

Wireless Networks Project Report

Prashant Dhillon

About MQTT:

MQTT is a lightweight publish/subscribe messaging protocol. It is useful for use with low power sensors, it is also applicable to many scenarios.

Mosquitto: Mosquitto is a light weight MQTT Broker service that implements a MQTT protocol with a client service.

➤ What is the status of the mosquitto service on your system?

After General setup of Mosquitto program. It's service status can be checked by "systemctl status mosquitto.service" command. It produces the following output:

```
prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ systemctl status mosquitto.service
● mosquitto.service - LSB: mosquitto MQTT v3.1 message broker
   Loaded: loaded (/etc/init.d/mosquitto; generated)
   Active: active (running) since Sat 2020-04-04 15:19:20 CEST; 3min 52s ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 1 (limit: 4915)
   CGroup: /system.slice/mosquitto.service
           └─25727 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

avril 04 15:19:20 prashant-Lenovo-ideapad-530S-14IKB systemd[1]: Starting LSB: mosquitto MQTT v3.1 message broker...
avril 04 15:19:20 prashant-Lenovo-ideapad-530S-14IKB mosquitto[25721]: * Starting network daemon: mosquitto
avril 04 15:19:20 prashant-Lenovo-ideapad-530S-14IKB mosquitto[25721]: ...done.
avril 04 15:19:20 prashant-Lenovo-ideapad-530S-14IKB systemd[1]: Started LSB: mosquitto MQTT v3.1 message broker.
prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$
```

Here we can observe that mosquitto service is status is active and running. Also it shows basic messages about how the service is started like daemon mode.

The mosquito.service is installed under /usr/lib/systemd/system/mosquitto.service.

➤ What configuration file is used by your system for mosquitto?

Config file use by system is found at location "/etc/mosquitto/mosquitto.conf"

Below is the specification of my file I used in the to run mosquitto.

```
pid_file /var/run/mosquitto.pid
persistence true
persistence_location /var/lib/mosquitto/
protocol mqtt
password_file /etc/mosquitto/users
allow_anonymous false
log_dest stdout
log_dest file /home/prashant/mosquitto.log
include_dir /etc/mosquitto/conf.d
```

Further specification of config file is explained in coming steps.

- what is the additional information given by the dpkg command earlier?

Dpkg command provided information about mosquitto service Log file, certificate file, its configuration file and configuration directory. Below is the command output.

```
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ dpkg -s mosquitto
Package: mosquitto
Status: install ok installed
Priority: optional
Section: net
Installed-Size: 309
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Multi-Arch: foreign
Version: 1.4.15-2ubuntu0.18.04.3
Depends: adduser (>= 3.10), libuuid1 (>= 2.16), lsb-base (>= 4.1+Debian3), libc6 (>= 2.14), libssl1.1 (>= 1.1.0), libwebsockets8 (>= 1.6.0), libwrap0 (>= 7.6-4~)
Suggests: apparmor
ConfFiles:
/etc/init.d/mosquitto 7c1c057c3c625deaac042ba97032ec69
/etc/logrotate.d/mosquitto 7f203558c910819d152ee3f033063cff
/etc/mosquitto/ca_certificates/README c1c6ae67f2def06c6a483be09b49d4de
/etc/mosquitto/certs/README 4d8a70d4cefab07d4dabc5be1f786c1f
/etc/mosquitto/conf.d/README b4ac621550824082a735732bfb42b51d
/etc/mosquitto/mosquitto.conf 379a6d6e30d19bfdadf0099a9f3f0770
Description: MQTT version 3.1/3.1.1 compatible message broker
This is a message broker that supports version 3.1 and 3.1.1 of the MQTT
protocol.
.
MQTT provides a method of carrying out messaging using a publish/subscribe
model. It is lightweight, both in terms of bandwidth usage and ease of
implementation. This makes it particularly useful at the edge of the network
where a sensor or other simple device may be implemented using an arduino for
example.
Homepage: http://mosquitto.org/
Original-Maintainer: Roger A. Light <roger@atchoo.org>
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$
```

Terminal created for logging:

Using command “Sudo tail -f **locaton of log file” we can see the live logging status of mosquitto broker.

```
prashant@prashant-Lenovo-Ideapad-530S-14IKB: ~
File Edit View Search Terminal Help
(base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~$ conda deactivate
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~$ clear
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~$ sudo tail -f /var/log/mosquitto/mosquitto.log
[sudo] password for prashant:
1586006360: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
^Z
[1]+  Stopped                  sudo tail -f /var/log/mosquitto/mosquitto.log
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~$ sudo tail -f /var/log/mosquitto/mosquitto.log
1586006360: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586006360: Config loaded from /etc/mosquitto/mosquitto.conf.
1586006360: Opening ipv4 listen socket on port 1883.
1586006360: Opening ipv6 listen socket on port 1883.
1586008161: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
```

- Where is the default location of your config file ? and what is the folder conf.d used for ?

Default location of config file is “/etc/mosquitto/mosquitto.conf”.

conf.d folder is used to put all the local configuration files in this directory. So that when a mosquitto service starts it will load all the local config settings from this directory.

For example: a users.txt(list of users), and a certificate can put under this dir to load at the start of the service.

- What Linux command should I use to understand the content of the file and parameters

CAT command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

- Change the listening port from the default one to 9999 and reload your mosquitto app and use lsof to check that it works

It can be done by two ways. First we can put the “port 9999” in mosquitto.conf file to initiate the port as default.

Another way is to initiate the mosquitto service by using command “mosquitto -p 9999”. Snippet of the same is shown below

```
prashant@prashant-Lenovo-ideapad-530S-14IK8:~$ mosquitto -p 9999
1586009948: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586009948: Using default config.
1586009948: Opening ipv4 listen socket on port 9999.
1586009948: Opening ipv6 listen socket on port 9999.
```

Writing first configuration file following these steps:

- Find the complete example file in the documentation of your system

File is present under /usr/share/doc/mosquitto/examples/mosquitto.conf.example

- Put the smallest set of instructions to start mosquitto correctly. Take inspiration from the default file

Put the Mqtt protocol in effect. Change the port no to 9999 and back to default.

