Private Mosquitto using docker

New to docker?

Me too!!!

So, brave as we are we fire up a linux server (a free AWS instance for now, may as well be a digital ocean basic server or something even cheaper) and just install and start it

```
ssh ...
sudo yum/apt install docker
sudo service docker start
```

It happens that downloading and starting services with docker is straightforward. First you need to know the name for your service and than just search for it on https://hub.docker.com

One of the most popular dockerized mosquitto at the time that we write this is https://hub.docker.com/r/toke/mosquitto/

So we simply read the page we just mentioned, and install it with

```
sudo docker pull toke/mosquitto
```

Et voilà!

We could now run it as it is, but we may want to configure it first to better suite our needs! After all the title says “Private”!

But we cannot wait, so we do it anyway :)

```
sudo docker run --name mqtt toke/mosquitto
```

We started the docker container without mapping any port or volumes, so that the service is not yet accessible from the outside, but we can still do something interesting with it, like having a look at what’s inside firing up a shell into the container:

```
sudo docker exec -i -t mqtt /bin/bash
```

Interestingly enough there is no `init` process, just our mosquitto server, and our home points to the place where the source for mosquitto was unpacked. Wandering around we find out that this image keeps the configuration in `/etc/mosquitto`.

When we are done, we need to stop and remove the instance from docker so that we can run it again later with different, more useful, options:

```
sudo docker stop mqtt
sudo docker rm mqtt
```

Let's create some empty folders that we will use later as docker volumes:

```
sudo mkdir -p /srv/toke-mosquitto/{config,data,log}
```

Obtaining a certificate

In order for our service to be (hopefully) private we need to set up a secure channel. We need a certificate.

My favourite at the moment is https://letsencrypt.org/

With a few commands we are set-up:

```
sudo yum install epel-release nginx
```

We configure our virtualhost in /etc/nginx/conf.d/virtual.conf as we usually do and we open the port 443 in the firewall (or in the Security groups if we are using AWS).

```
sudo service nginx restart
wget https://dl.eff.org/certbot-auto
chmod a+x certbot-auto
./certbot-auto --nginx -d <my-domain> certonly [--debug]
```

Our new certificate should now be in /etc/letsencrypt folder.

This is not a full guide on how to obtain a certificate, but following the errors that you will likely get on the screen, should allow you to

end up with working certificates.

We will need to copy the certificates in a place accessible by docker:

```
sudo cp -a /etc/letsencrypt /srv/toke-mosquitto/config/
```

(we should not forget to do this again when we renew the certs)

Following the instructions here https://mosquitto.org/2015/12/using-lets-encrypt-certificates-with-mosquitto/ we also create the `chain-ca.pem` file needed by mosquitto:

```
cd /srv/toke-mosquitto/config
wget https://letsencrypt.org/certs/isrgrootx1.pem
sudo bash -c 'cat /letsencrypt/live/<mydomain>/chain.pem is
rgrootx1.pem > chain-ca.pem'
```

Note: We learn as we do, but looking at the OwnTracks website, for example, and at the fact that MQTT applications use the certificates in a different way (the server is not public, so there is no need for a public CA), it may be that we do not need to go through letsencrypt, we could just have used some other tool to generate owr own CA, certificates and even pre shared keys for our devices… And with less problems of expired keys… Next time we'll do better!

Configuring Mosquitto

Merging infos from https://github.com/toke/docker-mosquitto and few other places we can to override the mosquitto configuration and also persist logs using docker volumes

In which we can put our own `mosquitto.conf` and access control files.

```
listener 8883
allow_anonymous false
password_file /mqtt/config/pwfile
cafile /mqtt/config/chain-ca.pem
certfile /mqtt/config/letsencrypt/live/<my-domain>/cert.pem
keyfile /mqtt/config/letsencrypt/live/<my-domain>/privkey.p
em
```

To create the missing `pwfile` we can use the mosquitto_pass utility using another machine or `sudo docker exec -i -t mqtt /bin/bash` on an image with the config file mounted `rw`.

Note: on AWS we also need to open port 8883 in the security group to make it reachable.

Ready to start

We are ready to start our mosquitto with ssl support using docker:

```
sudo docker run -ti -p 8883:8883 -v /srv/toke-mosquitto/con
fig/:/mqtt/config:ro -d --name mqtt toke/mosquitto
```

And we can test that it works from our laptop by issuing the following command:

```
mosquitto_pub -h <my-domain> -t test/topic -m test -p 8883
--capath /etc/ssl/certs/
```

To troubleshoot issue with SSL we can also use

```
wget https://letsencrypt.org/certs/isrgrootx1.pem
openssl s_client -connect <my-domain>:8883 -CAfile isrgroot
x1.pem
```

Happy hacking!

What I learned

I learned a couple of things while doing this:

- Next time I should generate my own CA and certificates, and also authenticate users with certificates instead of passwords.
- Instead of writing pages like this I could just have written a much shorter `Dockerfile` to do all the required steps!