MQTT

# Introduction

MQTT is a lightweight publish/subscribe messaging protocol, which is used widely outside of the BMS world for device-to-device communication. It has been designed for use on devices which may have very limited resources, and may operate on low-bandwidth, low-reliability networks. For these reasons it is a good choice for the emerging “Internet of Things” world of connected devices.

Devices (which are in most scenarios MQTT clients) can publish information to a ‘topic’ on an MQTT Broker. Other devices can subscribe to that topic, and have any information that is published to it pushed to them by the broker. The broker attempts to ensure that all messages are delivered to all clients that are subscribed, and it can queue messages up and keep trying to deliver them to unresponsive devices.

# Setting up a broker

## What are the options?

For clients to communicate a broker is always needed, and there are several options available. There are public brokers available online, but they are mostly for testing purposes, as for most applications you will probably want some assurance of security (on a totally open broker someone else could theoretically publish/subscribe to topics you have set up).

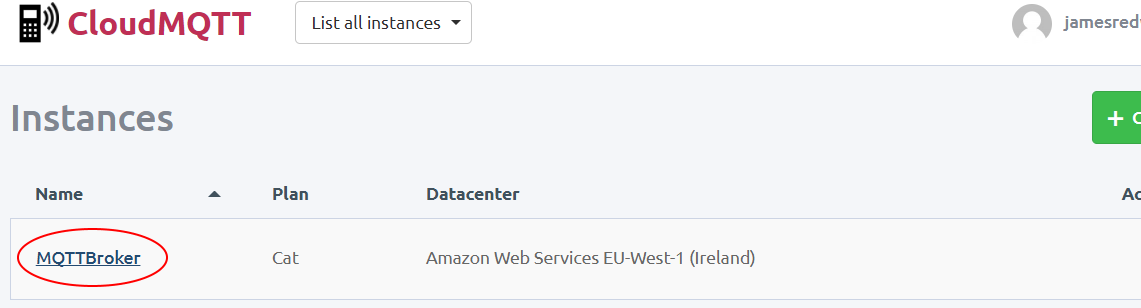
Cloud-based broker hosting is a sensible option in many scenarios, which involves renting a virtual server and installing the broker software. There are also a few cloud-based services which will provide a broker for you, allowing you to open an account which provides you a login to administer your broker, setting up security etc yourself. Many of these will give you a fairly generous, free-of-charge account, only requiring payment as the volume of traffic increases or you are using it commercially.

Another option is to install a broker locally, which may be necessary if you are on a private network which is not connected to the internet. This would typically be on a PC, but any device with sufficient computing resources could act as a broker if there is a broker implementation available.

## Getting started

To get started we will open a free cloud-based account. Options include Heroku, CloudMQTT, IBM Bluemix, ThingStudio and others. I have chosen to use CloudMQTT as sign up is quick and easy, and the admin console is nice and simple to use.

1. Visit <https://www.cloudmqtt.com/> and open a free account. You will just need to provide your email address, and they will email you a link to get started.
2. Click the link, chose a password and then select create a new instance.
3. Give the instance a name, and select the Ireland-based data centre, then click Create New Instance.
4. Click on the new instance you have just created



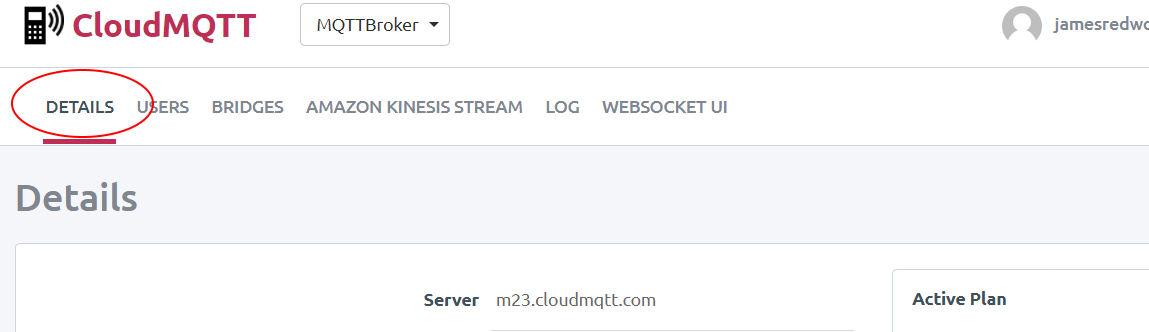
1. Select the users tab and create a new user called ‘workbench’, with password ‘workbench’. You are free to set up security as you wish, but I suggest that creating a user for each application or device that you will connect to the broker is a good idea.
2. MQTTLens is a free Chrome app which I have found useful for testing and debugging. We will install it and connect it to the broker later so create another new user ‘mqttlens’ with password ‘mqttlens’.
3. Keep this webpage open, as we will be coming back later to set up some Access Control Lists.

