🔍 What is Prometheus? Prometheus is an open-source monitoring system and time-series database.

🧩 Key Features: Collects metrics from applications (e.g., CPU, memory, request counts)

Uses pull-based scraping via HTTP (usually from /metrics endpoints)

Stores data in a time-series format (timestamped numeric data)

Has a query language called PromQL

http\_server\_requests\_seconds\_count{uri="/hello"} 3 jvm\_memory\_used\_bytes{area="heap"} 12345678

Features Prometheus's main features are:

a multi-dimensional data model with time series data identified by metric name and key/value pairs PromQL, a flexible query language to leverage this dimensionality no reliance on distributed storage; single server nodes are autonomous time series collection happens via a pull model over HTTP pushing time series is supported via an intermediary gateway targets are discovered via service discovery or static configuration multiple modes of graphing and dashboarding support

What are metrics? Metrics are numerical measurements in layperson terms. The term time series refers to the recording of changes over time. What users want to measure differs from application to application. For a web server, it could be request times; for a database, it could be the number of active connections or active queries, and so on.

Metrics play an important role in understanding why your application is working in a certain way. Let's assume you are running a web application and discover that it is slow. To learn what is happening with your application, you will need some information. For example, when the number of requests is high, the application may become slow. If you have the request count metric, you can determine the cause and increase the number of servers to handle the load.

📊 What is Grafana? Grafana is an open-source analytics and visualization tool.

🎯 Purpose: Connects to various data sources (like Prometheus, MySQL, InfluxDB)

Lets you visualize metrics with beautiful dashboards, graphs, and alerts

Provides interactive querying, filtering, and alerting

🖼️ Typical Grafana Dashboard: CPU usage over time

Request rate graphs

JVM memory usage charts

Custom alerts (e.g., notify if error rate > 5%)

🔗 How They Work Together Component Role Prometheus Collects + stores metrics data Grafana Visualizes data from Prometheus

You add Prometheus as a data source in Grafana, and then create dashboards to display and monitor metrics.

## **🔑 Prometheus Metric Types**

### **1. Counter**

* **Definition**: A value that **only increases** (or resets to 0).
* **Use cases**: Number of HTTP requests, errors, processed jobs.
* **Micrometer**:

registry.counter("http\_requests\_total").increment();

**Gauge**

* **Definition**: A value that can **increase or decrease**.
* **Use cases**: Current memory usage, queue size, active threads.
* **Micrometer**:

AtomicInteger queueSize = new AtomicInteger(0);

registry.gauge("queue\_size", queueSize);

**3. Histogram**

* **Definition**: Measures the **distribution of events over buckets** (e.g., request durations).
* **Use cases**: Request latency buckets (e.g., <100ms, <500ms, ...).
* **Micrometer**: Used via Timer.

### **4. Summary**

* **Definition**: Similar to histogram, but calculates **quantiles** (e.g., 95th percentile latency).
* Less commonly used than histograms due to higher resource cost.

## **📊 Grafana Concepts**

* **Data Source**: The system Grafana pulls data from (e.g., Prometheus).
* **Panel**: A single visualization (graph, gauge, stat, etc.).
* **Dashboard**: A collection of panels.
* **Templating**: Allows variables (e.g., to filter by service).
* **Alerting**: Create thresholds and get alerts (email, Slack, etc.).

## **🎤 Common Interview Questions**

### **💬 General**

1. **What is Prometheus?**
   1. A time-series database used for monitoring and alerting. It scrapes metrics over HTTP.
2. **What is Grafana?**
   1. A visualization and analytics platform that reads from data sources like Prometheus.
3. **Difference between Prometheus and Grafana?**
   1. Prometheus collects/stores data, Grafana visualizes it.

### **💬 Technical**

1. **What is the difference between a gauge and a counter?**
   1. A counter only increases. A gauge can go up and down.
2. **How does Prometheus scrape data?**
   1. Using HTTP pulls to targets defined in prometheus.yml.
3. **What is PromQL?**
   1. Prometheus Query Language used to filter and aggregate metrics.
4. **How do you create custom metrics in Spring Boot for Prometheus?**
   1. Use Micrometer's MeterRegistry with types like counter, gauge, timer.
5. **How does alerting work in Prometheus and Grafana?**
   1. Prometheus: Alert rules + Alertmanager.
   2. Grafana: UI-based alerts that can notify via Slack, email, etc.

## **🛠️ Sample PromQL Queries**

|  |  |
| --- | --- |
| **Use Case** | **Query** |
| Total HTTP requests | http\_server\_requests\_seconds\_count |
| Rate of requests | rate(http\_server\_requests\_seconds\_count[1m]) |
| JVM memory used | jvm\_memory\_used\_bytes |

✅ Option 1: Run Prometheus Locally (No Docker)

🔽 Step 1: Download Prometheus

Go to: https://prometheus.io/download

Choose your OS and download the appropriate binary.

🧰 Step 2: Extract and Configure

Unzip the download:

tar xvf prometheus-\*.tar.gz

cd prometheus-\*

scrape\_configs:

- job\_name: 'spring-boot-app'

static\_configs:

- targets: ['localhost:8080']

./prometheus --config.file=prometheus.yml

brew install prometheus

cd /opt/homebrew/etc/prometheus

global:

scrape\_interval: 5s

scrape\_configs:

- job\_name: 'spring-boot-app'

metrics\_path: '/actuator/prometheus'

static\_configs:

- targets: ['localhost:9009'] # Replace 9009 with your actual Spring Boot port

prometheus --config.file=/opt/homebrew/etc/prometheus/prometheus.yml

./prometheus --config.file=prometheus.yml

✅ Option 2: Run Grafana Locally (No Docker)

🔽 Step 1: Download Grafana

Go to: https://grafana.com/grafana/download

Pick your OS and follow the install instructions.

💻 Step 2: Start Grafana

For example on macOS:

brew install grafana

brew services start grafana

./bin/grafana-server

Visit: http://localhost:3000

Login: admin / admin

🔗 Step 3: Connect Prometheus to Grafana

In Grafana → Configuration → Data Sources → Add Prometheus

URL: http://localhost:9090

Save & Test