How to install a MQTT Broker on Raspberry Pi

What you will need:

Walking Dead Easy GIF from Walkingdead GIFs

- A Raspberry Pi 3 (and higher)
- A way to power the Pi (micro-USB cable)
- A micro-SD-Card
- A PC or Laptop (with (micro)-SD-Card Slot)
- Internet (of course)
- Some basic PC knowledge

Setting up your PC

First, we need some basic tools on our PC. Right now there is no Operating System for the Pi on our SD-Card so we will have to flash that ourselves. Therefore we need the program balenaEtcher. It is very easy to use tool for flashing operating systems for the Pi onto SD-Cards. Download this program and install it. Of course, we will also need an image of an operating system to flash onto the SD-Card. We will use Raspberry OS, which is developed especially for the Raspberry Pi. You can download it on the official website: Raspberry OS. We need the version “Raspberry Pi OS (32-bit) Lite”. This will come without any GUI and random software we don't need. Later on, in the project, we need a program to communicate with the Pi. I recommend the tool Putty, but you can use any other SSH program if you want to and already did something similar. Download and install it (Putty).

Flash Raspberry OS onto the SD Card

The Raspberry OS comes in a zip-archive that we fortunately do **not** have to extract. Plugin your SD-Card. Start balenaEtcher and select the image (the zip-archive). Now make sure that the correct target (the SD-Card) is chosen. If not change it. Keep in mind that everything that now is on the SD-Card will be erased. When you have done that click on flash and wait until you get a messaged that everything has finished. Because balenaEtcher automatically ejects the SD-Card out of your System you have to take it out of your PC and plug it back in, for the next step.

Preparing Raspberry OS

We now have to add some files to the SD-Card before we can start the Pi for the first time. Later we want to control the Pi over the network so we do not need an extra monitor, mouse, and keyboard. Therefore we can use SSH. This is not activated by default. To activate it we have to place an empty file into the root of the SD-Card. As you may see the SD-Card is recognized by your system and already has some files in it. Open up the standard windows editor and go to File > Save. Select the SD Card as the savepoint. Now choose as file-type: All. This is needed otherwise the editor will save the file with a .txt ending. Name the file ssh and click on save. (as mentioned under 3) If you want to connect the Pi with a LAN-Cable with your network, use it. It is a very simple option and you can directly jump to the next step. If you want to use a Wifi Connection read on. Open up an editor and paste this code:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
```

```
update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>

network={
 ssid="<Name of your wireless LAN>"
 psk="<Password for your wireless LAN>"
}
```

Fill in your country code (Germany is DE). You can find all of them here: Country Codes. Also, fill in your Wifi-SSID (Name of your network) and your Wifi-password. Save the file in the root of your SD-Card as type all again and name it “wpa_supplicant.conf”. Read more about this on the official website: Setting up wireless networking

On to the pi

Plugin the SD-Card into the desired port (it is on the bottom end of the device). Plugin your ethernet cable now if you did not set up wifi and also plug in the micro-USB cable for power. Make sure you are using at least a 5V 1A USB-Powersupply. If you have something more capable like 5V 2A or 3A it would be better. But don't worry. You can even drive it off of a USB-Port from your PC. It will just be a bit slower. As soon as you plug in the power cable it will boot up for the first time. On first boot, it will need some time to set itself up. Give at least 2 minutes, better 3. When you´ve done everything correctly we can now access the pi. Therefore open Putty, which we have downloaded earlier. Just type into the Host Name field:

```
raspberrypi
```

and click “Open”. It will ask for username and password. The standard username is “pi” and the standard password is “raspberry” (both without quotation marks). You can change these if you want to.

Installing Docker & docker-compose

We may want to use Docker. Docker is a tool for containerized apps. This makes it easier to deploy multiple services onto a single device and manage them all. Probably will the Pi not be under 100% load all the time so you may want to deploy more applications in the future.

First, we want to update all installed packages with:

```
sudo apt-get update
```

Then we need an script to install docker. You can easily get with this command:

```
curl -sSL -o install.sh https://get.docker.com
```

Then we just need to execute it:

```
sudo sh install.sh
```

After the installation is complete we have to add our user (pi in this case) to the group docker. This allows him to execute docker commands without sudo:

```
sudo usermod -aG docker pi
```

After you have done that close Putty and connect again for the changes to take effect. We now need to install docker-compose. This is a very handy tool for deploying docker-containers that allows the user to make configuration files and just build the containers from them. For docker-compose to work we first need to install some python related stuff:

```
sudo apt-get install -y libffi-dev libssl-dev
```

```
sudo apt-get install -y python3 python3-pip
```

```
sudo apt-get remove python-configparser
```

Then we can install docker-compose:

```
sudo pip3 install docker-compose
```

You can check if everything worked by executing:

```
docker
```

or

```
docker-compose
```

To clean up our space we can execute:

```
rm install.sh
```

This can be done but is not 100% necassairy.

Install mosquitto (MQTT Broker) and configure it

> mosquitto is an open-source (EPL/EDL licensed) message broker [...] and is suitable for use on all devices from low power single board computers to full servers.

This sounds perfect for our needs. Lucky as we are there is also a docker-image provided (you can find it here). For organization reasons we should create for a folder where we can save all of the data related to mosquitto. Just execute:

```
mkdir mosquitto
```

We now need to navigate into the new folder with:

```
cd mosquitto/
```

In this folder, we can create the needed docker-compose file. This file houses all the configuration that is needed to start and stop the docker-container in which mosquitto is located.

```
nano docker-compose.yml
```

You should now see a nearly empty/black windows. nano is kinda like the word of Linux. So basically a text editor. Paste in this text:

```
version: "3"
services:
  broker:
    image: arm32v6/eclipse-mosquitto
    volumes:
      - ./mosquitto:/mosquitto
    restart: always
    ports:
      - "1883:1883"
```

You can paste text in Putty by just clicking the right mouse button. If you want to navigate through the file use your arrow keys. For more information about docker you can read this article: The definitive Guide to Docker compose Save the file by hitting “Ctrl+S” and exit the file with “Ctrl+X”. We now need to make some new folders where mosquitto have is config files and some persistent storage. Make a new folder mosquitto and navigate into it:

```
mkdir mosquitto
cd mosquitto
```

In this new folder, we first create a folder named “config”:

```
mkdir config
```

In this folder, we create a new file with nano:

```
nano config/mosquitto.conf
```

In this file we paste:

```
persistence true

persistence_location /mosquitto/data
allow_anonymous false
password_file /mosquitto/config/passwd
```

Save it and exit nano. As you may see we also need a password file. We will create this shortly inside of the docker container itself. Now go back to the docker-compose file. Therefore you need to go up the hierarchy of your folder structure by one level. Just execute:

```
cd ..
```

You can now start mosquitto for the first time. Execute this:

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

It will now start pulling all the needed files from the internet. When it finished that it will start the container asap. When you are able again to type commands, we now need to generate the password file, because we don't want anyone to access our broker. This will happen in the container itself. Execute this:

```
docker-compose exec broker sh -c "mosquitto_passwd /mosquitto/config/passwd [username]"
```

Replace [username] with a username you want to. It will ask for a password. Also, choose one there. You can restart the container now:

```
docker-compose restart
```

Connection

You can now access the MQTT Broker inside of your network by the hostname raspberry and the port 1883. If you want to access the broker from the outside of your network, you have to open the 1883 TCP in your router. This is highly dependent on your own router. Here are some links to some tutorials: How to open or forward a port on a router (Linksys) Portfreigaben in FRITZ!Box einrichten (FRITZ!Box is one of the most used routers in Germany)

When you have done that, you can get your public IP address by executing:

```
curl ifconfig.me
```

But keep in mind that your public IP will probably change on a regular base so you have to update this in any service and device where you are using the broker. A workaround for this is using a DynDNS Service like no-ip. But for this you have to google a bit urself.

From:
https://wiki.eolab.de/ - **HSRW EOLab Wiki**

Permanent link:
https://wiki.eolab.de/doku.php?id=user:jan001:mqtt_broker&rev=1593082053

Last update: **2021/08/24 17:34**