

<http://linkedin.com/in/guijac>

# Curso DevOps

Aula 14 - Monitoramento e Observabilidade em DevOps -  
Laboratório

---

**Prof. Esp. Guilherme Jorge Aragão da Cruz**

 [guilherme.jacruz@sp.senac.br](mailto:guilherme.jacruz@sp.senac.br)

 [linkedin.com/in/guijac](https://linkedin.com/in/guijac)



**Senac**

# Laboratório

---

Monitorar e observar a sua aplicação Java Spring + Boot com Prometheus e Grafana:

1. Expor métricas da aplicação com Actuator e Micrometer;
2. Subir stack com Spring Boot + Prometheus + Grafana via Docker Compose;
3. Configurar Prometheus para coletar métricas;
4. Conectar Grafana ao Prometheus (Data Source);
5. Criar dashboard no Grafana com pelo menos 3 painéis:
  1. Memória usada (JVM);
  2. Throughput (RPS por status);
  3. Taxa de erro (4xx/5xx).

# Expor métricas da aplicação

1. Adicione as dependências do Spring Actuator e Micrometer Prometheus em seu 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. Inclua as novas propriedades em seu application.properties:

```
# Actuator / Micrometer - expor endpoints básicos
management.endpoints.web.exposure.include=health,info,metrics,prometheus

# Health endpoint - mostrar detalhes, apenas para fins didáticos
management.endpoint.health.show-details=always
```

# Expôr métricas da aplicação

1. Inicie a sua aplicação e valide a resposta nas URLs de exposição de métricas /actuator/health e /actuator/prometheus:
  1. [localhost:8080/actuator/prometheus](http://localhost:8080/actuator/prometheus);
  2. [localhost:8080/actuator/health](http://localhost:8080/actuator/health)
2. Estude algumas das métricas para posterior uso em dashboard.

```
← ↻ ⓘ localhost:8080/actuator/prometheus
# HELP jvm_threads_daemon_threads The current number of live daemon threads
# TYPE jvm_threads_daemon_threads gauge
jvm_threads_daemon_threads 18.0
# HELP jvm_gc_max_data_size_bytes Max size of long-lived heap memory pool
# TYPE jvm_gc_max_data_size_bytes gauge
jvm_gc_max_data_size_bytes 8.573157376E9
# HELP tomcat_sessions_active_max_sessions
# TYPE tomcat_sessions_active_max_sessions gauge
tomcat_sessions_active_max_sessions 0.0
# HELP jvm_buffer_count_buffers An estimate of the number of buffers in the pool
# TYPE jvm_buffer_count_buffers gauge
jvm_buffer_count_buffers{id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_count_buffers{id="mapped",} 0.0
jvm_buffer_count_buffers{id="direct",} 1.0
# HELP process_start_time_seconds Start time of the process since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.758633065799E9
# HELP jvm_gc_live_data_size_bytes Size of long-lived heap memory pool after reclamation
# TYPE jvm_gc_live_data_size_bytes gauge
jvm_gc_live_data_size_bytes 0.0
# HELP jvm_memory_used_bytes The amount of used memory
# TYPE jvm_memory_used_bytes gauge
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'",} 1.028992E7
jvm_memory_used_bytes{area="heap",id="G1 Survivor Space",} 5015904.0
jvm_memory_used_bytes{area="heap",id="G1 Old Gen",} 1.5210064E7
jvm_memory_used_bytes{area="nonheap",id="Metaspace",} 4.0816504E7
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",} 1565568.0
jvm_memory_used_bytes{area="heap",id="G1 Eden Space",} 1.6777216E7
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",} 5267136.0
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'",} 3238912.0
# HELP jvm_threads_states_threads The current number of threads
# TYPE jvm_threads_states_threads gauge
jvm_threads_states_threads{state="runnable",} 8.0
jvm_threads_states_threads{state="blocked",} 0.0
```

