

# Cronapp APM

Ter uma visão do desempenho e verificar os problemas que podem ocorrer em tempo real são algumas das vantagens que o sistema de monitoramento oferecem. Para facilitar o monitoramento de sistemas, o Cronapp possui a ferramenta **Cronapp APM**, que permite exportar métricas da aplicação para serem usadas com o [Prometheus](#). O Cronapp APM foi construído a partir da API Spring Boot Actuator e do mecanismo micrometer-prometheus, permitindo gerar recursos de monitoramento, auditoria e outras propriedades pertencentes do Spring Boot no formato que é utilizado pelo Prometheus.

## Endpoints

Um endpoint é uma URI que normalmente direciona para o serviço de uma API. Dessa forma, a API do Spring Boot disponibiliza alguns endpoints próprios e também utiliza o endpoint pertencente ao Prometheus: **<URL do Servidor>/actuator/prometheus**, que gera as métricas no formato que será lido pelo servidor do Prometheus.

O formato de utilização do endpoint é geral: **actuator/id** como, por exemplo, actuator/health, actuator/env.

Saiba mais

Existem diversos endpoints gerados pelo Actuator, como <URL do Servidor>/actuator/health ou <URL do Servidor>/actuator/env. A lista com todos os endpoints pertencentes ao Spring boot actuator podem ser acessados no [link](#).

## Permissão de segurança

Como a funcionalidade já vem habilitada ao criar o projeto, é necessário dar permissão ao endpoint (recomenda-se inicialmente configurar a permissão para authenticated), que fica no endereço **/actuator/\*\***. Siga os passos abaixo para configurá-la.

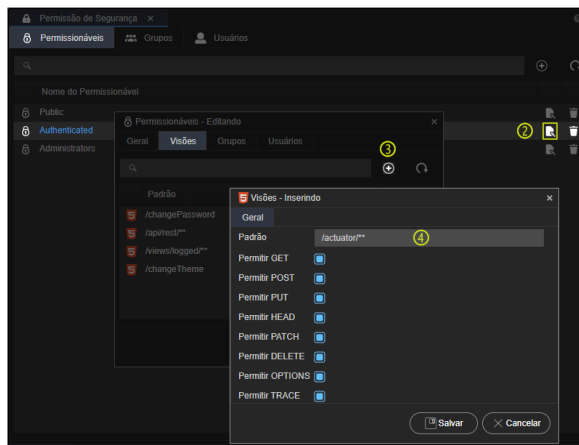


Figura 1.1 - Definir a permissão de segurança para acessar a página

1. Como o botão **Modo Avançado** habilitado, clique em **Projeto** no menu de sistemas e selecione **Permissão de Segurança**;
2. Após abrir a janela, na aba Permissionáveis clique no botão **Editar** de Authenticated;
3. Clique na aba **Visões** e, em seguida, no ícone "+" para **Adicionar**;
4. No campo Padrão escreva o caminho **/actuator/\*\***, deixe todos os itens selecionados, clique em **Salvar** e clique em **Salvar** novamente na janela de edição.

## Métricas exportadas

Com a permissão devidamente configurada, rode o projeto e acesse a URL **<URL do Servidor>/actuator** em seu navegador. Remova a parte **"#/home"** e acrescente **"actuator"** (Figura 1.2).

### Nesta página

- [Endpoints](#)
- [Permissão de segurança](#)
  - [Métricas exportadas](#)
- [Obtendo as métricas no Prometheus](#)
- [Visualizando as métricas em gráficos no Grafana](#)

### Figura 1.2 - Métricas exportadas pelo plugin

Nessa etapa será mostrado como utilizar as métricas geradas pela API no Prometheus.

### Figura 2.1 - Versão utilizada

1. Primeiramente, faça o download do [prometheus](#) e extraia a pasta.
2. Finalizada a extração, abra o arquivo **prometheus.yml** (pode ser aberto com o Bloco de Notas do Windows ou similar em outro sistema operacional) e **transcreva o código abaixo** para esse arquivo, de modo que ele esteja da mesma forma que é mostrado (destaque em vermelho) na Figura 2.1.
3. Em seguida volte ao ambiente Cronapp, rode o projeto, abra no navegador web e copie a **URL do Servidor**. Cole-o dentro (destaque 1 da Figura 2.1) da linha **targets**.
4. Salve o arquivo.