Change the default file log location to home directory, also changed the log format to **stdout** : terminal printing

Below is the edited custom config file.

```
File Edit View Search Terminal Help
GNU nano 2.9.3
## Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

protocol mqtt

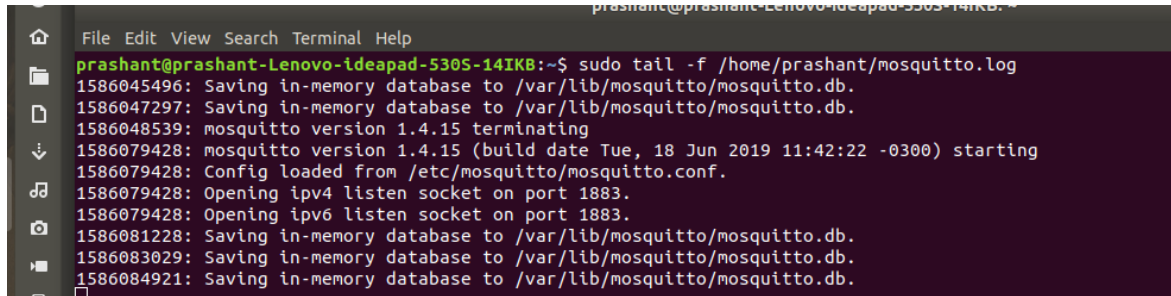
log_dest stdout

log_dest file /home/prashant/mosquitto.log

include_dir /etc/mosquitto/conf.d
```

- Change the default location for logging

Default logging location is changed in config file by changing the location. We can see below that now using sudo tail with new logging location

A terminal window with a dark background and light text. The title bar shows 'prashant@prashant-Lenovo-Ideapad-530S-14IKB'. The menu bar includes 'File Edit View Search Terminal Help'. The prompt is 'prashant@prashant-Lenovo-Ideapad-530S-14IKB:~\$'. The command entered is 'sudo tail -f /home/prashant/mosquitto.log'. The output shows several log messages: '1586045496: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.', '1586047297: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.', '1586048539: mosquitto version 1.4.15 terminating', '1586079428: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting', '1586079428: Config loaded from /etc/mosquitto/mosquitto.conf.', '1586079428: Opening ipv4 listen socket on port 1883.', '1586079428: Opening ipv6 listen socket on port 1883.', '1586081228: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.', '1586083029: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.', and '1586084921: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.'

```
prashant@prashant-Lenovo-Ideapad-530S-14IKB:~$ sudo tail -f /home/prashant/mosquitto.log
1586045496: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
1586047297: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
1586048539: mosquitto version 1.4.15 terminating
1586079428: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586079428: Config loaded from /etc/mosquitto/mosquitto.conf.
1586079428: Opening ipv4 listen socket on port 1883.
1586079428: Opening ipv6 listen socket on port 1883.
1586081228: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
1586083029: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
1586084921: Saving in-memory database to /var/lib/mosquitto/mosquitto.db.
```

Use the command line

Discover

- What are all the tools linked to mosquitto that we installed earlier

We installed following packages

Paho provides a client class which enable applications to connect to an MQTT broker to publish messages, and to subscribe to topics and receive published messages.

Pyserial: This module provides serial connection over serial ports required to publish and subscribe the message by PAHO MQTT

Openssl: this is used for the secure connection by identifying device ids.

- If I want to publish a message, which command do I use? same question if I want to subscribe to a topic?

Mosquitto_pub and Mosquitto_sub command is used for publishing and subscribing to a topic with mosquitto client broker service.

- What is the mosquitto_passwd used for?

mosquitto_passwd is a tool for managing password files for the mosquitto MQTT broker.

Action

- Start the mosquitto broker using your configuration file?

Initiating by command “mosquitto -c /etc/mosquitto/mosquitto.conf -v”

- Publish a random value in the broker. Decide of the topic you want (example: myTopic)?

Initiated broker by mosquitto

Publish a random message “test_message_random” with topic test

Command used : mosquitto_pub -m “message” -t “topic”

Below output shows, message to broker and subscriber

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto
1586096666: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586096666: Using default config.
1586096666: Opening ipv4 listen socket on port 1883.
1586096666: Opening ipv6 listen socket on port 1883.
1586096872: New connection from 127.0.0.1 on port 1883.
1586096872: New client connected from 127.0.0.1 as mosqpub|13155-prashant-(c1, k60).
1586096872: Client mosqpub|13155-prashant- disconnected.
1586096936: New connection from 127.0.0.1 on port 1883.
1586096936: New client connected from 127.0.0.1 as mosqpub|13180-prashant-(c1, k60).
1586096936: Client mosqpub|13180-prashant- disconnected.
1586097394: New connection from 127.0.0.1 on port 1883.
1586097394: New client connected from 127.0.0.1 as mosqpub|13450-prashant-(c1, k60).
1586097394: Client mosqpub|13450-prashant- disconnected.
1586097442: New connection from 127.0.0.1 on port 1883.
1586097442: New client connected from 127.0.0.1 as mosqpub|13475-prashant-(c1, k60).
1586097442: Client mosqpub|13475-prashant- disconnected.
[]

1: mosquitto, bash
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto pub -m test_message_random -t test -d
Client mosqpub|13475-prashant- sending CONNECT
Client mosqpub|13475-prashant- received CONNACK
Client mosqpub|13475-prashant- sending PUBLISH (d0, q0, r0, m1, 'test', .. (19 bytes))
Client mosqpub|13475-prashant- sending DISCONNECT
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto_sub -t test
[]

1: mosquitto_sub, mos
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto
1586097677: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586097677: Using default config.
1586097677: Opening ipv4 listen socket on port 1883.
1586097677: Opening ipv6 listen socket on port 1883.
1586097788: New connection from 127.0.0.1 on port 1883.
1586097788: New client connected from 127.0.0.1 as mosqsub|13683-prashant-(c1, k60).
[]
```

- Subscribe to the topic you created before. Have you received that message? why?

Yes, after subscribing the topic I have received the message published by publisher to the same topic.

This happens because publisher published the info to the topic to broker and when subscriber subscribe to that topic broker immediately provides the message to subscriber related to that topic (given time to live condition of message, in my case it is 2 min)

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto sub -t test
prashant_message
prashant_message
pr
prashant_testing
2815
[]

(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto pub -t test -m $RANDOMt_testing
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$
```

- Let's publish another random value. What happened at the subscriber side?

