

Proyecto de Montaje de Sistema de Monitorización en VMs



INDICE:

INTRODUCCIÓN.....	3
Paso 1º Configuración inicial de la VM Nodo Central (VM-MON).....	5
Paso 2º: Instalación de Docker y despliegue de Prometheus.....	5
Paso 2.1 Crear estructura, ficheros necesarios y prometheus.....	6
Paso 2.3 Despliegue del contenedor grafana.....	7
Paso 2.3.2 Enlazar prometheus con Grafana.....	10
Paso 3º Despliegue de servicio web.....	12
Paso 3.1 Base de datos en el nodo DB.....	13
Paso 3.2 Instalación web del sitio Wordpress.....	14
Paso 4º Instalación de Node Exporter en los nodos.....	17
Paso 4.1 Monitorización con telemetría.....	19
Paso 4.1.1 Monitorización a Wordpress.....	22
Paso 4.1.2 Instalar plugin de Redis Object Cache.....	27
Paso 4.1.3 Configurar la métrica de PromPress.....	29
Paso 4.1.4 Grafana y Prompress.....	31
CONCLUSIÓN.....	34

INTRODUCCIÓN.

Este proyecto, SmartMon, desarrolla una solución de monitorización inteligente para sistemas virtualizados en VMware Workstation. Combina Prometheus y Grafana para la recopilación y visualización de métricas, junto con un sistema básico de inteligencia artificial para detectar anomalías en tiempo real.

El objetivo es crear una plataforma local, sencilla y eficiente, que permita mejorar la administración y supervisión de las máquinas virtuales mediante tecnologías abiertas y automatización.

Máquinas que vamos a usar.

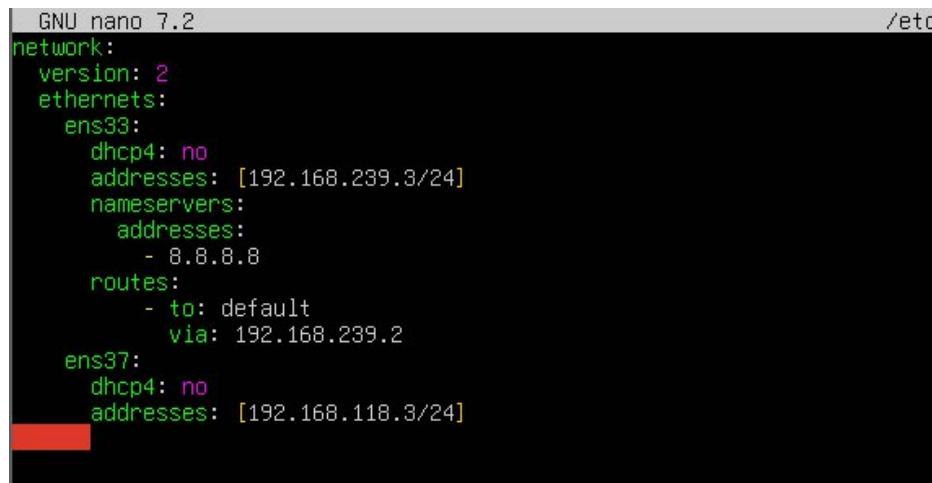
Nombre VM	Rol	IP (NAT)	IP (Host-Only / Custom)	Servicios / Aplicaciones
VM-MON	Nodo central de monitorización	192.168.137.10	192.168.56.10	Prometheus, Grafana, IA, Docker
VM-WEB	Servidor web monitoreado	—	192.168.56.11	Node Exporter, Apache/Nginx
VM-DB	Servidor base de datos	—	192.168.56.12	Node Exporter, MySQL/PostgreSQL

Nota: Este documento asume que las máquinas virtuales ya están creadas y cuentan con sistemas operativos instalados. Por ello, se omiten los pasos relacionados con la creación y configuración inicial de las VMs, enfocándose únicamente en la configuración específica para la monitorización y análisis inteligente del sistema. Está dirigido a usuarios con experiencia previa en administración de sistemas y virtualización.

Paso 1º Configuración inicial de la VM Nodo Central (VM-MON)

En este primer paso, vamos a configurar la la segunda interfaz de nuestro nodo monitorizado.

Nos vamos con sudo nano al fichero de netplan....

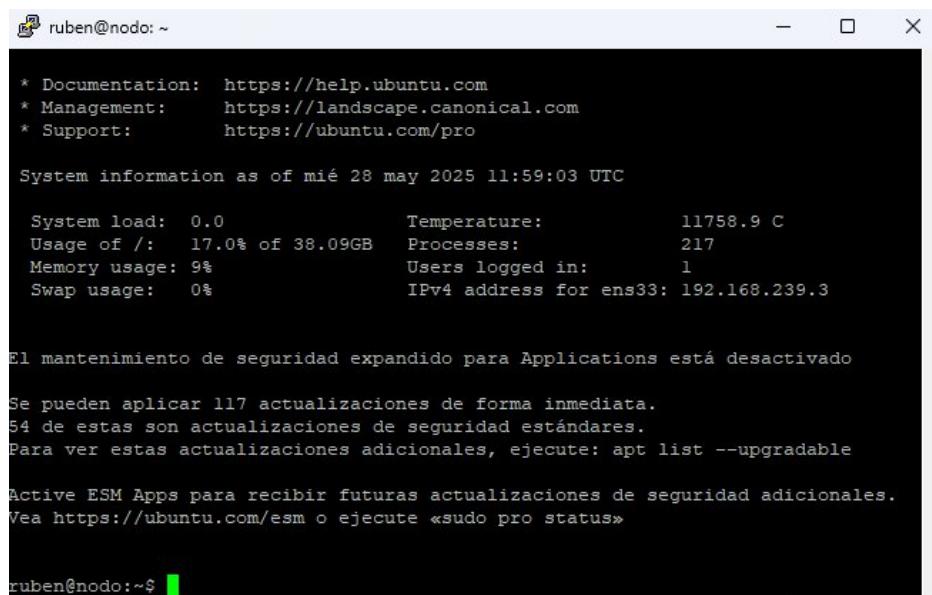


```
GNU nano 7.2 /etc
network:
  version: 2
  ethernets:
    ens33:
      dhcp4: no
      addresses: [192.168.239.3/24]
      nameservers:
        addresses:
          - 8.8.8.8
      routes:
        - to: default
          via: 192.168.239.2
    ens37:
      dhcp4: no
      addresses: [192.168.118.3/24]
```

Y ajustamos según la configuración de vmware.

Paso 2º: Instalación de Docker y despliegue de Prometheus

Para ello, vamos a trabajar con ssh lo primero de todo, yo voy a abrirme una consola con putty (cliente de ssh).



```
ruben@nodo: ~
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro

System information as of mié 28 may 2025 11:59:03 UTC

System load: 0.0 Temperature: 11758.9 C
Usage of /: 17.0% of 38.09GB Processes: 217
Memory usage: 9% Users logged in: 1
Swap usage: 0% IPv4 address for ens33: 192.168.239.3

El mantenimiento de seguridad expandido para Applications está desactivado

Se pueden aplicar 117 actualizaciones de forma inmediata.
54 de estas son actualizaciones de seguridad estándares.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»
```

Actualizamos repositorios....y actualizamos ya de paso el sistema

```
ruben@nodo:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for ruben:
Obj:1 http://archive.ubuntu.com/ubuntu noble InRelease
Obj:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Obj:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Obj:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Des:5 http://archive.ubuntu.com/ubuntu noble/main Translation-es [325 kB]
Des:6 http://archive.ubuntu.com/ubuntu noble/restricted Translation-es [816 B]
Des:7 http://archive.ubuntu.com/ubuntu noble/universe Translation-es [1.371 kB]
Des:8 http://archive.ubuntu.com/ubuntu noble/multiverse Translation-es [63,1 kB]
Descargados 1.759 kB en 1s (3.205 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 64 paquetes. Ejecute «apt list --upgradable» para verlos.
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... 0%
```

Instalamos docker y el compose....

```
ruben@nodo:~$ sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo systemctl start docker
```

Además de activar y arrancar el servicio.

E aquí la versión usada y descargada.

```
ruben@nodo:~$ docker --version
Docker version 26.1.3, build 26.1.3-0ubuntu1~24.04.1
ruben@nodo:~$
```

Paso 2.1 Crear estructura, ficheros necesarios y prometheus.

