# REDES INALÁMBRICAS DE SENSORES

## MÁSTER EN INGENIERÍA ELECTRÓNICA, ROBÓTICA Y AUTOMÁTICA

## PRÁCTICA 4. CONTIKI: MONITORIZACIÓN DE LA INFORMACIÓN


Eduardo Hidalgo Fort

ehidalgo@us.es

José María Hinojo Montero

jhinojo@us.es

Dpto. Ingeniería Electrónica


Curso 23-24

# ÍNDICE

# I. Docker Compose

## Introducción

- Docker Compose es una herramienta que permite definir y ejecutar múltiples contedores de Docker
- Utiliza ficheros con sintaxis YAML
  - docker-compose.yml
- Características
  - Un único comando permite desplegar varios contenedores de forma simultánea
  - Permite establecer y resolver dependencias entre contenedores
  - Es compatible con múltiples entornos
    - Producción
    - Puesta en marcha
    - Test
    - Integración continua


https://docs.docker.com/compose

```
version: "3"

services:
  image: prom/prometheus
    network_mode: host
    volumes:
      - ./prometheus/config:/etc/prometheus/
      - prometheus_data:/prometheus/data
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
      - "--storage.tsdb.path=/prometheus"
    restart: unless-stopped
```

# I. Docker Compose

## Estructura Compose

- Se corresponde con un fichero YAML
  - Formato de archivo orientado a la serialización de datos
  - Usado para definir archivos de configuración
  - Basado en indentación y definición de etiquetas
- **Secciones**
  - **Version:** establece la versión de Docker-compose que debe usarse para interpretar el archivo
  - **Services:** define los contenedores que serán creados al lanzar la aplicación. Esta sección es obligatoria.
  - **Network:** establece las conexiones a nivel de red entre los contenedores dentro de un mismo servicio
  - **Volumes:** fija un espacio para el almacenamiento persistente de los datos y para compartirlos entre contenedores.
  - **Configs:** se utiliza para establecer datos de configuración que son dependientes de la plataforma o que deben cargarse en tiempo de ejecución.
- Multitud de opciones

  https://docs.docker.com/compose/compose-file/

```
version: "3"
services:
  foo:
    image: foo
  bar:
    image: bar
    profiles:
      - test
  baz:
    image: baz
    depends_on:
      - bar
    profiles:
      - test
  zot:
    image: zot
    depends_on:
      - bar
    profiles:
      - debug
```

# I. Docker Compose

## Estructura Compose

```
version: "3"

services:
  mosquitto:
    image: eclipse-mosquitto
    network_mode: host
    volumes:
      - ./mosquitto/conf/mosquitto.conf:/mosquitto/conf/mosquitto.conf
      - ./mosquitto/conf/mosquitto.passwd:/mosquitto/conf/mosquitto.passwd
      - ./mosquitto/log:/mosquitto/log
      - ./mosquitto/data:/mosquitto/data
    ports:
      - 1883:1883
    restart: unless-stopped

  mqtt-exporter:
    build: ./mqtt_exporter
    network_mode: host
    devices:
      - /dev/ttyACM0
    depends_on:
      - mosquitto
      - prometheus
    environment:
      - MQTT_ADDRESS="localhost"
    ports:
      - 9000:9000
    restart: unless-stopped
```

```
  prometheus:
    image: prom/prometheus
    network_mode: host
    volumes:
      - ./prometheus/config:/etc/prometheus/
      - prometheus_data:/prometheus/data
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
      - "--storage.tsdb.path=/prometheus"
    restart: unless-stopped

  grafana:
    image: grafana/grafana
    network_mode: host
    ports:
      - 3000:3000
    volumes:
      - ./grafana/config/datasources:/etc/grafana/datasources
      - ./grafana/config/dashboards:/etc/grafana/dashboards
      - grafana_data:/var/lib/grafana
    restart: unless-stopped

volumes:
  prometheus_data:
  grafana_data:
```

# I. Docker Compose

## Instalación

- Verificar que "docker-compose" está instalado

```
vmu@vm-devel:~/ris/p4$ docker-compose -v
docker-compose version 1.22.0, build f46880fe
```

- Si no se reconoce el comando anterior, es necesario su instalación:

```
sudo curl -SL
https://github.com/docker/compose/releases/download/v2.12.2/d
ocker-compose-linux-x86_64 -o /usr/local/bin/docker-compose
sudo chmod 755 /usr/local/bin/docker-compose
```

