# ¿Qué es MQTT?

MQTT (Message Queuing Telemetry Transport) es un protocolo M2M (machine-to-machine) ligero de mensajes que corre sobre el protocolo TCP/IP. Aunque se originó en el seno de IBM, desde 2013 es gestionado por el [consorcio sin ánimo de lucro OASIS](https://es.wikipedia.org/wiki/OASIS_(organizaci%C3%B3n)). Su versión 3.1.1 ha sido recogido bajo el estándar internacional [**ISO/IEC 20922:2016**](http://www.iso.org/iso/catalogue_detail.htm?csnumber=69466)**.**

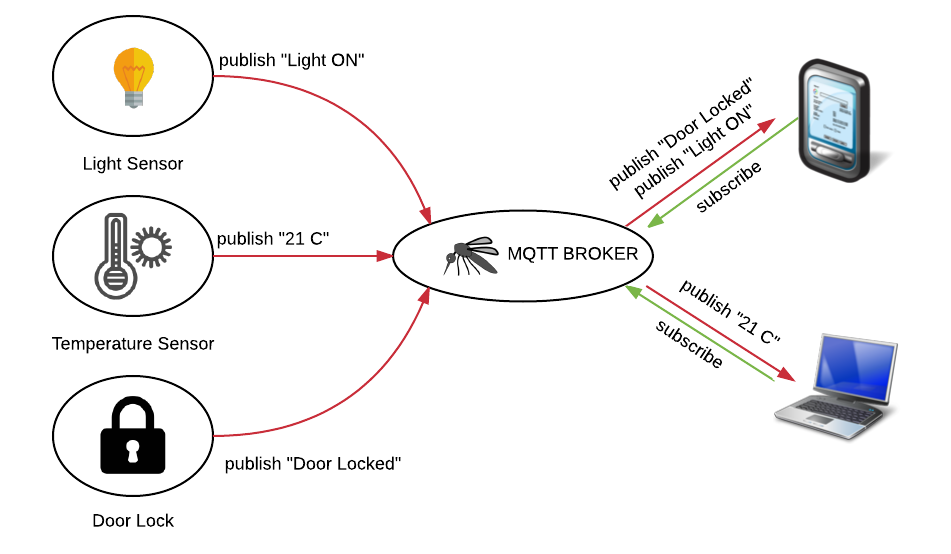
MQTT está basado en la publicación/subscripción de mensajes. Este funcionamiento requiere el uso de un bróker de mensajes, que es el encargado de la distribución de los mensajes a los diferentes clientes basándose en el tema (topic) del mismo. Actualmente, MQTT es el protocolo de conectividad para los dispositivos IoT con mayor difusión.

Los brokers exponen una interface de conexión TCP/IP, generalmente en el puerto tcp 1883, el cual ha sido [registrado por el IANA para el protocolo MQTT.](http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=1883)

Los clientes subscriptores, se conectan a ese puerto indicando el *topic* al que se subscriben y permanecen a la espera de que el broker les mande los mensajes.

Los clientes publicadores, publican el mensaje con el *topic* que especifiquen, conectándose al broker y enviándole el mensaje.

Tanto los clientes publicadores como subscriptores son conocidos como agentes (publish agent/subscribe agent); por tanto, un agente que publica puede ser un sensor de luz, y un agente que subscribe puede ser una App de nuestro móvil. El bróker revisará el *topic* del mensaje y lo reenviará hacia los agentes subscritos a él.



Existen diferentes implementaciones de brokers MQTT, siendo la más difundida “[**Mosquitto**](https://mosquitto.org/)”, gracias a su [licencia Open Source (EPL/EDL)](https://wikipedia.org/wiki/Eclipse_Public_License).

## 1.1 Jerarquías Publish/Subscribe

La técnica publish/subscribe que usa MQTT, utiliza jerarquías dentro de los topics. Las jerarquías quedan establecidas mediantes el carácter “/”. Así podemos tener el *topic*: *casa/piso1/habitacion1*. Este *topic* generará tres ramas: *casa*, *casa/piso1*, *casa/piso1/habitacion1*.

