



Tecnológico  
de Monterrey

# Manejo de Datos con MQTT

## Equipo 31

Ricardo Díez Gutiérrez González  
Rafael Fernando Olmedo Aguilar  
Brian Daniel López Alvarado

A01797151  
A01796862  
A01244885



# Agenda

- ¿Que es MQTT?
- Componentes
- ¿Cómo funciona?
- Características clave
- Configuraciones
  - Broker
  - IOS
  - Andriod
  - Python
- Demo
- Conclusiones

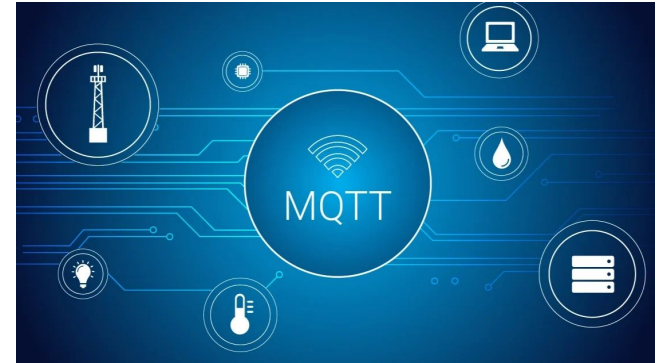




# Protocolo MQTT

## MQTT (Message Queuing Telemetry Transport)

- Protocolo de mensajería **ligero y eficiente**.
- Diseñado para redes con **ancho de banda limitado o alta latencia**.
- Ideal para **IoT, sensores, y dispositivos móviles**.  
Basado en el modelo **publicador / suscriptor** (*publish/subscribe*).

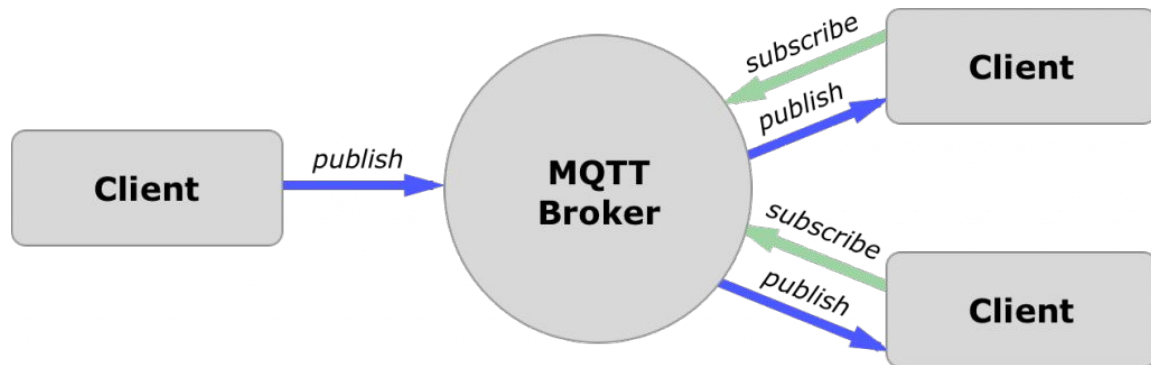




# Protocolo MQTT

## Componentes principales:

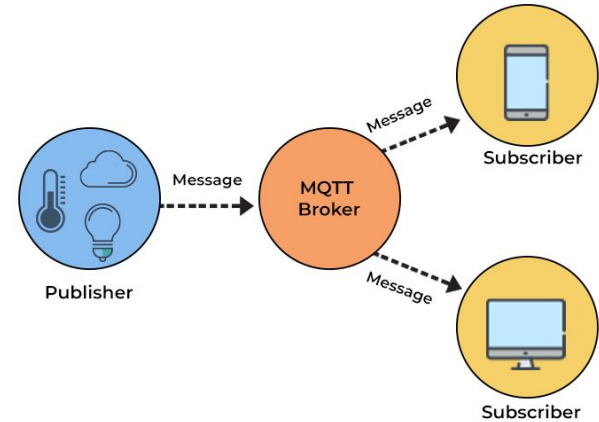
- **Broker:** Centro del sistema. Recibe y redirige mensajes.
- **Publisher:** Dispositivo que envía mensajes.
- **Subscriber:** Dispositivo que recibe mensajes de temas específicos (*topics*).