- Ejecutar el comando "docker-compose" para verificar su correcta instalación
- En caso de fallo, revisar la variable PATH y verificar que el directorio "/usr/local/bin" aparece

```
vmu@vm-devel:~/ris/p4$ echo $PATH
/home/vmu/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin
:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

# I. Docker Compose

## Comandos relevantes

- Para lanzar el proyecto

```
vmu@vm-devel:~/ris/p4$ docker-compose up
```

- En caso de hacer un cambio en la aplicación "mqtt-exporter", es necesario reconstruir el proyecto

```
vmu@vm-devel:~/ris/p4$ docker-compose build
```

- Parar los servicios

```
vmu@vm-devel:~/ris/p4$ docker-compose stop
```

- Arrancar los servicios

```
vmu@vm-devel:~/ris/p4$ docker-compose start
```

- Parar los servicios y eliminarlos de memoria, incluyendo los volúmenes y redes

```
vmu@vm-devel:~/ris/p4$ docker-compose down
```

# I. Docker Compose

## Comandos relevantes

- Listar todos los contenedores que están ejecutándose o se han ejecutado

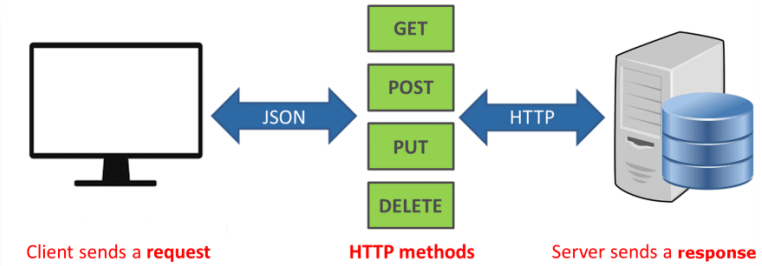
```
vmu@vm-devel:~/ris/p4$ docker-compose ps
