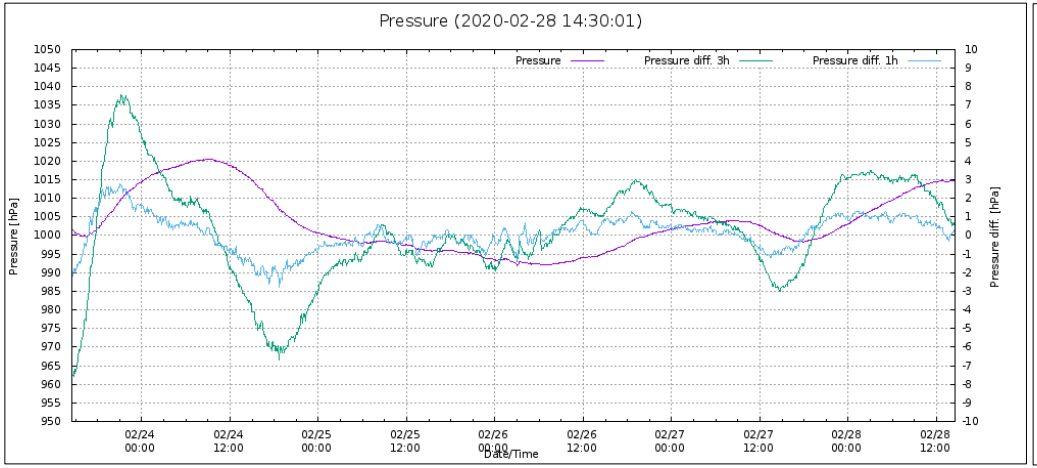
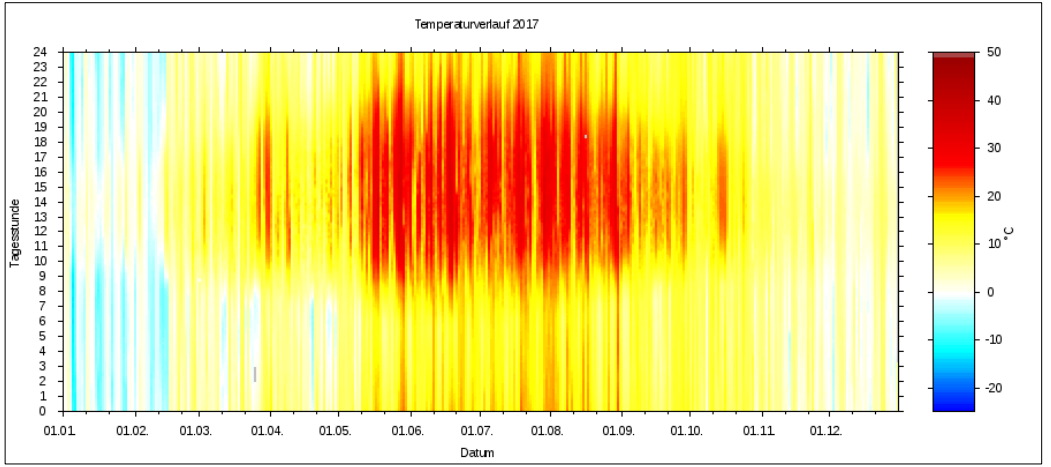
InfluxDB – eine Einführung

