

Dokumentation: MQTT-Anwendung mit Docker

Einführung

In diesem Projekt habe ich eine MQTT-basierte IoT-Anwendung mit Docker umgesetzt. Die Anforderung war, einen MQTT-Broker, einen Publisher und mehrere Sensoren in einer Docker-Umgebung zu verbinden und zum Laufen zu bringen. Ziel war es, Sensordaten zu erfassen, zu speichern und zu visualisieren.

Systemarchitektur

Meine Implementierung besteht aus folgenden Komponenten:

- **MQTT-Broker (Eclipse Mosquitto)**: Zentraler Nachrichtenverteiler
- **Publisher**: Sendet Testnachrichten an den Broker
- **Temperatursensoren (2)**: Simulieren Temperaturmessungen
- **Feuchtigkeitssensor**: Simuliert Feuchtigkeitsmessungen
- **InfluxDB**: Zeitreihendatenbank zur Speicherung der Sensorwerte
- **InfluxDB-Connector**: Verbindet MQTT mit InfluxDB
- **Grafana**: Visualisiert die gespeicherten Daten

Der Datenfluss ist wie folgt: Die Sensoren senden ihre Messwerte an den MQTT-Broker, der InfluxDB-Connector empfängt diese Nachrichten und schreibt sie in die InfluxDB-Datenbank. Grafana greift auf diese Datenbank zu und stellt die Daten visuell dar.

05C: Umsetzung der Aufgaben

1. Erstellung der Docker-Image-Dateien

Für jede Komponente des Systems habe ich ein eigenes Dockerfile erstellt:

- Für den Broker habe ich ein einfaches Dockerfile auf Basis des offiziellen Eclipse Mosquitto Images erstellt und eine angepasste Konfigurationsdatei hinzugefügt.
- Für die Java-basierten Komponenten (Publisher, Sensoren, InfluxDB-Connector) habe ich ein mehrstufiges Build-Verfahren verwendet:
 - In der ersten Stufe wird Maven genutzt, um die Java-Anwendung zu kompilieren
 - In der zweiten Stufe wird eine schlanke JRE-Umgebung genutzt, um die kompilierte Anwendung auszuführen
- Für Grafana und InfluxDB habe ich die offiziellen Images verwendet und entsprechend konfiguriert

2. Erstellung der Docker-Compose-Datei

In der Docker-Compose-Datei habe ich alle Komponenten definiert und ihre Abhängigkeiten, Netzwerke, Volumes und Umgebungsvariablen konfiguriert. Besonders wichtig war die

Konfiguration der Kommunikation zwischen den Containern. Zum Beispiel musste ich sicherstellen, dass die Sensoren den Broker unter dem richtigen Hostnamen erreichen können.

3. Ausführung des Docker-Compose-Befehls

Docker-Stack bauen und starten:

```
# Docker-Compose ausführen
```

```
docker-compose build
```

```
docker-compose up -d
```

```
# Status der Container überprüfen
```

```
docker ps
```

Bei der Ausführung von `docker-compose up -d` bin ich auf einige Probleme gestossen:

```
321 (Sicherungspunkt 5) [wird ausgeführt] - Oracle VirtualBox
ties  Terminal Apr 13 17:18
test@test: ~/mqtt-docker

✓ Container mqtt-docker-temperature-sensor-1 Removed 0.1s
✓ Container mqtt-docker-influxdb-1 Removed 0.4s
✓ Container mqtt-docker-broker-1 Removed 0.6s
✓ Network mqtt-docker_mqtt-network Removed 0.2s
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 2/2
✓ Network mqtt-docker_mqtt-network Created 0.3s
✓ Container mqtt-docker-broker-1 S... 0.7s
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 1/1
✓ Container mqtt-docker-influxdb-1 Started 0.6s
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 4/4
✓ Container mqtt-docker-influxdb-1 Running 0.8s
✓ Container mqtt-docker-grafana-1 Started 1.2s
✓ Container mqtt-docker-broker-1 Running 0.8s
✓ Container mqtt-docker-influxdb-connector-1 Started 1.0s
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 5/5
✓ Container mqtt-docker-broker-1 Running 0.8s
✓ Container mqtt-docker-humidity-sensor-1 Started 4.1s
✓ Container mqtt-docker-publisher-1 Started 4.7s
✓ Container mqtt-docker-temperature-sensor-1 Started 4.4s
✓ Container mqtt-docker-temperature-sensor-2-1 Started 4.9s
```

