/datadog-vs-new-relic-vs-grafana-a-comprehensive-comparison
- Datadog vs New Relic
- Spring Boot starter | OpenTelemetry
- L'observabilité sans limite avec OpenTelemetry – Olivier Gatimel