```

- Lanzar un comando en un contenedor específico

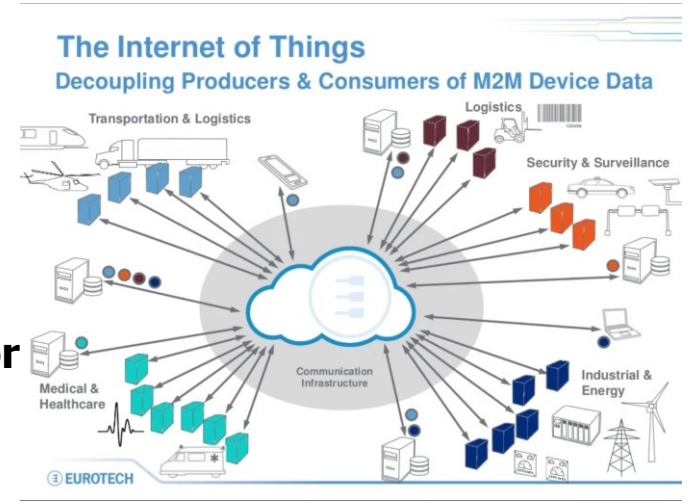
```
vmu@vm-devel:~/ris/p4$ docker exec  -it <nombre o id> <cmd>
```

# II. Intercambio de información

## Introducción



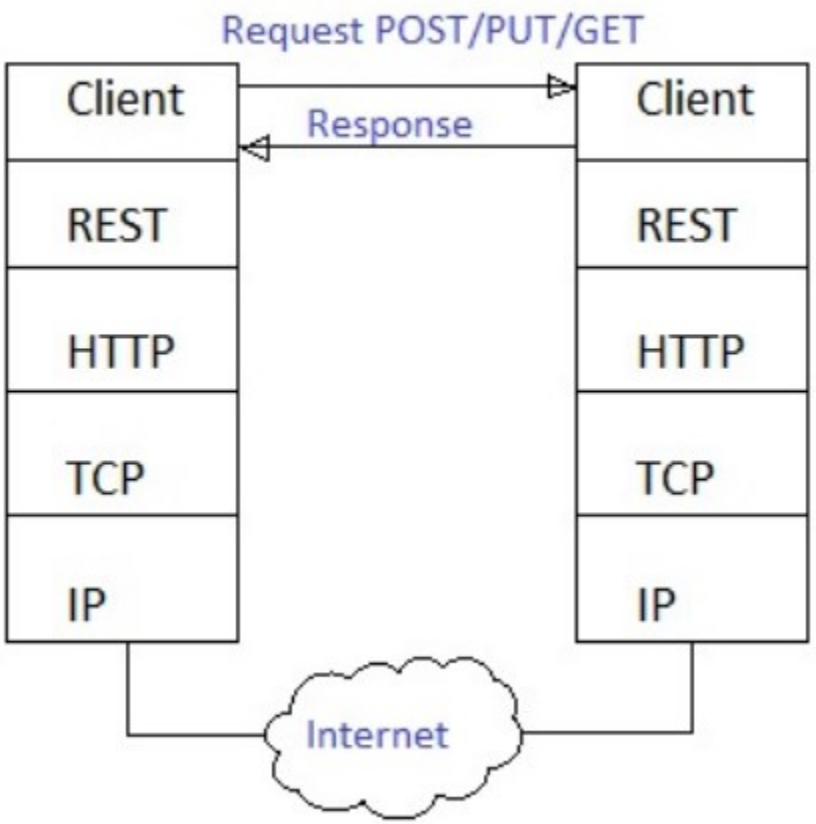

- Los sistemas de comunicaciones han evolucionado de intercambiar **flujos de datos** de forma fiable a **intercambiar mensajes**
  - Ejemplos
    - https://developer.twitter.com/en/docs
    - https://developer.amazon.com/en-US/docs/alexa/device-apis/message-guide.html
- Surgen dos paradigmas de comunicación
  - **Petición/respuesta (REST)**
    - En general se utiliza sobre HTTP
    - Se realiza una petición a una URL específica, se procesa en el servidor y devuelve la respuesta
  - **Productor/Consumidor o Publicador/Suscriptor**
    - Permite la distribucación de eventos a múltiples equipos
    - Los productores envían mensajes a un "broker"
    - Consumidores se suscriben a "topics"
    - Broker distribuye la información publicada a todos los suscriptores.



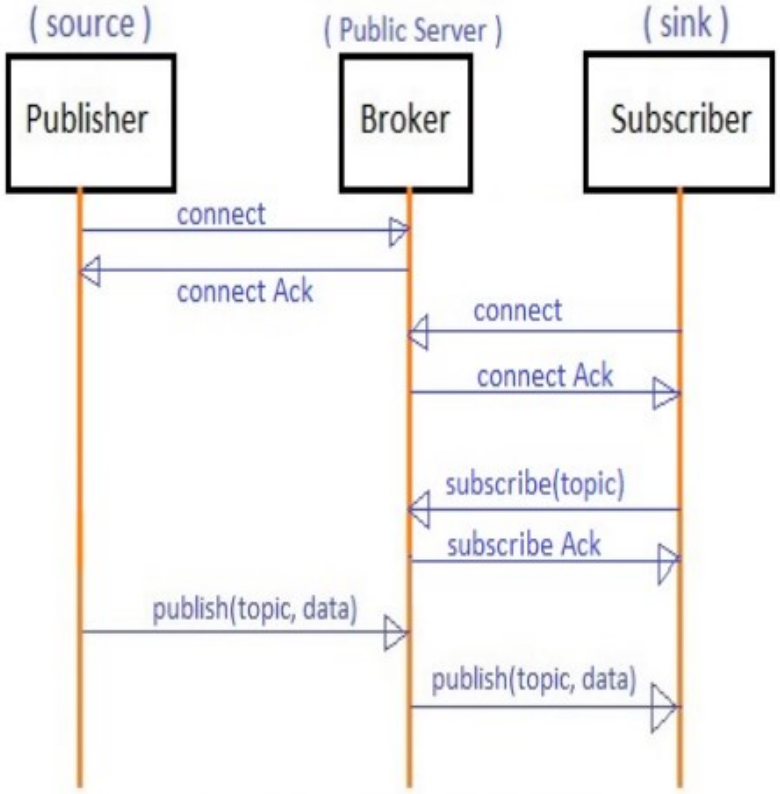

https://www.eurotech.com/attachment/download?id=1713