# Subir stack Prometheus + Grafana

1. Siga a seguinte estrutura de diretórios para uso da stack de forma mais organizada e compatível com meus arquivos de suporte:

```
seu-projeto/
├── app/                               # Código da aplicação Spring Boot
│   ├── src/                           # Código-fonte Java
│   ├── pom.xml                         # Configuração do Maven
│   └── Dockerfile                       # Build da imagem da aplicação
├── prometheus/                         # Configuração do Prometheus
│   └── prometheus.yml                  # Targets e scrape jobs
├── grafana/                             # Configuração do Grafana
│   └── datasource.yml                 # Data source pré-configurado (Prometheus)
├── docker-compose.yml                  # Orquestração da stack
└── README.md                           # Tutorial/documentação da prática
```

# Subir stack Prometheus + Grafana

1. Crie o arquivo prometheus.yml no diretório prometheus e estude o seu conteúdo:

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: "spring-boot"
    metrics_path: "/actuator/prometheus"
    static_configs:
      - targets: ["spring-app:8080"]
```

2. Crie o arquivo datasource.yml no diretório grafana e estude o seu conteúdo:

```
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    access: proxy
    orgId: 1
    isDefault: true
    url: http://prometheus:9090
    editable: true
    jsonData:
      httpMethod: POST
      timeInterval: 15s
```

# Subir stack Prometheus + Grafana

1. Utilize o arquivo docker-compose base exibido em aula e mapeie um volume para o service prometheus:

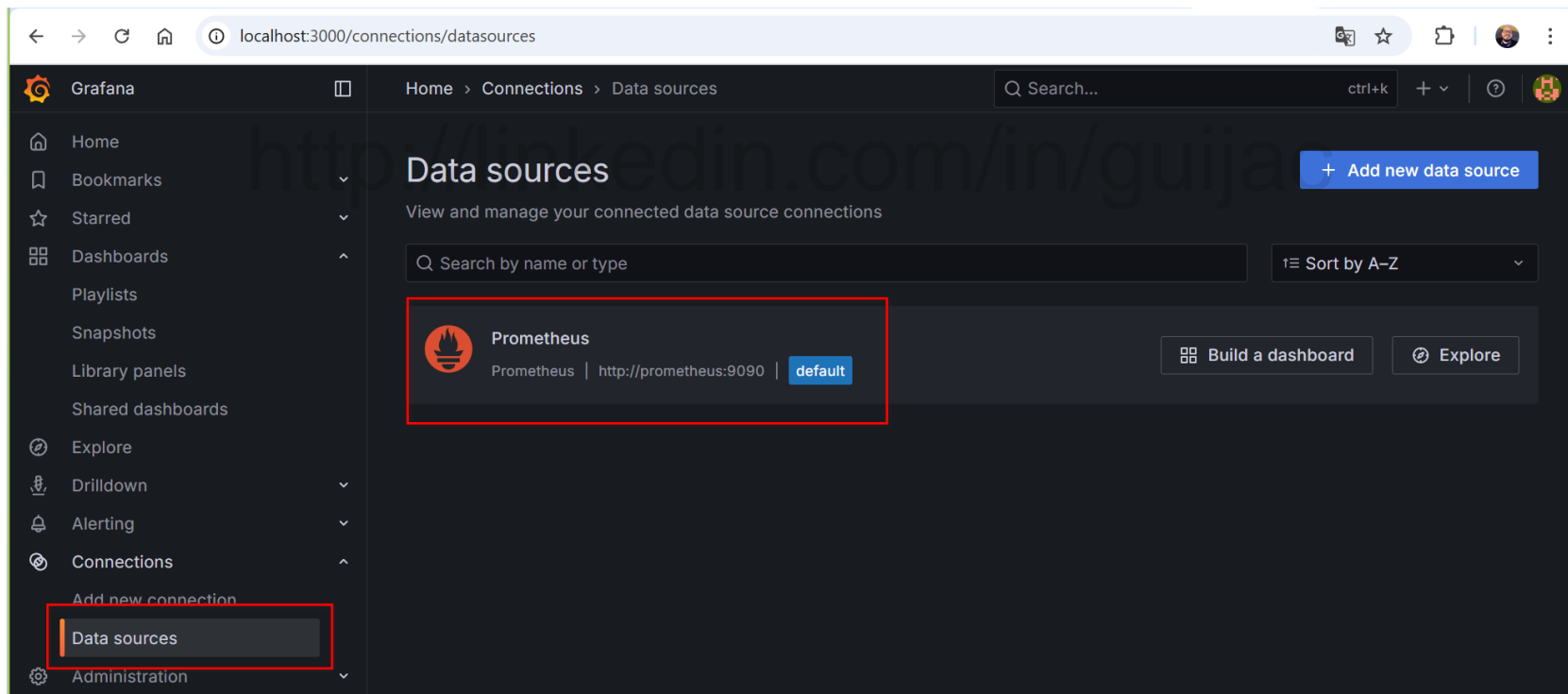
```
prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  ports:
    - "9090:9090"
  volumes:
    - ./prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
  depends_on:
    - spring-app
```

2. Utilize o arquivo docker-compose base exibido em aula e mapeie um volume para o service grafana:

```
grafana:
  image: grafana/grafana:latest
  container_name: grafana
  ports:
    - "3000:3000"
  environment:
    - GF_SECURITY_ADMIN_USER=admin
    - GF_SECURITY_ADMIN_PASSWORD=admin
  volumes:
    - ./grafana/datasource.yml:/etc/grafana/provisioning/datasources/default.yml
  depends_on:
    - prometheus
```

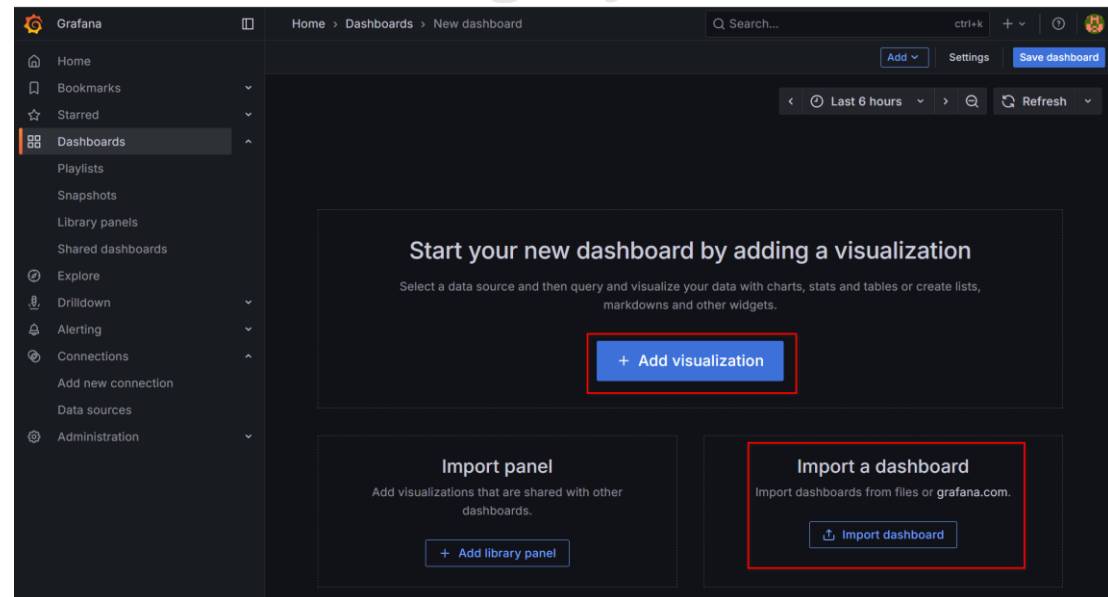
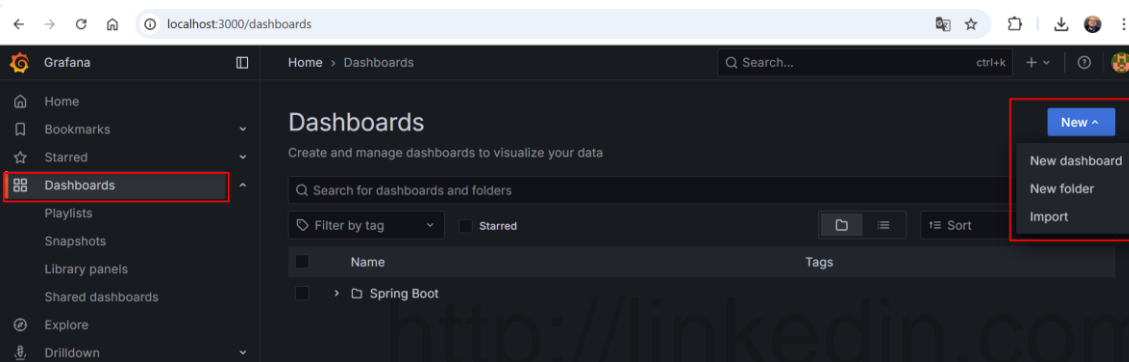
# Criar dashboard no Grafana

1. Suba o ambiente através do comando “docker-compose up --build”, acesse o Grafana com a credencial padrão, altere a senha e navegue pelos menus, se familiarizando com o ambiente;
2. Valide se o data source com o Prometheus foi mapeado corretamente:



# Criar dashboard no Grafana

1. Acesse o menu “Dashboards” > “New Dashboard”, note que também é possível importar dashboards diretamente da comunidade:



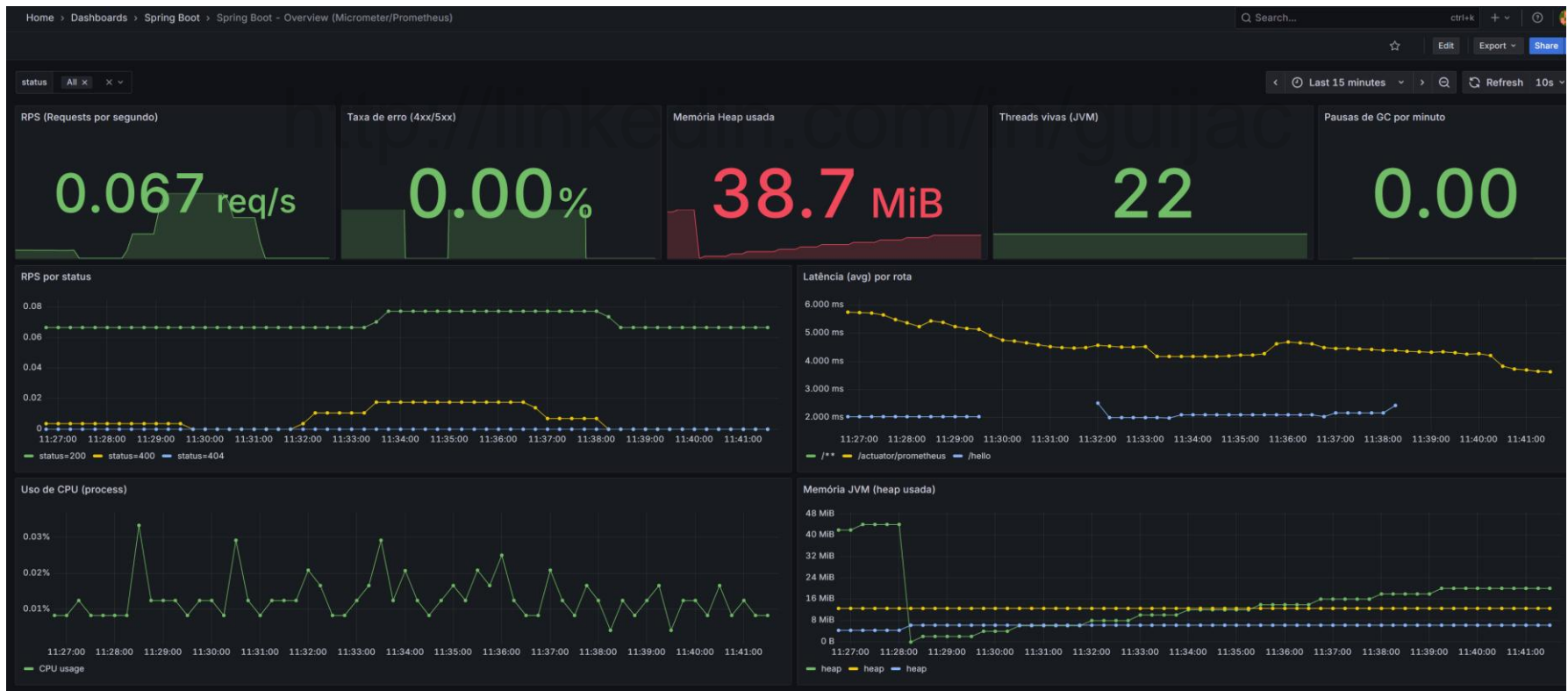
# Criar dashboard no Grafana

1. Crie a sua própria visualização, navegue pelas métricas existentes e avalie qual formato melhor se encaixa para sua necessidade:

The screenshot displays the Grafana dashboard editor interface. The main area shows a 'New panel' with 'No data' displayed. The right sidebar contains a 'Visualizations' menu with options like Time series, Bar chart, Stat, Gauge, Bar gauge, Table, Pie chart, State timeline, Heatmap, Status history, Histogram, and Text. The bottom section shows the 'Queries' tab with a 'Prometheus' data source selected and a 'Metric' dropdown menu.

# Criar dashboard no Grafana

1. Um dashboard de exemplo criado com as métricas:
  1. RPS (Requests por segundo) → `sum(rate(http_server_requests_seconds_count[5m]));`
  2. Taxa de erro (4xx/5xx) → `100 * sum(rate(http_server_requests_seconds_count{status=~"4..|5.."}[5m])) / sum(rate(http_server_requests_seconds_count[5m]));`
  3. Memória heap usada (JVM) → `jvm_memory_used_bytes{area="heap"};`
  4. Threads vivas (JVM) → `jvm_threads_live_threads;`
  5. Pausas de GC por minuto → `60 * sum(rate(jvm_gc_pause_seconds_count[5m]));`



# Referências Bibliográficas

---

12FACTOR. The Twelve-Factor App. Disponível em [https://12factor.net/pt\\_br/](https://12factor.net/pt_br/). Acesso em 19 jan 2025;

CNCF. **Cloud Native Definition v1.0**. Disponível em <https://github.com/cncf/toc/blob/main/DEFINITION.md#portugu%C3%AAs-brasileiro>. Acesso em 19 jan 2025;

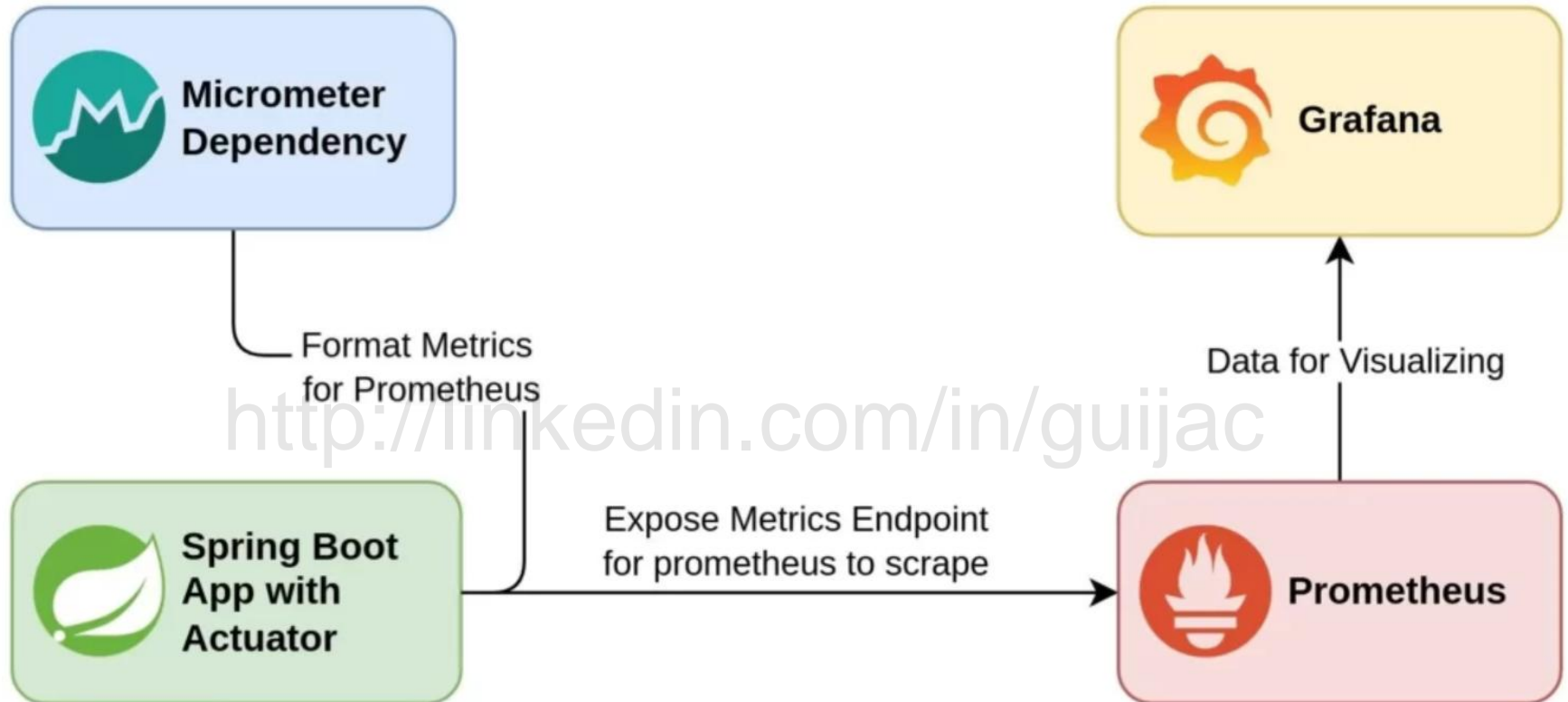
ERL, Thomas. **SOA Princípios de design de serviços**. Pearson Prentice Hall, São Paulo, 2014.

FOWLER, Martin; LEWIS, James. **Microservices a definition of this new architectural term**. Medium, 2014. Disponível em <http://martinfowler.com/articles/microservices.html>. Acesso em 19 jan 2025;

GUIJAC. **Microserviços: Uma Visão Geral — Parte I**. Disponível em <https://guijac.medium.com/microservi%C3%A7os-uma-vis%C3%A3o-geral-parte-i-88c0c9c547ee>. Acesso em 19 jan 2025;

