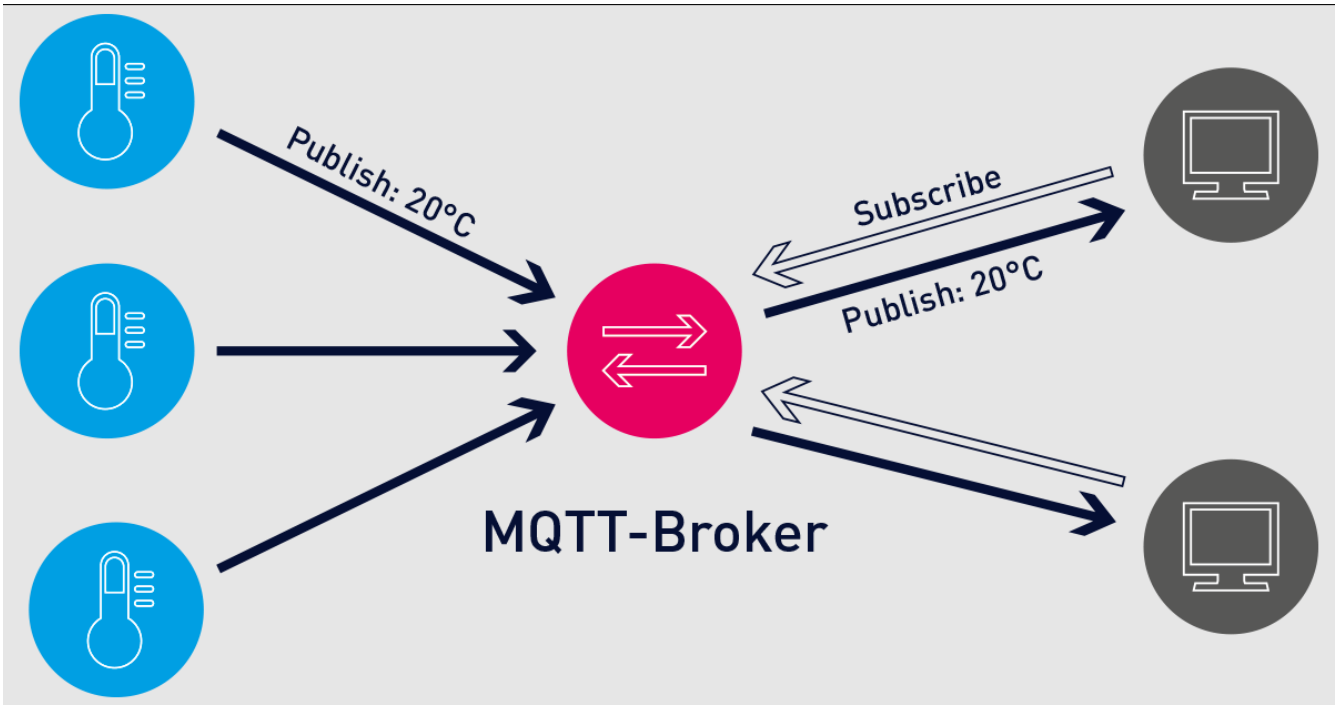
MQTT

Table of Contents

1. Was ist MQTT?

MQTT steht für Message Queuing Telemetry Transport MQTT ist ein Messaging-Protokoll das für die Kommunikation von Maschine zu Maschine verwendet wird.

2. Architektur



2.1. Was ist ein Broker?

Ein Server, welcher von den Clients benutzt wird. Der Broker empfängt die Nachricht der Clients und sendet diese zu andere Clients weiter. Das heißt die Clients koomunizieren nicht direkt mit einander, sondern verbinden sich beide mit dem Broker.

3. Tutorial

3.1. Verbindung zu MQTT herstellen

Zuerst muss die Verbindung zu dem Broker im `application.properties` konfiguriert werden:

```
mp.messaging.incoming.channelname.connector=smallrye-mqtt

mp.messaging.incoming.channelname.host=public.mqtthq.com

mp.messaging.incoming.channelname.port=1883

mp.messaging.incoming.channelname.topic=topicname/#
```

1. Es wird ein MQTT-Connector namens "smallrye-mqtt" verwendet.
2. Es wird zum öffentlichen MQTT-Broker "public.mqtthq.com"
3. auf Port 1883 verbunden.
4. Die Nachrichten werden dann unter "topicname/#" empfangen. Die Raute steht für alles was nach "topicname/" kommt.

`channelname` ist ein frei erfundener Name, welcher anschliesend im Code mittels der `@Incoming` Annotation refernziert wird.

Zum Testen, ob die Nachrichtent dann auch ankommen, habe ich einen online MQTT Web Client gefunden. Natürlich muss der Name des Topics übereinstimmen.

3.2. Einen Empfänger erstellen

```
@ApplicationScoped
public class MqttReceiver {

    private Logger logger = Logger.getLogger(MqttReceiver.class.getName());

    @Incoming("channelname")
    public void receiveMessage(byte[] byteArray) {
        String messageString = new String(byteArray);
        logger.info("Received raw message: " + messageString);
    }

}
```

Die Methode `getLogger()` sucht einen Logger und wenn ein Logger mit dem übergebenen Namen existiert, gibt die Methode diesen Logger zurück. Andernfalls erstellt die Methode einen neuen Logger mit diesem Namen und gibt diesen zurück.

Die Methode `receiveMessage()` akzeptiert ein Byte-Array als Eingabe und konvertiert es dann in ein String. Der Logger protokolliert die Message.

3.2.1. Starten

```
./mvnw quarkus:dev
```

Wenn dein Quarkus Projekt erfolgreich gebaut hat sollte folgendes ausgeben werden:

```
Connection with public.mqtthq.com:1883 established successfully
Connection with public.mqtthq.com:1883 established successfully
Subscription outcome: Future{result=0}
```

Sollten Probleme bezüglich des Ports auftreten, versuche dies:

```
netstat -anop | grep

kill -9 <ProzessID>
```

3.3. Einen Sender erstellen

3.3.1. `application.properties` konfigurieren:

```
# Outbound
mp.messaging.outgoing.channelname2.connector=smallrye-mqtt

mp.messaging.outgoing.channelname2.host=public.mqtthq.com

mp.messaging.outgoing.channelname2.port=1883

mp.messaging.outgoing.channelname2.topic=topicname/
```

Zum Senden muss logischerweise `outgoing` und ein anderer beliebiger Channelname verwendet werden.

3.3.2. Code

```
@ApplicationScoped
public class MqttSender {

    @Outgoing("channelname2")
    public byte[] sendMessage() {
            return "Hello i am David!".getBytes();
        }
```

```
}
```

Es wird ständig die Message gesendet. Bedauerlicherweise habe ich es noch nicht geschafft dieses Problem zu beheben.

3.4. Testen

Wie schon bereits im obigen Text beschrieben, kann nun unser Nachrichtenempfänger mittels einem online MQTT Web Client geprüft werden, ob auch wirklich Nachrichten ankommen.

Quelle