### Exemplo do scrape\_config para adicionar no arquivo

```
job_name: 'spring'
metrics_path: '/actuator/prometheus'
scheme: 'https'
static_configs:
  - targets: ['app-28-171-12118.ide.cronapp.io:443']
```

- `job_name`: define um nome para que o Prometheus reconheça
- `metrics_path`: informa o endereço em que as métricas são geradas e ele é fixo, não havendo necessidade de alteração.
- `scheme`: protocolo utilizado pela aplicação.
- `static_configs`: parâmetro utilizado para configurar os campos estáticos do target.
  - `targets`: informa o endereço da aplicação e a porta que ela utiliza no formato [**endereço:porta**]

### Observação

 **YAML**, a linguagem utilizada no Prometheus, utiliza uma notação baseada em indentação. Verifique no seu arquivo se está como na Figura 2.1.

```
prometheus.yml - Bloco de Notas
Arquivo Editar Formatar Exibir Ajuda
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
      - targets:
        # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label 'job=<job_name>' to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']

    - job_name: 'spring'
      metrics_path: '/actuator/prometheus'
      scheme: 'https'
      static_configs:
        - targets: ['app-28-171-12118-10.cronapp.io:443']
```

Figura 2.1 - Configurando arquivo prometheus.yml

Após salvar o arquivos, inicie a aplicação clicando duas vezes no arquivo *prometheus.exe*, abrirá a janela do terminal deixe-o aberta. Abra uma página no navegador e digite o endereço **localhost:9090**. Clique em **Status** (destaque 1 da Figura 2.2) e vá para a aba **Targets** (2) para visualizar se a aplicação está sendo reconhecida pelo Prometheus (3).

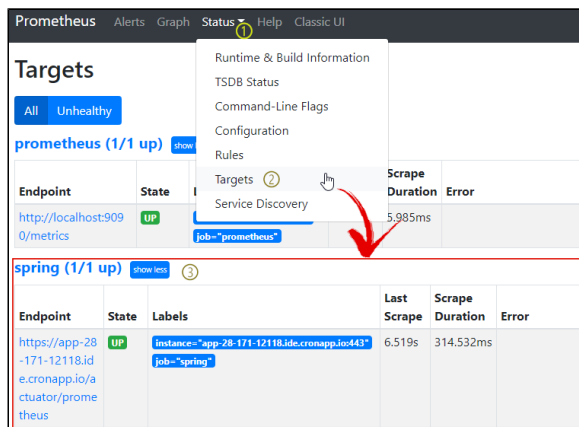


Figura 2.2 - Targets reconhecidas pelo Prometheus

Agora, clique no menu **Graph** do Prometheus e escolha uma query para obter os dados que são mostrados em **elements**, como mostrado na Figura 2.3.

Element	Value
jvm_memory_used_bytes{area="heap",id="Eden Space",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	23002968
jvm_memory_used_bytes{area="heap",id="Survivor Space",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	2416432
jvm_memory_used_bytes{area="heap",id="Tenured Gen",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	64551760
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	1311872
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'non-profiled nmethods'",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	7773056
jvm_memory_used_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	24249728
jvm_memory_used_bytes{area="nonheap",id="Compressed Class Space",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	10176344
jvm_memory_used_bytes{area="nonheap",id="Metaspace",instance="app-15-250-13150-10.cronapp.io:443",job="spring"}	88866616

Figura 2.3 - Dados retornados pela query escolhida

## Visualizando as métricas em gráficos no Grafana

A plataforma Grafana permite que as métricas disponibilizadas pelos softwares de monitoramento sejam visualizados através de gráficos. Com isso, faça o download do [Grafana](#) e instale-o em sua máquina. Após finalizar, abra o link default do Grafana, o **localhost:3000**, e entre com o usuário e senha padrão (admin/admin).