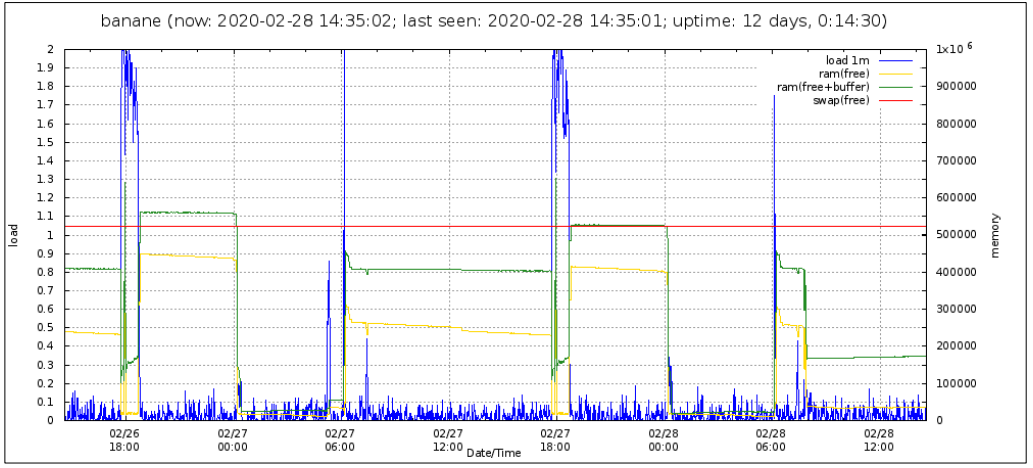
Uwe Berger
bergeruw@gmx.net

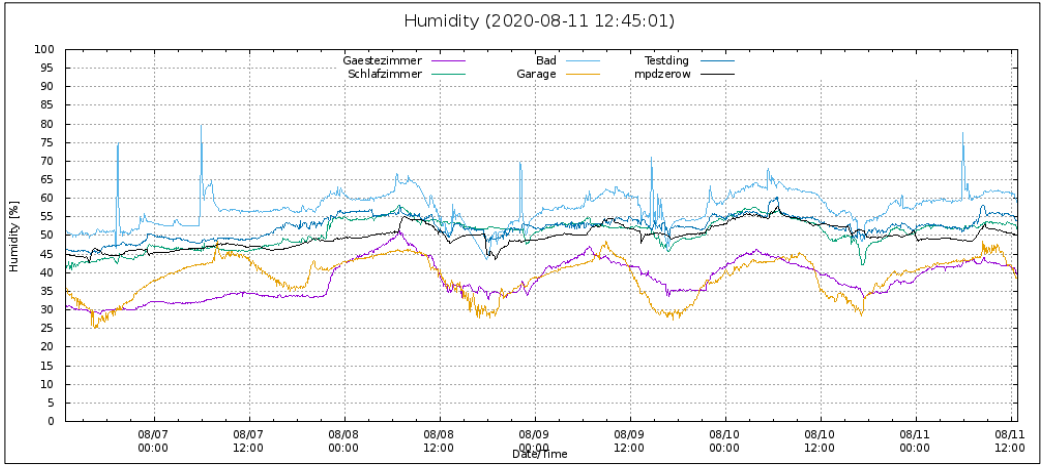
Uwe Berger

Warum jetzt InfluxDB...?









Daten...

- ...fallen überall und ständig an
- ...unterscheiden sich in ihren Eigenschaften
- ...landen meist in Datenbanken
- ...werden unterschiedlich weiterverarbeitet

Datenbanken...

- ...gab es früher wenige und diese waren meist auf „alles“ spezialisiert
- ...gibt es heute viele und diese sind jeweils meist auf ein spezielles Anwendungsgebiet optimiert

Datenbankmodelle (Beispiele)

- Relationale Datenbanken
- Dokumentorientierte Datenbanken
- Objektorientierte Datenbanken
- Diagrammdatenbanken
- Suchmaschinendatenbanken
- Zeitreihendatenbanken

Datenbankmodelle (Beispiele)

- Relationale Datenbanken
- Dokumentorientierte Datenbanken
- Objektorientierte Datenbanken
- Diagrammdatenbanken
- Suchmaschinendatenbanken
- **Zeitreihendatenbanken**

Zeitreihen...

- ...sind Serien von Messungen, Beobachtungen, Zuständen o.ä. an aufeinanderfolgenden Zeitpunkten
  - Beispiele: Wetterdaten, Monitoring-Daten, Börsenkurse ...halt alles, was über die Zeit betrachtet werden kann!
- Wozu?
  - Darstellung und Archivierung des zeitlichen Verlaufs von Mess-werten
  - Statistische Analysen → Trends, Zyklen, Vorhersagen etc.

Zeitreihendatenbanken

- Anforderungen:
  - Organisation der Daten nach Zeit
  - Hohe Anzahl von Schreibvorgängen (Echtzeit…)
  - Hohe Anzahl von parallelen Datenquellen
  - Hohe Flexibilität bei Definition/Typisierung von Daten
- Aktualisierungen von Datensätzen sind selten
- Funktionen zum automatisierten Löschen und Komprimieren von Daten wären cool…!

Zeitreihendatenbanken

Rang Feb 2020	Rang Jan 2020	Rang Feb 2019	DBMS	Datenbankmodell	Punkte Feb 2020	Punkte Jan 2020	Punkte Feb 2019
1.	1.	1.	InfluxDB	Time Series	21,57	+0,44	+5,81
2.	2.	2.	Kdb+	Time Series, Multi-Model	5,38	-0,11	-0,02
3.	3.	↑ 5.	Prometheus	Time Series	4,01	-0,06	+1,51
4.	4.	↓ 3.	Graphite	Time Series	3,34	-0,01	+0,39
5.	5.	↓ 4.	RRDtool	Time Series	2,68	-0,09	-0,02
6.	6.	6.	OpenTSDB	Time Series	2,14	+0,17	-0,10
7.	↑ 8.	7.	Druid	Multi-Model	1,91	+0,03	+0,42
8.	↓ 7.	8.	TimescaleDB	Time Series, Multi-Model	1,88	-0,04	+1,00
9.	9.	↑ 11.	FaunaDB	Multi-Model	0,98	+0,18	+0,62
10.	10.	↓ 9.	KairosDB	Time Series	0,56	+0,01	+0,04
11.	11.	↑ 13.	GridDB	Time Series, Multi-Model	0,48	-0,04	+0,21
12.	12.	↓ 10.	eXtremeDB	Multi-Model	0,42	-0,02	+0,01
13.	13.	↓ 12.	Amazon Timestream	Time Series	0,41	+0,01	+0,13
14.	↑ 17.	14.	IBM Db2 Event Store	Multi-Model	0,30	+0,08	+0,05
15.	↓ 14.	15.	Riak TS	Time Series	0,28	+0,02	+0,07
16.	16.	↑ 20.	Heroic	Time Series	0,27	+0,04	+0,17
17.	↓ 15.	↓ 16.	Axibase	Time Series	0,25	0,00	+0,05

Quelle: https://db-engines.com/de/ranking/time+series+dbms (28.02.2020)

InfluxDB

- InfluxData Inc. → https://www.influxdata.com
- 1.Release 2013; aktuelle Version 1.8.1 (07/2020)
- Varianten:
  - Open Source (MIT License)
  - InfluxCloud auf Amazon AWS
  - InfluxEnterprise (clusterfähig, „gehärtet“, Support...)
- Betriebssysteme: Linux, FreeBSD, Windows, macOS
- in Go programmiert
- Dokumentation → https://docs.influxdata.com/influxdb/v1.8/

InfluxDB → Installation

→ https://docs.influxdata.com/influxdb/v1.8/introduction/installation/

- Installation:
  - aus Respository der Distribution
  - oder Download → https://portal.influxdata.com/downloads/
- Programme:
  - Influxd → eigentlicher Server-Dienst
  - influx → InfluxDB Shell
- Es sollte ein Zeitdienst auf dem Rechner installiert sein!