En este paso, vamos a crear el directorio dónde tendremos los ficheros pertinente para nuestro prometheus.

```
ruben@nodo:~$ mkdir -p ~/smartmon/prometheus
cd ~/smartmon/prometheus
ruben@nodo:~/smartmon/prometheus$
```

Creamos un prometheus.yml qué será el fichero de nodos a manejar.

```
GNU nano 7.2                                     prometheus.yml *
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporters'
    static_configs:
      - targets:
          - '192.168.118.11:9100' # VM-WEB
          - '192.168.118.12:9100' # VM-DB
```

Aquí lo tenemos.

Creamos el docker compose de prometheus.

```
GNU nano 7.2                                            docker-compose.yml *
```

```
version: '3'

services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"
```

Uso sudo, porque el usuario nuestro no está en el grupo de docker, eso tiene facil solución.

```
ruben@nodo:~/smartmon/prometheus$ sudo docker-compose up -d
Creating network "prometheus_default" with the default driver
Pulling prometheus (prom/prometheus)...
latest: Pulling from prom/prometheus
9fa9226be034: Pull complete
1617e25568b2: Pull complete
01e225ff2cae: Extracting [>          ] 557.1kB/62.07MB
6e1c59ce43d3: Downloading [=====] 17.71MB/56.5MB
303f67c648a5: Download complete
d2597eeb55f4: Waiting
744c8ca72725: Waiting
e52136ad7a1b: Waiting
44750a25eb21: Waiting
b6b811691043: Waiting
```

Accedemos desde nuestro host...



Ya lo tenemos desplegado...

Continuaremos por grafana.

Paso 2.2 Despliegue del contenedor grafana.

Ahora vamos a desplegar Grafana, qué nos ayudará a la monitorización, lo haremos igual, con un contenedor de docker.

```
GNU nano 7.2                               docker-compose.yml *
version: '3'

services:
  prometheus:
    image: prom/prometheus
    container_name: prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"
  grafana:
    image: grafana/grafana
    container_name: grafana
    ports:
      - "3000:3000"
    volumes:
      - grafana_data:/var/lib/grafana
    depends_on:
      - prometheus

volumes:
  grafana_data:
```

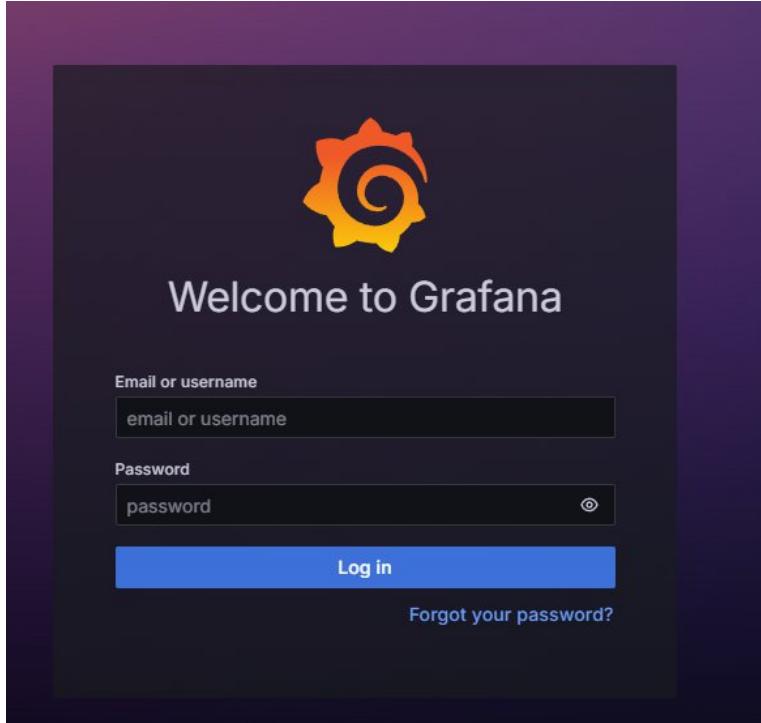
Lo más recomendable es por separado, pero...bueno, yo lo junto en el propio del de prometheus.

```
ruben@nodo:~/smartmon/prometheus$ nano docker-compose.yml
ruben@nodo:~/smartmon/prometheus$ sudo docker-compose up -d
Creating volume "prometheus_grafana_data" with default driver
Pulling grafana (grafana/grafana)...
latest: Pulling from grafana/grafana
f18232174bc9: Pull complete
65babbe3dfe5: Pull complete
65lb0ba49b07: Pull complete
d953cde4314b: Pull complete
aecd4cb03450: Pull complete
13fa68ca8757: Pull complete
f836d47fdc4d: Downloading [=====] 76.65MB/107.3MB
8b5292c940e1: Downloading [=====] 18.22MB/63.48MB
454a4350d439: Download complete
9a8c18aee5ea: Waiting
```

Levantamos...

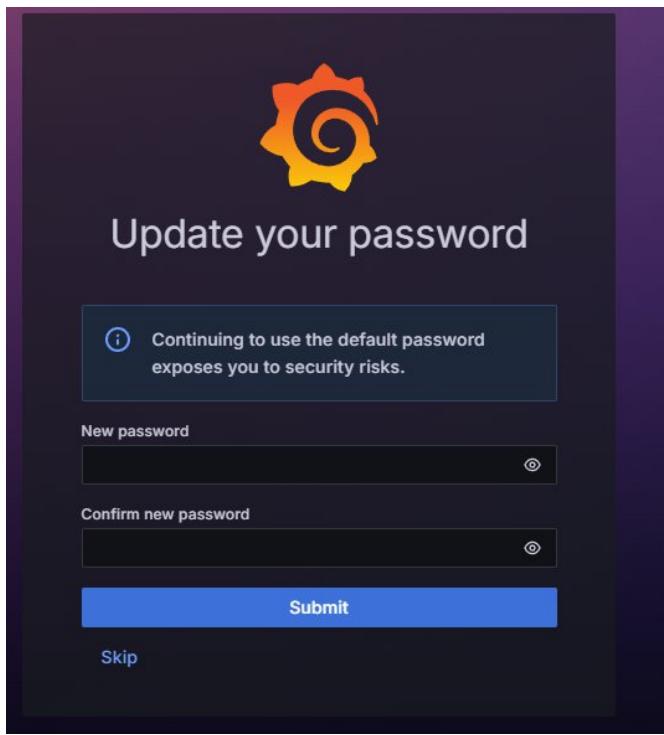
Nos vamos nuevamente a nuestro navegador...





Aquí tenemos el login, ahora...deberemos de acceder.

Por defecto, pondremos admin y admin y luego ya una nueva contraseña.



Una vez dentro, nos aparece ya el cuadro de mandos...

The screenshot shows the Grafana interface. On the left is a dark sidebar with a gear icon and the word 'Grafana'. Below it are links: Home, Bookmarks, Starred, Dashboards (with a 'Drilldown' option), Explore, Alerting, Connections, and Administration. The main area has a title 'Welcome to Grafana'. It includes a 'Basic' section with a 'Tutorial' link to 'DATA SOURCE AND DASHBOARDS' and 'Grafana fundamentals'. There's also a 'DATA SOURCES' section with a 'DASHBOARDS' link. A 'Remove this panel' button is in the top right. At the bottom, there's a sidebar for 'Dashboards' with 'Starred dashboards' and 'Recently viewed dashboards' sections. To the right, there's a 'Latest from the blog' section with two posts: one from May 27 about Prometheus alerts and another from May 23 about Grafana security releases.

Paso 2.3 Enlazar prometheus con Grafana.

Vamos a hacer qué Prometheus sea la fuente de datos de Grafana.

Para ello, nos dirigimos a Grafana, settings y Data Sources.

The screenshot shows the 'Data sources' page in Grafana. The top navigation bar includes a gear icon, 'Home', 'Connections', 'Data sources', and a search bar. The main content area has a heading 'Data sources' and a sub-instruction 'View and manage your connected data source connections'. In the center is a cartoon illustration of a yellow sun-like character with a purple visor, standing next to a cactus and some purple rocks. Below the illustration, the text 'No data sources defined' is displayed, followed by a note: 'You can also define data sources through configuration files. [Learn more](#)'. At the bottom is a large blue button with the text 'Add data source' and an icon of a gear and plus sign.

Seleccionamos Prometheus cuando demos en add data source.

Type: Prometheus

Alerting Supported

Explore data Build a dashboard

Settings Dashboards

Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

Name: prometheus Default:

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, [view the documentation](#).

Fields marked with * are required

Connection

Prometheus server URL *: http://prometheus:9090

En connections, ponemos la dirección del contenedor docker, qué es la que se ve en captura.

Comprobamos la conexión en save & test.

✓ Successfully queried the Prometheus API.

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).

Delete Save & test

Paso 3º Despliegue de servicio web.

