MQTT y logs bajo el mismo techo

Centralización y securización de datos de sensores industriales usando software de centralización de logs

Descripción breve

En este trabajo se ha implementado un sistema que permite el envío de datos MQTT a un servidor centralizado de recogida de logs en la nube (Stack ELK), prestando especial atención a la seguridad de los datos, desde que salen del sensor hasta que llegan a la nube.

Sergio Faraldo Graña

Ciberseguridad en entornos industriales – Trabajo tutelado

Índice

1 Introducción

1.1 Objetivo

El objetivo de este proyecto es securizar la comunicación entre clientes MQTT y un broker, y el envío seguro de los datos recogidos por el broker a un servidor de logging y analíticas de log en la nube.

1.2 Motivación

La recogida de datos sobre un proceso industrial es un proceso esencial en cualquier fábrica, dado que permite encontrar puntos de posible mejora en el sistema, así como detectar fallos. Actualmente, los sistemas industriales incorporan cada vez más elementos informáticos, pero la recogida y el almacenamiento de logs y métricas de estos sistemas se hace por separado de los datos de sensores industriales. La integración del logging de estos sistemas informáticos con la obtención de datos de sensores en un mismo sistema podría ser de gran ayuda en la búsqueda de errores y en la monitorización del estado del sistema en su conjunto. Tener todos los datos en la misma interfaz puede hacer visibles relaciones entre datos que no se podrían haber visto si la monitorización del "lado industrial" y del "lado informático" estuvieran en interfaces separadas.

Dicho esto, también es cierto que hay herramientas especializadas como Node-RED que proporcionan funcionalidades muy específicas del entorno de IIoT, las cuales no están disponibles en programas orientados a logging. Dado esto, es importante que la unificación de logging y monitorización industrial no impida el uso de software especializado cuando este sea necesario.

1.3 Descripción del sistema

El sistema creado usa una suite de recogida y análisis de logs para centralizar los datos de sensores de una o más plantas industriales. Al realizar la centralización utilizando un software de recogida y análisis de logs, es posible añadir al mismo los logs de los ordenadores que formen parte del sistema industrial.

El software de recogida y análisis de logs escogido es el stack ELK. Se ha elegido este software por ser de código abierto y por ser uno de los más usados. Debido a su popularidad, hay una gran cantidad de información online, además de la completa y bien organizada documentación oficial. Además, el hecho de que es de código abierto permite que se pueda compilar para plataformas no soportadas oficialmente (como es el caso de la Raspberry Pi).

Es importante remarcar que el sistema creado actúa simplemente como otro cliente MQTT, permitiendo la convivencia de este sistema con otros más especializados.

1.4 ¿Qué es el Stack ELK?

El Stack ELK es una suite software de recolección, procesamiento, almacenamiento y visualización de logs. En este proyecto, se utiliza para recolectar los mensajes MQTT de la Raspberry Pi, almacenarlos en el servidor de cloud y visualizarlos. El stack está formado por los siguientes programas:

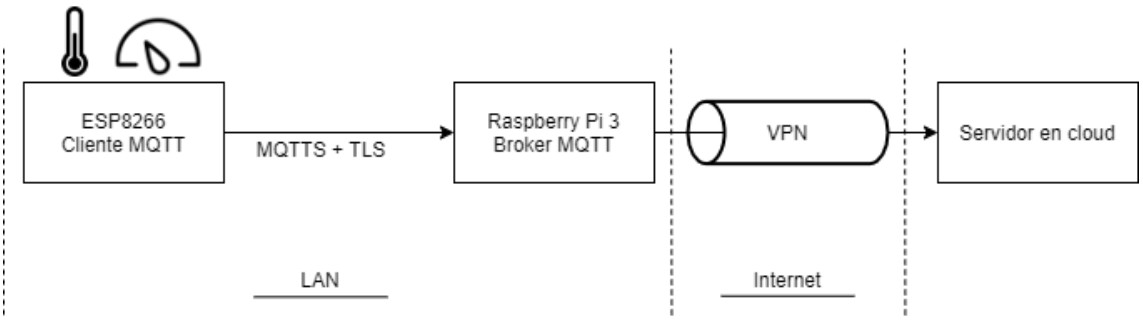
- Kibana: Interfaz web de visualizado, accede a los datos almacenados en Elasticsearch y proporciona herramientas para visualizarlos de forma gráfica.
- Elasticsearch: Base de datos
- Logstash: Programa de procesamiento de logs. Se encarga de recibir los datos y procesarlos antes de enviarlos a Elasticsearch para su almacenamiento.