InfluxDB → Konfiguration

→ https://docs.influxdata.com/influxdb/v1.8/administration/config/

- Konfiguration → /etc/influxdb/influxdb.conf
- Für den Anfang reicht es, ggf. den Speicherort der DB-Dateien zu konfigurieren...
- ...Restart von influxd nicht vergessen!

InfluxDB → Begrifflichkeiten

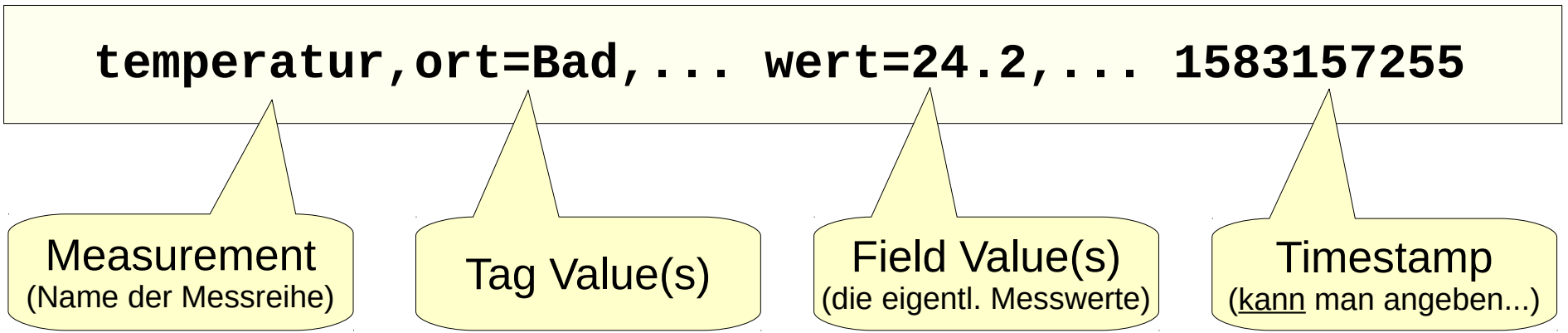
InfluxDB arbeitet mit einem schemalosen Datenmodell!

InfluxDB → Begrifflichkeiten

- *Databases* → sind Container für eine/mehrere Zeitreihen
- *Measurements* → sind die eigentlichen Zeitreihen
- ein *Datapoint* einer Zeitreihe wird beschrieben durch:
  - *Tag Values* → beschreiben/kennzeichnen die Datenpunkte
  - *Field Values* → die eigentlichen Messwerte
  - *Timestamp* → der Zeitpunkt der Messung

InfluxDB → Begrifflichkeiten

„InfluxDB-Format“:



InfluxDB → Datentypen

Datentypen müssen im Vorfeld nicht definiert werden...

- Field Values:
  - Value → String|Integer|Float|Boolean (...aber nicht gemischt!)
  - Integer und Float in 64-Bit-Auflösung
- Timestamp:
  - kleinste Auflösung in Nanosekunden
- Name der Messung, Keys, Tags etc.:
  - String

InfluxDB → Begrifflichkeiten

Wann „Tag Value“ oder „Field Value“?

- ...eine „Faustregel“:
  - Wenn man in einer relationalen DB über die Spalte einen Index legen würde → „Tag Value“
  - Alles andere → „Field Value“

InfluxDB → CLI: influx

- influx -help
- influx -host localhost -port 8086
- influx -precision rfc3339|h|m|s|ms|u|ns
- influx -database … -execute '...' -format …

```
$ influx
Connected to http://localhost:8086 version 1.7.9
InfluxDB shell version: 1.7.9
>
```

InfluxDB → Datenbank anlegen

```
> show databases
name: databases
name
----
_internal

> create database froscon

> show databases
name: databases
name
----
_internal
froscon

>
```

InfluxDB → Messwerte einfügen

```
> use froscon

> insert temperatur,ort=Bad,sensor=DHT22 value=24.4

> insert temperatur,ort=Bad value=24.4

> insert temperatur,ort=Bad,sensor=TMP36 luft=24.4,fenster=19.7

> insert temperatur,ort=Keller,sensor=TMP36 luft=24.4,fenster=19.7

> insert luftdruck,ort=BRB,sensor=BME280 value=998.4

> insert feuchtigkeit,ort=Keller,sensor=BME280 value=64.3

> insert temperatur,ort=Bad value=24.4 1597145244000000000
```

InfluxDB → API etc.

→ https://docs.influxdata.com/influxdb/v1.8/tools/api_client_libraries/

→ https://docs.influxdata.com/influxdb/v1.8/tools/api/#influxdb-1-x-http-endpoints

- die Kommunikation mit der Datenbank erfolgt grundsätzlich über HTTP(S)…, z.B. ein INSERT:

```
#!/bin/bash

while true
do
    curl -i -XPOST 'http://localhost:8086/write?db=froscon'\
         --data-binary 'random value='$((RANDOM%100000))
    sleep 1
done
```

InfluxDB → Messreihen & Serien?

```
> show measurements

name: measurements
name
----
feuchtigkeit
Luftdruck
random
temperatur
```

```
> show series

key
---
feuchtigkeit,ort=Keller,sensor=BME280
luftdruck,ort=BRB,sensor=BME280
random
temperatur,ort=Bad
temperatur,ort=Bad,sensor=DHT22
temperatur,ort=Bad,sensor=TMP36
temperatur,ort=Keller,sensor=TMP36
```

