Observability

Selected topics for Spring Boot developers
v2025-05-20

SQA & observability

A modern, robust SQA strategy must go beyond "testing in isolation/staging"

It must embed observability throughout the development lifecycle.

Sample **observability** concerns/activities:

What is happening right now?
Capture and track real-time **metrics** (e.g. error rates, request latency percentiles, throughput) and key **events**

Where in the system is it happening? For how long?
Pinpoint the affected service, host, container or code module. Correlate metric and log time-series.

What is the user- or business-impact?
Quantify affected users, transactions or business metrics (SLIs/SLOs) to prioritise response.

Observability

Observability is the ability to understand the internal state of a running system from the outside.

It consists of the three pillars: logging, metrics and traces.

→ every failure mode or performance degradation should leave a detectable footprint.

Logs are time-stamped, discrete records of events that occurred within a component (errors, warnings, informational messages or custom application events). Indispensable for forensic analysis: they provide contextual detail about what the system was doing at a precise moment.

Metrics are measurements aggregated over time, such as CPU utilisation, request rates, error counts or latencies. Thresholds and SLOs (Service Level Objectives) are defined against metric values to trigger notifications or scaling actions.

Traces (especially distributed traces) represent flows by recording spans (individual units of work) as they propagate through system components or microservices . Shows the causal relationship and timing between spans: pinpoint where in the call graph latency accumulates or where failures were injected.

→ see also: Adrian Cole - Observability 3 ways: Logging, Metrics & Tracing

Spring Boot instrumentation

SB Actuator

Actuator provides a set of endpoints that expose key metrics about the health of a Spring Boot application (such as CPU usage, memory consumption, and thread pool utilization)

https://docs.spring.io/spring-boot/reference/actuator/index.html#actuator

Include Spring Actuator endpoints (/actuator/health, /metrics).

Use:

- As part of your toolchain
- Export to Prometheus and visualize in Grafana.
- …

Compliant with https://micrometer.io/

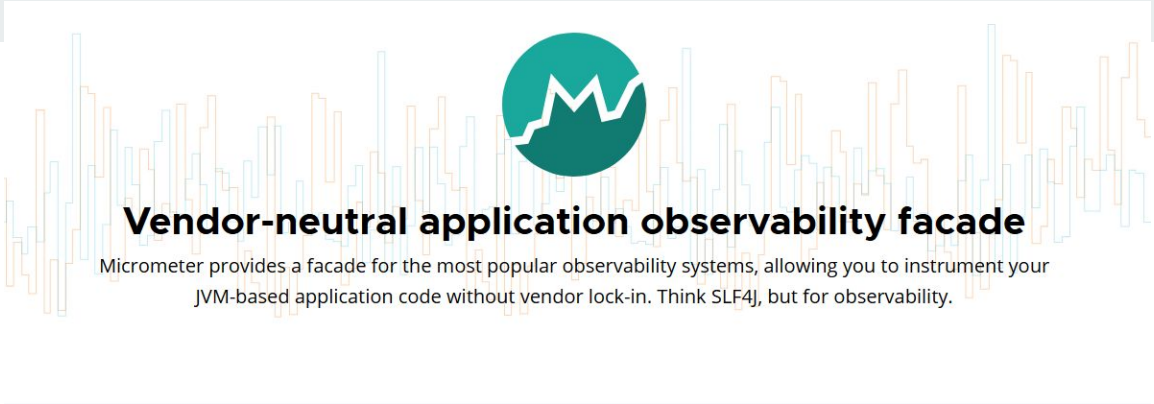
Definition of Actuator

An actuator is a manufacturing term that refers to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from a small change.

Micrometer

Micrometer provides a simple facade over the instrumentation clients for the most popular observability systems, allowing you to instrument your JVM-based application code without vendor lock-in.

"Think SLF4J, but for observability"



SB Actuator endpoints

https://spring.io/guides/gs/actuator-service

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Out-of-the-box it provides endpoints such as:

- `/actuator/health`: reports application liveness and readiness
- `/actuator/metrics`: exposes JVM, HTTP, and custom metrics
- `/actuator/prometheus` (with Micrometer): outputs Prometheus-formatted metrics Home

You can select which endpoints to expose via your `application.properties` or `application.yml`.

Sample endpoints.

For API reference → Actuator official docs.

