



PIG

PostgreSQL, InfluxDB and Grafana

SANTIAGO PREGO LÓPEZ – ASIR 2020 – IES SAN CLEMENTE

¿Qué es PIG?



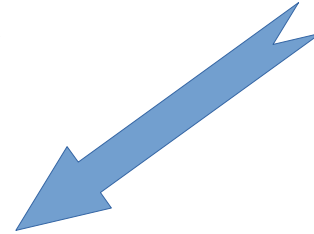
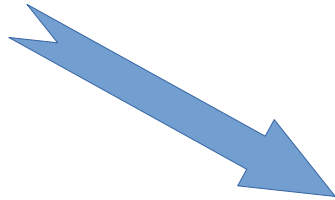
PostgreSQL



influxdb

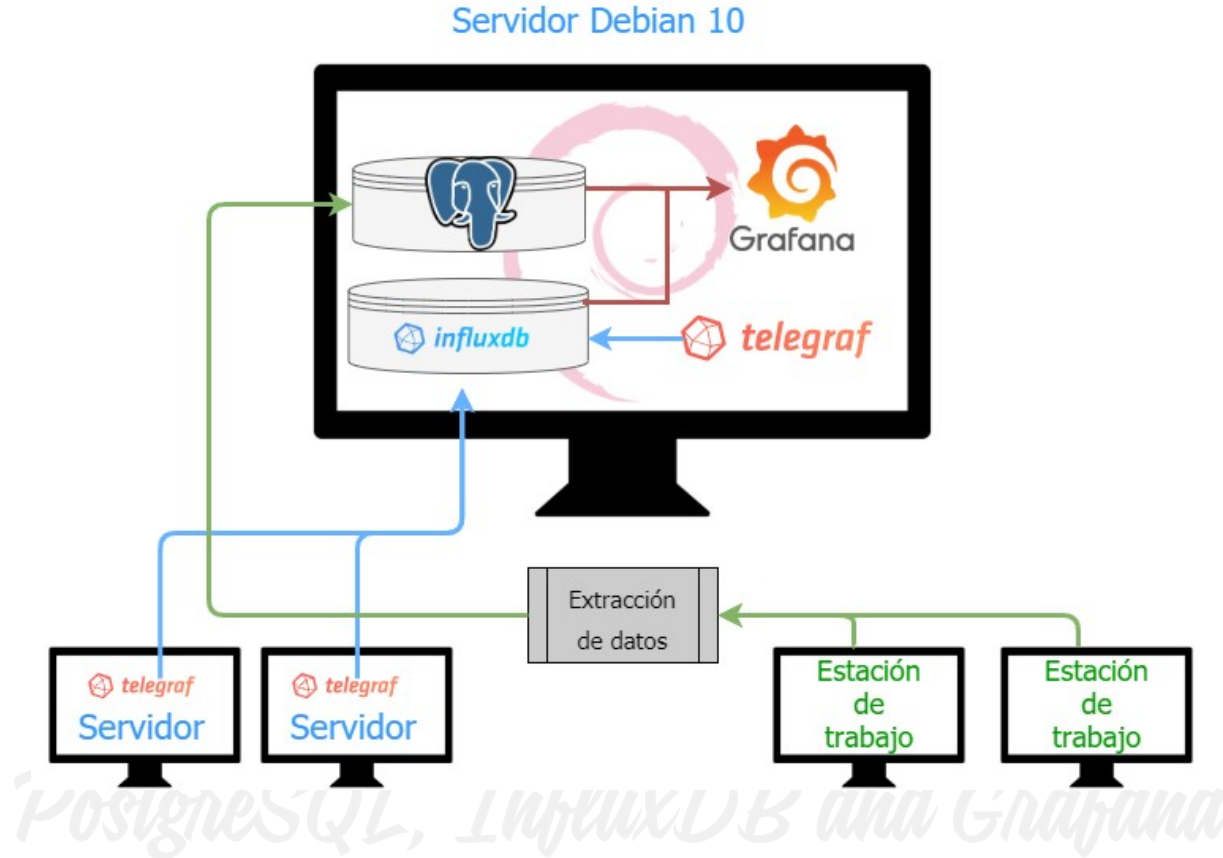


Grafana



PostgreSQL, InfluxDB and Grafana

Diagrama de componentes



Trabajo Realizado

Instalación Debian
Instalación PostgreSQL
Instalación Influxdb
Instalación Telegraf
Instalación Grafana
Escaneo

PostgreSQL, InfluxDB and Grafana

Instalación Debian 10

Realicé una instalación mínima sin entorno de escritorio y fui agregando los paquetes necesarios para reducir el tamaño final.

