Atividade Prática 07.

Prof. Dr. Eduardo dos Santos Pereira

9 de julho de 2024

1 Monitoramento de Contêineres

Existem diversas ferramentas para monitoramento de contêineres. Contudo é fundamental conhecer as ferramentas nativas utilizadas para inspecionar os contêineres.

Na lista a seguir são apresentados algumas ferramentas básicas. Execute cada uma delas.

a) Obter ID do contêiner:

```
docker container ps
```

b) Estatísticas de uso:

```
docker container stats CONTAINERID
```

c) Para visualizar todos os contêineres em execução:

```
docker container stats
```

d) Para verificar quais processos estão em execução em um determinado contêiner:

```
docker container top CONTAINERID
```

Responda:

I- Quais as métricas apresentadas ao utilizar o comando ‘docker container stats’?

2 Ferramentas de terceiros

Algumas das principais ferramentas de monitoramento de contêineres são Prometeus, cAdviosr e Grafana. Prometeus é uma ferramenta open source de monitoramento de sistemas a qual permite gerar e coletar métricas, plotando resultados em planilhas (dashboards) e alertando anomalias (`https://prometheus.io/`). cAdvisor permite monitorar a utilização de contêineres e entender a utilização de recursos e execuções. Trabalha em plano de fundo coletando, agregando, processando e exportando informações sobre os contêineres em execução. Por fim, o grafana é ´é uma plataforma para visualizar e analisar métricas de gráficos. Permite integração com diversas ferramentas e customização de dashboards.

2.1 Projeto Monitoramento Aplicação Web

Crie um repositório no GitHub com a seguinte estrutura de pastas:

```
monitoring_project/
|
|-- app/
|   |-- main.py
|   |-- requirements.txt
|
|-- prometheus/
|   |-- prometheus.yml
|
|-- grafana/
|   |-- provisioning/
|   |   |--dashboards/
|   |      |-- my_dashboard.json
|
|-- Dockerfile
|-- docker-compose.yml
```

A aplicação Python terá o seguinte conteúdo:

```
from flask import Flask
from prometheus_client import Counter, generate_latest
from prometheus_client.core import CollectorRegistry
from flask import Response

app = Flask(__name__)

REQUEST_COUNT = Counter('app_requests_total', 'Total App
   Requests')

@app.route('/')
def hello_world():
    REQUEST_COUNT.inc()
    return 'Bem-vindo ao mundo do DevOps!'

@app.route('/metrics')
def metrics():
    registry = CollectorRegistry()
    registry.register(REQUEST_COUNT)
    return Response(generate_latest(registry), mimetype='
       text/plain')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

O arquivo requirements.txt terá o seguinte conteúdo:

```
flask
prometheus_client
```

O Dockerfile é o arquivo responsável por gerar a imagem do contêiner e terá o seguinte conteúdo:

```
FROM python:3.8-slim

WORKDIR /app

COPY ./app /app

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "main.py"]
```

O arquivo de configuração do Prometheus (prometheus.yml) tem a seguinte informação:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'flask_app'
    static_configs:
      - targets: ['web:5000']

  - job_name: 'prometheus'
    scrape_interval : 5s
    static_configs :
      - targets: [ 'localhost:9090']

  - job_name : 'cadvisor'
    scrape_interval : 5s
    static_configs:
      - targets: ["cadvisor:8080"]
```

O Arquivo JSON de configuração do Grafan (my_dashboard.json)a será:

```
{
    "dashboard": {
        "annotations": {
            "list": []
         },
        "panels": [
        {
            "datasource": "Prometheus",
            "targets": [
            {
                "expr": "app_requests_total",
                "format": "time_series"
            }
            ],
            "title": "Total App Requests"
        }
        ]
    }
}
```

Por fim, o arquivo de configuração do docker-compose.yml será:

```
version: '3.8'

services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "5000:5000"

  prometheus:
        image: prom/prometheus
        volumes:
          - ./prometheus/prometheus.yml:/etc/prometheus/
            prometheus.yml
        ports:
          - "9090:9090"
        depends_on:
        - cadvisor

  cadvisor:
    image: gcr.io/cadvisor/cadvisor:latest
    container_name: cadvisor
        ports:
      - "8080:8080"
        volumes:
          - /:/rootfs:ro
          - /var/run:/var/run:rw
          - /sys:/sys:ro
          - /var/lib/docker/:/var/lib/docker:ro
        depends_on:
          - redis

  redis:
        image: redis:latest
        container_name: redis
        ports:
          - "6379:6379"
```

```
  grafana:
        image: grafana/grafana
        depends_on:
          - prometheus
        volumes:
          - ./grafana/provisioning:/etc/grafana/provisioning
        ports:
          - "3000:3000"
```

Inicialize a aplicação com o comando:

```
docker-compose up
```

Use os seguintes endereços para acessar as ferramentas:
Acessando as Ferramentas
Aplicação Flask: `http://localhost:5000`
Prometheus:`http://localhost:9090`
cAdvisor: `http://localhost:8080`
Grafana: `http://localhost:3000` Credenciais padrão: admin/admin