



SASE 2022

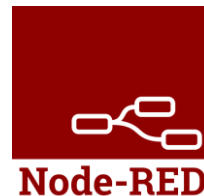
SIMPOSIO ARGENTINO DE
SISTEMAS EMBEBIDOS

Implementación de una plataforma de servicios IoT

Ing. Juan Eduardo Salvatore
Matías Gabriel Busum Fradera
Facundo Ariel Chazarreta

TEMARIO

- Introducción.
- Elección y configuración de un Broker MQTT.
- Utilización del firewall *ufw*.
- Captura y almacenamiento de los mensajes en una base de datos utilizando NodeRED y MySQL.
- Configuración de un proxy inverso con NGINX.
- Instalación de un certificado SSL/TLS gratuito con Let's Encrypt.





Introducción

En general una arquitectura IoT la podemos descomponer en distintas etapas:

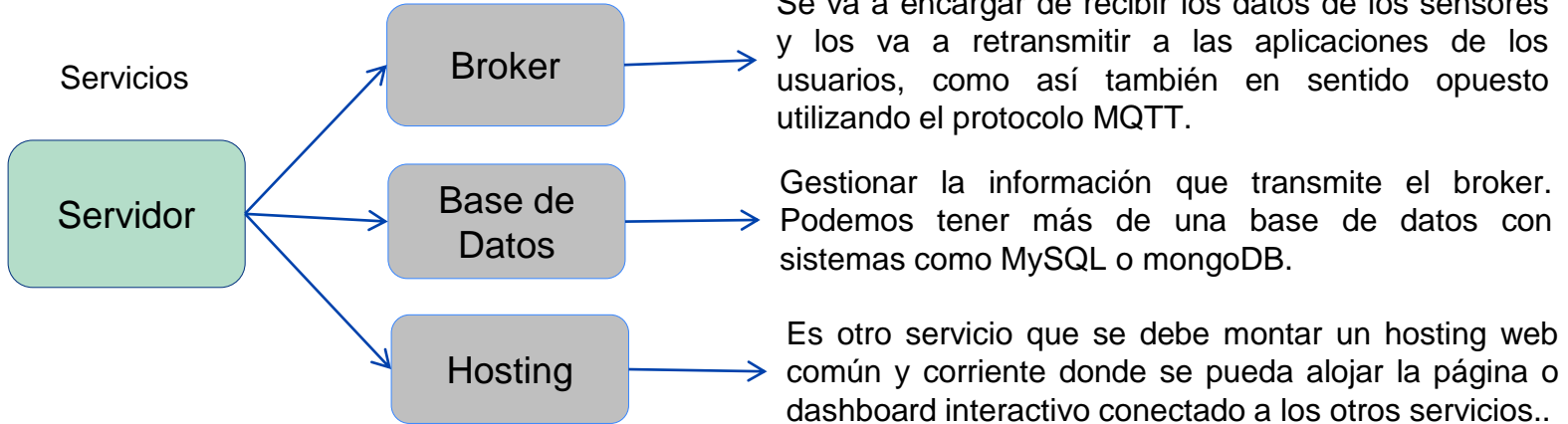
- Capa Objetos
- Capa Red
- Capa Servicios
- Capa Aplicaciones

Introducción

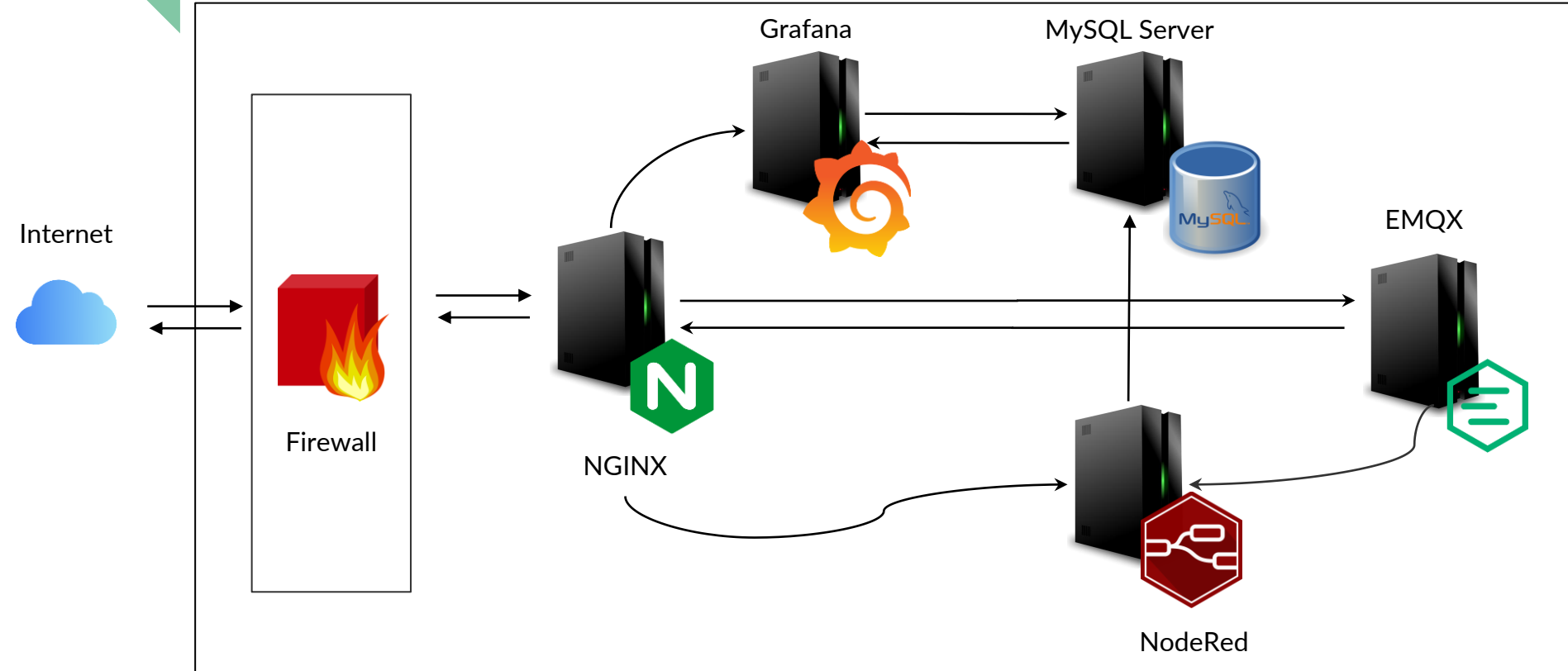


Introducción

CAPA DE SERVICIOS:



ESQUEMA DE LA PLATAFORMA DE SERVICIOS



ELECCIÓN DE UN BROKER



EMQX

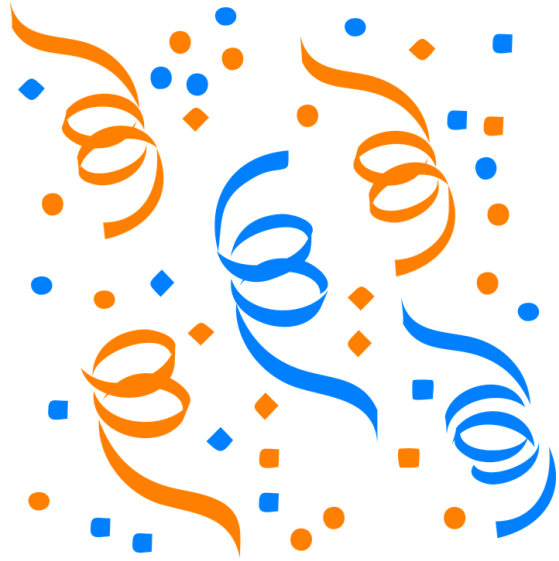
HIVEMQ



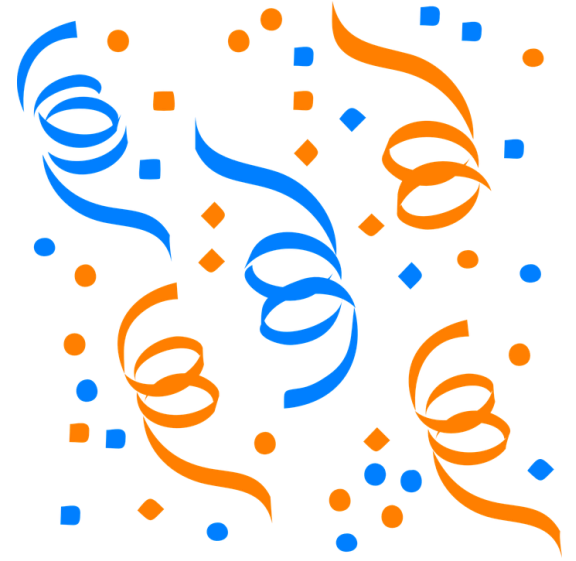


ELECCIÓN DE UN BROKER

 MQTT



EMQX





EMQX

DOCUMENTACIÓN: <https://www.emqx.io/docs/en/v5.0/>

EMQX Enterprise Docs MQTT Developers

10.2k

Download

Try Cloud

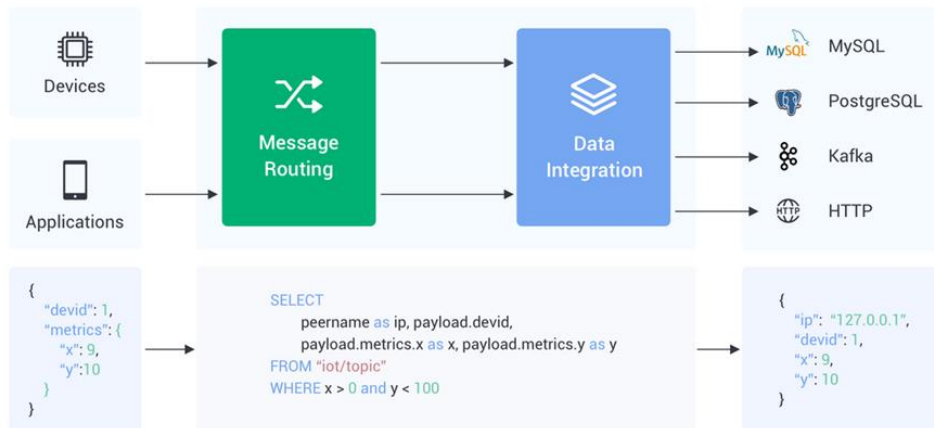
Search

- Introduction
- Getting Started
- Deployment & Provisioning
- Configuration
- Management
- Security
- Observability
- Data Integration
 - Introduction to Data Integration
 - Rules
 - Data Bridges
 - Examples and Tutorials
- Gateway
- Advanced MQTT Features
- Plugins & Extensions
- Design & Implementation
- Load Verification

Introduction to Data Integration

Data integration is a data processing and distribution component of EMQX based on the publish/subscribe model. Through simple and visual configuration, message flow and device events can be integrated with messaging brokers such as Kafka and RabbitMQ, as well as various SQL/NoSQL or time series databases.

EMQX provides a real-time, concise and efficient data integration scheme by combining **Rules** and **Data Bridges**. The rules are used to process messages or events, and the data bridge is used to connect the data system.



- Edit this page
- Request docs changes

What's on this page

- Rules
- Data Bridges
- Flow Charts

v5.0

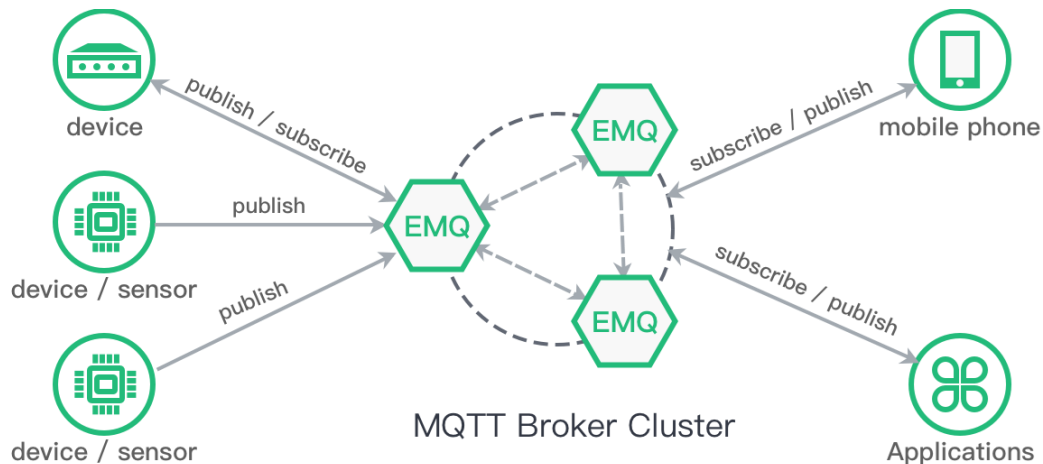




EMQX

De acuerdo a su página web oficial, EMQX se define como:

“El bróker MQTT distribuido más escalable del mundo con un motor de procesamiento de mensajes en tiempo real de alto rendimiento, que impulsa la transmisión de eventos para dispositivos IoT a gran escala.”





Comparación de EMQX con otros brokers:

Implementation	Open Source	Type	MQTT 5.0, SSL/TLS, TCP, WS/WSS	Thread Safety	Cross compile	Support operating systems
EMQX	x	x	x	x	x	CentOS, Debian, Docker, Mac OS X, Ubuntu, Red Hat Enterprise Linux, Windows 10-7, Raspbian (Raspberry Pi OS)
Mosquitto	x	x	x	x	x	BSD, Linux, macOS, QNX, Window.
RabbitMqtt	x	x	x	x	x	Linux, Window, Windows Server, Mac OS, Solaris, FreeBSD



EMQX

Beneficios

- **Escala masiva:** Posibilidad de escalar a 100 millones de conexiones MQTT simultáneas con un solo clúster EMQX.
- **Alto rendimiento:** Capacidad para procesar millones de mensajes MQTT por segundo en un único broker.
- **Baja latencia:** Latencia del orden de los milisegundos en la entrega de mensajes.
- **Totalmente compatible con MQTT 5.0:** 100% compatible con MQTT 5.0 y 3.x estándar para mejor escalabilidad, seguridad y rentabilidad.
- **Alta disponibilidad:** Lograr una alta disponibilidad y escalabilidad horizontal a través de una arquitectura distribuida sin maestros.
- **Cloud-Native & K8s:** Mayor facilidad para realizar el despliegue utilizando tecnologías cloud con Kubernetes.



EMQX

Conectividad

Soporte completo para MQTT v3.1, v3.1.1 and v5.0.

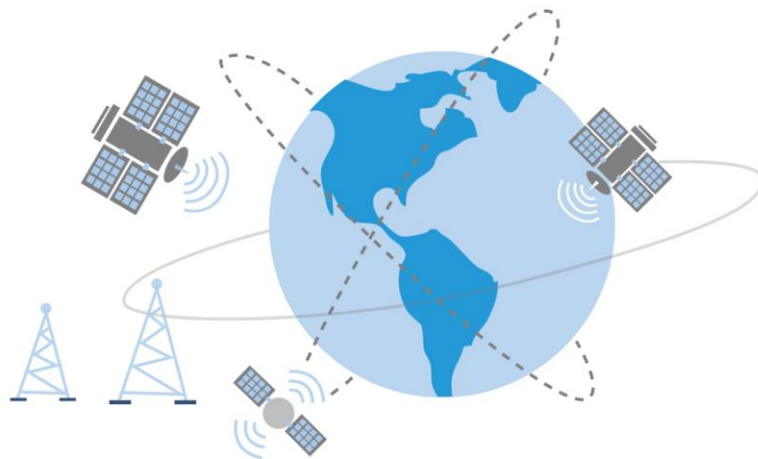
- Mensajes QoS 0, QoS 1 y QoS 2.
- Last Will.
- Mensajes retenidos.

4 protocolos de transporte:

- TCP.
- TLS.
- WebSocket.
- QUIC.

Publicación de mensajes a través de HTTP.

HTTP API para la integración con sistemas externos (obtener datos de clientes, publicación de mensajes y creación de reglas, entre otros).



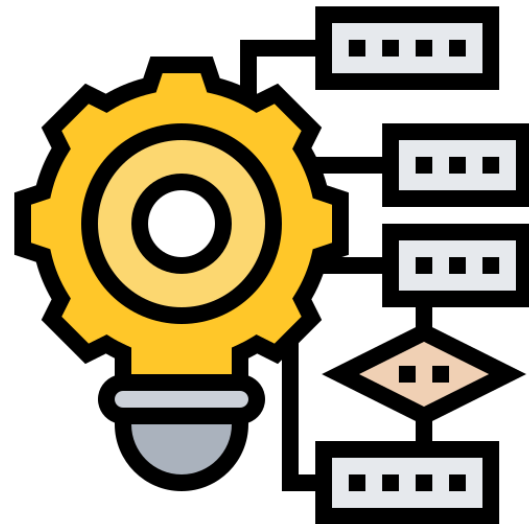


EMQX

Motor de reglas

El motor de reglas de EMQX se utiliza para configurar el flujo de mensajes y el procesamiento de eventos en los dispositivos y las respuestas asociadas a estos eventos.