# II. Intercambio de información

## Paradigma REST

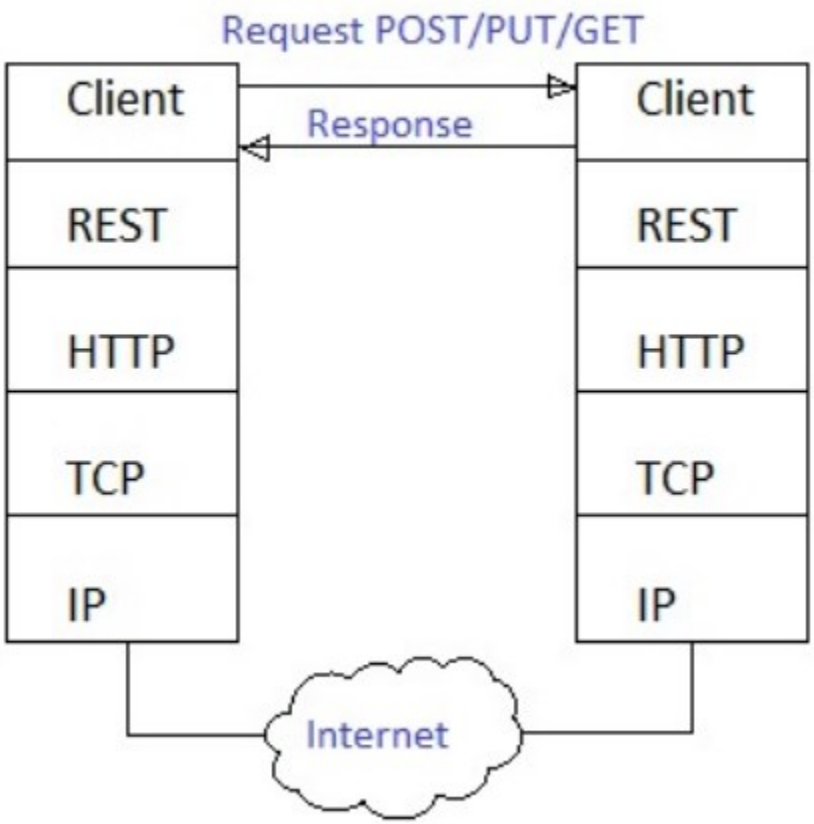



## Paradigma Pub/Subs

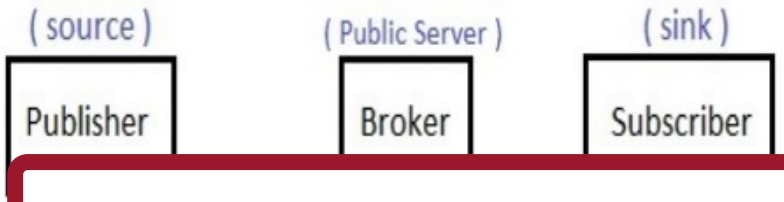


# II. Intercambio de información

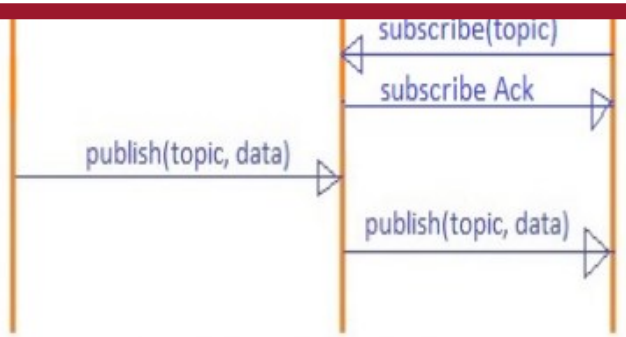
## Paradigma REST




## Paradigma Pub/Subs



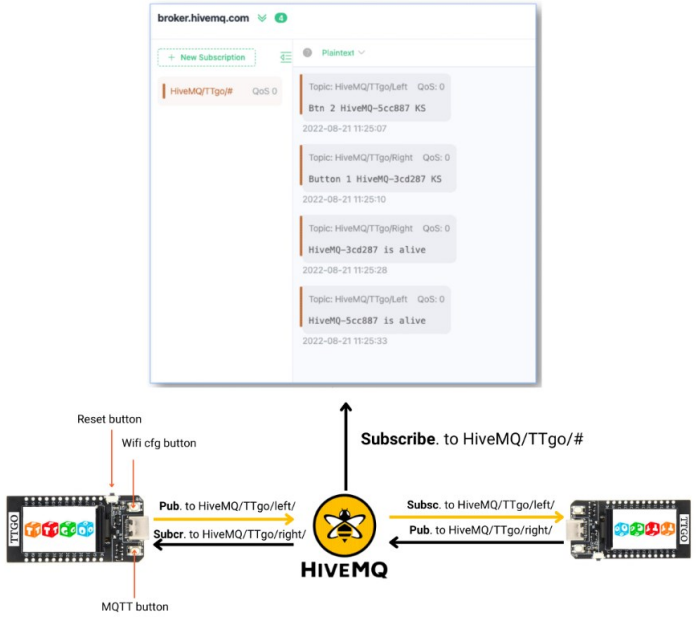

**Paradigma en el que se basa el Protocolo MQTT**



# II. Intercambio de información

## Message Queuing Telemetry Transport (MQTT)

- Protocolo basado en el **paradigma Publicador/Suscriptor**
  - Está diseñado para ejecutarse en **sistemas empotrados** y **plataformas móviles**
  - Suministra un **canal de comunicación de baja latencia**
    - Eficiente para distribuir mensajes a uno o varios suscriptores
    - Diseñado para minimizar la cantidad de datos enviados
    - Puede ser bidireccional
    - Está preparado para enviar grandes cantidades de mensajes de tamaño reducido.
      - El tamaño máximo de mensaje es de 256MB
  - Asegura el envío de mensajes sobre redes frágiles
  - Enfocado al bajo consumo
- **Estándar**