- Beats: Programas ligeros de recopilación de datos. Se instalan en los equipos en los que se quieran recopilar logs, métricas u otro tipo de información, y la envían a Logstash para su procesado. Existen bastantes Beats distintos, como Filebeat, Metricbeat, Packetbeat, Heartbeat… En este proyecto se utiliza únicamente Filebeat.

2 Funcionamiento del sistema

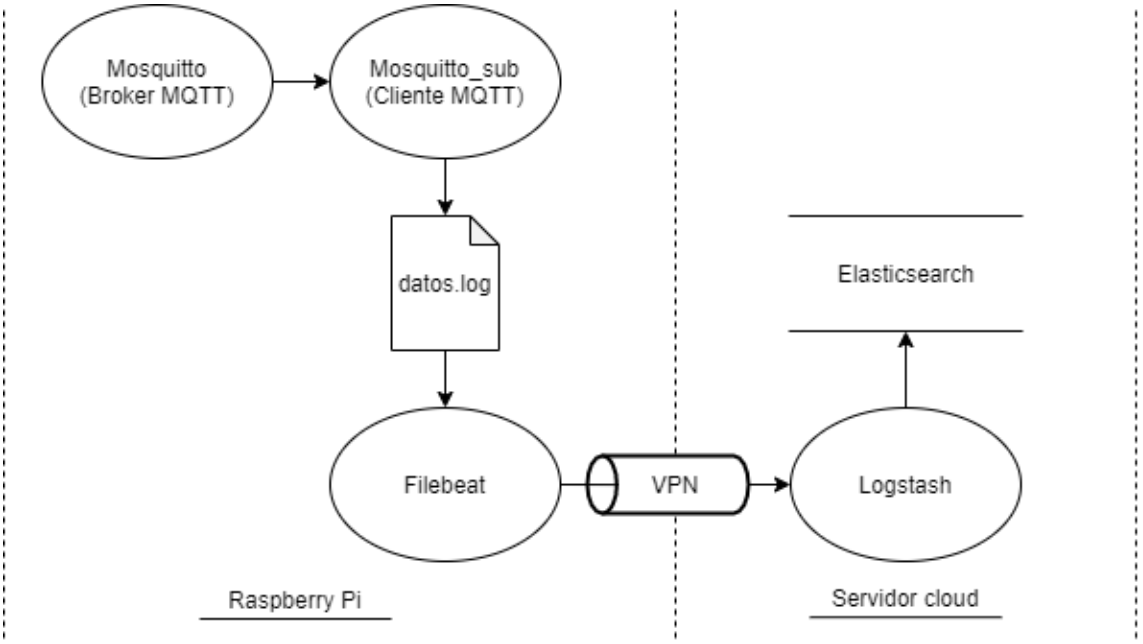
2.1 Esquema general

En este caso se ha realizado la implementación de un único sistema, pero para añadir más plantas al sistema bastaría con replicar la configuración de la LAN en las distintas plantas.



En la red local se encuentra un ESP8266, al cual se ha conectado un sensor de humedad y temperatura. Este comunica los datos sacados del sensor a una Raspberry Pi, utilizando el protocolo MQTT, protegido con TLS. Estos datos se envían a un servidor en la nube a través de una VPN.

2.2 Flujo de datos Raspberry Pi – Servidor cloud



Para extraer los datos que se envían al broker MQTT, se utiliza el cliente Mosquitto_sub. Este cliente se subscribe a todos los topics y apunta los mensajes MQTT en disco. Filebeat lee los mensajes de disco y los envía a Logstash, donde se procesan. Los datos son finalmente almacenados en Elasticsearch. Para poder visualizarlos, se ha creado un dashboard en Kibana.

La interfaz web de Kibana únicamente está expuesta al VPN, no a internet en general para evitar problemas.

3 Implementación

3.1 Servidor cloud

Para el servidor cloud, se ha optado por un *Droplet* de DigitalOcean. Dispongo de crédito gratuito en esta web, y no debería significar un problema el uso de un proveedor de servicios cloud distinto.

Para este proyecto son necesarios al menos 4GB de RAM, con lo cual se ha optado por un *Droplet* con 2 CPUs virtuales y 4GB de RAM. Para el sistema operativo, he escogido Ubuntu 18.04, dado que es la última versión de una distribución de Linux con la que tengo mucha experiencia.