Los agentes que deseen recibir un mensaje publicado en este *topic*, podrían subscribirse a cualquiera de las tres ramas. Para poder recibir el mensaje estando subscrito a una de las dos primeras ramas, es necesario añadir el carácter reservado *“#”,* que actúa como comodín. De esta forma podemos publicar un mensaje:

**publish --topic "home/floor1/room2" --message "light on"**

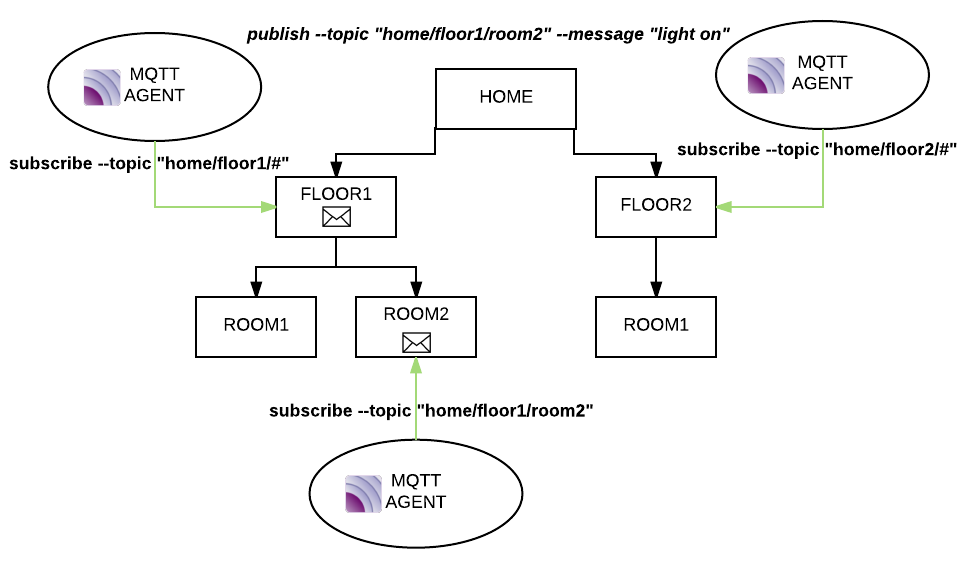
y será recibido por los agentes:

**subscribe --topic "home/floor1/room2"**

**subscribe --topic "home/floor1/#"**

pero no sería recibido por el agente:

**subscribe --topic "home/floor2/#"**



# 2. Seguridad en el broker MQTT

El protocolo MQTT dispone de unos mecanismos de seguridad muy básicos ya que su fundamento es ser ligero y poder ser implementado en cualquier dispositivo IoT. Por esta razón, la seguridad descansa sobre las protecciones que puedan existir o se puedan implementar tanto a nivel de red (uso de VPN por ejemplo), como las que el broker pueda aportar.

El broker Mosquitto dispone de diferentes elementos de seguridad para el control en la publicación/subscripción de mensajes. Todo ellos son realizados mediante el uso de un plugin de autorización/autenticación. Cada plugin define sus propios controles de seguridad.

El plugin de autenticación de Mosquitto (auth-plugin) dispone de las siguientes funcionalidades de seguridad implentadas:

## 2.1 Autenticación Usuario/Password

Este es el control más simple y sencillo de configurar. Cada agente que publica puede usar la combinación usuario/password y el broker verificará sus credenciales.

Para un "***publish agent***", esta acción se realiza en cada mensaje publicado. La misma configuración puede ser realizada en un "***subscribe agent***", aunque en esta ocasión, el broker únicamente comprobará usuario/password cuando el agente realiza la conexión y no en cada mensaje que es entregado al mismo.

## 2.2 ACLs - Access Control Lists

Con este mecanismo, se puede obtener un control total sobre las funciones publish/subscribe, mediante el filtro de los *topics*. La idea primaria de un ACL es establecer un control sobre un *topic* determinado o las ramas que de él puedan colgar si usamos comodines.

Las ACLs pueden verse como un conjunto de reglas; cuando un *topic* coincide por su filtro, se verifica el elemento a controlar (por ejemplo, si el *topic* fue publicado por un agente usando un usuario específico) y una vez evaluado el control, el broker decide si descarta el mensaje o no.

El formato de una entrada de una ACL es:

*<topic> <elemento de control>*

Como se ha comentado anteriormente, esta configuración es específica de cada broker.

Ejemplo de ACLs usando el auth-plugin de Mosquitto:

***home/floor1/room1 john***

***home/floor2/# john, paul***

Con este ejemplo, un agente usando el usuario *paul*, no podrá publicar ni subscribirse al *topic* *home/floor1/room1*, pero podrá hacerlo sobre cualquier *topic* que cuelgue de *home/floor2* (incluido ese mismo *topic*).