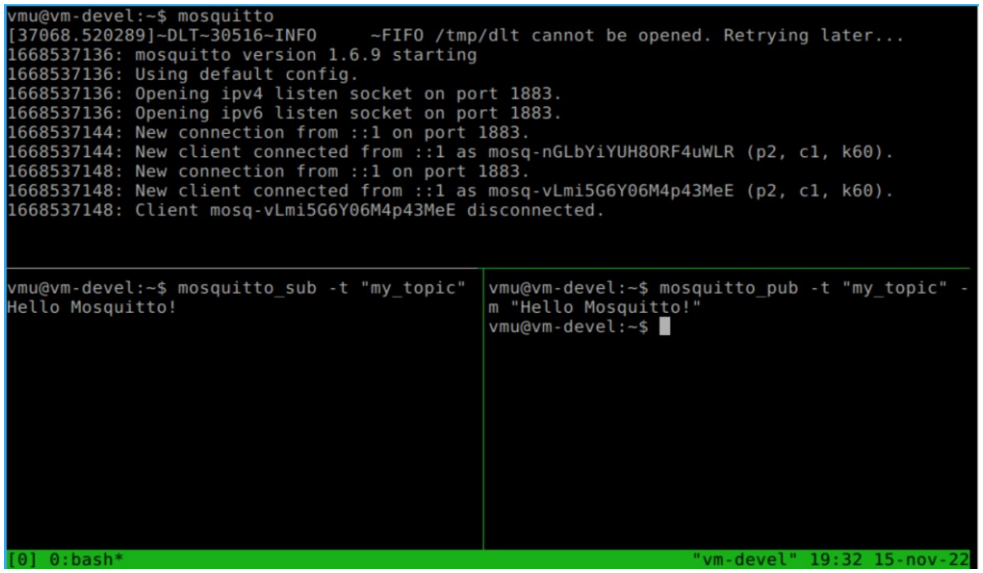

https://www.hivemq.com/blog/iot-tutorial-bidirectional-mqtt-communication-esp32/

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt

# II. Intercambio de información

## MOSQUITTO

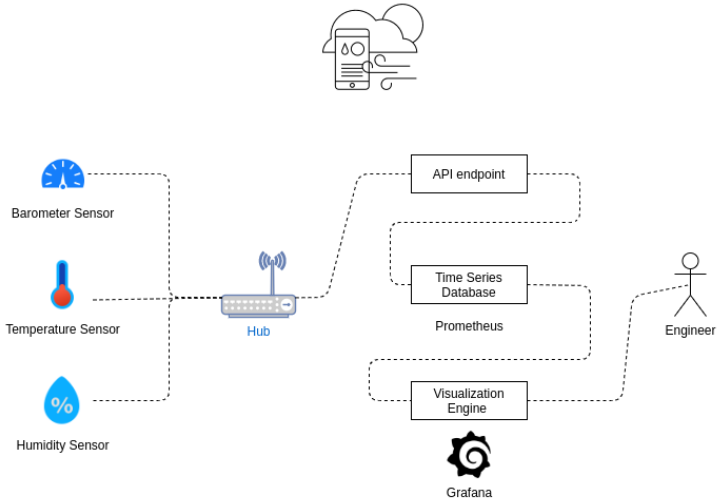
- Implementación Open Source desarrollada por Eclipse Foundation
  - Broker que implementa las versiones del protocolo MQTT 5.0, 3.1.1, y 3.1
  - Disponible para diferentes sistemas operativos
  - Suministra
    - **Broker**
      - mosquitto
    - **Publisher**
      - mosquitto_pub
    - **Subscriber**
      - mosquitto_sub
  - Se puede probar de forma nativa



# III. Sistema de Monitorización

## Prometheus

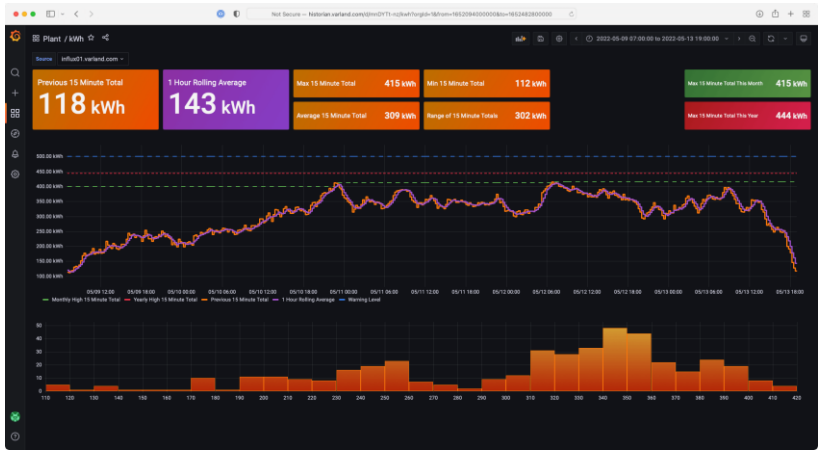
- Herramienta Open Source para monitorizar y generar alertas
  - Diseñado originalmente para monitorizar aplicaciones y servicios despleados en infraestructuras
  - La aplicación está optimizada para almacenar grandes volúmenes de datos temporales, pudiendo hacer consultas y escrituras muy rápidas
- Características
  - Filtrado por etiquetas
  - Se pueden usar funciones en las consultas
