Part 3 - Sensordaten in Datenbank speichern

Bislang haben wir in den vorherigen Tutorials die Daten der Sensoren jeweils in Logfiles lokal auf dem Raspberry Pi gespeichert bzw. über Alarmmeldungen mit Telegram versendet. In diesem Tutorial geht es nun darum diese Sensordaten zentral in einer Datenbank zu speichern um sie später bequem und flexibel grafisch auswerten zu können.

Inhaltsverzeichnis

Einleitung

Für diesen Zweck verwenden wir die Datenbank InfluxDB, eine OpenSource Time Series Datenbank. Man kann selbstverständlich auch die renommierten Datenbanken wie MySQL oder PostgreSQL verwenden, InfluxDB wurde aber speziell für hohe Schreibzugriffe und Abfragen entwickelt und eignet sich besonders für wiederkehrende Schreibzugriffe wie sie bei Sensordaten entstehen. Zusätzlich bietet InfluxDB eine SQL-ähnliche Sprache für den Zugriff auf die Daten und steht unter einer OpenSource Lizenz wodurch keine zusätzlichen Lizenzgebühren anfallen. Der Hersteller selbst bietet auch ein Cloud Modell sowie eine kostenpflichtige Enterprise Version der Datenbank an, wir verwenden hier aber die On-Premise Datenbank auf einem separaten System.

Voraussetzungen

Da durch den Einsatz von InfluxDB oder generell von Datenbanken sehr schnell hohe Schreibzugriffe auf dem Dateisystem entstehen können verwenden wir für InfluxDB ein separates System das mit einer SSD ausgestattet ist. Dafür eignet sich hervorragend ein Intel NUC der ausreichend Leistung bietet und kostengünstig erhältlich ist. Man kann natürlich auch den Raspberry Pi dafür verwenden allerdings würde bei hohen Schreibzugriffen die Lebensdauer der SD-Karte deutlich sinken. Für den dauerhaften Einsatz im Serverraum ist der Intel NUC wie auch der Raspberry Pi in unserem Shop auch mit 19" Gehäuse erhältlich.

Als Betriebssystem kommt auf dem Intel NUC Ubuntu 20.04 (Focal Fossa) in 64-Bit zum Einsatz. Dieses bietet als sogenannte LTS Version eine Update und Support Garantie des Betriebssystem bis April 2025. Da auf dem Ubuntu System lediglich InfluxDB und später für die grafische Auswertung Grafana benötigt werden ist eine Minimal Installation absolut ausreichend.

Installation InfluxDB

Die eigentlich Installation von InfluxDB ist sehr schnell erledigt, jedoch ist die in den offiziellen Ubuntu Repositorys vorhandene InfluxDB Version etwas veraltet, daher verwenden wir direkt die Paketquellen des InfluxDB Projektes.

Im ersten Schritt bringen wir das Ubuntu System auf den aktuellen Stand und installieren zusätzlich GnuPG für den öffentlichen Schlüssel des InfluxDB Projekts.

```
sudo apt update
sudo apt upgrade
sudo apt install gnupg2
```

Nun wird der Key des InfluxDB Projektes als vertrauenswürdige Quelle hinzugefügt.

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
```

Ein abschließendes "OK" zeigt das der Key erfolgreich hinzugefügt wurde. Nun wird im nächsten Schritt das Repository von InfluxDB zu den Paketquellen hinzugefügt.

```
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,} ${DISTRIB_CODENAME} stable" | sudo tee /etc/apt
/sources.list.d/influxdb.list
```

Der zweite Befehl legt die Datei influxdb.list unter /etc/apt/sources.list.d an die auf unserem System folgenden Inhalt hat:

```
jkm@nuc:~$ cat /etc/apt/sources.list.d/influxdb.list
deb https://repos.influxdata.com/ubuntu focal stable
```

Nun erfolgt die Aktualisierung der Paketquellen und die eigentliche Installation von InfluxDB

```
sudo apt update
sudo apt install influxdb
```

Abschließend fügen wir InfluxDB noch als Dienst hinzu damit bei jedem Neustart InfluxDB automatisch gestartet wird.