# Install Niagara MQTT Driver

At the time of writing this document the MQTT Driver by Tridium has only just left beta. I had to install the driver and license it, but it is likely to be included as a standard feature in future workbench releases and will probably be included on the standard license.

## Connecting to the broker

1. If you do not already have the driver you will need to install it in the modules folder of your workbench installation, and restart both workbench and any running station(s).
2. When you have installed the driver check it is licensed by opening the abstractMqttDriver palette, and dragging an AbstractMqttDriverNetwork into the Drivers container of your station.
3. Open the MqttClientDriverDeviceManager view on the AbstractMqttDriverNetwork and create a new AbstractMqttDriverDevice called WorkbenchMqttClient by clicking the New button.
4. Right-click the device you just created, and select the AX Property sheet view.
5. Go back to your browser window where you are logged into your CloudMQTT account and select the details tab.



1. Copy the server name and paste it into the Broker Ip Address property of your MQTT Driver Device in workbench.
2. Copy the SSL Port number and paste it into the Broker Port property. The standard port numbers for MQTT are 1883 for unsecured, and 8883 for MQTT over SSL/TLS. Most devices will use these as default, but CloudMQTT works by opening different ports for each user.
3. Scroll down the property sheet and under Username And Password enter ‘workbench’ for both.
4. Click Save, and then right-click the device in the nav window of workbench and select actions from the pop-up window.



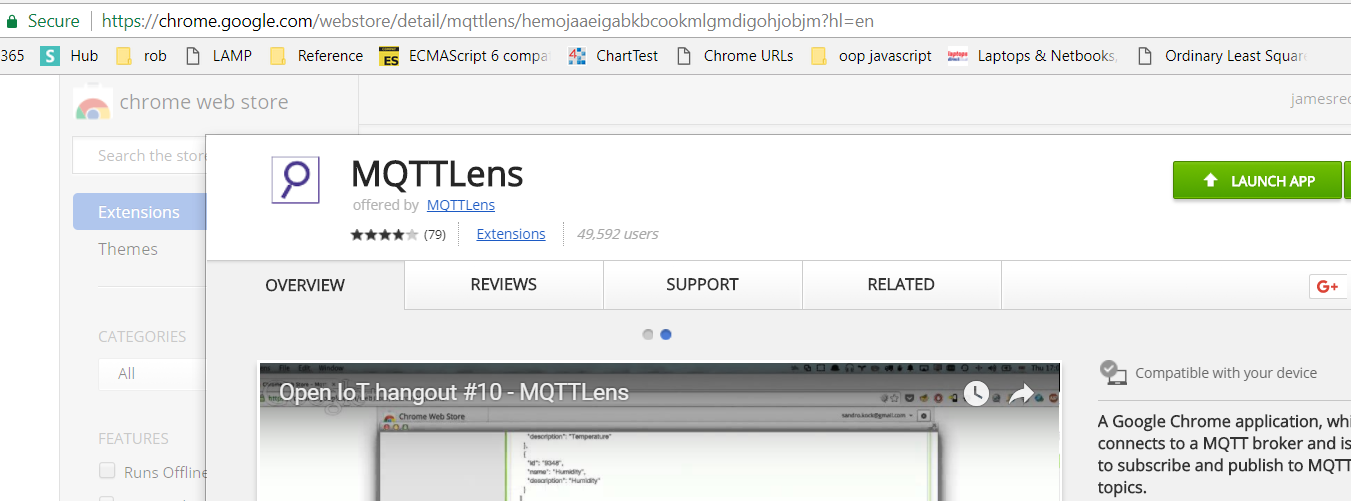
1. Select the Connect action. It is worth noting that if you select Ping, the ping will fail if you have not already connected. The status should change to show that you are now connected to the broker.