  - Permite agregar consultas
  - Métricas de rendimiento

# IV. Sistema de Monitorización

## Grafana


- Open Source Dashboard
  - Permite visualizar datos de series temporales
  - La visualización de los datos es personalizable
  - Permite mostrar la información por partes, alternando de forma automática entre ellas.
- Permite añadir plugins para mejorar su integración con otras bases de datos o agregadores
- Genera alertas
  - Permite definir umbrales
- Permite realizar transformaciones sobre los datos

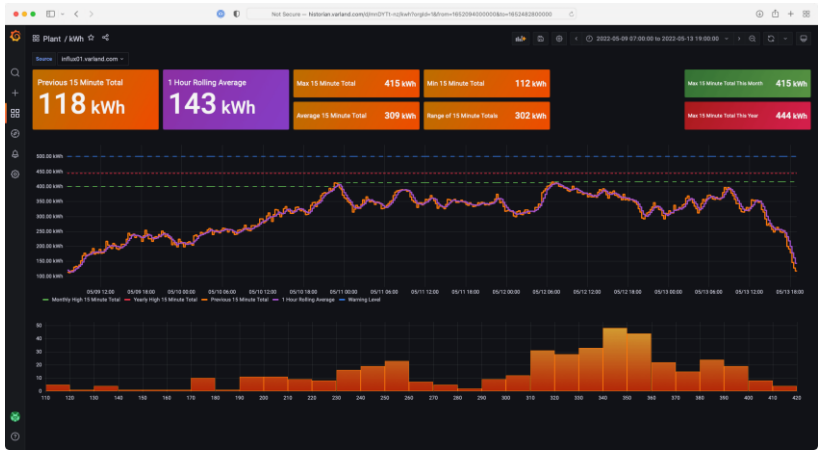



- Soporta Prometheus

# IV. Sistema de Monitorización

## Grafana


- Open Source Dashboard
  - Permite visualizar datos de series temporales
  - La visualización de los datos es personalizable
  - Permite mostrar la información por partes, alternando de forma automática entre ellas.
- Permite añadir plugins para mejorar su integración con otras bases de datos o agregadores
- Genera alertas
  - Permite definir umbrales
- Permite realizar transformaciones sobre los datos
- Soporta Prometheus

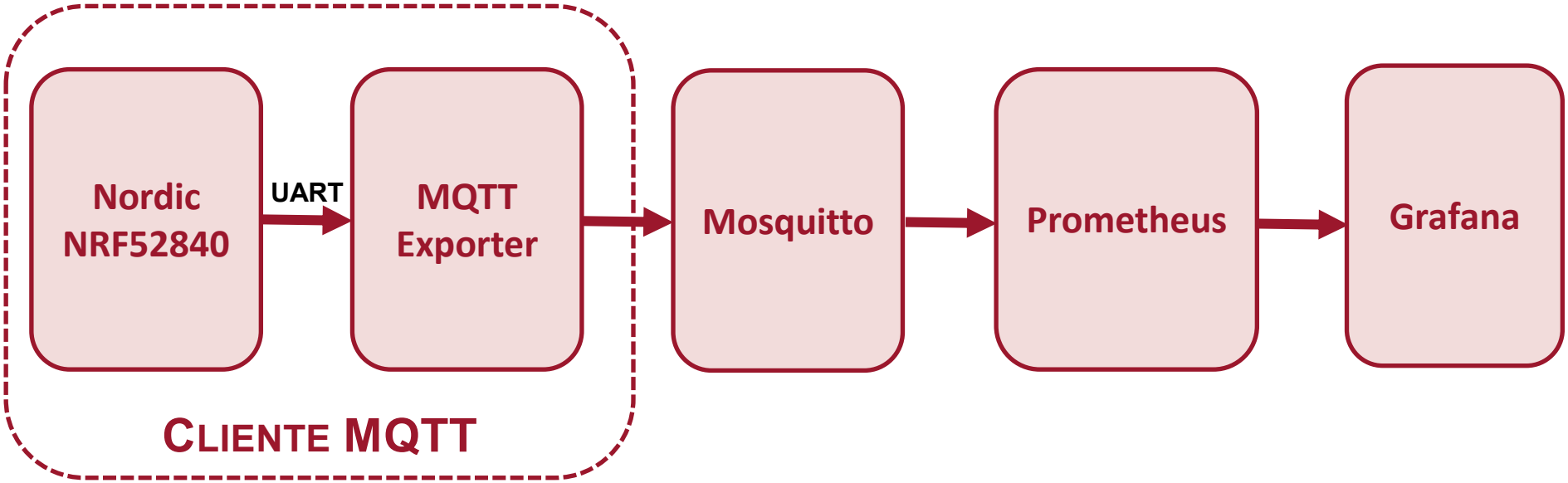



**Reset password:**