# 3. Plugin de Latch para Mosquitto

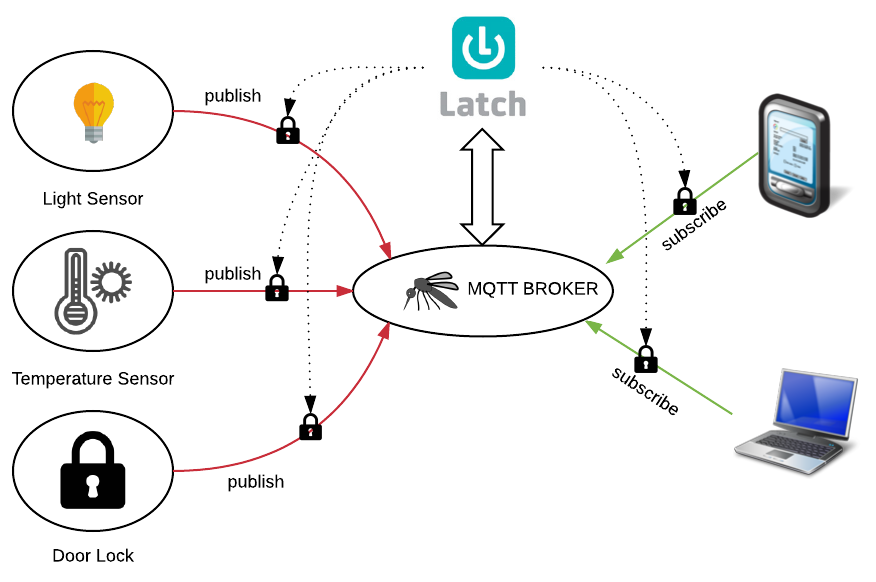
El plugin de Latch para Mosquitto nos permite añadir, de una forma fácil, un segundo factor de autorización dentro del ecosistema de IoT, realizando la integración en la plataforma (MQTT Broker), en lugar de en los dispositivos, permitiendo liberar recursos de estos, maximizando la compatibilidad y escalabilidad.

Se ha desarrollado en lenguaje Python sobre la base del auth-plugin de Mosquitto, por lo que permite utilizar tanto ACLs estándar, como autenticación usuario/password con Latch como 2FA.

El plugin de Latch opera de diferentes formas:

* Sobre las operaciones ***publish/subscribe***, permitiendo 2FA al publicar un mensaje o al conectar un subscriptor.

* Sobre los ***topics***, permitiendo añadir un 2FA de forma dinámica, mediante la utilización de **Instancias de Latch**.



## 3.1 Emparejamiento

Los usuarios del plugin que se emparejen, serán aquellos con capacidad de latchear las operaciones de publish/subscribe o un *topic*, dependiendo de la configuración que se haya realizado.

Por cada "pairing" realizado, obtendremos un *Account ID* que será el identificador del usuario en Latch. Este valor, es guardo en el fichero *latch.accounts*. Para que este proceso pueda realizarse, el usuario tiene que estar creado previamente, mediante el uso de *tools/users\_op.py* script.

El proceso de "pairing" puede realizarse de dos formas diferentes:

* Mediante el uso del script en *python: tools/pair\_op.py*. Este script nos pedirá el nombre del usuario, la password y el *token* generado por Latch App.
* Activando la opción *dynamic\_pairing*  en el fichero de configuración *latch.conf* y realizando la publicación por parte de cualquier agente sobre el *topic*:   
    
   ***LATCH/pairing=<token provided>***

De igual forma, se puede realizar el proceso de "unpairing":

* Mediante el uso del script en *python: tools/pair\_op.py*.
* Activando la opción de configuración *dynamic\_pairing*  en el fichero de configuración *latch.conf* y realizando la publicación por parte de cualquier agente sobre el *topic*:   
    
   ***LATCH/unpairing***

El ***topic LATCH/#***es un *topic* reservado por Latch plugin para la gestión de Latch. La publicación sobre este *topic*, tiene que ir acompañada de usario y password, debido a que el plugin realizará automáticamente el *pairing/unpairing* con el usuario indicado en el publish. El contenido del mensaje publicado sobre el topic *LATCH/#* es descartado.

## 3.2 Plugin de Latch para operaciones publish/subscribe

El modo más sencillo de "**latchear**" el broker MQTT, es mediante las operaciones publish y subscribe. El plugin de Latch permite controlar ambas operaciones a nivel de broker, haciendo que se verifique la autorización antes de publicar o subscribir.