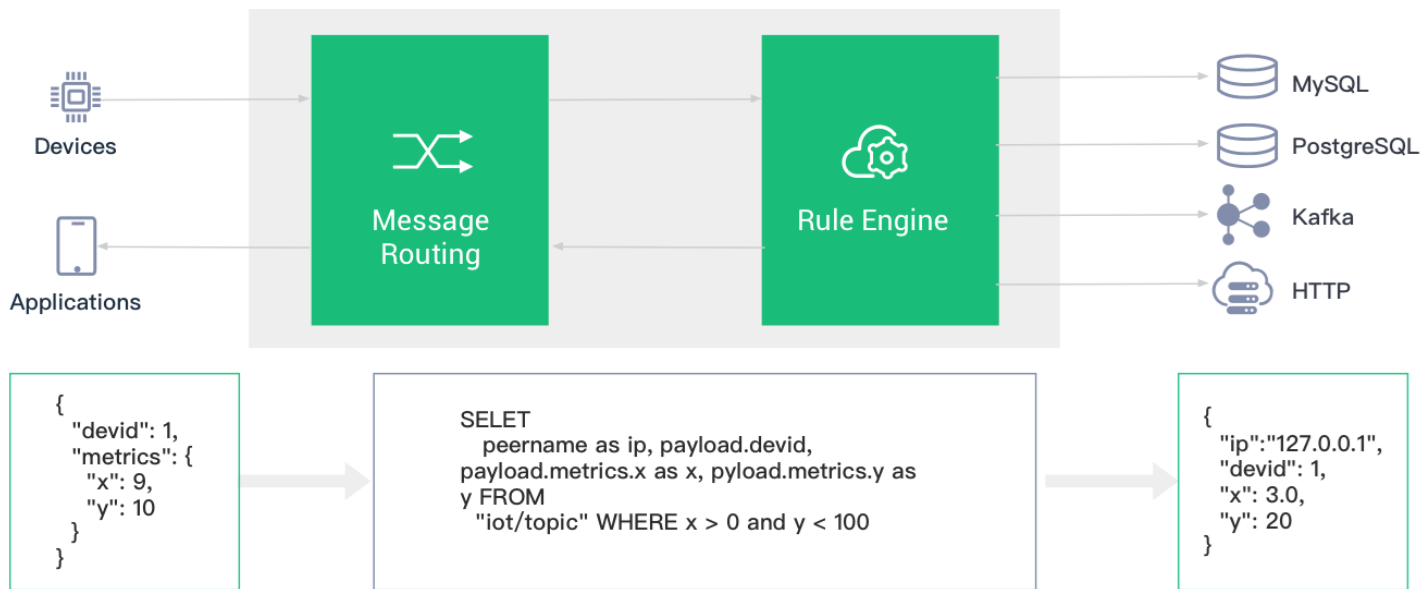
EMQX activa el motor de reglas cuando se publica un mensaje o se activa un evento. En ese momento, se ejecutan las sentencias SQL para filtrar y procesar la información de contexto en los mensajes y eventos.





EMQX

Motor de reglas





EMQX

Autenticación

El broker EMQX soporta múltiples formas de autenticación, basadas en:

- Nombre de usuario y contraseña,
- ClientID,
- JWT,
- LDAP.

Integración con bases de datos MySQL , Redis, PostgreSQL, MongoDB.

Autenticación a través de HTTP.





EMQX

ACL

Publicación/Subscripción ACL se refiere a los permisos de control para las operaciones de **PUBLICAR/SUSCRIBIRSE** a un determinado t3pico.

EMQX soporta la utilizaci3n de varias fuentes de reglas ACL:

- Archivos de configuraci3n.
- Bases de datos externas (MySQL, PostgreSQL, Redis, MongoDB).
- APIs HTTP.





EMQX

Instalación

Existen distintas maneras de instalar EMQX:

- EL7 (RedHat 7, CentOS 7)
- EL8 (RedHat 8, RockyLinux 8, AmazonLinux 2022)
- Raspbian 10
- Debian 9
- Debian 10
- Ubuntu 16.04
- Ubuntu 18.04
- Ubuntu 20.04
- macOS 10
- macOS 11
- Windows Server 2019

También se puede correr a través de Docker y compilar el código fuente.





EMQX

Puertos que utiliza:

<i>Aplicación</i>	<i>Puerto</i>
	<i>Descripción</i>
EMQX	1883
	MQTT/TCP
	8883
	MQTT/SSL
	8083
	MQTT/WS

Si queremos utilizar cada uno de estos servicios, debemos abrir los puertos correspondientes en el firewall

Dashboard



¿Qué es un firewall?

De acuerdo a CISCO:

“Un firewall es un dispositivo de seguridad de la red que monitorea el tráfico de red – entrante y saliente– y decide si permite o bloquea tráfico específico en función de un conjunto definido de reglas de seguridad.”

Pueden ser implementados en

- hardware,
- software,
- o una combinación de ambos.

Para este tutorial, utilizaremos ufw (Uncomplicated Firewall).



UFW

Desarrollado por Ubuntu para ser de fácil uso.

Si queremos permitir que los usuarios puedan ingresar a los servicios proporcionados por EMQX, debemos habilitar cada uno de los puertos que utiliza en el firewall:

```
sudo ufw allow 1883
sudo ufw allow 8883
sudo ufw allow 8083
sudo ufw allow 8084
sudo ufw allow 18083
```

MQTT/TCP
MQTT/SSL
MQTT/WS
MQTT/WSS
Das

Ver las reglas/servicios/puertos abiertos:

```
sudo ufw status verbose
```

Por último, para habilitar el firewall:

```
sudo ufw enable
```

```
Output
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
22/tcp ALLOW IN Anywhere
```



Utilizamos systemctl para arrancar EMQX:

```
systemctl start emqx
```

Si queremos que se inicie automáticamente cuando arranque el sistema:

```
systemctl enable emqx
```

Para comprobar el estado:

```
systemctl status emqx
```





EMQX

Ya podemos entrar al dashboard:

Usuario y contraseña por defecto: public

The screenshot shows the EMQX dashboard interface. On the left is a navigation sidebar with the following items: Monitor (active), Clients, Topics, Subscriptions, Rule Engine, Analysis, Plugins, Modules, Tools, Alarms, Settings, and General. The main content area is titled "Overview" and includes a user dropdown menu showing "emqx@127.0.0.1".

Broker Information:

System Name	Version	Uptime	System Time
EMQ X Broker	4.3.5	72 days, 22 hours, 51 minutes, 29 seconds	2022-08-17 19:00:25

Nodes(1):

Name	Erlang/OTP Release	Erlang Processes (used/available)	CPU Info (1load/5load/15load)	Memory Info (used/total)	MaxFds	Status
emqx@127.0.0.1	23.2.7.2-emqx-2/11.1.8	716 / 2097152	32.91 / 17.17 / 14.04	141.54M / 229.04M	524288	Running

Stats(1):

Name	Connections (count/max)	Topics (count/max)	Retained (count/max)	Sessions (count/max)	Subscriptions (count/max)	Subscriptions Shared (count/max)
emqx@127.0.0.1	32 / 32	15 / 19	4 / 4	32 / 32	15 / 23	0 / 0

Metrics:

Client	Count	Delivery	Count	Session	Count
connected	231	dropped	0	created	231
authenticate	684	dropped.no_local	0	resumed	0
auth.anonymous	0	dropped.too_large	0	takeovered	0
check_acl	4266	dropped.qos0_msg	0	discarded	5
subscribe	693	dropped.queue_full	0	terminated	194
unsubscribe	461	dropped.expired	0		

admin



EMQX

Autenticación con MySQL

Plugin: `emqx_auth_mysql`

IMPORTANTE: Desactivar el ingreso anónimo desde el archivo de configuración ***emqx.conf***.

```
# etc/plugins/emqx_auth_mysql.conf

## server address
auth.mysql.server = 127.0.0.1:3306

## Connection pool size
auth.mysql.pool = 8

auth.mysql.username = emqx

auth.mysql.password = public

auth.mysql.database = mqtt

auth.mysql.query_timeout = 5s
```




EMQX

ACL

Estructura de la tabla que contiene las reglas ACL

```
CREATE TABLE `mqtt_acl` (  
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
  `allow` int(1) DEFAULT 1 COMMENT '0: deny, 1: allow',  
  `ipaddr` varchar(60) DEFAULT NULL COMMENT 'IpAddress',  
  `username` varchar(100) DEFAULT NULL COMMENT 'Username',  
  `clientid` varchar(100) DEFAULT NULL COMMENT 'ClientId',  
  `access` int(2) NOT NULL COMMENT '1: subscribe, 2: publish, 3: pubsub',  
  `topic` varchar(100) NOT NULL DEFAULT '' COMMENT 'Topic Filter',  
  PRIMARY KEY (`id`),  
  INDEX (ipaddr),  
  INDEX (username),  
  INDEX (clientid)  
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



EMQX

ACL

Estructura de la tabla que contiene las reglas ACL