PIG

PostgreSQL, InfluxDB and Grafana

Instalación PostgreSQL 13

The screenshot displays the PgAdmin 4 web interface. The left sidebar shows a tree view of the database structure, including Databases (2), postgres, workstations, Casts, Catalogs (2), Event Triggers, Extensions (1), Foreign Data Wrappers, Languages, Schemas (1), public, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Procedures (5), Sequences, Tables (6), and Trigger Functions. The main panel shows the Properties tab for a PostgreSQL server. The server type is PostgreSQL, the version is PostgreSQL 13.0 (Debian 13.0-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit. The connection status is True, and the host name/address is 192.168.1.86. The port is 5432, the maintenance database is postgres, the username is postgres, and the role is postgres. The SSL mode is Require, and the SSL compression is No.

PgAdmin File ▾ Object ▾ Tools ▾ Help ▾

Browser Dashboard Properties SQL Statistics Dependencies Dependents

Server Properties Edit

Server type: PostgreSQL

Version: PostgreSQL 13.0 (Debian 13.0-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit

Comments:

Connection

Connected? ☒ True

Host name/address: 192.168.1.86

Port: 5432

Maintenance database: postgres

Username: postgres

Role:

Service:

SSL

SSL mode: Require

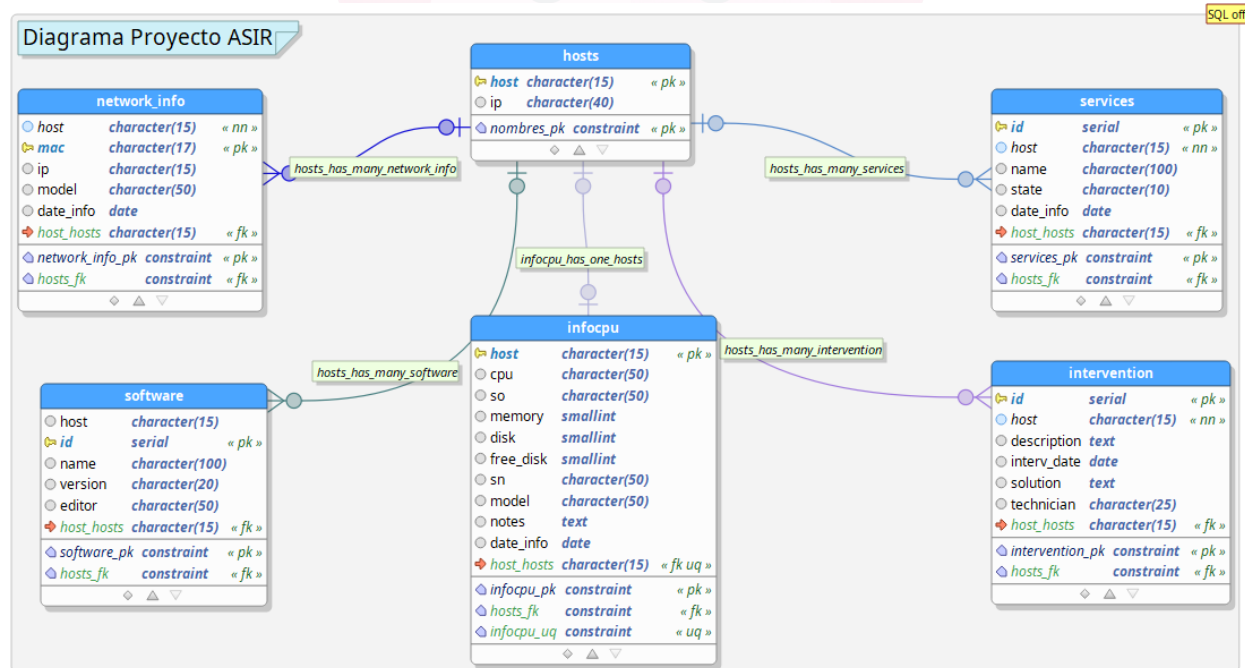
Client certificate key:

SSL compression? ☐ No

SSH Tunnel

Base de datos

Se creó una base de datos de muestra para almacenar los datos recopilados por el escaneo.