InfluxDB → Messreihen: Tags & Fields?

```
> show tag keys

name: feuchtigkeit
tagKey
------
ort
sensor

name: luftdruck
tagKey
------
ort
sensor

name: temperatur
tagKey
------
ort
sensor
```

```
> show field keys

name: feuchtigkeit
fieldKey  fieldType
--------  ---------
value     float

name: luftdruck
fieldKey  fieldType
--------  ---------
value     float

name: random
fieldKey  fieldType
--------  ---------
value          float

name: temperatur
fieldKey  fieldType
--------  ---------
fenster   float
luft      float
value     float
```

InfluxDB → Daten abfragen

→ https://docs.influxdata.com/influxdb/v1.8/query_language/data_exploration/

- FluxQL: SQL-ähnlicher Syntax
- Recht umfangreich, hier und heute „nur“:
  - einfache SELECT-Statements
  - ...WHERE...
  - ...GROUP BY...

InfluxDB → Daten abfragen

```
> select * from temperatur
name: temperatur
time                           fenster luft  ort     sensor  value
----                           ------- ----  ---     ------  -----
2020-08-11T11:27:24Z                         Bad             24.4
2020-08-11T14:38:31.990797447Z               Bad     DHT22   24.4
2020-08-11T14:38:41.883439294Z               Bad             24.4
2020-08-11T14:38:51.091224736Z 19.7    24.4  Bad     TMP36
2020-08-11T14:38:59.203358699Z 19.7    24.4  Keller  TMP36

> select value from temperatur
name: temperatur
time                           value
----                           -----
2020-08-11T11:27:24Z           24.4
2020-08-11T14:38:31.990797447Z 24.4
2020-08-11T14:38:41.883439294Z 24.4
```

InfluxDB → Daten abfragen

```
> select value, sensor from temperatur
name: temperatur
time                            value   sensor
----                            -----   ------
2020-08-11T11:27:24Z            24.4
2020-08-11T14:38:31.990797447Z  24.4    DHT22
2020-08-11T14:38:41.883439294Z  24.4

> select luft - fenster as diff from temperatur
name: temperatur
time                            diff
----                            ----
2020-08-11T14:38:51.091224736Z  4.699999999999999
2020-08-11T14:38:59.203358699Z  4.699999999999999
```

InfluxDB → Daten abfragen → where

```
> select * from temperatur where ort = 'Bad'
name: temperatur
time                           fenster luft   ort sensor  value
----                           ------- ----   --- ------  -----
2020-08-11T11:27:24Z                          Bad         24.4
2020-08-11T14:38:31.990797447Z                Bad DHT22   24.4
2020-08-11T14:38:41.883439294Z                Bad         24.4
2020-08-11T14:38:51.091224736Z 19.7    24.4   Bad TMP36

> select * from temperatur where value > 24
name: temperatur
time                           fenster luft   ort sensor  value
----                           ------- ----   --- ------  -----
2020-08-11T11:27:24Z                          Bad         24.4
2020-08-11T14:38:31.990797447Z                Bad DHT22   24.4
2020-08-11T14:38:41.883439294Z                Bad         24.4
```

InfluxDB → Daten abfragen → group by

```
> select * from temperatur group by ort
name: temperatur
tags: ort=Bad
time                            fenster luft    sensor  value
----                            ------- ----    ------  -----
2020-08-11T11:27:24Z                                    24.4
2020-08-11T14:38:31.990797447Z                  DHT22   24.4
2020-08-11T14:38:41.883439294Z                          24.4
2020-08-11T14:38:51.091224736Z  19.7    24.4    TMP36

name: temperatur
tags: ort=Keller
time                            fenster luft    sensor  value
----                            ------- ----    ------  -----
2020-08-11T14:38:59.203358699Z  19.7    24.4    TMP36
```

InfluxDB → Daten abfragen → time

```
> select * from random where time>now() - 2h

name: random
time                            value
----                            -----
2020-08-12T07:15:46.711516237Z  16941
2020-08-12T07:15:47.765404495Z  12018
2020-08-12T07:15:48.839362966Z  16233
2020-08-12T07:15:49.903382734Z  15048
2020-08-12T07:15:50.987349759Z  21219
2020-08-12T07:15:52.063080501Z  25817
2020-08-12T07:15:53.137238838Z  3966
2020-08-12T07:15:54.210646844Z  18612
2020-08-12T07:15:55.273491289Z  19323
2020-08-12T07:15:56.360338391Z  198
2020-08-12T07:15:57.422481544Z  29246
2020-08-12T07:15:58.485879266Z  31403
2020-08-12T07:15:59.560000237Z  6067
2020-08-12T07:16:00.632815177Z  2227
2020-08-12T07:16:02.577825396Z  17102
2020-08-12T07:16:03.635433838Z  8738
2020-08-12T07:16:04.711278358Z  10212
```

InfluxDB → Daten abfragen → time()

```
> select * from random where time>now() - 2h

name: random
time                            value
----                            -----
2020-08-12T07:15:46.711516237Z  16941
2020-08-12T07:15:47.765404495Z  12018
2020-08-12T07:15:48.839362966Z  16233
2020-08-12T07:15:49.903382734Z  15048
2020-08-12T07:15:50.987349759Z  21219
2020-08-12T07:15:52.063080501Z  25817
2020-08-12T07:15:53.137238838Z  3966
2020-08-12T07:15:54.210646844Z  18612
2020-08-12T07:15:55.273491289Z  19323
2020-08-12T07:15:56.360338391Z  198
2020-08-12T07:15:57.422481544Z  29246
2020-08-12T07:15:58.485879266Z  31403
2020-08-12T07:15:59.560000237Z  6067
2020-08-12T07:16:00.632815177Z  2227
2020-08-12T07:16:02.577825396Z  17102
2020-08-12T07:16:03.635433838Z  8738
2020-08-12T07:16:04.711278358Z  10212
```

