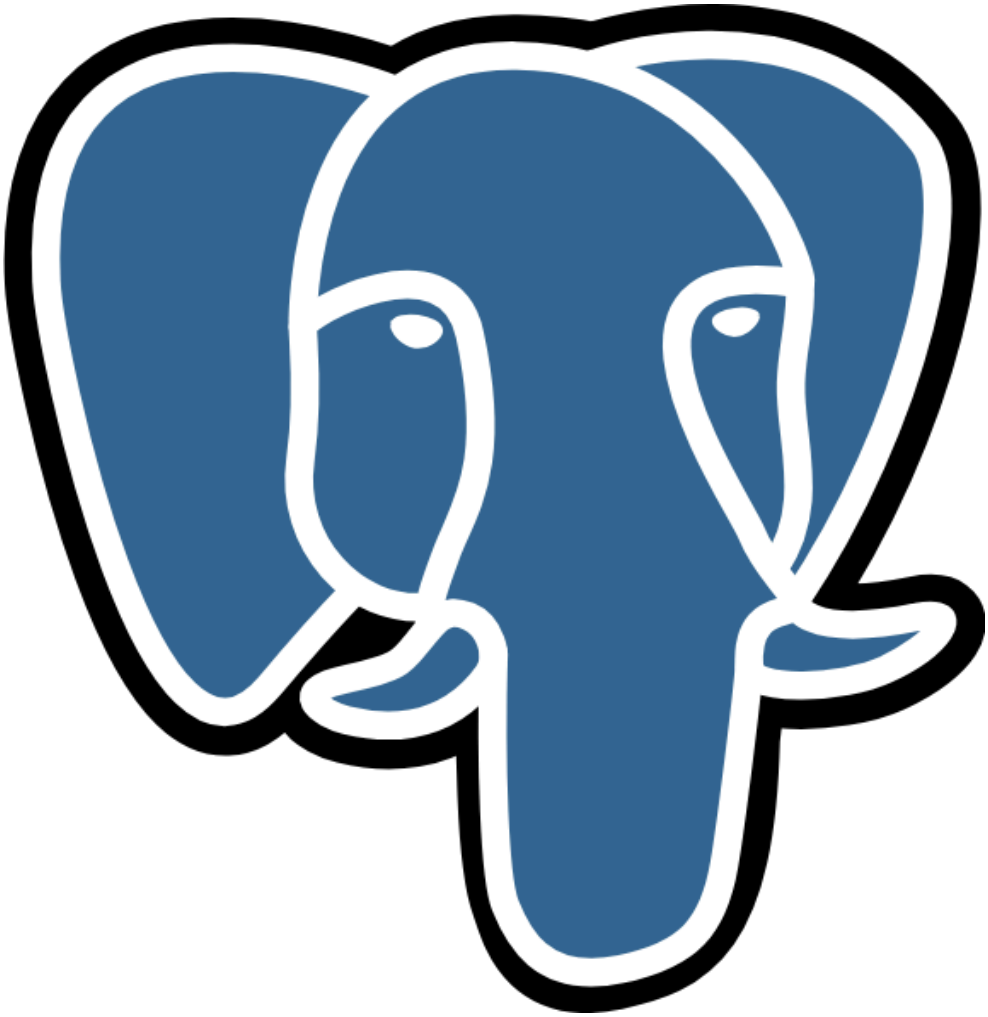
BBDD DISTRIBUIDA CON RÉPLICA Y HA



IES SUAREZ DE FIGUEROA

Técnico Superior en Administración de Sistemas Informáticos

Autor: Shenhao Zhou

Curso Académico: ASIR 2º

ÍNDICE

1. INTRODUCCIÓN 3
2. OBJETIVOS 4
3. MARCO TEÓRICO 5
    3.1 PostgreSQL 5
    3.2 Alta Disponibilidad (HA) 5
    3.3 Replicación en PostgreSQL 6
    3.4 Docker 6
    3.5 HAProxy 7
    3.6 Prometheus y Grafana 7
4. DESARROLLO DEL PROYECTO 8
    4.1 Diseño 8
    4.2 Requisitos de Hardware y Software 10
    4.3 Instalación de docker compose 11
    4.4 Configuración de los archivos de compose y estructura de las carpetas 15
    4.5 Ejecucion del Docker compose 16
    4.6 Testeo de BBDD 18
        Prueba de replicación 20
    4.7 HAProxy 21
    4.8 Grafana 24
5. CONCLUSIONES 27
    BIBLIOGRAFÍA 28
6. MANUAL DE USUARIO 29
    Introducción 29
    URLs y Puertos 30
    Conexión a la Base de Datos 31
    Operaciones Básicas 32
        Select Datos 32
        Insertar Datos 32
        Update Datos 33
        Delete Datos 33
    Panel de Estadísticas de HAProxy 34
    Grafana 35
7. MANUAL DE INSTALACIÓN 36
    Despliegue de infraestructura 36