Base de datos

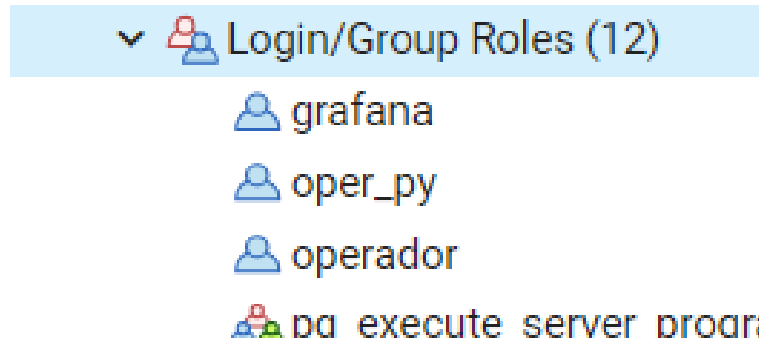
Cree procedimientos almacenados para la inserción de los datos.

▼ Code

```
1 BEGIN
2     INSERT INTO infocpu(host, cpu, so, memory, disk, free_disk, sn, model)
3     VALUES (P_host, P_cpu, P_so, P_memory, P_disk, P_free_disk, P_sn, P_model)
4     ON CONFLICT (host)
5     DO
6         UPDATE SET cpu = P_cpu, so = P_so, memory = P_memory, disk = P_disk,
7         free_disk = P_free_disk, sn = P_sn, model = P_model, date_info = NOW();
8
9 EXCEPTION WHEN OTHERS THEN
10     P_msg_error := SQLERRM;
11 END;
12
```


Base de datos

Se crearon diferentes usuarios para limitar el acceso a lo necesario.



PostgreSQL, InfluxDB and Grafana

Certificado auto firmado

Se crea el certificado y la clave privada para configurar posteriormente https en las aplicaciones.

```
openssl req -x509 -out localhost.crt -keyout localhost.key \
    -newkey rsa:2048 -nodes -sha256 \
    -subj '/CN=localhost' -extensions EXT -config <( \
printf "[dn]\nCN=localhost\n[req]\ndistinguished_name = dn\n[EXT]\n\
subjectAltName=DNS:localhost\nkeyUsage=digitalSignature\n\
nextendedKeyUsage=serverAuth")
```

PostgreSQL, InfluxDB and Grafana

Influxdb 1.8

Se instaló y configuró Influxdb habilitando https y creando un usuario de acceso.

```
GNU nano 3.2 /etc/influxdb/influxdb.conf
```

```
# ping-auth-enabled = false
```

```
# Determines whether HTTPS is enabled.
```

```
https-enabled = true
```

```
# The SSL certificate to use when HTTPS is enabled.
```

```
https-certificate = "/etc/influxdb/localhost.crt"
```

```
# Use a separate private key location.
```

```
https-private-key = "/etc/influxdb/localhost.key"
```

```
root@pig-server:~# influx -ssl -unsafeSsl -host localhost -username admin -password abc123.
```

```
Connected to https://localhost:8086 version 1.8.3
```

```
InfluxDB shell version: 1.8.3
```

```
> show users
```

```
user      admin
```

```
-----
```

```
admin     true
```

```
grafana   false
```

```
tg_user   false
```

```
> show databases
```

```
name: databases
```

```
name
```

```
-----
```

```
_internal
```

```
telegraf
```

```
>
```

PostgreSQL

Instalación Telegraf

Se realizó una instalación básica de Telegraf con la base de datos por defecto, y se configuró el acceso a Influxdb.

GNU nano 3.2 /etc/telegraf/telegraf.conf

```
## urls will be written to each interval.  
# urls = ["unix:///var/run/influxdb.sock"]  
# urls = ["udp://127.0.0.1:8089"]  
urls = ["https://127.0.0.1:8086"]  
  
## The target database for metrics: will be create
```

