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 Ã¼bertragen kann. Dies habe ich z.B. gebraucht, um Metriken in einer Analysesoftware darzustellen, die keine MÃ¶glichkeit bietet, die Daten aus Prometheus zu verwenden. So habe ich mir damit einen Workaround gebaut.

DurchfÃ¼hrung

Installation von Sling

Im ersten Schritt mÃ¼ssen wir auf unserem Linux-Server **sling** installieren. Dazu fÃ¼hren 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 mÃ¼ssten wir in der Lage sein, mit dem Befehl **sling** mit dem Programm zu interagieren.

Verbindungen einrichten

Im nÃ¤chsten Schritt mÃ¼ssen wir die Verbindungen fÃ¼r Sling auf die verschiedenen Datenquellen einrichten. Der Befehl lautet ***sling conns set <NAME> type=<TYP> <PARAMETER>***.

Zuerst richten wir die Datenquelle fÃ¼r 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 fÃ¼r unseren SQL-Server. Dabei mÃ¼ssen wir unterscheiden, mit was fÃ¼r einen SQL-Server wir uns verbinden mÃ¶chten.

```
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 nÃ¤chsten Schritt testen wir einmal die Verbindungen, ob diese sich erfolgreich mit ihren Datenquellen verbinden kÃ¶nnen. Dazu fÃ¼hren wir die folgenden Befehle aus:

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

Wenn alles geklappt hat, sollten wir mit einer grÃ¼nen Meldung begrÃ¼Ã?t werden.

Konfigurationsdatei erstellen

Die Ã?bertragung der Prometheus-Metriken mÃ¼ssen wir immer in einer Konfigurationsdatei spezifizieren. Sonst werden von Haus aus gar keine Metriken Ã¼bertragen. Dazu legen wir in einem Verzeichnis unserer Wahl eine **YAML-Datei** an. Diese enthÃ¤lt den folgenden Inhalt:

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

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

streams:
  <streams>
```

Die Streams werden quasi immer extra aufgezÃ¤hlt mit einer PromQL Query. So kÃ¶nnen wir DatensÃ¤tze filtern, wenn z.B. nicht alle Ã¼bertragen 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"}'
```

Ã?bertragung starten

Wenn wir jetzt eine Ã?bertragung starten mÃ¶chten, brauchen wir nur den folgenden Befehl verwenden:

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

Im Anschluss wird die Konfigurationsdatei eingelesen und auf den SQL-Server Ã¼bertragen. Das Skript wird aber nur einmal ausgefÃ¼hrt.

Daten periodisch Ã¼bertragen

Falls die Daten periodisch automatisch Ã¼bertragen werden sollen, kÃ¶nnen wir dies z.B. mit einem Crontab realisieren. Ich habe einen Crontab angelegt, der dann alle 15 Sekunden automatisch die Daten in die Datenbank Ã¼bertrÃ¤gt.

```
* * * * * /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