# Protocolo MQTT

## MQTT PROCESS



## ¿Cómo funciona?

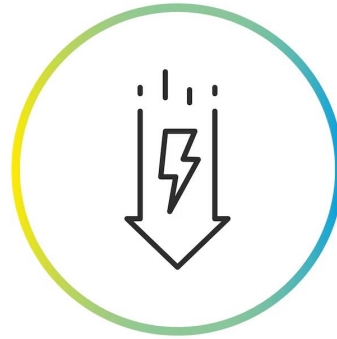
1. Publisher envía un mensaje a un **topic** (ej: "sensores/temperatura").
2. El **broker** recibe el mensaje y lo distribuye a todos los **subscribers** suscritos a ese topic.
3. Subscribers reciben el mensaje en tiempo real.



# Protocolo MQTT

## Características clave

- Bajo consumo de energía
- Ideal para redes inestables
- Soporta **QoS (Calidad de Servicio)**
- Modo persistente y "last will message"
- Usa **TCP/IP** y puede ir sobre **TLS para seguridad**





# Demostración Práctica de MQTT

Se hará la demostración utilizando tres tipos de publishers

- IOS
- Andriod
- Python

El broker que se utilizara será **HiveMQ**





# Configuración del Broker

## Creación de cuenta en HiveMQ Cloud

- Se accedió a HiveMQ Cloud utilizando una cuenta de Google del **Tecnológico de Monterrey**.

## Acceso al panel de administración

- En la página de inicio, se seleccionó **"Manage Cluster"** para ingresar a la consola de administración del broker MQTT.

## Obtención de datos del broker

- En la sección **"Overview"**, se copiaron los datos necesarios para la conexión:
  - **Hostname**
  - **Port (TLS o WebSocket)**
  - **Datos de autenticación**

## Creación de usuarios personalizados

- Se creó el usuario:
  - **Usuario:** equipo31
  - **Contraseña:** Equipo31
- Esta credencial se usó para autenticar las conexiones de los publishers/subscribers.





# Configuración del Broker

**Overview**   Access Management   Integrations   Web Client   Getting Started

## Connection Details

Comprehensive details and statistics for your cluster

### URL

fdfb48b793094632881abf5232112c4a.s1.eu.hivemq.cloud



### Port

8883



### Websocket Port

8884



### TLS MQTT URL

fdfb48b793094632881abf5232112c4a.s1.eu.hivemq.cloud:8883



### TLS Websocket URL

fdfb48b793094632881abf5232112c4a.s1.eu.hivemq.cloud:8884/mqtt





# Configuración del Publisher de IOS

Uso de MQTTAnalyzer en iOS con HiveMQ Cloud

## Aplicación seleccionada: MQTTAnalyzer (iOS)

- Utilizaremos la app gratuita **MQTTAnalyzer** para visualizar mensajes publicados a través del protocolo MQTT.
- Permite conectar con brokers como **HiveMQ Cloud** y suscribirse a distintos *topics*.

## Agregar nueva conexión

- Al abrir la app, toca el ícono + para crear un nuevo perfil de conexión MQTT.

## Configurar el broker HiveMQ Cloud

- **Alias:** `Broker_equipo31` (o el nombre que desees)
- **Host:** (copiado de Overview de HiveMQ)  
Ejemplo: `xxxxxxxx.s1.eu.hivemq.cloud`
- **Port:** `8883` (para conexión segura TLS)
- **Client ID:** `cliente_equipo31` (puede ser cualquier identificador único)
- **Versión:** `MQTT V3.1.1`

## Habilitar TLS

- Activa la opción `Use SSL/TLS` para cifrar la conexión.

## Autenticación con credenciales




- **Username:** `equipo31`
- **Password:** `Equipo31`

## Guardar y conectar

- Toca `Save`, luego `Connect`.



5:45



55

Cancel

Edit broker

Save

SERVER

Alias

Broker\_equipo31

Hostname

fdfb48b793094632881abf5232...

Port

8883

Protocol

MQTT

Websocket

Version

3.1.1

5.0

TLS

☒

Allow untrusted

☐

AUTHENTICATION

Username/password


☒

Username

equipo31

Password

Leave username and/or password empty. In order to not persist them. You will get a login dialog.



Certificate

☐

SUBSCRIBE TO

equipo31

>

A screenshot of a Telegram chat interface. At the top, the status bar shows the time 5:47, signal strength, Wi-Fi, and battery at 54%. The chat header includes a back arrow, the name 'Brokers' in blue, the group name 'equipo31', a paper plane icon, a circle icon, and a pause icon. Below the header, the chat title is 'Topics/Messages' with '0/0' and a menu icon, and the topic is 'equipo31'. The main chat area is black with a white star icon and the text 'Waiting for messages'.