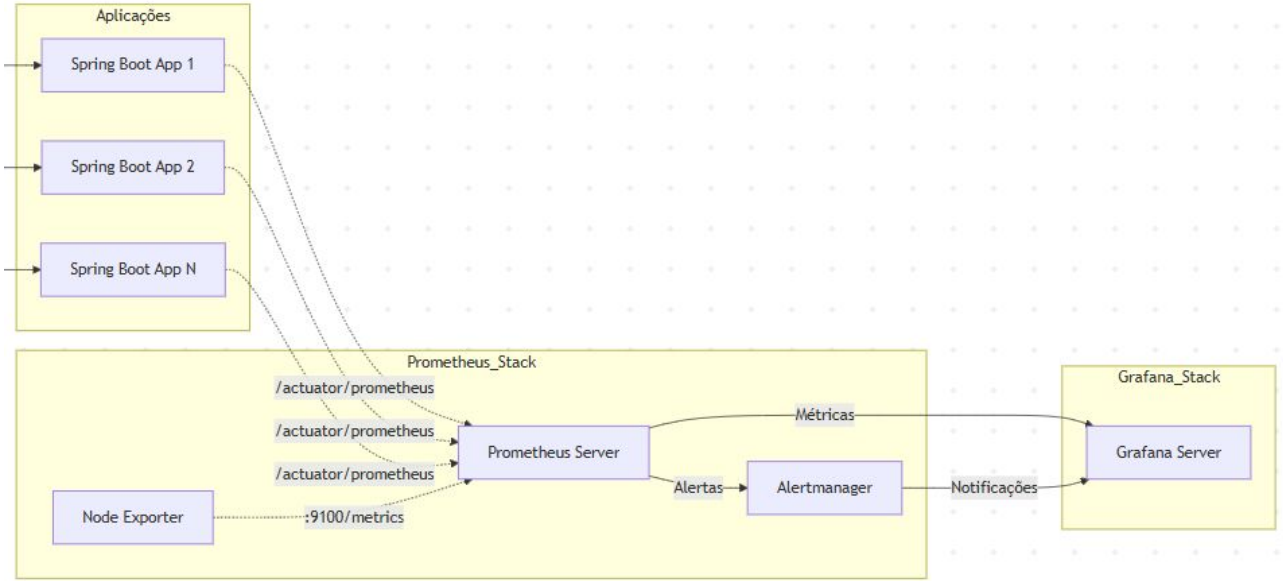
The following technology-agnostic endpoints are available:

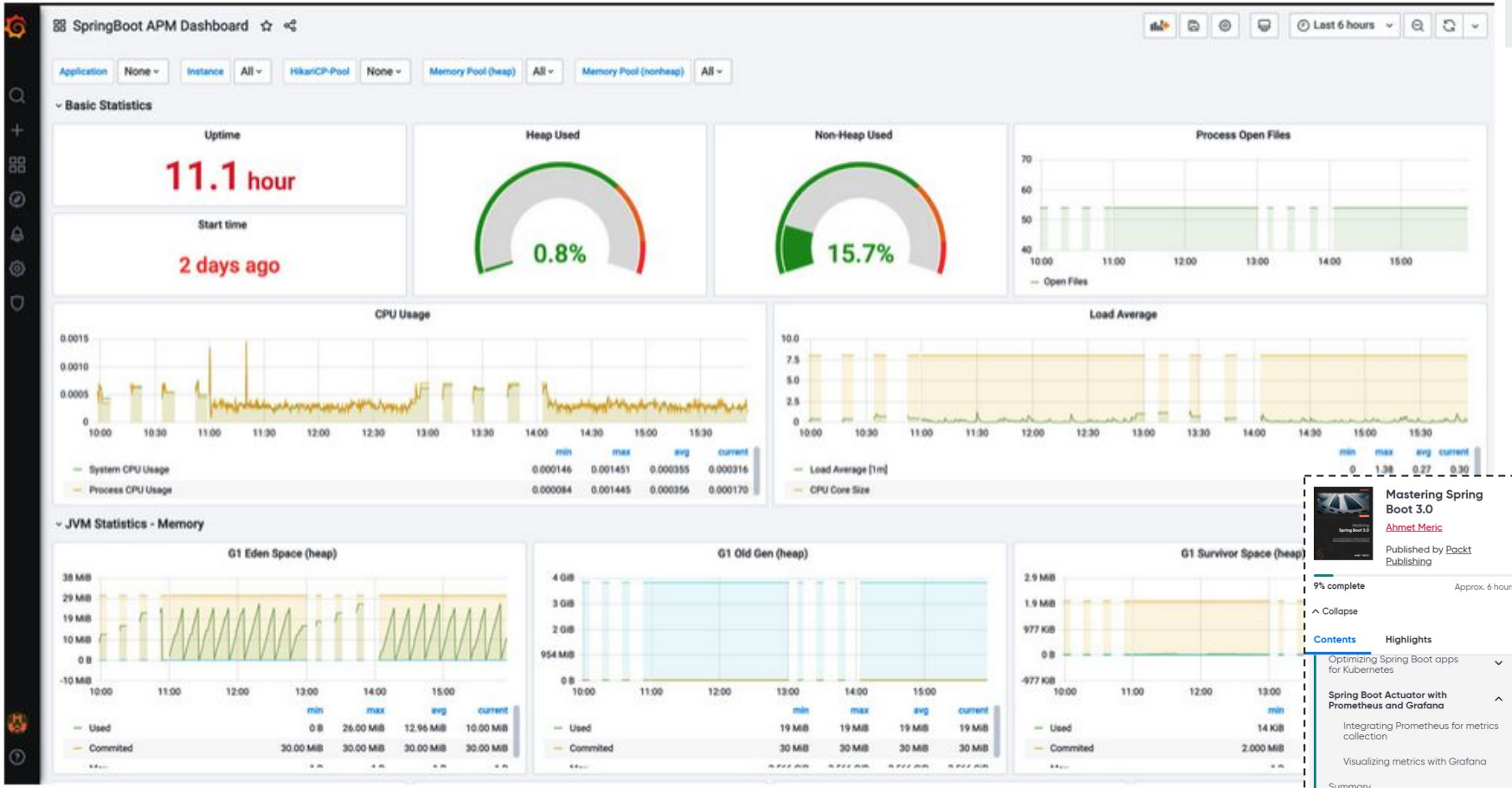
ID	Description
`auditevents`	Exposes audit events information for the current application. Requires an `AuditEventRepository` bean.
`beans`	Displays a complete list of all the Spring beans in your application.
`caches`	Exposes available caches.
`conditions`	Shows the conditions that were evaluated on configuration and auto-configuration classes and the reasons why they did or did not match.
`configprops`	Displays a collated list of all `@ConfigurationProperties`. Subject to sanitization.
`env`	Exposes properties from Spring's `ConfigurableEnvironment`. Subject to sanitization.
`flyway`	Shows any Flyway database migrations that have been applied. Requires one or more `Flyway` beans.
`health`	Shows application health information.
`httpexchanges`	Displays HTTP exchange information (by default, the last 100 HTTP request-response exchanges). Requires an `HttpExchangeRepository` bean.
`info`	Displays arbitrary application info.
`integrationgraph`	Shows the Spring Integration graph. Requires a dependency on `spring-integration-core`.
`loggers`	Shows and modifies the configuration of loggers in the application.
`liquibase`	Shows any Liquibase database migrations that have been applied. Requires one or more `Liquibase` beans.
`metrics`	Shows "metrics" information for the current application to diagnose the metrics the application has recorded.

Atuator (exporter)

Prometeus (timeseries)

Grafana (views)





In *Figure 7.1*, the Grafana dashboard showcases the performance metrics of a Spring Boot application visually

https://learning.oreilly.com/library/view/mastering-spring-boot/9781803230788/B18400_07.xhtml#_idParaDest-167

Traces

Figure 14.1: Example of a trace with its spans

Generic platforms

Nagios

Centralized Visibility: provides a unified web-based dashboard showing the status of all hosts, services and network devices in real time

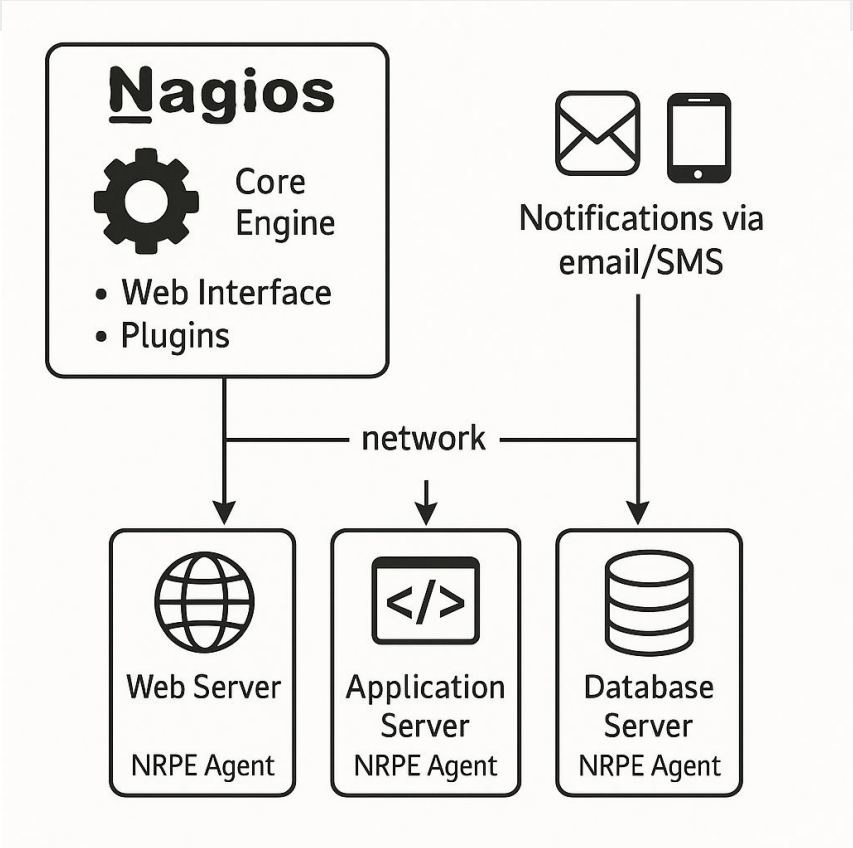
Extensible Plugin Architecture: thousands of community plugins