```
CREATE TABLE `mqtt_acl` (  
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,  
  `allow` int(1) DEFAULT 1 COMMENT '0: deny, 1: allow',  
  `ipaddr` varchar(60) DEFAULT NULL COMMENT 'IpAddress',  
  `username` varchar(100) DEFAULT NULL COMMENT 'Username',  
  `clientid` varchar(100) DEFAULT NULL COMMENT 'ClientId',  
  `access` int(2) NOT NULL COMMENT '1: subscribe, 2: publish, 3:  
pubsub',  
  `topic` varchar(100) NOT NULL DEFAULT '' COMMENT 'Topic  
Filter',  
  PRIMARY KEY (`id`),  
  INDEX (ipaddr),  
  INDEX (username),  
  INDEX (clientid)  
  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```



EMQX

Habilitar plugins

Iniciar un plugin a través del dashboard:

EMQX

Monitor
Clients
Topics
Subscriptions
Rule Engine
Analysis
Plugins
Modules
Tools
Alarms
Settings
General

Plugins

emqx@127.0.0.1

Search by plugin name

Name	Description	Status
emqx_auth_http	EMQ X Authentication/ACL with HTTP API	Stopped
emqx_auth_jwt	EMQ X Authentication with JWT	Stopped
emqx_auth_ldap	EMQ X Authentication/ACL with LDAP	Stopped
emqx_auth_mnesia	EMQ X Authentication with Mnesia	Stopped
emqx_auth_mongo	EMQ X Authentication/ACL with MongoDB	Stopped
emqx_auth_mysql	EMQ X Authentication/ACL with MySQL	Running
emqx_auth_pgsql	EMQ X Authentication/ACL with PostgreSQL	Stopped
emqx_auth_redis	EMQ X Authentication/ACL with Redis	Stopped

Inicio automático a través de un archivo de configuración:

```
#data/loaded_plugins  
  
{emqx_management, true}.  
{emqx_recon, true}.  
{emqx_retainer, true}.  
{emqx_dashboard, true}.  
{emqx_rule_engine, true}.  
{emqx_bridge_mqtt, false}.  
{emqx_auth_mysql, true}
```



DOCUMENTACIÓN: <https://nodered.org/docs/>

¿Qué es Node-Red?

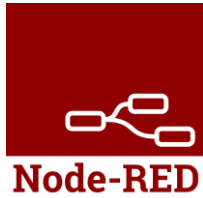
Node-RED es un motor de flujos con enfoque IoT, que permite definir gráficamente flujos de servicios, a través de protocolos estándares como REST, MQTT, Websocket, AMQP... además de ofrecer integración con APIs de terceros, tales como Twitter, Facebook



¿Cuáles son sus ventajas?

Está optimizado para poder tratar múltiples conexiones concurrentes de una manera óptima. Es el mayor ecosistema de código abierto que existe en el mundo y está siendo utilizado por empresas como PayPal y Netflix.





Instalación de Node-RED

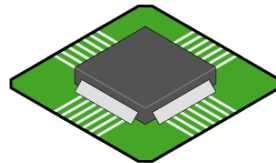
Requisitos:

- NodeJS. (Las versiones recomendadas son NodeJS LTS 8.x y 10.x. Node-RED no soporta las versiones de NodeJS 6.x o anteriores)

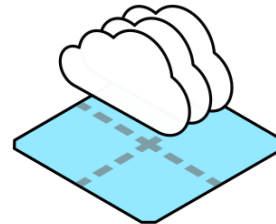
Plataformas de ejecución:



Ejecutar localmente



en un dispositivo



En las nubes



Configuración inicial

Ubicación del archivo de configuración

Para obtener donde se encuentra ubicado el archivo de configuración, podemos ejecutar el comando:

- `node-red --settings`

Este mismo se carga en el tiempo de ejecución como un módulo de Node.js que exporta un objeto JavaScript de pares clave/valor. Y tiene diversas opciones de configuración tanto de distintos protocolos como por ejemplo **HTTP** o como también diversas opciones de seguridad.





Creación de un usuario

Como primer paso debemos crear el hash de la contraseña que utilizaremos para cualquier usuario, para ello corremos el siguiente comando en la consola. (El mismo devolverá una contraseña con el formato mencionado anteriormente)

- `node-red admin hash-pw`

Ejemplo

- `$2a$08$zZWtXTja0fB1pzD4sHCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.`

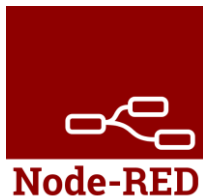
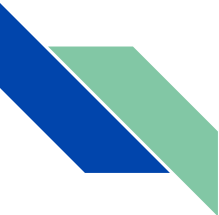


Archivo de configuración ejemplo:

```
adminAuth: {  
  type: "credentials",  
  users: [  
    {  
      username: "admin",  
      password: "*hash*",  
      permissions: "*"   
    },  
    {  
      username: "guest",  
      password: "*hash*",  
      permissions: "read"  
    }  
  ]  
}
```

NOTA: Para solicitar autenticación al entrar al dashboard también debemos editar los campos 'httpNodeAuth' y 'httpStaticAuth'.

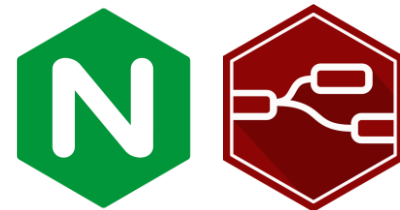


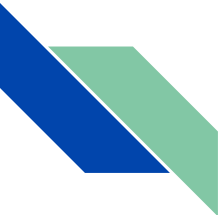


Configuración para el proxy inverso de Nginx

Primero, hay que añadir las siguientes líneas en el archivo *nginx.conf* ubicado en la ruta */etc/nginx/*:

```
http {  
    (...)  
    map $http_upgrade $connection_upgrade {  
        default upgrade;  
        "" close;  
    }  
    upstream websocket {  
        server 127.0.0.1:1880;  
    }  
}
```





Configuración para el proxy inverso de Nginx

El archivo que estará dentro de *sites-available*, por su parte, estará constituido de la siguiente manera



```
server {
    listen 80;
    listen [::]:80;

    server_name nodered.example.com;
    location / {
        proxy_pass
http://127.0.0.1:1880;
        proxy_http_version 1.1;
        proxy_set_header          Upgrade
$http_upgrade;
        proxy_set_header          Connection
$http_connection_upgrade;
        proxy_set_header Host $host;
    }
}
```



Nodos MQTT

Para que un nodo MQTT funcione, ya sea de suscripción o publicación, primero se debe configurar el servidor y tópicos a utilizar. Luego editamos la información del servidor y cargamos los siguientes valores



Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

⚙️ Properties ⚙️ 📄

📁 Name EMQ X

Connection Security Messages

📁 Server localhost Port 1883

Use TLS

⚙️ Protocol MQTT V5 ▾

📁 Client ID Leave blank for auto generated

💓 Keep Alive 60

📄 Session Use clean start

Session Expiry (secs)

User Properties ▾

Arrows point to the Name, Server, Protocol, and Port fields.



Nodos MQTT

En la pestaña de seguridad, ingresamos el usuario y contraseña del usuario que añadimos al broker para autenticarnos.



Edit mqtt in node > Edit mqtt-broker node

Delete Cancel Update

Properties

Name EMQ X

Connection Security Messages

Username admin

Password





Nodos MQTT

Finalmente, seleccionamos el t3pico, el QoS que utilizaremos, y le presionamos el bot3n de **Done**.



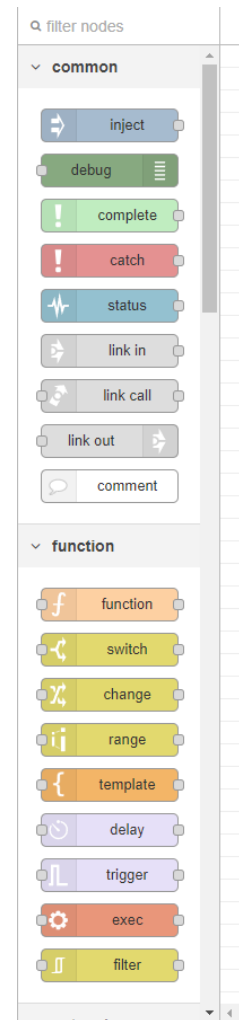
A screenshot of the "Edit mqtt in node" configuration window in Node-RED. The window has a title bar with "Delete", "Cancel", and "Done" buttons. Below the title bar is a "Properties" section with a gear icon, a save icon, and a refresh icon. The configuration fields are: "Server" (EMQ X), "Topic" (v1/devices/cliente1/pot), "QoS" (2), "Output" (auto-detect (string or buffer)), "Flags" (Do not receive messages published by this client, Keep retain flag of original publish), "Retained message handling" (Send retained messages), and "Name" (Name). Three arrows point to the Topic, QoS, and Retained message handling fields, which are also circled in pink, yellow, and brown respectively.

