

## Simulación de una Red Local utilizando MQTT

### Objetivos:

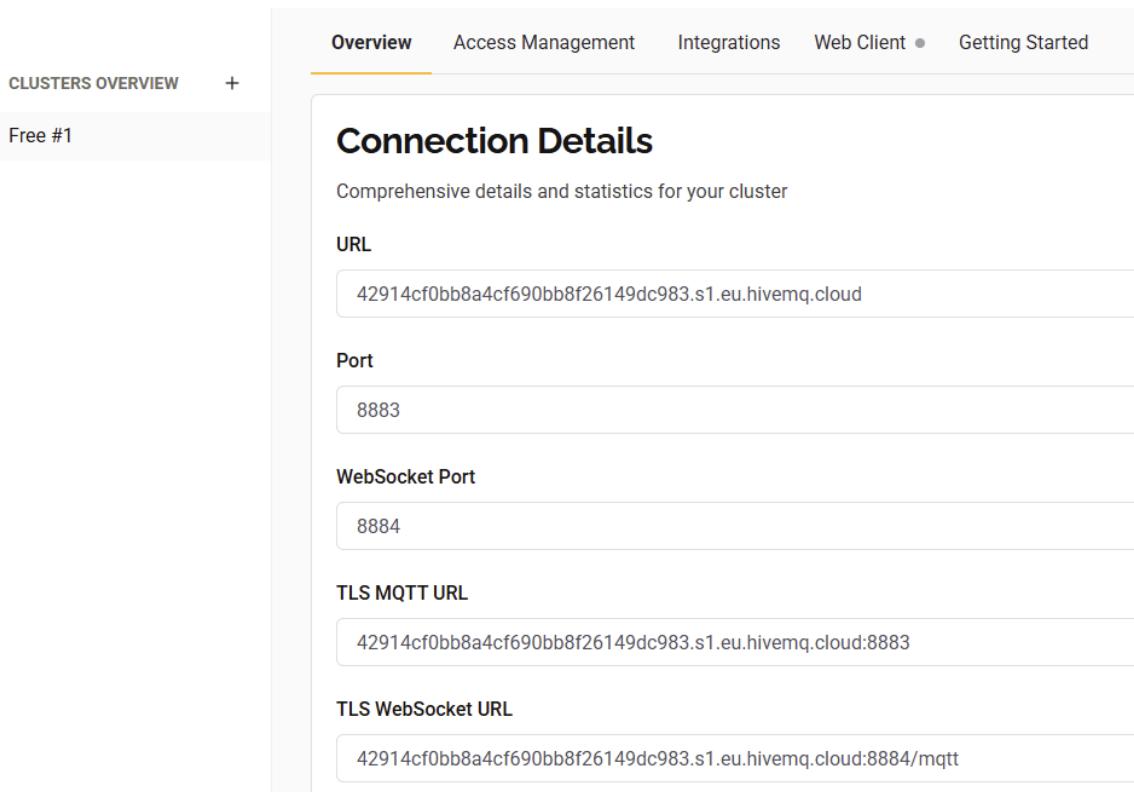
- Comprender el modelo de comunicación Publish/Subscribe (pub/sub) y cómo se diferencia del modelo cliente-servidor tradicional.
- Configurar y ejecutar un broker MQTT, cloud-based o local
- Implementar comunicación entre múltiples clientes MQTT simulando dispositivos IoT en una red local.
- Experimentar con tópicos jerárquicos y comodines (+, #) para realizar broadcasts y routing lógico.