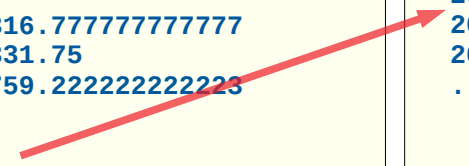
```
> select mean(*) from random where time>now() - 2h
  group by time(10s)

name: random
time                    mean_value
----                    ----------
2020-08-12T07:09:50Z
2020-08-12T07:10:00Z
...
2020-08-12T07:15:50Z    17316.777777777777
2020-08-12T07:16:00Z    13331.75
2020-08-12T07:16:10Z    18759.222222222223
2020-08-12T07:16:20Z
2020-08-12T07:16:30Z
2020-08-12T07:16:40Z
2020-08-12T07:16:50Z
2020-08-12T07:17:00Z    2350
2020-08-12T07:17:10Z    13584
...
2020-08-12T09:09:30Z
2020-08-12T09:09:40Z
2020-08-12T09:09:50Z
...
```

InfluxDB → Daten abfragen → fill()

```
> select mean(*) from random where time>now() - 2h
  group by time(10s) fill(0)

name: random
time                   mean_value
----                   ----------
2020-08-12T07:09:50Z   0
2020-08-12T07:10:00Z   0
...
2020-08-12T07:15:50Z   17316.777777777777
2020-08-12T07:16:00Z   13331.75
2020-08-12T07:16:10Z   18759.222222222223
2020-08-12T07:16:20Z   0
2020-08-12T07:16:30Z   0
2020-08-12T07:16:40Z   0
2020-08-12T07:16:50Z   0
2020-08-12T07:17:00Z   2350
2020-08-12T07:17:10Z   13584
...
2020-08-12T09:09:30Z   0
2020-08-12T09:09:40Z   0
2020-08-12T09:09:50Z   0
...
```



```
> select mean(*) from random where time>now() - 2h
  group by time(10s) fill(none)

name: random
time                   mean_value
----                   ----------
2020-08-12T07:15:50Z   17316.777777777777
2020-08-12T07:16:00Z   13331.75
2020-08-12T07:16:10Z   18759.222222222223
2020-08-12T07:17:00Z   2350
2020-08-12T07:17:10Z   13584
...
```

InfluxDB → Daten abfragen (Funktionen)

→ https://docs.influxdata.com/influxdb/v1.8/query_language/functions/

- Aggregat-Funktionen (count(), mean(), sum(), spread()...)
- Selektoren (max(), first(), top(), ...)
- Mathematische Funktionen (sin(), round(), difference(), moving_average(), ...)
- Analyse-Funktionen(...)

→ Abgrenzung immer mit GROUP BY (über Tag Value oder Zeitbereich)...

InfluxDB → Datenpunkte manipulieren?

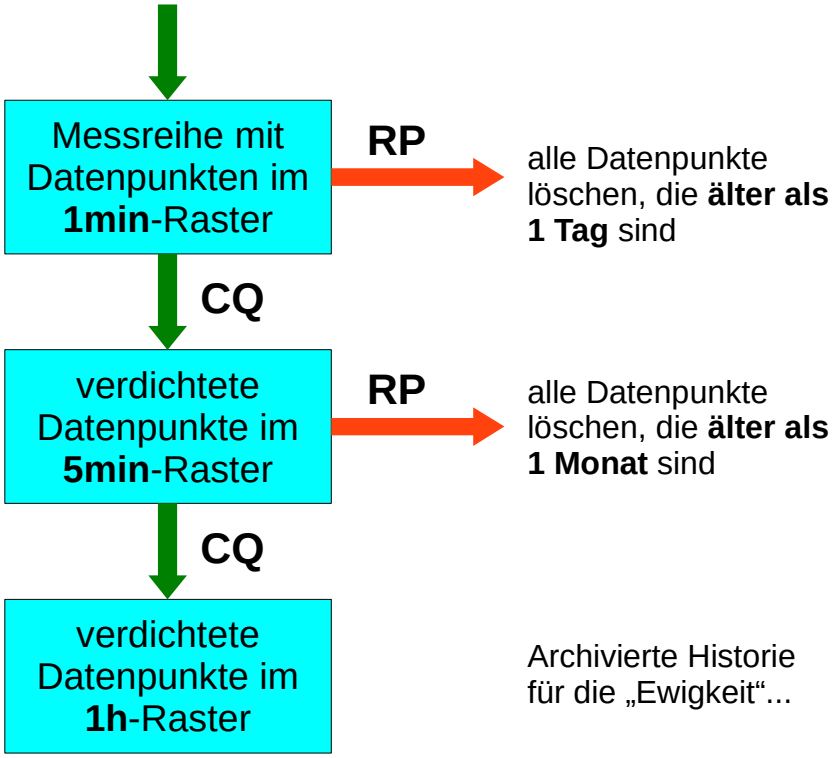
→ https://docs.influxdata.com/influxdb/v1.8/query_language/database_management/

- kein UPDATE
- es gibt DELETE FROM <measurement> WHERE …
  - akzeptiert keine Field Values im WHERE
- Messung insgesamt löschen via DROP …