It's just prints a new random value on subscriber side. But on mosquitto terminal we can observe that a new client connects every time and disconnect just after publishing. Shown on above screenshot.

- Let's stop the broker. What happened to previous connections? what if a new connection is attempted.

After stopping the broker, we could not able to publish a message because it is now not able to get the Acknowledgement correspond to successful connection.

- Restart the broker with the `-v` switch in your command line. Replay the scenario of before and see the logs.

After restarting it automatically connects to the subscriber topic created before. As we used a Verbose command so it will show in details about connection. Below is the output snippet

```
(env) (base) prashant@prashant-Lenovo-ideapad-530
ing/iotprashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto -v
1586098628: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586098628: Using default config.
1586098628: Opening ipv4 listen socket on port 1883.
1586098628: Opening ipv6 listen socket on port 1883.
1586098628: New connection from 127.0.0.1 on port 1883.
1586098628: New client connected from 127.0.0.1 as mosqsub|13683-prashant- (c1, k60).
1586098628: Sending CONNACK to mosqsub|13683-prashant- (0, 0)
1586098628: Received SUBSCRIBE from mosqsub|13683-prashant-
1586098628:      test (QoS 0)
1586098628: mosqsub|13683-prashant- 0 test
1586098628: Sending SUBACK to mosqsub|13683-prashant-
1586098688: Received PINGREQ from mosqsub|13683-prashant-
1586098688: Sending PINGRESP to mosqsub|13683-prashant-
[]
```

- What happens after one minute in the logs of the broker? can you explain using the MQTT protocol? how do I set a personalized value for this parameter ? (say 2 minutes).

As in the previous question screenshot we can see that Subscriber is trying to reach broker in every one minute and after that it discarded the message as according to MQTT protocol. IN MQTT protocol messages expires if they cannot be delivered within a user-defined period of time.

We can change the value to 2 min by using below function

```
connect(host, port=1883, keepalive=60, bind_address="")
```

Also, this value can be fixed in mosquitto.conf file while starting the broker according to configurations.

- What are the consequences of this behaviors on battery constrained devices?

It will take more power as we can see that it takes 2 minutes to expire the message and wait. So definitely it keeps alive a connection longer putting more strain on battery.

- What is the difference at the broker side (in the logs) when we publish messages with different QoS levels?

Messages published with a QoS of 1 and 2 are acknowledged by the server. This results in several messages being sent.

Messages published with a QoS of 0 require only 1 message., and are not acknowledged by the server. That's why we only see a single message with no Mid in terminal.

Published messages with a QoS of 1 or 2 also have a Message ID number which can be used to track the message.

Below Screenshot shows the message published with qos 0, qos 1, and qos 2

```
1586099837: New connection from 127.0.0.1 on port 1883.
1586099837: New client connected from 127.0.0.1 as mosqpub|14766-prashant- (c1, k60).
1586099837: Sending CONNACK to mosqpub|14766-prashant- (0, 0)
1586099837: Received PUBLISH from mosqpub|14766-prashant- (d0, q2, r0, m1, 'test', ... (5 bytes))
1586099837: Sending PUBREC to mosqpub|14766-prashant- (Mid: 1)
1586099837: Received PUBREL from mosqpub|14766-prashant- (Mid: 1)
1586099837: Sending PUBCOMP to mosqpub|14766-prashant- (Mid: 1)
1586099837: Sending PUBLISH to mosqsub|13683-prashant- (d0, q0, r0, m0, 'test', ... (5 bytes))
1586099837: Received DISCONNECT from mosqpub|14766-prashant-
1586099837: Client mosqpub|14766-prashant- disconnected.
1586099853: New connection from 127.0.0.1 on port 1883.
1586099853: New client connected from 127.0.0.1 as mosqpub|14767-prashant- (c1, k60).
1586099853: Sending CONNACK to mosqpub|14767-prashant- (0, 0)
1586099853: Received PUBLISH from mosqpub|14767-prashant- (d0, q2, r0, m1, 'test', ... (5 bytes))
1586099853: Sending PUBREC to mosqpub|14767-prashant- (Mid: 1)
1586099853: Received PUBREL from mosqpub|14767-prashant- (Mid: 1)
1586099853: Sending PUBCOMP to mosqpub|14767-prashant- (Mid: 1)
1586099853: Sending PUBLISH to mosqsub|13683-prashant- (d0, q0, r0, m0, 'test', ... (5 bytes))
1586099853: Received DISCONNECT from mosqpub|14767-prashant-
1586099853: Client mosqpub|14767-prashant- disconnected.
1586099889: Received PINGREQ from mosqsub|13683-prashant-
1586099889: Sending PINGRESP to mosqsub|13683-prashant-
1586099949: Received PINGREQ from mosqsub|13683-prashant-
1586099949: Sending PINGRESP to mosqsub|13683-prashant-
1586099978: New connection from 127.0.0.1 on port 1883.
1586099978: New client connected from 127.0.0.1 as mosqpub|14803-prashant- (c1, k60).
1586099978: Sending CONNACK to mosqpub|14803-prashant- (0, 0)
1586099978: Received PUBLISH from mosqpub|14803-prashant- (d0, q1, r0, m1, 'test', ... (4 bytes))
1586099978: Sending PUBACK to mosqpub|14803-prashant- (Mid: 1)
1586099978: Sending PUBLISH to mosqsub|13683-prashant- (d0, q0, r0, m0, 'test', ... (4 bytes))
1586099978: Received DISCONNECT from mosqpub|14803-prashant-
1586099978: Client mosqpub|14803-prashant- disconnected.
```

Setup of basic auth and more

Action

- Create two users with credentials in a file named users.txt. Use this file in your configuration.

Using the command “mosquitto_passwd -c users.txt prashant_user1” than it will ask to enter the password for user.