# Installing a test client

While workbench can both publish and subscribe to topics on the broker it is useful for testing and debugging to have another client that we can test with. This is especially true if you are having trouble getting workbench to connect! By using an additional client app we can verify that the broker is correctly set up before looking for any problems elsewhere.

You can get MQTT Client apps on your smartphone, although some are intended for specific purposes and may not be good for general testing. I found a good one for iPhone called MQTT Tool, if you want to you can check your app store and find one for your phone, but for the purposes of this lab we will use a Google Chrome App.

1. Open Google Chrome, or download and install it if you do not already have it. Use Google search to search for MQTTLens, the top result should be a link to the Chrome Web Store. Go to the Web Store and install the app.



1. Once you have installed the App you can access it in future by opening a new Chrome window and selecting Apps from the bookmark bar (by default), or by visiting chrome://apps/.

# Publishing and subscribing to a topic

All messages between devices using MQTT are published to a ‘topic’ on the broker. You do not need to explicitly create topics, the broker will create the topic if something publishes or subscribes to it. A topic is a UTF-8 string, consisting of one or more topic levels which are separated by a forward slash / character.

You can structure your topics however you wish, but for the lab today I would suggest structuring them based on the device that will be doing the publishing (as there will only be one publisher, but possibly many subscribers). There are other approaches you could use but I found this helpful for getting started, so we will use topics that look as follows:

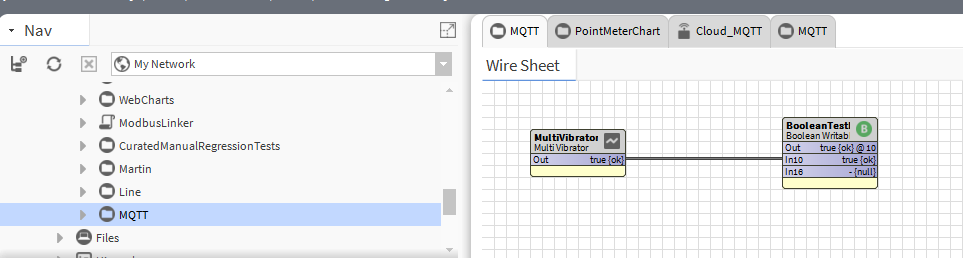
workbench/1stFloor/Lobby/Lights

It’s a good idea to have a quick read online about topic best practises and naming rules. We won’t use it much today, but it is worth mentioning now that topic strings can have wildcard characters. A single level wildcard is a + symbol, and a multi-level wildcard is a # symbol. These are useful when you want a device to subscribe to lots of information, for instance all information relating to 1stFloor, or all information relating to ‘heating valves’ on any level.

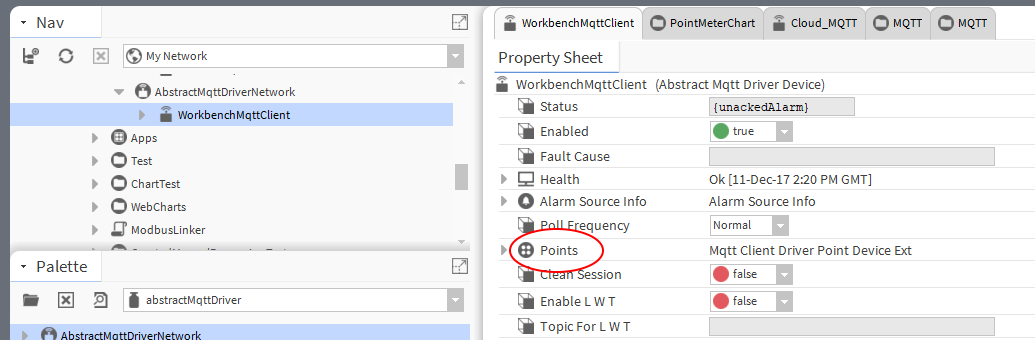
There are also topics that start with a $ symbol. These get treated differently to other topics, and contain information about the broker such as how many clients are currently connected (e.g. $SYS/broker/clients/connected). If you look at the statistics page of your CloudMQTT account you will see a full list of these topics, which already exist on your MQTT broker.