# Configuración del Publisher de Android

Uso de MyMQTT en Android con HiveMQ Cloud

## Aplicación seleccionada: MyMQTT (Android)

- Utilizaremos la app gratuita **MyMQTT**.
- Permite conectar con brokers como **HiveMQ Cloud** y suscribirse a distintos *topics*.

## Agregar nueva conexión

- Al abrir la app, toca el ícono **de los 3 puntos** para crear un nuevo perfil de conexión MQTT.

## Configurar el broker HiveMQ Cloud

- **Host:** *(copiado de Overview de HiveMQ)*  
Ejemplo: [xxxxxxxx.s1.eu.hivemq.cloud](https://xxxxxxxx.s1.eu.hivemq.cloud)
- **Port:** **8883** (para conexión segura TLS)
- **Versión:** **MQTT V3**

## Habilitar TLS

- Activa la opción **Use SSL/TLS** para cifrar la conexión.

## Autenticación con credenciales

- **Username:** **equipo31**
- **Password:** **Equipo31**

## Guardar y conectar

- Toca **Connect**.



Android 

8:35 98%

## MQTT Broker

Host  
778478cf17dc4abea630ebe7ac4bb5f1.s1.

Port  
8883 ☒ SSL

☒ MQTT V3 ☐ MQTT V5

### Credentials

Username (optional)  
equipo31

Password (optional)  
.....

Connect

8:40 97%

778478cf17dc4abea630e...  
Connected

PARAMOUNT+ WITH SHOWTIME PLAN ONLY

Dashboard Subscribe Publish

8:41 97%

## Subscribe

Topic

Subscribe

Prueba1  
Enabled

Bit Pay Rent, Unlock Rewards OPEN >

Dashboard Subscribe Publish

8:41 97%

## Publish

Topic

Message

Advanced Save Publish

Bit Pay Rent, Unlock Rewards OPEN >

Dashboard Subscribe Publish



# Python Script

- **Paso 1: Importar la librería paho-mqtt:** <https://pypi.org/project/paho-mqtt/>

```
# This library allows Python applications to connect to an MQTT broker
import paho.mqtt.client as paho
from paho import mqtt
```

- **Paso 2: Definir las funciones callbacks para ciertos eventos**

```
# setting callbacks for different events to see if it works, print the message etc.
# callback for when the client receives a CONNACK response from the server
def on_connect(client, userdata, flags, rc, properties=None):
    print("CONNACK received with code {}".format(rc)) # rc: reason code

# with this callback you can see if your publish was successful
def on_publish(client, userdata, mid, properties=None):
    print("mid: " + str(mid)) # mid: message ID

# print which topic was subscribed to
def on_subscribe(client, userdata, mid, granted_qos, properties=None):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

# callback for when a PUBLISH message is received from the server
def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
```



# Python Script

- **Paso 3: Crear una nueva instancia de cliente MQTT utilizando la biblioteca Paho:**

```
# using MQTT version 5 here, for 3.1.1: MQTTv311, 3.1: MQTTv31
# userdata is user defined data of any type, updated by user_data_set()
# client_id is the given name of the client
client = paho.Client(client_id="", userdata=None, protocol=paho.MQTTv5)
client.on_connect = on_connect
```

- **Paso 4: Establecer una conexión segura y autenticar con credenciales**

```
# enable TLS for secure connection
client.tls_set(tls_version=mqtt.client.ssl.PROTOCOL_TLS)
# set username and password
client.username_pw_set(hivemq_username, hivemq_password)
# connect to HiveMQ Cloud on port 8883 (default for MQTT)
client.connect(hivemq_cluster_url, 8883)
```

- **Paso 5: Asignar las funciones callbacks**

```
# setting callbacks, use separate functions like above for better visibility
client.on_subscribe = on_subscribe
client.on_message = on_message
client.on_publish = on_publish
```



# Python Script

- **Paso 6: Suscribirse a un topic (o a todos #). QoS in [0,1,2]**

```
# subscribe to all topics of encyclopedia by using the wildcard "#"
client.subscribe("#", qos=1) # qos: quality of service
```

- **Paso 7: Publicar un mensaje a un topic determinado**

```
# a single publish, this can also be done in loops, etc.
client.publish("topic1", payload="This is a meesage from a Python script", qos=1)
```