Levantaremos un wordpress en docker en el nodo web, y en el nodo db, la base de datos.

```
server@nodol:~$ sudo apt update  
sudo apt install -y docker.io docker-compose  
sudo systemctl enable docker  
sudo systemctl start docker
```

Vamos a montar todos los servicios en contenedores, ya que es más rápido.

En el nodo web, instalaremos docker, cómo bien se ve....y activamos el servicio.

Creamos un directorio...

```
server@nodol:~/wordpress$ nano docker-compose.yml  
server@nodol:~/wordpress$ D
```

Y luego el compose.yml.

```
GNU nano 7.2                               docker-compose.yml  
version: '3.1'  
  
services:  
  wordpress:  
    image: wordpress:latest  
    container_name: wordpress  
    ports:  
      - "80:80"  
    environment:  
      WORDPRESS_DB_HOST: 192.168.118.128:3306  
      WORDPRESS_DB_USER: wpuser  
      WORDPRESS_DB_PASSWORD: wppass  
      WORDPRESS_DB_NAME: wpdb  
    volumes:  
      - wordpress_data:/var/www/html  
  
volumes:  
  wordpress_data:
```

Levantamos el servicio...

```
server@nodol:~/wordpress$ sudo docker-compose up -d  
Creating network "wordpress_default" with the default driver  
Creating volume "wordpress_wordpress_data" with default driver  
Pulling wordpress (wordpress:latest)...  
latest: Pulling from library/wordpress  
61320b0lae5e: Pull complete  
b4ce612dc732: Pull complete  
093338982d92: Pull complete  
0c2a0ba9eb0c: Pull complete  
442abaed7751: Pull complete  
aa4e51934eeef: Pull complete  
924949db942b: Pull complete  
153d2fb08c64: Pull complete  
26f17ce45149: Pull complete  
64a857ca8a6a: Extracting [=====]>          ]  9.306MB/11.42MB  
  6.892MB/11.42MB download complete  
d4dfcf68eba: Download complete  
91e76e37da73: Download complete  
4f4fb700ef54: Download complete  
79eaebcf6f91: Download complete  
70a6388697d7: Download complete  
d33dc2ec72e5: Download complete  
116f8f9c524a: Download complete  
8220bc2ecbd0: Download complete  
2cc26c3ac19c: Download complete  
f06097973e73: Download complete  
369bf408b523: Download complete
```

Y una vez terminado...lo tendremos listo...

```
Digest: sha256:56c5be977240677109850cc  
Status: Downloaded newer image for wordpress:  
Creating wordpress ... done  
server@nod01:~/wordpress$
```

Paso 3.1 Base de datos en el nodo DB.

Vamos a hacer lo mismo, pero levantando un contenedor de mariadb.

```
db@db:~$ sudo apt update  
sudo apt install -y docker.io docker-compose  
sudo systemctl enable docker  
sudo systemctl start docker  
[sudo] password for db:  
0% [Esperando las cabeceras] [Conectando a security.ubuntu.com (2620:
```

Creamos un directorio, y dentro un docker-compose como el siguiente...

```
GNU nano 7.2                               docker-compose.yml *  
version: '3.1'  
  
services:  
  mysql:  
    image: mysql:5.7  
    container_name: mysql  
    restart: always  
    environment:  
      MYSQL_ROOT_PASSWORD: rootpass  
      MYSQL_DATABASE: wpdb  
      MYSQL_USER: wpuser  
      MYSQL_PASSWORD: wppass  
    ports:  
      - "3306:3306"  
    volumes:  
      - db_data:/var/lib/mysql  
  
volumes:  
  db_data:
```

```
db@db:~/mysql$ sudo docker-compose up -d  
Creating network "mysql_default" with the default driver  
Creating volume "mysql_db_data" with default driver  
Pulling mysql (mysql:5.7)...  
5.7: Pulling from library/mysql  
20e4dcae4c69: Extracting [=====>  
10.49MB/50.5MBDownload complete  
  869B/869B Download complete  
68c3898c2015: Downloading [====>  
  396.5kB/4.578MBDownload complete  
  3.08kB/3.08kBWaiting  
ae71319cb779: Waiting  
ffc89e9dfd88: Waiting  
43d05e938198: Waiting  
064b2d298fba: Waiting  
df9a4d85569b: Waiting
```

Ahora, ras levantarse, deberemos de entrar dentro del contenedor, y modificar unas cosas...

```
sudo snap install mysql-client
db@db:~/mysql$ sudo docker exec -it mysql mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.44 MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> [REDACTED]
```

Modificamos los privilegios...

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'wpuser'@'%' IDENTIFIED BY 'wppass';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> [REDACTED]
```

Ahora nos vamos al server web, y modificamos el archivo compose.

```
GNU nano 7.2                               docker-compose.yml
version: '3.1'

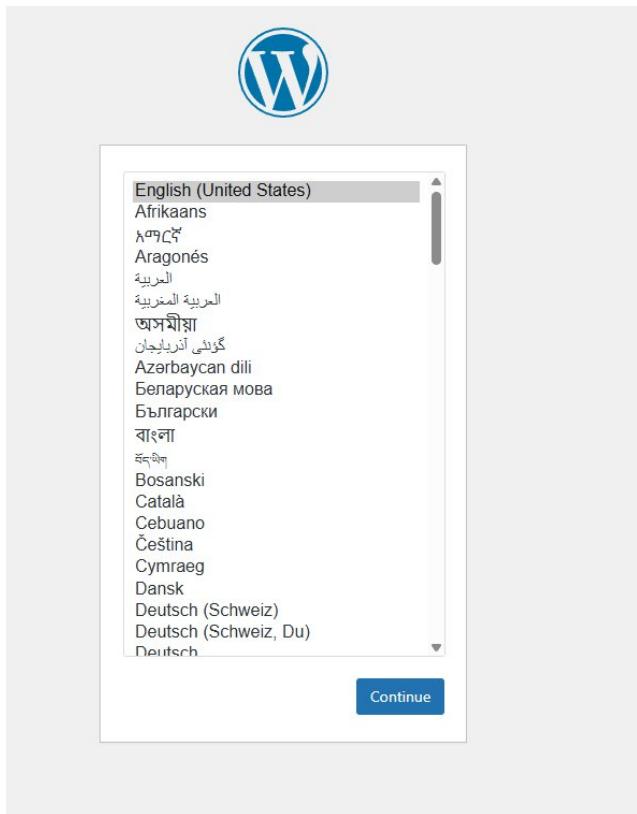
services:
  wordpress:
    image: wordpress:latest
    container_name: wordpress
    ports:
      - "80:80"
    environment:
      WORDPRESS_DB_HOST: 192.168.239.182:3306
      WORDPRESS_DB_USER: wpuser
      WORDPRESS_DB_PASSWORD: wppass
      WORDPRESS_DB_NAME: wpdb
    volumes:
      - wordpress_data:/var/www/html