A continuación, mostraré los pasos realizados para instalar y configurar todo el software necesario:

3.1.1 Instalación del servidor VPN

La instalación se ha realizado con el script *openvpn-install*, disponible en GitHub. Esto no solamente simplificó la creación del servidor, sino también simplifica el proceso de añadir nuevos clientes a la red y revocar el acceso a clientes existentes.

```
apt update
apt upgrade
curl -O https://raw.githubusercontent.com/angristan/openvpn-
    install/master/openvpn-install.sh
chmod +x openvpn-install.sh
AUTO_INSTALL=y ./openvpn-install.sh
```

3.1.2 Instalación del stack ELK

En el servidor se han instalado los programas Elasticsearch, Logstash y Kibana. Para ello se ha utilizado como base una guía disponible en la página de comunidad de DigitalOcean. No se ha seguido al pie de la letra, algunos pasos se han omitido y otros se han modificado ligeramente. También se ha hecho un esfuerzo por reducir al máximo el número de archivos a editar manualmente, con el objetivo de que esta configuración se pueda dockerizar fácilmente.

En los siguientes comandos se instalan los tres programas de ELK mencionados y se crea un certificado para cifrar los datos que se envíen a Logstash. No es estrictamente necesario, ya que los datos se transmiten a través de una VPN, pero no es mala idea, dado que va a haber otras máquinas en la misma VPN.

```
apt install openjdk-8-jdk
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
    apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" |
    sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
apt update
apt upgrade
apt install elasticsearch
echo "network.host: localhost" >> /etc/elasticsearch/elasticsearch.yml
apt install kibana
```

```
echo "server.host: `ip addr show dev tun0 | grep -o "inet [0-9]*\.[0-
    9]*\.[0-9]*\.[0-9]*" | grep -o "[0-9]*\.[0-9]*\.[0-9]*\.[0-9]*"`">>
    /etc/kibana/kibana.yml
apt install logstash

openssl genrsa -out /etc/logstash/logstash.key 2048
openssl req -new -x509 -days 3650 -nodes -out
    /etc/logstash/logstash.crt -key /etc/logstash/logstash.key -subj
    "/C=ES/ST=Pontevedra/L=Vigo/O=SInd/CN= Logstash" -extensions v3_ca
    -addext "subjectAltName=IP: 10.8.0.1"
openssl pkcs8 -in /etc/logstash/logstash.key -topk8 -out
    /etc/logstash/logstash-pkcs8.key -nocrypt
rm /etc/logstash/logstash.key
mv /etc/logstash/logstash-pkcs8.key /etc/logstash/logstash.key
chmod 666 /etc/logstash/logstash.key
```

Tras la ejecución de estos comandos, ya casi está finalizada la configuración del servidor, solamente falta añadir el pipeline de Logstash, adjunto a este archivo (*00-mqtt.conf*) a la carpeta */etc/logstash/conf.d*.

En caso de que el servidor no tenga la IP 10.8.0.1, será necesario cambiar la tercera línea del archivo (*host => "10.8.0.1"*).

Una vez hecho esto, lo único que falta es habilitar los servicios y reiniciar el servidor.

```
systemctl enable elasticsearch
systemctl enable kibana
systemctl enable logstash
reboot
```

Desde ahora, cada vez que se arranque el servidor, arrancarán también todos los servicios necesarios para el procesado y almacenamiento de datos, así como la interfaz web. En la

3.2 Raspberry Pi

En este caso parto de una Raspberry Pi 3, con Raspbian Buster. Se han instalado Mosquitto y Filebeat. La instalación de Mosquitto se realiza a través del repositorio de Raspbian, pero Filebeat se ha tenido que compilar de cero, dado que la empresa responsable del desarrollo del Stack ELK no proporciona binarios para dispositivos ARM (como la Raspberry Pi).

