Prometheus

- **Purpose** → A **metrics collection and storage** tool.
- Think of it like a **time-series database** that pulls numeric data from your apps/services at regular intervals.
- Examples of metrics it stores:
  - CPU usage
  - Memory usage
  - HTTP request count
  - Average response time
  - Kafka topic lag
- It **pulls** metrics from endpoints your app exposes (e.g., `/actuator/prometheus` in Spring Boot).
- It stores them in a **time-series format** (value + timestamp).

Grafana

- **Purpose** → A **visualization and dashboarding** tool.
- Think of it like a **real-time chart builder** that reads from Prometheus (or other data sources).
- It doesn't store data itself — instead, it queries Prometheus (or MySQL, Elasticsearch, Loki, etc.).
- You use it to:
  - Build real-time dashboards
  - Set up alerts (e.g., send Slack/email if CPU > 90%)
  - Monitor service health visually

💡 Analogy:

- **Prometheus** = the warehouse storing the measurements.
- **Grafana** = the control room wall with big screens showing graphs.

Add these dependencies in `pom.xml`:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

Enable the Prometheus endpoint in `application.properties`:

```properties
management.endpoints.web.exposure.include=prometheus
management.endpoint.prometheus.enabled=true
```

```
http://localhost:8080/actuator/prometheus
```

Updated `prometheus.yml`

```yaml
scrape_configs:
  # Scrape Prometheus itself
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090"]
        labels:
          app: "prometheus"

  # Scrape Spring Boot Quiz App
  - job_name: "quiz-app"
    metrics_path: "/actuator/prometheus"
    static_configs:
      - targets: ["localhost:8085"]
        labels:
          app: "quiz"
```

```
prometheus.exe --config.file=prometheus.yml
```

Open: http://localhost:9090

3 Install Grafana (Local)

1. Download: https://grafana.com/grafana/download
2. Extract/install.
3. Run Grafana:

```bash
./bin/grafana-server
```

4. Open: http://localhost:3000

   Login → `admin / admin` (change password on first login).
5. **Add Prometheus Data Source**:
   - Go to **Connections → Data Sources → Add data source**.
   - Choose **Prometheus**.
   - URL: `http://localhost:9090`
   - Save & Test.

4️⃣ View Metrics in Grafana

- Go to **Dashboards → Import**.
- Use Grafana's built-in JVM dashboard ID: `4701`.
- Now you'll see CPU, memory, requests, etc. in real time.

For Spring Boot / JVM apps

- **4701** → JVM (Micrometer) Statistics
- **11378** → Spring Boot Statistics (Micrometer + Actuator)
- **12900** → Spring Boot 2.x Statistics (Micrometer)
- **11985** → JVM Dashboard by Prometheus Community

For general system metrics

- **1860** → Node Exporter Full (Linux server metrics: CPU, RAM, Disk, Network)
- **3662** → Docker and System Monitoring
- **8919** → Windows Node Exporter (Windows system metrics)

For HTTP & Web Apps

- **7362** → Web Application Metrics
- **11074** → NGINX Metrics
- **10427** → API Request Monitoring

1️⃣ JVM Statistics

The **Java Virtual Machine (JVM)** is the "engine" that runs your Java/Spring Boot app.

JVM statistics usually include:

- **CPU usage** → How much processor time your app is taking.
- **Memory usage** → How much RAM your app is using.
- **Garbage Collection (GC)** → How often old objects are removed from memory.
- **Thread counts** → How many threads (tasks) are running in the JVM.
- **Classes loaded/unloaded** → How many Java classes are currently in memory.

2️⃣ Heap Usage

Think of the **heap** as a big "box" where your Java program stores all objects (data, variables, lists, etc.) while it's running.

- **Used Heap** → Memory currently holding active objects.
- **Committed Heap** → Memory reserved by the JVM for your app.
- **Max Heap** → The maximum memory the JVM can use (set by `-Xmx`).
- **Heap Usage %** = `(Used Heap / Max Heap) * 100`