- **Paso 8: Establecer un loop para monitorear los mensajes**

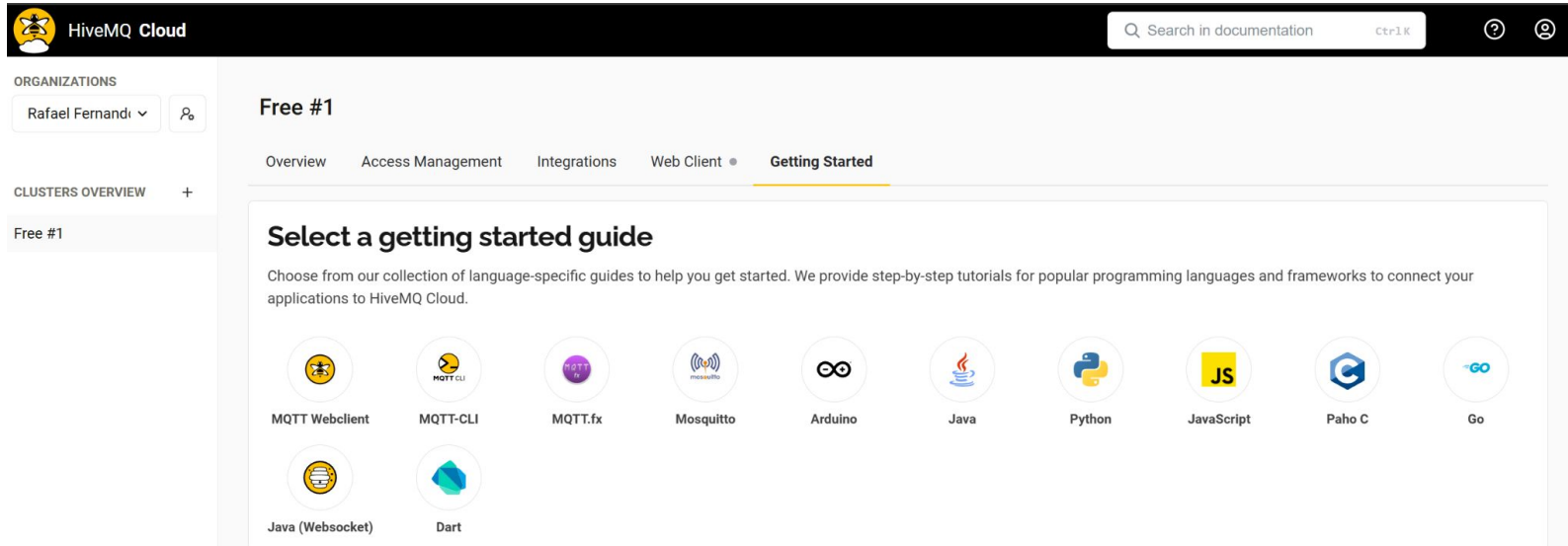
```
# loop_forever for simplicity, here you need to stop the loop manually
# you can also use loop_start and loop_stop
client.loop_forever() # continuously monitors network traffic for incoming and outgoing messages.
```

**GitHub:** <https://github.com/rfolmedoa/python-paho-hivemq-cloud.git>



# Otros lenguajes de programación

HiveMQ Cloud también proporciona la documentación necesaria para conectarse a un cluster en HiveMQ Cloud utilizando otros lenguajes de programación populares (e.g., Java, Go, JavaScript, etc.)



The screenshot displays the HiveMQ Cloud documentation interface. At the top, a black header bar contains the HiveMQ logo, the text 'HiveMQ Cloud', a search bar with the placeholder 'Search in documentation', and user icons. On the left, a sidebar shows 'ORGANIZATIONS' with 'Rafael Fernandez' and 'CLUSTERS OVERVIEW' with a '+' icon. The main content area is titled 'Free #1' and features a navigation bar with 'Overview', 'Access Management', 'Integrations', 'Web Client', and 'Getting Started' (which is highlighted). Below the navigation bar, the section 'Select a getting started guide' is displayed, followed by a descriptive paragraph: 'Choose from our collection of language-specific guides to help you get started. We provide step-by-step tutorials for popular programming languages and frameworks to connect your applications to HiveMQ Cloud.' A grid of ten circular icons represents different guides: MQTT Webclient, MQTT-CLI, MQTT.fx, Mosquitto, Arduino, Java, Python, JavaScript, Paho C, and Go. A second row shows 'Java (Websocket)' and 'Dart'.

HiveMQ Cloud

Search in documentation

ORGANIZATIONS

Rafael Fernandez

CLUSTERS OVERVIEW

Free #1

Free #1

Overview Access Management Integrations Web Client **Getting Started**

### Select a getting started guide

Choose from our collection of language-specific guides to help you get started. We provide step-by-step tutorials for popular programming languages and frameworks to connect your applications to HiveMQ Cloud.