Below terminal snippet shows all the necessary steps to create two users with password using hashing.

```

ca-certificates Certs com:0 Mosquitto.com Mosquitto.log
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto$ cd conf.d
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ ls
README
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ mosquitto_passwd -c users.txt prashant_user1
Password:
Reenter password:
Error: Unable to open file users.txt for writing. Permission denied.
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ sudo mosquitto_passwd -c users.txt prashant_user1
[sudo] password for prashant:
Password:
Reenter password:
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ sudo mosquitto_passwd -c users.txt prashant_user2
Password:
Reenter password:
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ ls
README users.txt
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ sudo cat users.txt
prashant_user2:$6$UjPhZTF8Dodqqwh3$etkFhn3aaRPYv6XuWpT65IwhooDVTk1+YWHapu8A37/wh01bg6VC79dN4K5avZ9L21HPQIvw5fHqfdeVWCL1Ig==
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ sudo mosquitto_passwd -b users.txt prashant_user1 pdhillon
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$ sudo cat users.txt
prashant_user2:$6$UjPhZTF8Dodqqwh3$etkFhn3aaRPYv6XuWpT65IwhooDVTk1+YWHapu8A37/wh01bg6VC79dN4K5avZ9L21HPQIvw5fHqfdeVWCL1Ig==
prashant_user1:$6$Ab4MFpWL7vAZHQVv$DKTb89iNXFPsw7HFwhPdkWM637UbeB0mAL2toB90SrWmAaDHqXCMo+kDXoEX1POVQeBicR6owYj5j3GMq7gHbw==
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:/etc/mosquitto/conf.d$

```

- Try to connect like you did previously. Does it work? what is the purpose of the parameter `allow_anonymous` in the file and what is the default value of that.

Default value of `allow_anonyms` is `False`. No it do not connect normally. Now we have to provide username and password to connect to broker. Below snippet shows the procedure for publisher, subscriber and broker.

```

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ sudo service mosquitto stop
[sudo] password for prashant:
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto -c /etc/mosquitto/manual.conf
1586103932: mosquitto version 1.4.15 (build date Tue, 18 Jun 2019 11:42:22 -0300) starting
1586103932: Config loaded from /etc/mosquitto/manual.conf
1586103932: Opening ipv4 listen socket on port 9999.
1586103932: Opening ipv6 listen socket on port 9999.
1586104185: New connection from 127.0.0.1 on port 9999.
1586104185: New client connected from 127.0.0.1 as mosqsub|18022-prashant- (c1, k60, u'prashant').
1586104272: New connection from 127.0.0.1 on port 9999.
1586104272: New client connected from 127.0.0.1 as mosqpub|18074-prashant- (c1, k60, u'prashant').
1586104272: Client mosqpub|18074-prashant- disconnected
^C
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto_sub -t test -u prashant -P pdhillon
^Z
[2]+  Stopped mosquitto_sub -t test -u prashant -P pdhillon
prashant -P pdhillon
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto_sub -t test -u prashant -P pdhillon
this is a message from USER prashant
^C
(base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ source /home/prashant/Documents/wirelessNetworks/training/iot/env/bin/activate
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto_pub -t test -m "this is a message from USER prashant" -u prashant -P pdhillon
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ mosquitto_pub -p 9999 -t test -m "this is a message from USER prashant" -u prashant -P pdhillon
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$

```

- How do I see the logs of my program? where is the default location?

After changing of default location of my log file in start of TP now the location is `/home/prashant/mosquitto.log`

We can see the log of program by two methods, one is interactive :

`Sudo tail -f /home/prashant/mosquitto.log` by this way we can see the log info on terminal

Another way is `sudo cat /home/prashant/mosquitto.log`

- All my logs start with this `1582912839`: what does it mean? how do I change it to something human-readable?

Normally by default it logs into Timestamp. But we can change its format into human readable by following command:

`log_timestamp_format %Y-%m-%dT%H:%M:%S`

Writing our client program

Writing your publish and subscribe programs

- Start the mosquitto broker with a minimal and default configuration (i.e. no auth or certificates)
- Write a small program that uses argparse and accepts default values that you can change for the broker address and port, a topic name, and a message to be sent

In below terminal snippet we can see the command using argparse to pass the port no, topic and message.

```
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ python test_subscribe.py -p 9999 -t test
Connected to server with code 0
Connection Accepted.
test b'prashant:message_receive_by_subscriber_check'
█
```

```
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ python test_publish.py -p 9999 -t test -m prashant:message_receive_by_subscriber_check
All done publishing mid: 1
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ █
```

- Write your first subscriber using the examples in the documentation. This program should be:
 - capable of changing the connection options (address, port, topic) from the CLI
 - shows the message that he receives from the broker (use mosquitto_pub to publish messages)
 - loops forever and gracefully disconnect when you type CTRL+C on the keyboard

Using argparse in the test_subscribe.py file I am taking the port no, topic and qos information. Below is the output shows

```
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ python test_subscribe.py -p 9999 -t test
Connected to server with code 0
Connection Accepted.
test b'prashant:message_receive_by_subscriber_check' 0
█
```

- Write your first publish program with the same expectations as for the subscriber.

Using argparse in the test_subscribe.py file I am taking the port no, topic and qos information. Below is the output shows

```
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ python test_publish.py -p 9999 -t test -m prashant:message_receive_by_subscriber_check --qos 0
All done publishing mid: 1
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ █
```

Last will and reconnection

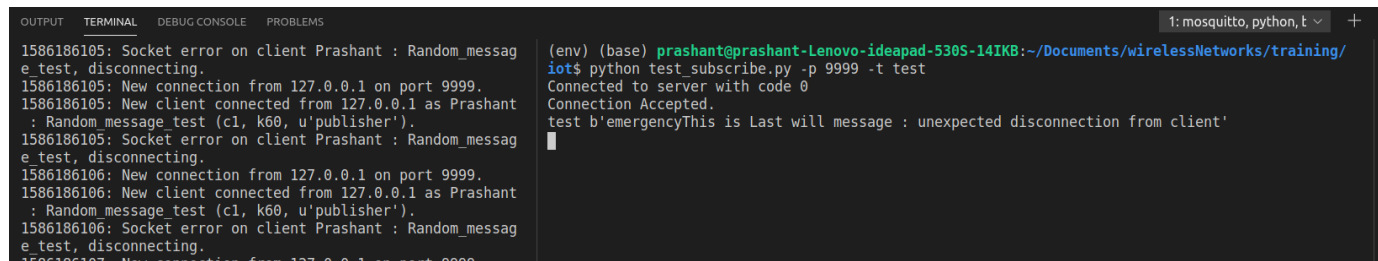
Action

- Implement the option of last will as shown in the documentation

client.will_set("This is last will: sudden disconnection", qos=0,retain=True)

- Implement a failure scenario where this message is sent.