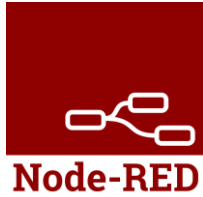


Agregar nodos al campo de trabajo

Nos dirigimos a la sección izquierda de la página donde nos encontramos con las herramientas que Node-red nos ofrece.

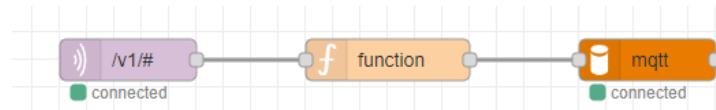
Para añadir las mismas simplemente debemos arrastrarlas hasta el campo de trabajo





Utilización de nodos

Este es un ejemplo de una secuencia de nodos la cual está encargada de escuchar todos los mensajes que llegan del Nodo "MQTT In" (/V1/#) a través del nodo (**function**) donde se ejecuta un código javascript el cual lee los mensajes y los almacena en la base de datos del nodo (**mqtt**).





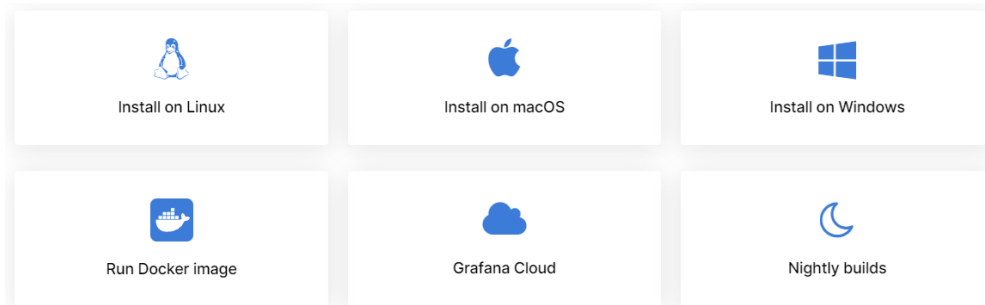
DOCUMENTACIÓN: <https://grafana.com/docs/grafana/latest/>

Instalación de Grafana

Para ejecutar Grafana, debe tener un sistema operativo compatible, hardware que cumpla o supere los requisitos mínimos (255 MB CPU mínima recomendada: 1), una base de datos compatible y un navegador compatible.

Grafana utiliza otro software de código abierto. Consulte [package.json](#) para obtener una lista completa.

Sistemas operativos:





Bases de datos compatibles:

- SQLite
- MySQL
- PostgreSQL



Navegadores web compatibles:

- Chrome/Chromium
- Firefox
- Safari
- Microsoft Edge
- Internet Explorer 11





Configuración de firewall

Para poder acceder al dashboard, nos tenemos que conectar al servidor a través del puerto 3000. Por lo tanto, debemos abrirlo dentro de ufw.

- `sudo ufw allow 3000/tcp`

Luego, comprobamos el estado del firewall para confirmar que este puerto se abrió correctamente.

- `sudo ufw status`
- `sudo ufw status verbose`



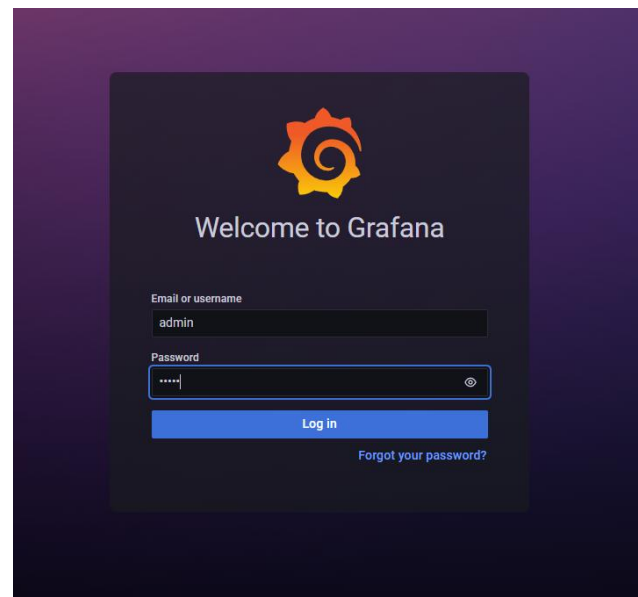


Inicio de sesión

Una vez instalado Grafana, ingresamos a través de un navegador por medio del puerto 3000 (*en caso de que no se cambie el puerto por defecto en la configuración*). P

<http://127.0.0.1:30000/>

Ya cargada la página, cargamos las credenciales por defecto, tanto el **usuario** como la **contraseña** son “*admin*”, seguido de esto Grafana nos solicitara agregar una nueva contraseña.



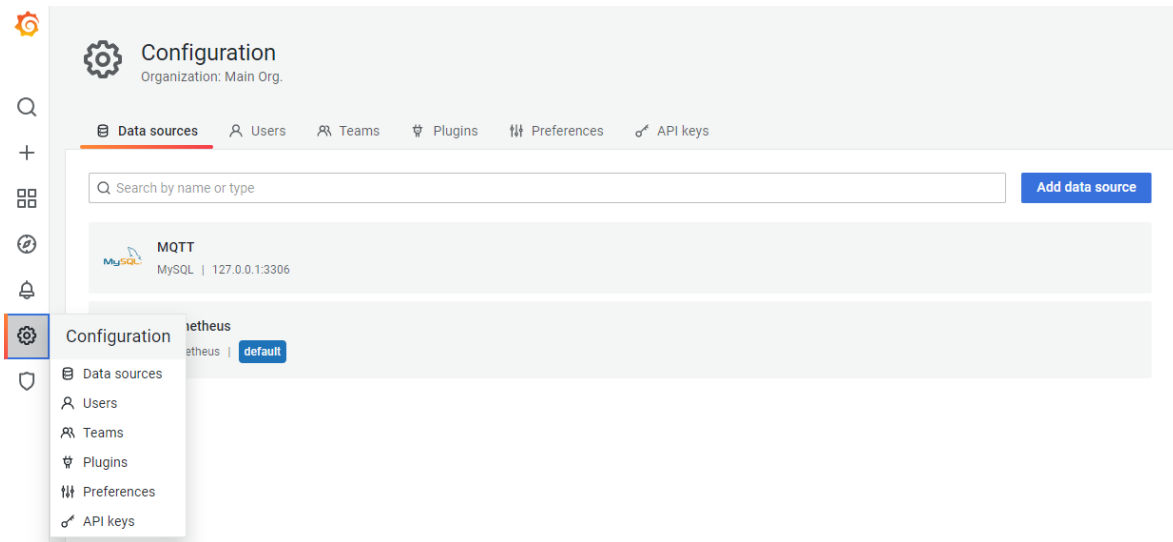


Y



Adición del data source

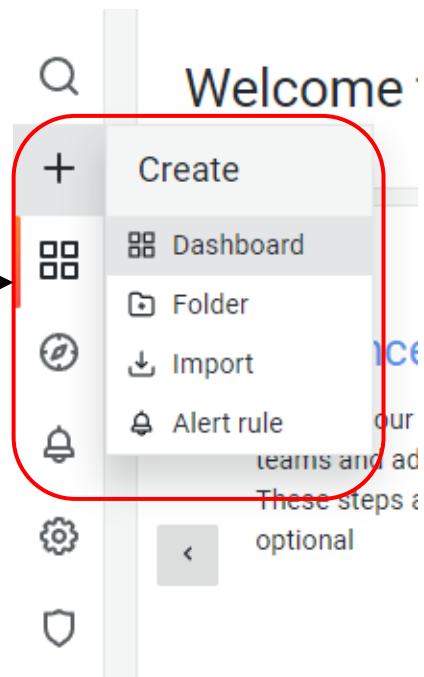
Para añadir el data source nos dirigimos al dashboard de Grafana y en el apartado de **Configuration** > **Data sources** añadimos una nueva base de datos **MySQL**.





Creación de un dashboard

En la parte superior izquierda seleccionamos el botón de "+" → **Dashboard**





Añadir un panel

The screenshot shows the Grafana interface with a sidebar on the left containing icons for search, home, dashboard, refresh, notifications, settings, and help. The main area displays a "New dashboard" header and a "Add panel" dialog box. The dialog box has a title bar with a close button and contains three options: "Add a new panel" (highlighted with a red circle), "Add a new row", and "Add a panel from the panel library".