- Die Ports 1883 (MQTT), 8086 (InfluxDB) und 3000 (Grafana) waren bereits auf meinem System belegt
- Es gab Probleme mit den Java-JAR-Dateien aufgrund von Signaturproblemen

Nach der Lösung dieser Probleme konnte ich alle Container erfolgreich starten. Im Screenshot ist zu sehen, dass alle Container gestartet wurden und laufen.

```
test@test:~$ sudo docker ps
[sudo] password for test:
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
da978855c0d5   mqtt-docker-influxdb-connector      "java -jar influxdb-..." 9 minutes ago  Up 9 minutes
5a994ed08a55   grafana/grafana:latest             "/run.sh"                20 minutes ago Up 18 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
3bb7e386a6ae   influxdb:1.8                       "/entrypoint.sh infl..." 20 minutes ago Up 20 minutes   0.0.0.0:8086->8086/tcp, :::8086->8086/tcp
bef252bb6b7a   mqtt-docker-publisher              "java -jar publisher-..." 20 minutes ago Restarting (1) 29 seconds ago
e935d1516741   mqtt-docker-temperature-sensor-2    "java -jar temperatu..." 20 minutes ago Restarting (1) 29 seconds ago
47ec5770f0b3   mqtt-docker-temperature-sensor-2-1  "java -jar temperatu..." 22 minutes ago Up 21 minutes
c45089f3c78d   mqtt-docker-humidity-sensor-1      "java -jar humidity-..." 22 minutes ago Up 21 minutes
a40d8d7209b4   mqtt-docker-broker-1              "/docker-entrypoint...." 22 minutes ago Up 22 minutes   0.0.0.0:1883->1883/tcp, :::1883->1883/tcp, 0.0.0.0:9001->9001/tcp, :::9001->9001/tcp
test@test:~$
```

4. Funktionsüberprüfung der Komponenten

Ich habe mehrere Tests durchgeführt, um die korrekte Funktion aller Komponenten zu überprüfen:

1. **Container-Status:** Mit `sudo docker ps` habe ich überprüft, dass alle Container laufen
2. **Sensordaten:** Mit `sudo docker-compose logs --tail=10 temperature-sensor` und entsprechenden Befehlen für die anderen Sensoren habe ich verifiziert, dass die Sensoren Daten generieren und senden.

```
test@test:~/mqtt-docker$ sudo docker-compose logs --tail=10 temperature-sensor
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
temperature-sensor-1 | Publishing temperature: 28.8°C
temperature-sensor-1 | Publishing temperature: 26.3°C
temperature-sensor-1 | Publishing temperature: 26.1°C
temperature-sensor-1 | Publishing temperature: 18.0°C
temperature-sensor-1 | Publishing temperature: 19.3°C
temperature-sensor-1 | Publishing temperature: 30.0°C
temperature-sensor-1 | Publishing temperature: 22.7°C
temperature-sensor-1 | Publishing temperature: 22.1°C
temperature-sensor-1 | Publishing temperature: 19.1°C
temperature-sensor-1 | Publishing temperature: 19.2°C
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
temperature-sensor-2-1 | at java.base/jdk.internal.loader.BuiltinClassLoader.defineClass(Unknown Source)
temperature-sensor-2-1 | at java.base/jdk.internal.loader.BuiltinClassLoader.findClassOnClassPathOrNull(Unknown Source)
temperature-sensor-2-1 | at java.base/jdk.internal.loader.BuiltinClassLoader.loadClassOrNull(Unknown Source)
temperature-sensor-2-1 | at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(Unknown Source)
temperature-sensor-2-1 | at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(Unknown Source)
temperature-sensor-2-1 | at java.base/java.lang.ClassLoader.loadClass(Unknown Source)
temperature-sensor-2-1 | at java.base/java.lang.Class.forName0(Native Method)
temperature-sensor-2-1 | at java.base/java.lang.Class.forName(Unknown Source)
temperature-sensor-2-1 | at java.base/sun.launcher.LauncherHelper.loadMainClass(Unknown Source)
temperature-sensor-2-1 | at java.base/sun.launcher.LauncherHelper.checkAndLoadMain(Unknown Source)
WARN[0000] /home/test/mqtt-docker/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
humidity-sensor-1 | Publishing humidity: 42.6%
humidity-sensor-1 | Publishing humidity: 59.5%
humidity-sensor-1 | Publishing humidity: 73.7%
humidity-sensor-1 | Publishing humidity: 49.9%
humidity-sensor-1 | Publishing humidity: 31.4%
humidity-sensor-1 | Publishing humidity: 45.2%
humidity-sensor-1 | Publishing humidity: 78.8%
humidity-sensor-1 | Publishing humidity: 45.8%
humidity-sensor-1 | Publishing humidity: 59.2%
humidity-sensor-1 | Publishing humidity: 60.6%
test@test:~/mqtt-docker$
```