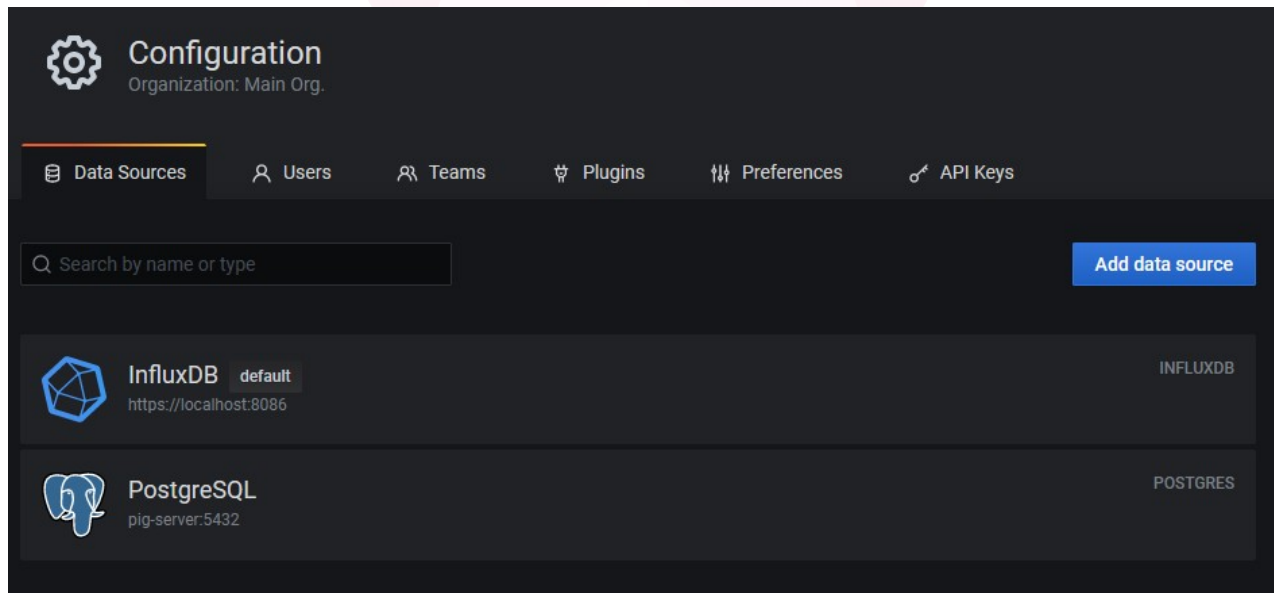
GNU nano 3.2 /etc/telegraf/telegraf.conf

```
## Timeout for HTTP messages.  
# timeout = "5s"  
  
## HTTP Basic Auth  
username = "admin"  
password = "abc123."
```

PostgreSQL, In

Instalación de Grafana 7.2

Se instala y configuran las fuentes de datos en Grafana



Instalación de Grafana 7.2

Se habilita la conexión a través de https

```
GNU nano 3.2 /etc/grafana/grafana.ini
##### Server #####
[server]
# Protocol (http, https, h2, socket)
protocol = https
```

```
GNU nano 3.2 /etc/grafana/grafana.ini
# https certs & key file
cert_file = /etc/grafana/localhost.crt
cert_key = /etc/grafana/localhost.key
```

PostgreSQL, InfluxDB and Grafana

Instalación de Grafana 7.2

Se crean paneles de muestra para el servidor y los datos de las estaciones de trabajo.



Instalación de Grafana 7.2

Para el panel de software se usaron consultas de ejemplo sobre la base de datos PostgreSQL.

The image displays three overlapping screenshots of the Grafana 7.2 web interface, specifically the 'Query' editor for a PostgreSQL data source. The interface is dark-themed. Each screenshot shows a query editor with a SQL query, a 'Format as' dropdown set to 'Table', and buttons for 'Query Builder', 'Show Help', and 'Generated SQL'.

Top Left Screenshot: Shows a query for LibreOffice software.

```
SELECT
(count(*) * 100 / (SELECT count(*) from software where name LIKE 'LibreOffice%'))
FROM
software
where name = 'LibreOffice 6.2.7.1';
```

Top Right Screenshot: Shows a query for Microsoft .NET Framework software.

```
SELECT
(count(*) * 100 / (SELECT count(*) from software where name LIKE 'Microsoft .NET%'))
FROM
software
where name = 'Microsoft .NET Framework 4.8';
```

Bottom Screenshot: Shows a query for Microsoft Windows software.

```
SELECT
count(*) FROM infocpu
where so LIKE 'Microsoft Windows 10%';
```

The background of the image features a large, faint, pink circular logo with a stylized face, and the words 'PostgreSQL' and 'Grafana' are written in a light, cursive font at the bottom.

Escaneo de equipos

Se creó una muestra de software para escaneo de equipos usando Python y WMI.