1. INTRODUCCIÓN

Este es un proyecto que trata sobre hacer un base de datos distribuida en réplica, donde se utilizara Docker Compose para los distintos contenedores que se implementan en la arquitectura del proyecto. La arquitectura se basa en 3 contenedores de PostgreSQL para la bbdd, 1 HAProxy para el failover y balanceo de cargas, Prometheus, Exporter y Grafana para la monitorización de los datos con Dashboard, donde se virtualiza en VMware Workstation mediante una instalación de ISO de Debian 12

2. OBJETIVOS

El objetivo es poder diseñar e implementar un sistema de bbdd PostgreSQL con HA utilizando tecnologías de código abierto (Docker Compose, VMware, PostgreSQL)

También el objetivo es el aprendizaje de:

- Crear una arquitectura distribuida basada en un nodo principal y múltiples réplicas sincrónicas
- Implementar un sistema de balanceo de carga y failover mediante HAProxy
- Establecer un sistema de monitorización utilizando Prometheus y Grafana
- Realizar pruebas de rendimiento, disponibilidad y recuperación ante fallos para validar la eficacia de la solución implementada
- Proporcionar una base sólida que pueda ser adaptada y escalada para entornos de producción reales con requisitos específicos de disponibilidad y rendimiento

3. MARCO TEÓRICO

Para entender el proyecto se describen los conceptos y tecnologías que se utilizan en este proyecto. Donde se dividirá en distintos apartados

3.1 PostgreSQL

PostgreSQL es una base de datos de código abierto que tiene una sólida reputación por fiabilidad, flexibilidad y código abierto. Donde soporta transacciones seguras (ACID), múltiples tipos de datos (datos estructurados y no estructurados), y permite usar funciones personalizadas

3.2 Alta Disponibilidad (HA)

La alta disponibilidad (HA) se refiere a la capacidad de un sistema para mantenerse funcional durante mucho tiempo, si ocurre alguna fallo se minimice el tiempo de la parada del sistema, para este proyecto implica en:

- Redundancia: Múltiples copias de los datos y servidores que pueden asumir la carga en caso de fallo en el primary
- Detección de fallos: Identificar rápidamente cuando un componente no funciona correctamente mediante la monitorización
- Recuperación automática: Poder restaurar la funcionalidad de la BBDD de forma automática
- Balanceo de carga: Distribuir la carga a la BBDD de forma eficiente para optimizar el rendimiento

3.3 Replicación en PostgreSQL

PostgreSQL ofrece varios métodos de replicación:

- Replicación física: Copia byte a byte de los archivos de datos.
    - Streaming Replication: Método principal que transfiere registros WAL (Write-Ahead Log) en tiempo real
    - Replicación sincrónica: Garantiza que las transacciones se confiRman solo cuando se han aplicado en al menos un servidor de réplica
    - Replicación asincrónica: Permite cierto retraso entre el nodo primario y las réplicas
- Replicación lógica: Réplica cambios específicos a nivel de objeto, permitiendo replicación selectiva de tablas o esquemas.
- Slots de replicación: Mecanismo para garantizar que los segmentos WAL necesarios para la replicación no se eliminen prematuramente.