3. **MQTT-Nachrichten:** Mit dem Befehl `sudo docker exec -it mqtt-docker-broker-1 /bin/sh -c "apk update && apk add mosquitto-clients && mosquitto_sub -t 'sensors/#' -v -C 10"` habe ich überprüft, dass die MQTT-Nachrichten korrekt übertragen werden (Image 5)

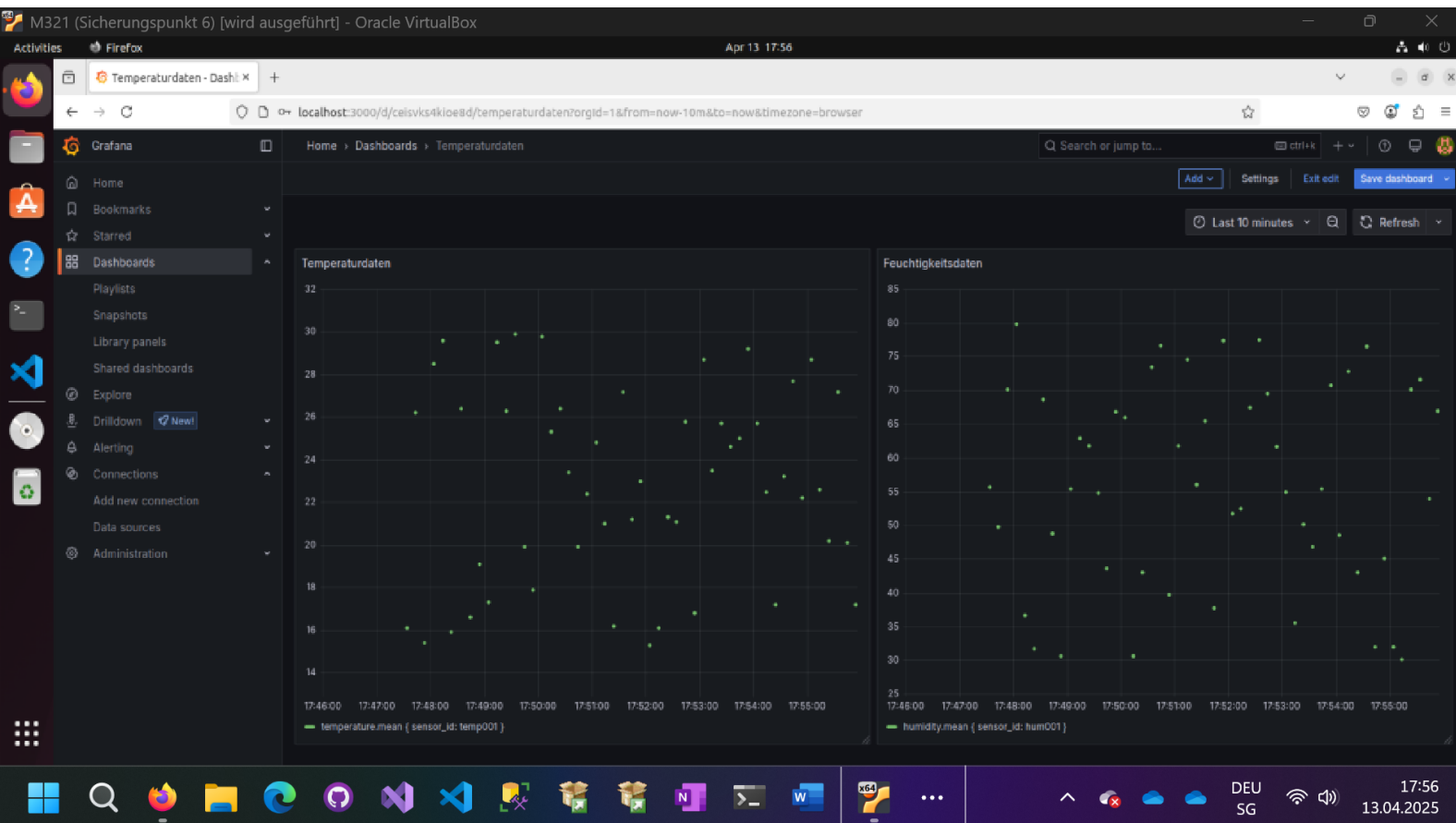
```
test@test: ~/mqtt-docker
test@test:~/mqtt-docker$ sudo docker exec -it mqtt-docker-broker-1 /bin/sh -c "apk update && apk add mosquitto-clients && mosquitto_sub -t 'sensors/#' -v -C 10"
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/community/x86_64/APKINDEX.tar.gz
v3.21.3-311-g7bb846c6e94 [https://dl-cdn.alpinelinux.org/alpine/v3.21/main]
v3.21.3-310-g56edb80e0d9 [https://dl-cdn.alpinelinux.org/alpine/v3.21/community]
OK: 25396 distinct packages available
(1/3) Installing c-ares (1.34.5-r0)
(2/3) Installing mosquitto-libs (2.0.20-r0)
(3/3) Installing mosquitto-clients (2.0.20-r0)
Executing busybox-1.37.0-r12.trigger
OK: 8 MiB in 21 packages
sensors/humidity {"sensor":"humidity","sensor_id":"hum001","value":53.4,"unit":"%", "timestamp":1.7445599243E9}
sensors/temperature {"sensor":"temperature","sensor_id":"temp001","value":23.5,"unit":"°C","timestamp":1.744559924327E9}
```