Este proceso es independiente de los controles de seguridad que se hayan configurado anteriormente en el broker. De esta forma, por ejemplo, el broker MQTT puede verificar el usuario/password primero, y luego usar Latch como segundo factor de autorización para la operación que se haya solicitado (publish o subscribe), siempre que el agente esté emparejado.

Cuando un agente emparejado solicita publicar un mensaje, cada publicación se autorizará con Latch.

Para los agentes emparejados que se subscriben, se verificará con Latch cada subscripción, pero una vez que hayan sido autorizados, no se vuelve a verificar el estado del Latch.



## 3.3 Plugin de Latch para los topics

Esta es la característica más potente del plugin de Latch, permitiendo establecer un "**latch**" para cualquier *topic* de forma dinámica. De esta forma, cuando un agente emparejado publica un mensaje en un *topic* o un agente emparejado se subscribe a un *topic*, después de los controles de seguridad ya definidos en el plugin, puede realizarse una segunda verificación contra Latch. En Latch, la correspondencia a un *topic* es una instancia.

Cuando un agente emparejado publica un mensaje, cada uno de ellos, se autorizará con Latch. Para los agentes emparejados que se subscriben, se verificará con Latch cada vez que se subscriban, pero una vez que hayan sido autorizados, no se les vuelve a verificar.



El plugin de Latch permite la creación de estos ***topic "latcheables"*** o instancias, de varias formas:

* Mediante el uso del script en *Python: tools/instances\_op.py* con el que se pueden crear y borrar *topics/*instancias.
* Realizando la publicación por parte de un agente sobre el *topic*:   
    
   ***LATCH/create\_instance=<instance\_name>***   
    
  El Plugin creará automáticamente la nueva instancia con el nombre indicado y asociada al usuario con el que se realice la publicación. El contenido del mensaje publicado sobre este *topic* es descartado.
* Dinámicamente, mediante la opción ***instance\_mode=dynamic***en el archivo de configuración *latch.conf.* Activando esta opción, por cada mensaje publicado se creará una instancia nueva, si no existiera, con el nombre del *topic* usado y asociada al usuario con el que se realice la publicación.

# 4. Instalación del Plugin de Latch

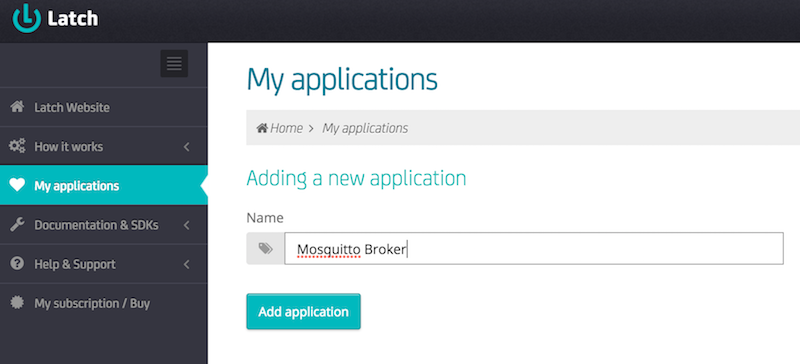
## 4.1 Ficheros del Plugin

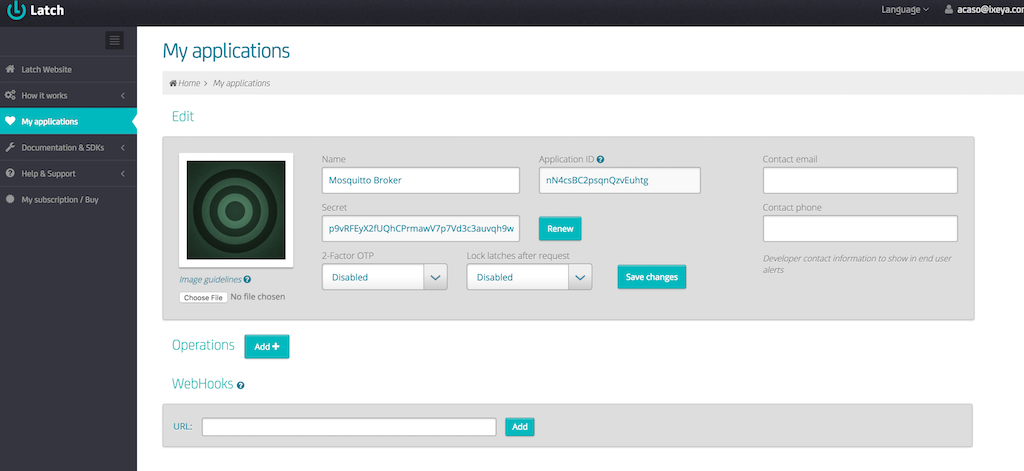
El plugin de Latch consta de los siguientes archivos:

* *mosquitto\_latch.py:* plugin de Latch para Mosquitto
* *mosquitto\_latch\_bag.py:* librería del plugin de Latch para Mosquitto
* *latch.conf: archivo de configuración del plugin de Latch.*
* *latch.instances*: archivo con las instancias de Latch.
* *latch.accounts*: archivo con los accountID de Latch.
* *mosquitto.acl*: archivo con las ACLs .
* *mosquitto.users*: archivo con los usuarios.
* *tools/install.py*: script de instalación del plugin.
* *tools/instances\_op.py*: script para la gestión estática de instancias.
* *tools/pair\_op.py*: script para la gestión estática del "pairing".
* *tools/users\_op.py*: script para la gestión de usuarios.

## 4.2 Pre-instalación

Antes de la instalación del Plugin, lo primero a realizar será la creación de la aplicación dentro del portal de Latch (<https://latch.elevenpaths.com>). La aplicación puede ser creada con cualquier nombre, y debe tener "*2-Factor OTP*" y "*Lock latches after request*" deshabilitados. Una vez creada la aplicación, hay que tomar nota de los valores *Application ID* y *Secret*, que serán utilizados posteriormente en la configuración del plugin.





Con el proceso anterior finalizado, hay que verificar que el sistema que contiene nuestro broker MQTT tiene el software necesario para el plugin de Latch. Para ello, las siguientes librerías serán necesarias:

**sudo apt-get install python-dev mosquitto-dev libmosquitto-dev**

Además será necesario instalar una librería especial de Python: paho-mqtt. Puede ser instalada con:

**pip install paho-mqtt**

De la misma forma, el plugin de Latch necesita un plugin de Mosquitto para el desarrollo de plugins de seguridad en Python. Puede ser instalado desde <https://github.com/oribit/mosquitto_pyauth> siguiendo las instrucciones allí detalladas.

## 4.3 Instalación

Con los prerrequisitos satisfechos, el Plugin de Latch puede descargarse de <https://github.com/oribit/latch-plugin> Por favor, descargar este repositorio en el directorio que se quiere usar para el plugin.

Un vez descargado el plugin ejecutaremos desde el directorio dónde se haya descargado, *python ./tools/install.py* que procederá a la verificación e instalación del plugin en el sistema. Este script nos solicitará los datos de *Application ID* y *Secret* anteriormente mencionados y los guardará dentro del fichero *latch.conf*.

