介紹如何在 Ubuntu Linux 中安裝與使用 Mosquitto 伺服器，並設定帳號、密碼與 SSL 加密傳輸。

Mosquitto (https://mosquitto.org/) 是一個 Eclipse 基金會轄下開放原始碼的 MQTT broker 伺服器，屬於輕量級（lightweight）的伺服器，可用於各種規格的硬體，從普通的單板電腦（例如樹莓派）到專用的伺服器主機都能夠運行 Mosquitto。

以下我們以 Ubuntu Linux 環境作為測試環境，示範如何架設 Mosquitto 伺服器，並設定帳號密碼管控全線，以及 SSL 加密傳輸。

安裝 Mosquitto 伺服器

若在 Ubuntu Linux 中可以使用 `apt` 安裝 Mosquitto 相關套件：

```
# 安裝 Mosquitto 相關套件
sudo apt install mosquitto mosquitto-clients
```

其中 `mosquitto` 套件是主要的 Mosquitto 伺服器，而 `mosquitto-clients` 則包含 `mosquitto_pub` 與 `mosquitto_sub` 等 MQTT client 指令工具。

安裝 Mosquitto 伺服器之後，會自動啟動 `mosquitto` 服務，我們可以使用一般的 `systemctl` 指令來操控 `mosquitto` 服務：

```
# 查詢 mosquitto 服務狀態
systemctl status mosquitto

# 啟動 mosquitto 服務
sudo systemctl start mosquitto

# 停止 mosquitto 服務
sudo systemctl stop mosquitto

# 重新啟動 mosquitto 服務
sudo systemctl restart mosquitto
```

是主哪一個主題，而訂閱者則是根據主題來接收訊息。主題的命名方式類似檔案路徑，是階層式的架構，例如 `sensors/outside/temp` 或 `sensors/outside/humidity`。

若要進行 MQTT 的訊息交換測試，可以先用 `mosquitto_sub` 訂閱指定的主題，例如訂閱 `hello/world` 主題：

```
# 連線至 localhost 訂閱 hello/world 主題
mosquitto_sub -h localhost -t hello/world
```

這裡的 `-t` 參數是用來設定訂閱的主題，而 `-h` 參數則是設定 MQTT 伺服器。

接著開啟另外一個終端機，以 `mosquitto_pub` 發布訊息至同樣這一個 `hello/world` 主題，這裡使用 `-m` 參數指定要傳送的訊息，而 `-t` 與 `-h` 參數的用法都相同：

```
# 連線至 localhost 發布訊息至 hello/world 主題
mosquitto_pub -h localhost -t hello/world -m "test message"
```

當訊息發布之後，正常來說 `mosquitto_sub` 就會收到 `test message` 這一條訊息，並顯示在終端機中。

```
test message
```

若要離開 `mosquitto_sub` 可以按下 Ctrl + c 終止程式。以上就是基本的 MQTT 發布與訂閱訊息的運作方式。

永康火車站、客運站，距離台南市中心15分鐘。

設定 Mosquitto 帳號與密碼

預設的 Mosquitto 伺服器是只要連線進去就可以直接使用的，如果希望加上帳號與密碼的登入認證機制，可以使用 `mosquitto_passwd` 這個指令工具來建立 Mosquitto 的帳號，並設定密碼：

```
# 建立 myuser 帳號與密碼，儲存於 /etc/mosquitto/passwd
sudo mosquitto_passwd -c /etc/mosquitto/passwd myuser
```

在建立帳號時，會要求輸入帳號的密碼，而這裡建立好的帳號我們將其儲存於 `/etc/mosquitto/passwd` 這個檔案中。

接著開編輯 Mosquitto 設定檔 `/etc/mosquitto/conf.d/default.conf`，在這個設定檔案中指定 Mosquitto 帳號與密碼設定檔的位置，這個檔案預設應該是不存在的，建立這個檔案之後，寫入以下設定：

```
# 禁止匿名連線
allow_anonymous false

# 指定帳號與密碼設定檔位置
password_file /etc/mosquitto/passwd
```

在編輯這個設定檔時，記得在最後一行的結尾處要加上換行字元。

這時候若沒有指定登入的帳號與密碼，是無法連線至 Mosquitto 伺服器的：

```
# 連線至 localhost 訂閱 hello/world 主題
mosquitto_sub -h localhost -t hello/world
```

```
Connection error: Connection Refused: not authorised.
```

我們可以在執行 mosquitto_sub 與 mosquitto_pub 指令時，額外加上 -u 與 -P 參數來指定登入 Mosquitto 伺服器的帳號與密碼：

```
# 以帳號密碼登入 Mosquitto 並訂閱 hello/world 主題
mosquitto_sub -h localhost -t hello/world \
  -u myuser -P MY_PASSWORD

# 以帳號密碼登入 Mosquitto 並發布訊息至 hello/world 主題
mosquitto_pub -h localhost -t hello/world -m "test message" \
  -u myuser -P MY_PASSWORD
```

指令中的 MY_PASSWORD 要替換成自己設定的密碼。

問題，若在公開網路中傳送這條的資料是相當不安全的，若要解決這個問題，可以採用 SSL 加密傳輸的方式來保護資料。

申請 Let's Encrypt 的 SSL 憑證

若要啟用 Mosquitto 的 SSL 加密傳輸，首先要先準備好 SSL 憑證，如果伺服器上已經有網頁伺服器用的 SSL 憑證，這裡就可以直接給 Mosquitto 使用，若是尚未申請任何 SSL 憑證，也可以使用 Let's Encrypt 取得免費的 SSL 憑證。