Flexible Alerting

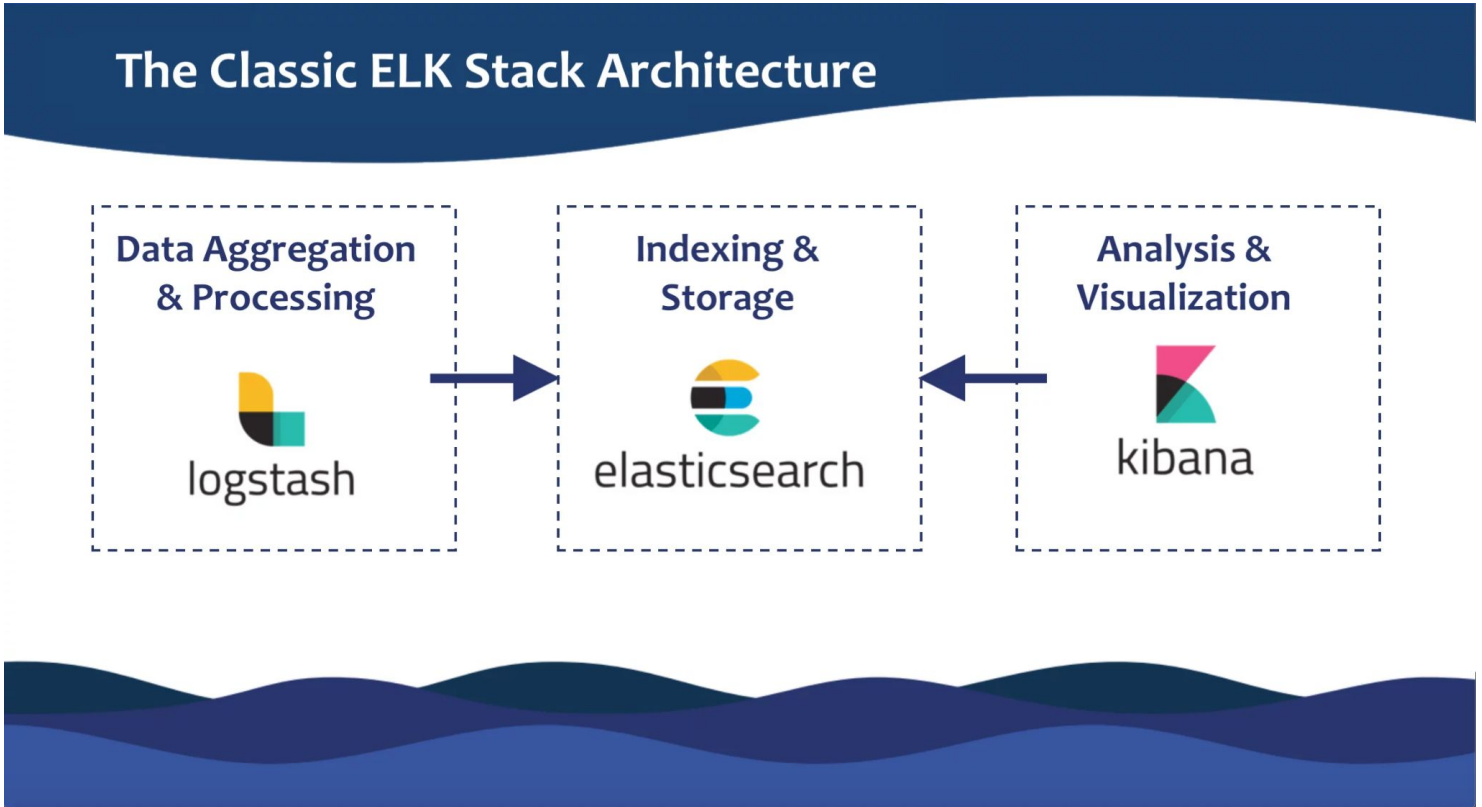
High Scalability & Performance

APIs & Third-Party Integrations

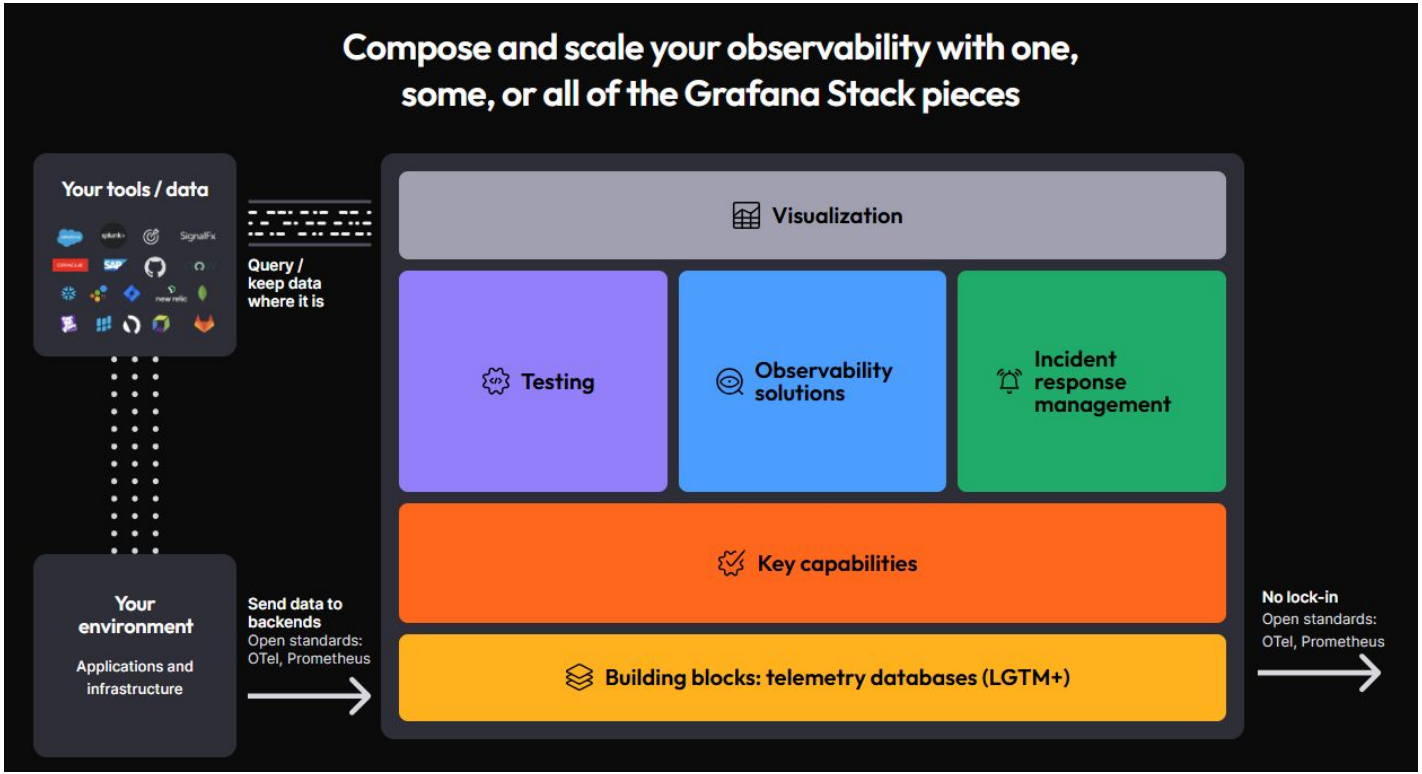
Nagios Remote Plugin Executor (NRPE) is a lightweight agent/service you may install on each remote host you wish to monitor.



ELK


The Classic ELK Stack Architecture
Data Aggregation & Processing
logstash
Indexing & Storage
elasticsearch
Analysis & Visualization
kibana

Grafana


Compose and scale your observability with one, some, or all of the Grafana Stack pieces
Your tools / data
Query / keep data where it is
Visualization
Testing
Observability solutions
Incident response management
Key capabilities
Building blocks: telemetry databases (LGTM+)
Your environment
Applications and infrastructure
Send data to backends
Open standards: OTel, Prometheus
No lock-in
Open standards: OTel, Prometheus