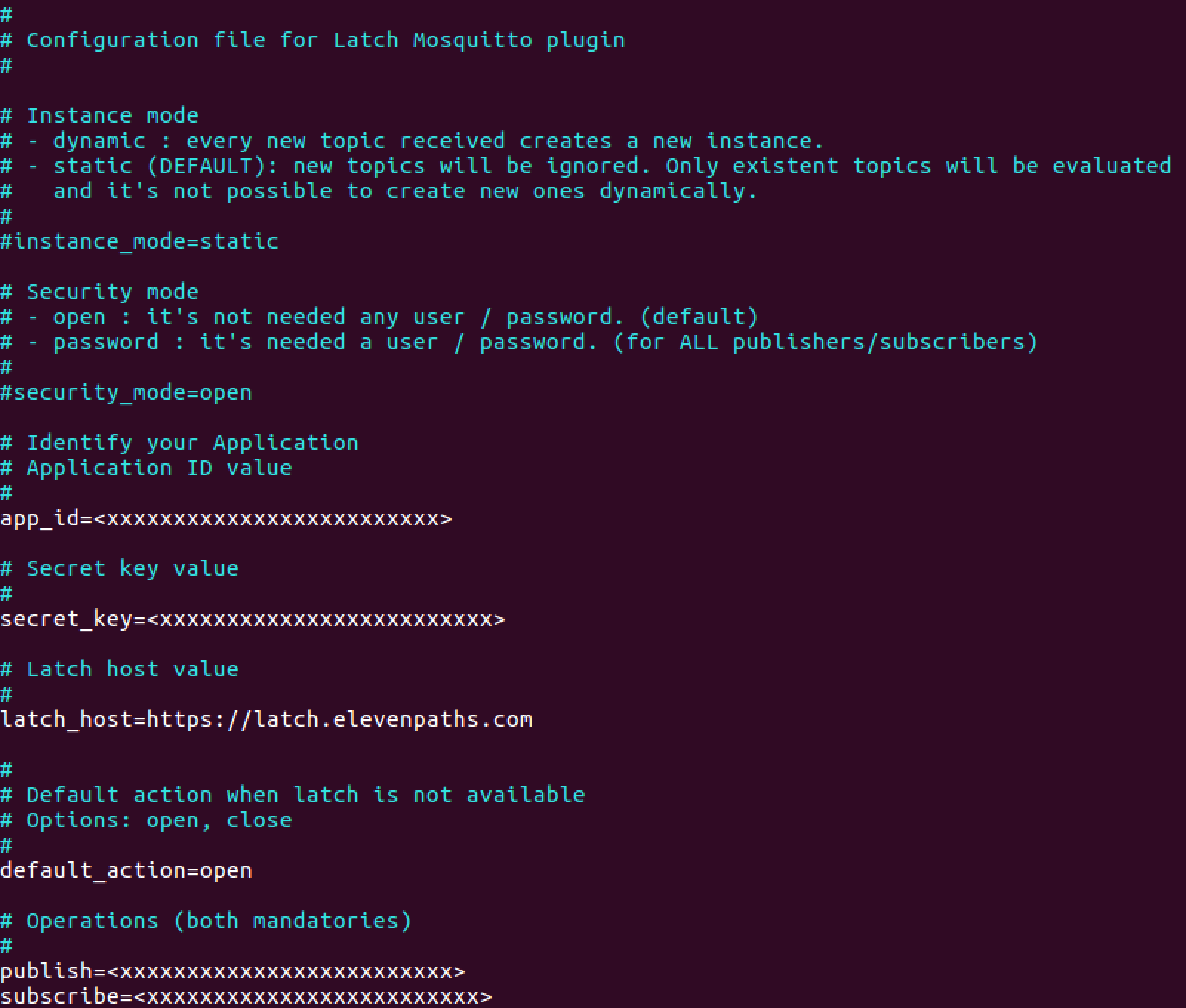
El script de instalación realizará una verificación de los pre-requisitos, y a continuación procederá a la copia de los archivos del Plugin en el directorio que se haya especificado.

Los ficheros de configuración y scripts de utilidades (*tools*), se copiarán en el directorio: */etc/mosquitto/plugin/latch.* El proceso de instalación posiblemente requerirá **permisos root al ejecutarse** **(sudo),** debido a que el fichero de configuración de Mosquitto suele pertenecer a root.

Una vez finalizada la instalación, será necesario crear un usuario con el que poder empezar a usar **Latch**. Esto puedo realizarse mediante el script: *python ./tools/users\_op.py* Con el usuario creado, se podrá realizar el "**emparejamiento**" de la aplicación tal como está explicado en la sección 3.1.

## 4.4 Configuración

Finalizada la instalación, el fichero de configuración del Plugin estará situado en */etc/mosquitto/plugin/latch/latch.conf:*



Los campos *app\_id* y *secret\_key* contendrán los datos facilitados en el script de instalación. Los campos *publish* y *subscribe* contendrán los IDs creados automáticamente durante el proceso de instalación.

## 4.5 Ejecución

La ejecución del Plugin se realiza de forma automática con la ejecución del broker MQTT. El script de instalación añadirá las siguientes líneas al archivo de configuración de Mosquitto:

***auth\_plugin /usr/lib/mosquitto/auth\_plugin\_pyauth.so***

***auth\_opt\_pyauth\_module mosquitto\_latch***

***auth\_opt\_latch\_conf latch.conf***

Todos los mensajes generados por el plugin, son redireccionados a la salida estándar que se haya configurado en el broker, por lo que si se ha configurado una salida a un syslog o a un archivo, los mensajes del Plugin serán escritos en el mismo lugar.

Además del uso de la redirección estándar, el Plugin publica sobre el *topic* ***LATCH/status*** cualquier bloqueo que se realice por estar Latch habilitado, tanto a nivel global, como por operación (*publish/subscribe*) o por *topic*. A este *topic* se puede subscribir cualquier agente.