

Observabilidade + Ferramentas

- Clone a api de teste:

<https://github.com/AmandaGAB/api-observabilidade-teste>

PROMETHEUS

- Incluir no arquivo pom.xml as dependências do **spring actuator**

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

- Acessar endpoint criado pelo spring actuator
<http://localhost:8080/actuator>

O que está sendo exibido é algo parecido com:

```
{ "_links": { "self": { "href": "http://localhost:8080/actuator", "templated": false }, "health": { "href": "http://localhost:8080/actuator/health", "templated": false }, "health-path": { "href": "http://localhost:8080/actuator/health/{*path}", "templated": true } } }
```

As métricas estão sendo coletadas, mas ainda não estão sendo exibidas. Vamos ativá-las:

- Abra o arquivo application.properties e adicione as mettricas que devem ser exibidas nos endpoints:

management.endpoints.web.exposure.include=health, metrics

Agora ao acessar <http://localhost:8080/actuator> os seguintes endpoints serão exibidos:

```
{"_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"health":{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":{"href":"http://localhost:8080/actuator/health/{*path}","templated":true},"metrics-requiredMetricName":{"href":"http://localhost:8080/actuator/metrics/{requiredMetricName}","templated":true},"metrics":{"href":"http://localhost:8080/actuator/metrics","templated":false}}}
```

Agora diversas chaves serão exibidas com cada uma delas representando uma métrica .

- É preciso adicionar as dependências do micrometer para dar suporte ao prometheus no spring boot:

```
<dependency>  
  <groupId>io.micrometer</groupId>  
  <artifactId>micrometer-registry-prometheus</artifactId>  
  <version>1.9.0</version>  
</dependency>
```

- Modifique as configurações das métricas para apenas incluir o endpoint do prometheus:

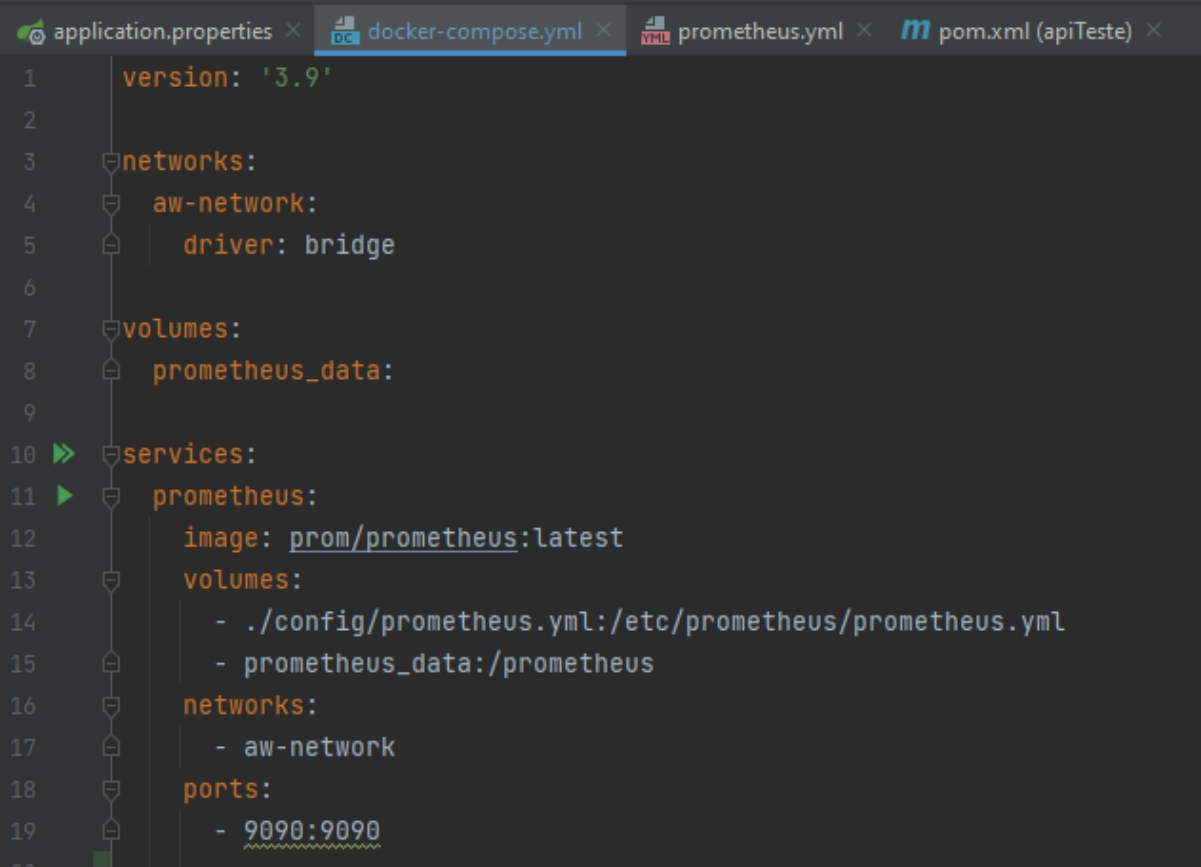
management.endpoints.web.exposure.include=prometheus