### Requerimientos:

- Broker MQTT ([HiveMQ](#), [Mosquitto](#), [EMQX](#))
- Cliente MQTT (Pueden usar Python, JAVA, Node.js, lo que gusten)
- Hardware: PCs o máquinas virtuales locales

### Consignas:

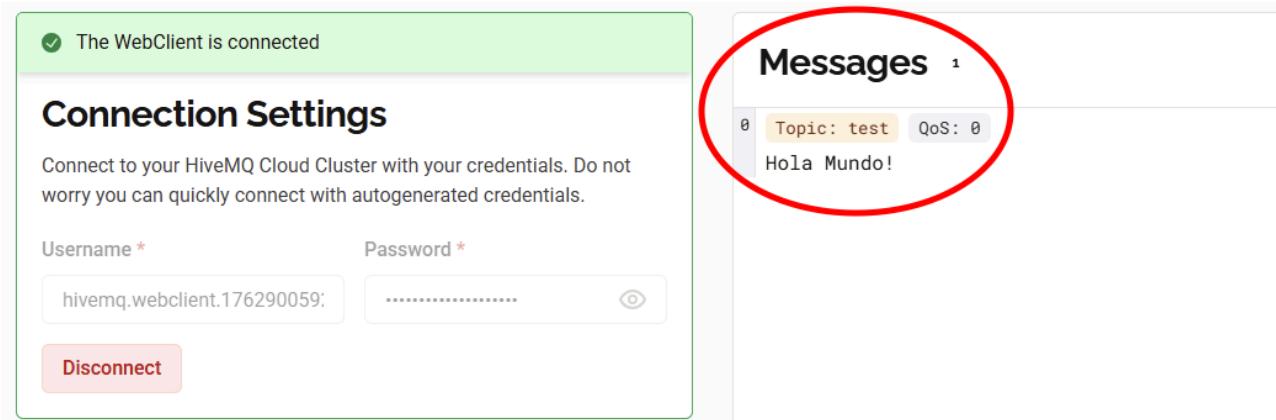
- 1) Resumir brevemente las características del protocolo MQTT. Incluir ventajas, desventajas y principales usos. Responder: ¿Qué es el patrón de diseño PubSub?.
- 2) Instalar/desplegar y ejecutar un broker MQTT. Por ejemplo, en HiveMQ, tendremos un dashboard con parámetros de red para conectarnos (no te preocupes que es gratis):



The screenshot shows the HiveMQ Cloud interface. At the top, there's a navigation bar with tabs: Overview (which is selected), Access Management, Integrations, Web Client, and Getting Started. Below the navigation bar, there's a sidebar on the left with a 'CLUSTERS OVERVIEW' section containing a button '+'. Underneath it, a cluster named 'Free #1' is listed. The main content area is titled 'Connection Details' and contains the following fields:

- URL:** 42914cf0bb8a4cf690bb8f26149dc983.s1.eu.hivemq.cloud
- Port:** 8883
- WebSocket Port:** 8884
- TLS MQTT URL:** 42914cf0bb8a4cf690bb8f26149dc983.s1.eu.hivemq.cloud:8883
- TLS WebSocket URL:** 42914cf0bb8a4cf690bb8f26149dc983.s1.eu.hivemq.cloud:8884/mqtt

- 3) Verificar que el broker funciona, suscribiendote con un cliente (pueden encontrar tutoriales para utilizar [Java](#), [python](#) u otros lenguajes para este propósito)



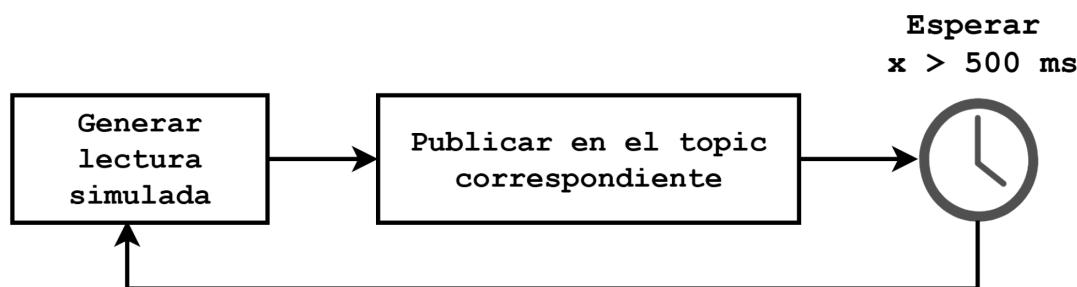
- 4) Una vez que tenemos nuestra arquitectura funcionando:

- Simular una comunicación directa entre dos nodos de una red local. Para ello crear dos clientes: **Dispositivo A**, que publica en lan/deviceA/status, **Dispositivo B** se suscribe a ese tópico y muestra los mensajes recibidos. Capturar y documentar resultados.
- Crear un tópico general lan/broadcast/#. Configurar al menos dos clientes para suscribirse a lan/broadcast/#. Desde un cliente “central”, publicar mensajes en lan/broadcast/all. Capturar y documentar resultados. Con esto simularemos **broadcasting** en esta pequeña LAN.

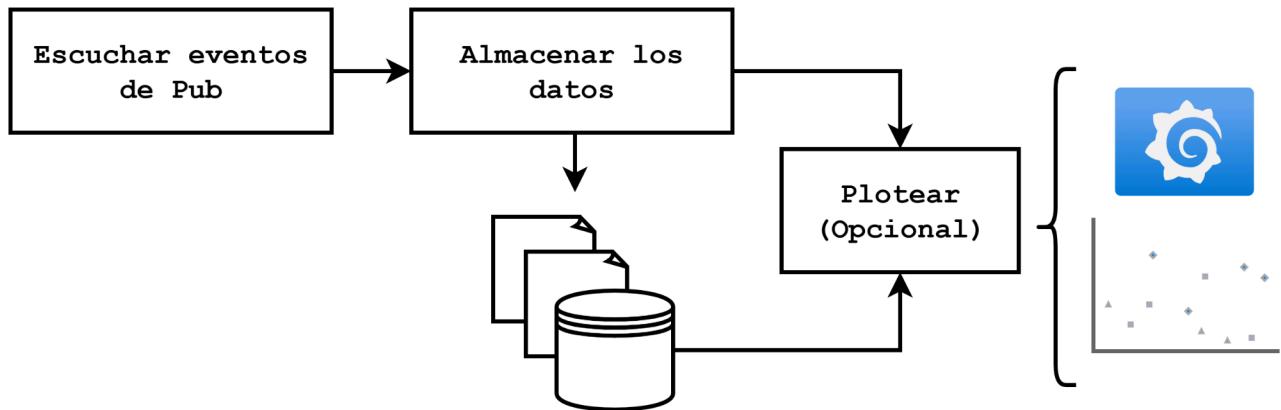
- 5) Implementar una jerarquía de tópicos como:

lan/sala1/sensor/temp  
lan/sala1/sensor/hum  
lan/sala2/sensor/temp

- Simular, en cada cliente, un sensor que genere datos (utiliza un generador de número aleatorios). Recolectar estos en una gateway/servidor suscrito a estos tópicos.



- b) Ahora nuestro cliente “central” (ahora gateway) se suscribirá y recopilará los datos generados por los sensores en archivos locales (texto, CSV, serializado, o si te animás, en una base de datos).



- c) **Opcional:** si te animás, investigá como plotear los datos (usando, por ejemplo, [Grafana](#)).
- d) Mediante **broadcasting** deberemos poder enviar al menos dos mensajes de comando a los clientes (ahora sensores): **comenzar la simulación de datos y apagarse**.
- e) Capturar un paquete usando un sniffer y realizar un análisis simple de la composición del mismo.

5) Responder:

- ¿Sobre qué protocolos de capa de transporte están trabajando en esta actividad?
- ¿Qué pueden decir sobre la garantía de **Integridad, Confidencialidad y Disponibilidad** en esta arquitectura?
- ¿Qué rol juegan los niveles de QoS en la fiabilidad de los mensajes?
- ¿Qué ventajas ofrece el modelo pub/sub frente al modelo cliente-servidor?
- ¿Qué limitaciones tiene MQTT respecto a una red LAN real?
- ¿Qué implicaciones tiene depender de un broker central para la comunicación?

**Podés usar cualquier herramienta que necesites para resolver este laboratorio.**