## Publishing to your broker from Niagara

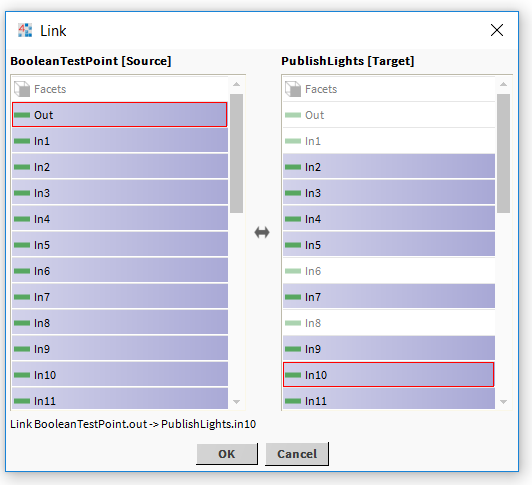
1. Under Config in your station make a new folder called MQTT, and in the wiresheet create a Boolean point called BooleanTestPoint which switches every few seconds with a multivibrator from the kitControl palette.



1. Right-click the Boolean point and select link mark.
2. Open the points folder of the device you created under the MQTT Driver.



1. Click new and create a MqttBooleanPublishPoint. Call the point ‘PublishLights’ and give it the topic ‘workbench/1stFloor/Lobby/Lights’.
2. Right-click your newly created point, and select Link From BooleanTestPoint, linking the Out slot of the test point to the In10 of the MQTT Point.

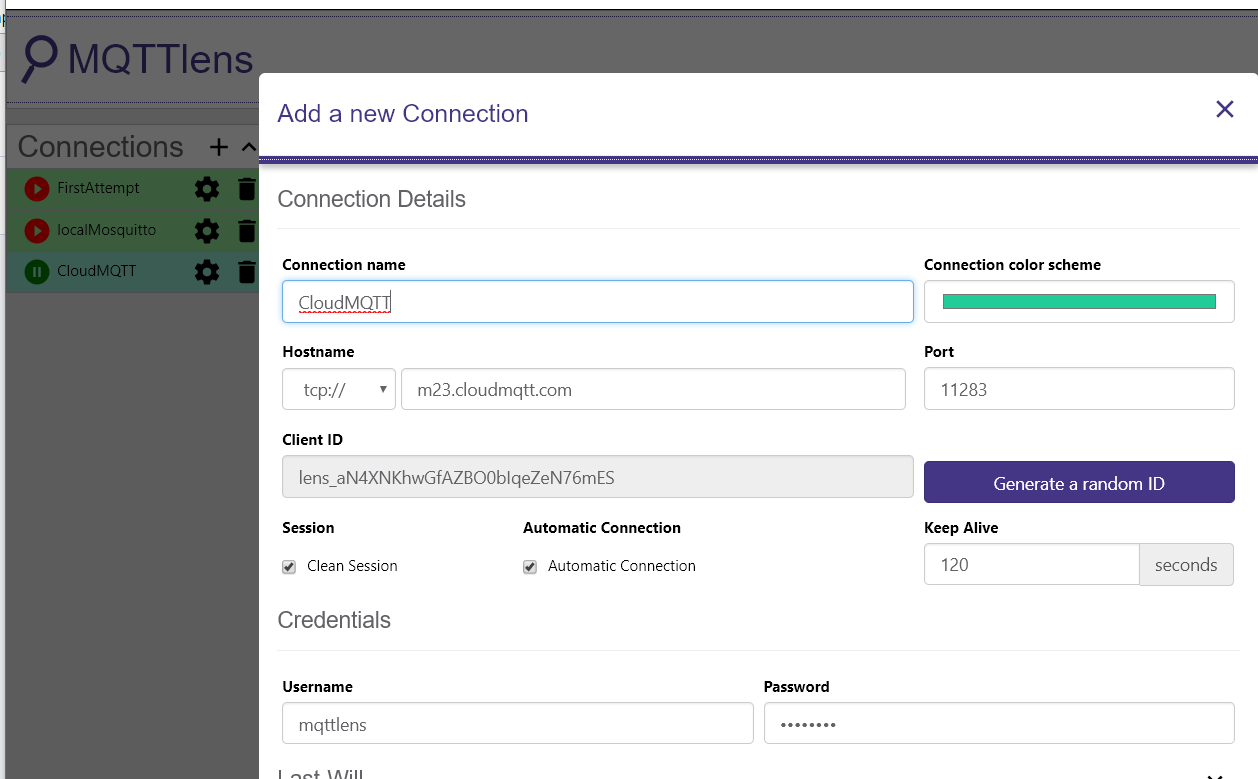


This point will appear to be healthy, and you may be optimistically hoping you are now publishing data to the broker, but as we are about to see when we try to subscribe to the point, there is another step to go.

