Instrumentação do Postgress - Prometheus

Laborátorio

O intuito desse laborátorio é instrumentar o Postgres e criar coletas customizadas (Querys) para Prometheus afim de exibir esses dados no Grafana, tudo isso afim de evitar que o Grafana executa querys diretamente no banco de dados Postgres.

Criação do Laboratório em Docker:

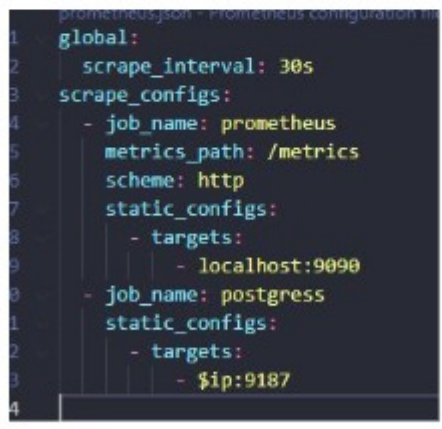
Prometheus:

docker run --name prometheus --rm -d -p 9090:9090 -v \Caminho\prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus

--name : Nome do Container
--rm : Excluir Prometheus e suas configurações sempre que ele for parado.
-d : Executar em segundo plano
-p : Porta de acesso
-v : Volume. Crie um arquivo .yml para adicionar as configurações de targets.
prom/prometheus : Imagem

Arquivo de Configuração prometheus.yml (Substituir $ip pelo ip da máquina)

```
global:
  scrape_interval: 30s
scrape_configs:
  - job_name: prometheus
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets:
          - localhost:9090
  - job_name: postgress
    static_configs:
      - targets:
          - $ip:9187
```



Grafana:

docker run -d --name=grafana -p 3000:3000 grafana/grafana-oss

--name : Nome do Container
-d : Executar em segundo plano
-p : Porta de acesso
grafana/grafana-oss : Imagem

Postgres:

docker run -d --name postgres -p 5432:5432 -e POSTGRES_PASSWORD=123 postgres

--name = Nome do Container
-d : Executar em segundo plano
-p : Porta de execução
POSTGRES_PASSWORD : Senha de acesso ao postgres (user postgres)
postgres : Imagem

Postgres_Exporter

docker run -d --name postgres_exporter -p 9187:9187 -v \caminho\queries.yml:/etc/queries.yml -e PG_EXPORTER_EXTEND_QUERY_PATH=/etc/queries.yml -e DATA_SOURCE_NAME="postgresql://postgres:123@192.168.15.102:5432/postgres?sslmode=disable" quay.io/prometheuscommunity/postgres-exporter:v0.10.0

--name : Nome do container

-d ; Executar em segundo plano

-e : definir variaveis

-v : Volume (Necessário criar um arquivo queries.yml e vincular a /etc/queries.yml (substituita o /caminho/ pelo caminho do arquivo em sua máquina)

PG_EXPORTER_EXTEND_QUERY_PATH: Caminho do arquivo de queries customizadas

DATA_SOURCE_NAME : Conexão com o banco postgres (no comando está utilizando postgres como user e 123 como password, está utilizando o IP DA MÁQUINA para se conectar ao postgres e não localhost.)

quay.io/prometheuscommunity/postgres-exporter:v0.10.0 : Imagem

Acessar: localhost:9187

Arquivo de exemplo: queries.yml

```
queriesyml > {} pg_replication_aula_teste > [ ] metrics > {} 0 > {} lag
pg_replication_aula_teste:
  query: "SELECT CASE WHEN NOT pg_is_in_recovery() THEN 0 ELSE GREATEST (0, EXTRACT(EPOCH FROM (now() - pg_last_xact_replay_timestamp()))) END AS lag"
  master: true
  metrics:
    - lag:
        usage: "GAUGE"
        description: "Replication lag behind master in seconds"
```

Comando para verificar se metrica customizada foi criada:
curl -s http://localhost:9187/metrics | grep pg_replication_aula_teste

```
┌──(root㉿NT-AGERI-21)-[/home/dennis]
└─# curl -s http://localhost:9187/metrics | grep pg_replication_aula_teste
# HELP pg_replication_aula_teste_lag Replication lag behind master in seconds
# TYPE pg_replication_aula_teste_lag gauge
pg_replication_aula_teste_lag{server="192.168.15.102:5432"} 0
```