```
sudo systemctl unmask influxdb.service
sudo systemctl start influxdb
sudo systemctl enable influxdb
```


Konfiguration InfluxDB

Der Datenbankserver ist soweit erst einmal einsatzbereit, allerdings sind Zugriffe momentan auch ohne Passwort möglich. Auch ist der Zugriff nur über HTTP und nicht gesichert über HTTPS. Unverschlüsselte Zugriffe in internen Netzen oder für erste Tests sind möglich, für produktive Umgebungen oder gar bei Zugriffen über das Internet sollte zwingend die Verschlüsslung verwendet werden. Weiter unten im Tutorial zeigen wir wie dies mit selbst erstellten Zertifikaten und auch mit offiziellen Zertifikaten umgesetzt werden kann. Für größere Installationen lohnt es sich dann in ein Wildcard Zertifikat zu investieren.

Für die erste Absicherung aktivieren wir nun die Benutzerauthentifizierung. Dazu legen wir eine Kopie der Original Konfigurationsdatei an und editieren die Konfigurationsdatei mit einem Editor nach Wahl.

```
sudo cp /etc/influxdb/influxdb.conf /etc/influxdb/influxdb.conf.orig
sudo vi /etc/influxdb/influxdb.conf
```

Dort befindet sich in der Sektion [http] der folgende Eintrag:

```
  # Determines whether user authentication is enabled over HTTP/HTTPS.
  # auth-enabled = false
```

Diesen ändern wir wie folgt und starten anschließend die InfluxDB neu damit die Änderungen aktiv werden.

```
  # Determines whether user authentication is enabled over HTTP/HTTPS.
auth-enabled = true
```

```
sudo systemctl restart influxdb
```

Dadurch sind die Zugriffe über HTTP und später auch HTTPS nur noch mit Passwort möglich.

Datenbank erstellen

Wie bereits erwähnt besitzt InfluxDB eine SQL ähnliche Syntax und kann lokal über den "influx" Client, ähnlich wie mysql für MySQL oder pgsql für PostgreSQL, administriert werden. Im ersten Schritt müssen wir einen "admin" User mit Passwort anlegen da in der Konfiguration die Authentifizierung aktiviert worden ist. Sollte dieser noch nicht angelegt sein bekommt man aber auch von InfluxDB einen netten Hinweis 🙂

```
jkm@nuc:~$ influx
Connected to http://localhost:8086 version 1.8.2
InfluxDB shell version: 1.8.2
> CREATE DATABASE "temperature"
ERR: error authorizing query: create admin user first or disable authentication
Warning: It is possible this error is due to not setting a database.
Please set a database with the command "use <database>".
>
```

Also im ersten Schritt einen Admin Benutzer mit Kennwort anlegen:

```
> CREATE USER admin WITH PASSWORD 'influxadmin' WITH ALL PRIVILEGES
```

Mit der gleichen Syntax können nun auch weitere administrative Benutzer angelegt werden.

Nun müssen wir uns an InfluxDB mit dem neu angelegte Admin User authentifizieren:

```
> AUTH
username: admin
password:
>
```

Im nächsten Schritt legen wir eine Datenbank "temperature" an, einen neuen Benutzer "rpi-01" mit Vollzugriff auf diese Datenbank für unseren Raspberry Pi sowie einen Benutzer "grafana" mit lediglich Lese-Rechten. Dieser dient später für die grafische Darstellung der Temperaturen. Zusätzlich geben wir dem eben angelegten Admin ebenfalls Vollzugriff auf die Datenbank.

```
> CREATE DATABASE "temperature"
> CREATE USER "rpi-01" WITH PASSWORD 'rpi-01'
> CREATE USER "grafana" WITH PASSWORD 'grafana'
> GRANT ALL ON "temperature" TO "rpi-01"
> GRANT ALL ON "temperature" TO "admin"
> GRANT READ ON "temperature" TO "grafana"
```

Wichtig ist hierbei das der Username in Anführungszeichen gesetzt wird da er das Sonderzeichen "-" enthält. Bei dem Benutzer grafana wären diese nicht notwendig gewesen, wurde aber der Übersichtlichkeit halber auch in Anführungszeichen gesetzt. Das Kennwort hingegen muss in einfache Hochkomma gesetzt werden.

Für einen produktive Betrieb sollten stets nur sichere Kennwörter verwendet werden!

Nun können wir uns die Datenbank(en) und die angelegten Benutzer nochmals anschauen:

```
> SHOW DATABASES
name: databases
name
----
_internal
temperature
>
> SHOW USERS
user    admin
----    -----
admin   true
rpi-01  false
grafana false
>
> QUIT
```

SSL

Bei einem produktiven Einsatz, insbesondere wenn der Austausch mit der Datenbank über das Internet erfolgt, sollte neben der Authentifizierung mit Benutzername und Passwort auch eine SSL Verschlüsslung vorgenommen werden. Dadurch wird der komplette Traffic verschlüsselt und kann von Dritten nicht mit geschnitten oder ausgelesen werden.

Bei den SSL Zertifikaten gibt es nun zwei Möglichkeiten. Zum einen kann man diese selbst erstellen oder aber offiziell signierte Zertifikate bei einem Hersteller ordern. Letztere sind in der Regel kostenpflichtig, eine Ausnahme bildet die Let's Encrypt Initiative. In diesem Tutorial gehen wir nur auf selbst signierte Zertifikate ein.

Selbst signiertes Zertifikat erzeugen

Das Erstellen eines selbst signierten Zertifikats unter Linux ist mit lediglich einem Befehl erledigt. Dazu führen wir folgenden Befehl auf unserem Intel NUC aus:

```
jkm@nuc:~$ openssl req -x509 -nodes -newkey rsa:2048 -keyout influxdb-selfsigned.key -out influxdb-
selfsigned.crt -days
365
Generating a RSA private key
.....................................................................................................
.................................+++++
.....................................................................................................+++++
writing new private key to 'influxdb-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Rheinland-Pfalz
Locality Name (eg, city) []:Diez
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ICO GmbH
Organizational Unit Name (eg, section) []:IoT
Common Name (e.g. server FQDN or YOUR name) []:nuc.iot.ico.de
Email Address []:it@ico.de
jkm@nuc:~$ ls -l
total 8
-rw-rw-r-- 1 jkm jkm 1436 Sep 20 14:55 influxdb-selfsigned.crt
-rw------- 1 jkm jkm 1704 Sep 20 14:54 influxdb-selfsigned.key
```

Der SSL Befehl erzeugt einen privaten Key sowie das damit signierte Zertifikat mit einer Laufzeit von einem Jahr (365 Tage). Die notwendigen Werte wie Country Name, State, etc. werden dabei interaktiv abgefragt. Die private Schlüsseldatei "influxdb-selfsigned.key" sollte unbedingt zusätzlich an einem sicheren Ort aufbewahrt werden da ohne sie das Zertifikat nicht mehr zu gebrauchen ist.

Nun können das Zertifikat und der Key an die passende Stelle im Dateisystem abgelegt werden:

```
jkm@nuc:~$ sudo cp influxdb-selfsigned.* /etc/ssl/
jkm@nuc:~$ sudo chown influxdb:influxdb /etc/ssl/influxdb-selfsigned.*
jkm@nuc:~$ sudo chmod 644 /etc/ssl/influxdb-selfsigned.crt
jkm@nuc:~$ sudo chmod 600 /etc/ssl/influxdb-selfsigned.key
jkm@nuc:~$ sudo ls -l /etc/ssl/
total 40
drwxr-xr-x 2 root     root     16384 Sep 20 16:46 certs
-rw-r--r-- 1 influxdb influxdb  1436 Sep 20 18:41 influxdb-selfsigned.crt
-rw------- 1 influxdb influxdb  1704 Sep 20 18:41 influxdb-selfsigned.key
-rw-r--r-- 1 root     root     10909 Apr 20 13:53 openssl.cnf
drwx------ 2 root     root      4096 Sep 20 15:03 private
```

Änderungen in der influxdb.conf

Nach dem Erzeugen der Zertifikate sind noch folgende Änderungen in der Konfigurationsdatei /etc/influxdb/influxdb.conf notwendig:

Alt

```
  # Determines whether HTTPS is enabled.
  # https-enabled = false

  # The SSL certificate to use when HTTPS is enabled.
  # https-certificate = "/etc/ssl/influxdb.pem"

  # Use a separate private key location.
  # https-private-key = ""
```

Muss geändert werden nach:

Neu

```
  # Determines whether HTTPS is enabled.
https-enabled = true

  # The SSL certificate to use when HTTPS is enabled.
https-certificate = "/etc/ssl/influxdb-selfsigned.crt"

  # Use a separate private key location.
https-private-key = "/etc/ssl/influxdb-selfsigned.key"
```

Aufgrund dieser Änderung ist ein Neustart der InfluxDB notwendig.

```
sudo systemctl restart influxdb
```

Testen des SSL Zugriffs

Um zu Testen ob der Zugriff nun auch über SSL funktioniert starten wir erstmal eine normale HTTP Verbindung zu dem Server. Erwartungsgemäß wird die Anfrage aber vom Server abgelehnt da wir nun eine HTTP Anfrage senden, erwartet wird aber HTTPS:

```
jkm@nuc:~$ influx
WARN: Connected to http://localhost:8086, but found no server version.
Are you sure an InfluxDB server is listening at the given address?
InfluxDB shell version: 1.8.2
> auth
username: admin
password:
> show databases
ERR: Client sent an HTTP request to an HTTPS server.
> quit
```

Nun starten wir den Client mit der Option -ssl, der Zugriff wird aber direkt abgelehnt da der Name in dem Zertifikat nicht mit dem Hostnamen übereinstimmt:

