Daten aus Prometheus in eine SQL-Datenbank schreiben

Description

In diesem Artikel beschreibe ich, wie man Daten aus einer Prometheus-Datenbank in eine SQL-Datenbank automatisiert übertragen kann. Dies habe ich z.B. gebraucht, um Metriken in einer Analysesoftware darzustellen, die keine Möglichkeit bietet, die Daten aus Prometheus zu verwenden. So habe ich mir damit einen Workaround gebaut.

Durchführung

Installation von Sling

Im ersten Schritt müssen wir auf unserem Linux-Server **sling** installieren. Dazu führen wir den folgenden Befehl aus:

```
curl -LO
'https://github.com/slingdata-io/sling-cli/releases/latest/download/sling_linux_amd64.tar
 \
  && tar xf sling_linux_amd64.tar.gz \
  && rm -f sling_linux_amd64.tar.gz \
  && chmod +x sling
  && mv ./sling /usr/local/bin/sling
  && chmod +x /usr/local/bin/sling
```

Im Anschluss müssten wir in der Lage sein, mit dem Befehl **sling** mit dem Programm zu interagieren.

Verbindungen einrichten

Im nächsten Schritt müssen wir die Verbindungen für Sling auf die verschiedenen Datenquellen einrichten. Der Befehl lautet ***sling conns set <NAME> type=<TYP> <PARAMETER>***.

Zuerst richten wir die Datenquelle für unseren Prometheus Server ein. Dazu verwenden wir den folgenden Befehl:

```
sling conns set <Name> type=prometheus http_url=<http://IP:9090
```

Im Anschluss erstellen wir jetzt die Verbindung her für unseren SQL-Server. Dabei müssen wir unterscheiden, mit was für einen SQL-Server wir uns verbinden möchten.

```
sling conns set <Name> type=<typ> url="<typ>://<sql-url>"
```

Datenbanktyp	Typ in Verbindung	SQL-URL
Microsoft SQL	sqlserver	sqlserver://myuser:mypass@host.ip:1433?database=mydatabase

PostgreSQL	postgres	postgresql://myuser:mypass@host.ip:5432/mydatabase?sslmode=requ
MySQL	mysql	mysql://myuser:mypass@host.ip:3306/mydatabase?tls=skip-verify

Im nächsten Schritt testen wir einmal die Verbindungen, ob diese sich erfolgreich mit ihren Datenquellen verbinden können. Dazu führen wir die folgenden Befehle aus:

```
sling conns test <Name>
```

Wenn alles geklappt hat, sollten wir mit einer grünen Meldung begrüßt werden.

Konfigurationsdatei erstellen

Die Übertragung der Prometheus-Metriken müssen wir immer in einer Konfigurationsdatei spezifizieren. Sonst werden von Haus aus gar keine Metriken übertragen. Dazu legen wir in einem Verzeichnis unserer Wahl eine **YAML-Datei** an. Diese enthält den folgenden Inhalt:

```
source: <Quelle>
target: <Ziel>

defaults:
  object: <sql-object>_{stream_name}
  mode: <Mode>

streams:
  <streams>
```

Die Streams werden quasi immer extra aufgezählt mit einer PromQL Query. So können wir Datensätze filtern, wenn z.B. nicht alle übertragen werden sollen. Bei einem Microsoft SQL-Server kann die Datei wie folgt aussehen:

```
source: PROMETHEUS
target: MSSQL

defaults:
  object: dbo.prometheus_{stream_name}
  mode: full-refresh

streams:
  up:
    sql: 'sum(up) by (job, instance) # {"start": "now-7d", "end": "now", "step": "1d"}'
  sensor_temperature:
    sql: 'sensor_temperature # {"start": "now-7d", "end": "now", "step": "1d"}'
  vmware_host_cpu_usage_average:
    sql:
'vmware_host_cpu_usage_average / 100 # {"start": "now-7d", "end": "now", "step": "1d"}'

  vmware_host_memory_usage:
    sql: 'vmware_host_memory_usage # {"start": "now-7d", "end": "now", "step": "1d"}'
  vmware_host_memory_max:
    sql: 'vmware_host_memory_max # {"start": "now-7d", "end": "now", "step": "1d"}'
  vmware_datastore_freespace_size:
    sql:
'vmware_datastore_freespace_size # {"start": "now-7d", "end": "now", "step": "1d"}'
  upsBatteryTemperature:
    sql: 'upsBatteryTemperature # {"start": "now-7d", "end": "now", "step": "1d"}'
  upsEstimatedMinutesRemaining:
    sql:
'upsEstimatedMinutesRemaining # {"start": "now-7d", "end": "now", "step": "1d"}'
  windows_logical_disk_free_bytes:
    sql:
'windows_logical_disk_free_bytes # {"start": "now-7d", "end": "now", "step": "1d"}'
  windows_logical_disk_size_bytes:
    sql:
'windows_logical_disk_size_bytes # {"start": "now-7d", "end": "now", "step": "1d"}'
  upsBatteryStatus:
    sql: 'upsBatteryStatus # {"start": "now-7d", "end": "now", "step": "1d"}'
  upsOutputPower:
    sql: 'upsOutputPower # {"start": "now-7d", "end": "now", "step": "1d"}'
  mssql_client_connections:
    sql: 'mssql_client_connections # {"start": "now-7d", "end": "now", "step": "1d"}'
  mssql_transactions:
    sql: 'mssql_transactions # {"start": "now-7d", "end": "now", "step": "1d"}'
  mssql_database_filesize:
    sql: 'mssql_database_filesize # {"start": "now-7d", "end": "now", "step": "1d"}'
```

Übertragung starten

Wenn wir jetzt eine Übertragung starten möchten, brauchen wir nur den folgenden Befehl verwenden:

```
sling run -r <pfad-zur-datei>
```

Im Anschluss wird die Konfigurationsdatei eingelesen und auf den SQL-Server übertragen. Das Skript wird aber nur einmal ausgeführt.

Daten periodisch übertragen

Falls die Daten periodisch automatisch übertragen werden sollen, können wir dies z.B. mit einem Crontab realisieren. Ich habe einen Crontab angelegt, der dann alle 15 Sekunden automatisch die Daten in die Datenbank überträgt.

```
* * * * * /usr/local/bin/sling run -r /opt/replication.yaml > /var/log/sling.log 2>&1
* * * * *
sleep 15; /usr/local/bin/sling run -r /opt/replication.yaml >> /var/log/sling.log 2>&1
* * * * *
sleep 30; /usr/local/bin/sling run -r /opt/replication.yaml >> /var/log/sling.log 2>&1
* * * * *
sleep 45; /usr/local/bin/sling run -r /opt/replication.yaml >> /var/log/sling.log 2>&1
```

Category

1. Allgemein SQL
2. Prometheus

Date Created
23.05.2025
Author
administrator