I achieved the connection break simply by closing the python IDE for the publisher client. When there is a sudden disconnection from publisher to broker, broker sends the last will to subscribers of that topic as set by publisher. Below is last will implementation



```

1586186105: Socket error on client Prashant : Random_message_test, disconnecting.
1586186105: New connection from 127.0.0.1 on port 9999.
1586186105: New client connected from 127.0.0.1 as Prashant : Random_message_test (c1, k60, u'publisher').
1586186105: Socket error on client Prashant : Random_message_test, disconnecting.
1586186106: New connection from 127.0.0.1 on port 9999.
1586186106: New client connected from 127.0.0.1 as Prashant : Random_message_test (c1, k60, u'publisher').
1586186106: Socket error on client Prashant : Random_message_test, disconnecting.
1586186107: New connection from 127.0.0.1 on port 9999.
(env) (base) prashant@prashant-Lenovo-ideapad-530S-14IKB:~/Documents/wirelessNetworks/training/iot$ python test_subscribe.py -p 9999 -t test
Connected to server with code 0
Connection Accepted.
test b'emergencyThis is Last will message : unexpected disconnection from client'

```

Setting up options for QoS 1 and QoS 2

Action

- Implement the QoS levels for your publisher messages
- Watch the log in the broker and find the messages of each QoS level

Using the argparse I have added the qos option to the PUB/Sub system.

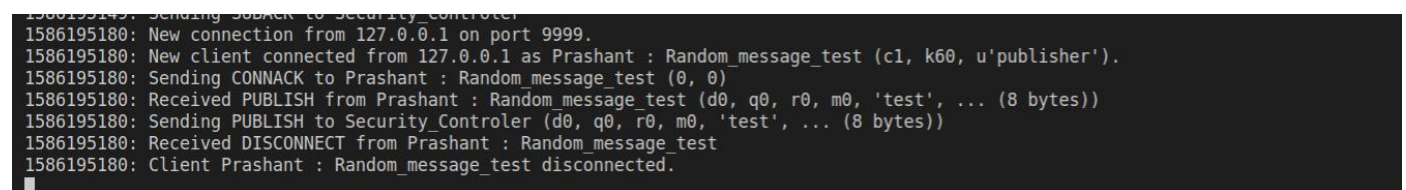
MQTT provides 3 QOS levels-

QOS 0 – Once (not guaranteed)

QOS 1 – At Least Once (guaranteed)

QOS 2 – Only Once (guaranteed)

I have published the message using qos -0,1,2 and observed the output in mqtt log. Below are my observations which stands correct according to theory.

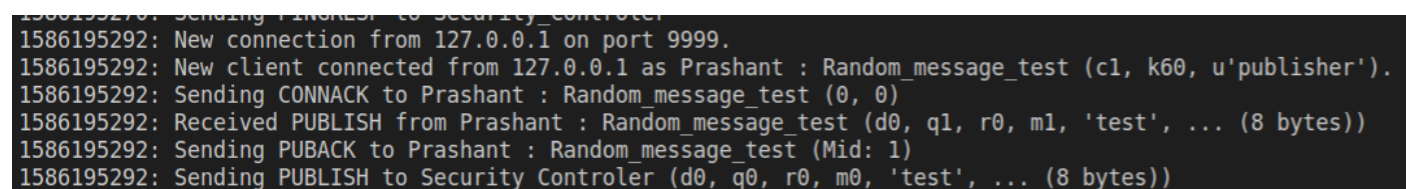


```

1586195147: Sending SUBACK to Security_Controller
1586195180: New connection from 127.0.0.1 on port 9999.
1586195180: New client connected from 127.0.0.1 as Prashant : Random_message_test (c1, k60, u'publisher').
1586195180: Sending CONNACK to Prashant : Random_message_test (0, 0)
1586195180: Received PUBLISH from Prashant : Random_message_test (d0, q0, r0, m0, 'test', ... (8 bytes))
1586195180: Sending PUBLISH to Security_Controller (d0, q0, r0, m0, 'test', ... (8 bytes))
1586195180: Received DISCONNECT from Prashant : Random_message_test
1586195180: Client Prashant : Random_message_test disconnected.

```

QOS -0



```

1586195270: Sending PINGREQ to Security_Controller
1586195292: New connection from 127.0.0.1 on port 9999.
1586195292: New client connected from 127.0.0.1 as Prashant : Random_message_test (c1, k60, u'publisher').
1586195292: Sending CONNACK to Prashant : Random_message_test (0, 0)
1586195292: Received PUBLISH from Prashant : Random_message_test (d0, q1, r0, m1, 'test', ... (8 bytes))
1586195292: Sending PUBACK to Prashant : Random_message_test (Mid: 1)
1586195292: Sending PUBLISH to Security_Controller (d0, q0, r0, m0, 'test', ... (8 bytes))

```

QOS 1

```

1586195773: New connection from 127.0.0.1 on port 9999.
1586195773: New client connected from 127.0.0.1 as Prashant : Random_message_test (c1, k60, u'publisher').
1586195773: Sending CONNACK to Prashant : Random_message_test (0, 0)
1586195773: Received PUBLISH from Prashant : Random_message_test (d0, q2, r0, m1, 'test', ... (8 bytes))
1586195773: Sending PUBREC to Prashant : Random_message_test (Mid: 1)
1586195794: Sending PUBREC to Prashant : Random_message_test (Mid: 1)