## Try subscribing to the point with the test client

You will notice if you look around the CloudMQTT Account that we can’t actually see the data in any way. There are some statistics we can look at, but no way to check what topics are carrying messages. We could try subscribing to our published point using Niagara, but if we are suspicious that something may not be correct this isn’t that helpful. Instead let’s use the test client to see if our point is really being published.

1. Open MQTTLens App in Chrome that we installed earlier (either from the bookmarks bar or by visiting chrome://apps/).
2. Click the + symbol next to Connections on the right to make a new connection.
3. Copy the connection details you used previously, except this time use username and password ‘mqttlens’.
4. While Niagara connects successfully on what CloudMQTT Details Page lists as the SSL Port, MQTTLens requires that you use the Port number instead. This is normally the same as the port number we connected Niagara on, except it starts with a 1 not a 2. You can find both under the details tab in your CloudMQTT account.



When you have a successful connection with MQTTLens try and subscribe to the topic we published to. Enter ‘workbench/1stFloor/Lobby/Lights’ into the subscription box and hit Subscribe. You should see that not a lot is happening, so something is not right.

## Subscribing to a point in Niagara

As we mentioned earlier there are already some points on our broker that we could try and subscribe to, and we know for sure that data is being published to these. Lets attempt to read some of the broker statistics we mentioned earlier.

1. Create a new MqttStringSubscribePoint and call it ‘BrokerUpTime’.
2. Give it the topic $SYS/broker/uptime
3. You will see that the point is stale as soon as it is created. When you create a subscribe point it is necessary to activate it by going into the property sheet and firing a subscribe action on the proxy extension. Right click the point, select AX Property sheet, and Actions > Subscribe.

At this point in the lab the point will remain stale, so while Niagara may mislead you into thinking it is doing something, in reality neither client is able to subscribe or publish yet. This is because we still have some configuration to do on our broker.

## Setting up Access Control Lists

By default, the security settings on our broker will not allow clients access to anything, even though they have provided a valid username and password to connect. This might vary depending on the broker you are using, but it is a sensible policy and one that most brokers will adopt (it is also worth noting that access to $ system topics can vary by broker as well).

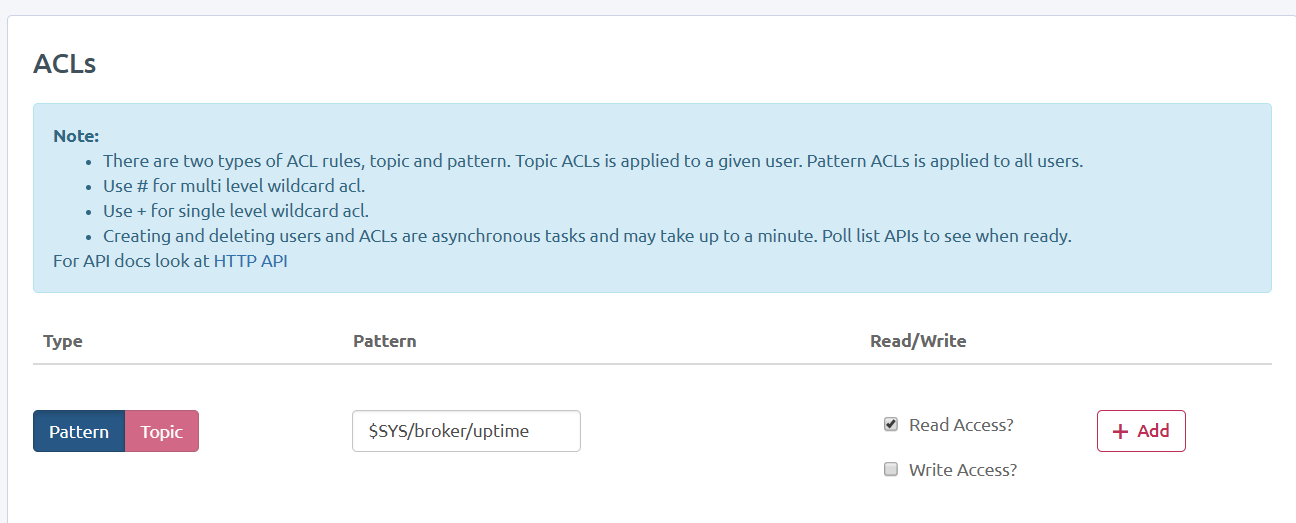
1. Open your CloudMQTT account in your web browser.
2. Under the Users tab, scroll down until you reach the ACLs section.