JUNG, Matthias et al. **Microservices on AWS**. Amazon Web Services, Inc., New York, NY, USA, Tech. Rep, 2016.

# Por hoje (agora sim!) é só!



Fonte: [Run Prometheus and Grafana with Spring boot Actuator](#)

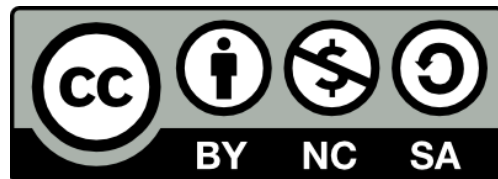
**Prof. Esp. Guilherme Jorge Aragão da Cruz**

✉ [guilherme.cruz@alumni.usp.br](mailto:guilherme.cruz@alumni.usp.br)

in [linkedin.com/in/guijac](https://linkedin.com/in/guijac)

# Licença

- Este conteúdo está licenciado sob a Licença Creative Commons Atribuição-NãoComercial-Compartilhalgual 4.0 Internacional (CC BY-NC-SA 4.0).
- Todos os direitos autorais sobre este conteúdo pertencem ao autor, e este material não pode ser usado comercialmente sem autorização expressa.
- Para ver o texto completo da licença, acesse o <https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode>.



---

**Prof. Esp. Guilherme Jorge Aragão da Cruz**

 [guilherme.cruz@alumni.usp.br](mailto:guilherme.cruz@alumni.usp.br)

 [linkedin.com/in/guijac](https://www.linkedin.com/in/guijac)