- MQTT Webclient
- MQTT-CLI
- MQTT.fx
- Mosquitto
- Arduino
- Java
- Python
- JavaScript
- Paho C
- Go
- Java (Websocket)
- Dart



# DEMO

Se realizarán pruebas de conexión al broker HiveMQ Cloud en el siguiente orden:

- IOS
- Andriod
- Python



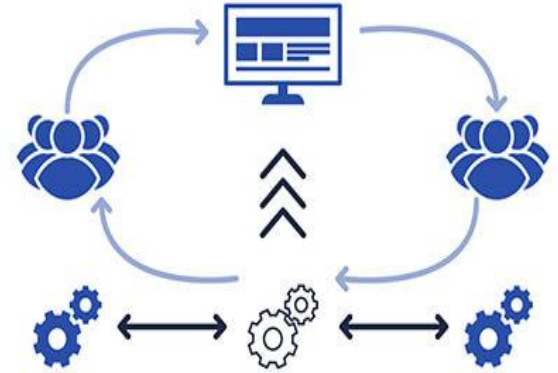
# Conclusiones

## **Interoperabilidad y flexibilidad en tiempo real:**

La demostración práctica evidenció cómo MQTT permite una comunicación eficiente y en tiempo real entre dispositivos heterogéneos, como un iPhone, un dispositivo Android y un cliente Python. Esta interoperabilidad confirma su utilidad en entornos donde coexisten múltiples plataformas y lenguajes de programación, como en aplicaciones IoT o sistemas distribuidos.

## **Ligereza y eficiencia del protocolo:**

Una de las principales ventajas observadas fue la eficiencia del protocolo MQTT en cuanto al uso de ancho de banda y consumo de recursos. Su arquitectura basada en eventos y su modelo de publicación/suscripción minimizan la sobrecarga de datos, lo cual es esencial para dispositivos móviles con limitaciones de batería o conectividad intermitente



# Conclusiones

## **Simplicidad en la implementación y escalabilidad:**

El uso de MQTT resultó ser sorprendentemente sencillo tanto en dispositivos móviles como en entornos de programación como Python. Este bajo nivel de complejidad facilita una curva de aprendizaje accesible y permite escalar soluciones fácilmente, añadiendo más dispositivos o nodos sin necesidad de rediseñar la arquitectura base.

## **Fiabilidad en la comunicación:**

Durante la demostración se pudo comprobar que, a pesar de posibles cambios de red o desconexiones momentáneas, MQTT mantiene un alto nivel de fiabilidad gracias a sus distintos niveles de QoS (Calidad de Servicio), permitiendo adaptar el protocolo a diferentes necesidades de entrega de mensajes.



# Conclusiones

## **Comprensión profunda del modelo de publicación/suscripción:**

Explicar y observar el comportamiento de MQTT en acción permitió reforzar el entendimiento de su modelo centrado en el broker, donde los dispositivos actúan como clientes publicadores o suscriptores. Este paradigma desacoplado mejora la modularidad del sistema y permite una mayor independencia entre componentes.

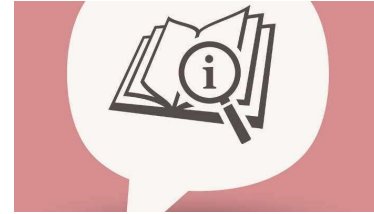
## **Aplicaciones prácticas y proyección futura:**

Esta actividad no solo brindó conocimientos técnicos, sino también una visión clara de las múltiples aplicaciones de MQTT en el mundo real, como hogares inteligentes, monitoreo remoto, domótica, y más. Además, resaltó cómo este protocolo se perfila como una herramienta clave para el desarrollo de soluciones IoT robustas, escalables y orientadas al futuro.





# Referencias



Zhang, Y., & Wang, L. (2023). *Secure Data Distribution Architecture in IoT Using MQTT*. *MDPI Electronics*, 13(4), 2515

EMQX (2024). *MQTT in Python with Paho Client: Beginner's Guide 2024*.

González, J. R., Martínez, J. C., & García, J. M. C. (2021). *Security Analysis of the MQTT-SN Protocol for the Internet of Things*. *MDPI Electronics*, 12(21), 10991

Ahmed, M., & Akhtar, M. M. (2021). *Smart Home: Application using HTTP and MQTT as Communication Protocols*.

Lima, K., Oyetoyan, T. D., Heldal, R., & Hasselbring, W. (2025). *Evaluation of MQTT Bridge Architectures in a Cross-Organizational Context*.