InfluxDB → nützliche „Automatismen“

Stichworte:

- Continuous Queries (CQ)
- Retention Policies (RP)



InfluxDB → Continuous Queries

→ https://docs.influxdata.com/influxdb/v1.8/query_language/continuous_queries/

- Automatisches Zusammenfassen/Weiterverarbeiten von Messungen in eine neue Messung

```
> create continuous query "random_1m" on "clt"
begin
select mean(value) into random_1m from random group by time(1m)
end

> show continuoues queries
...
name: froscon
name      query
----      -----
random_1m CREATE CONTINUOUS QUERY random_1m ON clt BEGIN SELECT ... END
```

InfluxDB → Continuous Queries

```
create continuous query "random_1m" on "clt"
begin
select mean(value), count(value) into random_1m from random group by time(1m)
end
```

```
> select * from random
2020-07-22T14:31:02Z 50
2020-07-22T14:31:22Z 92
2020-07-22T14:31:42Z 69
2020-07-22T14:32:02Z 54
2020-07-22T14:32:22Z 70
2020-07-22T14:32:42Z 22
2020-07-22T14:33:02Z 56
2020-07-22T14:33:22Z 65
2020-07-22T14:33:42Z 62
2020-07-22T14:34:02Z  5
2020-07-22T14:34:22Z 51
2020-07-22T14:34:42Z 67
```

```
> select * from random_1m
2020-07-22T14:31:00Z 3     70.3333333
2020-07-22T14:32:00Z 3     48.6666666
2020-07-22T14:33:00Z 3     61
2020-07-22T14:34:00Z 3     41
```

InfuxDB → Retention Policies

→ https://docs.influxdata.com/influxdb/v1.8/query_language/database_management/#retention-policy-management

- „Haltbarkeitsdatum“ von Datensätze festlegen

```
> create retention policy "del_old" on "froscon" duration 1d replication 1

> show retention policies
name    duration shardGroupDuration replicaN default
----    -------- ------------------ -------- -------
autogen 0s       168h0m0s           1        true
del_old 24h0m0s  1h0m0s             1        false

> insert into del_old.luftdruck,ort=BRB,sensor=BME280 value=1000.3

> select * from del_old.luftdruck
```

InfluxDB → Datenimport

- ...wenn man asynchron Daten in eine InfluxDB schreiben will...:

```
$ curl -i -XPOST 'http://localhost:8086/write?db=wetter'
--data-binary @daten.txt
```

```
myweather,type=light adc0=234,adc1=156,tsl45315=1612 1578221982000000000
myweather,type=light adc0=234,adc1=156,tsl45315=1608 1578222043000000000
myweather,type=light adc0=235,adc1=157,tsl45315=1614 1578222103000000000
myweather,type=light adc0=235,adc1=157,tsl45315=1614 1578222163000000000
myweather,type=light adc0=234,adc1=156,tsl45315=1612 1578222224000000000
myweather,type=light adc0=234,adc1=156,tsl45315=1624 1578222284000000000
myweather,type=light adc0=235,adc1=157,tsl45315=1660 1578222345000000000
myweather,type=light adc0=236,adc1=158,tsl45315=1712 1578222405000000000
myweather,type=light adc0=237,adc1=159,tsl45315=1762 1578222466000000000
myweather,type=light adc0=237,adc1=159,tsl45315=1818 1578222526000000000
myweather,type=light adc0=236,adc1=159,tsl45315=1868 1578222586000000000
```

InfluxDB → was noch?

Mehr ist nicht, außer vielleicht noch...:

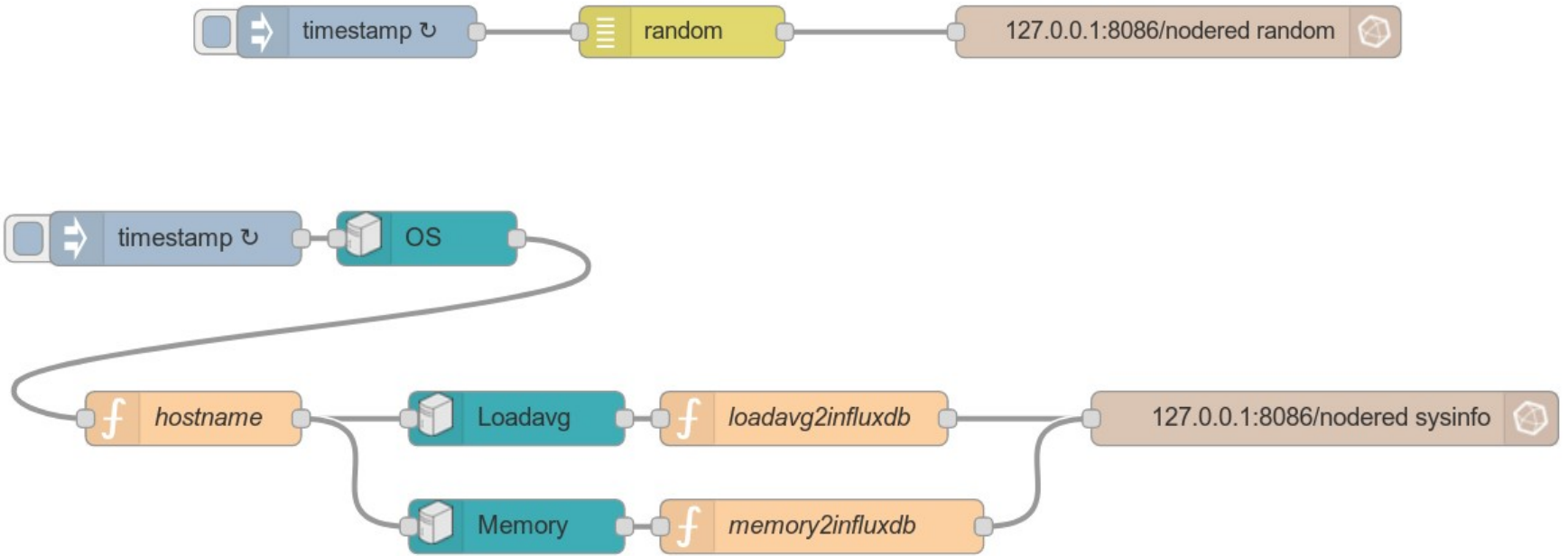
- Viele weitere Abfrageoptionen/-parameter/-varianten
- Datenbank-Backup
- Usermanagment
- InfluxDB-Interna
- ...?