```
jkm@nuc:~$ influx -ssl
Failed to connect to https://localhost:8086: Get https://localhost:8086/ping: x509: certificate is valid for
nuc.iot.ico.de, not localhost
You may use -unsafeSsl to connect anyway, but the SSL connection will not be secure.
jkm@nuc:~$
```

Nun könnte man in der Datei /etc/hosts den Hostnamen des Zertifikats als zusätzlichen localhost hinzufügen. Dies würde bedeuten dass wir den Namen unseres NUCs auf den gleichen Namen des Zertifikats gesetzt haben.

/etc/hosts

```
127.0.0.1        nuc.iot.ico.de
```

Ein kurze Überprüfung mit nslookup zeigt uns das der Namen nun auf unseren NUC zeigt:

```
jkm@nuc:~$ nslookup nuc.iot.ico.de
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   nuc.iot.ico.de
Address: 127.0.0.1
```

Eine erneute Verbindung mit dem Client und der Option SSL, diesmal direkt auf den Hostnamen, schlägt aber erneut fehl da wir ja ein selbst signiertes Zertifikat verwenden und es nicht von einer offiziellen Stelle validiert wurde:

```
jkm@nuc:~$ influx -ssl -host nuc.iot.ico.de
Failed to connect to https://nuc.iot.ico.de:8086: Get https://nuc.iot.ico.de:8086/ping: x509: certificate
signed by unknown authority
Please check your connection settings and ensure 'influxd' is running.
```

Um dies zu umgehen gibt es nun zwei Möglichkeiten. Entweder wir verwenden die Option -unsafeSsl die eine Überprüfung der Autorität überspringt oder aber wir fügen das Zertifikat zu den Vertrauenswürdigen Zertifikaten hinzu.

Zuerst die Verbindung mit -unsafeSsl:

```
jkm@nuc:~$ influx -ssl -unsafeSsl -host nuc.iot.ico.de
Connected to https://nuc.iot.ico.de:8086 version 1.8.2
InfluxDB shell version: 1.8.2
> AUTH
username: admin
password:
> SHOW DATABASES
name: databases
name
----
_internal
temperature
> QUIT
```

Dies hat funktioniert, wir können nun über HTTPS die Anfragen an die InfluxDB senden.

Oder aber wir fügen das Zertifikat als Vertrauenswürdig hinzu, danach ist dann die Option -unsafeSsl nicht mehr notwendig. Dazu legen wir ein neues Unterverzeichnis im Zertifikatsspeicher an, kopieren das Zertifikat dorthin und führen ein Update des Zertifikatsspeichers durch. Anschließend ist ein direkter Zugriff mit SSL möglich:

```
jkm@nuc:~$ sudo mkdir /usr/local/share/ca-certificates/influxdb
jkm@nuc:~$ sudo cp /etc/ssl/private/influxdb-selfsigned.crt /usr/local/share/ca-certificates
/influxdb/

jkm@nuc:~$ sudo chmod 644 /usr/local/share/ca-certificates/influxdb/influxdb-selfsigned.
crt

jkm@nuc:~$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
jkm@nuc:~$ influx -ssl -host nuc.iot.ico.de
Connected to https://nuc.iot.ico.de:8086 version 1.8.2
InfluxDB shell version: 1.8.2
> AUTH
username: admin
password:
> SHOW DATABASES
name: databases
name
----
_internal
temperature
> QUIT
```

Das Zertifikat muss zusätzlich auf allen beteiligten Clients, die Zugriff auf die InfluxDB bekommen sollen, auf diese Weise installiert werden. Auch nach dem Ablauf des Zertifikats nach 365 Tagen muss dann das neue Zertifikat ebenfalls wieder auf allen Clients ausgetauscht werden.

Für Testzwecke ist dies aber absolut ausreichend, für produktive Umgebungen empfiehlt es sich aber ein offizielles Zertifikat zu verwenden da diese Problematik damit nicht mehr auftritt.

Zusammenfassung

In diesem Tutorial haben wir die Time Series Datenbank InfluxDB auf einem Intel NUC installiert und eingerichtet. Ebenfalls wurde eine erste Datenbank angelegt und Rechte an verschiedene Benutzer vergeben. Im nächsten Tutorial füttern wir dann die Datenbank vom Raspberry Pi aus mit den Sensordaten und richten Grafana auf dem Intel NUC ein damit diese übersichtlich dargestellt werden können.