3.2.1 Instalación de Mosquitto

El primer paso en la configuración de la Raspberry Pi es la instalación y configuración de Mosquitto. En los siguientes comandos está la configuración de Mosquitto realizada, con el usuario cambiado a "user" y la contraseña a "passwd". Este usuario y contraseña es necesario incluirlos en el código del ESP8266. También será necesario incluir el fingerprint del certificado que se va a generar para el broker.

```
sudo apt update
sudo apt upgrade
sudo apt install mosquitto
echo "user:passwd" > mqttpwdfile
mosquitto_passwd -U mqttpwdfile
sudo mv mqttpwdfile /etc/mosquitto/
sudo sh -c 'echo "allow_anonymous false" >>
    /etc/mosquitto/mosquitto.conf'
sudo sh -c 'echo "password_file /etc/mosquitto/mqttpwdfile" >>
    /etc/mosquitto/mosquitto.conf'
sudo service mosquitto restart
sudo apt install openssl
```

```
openssl genrsa -des3 -passout pass:passwd -out ca.key 2048
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt -passin
    pass:passwd -subj "/C=ES/ST=Pontevedra/L=Vigo/O=SInd/CN=Ca-MQTT-
    SInd"
openssl genrsa -out broker.key 2048
openssl req -new -out broker.csr -key broker.key -subj
    "/C=ES/ST=Pontevedra/L=Vigo/O=SInd/CN= MQTT-SInd" -extensions v3_ca
    -addext "subjectAltName=IP: 192.168.1.46"
openssl x509 -req -in broker.csr -CA ca.crt -CAkey ca.key -
    CAcreateserial -out broker.crt -passin pass:passwd -days 3650
openssl verify -CAfile ca.crt broker.crt
sudo cp ca.crt /etc/mosquitto/ca_certificates/
sudo cp broker.crt broker.key /etc/mosquitto/certs/
sudo sh -c 'echo "cafile /etc/mosquitto/ca_certificates/ca.crt" >>
    /etc/mosquitto/mosquitto.conf'
sudo sh -c 'echo "certfile /etc/mosquitto/certs/broker.crt" >>
    /etc/mosquitto/mosquitto.conf'
sudo sh -c 'echo "keyfile /etc/mosquitto/certs/broker.key" >>
    /etc/mosquitto/mosquitto.conf'
sudo sh -c 'echo "tls_version tlsv1.2" >>
    /etc/mosquitto/mosquitto.conf'
openssl x509 -noout -in broker.crt -fingerprint
```

Este último comando devuelve el fingerprint del certificado del broker. Es necesario guardarlo para luego poder incluirlo en el código del ESP8266.

3.2.2 Envío de datos al servidor cloud

Una vez configurado Mosquitto, pasamos a configurar todo lo necesario para enviar los datos al servidor cloud: el VPN, Filebeat y Mosquitto_sub.

Mosquitto_sub es un cliente sencillo de MQTT que permite imprimir por pantalla todos los mensajes MQTT de los *topics* a los que está suscrito. Con este programa y una herramienta de rotación de logs he creado un script para almacenar en disco los mensajes de MQTT. Tanto el script como la conexión al VPN se ejecutarán a través de servicios, para poder arrancarlos y detenerlos de forma fácil.

Los archivos con los mensajes MQTT serán leídos por Filebeat, el cual los enviará al servidor cloud. Para evitar que se llene el disco de la Raspberry Pi con estos logs, he creado un script de Python que se ejecutará semanalmente (programado con crontab) para eliminar los archivos cuyos datos ya han sido transmitidos al servidor.

```
sudo apt install mosquitto-clients apache2-utils openvpn python3
    python3-pip -y
sudo pip3 install pathlib
```

Una vez ejecutados los comandos anteriores, es necesario copiar el archivo *client.ovpn*, que se debería haber generado en la carpeta */root* del servidor a */home/pi* en la Raspberry. También se deben copiar a esta carpeta los scripts *cleanFiles.py* y *mqtt_sub.sh*, adjuntos a este documento.

También es necesario copiar los archivos *stVPN.service* y *stMQTT.service* a la carpeta */etc/systemd/system*. Esto permitirá ejecutar como servicio el cliente de OpenVPN con la configuración correspondiente a la VPN cloud-Raspberry y el script que se encarga de la escritura de mensajes MQTT en disco.

Una vez copiados todos los archivos, añadimos el script de limpieza al crontab de root, y habilitamos los servicios para que se enciendan al arrancar la Raspberry.

```
sudo sh -c ' (crontab -l; echo "@weekly /usr/bin/python3
    /home/pi/cleanFiles.py
    /var/lib/filebeat/registry/filebeat/data.json") | crontab -'
sudo systemctl enable stVPN
sudo systemctl enable stMQTT
```

3.2.2.1 *Compilación e instalación de Filebeat*

Para compilar e instalar Filebeat, se utilizará el script easyBEATS, disponible en GitHub. Es necesario especificarle la versión de Filebeat que se quiere instalar, así como el ID de commit correspondiente a dicha versión. Como se ha instalado la versión 7.6.1 de Elasticsearch, Kibana y Logstash, es necesario instalar la misma versión de Filebeat. Se puede comprobar cuál es el ID de commit correspondiente a cada versión en https://github.com/elastic/beats/releases.