4. **InfluxDB-Daten:** Mit entsprechenden Befehlen habe ich die Speicherung der Daten in InfluxDB überprüft (Image 2 und 6)

```
test@test: ~
test@test:~$ sudo docker exec -it mqtt-docker-influxdb-1 influx -username admin -password adminpassword -database mqtt -execute 'SHOW MEASUREMENTS'
sudo docker exec -it mqtt-docker-influxdb-1 influx -username admin -password adminpassword -database mqtt -execute 'SELECT * FROM temperature LIMIT 5'
sudo docker exec -it mqtt-docker-influxdb-1 influx -username admin -password adminpassword -database mqtt -execute 'SELECT * FROM humidity LIMIT 5'
name: measurements
name
----
humidity
temperature
name: temperature
time                sensor      sensor_id unit value
-----
1744559253962000000 temperature temp001  °C  16.1
1744559263973000000 temperature temp001  °C  26.2
1744559273980000000 temperature temp001  °C  15.4
1744559283986000000 temperature temp001  °C  28.5
1744559293991000000 temperature temp001  °C  29.6
name: humidity
time                sensor      sensor_id unit value
-----
1744559253962000000 humidity hum001  %  55.7
1744559263969000000 humidity hum001  %  49.8
1744559273978000000 humidity hum001  %  70.2
1744559283983000000 humidity hum001  %  79.9
1744559293986000000 humidity hum001  %  36.6
test@test:~$
```

```
test@test: ~/mqtt-docker
test@test:~/mqtt-docker$ sudo docker exec -it mqtt-docker-influxdb-1 influx -username admin -password adminpassword -database mqtt -execute 'SHOW MEASUREMENTS'
name: measurements
name
----
humidity
temperature
test@test:~/mqtt-docker$
```

5. **Grafana-Visualisierung:** Ich habe ein Dashboard in Grafana erstellt, das die Temperatur- und Feuchtigkeitsdaten visualisiert (Image 1)



Aufgabe 6A: Erweiterte Tests und Dokumentation

1. Mindestens 2 Sensoren liefern Werte

Wie in Image 1 zu sehen ist, werden Daten von zwei Sensoren angezeigt:

- Ein Temperatursensor (`sensor_id: temp001`)
- Ein Feuchtigkeitssensor (`sensor_id: hum001`)

Die Sensordaten werden kontinuierlich aktualisiert, wie in Image 4 zu sehen ist, das die Logs der Sensoren zeigt.