3.4 Docker

Docker es una plataforma de código abierto para el despliegue de aplicaciones dentro de contenedores. Las ventajas usar Docker son:

- Aislamiento: Los contenedores funcionan de manera independiente sin interferir con otras máquinas o servidor
- Portabilidad: Un contenedor se ejecuta de la misma manera en cualquier entorno que soporte Docker
- Escalabilidad: Permite crear múltiples instancias de un contenedor y distribuir la carga de trabajo de manera eficiente
- Eficiencia de recursos: Los contenedores comparten el mismo kernel del sistema operativo, consumiendo menos recursos que las máquinas virtuales
- Docker Compose: Herramienta que te permite definir, configurar y ejecutar múltiples contenedores Docker de forma sencilla, usando un único archivo de configuración .yml

3.5 HAProxy

HAProxy (High Availability Proxy) es una herramienta de código abierto que proporciona balanceo de carga y proxy de alta disponibilidad

- Balanceo de carga para las consultas entre múltiples servidores.
- Failover cuando un servidor no está disponible y lo excluye automáticamente.
- Separa las operaciones de lectura y escritura de la BBDD
- Panel de estadísticas para monitorizar el tráfico y el estado de los servidores.

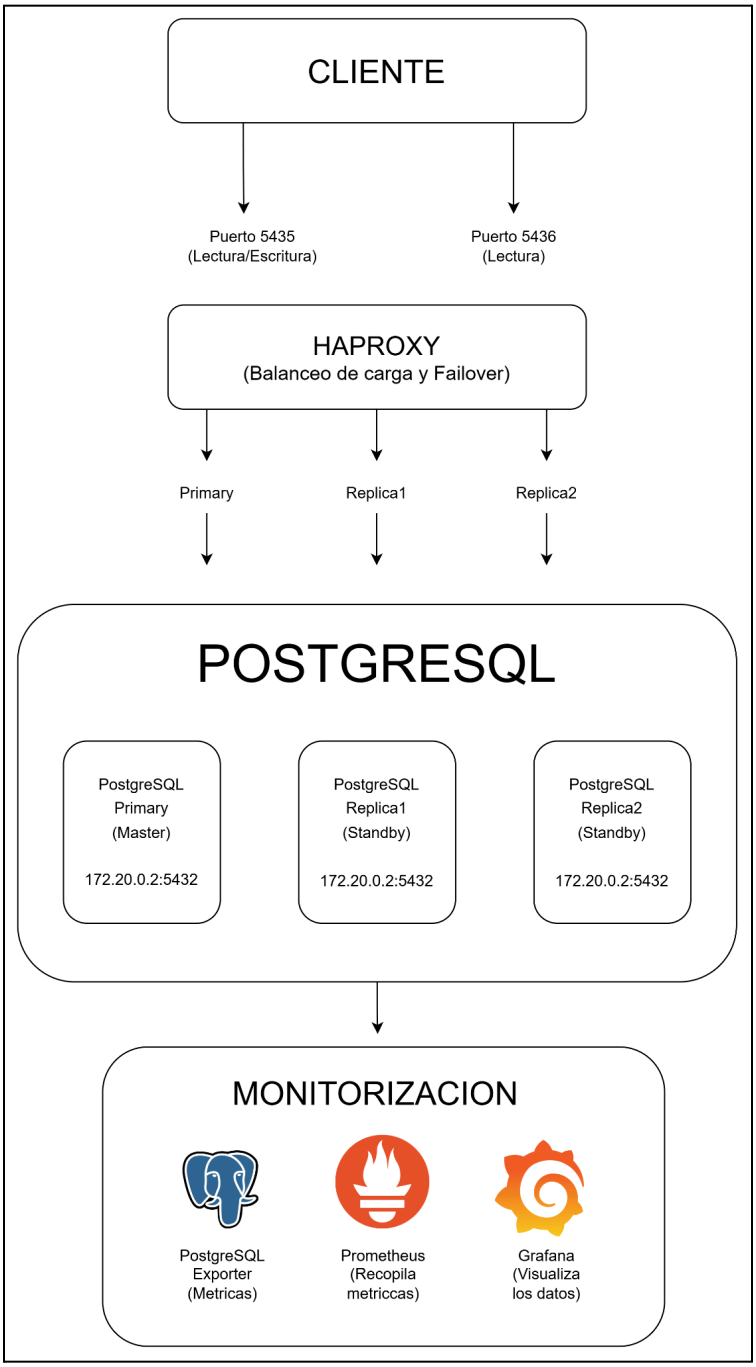
3.6 Prometheus y Grafana

La monitorización permite ver y recopilar información sobre el funcionamiento de un sistema para detectar los problemas, optimizar los recursos y garantizar su funcionamiento

