✅ Prometheus + Grafana Monitoring for Spring Boot (Essential DevOps Setup)

📌 What You'll Learn

- Why Prometheus and Grafana?
- How Spring Boot exposes metrics
- How Prometheus collects them
- How Grafana visualizes them
- Setup steps & real project config

🧠 Why Do We Need Monitoring?

DevOps is not complete without observability.

- 🔍 Track performance & health
- ⚠️ Get alerts when things go wrong
- 📊 View metrics like CPU, memory, HTTP requests, DB connections, etc.

🧰 Tools Overview

Tool	Role
Spring Boot Actuator	Exposes metrics (like /actuator/metrics)
Micrometer	Standard library to export metrics
Prometheus	Scrapes metrics & stores them
Grafana	Dashboards for visualization

⚙️ Step-by-Step Integration

✅ 1. Add Spring Boot & Micrometer Dependencies

xml

CopyEdit

```
<!-- pom.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>
```

✅ 2. Configure application.yml or application.properties

yaml

CopyEdit

```
management:
 endpoints:
  web:
   exposure:
    include: health, info, metrics, prometheus
 metrics:
  export:
   prometheus:
    enabled: true
 endpoint:
  prometheus:
   enabled: true
```

🔗 This exposes metrics at: http://localhost:8080/actuator/prometheus

---

✅ 3. Install & Run Prometheus

prometheus.yml config:

yaml

CopyEdit

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'springboot-app'
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['host.docker.internal:8080']  # Or use service name in Docker
```

Save as prometheus.yml

Start Prometheus:

bash

CopyEdit

```
docker run -d --name=prometheus \
  -p 9090:9090 \
  -v $(pwd)/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
```

✅ 4. Install & Run Grafana

bash

CopyEdit

```
docker run -d --name=grafana -p 3000:3000 grafana/grafana
```

- Login → http://localhost:3000
- Username: admin, Password: admin

✅ 5. Connect Prometheus to Grafana

1. **In Grafana** → Settings → Data Sources → Add Prometheus
2. URL: http://host.docker.internal:9090 or your Prometheus IP
3. Save & Test ✅

✅ 6. Import Dashboards

- Grafana → Dashboards → Import
- Use ID like: 4701 (Micrometer dashboard) or 12559
- You'll see metrics like:
  - JVM memory
  - GC activity
  - HTTP request counts
  - Thread pools
  - DB connection pool

✅ Real-Time Monitoring Result

- 👀 View memory usage, HTTP requests, DB pool stats
- 📈 Live charts auto-refresh every few seconds
- 🚨 Alerts can be set when thresholds are crossed

📦 Real-World Folder Structure

pgsql

CopyEdit

```
spring-boot-app/
 ├── src/
 ├── pom.xml
 ├── application.yml
 ├── prometheus.yml
```

```
├── docker-compose.yml  (optional: one command to start everything)
```

🎯 Benefits

Benefit	Description
Real-time visibility	Monitor health and performance instantly
Debug faster	Identify memory leaks, high latency, etc.
Alerting	Be notified before customers complain
Kubernetes friendly	Easily integrates with container clusters

---

⚠️ Drawbacks / Considerations

- Slight memory overhead for exposing metrics
- Prometheus and Grafana setup can be complex if unmanaged
- You must **secure endpoints** (/actuator/**) in production

✅ Final Tips

- Use @Timed, @Counted, etc. annotations for custom metrics.
- Use **Grafana dashboards templates**—no need to build from scratch.
- Make sure /actuator/prometheus is reachable from Prometheus.