2. Überwachung des Container-Status

Mit dem Befehl `sudo docker ps` habe ich den Status aller Container überwacht (Image 7). Die Ausgabe zeigt, dass alle relevanten Container laufen:

- `mqtt-docker-broker-1`
- `mqtt-docker-temperature-sensor-1`
- `mqtt-docker-temperature-sensor-2-1`
- `mqtt-docker-humidity-sensor-1`
- `mqtt-docker-influxdb-1`
- `mqtt-docker-influxdb-connector-1`
- `mqtt-docker-grafana-1`
- `mqtt-docker-publisher-1`

3. Testplan und Interaktion zwischen Komponenten

Ich habe einen umfassenden Testplan durchgeführt:

a) **Container-Status-Test:** Alle Container laufen (Image 7)

b) **MQTT-Broker-Test:** Der Broker empfängt Nachrichten von den Sensoren (Image 5)

```
sensors/humidity
{"sensor":"humidity","sensor_id":"hum001","value":53.4,"unit":"%", "timestamp":1.74455992439E9}
sensors/temperature
{"sensor":"temperature","sensor_id":"temp001","value":23.5,"unit":"°C", "timestamp":1.74455992427E9}
```

c) **Sensordaten-Test:** Die Sensoren veröffentlichen kontinuierlich Daten (Image 4)

```
temperature-sensor-1 | Publishing temperature: 28.8°C
humidity-sensor-1 | Publishing humidity: 60.6%
```

d) **InfluxDB-Datenspeicherung-Test:** Die Daten werden korrekt in InfluxDB gespeichert (Image 6)

```
name: temperature
time                sensor      sensor_id unit value
-----
-----
```

```
1744559253962000000 temperature temp001 °C 16.1
```

e) **Grafana-Visualisierung-Test:** Die Daten werden korrekt in Grafana visualisiert (Image 1)

Die Kommunikation zwischen den Komponenten funktioniert wie erwartet. Die Sensoren senden Daten an den MQTT-Broker, der InfluxDB-Connector empfängt diese Nachrichten und speichert sie in InfluxDB, und Grafana visualisiert die Daten.

4. Probleme und Lösungen

Während der Implementierung bin ich auf folgende Probleme gestoßen:

1. **Port-Konflikte:** Die Ports 1883, 8086 und 3000 waren bereits auf meinem System belegt. Ich habe die lokalen Dienste (Mosquitto, InfluxDB, Grafana) gestoppt, um die Ports freizugeben.
2. **JAR-Signierungsprobleme:** Bei den Java-Komponenten gab es Probleme mit den Signaturen in den JAR-Dateien:

```
Exception in thread "main" java.lang.SecurityException: Invalid signature  
file digest for Manifest main attributes
```

Dies trat bei den Sensoren und dem InfluxDB-Connector auf. Ich musste die pom.xml-Dateien anpassen, um die Signaturprüfung zu deaktivieren.

3. **Container-Abhängigkeiten:** Die Reihenfolge des Starts der Container war wichtig. Ich musste sicherstellen, dass der Broker und InfluxDB vollständig gestartet sind, bevor die anderen Container gestartet werden.

5. Zusammenfassung und Ausblick

Das implementierte System erfüllt alle Anforderungen der Aufgabenstellung. Die Docker-Containerisierung ermöglicht eine einfache Bereitstellung und Skalierung des Systems. Das System kann wie folgt erweitert werden:

1. **Weitere Sensoren:** Hinzufügen weiterer Sensortypen wie Luftdruck oder Lichtsensoren
2. **Authentifizierung:** Implementierung von Sicherheitsmechanismen für den MQTT-Broker
3. **Alarme:** Konfiguration von Alarmregeln in Grafana für kritische Sensorwerte
4. **Persistente Speicherung:** Konfiguration von Retention-Policies für die langfristige Datenspeicherung

Die Nutzung von Docker hat sich als effektiv erwiesen, um ein verteiltes System mit mehreren Komponenten zu implementieren und zu verwalten. Die Container-Technologie ermöglicht eine klare Trennung der Komponenten und vereinfacht die Bereitstellung.