Ao logar no sistema, clique no ícone de configurações (1 da Figura 3.1) e clique em **data sources** (2 da Figura 3.1).

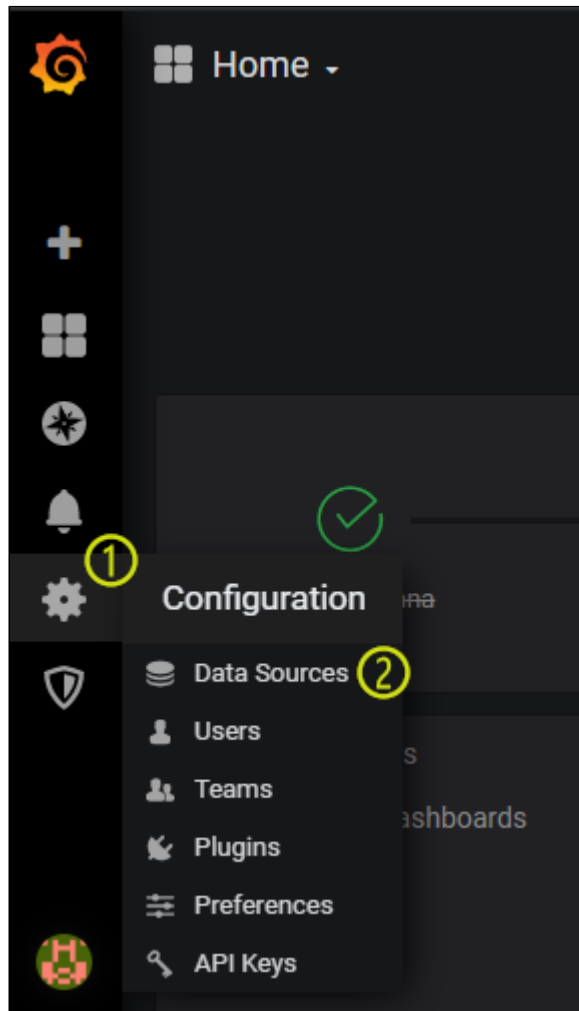


Figura 3.1 - Configurações > data sources

Clique em adicionar nova fonte de dados e selecione o Prometheus. No campo **url**, adicione o endereço do Prometheus (nesse caso, localhost:9090) (Figura 3.2) e salve.

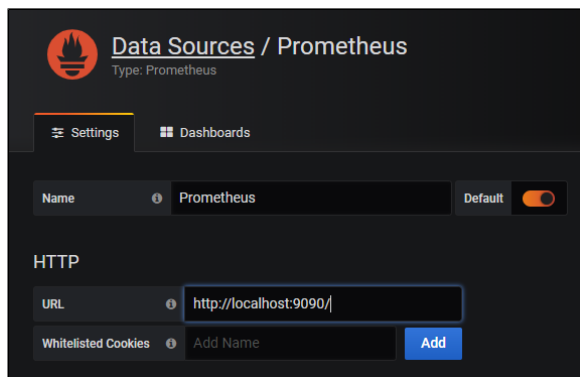


Figura 3.2 - Adicionar endereço do Prometheus

Por fim, crie um dashboard e clique em **add query**. Copie a query escolhida na página do prometheus no campo **metrics**.

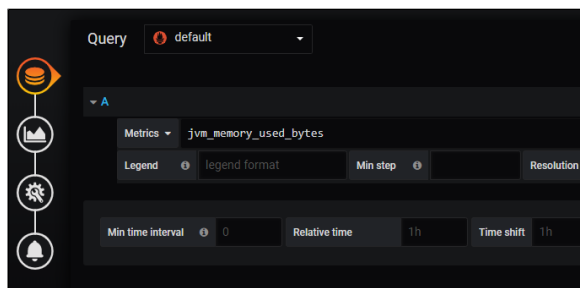


Figura 3.3 - Adicionar query no campo

Com isso, o gráfico será gerado automaticamente.



Figura 3.4 - Gráfico gerado pela query