```

QOS 2

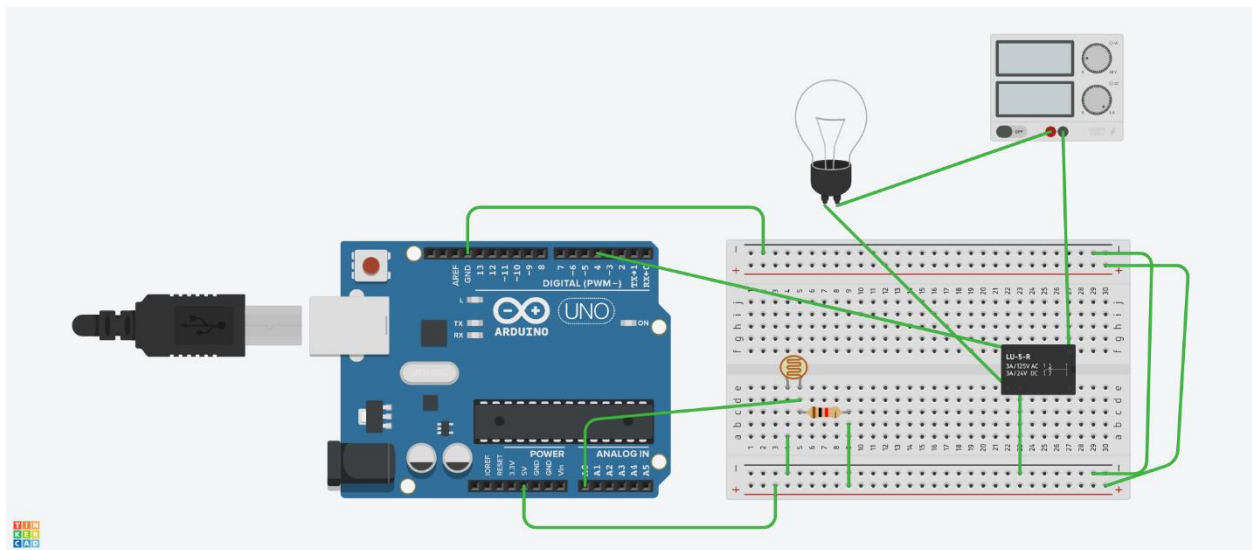
Using Arduino Kit:

I have used the online simulating platform Tinkercad. I have simulated a **Arduino Relay activated Lamp**.

Components used for this experiment are:

Name	Quantity	Component
U1	1	Arduino Uno R3
K1	1	Relay SPDT
L1	1	Light bulb
P1	1	5 , 5 Power Supply
R3	1	1 k Ω Resistor
R1	1	Photoresistor

Connected circuit:

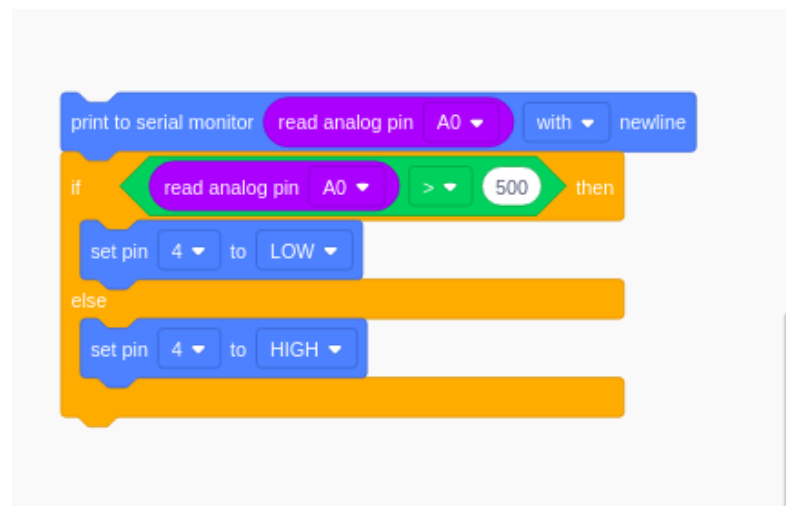


Explanation:

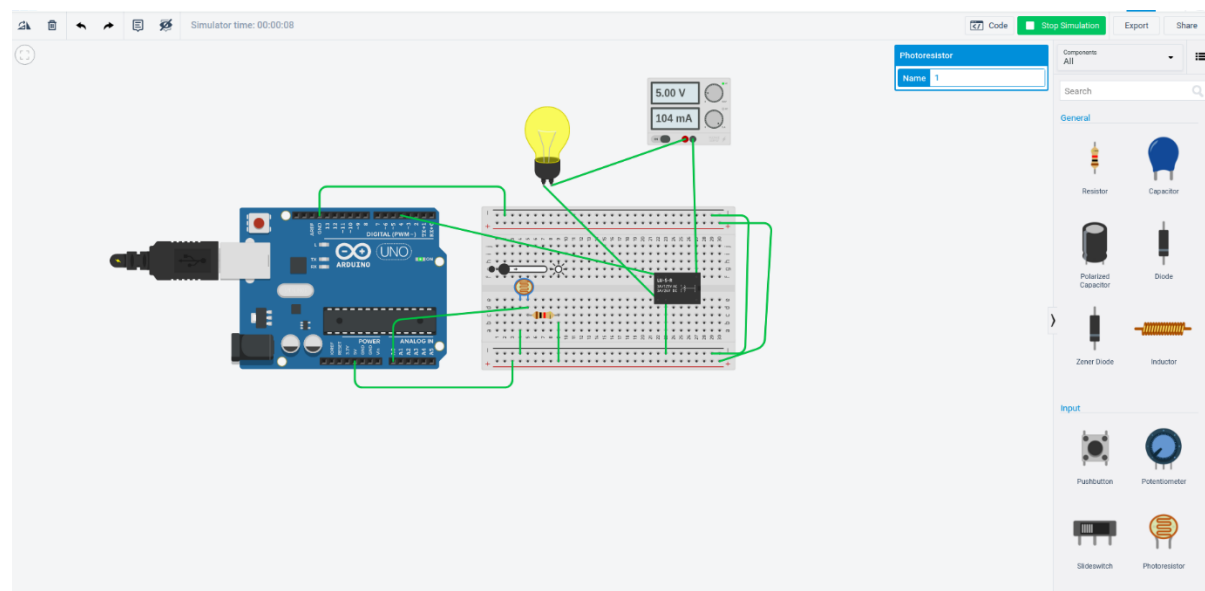
Basically, a relay works as a switch whereby it can be either open or closed. The relay uses electromagnetism from small voltage to provide higher voltages. The connection for the LDR sensor is connected as per the image above. One end is connected to the Ground and the other side is connected to a resistor and from the resistor to VCC. The output signal from the LDR is tapped between the LDR leg and the resistor leg.

Code Block Logic:

The coding part for this is pretty straightforward. I have used blocks for coding. The first line shows that we read the input from the analog pin A0 and print it to the serial monitor. Next, we do the conditional formatting, whereby we check the value of A0. If the value of A0 is equal or more than 500 it sets the digital pin 4 to LOW, and if the value is less it sets the pin 4 to HIGH. The pin 4 is connected to the relay.