Se realizó un escaneo de prueba obteniendo datos de casi 800 equipos.

PostgreSQL, InfluxDB and Grafana

Escaneo de equipos

Se puede escanear mediante un rango de IP o usando un listado de equipos.

```
import data_module as dataw
import dbpg_module as dbpg
import socket

red_ini = [192,168,1,86] # Primera IP donde comenzará el escaneo
red_fin = [192,168,1,142] # Ultima IP que escaneará

# Comprobamos si el host admite la conexión al puerto 135 (RPC) us
def scan(IP):
    s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    socket.setdefaulttimeout(0.50)
```

```
try:
    #En esta conexión se especifica un usuario concreto con el que conectarse a los equipos remotos, este debe de tener permisos para realiza
    #c = wmi.WMI(computer=computer, impersonation_level='impersonate', authentication_level='pktPrivacy', user="sprego", password="abc123.")
    c = wmi.WMI(computer=computer, impersonation_level='impersonate', authentication_level='pktPrivacy') # Con esta conexión el usuario que e
    for interface in c.Win32_NetworkAdapterConfiguration(IPEnabled=1): # IPEnabled=1 indica la tarjeta activa
        model = interface.description
        mac = interface.MACAddress
        ip = interface.IPAddress[0]
        dbpg.SaveData().network_info(computer,mac,ip,model)
except wmi.x_wmi as e:
    print(e.com_error.strerror,e.com_error.exceinfo[2])
    return e.com_error.strerror,e.com_error.exceinfo[2]
```

```
try:
    with open("PCS.txt", "r", encoding='utf-8') as PCS: # Abrimos el fichero donde almacenamos los equipos a escanear.
        hosts = PCS.readlines()
        i = 0
        for host in hosts:
            i=i+1
            host=host.split("\n")
            scan(host[0]) # Escaneamos cada host para ver si nos podemos conectar.
except IOError as e:
    print("I/O error({0}): {1}".format(e.errno, e.strerror))
    pass
```

Tiempos de escaneo

Se realizaron varias pruebas para comprobar el tiempo que llevaba el escaneo.

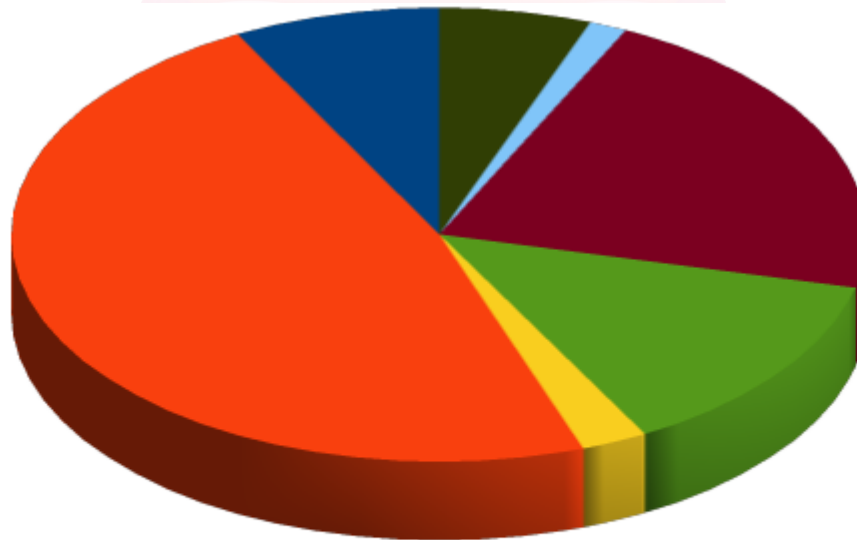
Escaneo de lista de 800 pcs
Detectados 733 equipos
Tiempo empleado entre 18 y 24 horas

Escaneo de 102 ips
Detectados 25 equipos
Tiempo empleado 23 Minutos

Escaneo de 253 ips
Detectados 63 equipos
Tiempo Empleado 198 minutos

PostgreSQL, InfluxDB and Grafana

Distribución de tiempos



■ F1 - Creacion del entregable

■ F1 - Documentación

■ F2 - Modulo de escaneo de red.

■ F2 - Modulo de almacenamiento de datos.

■ F2 - Modulo de extraccion de datos.

■ F2 - Modulo de escaneo de lista

■ F2 - Pruebas de escaneo

PostgreSQL, InfluxDB and Grafana