volumes:
  wordpress_data:
```

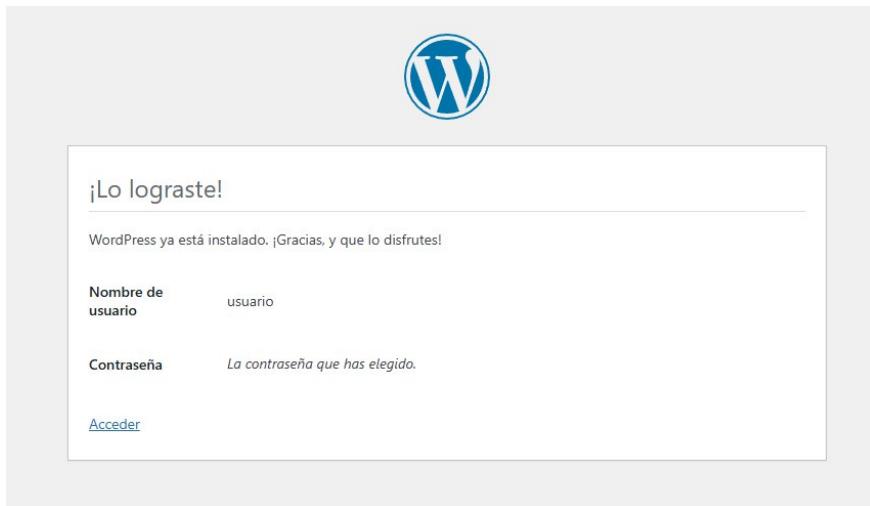
Eliminamos el anterior contenedor de docker y lo volvemos a levantar.

Paso 3.2 Instalación web del sitio Wordpress

Nos vamos al navegador....

A screenshot of the WordPress setup wizard. The title "Hola" is at the top. Below it, a message says: "[Este es el famoso proceso de instalación de WordPress en cinco minutos! Simplemente completa la información siguiente y estarás a punto de usar la más enriquecedora y potente plataforma de publicación personal del mundo.]". The main section is titled "Información necesaria". It contains fields for "Nombre de sitio" (misitio.org), "Nombre de usuario" (usuario), and "Contraseña" (a masked password). A note below the password field says: "Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @.". A "Mostrar" (Show) button is next to the password field. A note below the password field says: "Importante: Necesitas esta contraseña para acceder. Por favor, guárda en un lugar seguro.". Below these are fields for "Tu correo electrónico" (jgjgjgjgj@gmail.com) and "Visibilidad en los motores de búsqueda" (checkbox options: "Pedir a los motores de búsqueda que no indexen este sitio" and "Depende de los motores de búsqueda atender esta petición o no"). At the bottom is a "Instalar WordPress" (Install WordPress) button.

Y ya lo tenemos casi instalado...le damos en instalar.



Y si accedemos...

A screenshot of the WordPress dashboard. The top navigation bar includes the 'misitio.org' site name, a house icon, a refresh icon, a '2' notifications icon, a comment icon, an 'Añadir' (Add) button, and user info 'Hola, usuario'. On the right are 'Opciones de pantalla' (Screen Options) and 'Ayuda' (Help) dropdowns. The main title 'Escritorio' (Dashboard) is visible. The central area features a large dark box with the text '¡Te damos la bienvenida a WordPress!' (Welcome to WordPress!) and 'Aprende más sobre la versión 6.8.1.' (Learn more about version 6.8.1.). Below this are three columns: 'Crea contenido rico con bloques y patrones' (Create rich content with blocks and patterns), 'Personaliza todo tu sitio con temas de bloques' (Customize your site with block themes), and 'Cambia la apariencia de tu sitio con los estilos' (Change your site's appearance with styles). Each column has descriptive text and links like 'Abrir el editor del sitio' (Open site editor) and 'Editar los estilos' (Edit styles). At the bottom are two search/filter boxes: 'Estado de salud del sitio' (Site health status) and 'Borrador rápido' (Quick draft).

Paso 4º Instalación de Node Exporter en los nodos.

Vamos a instalar Node exporter en ambos nodos, para así poder monitorizarlos.

En el web

```
server@nodol:~/wordpress$ cd /opt
sudo wget https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz
sudo tar -xzf node_exporter-1.8.1.linux-amd64.tar.gz
sudo mv node_exporter-1.8.1.linux-amd64 node_exporter
```

Lo tenemos ya descargado y descomprimido.

```
node_exporter-1.8.1.linux-amd 100%[=====] 10,18M 53,9MB/s in 0,2s
2025-05-28 14:32:56 (53,9 MB/s) - 'node_exporter-1.8.1.linux-amd64.tar.gz' saved [10672684/10672684]
server@nodol:/opt$ ls
containerd  node_exporter  node_exporter-1.8.1.linux-amd64.tar.gz
server@nodol:/opt$
```

Creamos usuario sin acceso a login..

```
server@nodol:/opt$ sudo useradd --no-create-home --shell /usr/sbin/nologin node_exporter
```

Luego, debemos de crear un fichero de archivo de servicio.

```
GNU nano 7.2                                     /etc/systemd/system/node_exporter.service *
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
ExecStart=/opt/node_exporter/node_exporter

[Install]
WantedBy=multi-user.target
```

Activamos y arrancamos el servicio.

```
server@nodol:/opt$ sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
server@nodol:/opt$
```

Actualizamos el fichero de prometheus de los targets...

```
GNU nano 7.2                                     prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporters'
    static_configs:
      - targets:
          - '192.168.239.181:9100' # VM-WEB
          - '192.168.239.182:9100' # VM-DB
```

Y reiniciamos...

```
ruben@nodo:~/smartmon/prometheus$ sudo docker restart prometheus
[sudo] password for ruben:
prometheus
rulen@nodo:~/smartmon/prometheus$
```

Importante, en mi caso no funcionaba con la otra vm, me explico, al no tener instalado el node, era obvio qué no funcionaría, pero el otro nodo, qué es el que me he concentrado....al reiniciar el nodo....

Endpoint	Labels	Last scrape	State
http://192.168.239.181:9100/metrics	instance="192.168.239.181:9100", job="node_exporters"	9.487s ago	11ms UP

Ya lo tenemos aquí.

Hacemos lo mismo en el nodo db.

```
cd /tmp
```

```
# 2. Descargar Node Exporter
```

```
wget https://github.com/prometheus/node_exporter/releases/latest/download/node_exporter-1.9.1.linux-amd64.tar.gz
```