- Prometheus: Es un software especializado como sistema de monitorización y alertas, donde los datos se almacenan en una base de datos de series temporales
- Grafana: Es un software libre que permite la visualización y el formato de datos métricos
- Postgres Exporter: Proporciona métricas detalladas sobre el estado y rendimiento de la base de datos

4. DESARROLLO DEL PROYECTO

4.1 Diseño



La arquitectura de este proyecto se basa en un modelo de alta disponibilidad para PostgreSQL que está formado por los siguientes componentes:

PostgreSQL Primario: Servidor principal que permite operaciones de lectura y escritura

PostgreSQL Réplica: Dos servidores configurados en modo réplica sincrónica que copian los datos del primary y pueden promocionarse para sustituir al primary en caso de que caiga

HAProxy: Actúa como punto de entrada al sistema, proporciona dos endpoints:

- Puerto 5435: Para lectura/escritura, dirigidas al nodo primario
- Puerto 5436: Sólo para lectura, con balanceo de carga entre todos los nodos disponibles

Sistema de Monitorización:

- Postgres Exporter: Extrae métricas de PostgreSQL
- Prometheus: Recopila las métricas de todos los componentes
- Grafana: Visualiza las métricas en dashboards personalizados

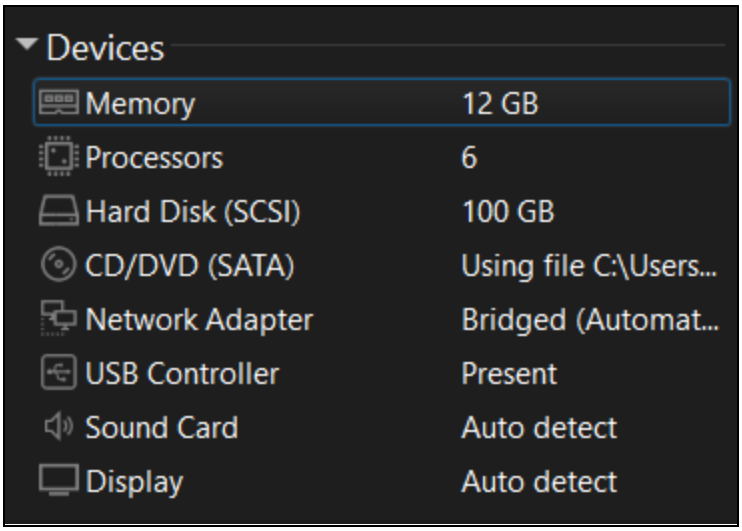
4.2 Requisitos de Hardware y Software

Requisitos hardware

- CPU: 2 núcleos (recomendado 4+)
- RAM: 8GB (recomendado 12GB)
- Almacenamiento: 30 GB (recomendado 50GB+)

Software base:

- VMware Workstation 17 Pro (17.5.1 build-23298084)
- Sistema Operativo: Debian 12 (Bookworm)
- Docker Engine: 24.0.0 o superior
- Docker Compose: 2.20.0 o superior



4.3 Instalación de docker compose

Se actualiza el sistema de debian 12

```
shen@debian:~$ sudo apt update && sudo apt upgrade -y
Obj:1 http://deb.debian.org/debian bookworm InRelease
Des:2 http://security.debian.org/debian-security bookworm-security InRelease [48,0 kB]
Des:3 http://deb.debian.org/debian bookworm-updates InRelease [55,4 kB]
Des:4 https://download.docker.com/linux/debian bookworm InRelease [47,0 kB]
Des:5 http://security.debian.org/debian-security bookworm-security/main Sources [148 kB]
Des:6 http://security.debian.org/debian-security bookworm-security/main amd64 Packages [250 kB]
Des:7 https://download.docker.com/linux/debian bookworm/stable amd64 Packages [38,4 kB]
```

Se instala la herramienta de curl para transferencia de datos desde o hacia un servidor y el vim un editor de texto avanzado

```
shen@debian:~$ sudo apt install -y curl vim
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  git-man liberror-perl openssh-sftp-server patch runit-helper
Utilice «sudo apt autoremove» para eliminarlos.
Paquetes sugeridos:
  ctags vim-doc vim-scripts
Se instalarán los siguientes paquetes NUEVOS:
```

Se instala el ufw para poder abrir los puertos

```
shen@debian:~$ sudo apt install -y ufw
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  git-man liberror-perl openssh-sftp-server patch runit-helper
Utilice «sudo apt autoremove» para eliminarlos.
Paquetes sugeridos:
  rsyslog
Se instalarán los siguientes paquetes NUEVOS:
  ufw
```