→ **RTFM**

InfluxDB → und nun?

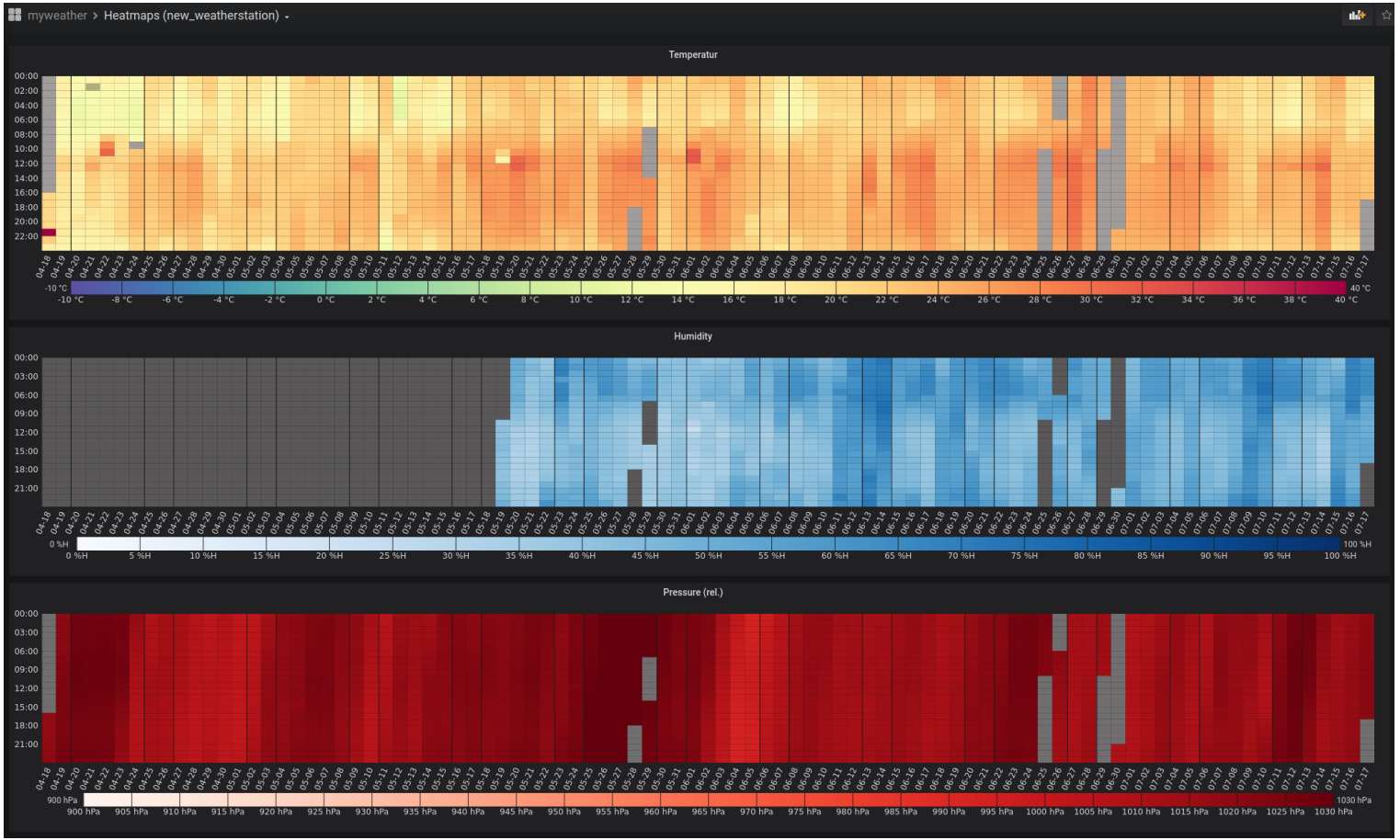
- Weitere Produkte von InfluxData Inc.:
  - Telegraf: Einsammeln von Zeitreihendaten
  - Chronograf: Visualisierung von Zeitreihendaten
  - Kapacitor: Verarbeitung von Zeitreihendaten
- Grafana: Visualisierung von Zeitreihendaten
- NodeRed: kann Zeitreihendaten in eine InfluxDB schreiben bzw. wieder auslesen...
- ...