參考 Certbot 官方的說明 (https://certbot.eff.org/instructions)，以 `snap` 安裝最新版的 `certbot` ：

```
# 更新 snapd 至最新版
sudo snap install core
sudo snap refresh core

# 移除舊版 certbot
sudo apt-get remove certbot

# 安裝最新版 certbot
sudo snap install --classic certbot
```

若 Ubuntu Linux 伺服器有啟用防火牆的話，要將 `80` 連接埠打開，這樣才能讓 `certbot` 取得 SSL 憑證：

```
# 開啟 Ubuntu 防火牆 80 連接埠
sudo ufw allow 80
```

接著就可以利用 `certbot` 取得免費的 SSL 憑證了：

在取得 SSL 憑證的過程中，需要輸入自己的 email 位址，並且同意使用條款，取得 SSL 憑證之後，在輸出的訊息中會有 SSL 憑證存放的路徑：

```
[略]
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/mqtt.example.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/mqtt.example.com/privkey.pem
[略]
```

這樣就完成 Let's Encrypt 的 SSL 憑證申請了。

Mosquitto SSL 加密連線設定

編輯 `/etc/mosquitto/conf.d/default.conf` 設定檔，加入以下設定：

```
# 僅允許本機使用未加密連線
listener 1883 localhost

# SSL 加密連線設定
listener 8883
certfile /etc/letsencrypt/live/mqtt.example.com/cert.pem
cafile /etc/letsencrypt/live/mqtt.example.com/chain.pem
keyfile /etc/letsencrypt/live/mqtt.example.com/privkey.pem
```

在這段設定中包含了兩個 `listener` 設定，在第一個 `listener` 設定中，我們僅允許本機使用未加密連線（未加密 MQTT 標準連接埠為 `1883`）；第二個 `listener` 則是設定對於外部的連線統一都採用 SSL 加密連線（加密的 MQTTS 標準連接埠為 `8883`），`certfile`、`cafile` 與 `keyfile` 設定則指向由 Let's Encrypt 所頒發的憑證檔案。同樣地在編輯這個設定檔時，記得在最後一行的結尾加上換行字元。

```
sudo systemctl restart mosquitto
```

若有啟用防火牆的話，記得開啟 MQTTS 所使用的 `8883` 連接埠：

```
# 開啟 Ubuntu 防火牆 8883 連接埠
sudo ufw allow 8883
```

設定完成 Mosquitto 的 SSL 加密連線之後，就可以採用加密的 MQTTS 來傳送資料了：

```
# 以帳號密碼、加密協定登入 Mosquitto 並訂閱 hello/world 主題
mosquitto_sub -h mqtt.example.com -t hello/world \
  -u myuser -P MY_PASSWORD -p 8883 --capath /etc/ssl/certs/

# 以帳號密碼、加密協定登入 Mosquitto 並發布訊息至 hello/world 主題
mosquitto_pub -h mqtt.example.com -t hello/world -m "test message" \
  -u myuser -P MY_PASSWORD -p 8883 --capath /etc/ssl/certs/
```

SSL 續約與 Mosquitto 服務設定

編輯 `/etc/letsencrypt/renewal/mqtt.example.com.conf` 這個 SSL 憑證續約的設定檔，在檔案的最後加上以下設定，讓 SSL 憑證更新之後，可以自動重新啟動 `mosquitto` 服務：

```
renew_hook = systemctl restart mosquitto
```

這裡 `renew_hook` 所指定的指令就是 `certbot` 更新 SSL 憑證之後會執行的指令。設定好之後，測試一下 `certbot` 更新 SSL 憑證的執行是否正常：

如果沒有發現錯誤訊息，就表示沒問題」。

MQTT 與 WebSocket

如果希望網頁應用程式可以透過 WebSocket 連線至 Mosquitto 伺服器，可以在 `/etc/mosquitto/conf.d/default.conf` 設定檔中另外加入以下設定，多開一個 WebSocket 的 `listener`：

```
# WebSocket 與 SSL 加密連線設定
listener 8083
websockets
certfile /etc/letsencrypt/live/mqtt.example.com/cert.pem
cafile /etc/letsencrypt/live/mqtt.example.com/chain.pem
keyfile /etc/letsencrypt/live/mqtt.example.com/privkey.pem
```

修改好 Mosquitto 的設定檔之後，重新啟動 `mosquitto` 系統服務：

```
# 重新啟動 mosquitto 服務
sudo systemctl restart mosquitto
```

若有啟用防火牆的話，記得開啟 WebSocket 所使用的 `8083` 連接埠：

```
# 開啟 Ubuntu 防火牆 8083 連接埠
sudo ufw allow 8083
```

如果要測試 WebSocket 的連線，可以使用 Eclipse Paho JavaScript Client (https://www.eclipse.org/paho/clients/js/utility/)。

Ubuntu 18.04 [Quickstart] (https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-mosquitto-mqtt-messaging-broker-on-ubuntu-18-04-quickstart)

- DigitalOcean：How To Use Certbot Standalone Mode to Retrieve Let's Encrypt SSL Certificates on Ubuntu 18.04 (https://www.digitalocean.com/community/tutorials/how-to-use-certbot-standalone-mode-to-retrieve-let-s-encrypt-ssl-certificates-on-ubuntu-1804)
- DigitalOcean：How to Install and Secure the Mosquitto MQTT Messaging Broker on Ubuntu 18.04 (https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-mosquitto-mqtt-messaging-broker-on-ubuntu-18-04)