```
# 3. Descomprimir el archivo descargado
```

```
tar xvf node_exporter-1.9.1.linux-amd64.tar.gz
```

```
# 4. Entrar en la carpeta descomprimida
```

```
cd node_exporter-1.9.1.linux-amd64
```

```
# 5. Ejecutar Node Exporter en segundo plano
```

```
./node_exporter &
```

```
# 6. Mostrar el proceso para confirmar que está corriendo
```

```
ps aux | grep node_exporter
```

```
GNU nano 7.2                               prometheus.yml *
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporters'
    static_configs:
      - targets:
          - '192.168.239.181:9100' # VM-WEB
          - '192.168.239.182:9100'
```

Agregamos el target DB.

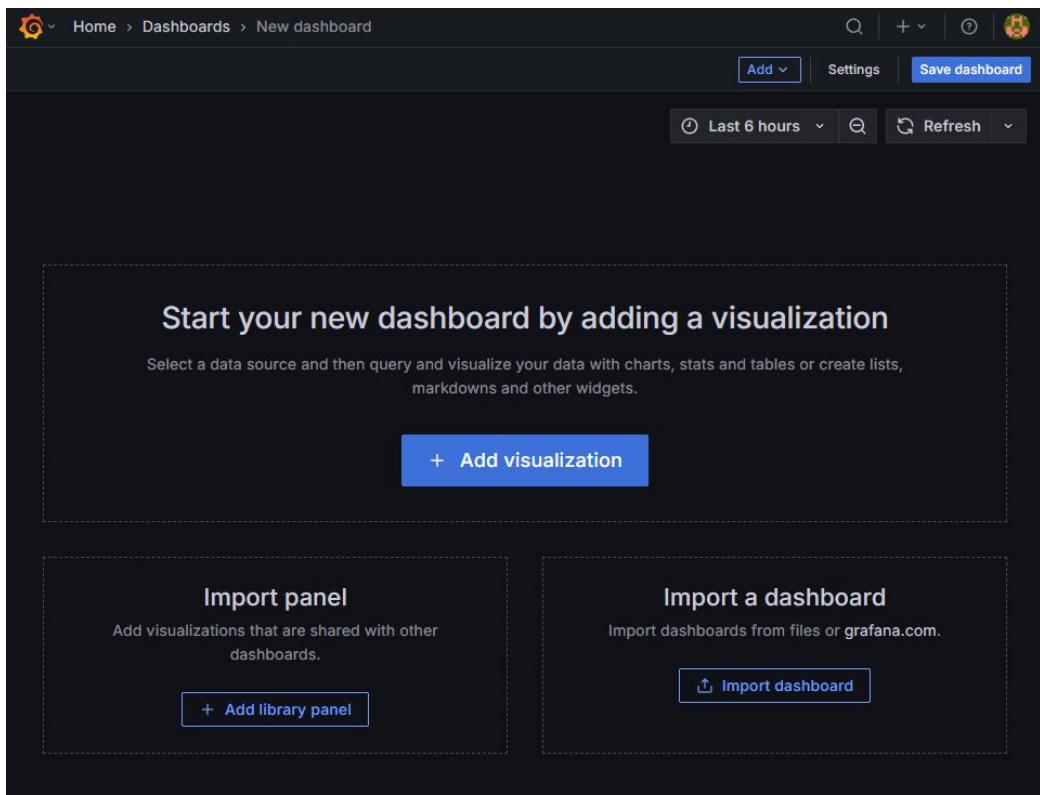
Y si vamos a targets...

Endpoint	Labels	Last scrape	State
http://192.168.239.181:9100/metrics	instance="192.168.239.181:9100" job="node_exporters"	1.907s ago 12ms	UP
http://192.168.239.182:9100/metrics	instance="192.168.239.182:9100" job="node_exporters"	13.14s ago 15ms	UP

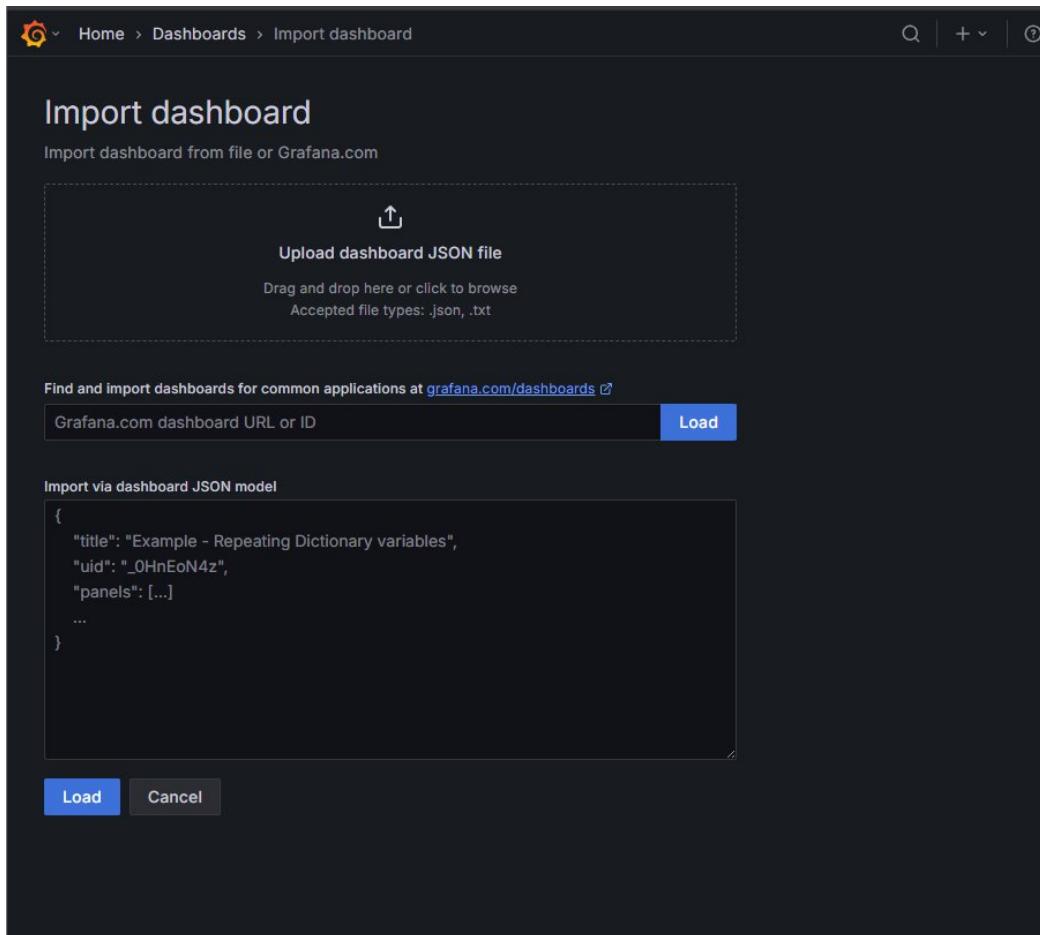
Los tenemos monitorizados, ahora...podemos ir a grafana y tener los gráficos.

Paso 4.1 Monitorización con telemetría.

Para ello, nos vamos a Grafana....



Le damos en Import a dashboard.



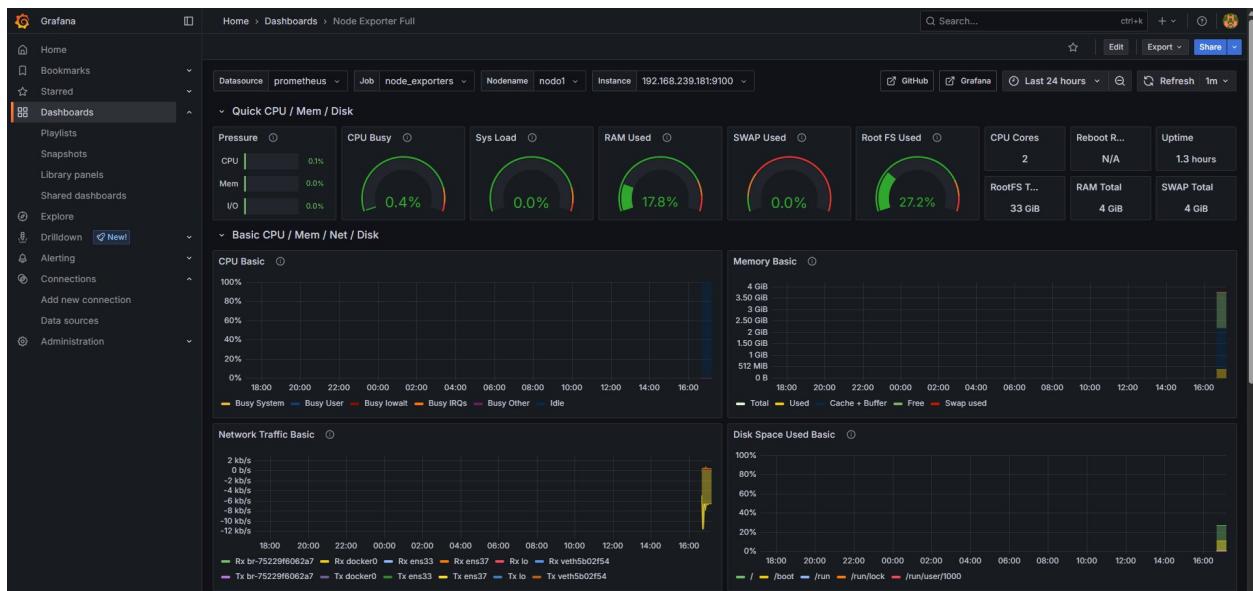
Ponemos en el buscador de ID...lo siguiente 1860

The screenshot shows the 'Import dashboard' screen in Grafana. At the top, it says 'Import dashboard' and 'Import dashboard from file or Grafana.com'. Below that, it says 'Importing dashboard from Grafana.com'. It shows details: 'Published by rfmoz' and 'Updated on 2025-04-27 19:06:41'. Under 'Options', there are fields for 'Name' (set to 'Node Exporter Full'), 'Folder' (set to 'Dashboards'), and 'Unique identifier (UID)' (set to 'rYdddIPWk'). A 'Change uid' button is available. Under 'prometheus', it says 'Select a Prometheus data source' and shows a dropdown menu with 'prometheus' selected. At the bottom are 'Import' and 'Cancel' buttons.