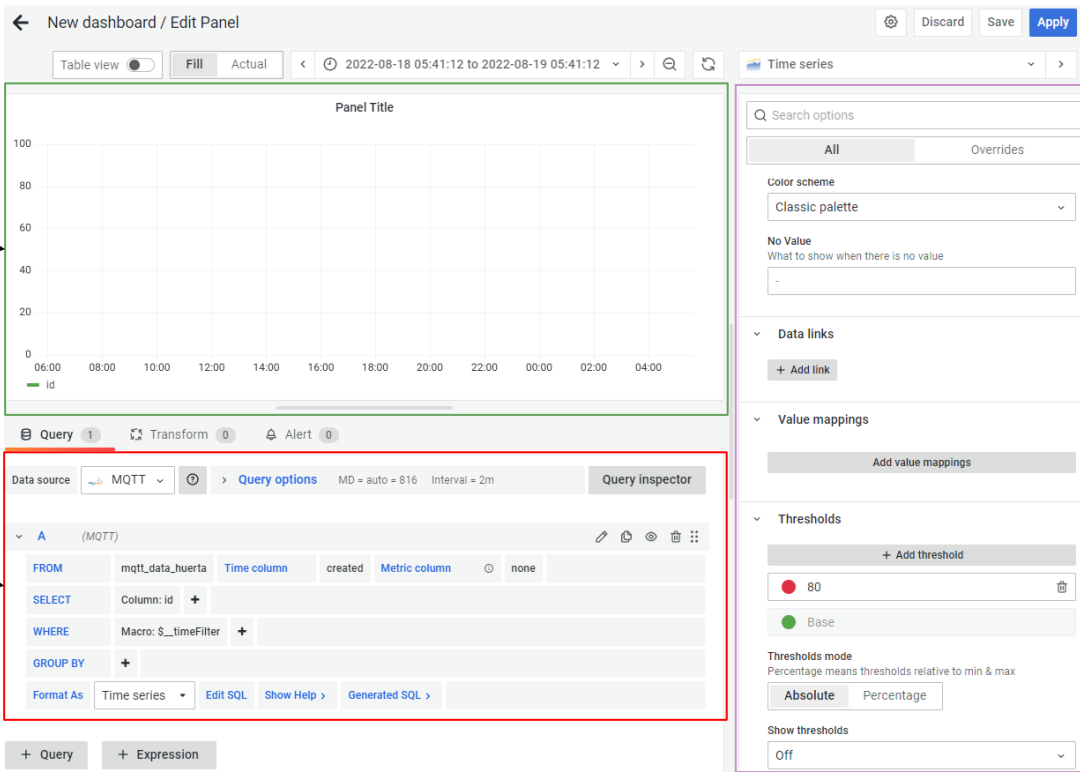
New dashboard

Add panel

- Add a new panel
- Add a new row
- Add a panel from the panel library

Configuración del panel

Previsualización



The screenshot shows the Grafana 'New dashboard / Edit Panel' interface. At the top, there are buttons for 'Discard', 'Save', and 'Apply'. Below this is a 'Table view' toggle and a 'Fill' button. The main area is a time series chart titled 'Panel Title' with a y-axis from 0 to 100 and an x-axis from 06:00 to 04:00. A legend at the bottom left of the chart shows a green line for 'Id'. Below the chart is a 'Query' section with '1' query, 'Transform' '0', and 'Alert' '0'. The data source is 'MQTT' with 'Query options' and 'MD = auto = 816 Interval = 2m'. The query editor shows the following SQL:

```
FROM mqtt_data_huerta Time column created Metric column none
SELECT Column: id +
WHERE Macro: $_timeFilter +
GROUP BY +
Format As Time series Edit SQL Show Help > Generated SQL >
```

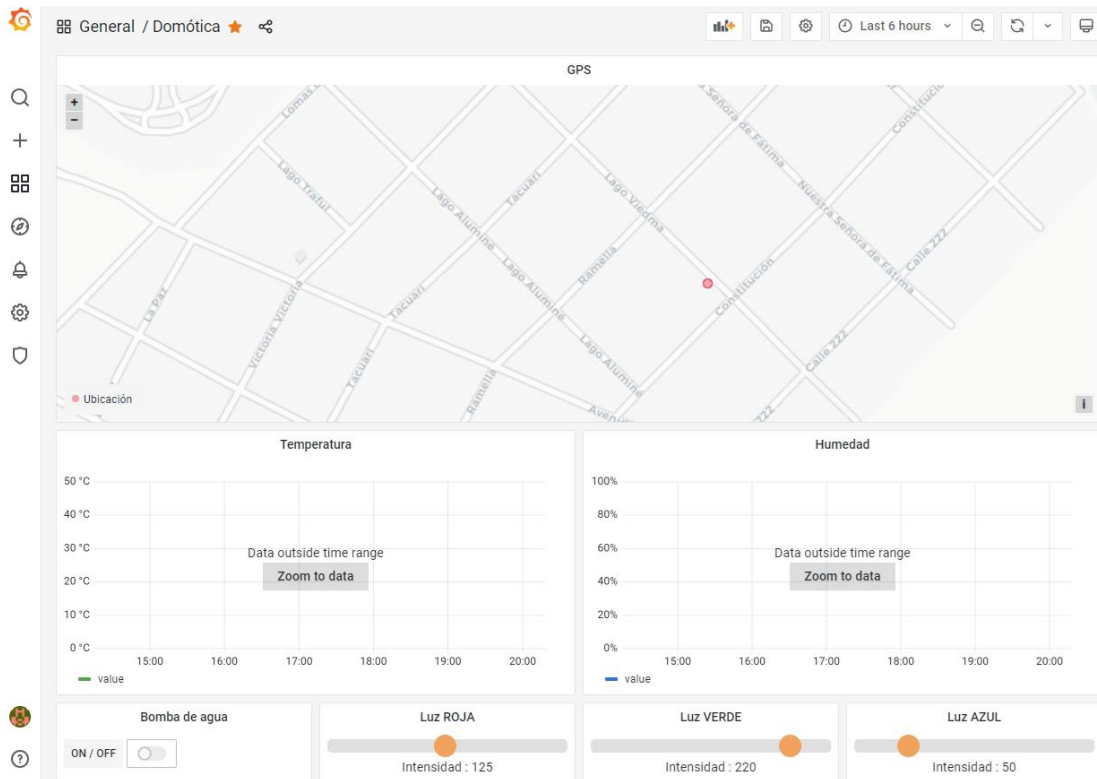
On the right side, there is a configuration panel for 'Time series' with sections for 'Search options', 'Color scheme' (Classic palette), 'No Value' (-), 'Data links' (+ Add link), 'Value mappings' (+ Add value mappings), and 'Thresholds' (+ Add threshold). A threshold is set at 80 with 'Base' mode and 'Absolute' type. The 'Show thresholds' dropdown is set to 'Off'.