We are going to need to create a rule that allows access to the UpTime system variable we are trying to read. The rule will allow access to any client trying to subscribe to the topic.

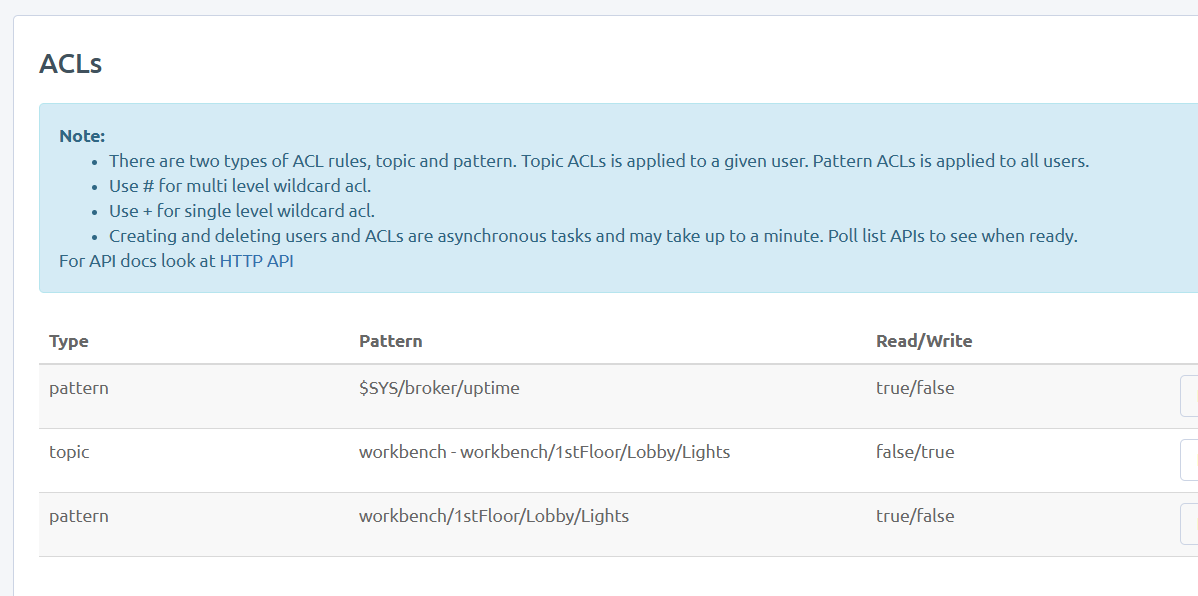
We also need to create two rules for the PublishLights point that we want to use. One rule will allow workbench (and workbench only) to publish to the topic, and another rule will allow anybody to subscribe to the point.

Note that we only need one rule for the subscription point in this case, because the broker already has access to publish to itself.

1. Create a Pattern rule with read access as in the picture below, and click the Add button



1. If you check back in workbench, you should see that we are now reading the point successfully, and it is updating every few seconds. If it is still stale, go into the AX property sheet view of the point, and right click the proxy extension, and Actions > Subscribe.
2. Now we will get our Niagara point publishing – add two new rules to the ACL, so that it is set up as below



Opening MQTTLens should now show that our data is publishing from Niagara successfully.

The exact process for configuring the security/access control will vary depending on the broker you use, but it is likely to be very similar to what we have seen today using CloudMQTT.

# Further Exercises

If you would like to do some further playing with this you can try subscribing to points created by other people in the lab, installing a client app on your smart phone and using it to update some data in Niagara, or setting up a local broker. Mosquitto MQTT Broker is a free, open source broker that you can install on your laptop and host your own broker (although you will need to install some dependencies first).

END OF LAB