Seleccionamos la fuente de datos...la de prometheus.

This screenshot shows the same 'Import dashboard' screen as above, but with a difference in the 'prometheus' section. The dropdown menu now shows 'prometheus' selected, indicated by a blue border around the option and a small orange icon next to it. The other fields ('Name', 'Folder', 'Unique identifier (UID)', and buttons) remain the same as in the first screenshot.

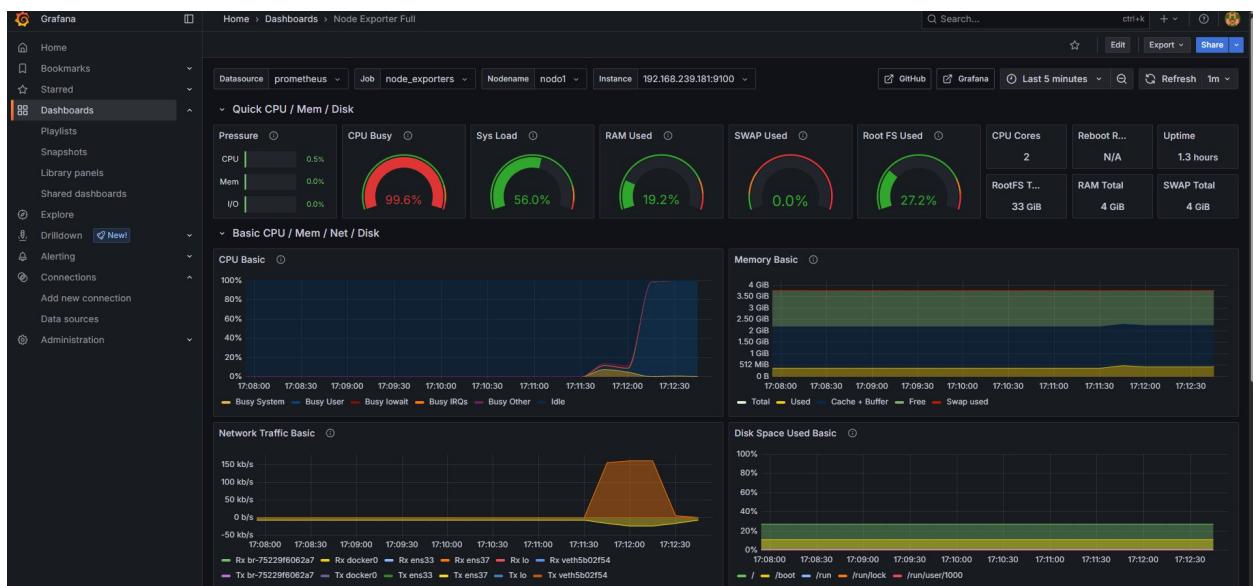
Y....



Ya tenemos la monitorización, en tiempo real.

Para comprobar si funciona, voy a estresar la vm, lo haremos con un comando e instalaremos un pequeño paquete.

`sudo apt update && sudo apt install stress -y`

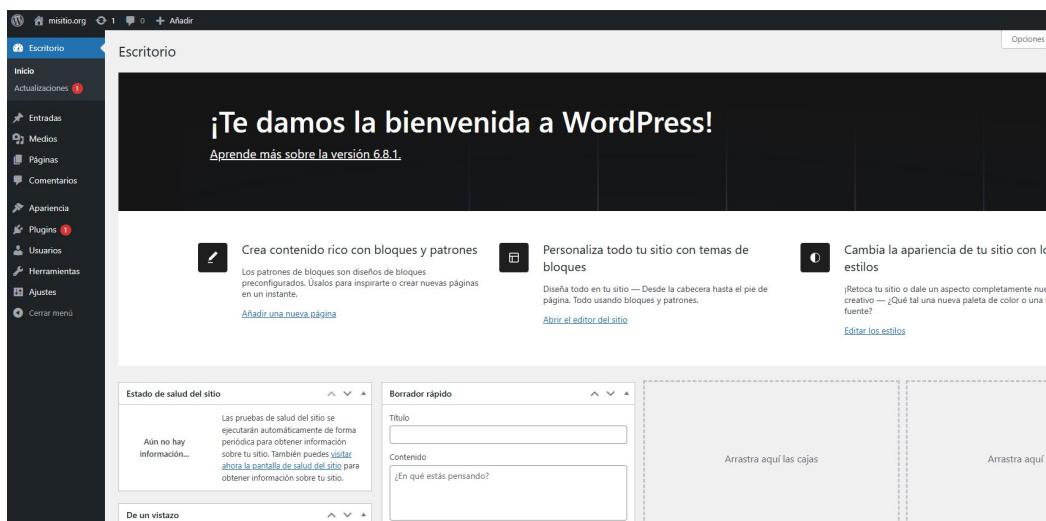


Si filtramos en los últimos 5 minutos...estará a reventar.

Paso 4.1.1 Monitorización a Wordpress

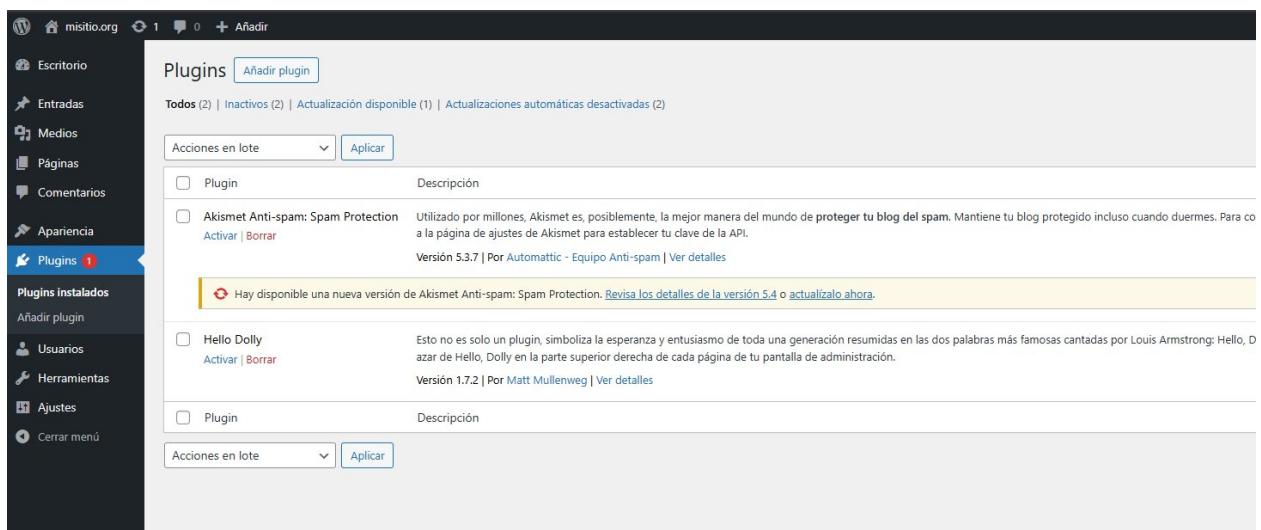
En este apartado, cómo bien hemos montado en un paso anterior, tenemos nuestro gran wordpress, el cuál también podemos monitorizar.

Usaremos un plugin....llamado PromPress



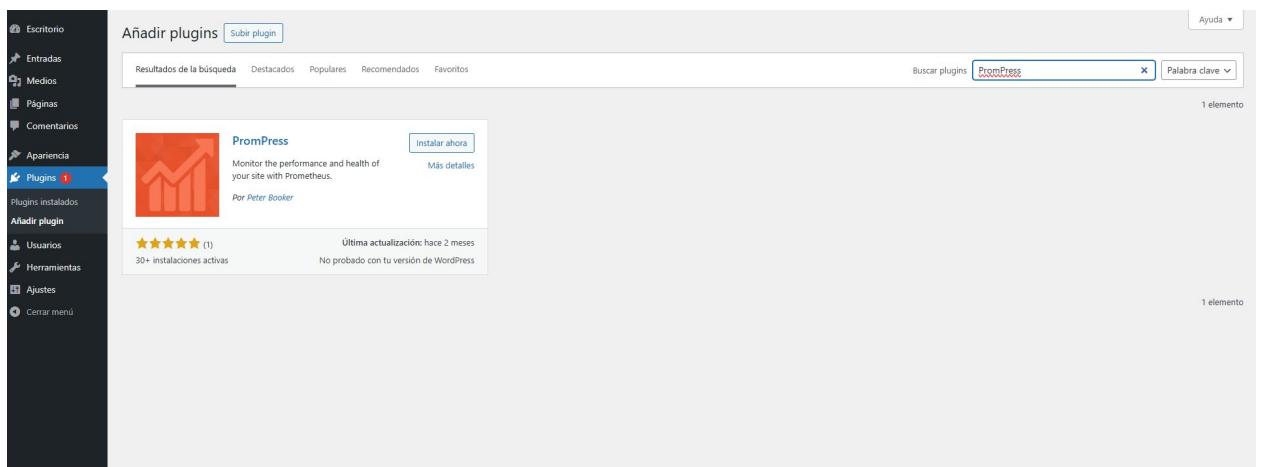
Entramos al back-end....

Vamos a plugins..



Damos en añadir plugin...

Y buscamos el siguiente...



Lo instalamos.

Nos aparecerá lo siguiente...

The screenshot shows the WordPress admin interface under the 'Plugins' tab. A red error message box at the top states: 'PromPress plugin requires the Redis (PECL) PHP extension.' Below it, a green success message box says: 'Plugin activado.' At the bottom, there are links for 'Todos (3)', 'Activo (1)', 'Inactivos (2)', 'Actualización disponible (1)', and 'Actualizaciones automáticas desactivadas (3)'.

Para ello, iremos a nuestro nodo con docker de web.