Con los siguientes comandos, se descargará, configurará y ejecutará el script easyBEATS para compilar e instalar Filebeat.

```
git clone https://github.com/RaoulDuke-Esq/easyBEATS.git
cd easyBEATS
sudo chmod 755 easyBEATS
sed -i "s/BEAT_VERSION_NUM=.*/BEAT_VERSION_NUM=\"7.6.1\"/g" easyBEATS
sed -i "s/BEAT_VERSION=.*/BEAT_VERSION=\"c1c4943\"/g" easyBEATS
sed -i "s/BEAT_NAME=.*/BEAT_NAME=( filebeat )/g" easyBEATS
mkdir /root/easyBEATS/services/
sudo ./easyBEATS
```

Lo último que falta es sustituir el archivo */etc/filebeat/filebeat.yml* con el *filebeat.yml* adjunto a este documento, copiar el archivo */etc/logstash/logstash.crt* del servidor de la nube a la carpeta */home/pi*, habilitar el servicio de Filebeat y reiniciar la Raspberry.

```
sudo systemctl enable filebeat
sudo reboot
```

3.3 ESP8266

Se incluye el sketch Arduino para programar el ESP8266. Es necesario modificar los datos de la red Wifi, así como las credenciales de MQTT y el fingerprint del certificado TLS del broker. El fingerprint tiene que estar en el formato en que está en el sketch (pares de valores hexadecimales separados por espacios).

Se ha utilizado la librería Adafruit MQTT (github.com/adafruit/Adafruit_MQTT_Library/) para el envío de mensajes MQTT y la librería DHTesp (github.com/beegee-tokyo/DHTesp) para leer los valores del sensor de humedad y temperatura. Se han utilizado los sketches de ejemplo de ambas librerías.

Toda la configuración se encuentra en la parte superior del sketch, para facilitar el acceso:

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
#include "DHTesp.h"

/************************ WiFi *********************************/

#define WLAN_SSID       "-Escribe aquí el ssid de la red-"
#define WLAN_PASS       "-Y aquí la contraseña-"

/************************ MQTT *********************************/

#define IP_BROKER       "192.168.0.4"
```

```
#define PUERTO_BROKER   1883
#define USUARIO         "user"
#define PASSWORD        "passwd"

// Feeds de humedad y temperatura
#define FEED_HUM        "/feeds/0/humid"
#define FEED_TEM        "/feeds/0/temp"

// fingerprint SHA1 del certificado del broker
static const char *fingerprint PROGMEM = "34 09 42 B2 91 97 58 DA EC
    EB 13 D0 FC 41 5A 0C 27 91 77 26";

// Cambio mínimo necesario para que se envíe un mensaje MQTT
#define HUM_DELTA 1
#define TEM_DELTA 0.1


// Tiempo entre envíos (milisegundos)
#define SLEEP_TIME 60000
```

El loop principal se ejecuta una vez cada *SLEEP_TIME* milisegundos (en este caso, una vez cada minuto). En el loop, se comprueba si los valores han cambiado por encima de los especificados en *HUM_DELTA* y *TEM_DELTA*. Si han cambiado suficiente, se envía un publish con los nuevos valores. En caso de que se haya perdido la conexión con el broker, se intenta volver a establecer.

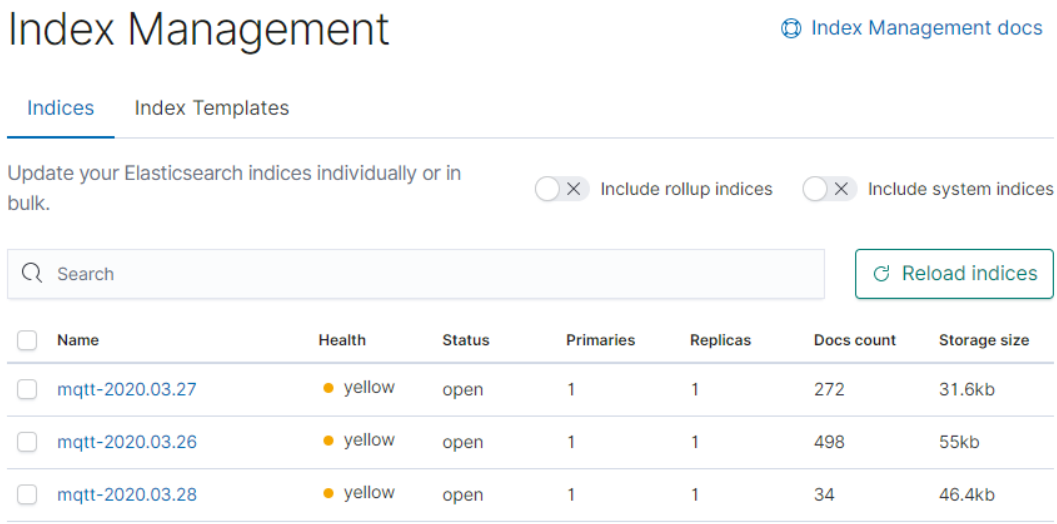
```
void loop() {
  // Lectura de sensores
  float humidity = dht.getHumidity();
  float temperature = dht.getTemperature();
  // Si los valores han cambiado suficiente
  if(prevhum-humidity>HUM_DELTA || humidity-prevhum>HUM_DELTA ||
    prevtemp-temperature>TEM_DELTA || temperature-prevtemp>TEM_DELTA){