Query para obtener los datos que se van a visualizar

Ajustes visuales del panel

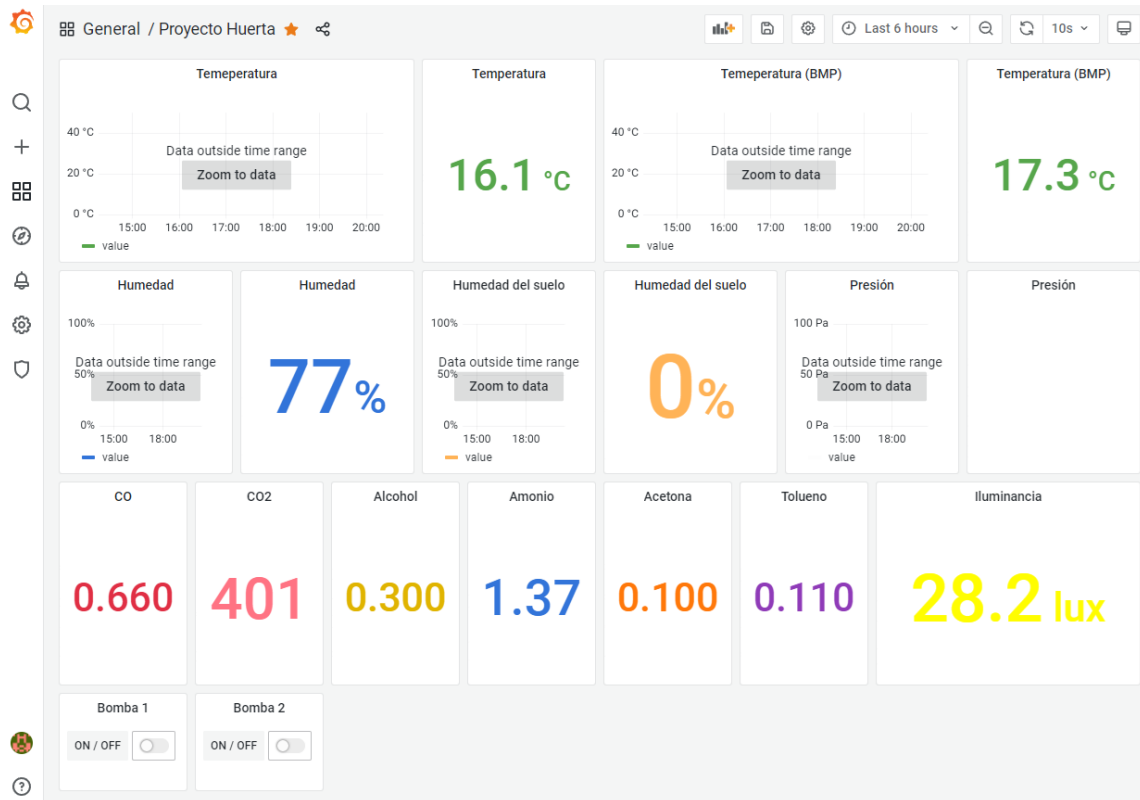


Ejemplo de dashboard para una aplicación de Domótica





Ejemplo de dashboard para un proyecto de huerta



ACCESO A LOS SERVICIOS

Aplicación	Puerto
	<i>Descripción</i>
EMQX	1883
	MQTT/TCP
	8883
	MQTT/SSL
	8083
	MQTT/WS
	8084
	MQTT/WSS
	18083
	Dashboard

Si dispusiéramos del dominio:
compumundoiotred.com

Deberíamos acceder a cada servicio de la siguiente manera:

compumundoiotred.com:1883
compumundoiotred.com:8883
compumundoiotred.com:8083
compumundoiotred.com:8084
compumundoiotred.com:18083
compumundoiotred.com:1880
compumundoiotred.com:3000

NodeRED 1880 Dashboard

No es fácil de recordar, de compartir y de acceder desde un navegador

SUBDOMINIOS

<i>Aplicación</i>	<i>Puerto</i>
	<i>Descripción</i>
EMQX	1883
	MQTT/TCP
	8883
	MQTT/SSL
	8083
MQTT/WS	
8084	
MQTT/WSS	
18083	
Dashboard	

Con el uso de subdominios, podemos acceder a los dashboard de manera mucho más sencilla:

Para EMQX:

`emqx.compumundoiotred.com`

Para NodeRED:

`nodered.compumundoiotred.com`

Para Grafana:

`grafana.compumundoiotred.com`

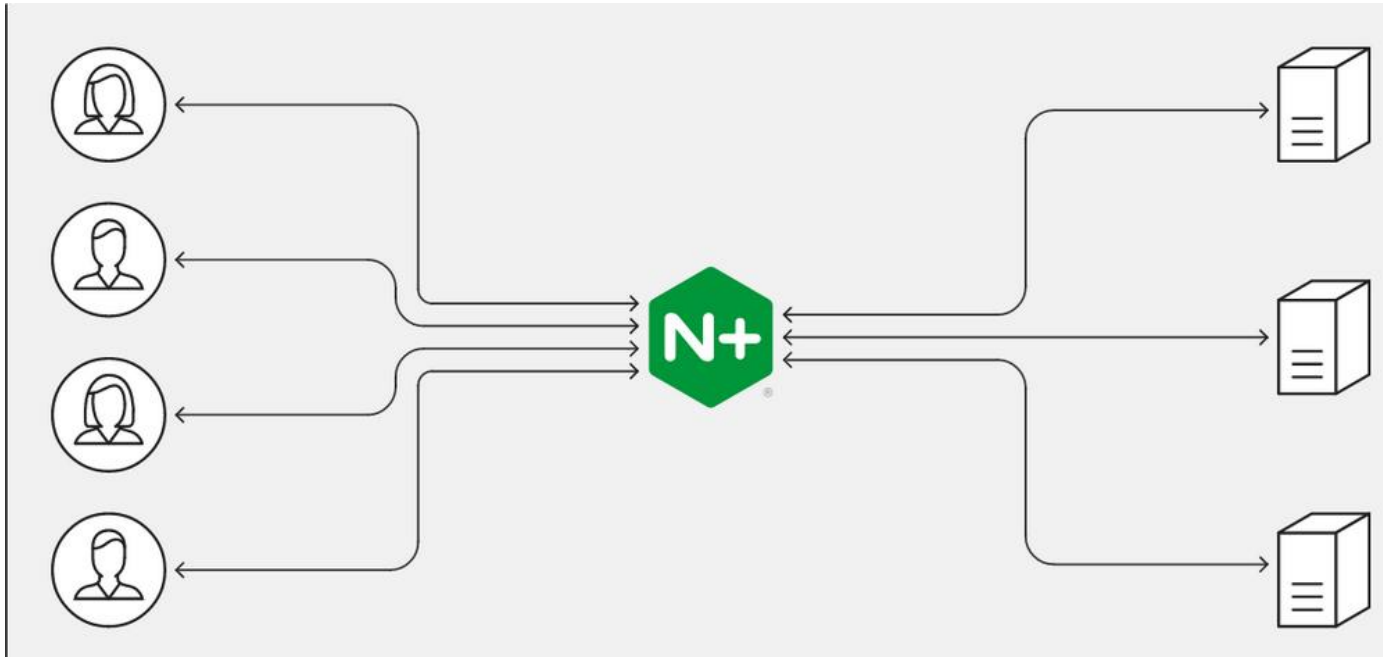
NodeRED

1880

Dashboard

Más fácil de recordar

PROXY INVERSO





Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3).

Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada.

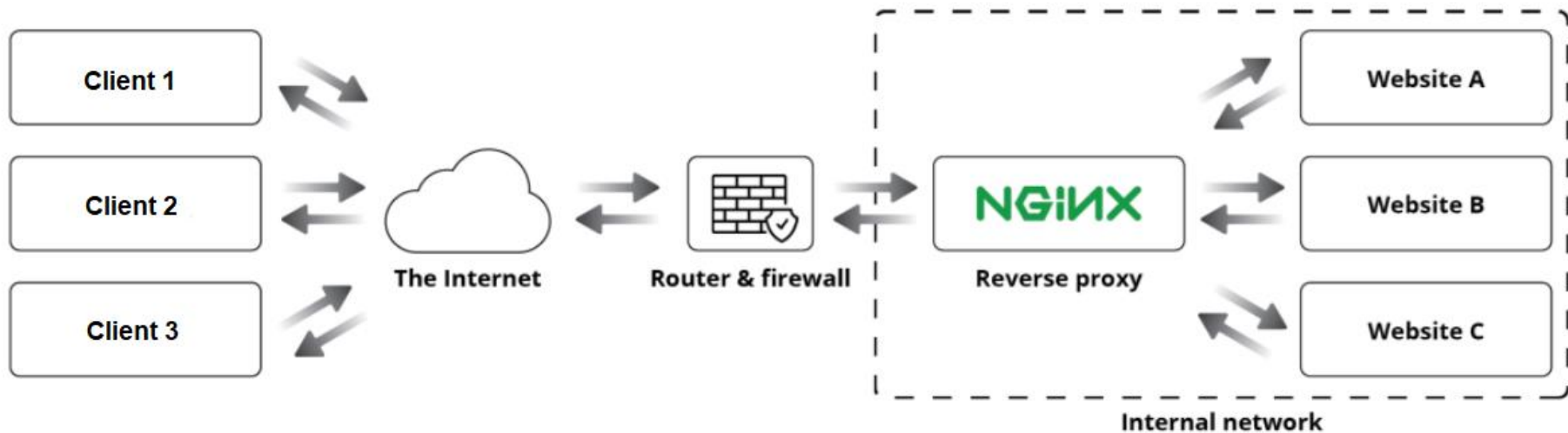
También existe una versión comercial distribuida bajo el nombre de Nginx Plus.





Un servidor *proxy inverso* es un servidor intermediario que se encarga de redirigir requests de múltiples clientes a diferentes servidores por toda la internet. Típicamente se encuentra por detrás de un firewall en una red privada y se conecta al servidor apropiado del backend.