```
server@nodol:~$ sudo docker exec -it dc bash
root@dc472b99fe94:/var/www/html# apt-get update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8793 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [512 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [261 kB]
93% [4 Packages store 0 B]
```

Actualizamos....

Instalamos el server redis..

```
root@dc472b99fe94:/var/www/html# apt-get install -y redis-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjemalloc2 liblzfl redis-tools
Suggested packages:
  ruby-redis
The following NEW packages will be installed:
  libjemalloc2 liblzfl redis-server redis-tools
0 upgraded, 4 newly installed, 0 to remove and 1 not upgraded.
Need to get 1346 kB of archives.
After this operation, 7095 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 libjemalloc2 amd64 5.3.0-1 [275 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 liblzfl amd64 3.6-3 [10.2 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 redis-tools amd64 5:7.0.15-1~debl2u4 [988 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 redis-server amd64 5:7.0.15-1~debl2u4 [72.7 kB]
Fetched 1346 kB in 1s (2382 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libjemalloc2:amd64.
(Reading database ... 16610 files and directories currently installed.)
Preparing to unpack .../libjemalloc2_5.3.0-1_amd64.deb ...
Unpacking libjemalloc2:amd64 (5.3.0-1) ...
Selecting previously unselected package liblzfl:amd64.
Preparing to unpack .../liblzfl_3.6-3_amd64.deb ...
Unpacking liblzfl:amd64 (3.6-3) ...
Selecting previously unselected package redis-tools.
Preparing to unpack .../redis-tools_5%3a7.0.15-1~debl2u4_amd64.deb ...
Unpacking redis-tools (5:7.0.15-1~debl2u4) ...
Selecting previously unselected package redis-server.
Preparing to unpack .../redis-server_5%3a7.0.15-1~debl2u4_amd64.deb ...
Unpacking redis-server (5:7.0.15-1~debl2u4) ...
Setting up libjemalloc2:amd64 (5.3.0-1) ...
Setting up liblzfl:amd64 (3.6-3) ...
Setting up redis-tools (5:7.0.15-1~debl2u4) ...
Setting up redis-server (5:7.0.15-1~debl2u4) ...
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin (2.36-9+debl2u10) ...
root@dc472b99fe94:/var/www/html#
```

Luego aparecerán unas preguntas, le damos enter a todas.

Responderás con el por defecto, no.

```
root@nodol:~$ sudo docker restart wordpress
wordpress
server@nodol:~$
```

Salimos, y reiniciamos el contenedor.

```
server@nodol:~$ sudo docker restart wordpress
wordpress
server@nodol:~$
```

Y al reinciar....

The screenshot shows the WordPress admin interface under the 'Plugins' tab. There are three items listed:

- Akismet Anti-spam: Spam Protection**: Version 5.3.7 by Automattic. A yellow box highlights a notice: "Hay disponible una nueva versión de Akismet Anti-spam: Spam Protection. [Revisa los detalles de la versión 5.4 o actualízalo ahora.](#)"
- Hello Dolly**: Version 1.7.2 by Matt Mullenweg.
- PromPress**: Version 1.2.2 by Peter Booker.

At the bottom, there are buttons for "Acciones en lote" and "Aplicar".

Refrescamos, y ya desapareció.

Vamos a levantar un compose con redis, ya qué nos hará falta.

```
GNU nano 7.2                                            docker-compose.yml *
redis:
  image: redis:latest
  container_name: redis
  ports:
    - "6379:6379"
```

En un directorio nuevo, lo llamé redis.

```
server@nodol:~/redis$ sudo docker-compose up -d redis
Pulling redis (redis:latest)...
latest: Pulling from library/redis
61320b01ae5e: Already exists
f8f9e5369fdd: Pull complete
b8f8315b617a: Pull complete
eb6labf5b105: Pull complete
03bc4ee78aa8: Pull complete
4f4fb700ef54: Pull complete
93cd3c5153ab: Pull complete
Digest: sha256:b3ad79880c88e302deb5e0fed6cee3e90c0031eb90cd936b01ef2f83ff5b3ff2
Status: Downloaded newer image for redis:latest
Creating redis ... done
server@nodol:~/redis$
```

Nos volvemos a meter en el contenedor de Wordpress...

```

GNU nano 7.2                               wp-config.php

?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the website, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * This has been slightly modified (to read environment variables) for use in Docker.
 *
 * @link https://developer.wordpress.org/advanced-administration/wordpress/wp-config/
 *
 * @package WordPress
 */

// IMPORTANT: this file needs to stay in-sync with https://github.com/WordPress/WordPress/blob/master/wp-conf
// (it gets parsed by the upstream wizard in https://github.com/WordPress/WordPress/blob/f27cb65e1ef25d1b535

// a helper function to lookup "env FILE", "env", then fallback
if (!function_exists('getenv_docker')) {
    // https://github.com/docker-library/wordpress/issues/588 (WP-CLI will load this file 2x)
    function getenv_docker($env, $default) {
        if ($fileEnv = getenv($env . '_FILE')) {
            return rtrim(file_get_contents($fileEnv), "\r\n");
        }
        else if ((${val} = getenv($env)) !== false) {
            return ${val};
        }
        else {
            return $default;
        }
    }
}

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', getenv_docker('WORDPRESS_DB_NAME', 'wordpress') );

/** Database username */
define( 'DB_USER', getenv_docker('WORDPRESS_DB_USER', 'example username') );

/** Database password */
define( 'DB_PASSWORD', getenv_docker('WORDPRESS_DB_PASSWORD', 'example password') );

/**
 * Docker image fallback values above are sourced from the official WordPress installation wizard:
 * https://github.com/WordPress/WordPress/blob/1356f6537220ffdc32b9dad2a6cdbe2d010b7a89/wp-admin/setup-config
 * (However, using "example username" and "example password" in your database is strongly discouraged. Please
 */

```

Buscamos una línea de stop editing, y justo antes de esa línea..ponemos lo siguiente.

```

        eval($configExtra);
}

define( 'WP_REDIS_HOST', 'redis' );
define( 'WP_REDIS_PORT', 6379 );
define( 'WP_CACHE_KEY_SALT', 'misitio.org:' );
define( 'WP_REDIS_DATABASE', 0 );

/* That's all, stop editing! Happy publishing. */

```

Hecho esto...proseguiremos en el siguiente paso.

Paso 4.1.2 Instalar plugin de Redis Object Cache.

Para ello, nos vamos a plugins...

Añadir plugins [Subir plugin](#)

Resultados de la búsqueda Destacados Populares Recomendados Favoritos

Buscar plugins Redis Object Cache Palabra clave ▾

61 elementos << < 1 de 2 > >>



LiteSpeed Cache

[Instalar ahora](#) [Más detalles](#)

Aceleración inmejorable todo en uno y mejora de PageSpeed: almacenamiento en caché, optimización de imágenes/CSS/JS...

Por LiteSpeed Technologies

★★★★★ (2.639) Última actualización: hace 1 mes
7+ millones instalaciones activas ✓ Compatible con tu versión de WordPress



Redis Object Cache

[Instalar ahora](#) [Más detalles](#)

A persistent object cache backend powered by Redis®¹. Supports Predis, PhpRedis, Relay, replication, sentinels, clustering and WP-CLI.

Por Till Krüss

★★★★★ (165) Última actualización: hace 8 meses
200.000+ instalaciones activas No probado con tu versión de WordPress

Tras activarlo, es muy posible que dé un error, pero es porque los contenedores no se ven, la solución, crear una red nueva y meter ambos...