```
grafana-cli admin reset-admin-password <new password>
```

# V. Sistema Propuesto

## Diagrama de Bloques



# VI. Ejercicios

## Ejercicio 1 (5 puntos)

- Haciendo uso de los archivos facilitados, se pide
  - Leer el sensor de temperatura interno de la tarjeta NORDIC NRF52840
  - Enviar la temperatura, expresada en grados Farenheit, a la aplicación MQTT-exporter cada 2 segundos
  - Crear un topic denominado "temp_F"
  - Representar la información en Prometheus
  - Representar la información en Grafana

**Nota:** añada todas las capturas de pantalla y ficheros logs que necesite para demostrar el correcto funcionamiento

# VI. Ejercicios

## Ejercicio 2 (5 puntos)

- Se pide diseñar una aplicación que muestre en Grafana los siguientes paneles:
  - La medida tomada por el sensor de temperatura interno expresado en grados centígrados.
    - El nombre del topic creado deberá ser "temp_c"
  - La medida tomada por el sensor de temperatura interno expresado en grados Fahrenheit.
    - El nombre del topic creado deberá ser "temp_F"
  - El valor del pulsador que integra la tarjeta. Para ello considere que su funcionamiento será como el de un interruptor. Una pulsación activaría su estado. Una segunda pulsación, desactivaría su valor.
    - El nombre del topic creado deberá ser "switch"

**Nota:** añada todas las capturas de pantalla y ficheros logs que necesite para demostrar el correcto funcionamiento