Output:



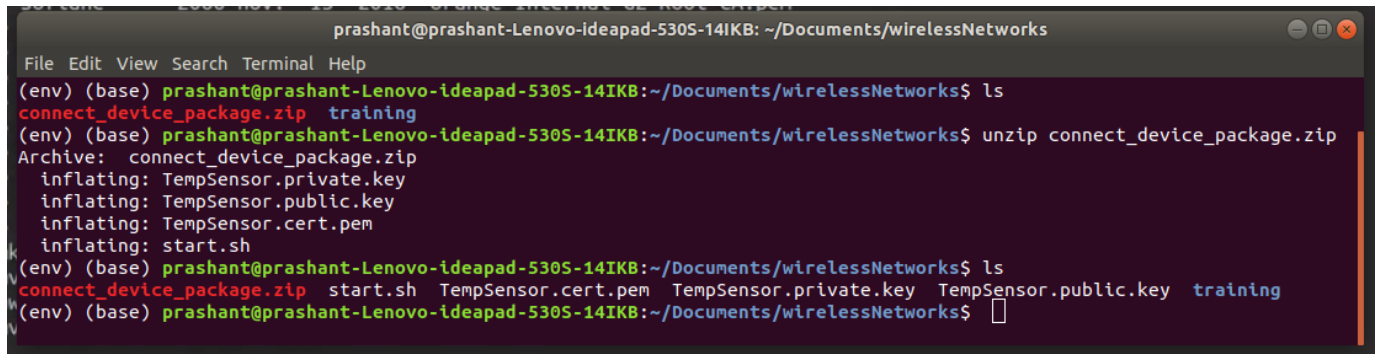
Here is the link to simulate in live environment: <https://www.tinkercad.com/things/9HRcdoM8VcJ>

AWS:

Following your tutorial, I have tried to reproduce the same.

I created an IOT service called TempSensor over AWS. That corresponds to my temperature sensor. After creation of this cloud service I downloaded the linux python SDK kit to have dummy scripts.

File contains

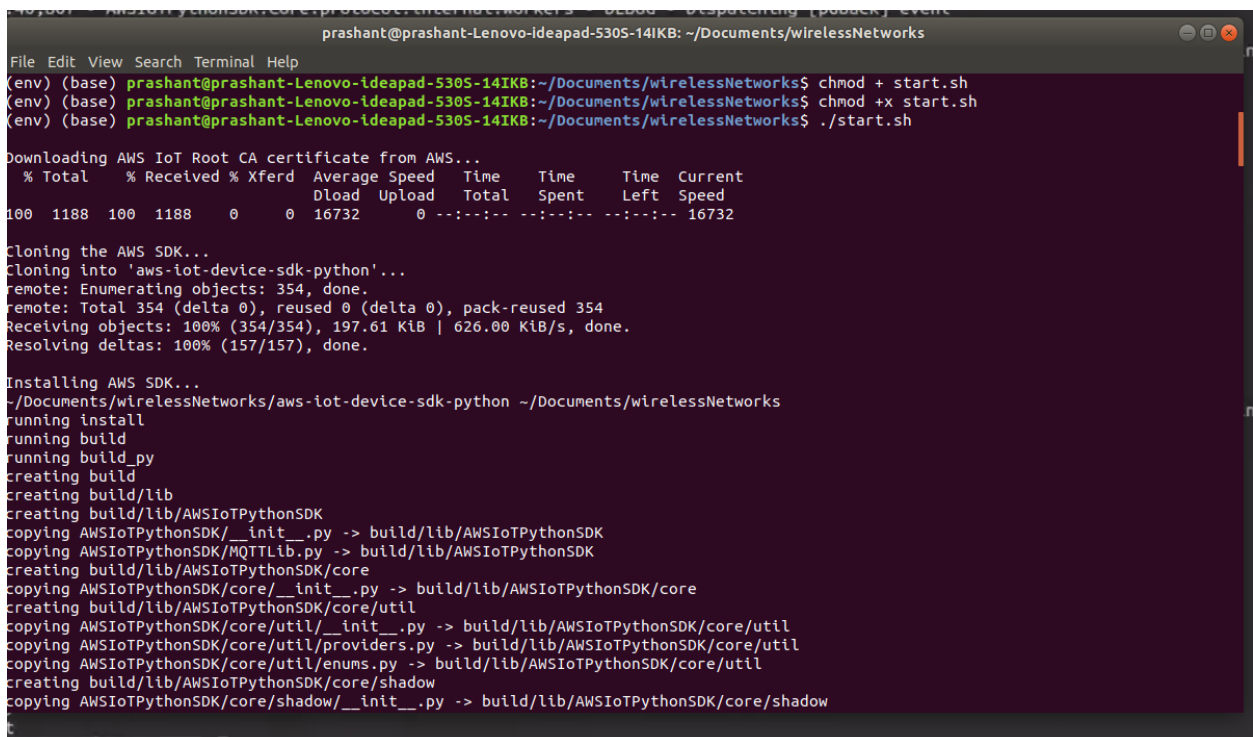


```
prashant@prashant-Lenovo-Ideapad-530S-14IKB: ~/Documents/wirelessNetworks
File Edit View Search Terminal Help
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$ ls
connect_device_package.zip  training
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$ unzip connect_device_package.zip
Archive:  connect_device_package.zip
  inflating: TempSensor.private.key
  inflating: TempSensor.public.key
  inflating: TempSensor.cert.pem
  inflating: start.sh
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$ ls
connect_device_package.zip  start.sh  TempSensor.cert.pem  TempSensor.private.key  TempSensor.public.key  training
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$
```

TempSensor.cert.pem contains certificate used for my device.

Other two files are private and public key for secure communication to my device and cloud console.