```
server@nodol:~/redis$ sudo docker network create mynet
46e5e8204e6b73dlf0f5b992c7b752aeee2df3b2b2b4fd8ebc75402f8bdf93f0
server@nodol:~/redis$ sudo docker network connect mynet wordpress
sudo docker network connect mynet redis
server@nodol:~/redis$ sudo docker network inspect mynet
[
    {
        "Name": "mynet",
        "Id": "46e5e8204e6b73dlf0f5b992c7b752aeee2df3b2b2b4fd8ebc75402f8bdf93f0",
        "Created": "2025-05-29T11:42:20.475676196Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "a68979f3f64517744ab4cd8480f3f1407004f65eff4e6abbe985e880ad68ec43": {
                "Name": "redis",
                "EndpointID": "b6a008de049c5c03c8f9dccbb4d64fab16e4381d561847613eba012ea77d5157",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            },
            "dc472b99fe94laeb273ab6b4f49b64503e58644c9ae6a145f5e5589f9dd8b6fb": {
                "Name": "wordpress",
                "EndpointID": "ad6f72166b45320ad67c8110ca3bfcd0bf641c14b475217416a3af1fa64504c1",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
server@nodol:~/redis$ ^C
server@nodol:~/redis$ sudo docker restart wordpress
wordpress
```

Redis Object Cache

Resumen Métricas Diagnóstico

Estado	! No activado
Sistema de archivos:	✓ Con permisos de escritura
Redis:	✓ Disponible

Activar la caché de objetos

Recursos

Redis Object Cache

Resumen Métricas Diagnóstico

Estado	✓ Conectado
Sistema de archivos:	✓ Con permisos de escritura
Redis:	✓ Disponible
Prefijo de clave:	misitio.org:

Conexión

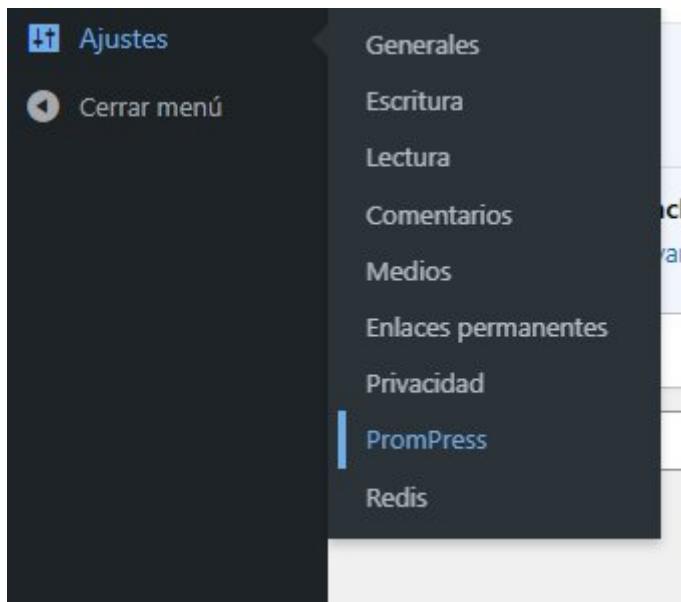
Cliente:	PhpRedis (v6.2.0)
Host:	redis
Puerto:	6379
Base de datos:	0
Tiempo de inactividad de conexión:	1s
Tiempo de inactividad de lectura:	1s
Versión de Redis:	8.0.1

Vaciar Caché Desactivar la caché de objetos

Ya lo tenemos activo.

Paso 4.1.3 Configurar la métrica de PromPress.

Vamos a hacer qué recojan todo lo posible para poder monitorizar nuestro wordpress, el prometheus y grafana.



Y seleccionamos los qué queremos monitorizar.

A screenshot of the 'PromPress Settings' interface. It has three main sections: 'General', 'REST API', and 'Features'.

- General:** A toggle switch labeled 'Active' is set to 'Monitoring is active.' Below it is a button labeled 'Wipe Storage'.
- REST API:** A toggle switch labeled 'Require Authentication' is set to 'Authentication is not required.'
- Features:** A list of seven monitoring features, each with a toggle switch:
 - Emails: 'Track the number of emails sent.'
 - Errors: 'Track the number of errors thrown.'
 - Options: 'Track the number of options.'
 - Posts: 'Track the number of posts (with post type).'
 - Queries: 'Track the number and length of database queries. Note: The 'SAVEQUERIES' constant must be set and true.'
 - Requests: 'Track the number of requests.'

Nos vamos al nodo central, y modificamos el prometheus.yml...

```

GNU nano 7.2                               prometheus.yml

global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporters'
    static_configs:
      - targets:
          - '192.168.239.181:9100'  # VM-WEB
          - '192.168.239.182:9100'  # VM-DB

  - job_name: 'wordpress_prompress'
    metrics_path: /wp-json/prompress/v1/metrics
    static_configs:
      - targets:
          - '192.168.239.181:80'  # WordPress

```

Reiniciamos....y en nuestro prometheus veremos esto...

The screenshot shows the Prometheus web interface under the 'Status > Target health' tab. It displays two sections: 'node_exporters' and 'wordpress_prompress'. Each section lists targets with their endpoints, labels, last scrape time, and state.

node_exporters		1 / 2 up	
Endpoint	Labels	Last scrape	State
http://192.168.239.181:9100/metrics	instance="192.168.239.181:9100", job="node_exporters"	never, 0s	UNKNOWN
http://192.168.239.182:9100/metrics	instance="192.168.239.182:9100", job="node_exporters"	3.061s ago, 12ms	UP

wordpress_prompress		1 / 1 up	
Endpoint	Labels	Last scrape	State
http://192.168.239.181/wp-json/prompress/v1/metrics	instance="192.168.239.181:80", job="wordpress_prompress"	8.63s ago, 23ms	UP

Funciona, ahora podemos agregarlo a grafana y tener todo monitorizado.

Paso 4.1.4 Grafana y Prompress.

En este paso, vamos a agregar el host de dónde tenemos nuestro Wordpress, para poder monitorizarlo más de cerca, y así tenerlo más controlado.

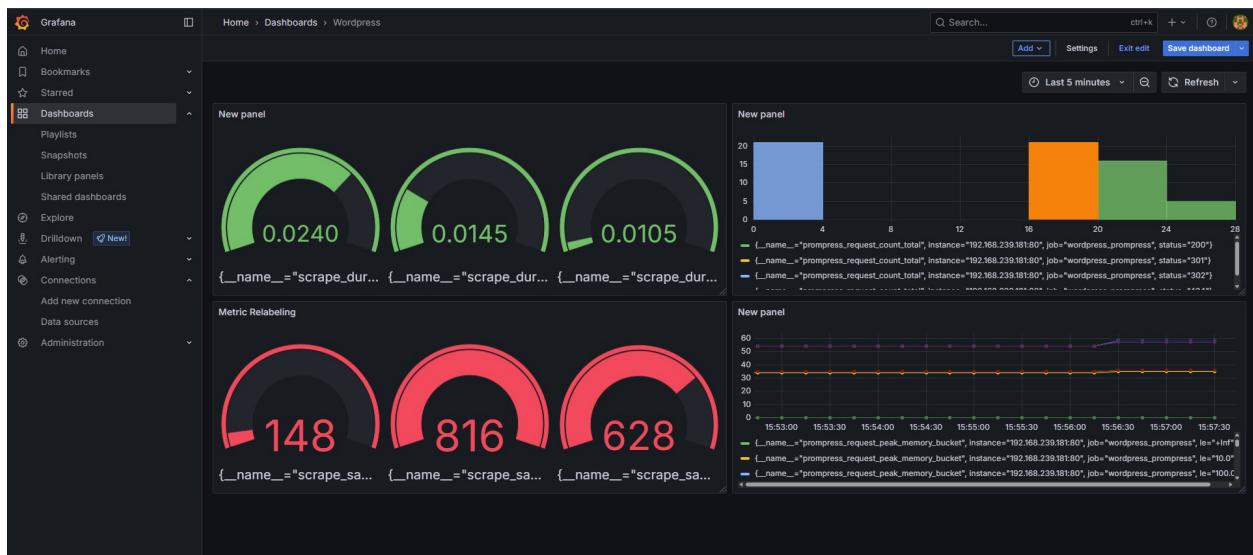
The screenshot shows the Grafana interface for managing dashboards. At the top, there's a navigation bar with a gear icon, 'Home', and 'Dashboards'. To the right are search, add, help, and user icons. Below the navigation is a title 'Dashboards' and a subtitle 'Create and manage dashboards to visualize your data'. A search bar and filter dropdown ('Filter by tag') are present. A 'Starred' checkbox is checked. On the right, there are 'Sort' and 'Tags' buttons. A table lists dashboards: 'Node Exporter Full' (with a 'linux' tag). There are also icons for creating a new dashboard ('New') and a plus sign.

Clicamos en new...y new dashboard.

Y creamos uno, desde cero, ya que no hay muchos qué funcionen decentemente.

This screenshot shows the 'Edit panel' screen for creating a new dashboard. The left sidebar has 'Dashboards' selected. The main area shows a 'New panel' with a 'Time series' visualization. The right sidebar contains various configuration options: 'Panel options' (Title: 'New panel', Description), 'Transparent background' (checkbox), 'Panel links', 'Repeat options', 'Tooltip' (mode: Single, All, Hidden), 'Hover proximity' (checkbox), 'Max width' (input field), and 'Legend' (visibility: checkbox).

Y aquí tenemos un ejemplo de dashboard a partir de las consultas de Prometheus.



CONCLUSIÓN.

En este proyecto se ha demostrado la capacidad y flexibilidad de las herramientas Prometheus y Grafana para implementar una solución de monitorización robusta y escalable. La monitorización es una práctica esencial en cualquier infraestructura moderna, y su correcta implementación aporta un gran valor tanto a los equipos de desarrollo como a los de operaciones, facilitando la detección temprana de problemas y la optimización del rendimiento.

A nivel personal, este proyecto ha sido muy satisfactorio. Tenía ganas de probar este stack de herramientas, y la experiencia ha sido muy positiva. Tras haber trabajado anteriormente con otras soluciones como Zabbix, puedo decir que esta combinación de Prometheus y Grafana ofrece una gran versatilidad y facilidad de uso, además de un ecosistema en constante evolución que se adapta muy bien a diferentes escenarios.

En resumen, Prometheus y Grafana constituyen una opción potente y moderna para la monitorización que merece la pena explorar y adoptar en entornos profesionales y personales.