InfluxDB Übung 2

Aufgabe 1

UE11-02-Wetterdaten importieren und Abfragen sowie Löschvorgänge mit Flux ausführen

1. Erstelle eine einfache Datenbank (Bucket) mit einigen Datensätzen zum Wetter. Die Daten kommen von 3 Wetterstationen `Station1`, `Station2` und `Station3`. Jede Station zeichnet die *Temperatur*, die *Feuchtigkeit* und die Uhrzeit jede Stunde auf. Die Wetterstationen sollen als *Tags* gespeichert werden. Der *Bucket*, welcher Wetterdaten abspeichert hat den Namen `weather`. Das Einfügen der Daten in den *Bucket* soll mit einem Python-Script erfolgen.
2. Führe mit *Flux* Abfragen im *Data Explorer* durch. InfluxDB UI - Execute a Flux query
   a. Alle Daten selektieren
   b. Daten von `Station1` selektieren. Hinweis hierzu: *Um Selektionen auf Tags in Flux zu machen, nutzt du die* `filter` *-Funktion. Tags sind in Flux einfach zu filtern, da sie indexiert sind und schnelle Abfragen ermöglichen.*
   c. Durchschnittstemperatur berechnen
   d. Daten innerhalb eines bestimmten Zeitraums selektieren
3. Daten mit Flux löschen
   a. Alle Daten von `Station2` löschen
   b. Daten innerhalb eines bestimmten Zeitraums löschen

Wie man Daten in einen Bucket schreibt

Write data to InfluxDB with Python

Lösungsvorschläge

Lösungsvorschlag Python-Skript zum Generieren und Laden von Wetterdaten für 3 Wetterstationen

UE11-02-Station123Weather.py

```python
import random
import influxdb_client
from influxdb_client.client.write_api import SYNCHRONOUS
from datetime import datetime, timedelta

# Verbindung zu InfluxDB herstellen
bucket = "weather"
org = "BFH"
token = "IHR TOKEN"
url = "http://localhost:8086"

client = influxdb_client.InfluxDBClient(
    url=url,
    token=token,
    org=org
)

write_api = client.write_api(write_options=SYNCHRONOUS)

# Beispiel-Daten von 3 Wetterstationen hinzufügen
stations = ["Station1", "Station2", "Station3"]
start_time = datetime.utcnow() - timedelta(days=2)  # Vor 2 Tagen starten

data_points = []
for station in stations:
    current_time = start_time
    for _ in range(100):  # 100 verschiedene Werte für jede Station
        temperature = round(random.uniform(15, 25), 2)  # Zufällige Temperatur zwischen 15 und
25 Grad
        humidity = round(random.uniform(50, 80), 2)     # Zufällige Feuchtigkeit zwischen 50
und 80 Prozent
        point = influxdb_client.Point("weather") \
            .tag("station", station) \
            .field("temperature", temperature) \
            .field("humidity", humidity) \
            .time(current_time)

        data_points.append(point)
        current_time += timedelta(hours=1)

# Datenpunkte in den Bucket schreiben
write_api.write(bucket=bucket, record=data_points)
print("Wetterdaten erfolgreich in den InfluxDB-Bucket geschrieben")

# Verbindung schliessen
client.close()
```

Query-Bedingung

Alle Daten selektieren

FLUX

```
from(bucket: "weather")
|> range(start: -2d)
|> filter(fn: (r) => r._measurement == "weather")
```

Query-Bedingung

Daten von Station1 selektieren

FLUX

```
from(bucket: "weather")
|> range(start: -2d)
|> filter(fn: (r) => r._measurement == "weather" and r.station == "Station1")
```

Query-Bedingung

Durchschnittstemperatur berechnen

FLUX

```
from(bucket: "weather")
|> range(start: -2d)
|> filter(fn: (r) => r._measurement == "weather")
|> mean(column: "temperature")
```

Query-Bedingung

Daten innerhalb eines bestimmten Zeitraums selektieren

FLUX

```
from(bucket: "weather")
|> range(start: 2023-12-05T00:00:00Z, stop: 2023-12-07T00:00:00Z)
|> filter(fn: (r) => r._measurement == "weather")
```

Query-Bedingung

Durchschnittstemperatur berechnen

FLUX

```
from(bucket: "weather")
|> range(start: -2d)
|> filter(fn: (r) => r._measurement == "weather")
|> mean(column: "temperature")
```

Query-Bedingung

Alle Daten von Station2 löschen

FLUX

```
import "influxdata/influxdb/v1"

v1.delete(
  bucket: "weather",
  predicate: (r) => r._measurement == "weather" and r.station == "Station2",
  start: -30d,
  stop: now()
)
```

Query-Bedingung

Daten innerhalb eines bestimmten Zeitraums löschen

FLUX

```
import "influxdata/influxdb/v1"

v1.delete(
  bucket: "weather",
  predicate: (r) => r._measurement == "weather",
  start: 2023-12-05T00:00:00Z,
  stop: 2023-12-07T00:00:00Z
)
```

Aufgabe 2

UE11-02-InfluxDB-CLI

In anderen Modulen werden TSDB und insb. InfluxDB ein Thema bleiben. Lernen Sie auch das CLI (Command Line Interface) kennen. Abfragen mit dem CLI sind praktisch und schnell.

CLI Installation

Führen Sie die Abfragen von oben mit dem CLI aus.

CLI Execute a Flux query