NodeRed → InfluxDB

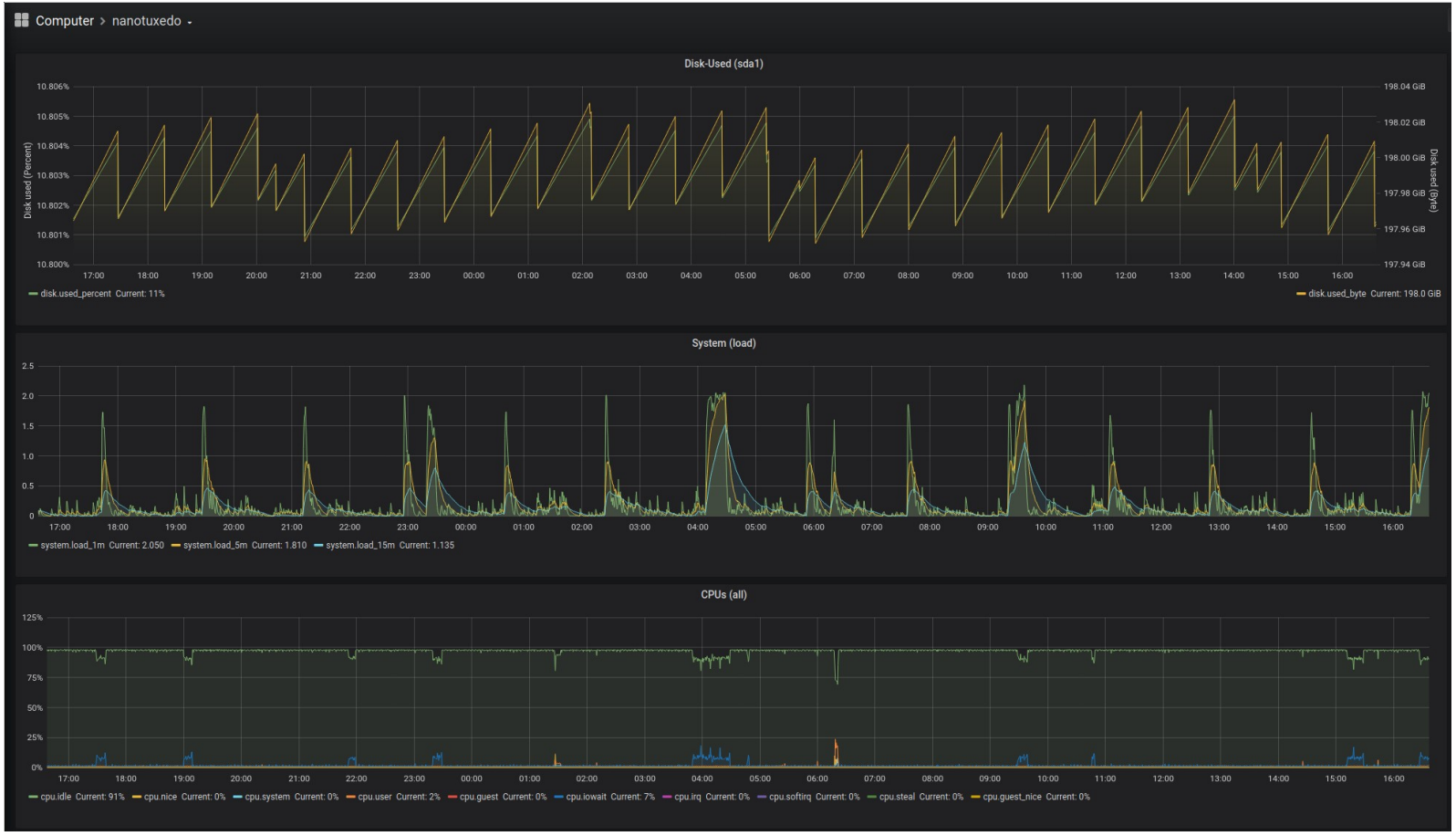


InfluxDB → Grafana

InfluxDB → Grafana



Telegraf → InfluxDB → Grafana



Weiterführende Informationsquellen

InfluxDB
→ https://www.influxdata.com/products/influxdb-overview/
→ https://docs.influxdata.com/influxdb/v1.8/

Telegraf
→ https://www.influxdata.com/time-series-platform/telegraf/
→ https://docs.influxdata.com/telegraf/v1.15/

NodeRed
→ https://nodered.org/
→ https://programm.froscon.de/2018/events/2209.html

Grafana
→ https://grafana.com/

Fragen?

...ansonsten Danke & Ende!

Zusatzfolien...

InfluxDB → API?

→ https://docs.influxdata.com/influxdb/v1.8/tools/api_client_libraries/

- die Kommunikation mit der Datenbank erfolgt grundsätzlich über HTTP(S)...
- ...deshalb geht z.B. auch dies...:

```
$ curl -i -XPOST http://localhost:8086/query
--data-urlencode "q=CREATE DATABASE clt"

$ curl -i -XPOST 'http://localhost:8086/write?db=clt'
--data-binary 'temperatur,ort=Garage value=15.7'

$ curl -G 'http://localhost:8086/query?pretty=true'
--data-urlencode "db=clt" --data-urlencode
"q=SELECT * FROM temperature"
```

Grafana

- Open Source (Lizenz: Apache 2.0)
- Analyse und Visualisierung von Daten
- Datenquelle u.a. auch InfluxDBs
- zahlreiche Plugins
  → https://grafana.com/grafana/plugins

Telegraf

- InfluxData Inc.
  → https://www.influxdata.com/time-series-platform/telegraf/
- Dienst zum Sammeln und Versenden von Daten...
- ...auch in eine InfluxDB...
- In Go geschrieben, keine externen Abhängigkeiten, ressourcenschonend
- Zahlreiche Plugins
  → https://docs.influxdata.com/telegraf/v1.15/plugins/plugin-list/