Script file i.e, start.sh runs and download all the required setting to connect to cloud service. It git download the python sdk kit. Also verify the device certificate generated by AWS for our device on initial package download. Below is the snippet for that



```
prashant@prashant-Lenovo-Ideapad-530S-14IKB: ~/Documents/wirelessNetworks
File Edit View Search Terminal Help
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$ chmod +x start.sh
(env) (base) prashant@prashant-Lenovo-Ideapad-530S-14IKB:~/Documents/wirelessNetworks$ ./start.sh

Downloading AWS IoT Root CA certificate from AWS...
% Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left     Speed
100 1188    100 1188    0     0  16732      0  --:--:-- --:--:-- --:--:-- 16732

Cloning the AWS SDK...
Cloning into 'aws-iot-device-sdk-python'...
remote: Enumerating objects: 354, done.
remote: Total 354 (delta 0), reused 0 (delta 0), pack-reused 354
Receiving objects: 100% (354/354), 197.61 KiB | 626.00 KiB/s, done.
Resolving deltas: 100% (157/157), done.

Installing AWS SDK...
~/Documents/wirelessNetworks/aws-iot-device-sdk-python ~/Documents/wirelessNetworks
running install
running build
running build_py
creating build
creating build/lib
creating build/lib/AWSIoTPythonSDK
copying AWSIoTPythonSDK/__init__.py -> build/lib/AWSIoTPythonSDK
copying AWSIoTPythonSDK/MQTTLib.py -> build/lib/AWSIoTPythonSDK
creating build/lib/AWSIoTPythonSDK/core
copying AWSIoTPythonSDK/core/__init__.py -> build/lib/AWSIoTPythonSDK/core
creating build/lib/AWSIoTPythonSDK/core/util
copying AWSIoTPythonSDK/core/util/__init__.py -> build/lib/AWSIoTPythonSDK/core/util
copying AWSIoTPythonSDK/core/util/providers.py -> build/lib/AWSIoTPythonSDK/core/util
copying AWSIoTPythonSDK/core/util/enums.py -> build/lib/AWSIoTPythonSDK/core/util
creating build/lib/AWSIoTPythonSDK/core/shadow
copying AWSIoTPythonSDK/core/shadow/__init__.py -> build/lib/AWSIoTPythonSDK/core/shadow
```

Moreover, this file downloads the IOT root certificate for the system.

At last this script runs the following command to run MQTT service to connect and communicate to AWS cloud.

```
“python aws-iot-device-sdk-python/samples/basicPubSub/basicPubSub.py -e a240ahocu5uc2l-ats.iot.us-west-2.amazonaws.com -r root-CA.crt -c TempSensor.cert.pem -k TempSensor.private.key”
```

As we have learned earlier about all the concepts of Mqtt and usage of argparse. This basicPubSub.py uses argparse to take input about broker address, topic , message and end point information. End point is defined by AWS for our service.

Below is the communication shown between cloud and our device (i.e, terminal in our case)

The screenshot displays the AWS IoT console's 'Publish' interface and a terminal window. The console shows a topic 'sdk/test/Python' and a message '{"message": "Prashant AWS Console testing"}'. Below this, the terminal window shows the execution of the 'basicPubSub.py' script, which successfully publishes the message to the specified topic.

AWS IoT Console - Publish

Specify a topic and a message to publish with a QoS of 0.

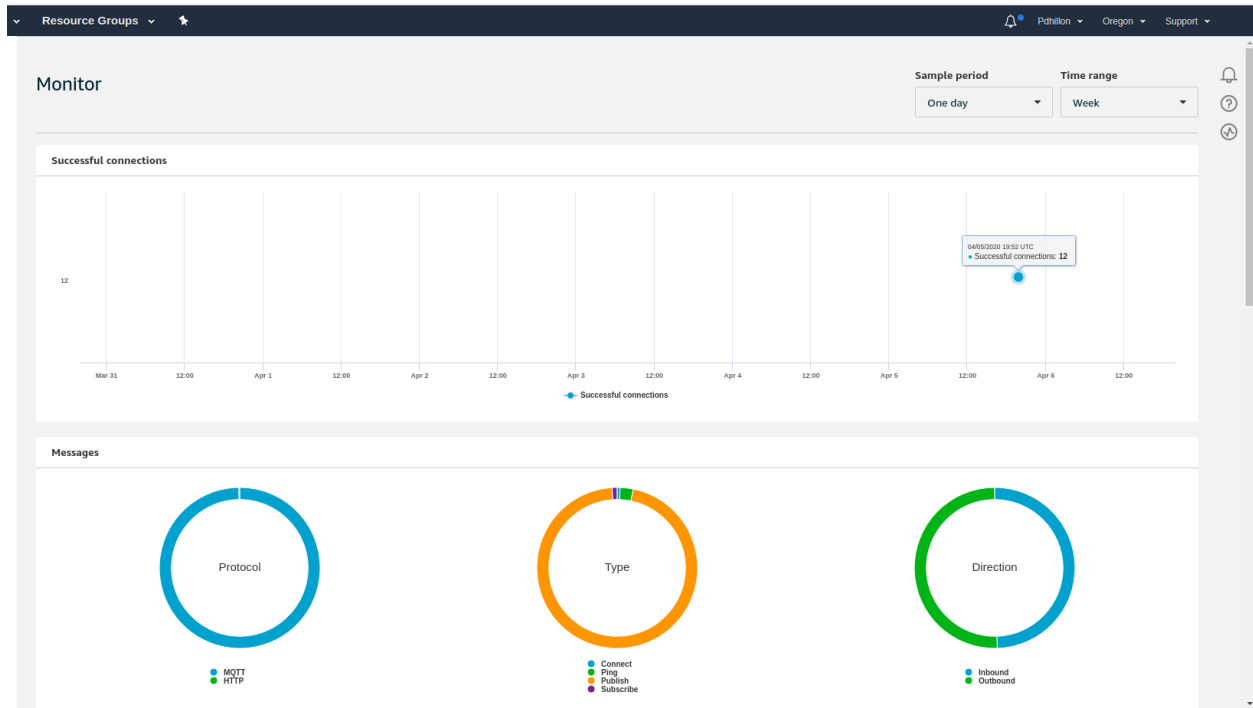
Topic: `sdk/test/Python` [Publish to topic]

Message: `{ "message": "Prashant AWS Console testing" }`

Terminal Output

```
prashant@prashant-Lenovo-Ideapad-530S-14IKB: ~/Documents/wirelessNetworks
File Edit View Search Terminal Help
sdk/test/Python
-----
2020-04-06 21:47:19,614 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
2020-04-06 21:47:19,726 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2020-04-06 21:47:19,727 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
Received a new message:
b'\n "message": "Prashant AWS Console testing"\n'
from topic:
sdk/test/Python
-----
```


AWS Dashboard view:



Also, AWS provides options like, Shadow is a previous version of data over AWS cloud. This helps prevent request to battery limited sensors to get data. We can use Tunnel for remote dubbing.