    // Asegurarse de que aún hay conexión MQTT
    MQTT_connect();

    Serial.print(F("\nEnviando valores: Temp="));
    Serial.print(temperature);
    Serial.print(", humid=");
    Serial.println(humidity);
    if (humid.publish(humidity) && temp.publish(temperature)) {
      Serial.println(F("OK"));
        // asigno los valores actuales a prevhum y prevtemp
        prevhum = humidity;
        prevtemp = temperature;
    } else {
      Serial.println(F("ERROR"));
    }
  }

  // espero SLEEP_TIME minuto antes de volver a ejecutar el loop
  delay(SLEEP_TIME);

}
```

El sketch completo está en el archivo *esp8266_MQTT.ino*.

4 Resultado

Una vez realizada toda la instalación, simplemente con encender todos los dispositivos (el ESP8266, la Raspberry Pi y el servidor), ya se comienzan a recoger y almacenar los datos.

En Kibana se puede configurar la visualización como se desee, los datos provenientes de MQTT se almacenan bajo el índice mqtt-[yyy].[MM].[dd]:



Los datos son guardados almacenando la fecha y la hora de medición (sacada de la fecha y hora en la que el broker transmitió los datos a los clientes), la IP de la que proviene (útil en caso de que haya varias plantas industriales utilizando este sistema), el *topic*, y el *value*.

Para este caso concreto, de datos de humedad y presión se ha creado un dashboard donde poder visualizar los últimos datos y el histórico (el archivo de configuración, importable a través de la interfaz web de kibana, está adjunto con el nombre *dashboard.ndjson*):

Con muchas otras medidas se podrían crear dashboards más avanzados y también se podrían emplear algunas de las herramientas más avanzadas que ofrece este software, como las herramientas de visualización o detección de anomalías mediante machine leraning o el (por lo de ahora aún en desarrollo) avanzado sistema de alertas.

5 Conclusiones

Mediante la utilización de un sistema de centralización de métricas y logs propio del ámbito de la informática se ha conseguido crear una solución simple, segura y cohesionada. La seguridad del sistema queda más que reforzada:

- En la red local: Los datos van encriptados mediante TLS, y, además, también hay que tener en cuenta la naturaleza de este tipo de redes, que suelen estar protegidas además por firewalls o, al menos, por un NAT.
- En la conexión “red local-nube”: Los datos van con dos capas de encriptación: la configurada en Filebeat, y la de la red VPN.

El software de gestión de logs utilizado ofrece una gran cantidad de opciones en cuanto al análisis de los datos recibidos y, dado que están en un proceso de desarrollo constante, con cada actualización se añaden más herramientas al arsenal de utilidades de análisis y visualización que este proporciona.

Todo esto se ha logrado sin limitar la posibilidad de añadir herramientas específicas al sistema, dado que no se ha realizado ninguna modificación a la infraestructura MQTT básica, más allá del securizado mediante TLS.

El sistema creado representa, pues, una buena forma de centralizar la monitorización de un entorno industrial completo, desde las máquinas hasta los ordenadores que las controlan, sin que ello suponga cambiar los sensores o máquinas ya instaladas.