Archivo de configuración de ejemplo para una aplicación que escucha en el puerto 3000:

```
server {  
    listen 80;  
    listen [::]:80;  
  
    server_name example.com www.example.com;  
  
    location / {  
        proxy_pass http://127.0.0.1:3000/;  
    }  
}
```

Todas las direcciones IPv4, en el puerto 80

Todas las direcciones IPv6, en el puerto 80

Nombres de dominio

Redirige la petición a la aplicación que se está ejecutando localmente en el puerto 3000



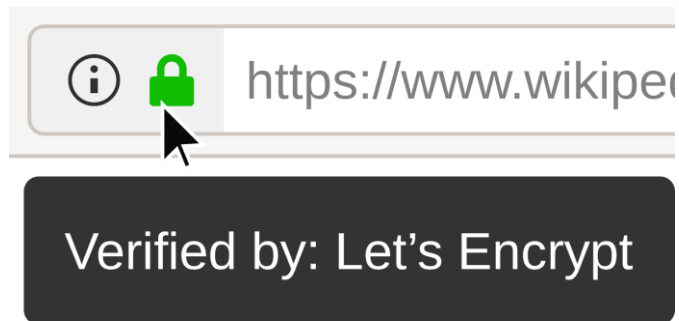
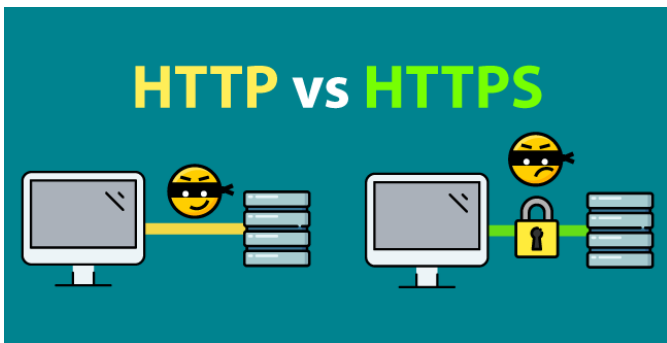


Conexión segura

El propósito de una conexión segura es proteger los datos que se transmiten. Una transmisión no segura puede ser fácilmente interceptada, permitiendo ataques como la interceptación de datos y el phishing.

Los datos enviados a través de una conexión pueden ser **privados** y personales y deben ser **protegidos**.

El uso de un Certificado SSL/TLS nos garantiza que la información que se intercambie entre el cliente y el servidor esté **encriptada**.





Conexión segura

Una de las ventajas del uso de **Certificados SSL/TLS** durante mucho tiempo fue el elevado costo de adquirirlos.

El 12 de abril de 2016, se puso en marcha **Let's Encrypt**, una autoridad de certificación que proporciona certificados gratuitos.

Let's Encrypt proporciona varios beneficios.

- Es **gratuito**.
- Es **automática**.
- Es **simple**.
- Es **seguro**.



Let's Encrypt

Sólo proporciona certificados de validación de dominio (DV).

No son compatibles con certificados de validación organizacional (OV).

Diferencias:

Los **certificados DV** solo pueden garantizar una conexión segura al sitio web.

Los **certificados OV** validan todo lo que hace un (DV), a la vez que validan información organizativa adicional sobre quién compra el certificado, como su nombre, ciudad, estado o país.





Let's Encrypt

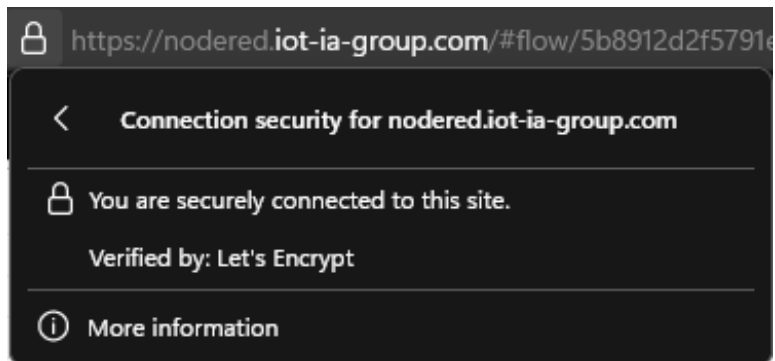
Fácil de instalar en cualquier distribución **Linux**. Se distribuye principalmente como **snap**.

- Para crear el certificado e instalarlo en nuestras aplicaciones de NGINX:

```
certbot --nginx -d example.com -d www.ejemplo.com
```

- Renovar los certificados:

```
certbot renew
```

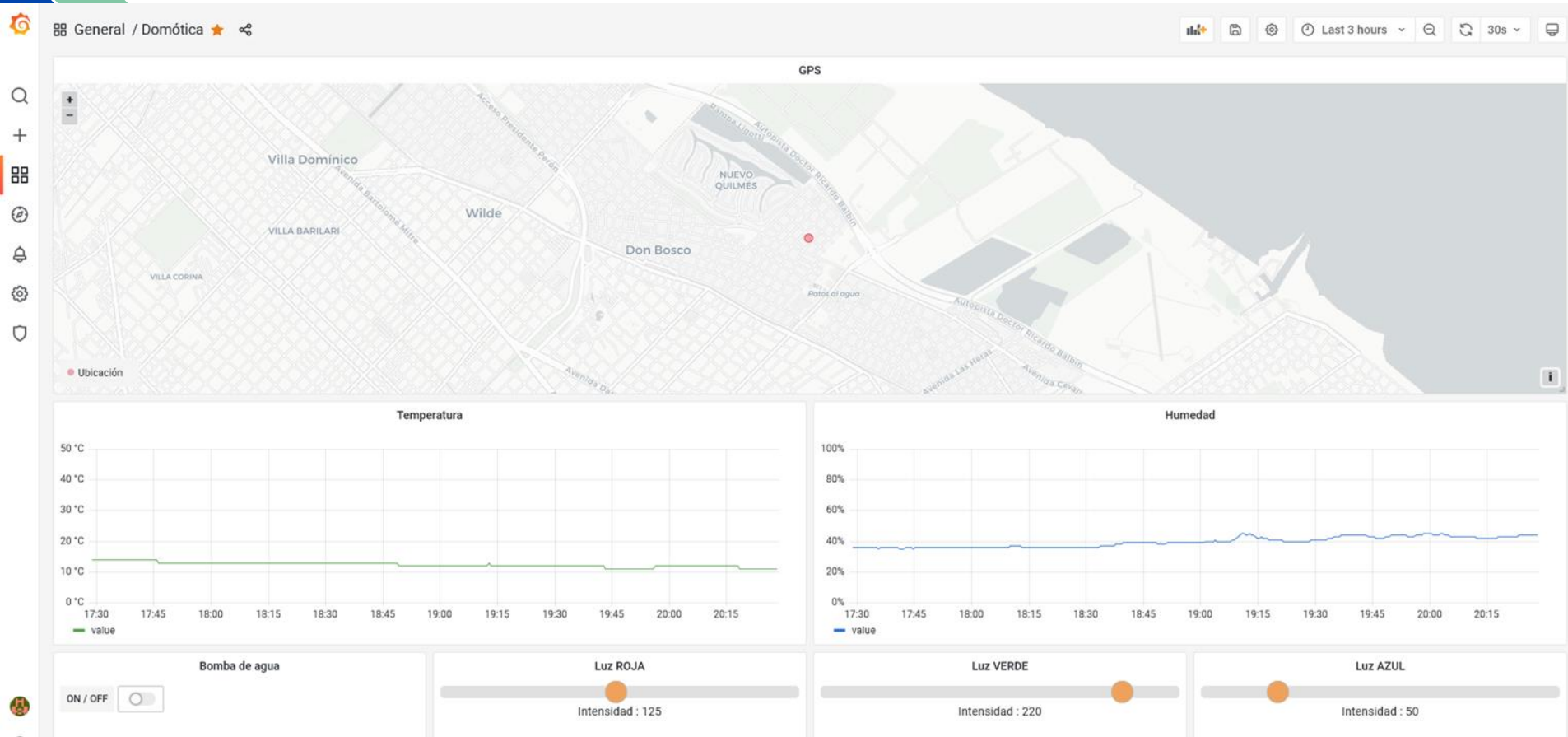




Si queremos agregar una redirección HTTP a HTTPS en algún subdominio o aplicación, debemos agregar el siguiente bloque dentro del archivo de configuración `**.conf*` correspondiente:

```
server {  
    if ($host = ejemplo.com) {  
        return 301 https://$host$request_uri;  
    }  
    listen 80 ;  
    listen [::]:80 ;  
    server_name ejemplo.com;  
    return 404;  
}
```

EJEMPLO DE APLICACIÓN





CONTACTO

MATÍAS GABRIEL BUSUM FRADERA

Universidad Nacional Arturo Jauretche

E-mail: matibf99@gmail.com

FACUNDO ARIEL CHAZARRETA

Universidad Nacional Arturo Jauretche

E-mail: facundochaza@hotmail.com

JUAN EDUARDO SALVATORE

Universidad Nacional Arturo Jauretche

E-mail: juaneduardosalvatore@gmail.com

JORGE OSIO

Universidad Nacional Arturo Jauretche

E-mail: jorgeosio@gmail.com
