|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | **Purpose** | **Role in Observability** | **Stores Data?** | **Visualization?** | **Example Use Case** |
| **Micrometer** | Instrumentation (metrics + tracing) | Collector/Bridge | ❌ No | ❌ No | Add metrics like request.count to your code |
| **Prometheus** | Metrics collection (time-series DB) | Metrics backend | ✅ Yes | ❌ No (basic UI) | Store CPU, memory, HTTP requests per second |
| **Grafana** | Dashboarding / Visualization | Metrics frontend | ❌ No | ✅ Yes | View Prometheus metrics in graphs |
| **Zipkin** | Distributed Tracing (span/trace visualizer) | Traces backend + UI | ✅ Yes | ✅ Yes | Trace user request flow across services |

## **🛠️ What Each Tool Does in a Microservice Stack**

### **✅ Micrometer**

* It's the **instrumentation library**.
* Works as a **facade** over Prometheus, Zipkin, Datadog, etc.
* Adds **metrics + tracing** to your Spring Boot code.
* Example:

registry.counter("orders\_created").increment();

Use when:

* You want **vendor-neutral metrics/tracing**.
* You're using **Spring Boot 2+** (it's built-in).

### **✅ Prometheus**

* A **time-series database** for scraping metrics from applications.
* Scrapes /actuator/prometheus in Spring Boot.
* Good for tracking:
  + CPU, memory
  + Request latency
  + Custom app metrics (via Micrometer)

Use when:

* You need **metrics backend** to store and query data.
* You want to **alert** on metrics.

### **✅ Grafana**

* A **dashboard tool**.
* Visualizes Prometheus metrics in real-time graphs.
* Allows custom queries, alerts, thresholds.

Use when:

* You want to create **real-time dashboards**.
* Need to show metrics to **teams or stakeholders**.

### **✅ Zipkin**

* A **distributed tracing system**.
* Captures **spans** for each operation and shows **request flow** across services.
* Works with **Micrometer Tracing**, Brave, or OpenTelemetry.

Use when:

* You want to debug **performance issues or bottlenecks**.
* Need visibility into **which service is slow** in a chain of microservices.

|  |  |
| --- | --- |
| **Use Case** | **Tool(s) to Use** |
| Collect application metrics (e.g., HTTP count) | Micrometer + Prometheus |
| Visualize metrics in real-time | Prometheus + Grafana |
| Monitor request traces across services | Micrometer Tracing + Zipkin |
| Debug which service is slow in a distributed app | Zipkin |
| Alert if CPU usage > 80% | Prometheus AlertManager + Grafana (optional) |
| Custom business metrics (e.g., orders placed) | Micrometer + Prometheus + Grafana |
| Single-view of infra + app health | Grafana (with Prometheus and other datasources) |

|  |  |
| --- | --- |
| **Tool** | **You Need It When...** |
| **Micrometer** | You want to measure metrics/traces from your app |
| **Prometheus** | You want to **store and query metrics** |
| **Grafana** | You want **dashboards and visualizations** |
| **Zipkin** | You need to **debug request flows and latency** |

A **dashboard in Grafana** is a **collection of visual panels** (like graphs, gauges, tables) that display metrics from data sources like **Prometheus**. It helps you **monitor your system, application performance, infrastructure, or custom metrics in real time.**

## **✅ What is a Dashboard in Grafana?**

* A **dashboard** consists of **one or more panels**.
* Each **panel** shows a specific metric (e.g., HTTP request rate, CPU usage).
* Dashboards can be customized with **filters, time ranges, alerts, and visualizations**.
* You can **share** dashboards with teams, export/import them, or **trigger alerts** based on thresholds.

|  |  |
| --- | --- |
| **Use Case** | **Panel Examples** |
| Web App Health | HTTP 2xx/5xx rate, response latency |
| Infrastructure Monitoring | CPU, RAM, disk usage |
| Business Metrics | Orders placed, logins per hour |
| Cache Monitoring | Redis hit rate, key eviction count |

📦 Sample Prometheus Queries for Grafana

|  |  |
| --- | --- |
| **Metric Name** | **Purpose** |
| http\_server\_requests\_seconds\_count | Number of HTTP requests |
| jvm\_memory\_used\_bytes | JVM heap usage |
| my\_custom\_counter | Custom metric via Micrometer |
| system\_cpu\_usage | System CPU usage |
| process\_uptime\_seconds | App uptime |

Your App

│

├──> Micrometer

│ ├─> Prometheus (metrics)

│ └─> Zipkin (traces)

│

Prometheus ──> stores metrics

│

Grafana ──> queries Prometheus and visualizes

│

Zipkin ──> stores & visualizes traces