- Acesse o endpoint <http://localhost:8080/actuator/prometheus>

Deverá ser mostrado o conjunto de chave-valor que o prometheus aceita.

- Agora crie uma pasta para adicionar o arquivo yml do prometheus. Neste caso criei a pasta config e dentro dela adicionei um arquivo chamado **prometheus.yml**

- Crie na raiz do projeto um arquivo nomeado **docker-compose.yml** para subir o prometheus e o grafana (posteriormente) utilizando docker, mas você também pode instalá-los e rodar localmente. Adicione as linhas seguintes no arquivo docker-compose:



```
1  version: '3.9'
2
3  networks:
4    aw-network:
5      driver: bridge
6
7  volumes:
8    prometheus_data:
9
10 services:
11   prometheus:
12     image: prom/prometheus:latest
13     volumes:
14       - ./config/prometheus.yml:/etc/prometheus/prometheus.yml
15       - prometheus_data:/prometheus
16     networks:
17       - aw-network
18     ports:
19       - 9090:9090
```

```
version: '3.9'
```

```
networks:
```

```
  aw-network:
```

```
    driver: bridge
```

```

volumes:
  prometheus_data:

services:
  prometheus:
    image: prom/prometheus:latest
    volumes:
      -
    ./config/prometheus.yml:/etc/prometheus/prometheus.y
ml
      - prometheus_data:/prometheus
    networks:
      - aw-network
    ports:
      - 9090:9090

```

- No terminal execute **docker-compose up**
- Acesse a interface gráfica do prometheus em <http://localhost:9090/>
- Em **status** -> clique em **targets**. Você irá visualizar onde o prometheus está consumindo os dados de outras aplicações.
- Agora adicione as seguintes linhas no arquivo **prometheus.yml**:

```

global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    metrics_path: '/actuator/prometheus'
    scrape_interval: 5s
    static_configs:
      - targets: ['host.docker.internal:8080']

```

- Pare o container docker e execute novamente **docker-compose up**

- Agora na interface gráfica do prometheus acesse -> Graph -> digite a metrica `http_server_requests_seconds_max` e analise o gráfico

GRAFANA

Para configurar com docker:

- No **arquivo docker-compose.yml** adicione a linha **grafana_data** em **volumes** e em services as linhas abaixo:

```
grafana:
  image: grafana/grafana:latest
  ports:
    - 3000:3000
  networks:
    - aw-network
```

- Pare e execute novamente `docker-compose up`
- Acesse a interface gráfica do grafana em <https://localhost:3000>
- Nela clique no ícone da engrenagem -> configuration
- Depois clique em Add data Source -> Prometheus
- Em Http - Url digite **`http://localhost: 9090`** e em Acess selecione **server(default)** -> clique em save e teste
- Clique em Explore -> Metric > Adicione `http_server_requests_seconds_count`
- Analise o gráfico

Lembre-se de rodar a sua api na porta 8080!

REFERÊNCIAS

- <https://micrometer.io/docs>
- <https://docs.spring.io/spring-boot/docs/current/reference/html/actuator.html#actuator>
- <https://grafana.com/docs/>
- https://prometheus.io/docs/prometheus/latest/getting_started/
- <https://www.baeldung.com/spring-boot-actuators>
- <https://ordina-jworks.github.io/monitoring/2020/11/16/monitoring-spring-prometheus-grafana.html>