Se abren los distintos puertos necesarios para las del proyecto

```
shen@debian:~$ sudo ufw allow 5432/tcp
sudo ufw allow 5433/tcp
sudo ufw allow 5434/tcp
sudo ufw allow 5435/tcp
sudo ufw allow 5436/tcp
sudo ufw allow 8404/tcp
sudo ufw allow 9090/tcp
sudo ufw allow 3000/tcp
sudo ufw allow 9187/tcp
```

Se habilita el ufw y comprueba el estado de los puertos que previamente que he abierto

```
shen@debian:~$ sudo ufw enable
Firewall is active and enabled on system startup
shen@debian:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
5432/tcp                   ALLOW       Anywhere
5433/tcp                   ALLOW       Anywhere
5434/tcp                   ALLOW       Anywhere
5435/tcp                   ALLOW       Anywhere
5436/tcp                   ALLOW       Anywhere
8404/tcp                   ALLOW       Anywhere
9090/tcp                   ALLOW       Anywhere
3000/tcp                   ALLOW       Anywhere
9187/tcp                   ALLOW       Anywhere
5432/tcp (v6)              ALLOW       Anywhere (v6)
5433/tcp (v6)              ALLOW       Anywhere (v6)
5434/tcp (v6)              ALLOW       Anywhere (v6)
5435/tcp (v6)              ALLOW       Anywhere (v6)
5436/tcp (v6)              ALLOW       Anywhere (v6)
8404/tcp (v6)              ALLOW       Anywhere (v6)
9090/tcp (v6)              ALLOW       Anywhere (v6)
3000/tcp (v6)              ALLOW       Anywhere (v6)
9187/tcp (v6)              ALLOW       Anywhere (v6)
```

Al estar en debian se necesita instalar dependencias para el docker

```
shen@debian:~$ sudo apt install -y apt-transport-https ca-certificates curl gnupg
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
ca-certificates ya está en su versión más reciente (20230311).
curl ya está en su versión más reciente (7.88.1-10+deb12u12).
gnupg ya está en su versión más reciente (2.2.40-1.1).
```

Se agrega la clave GPG oficial de Docker

```
shen@debian:~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings
/docker-archive-keyring.gpg
```

Se añade un repositorio para poder instalar el paquete de Docker

```
shen@debian:~$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.g
pg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.
list > /dev/null
```

Una vez hecho todo, se actualiza el índice de los paquetes de debian e instala el Docker, y Docker compose

```
shen@debian:~$ sudo apt update && sudo apt install -y docker-ce docker-ce-cli containerd.io
Obj:1 http://security.debian.org/debian-security bookworm-security InRelease
Obj:2 http://deb.debian.org/debian bookworm InRelease
Obj:3 http://deb.debian.org/debian bookworm-updates InRelease
Obj:4 https://download.docker.com/linux/debian bookworm InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Todos los paquetes están actualizados.
```

```
shen@debian:~$ sudo apt install -y docker-compose-plugin
[sudo] contraseña para shen:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
docker-compose-plugin ya está en su versión más reciente (2.35.1-1~debian.12~bookworm).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 47 no actualizados.
```

```
shen@debian:~$ sudo apt install -y docker-compose
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  libslirp0 openssh-sftp-server pigz runit-helper slirp4netns
Utilice «sudo apt autoremove» para eliminarlos.
```

Una vez hecho todo, lo que se hará es añadir el usuario al grupo de docker para no tener que solicitar el sudo para usar docker

```
shen@debian:~$ sudo usermod -aG docker $USER
newgrp docker
```

Para verificar la correcta instalación, se comprueba las versiones del docker y docker compose, si nos muestra sin problema es que esta instalado correctamente

```
shen@debian:~$ docker --version
docker compose version
Docker version 28.1.1, build 4eba377
Docker Compose version v2.35.1
```

```
shen@debian:~$ docker compose version
Docker Compose version v2.35.1
```

4.4 Configuración de los archivos de compose y estructura de las carpetas

Se crea un directorio raíz para el proyecto

```
shen@debian:~$ mkdir -p ~/bbdd-ha
shen@debian:~$ cd ~/bbdd-ha
shen@debian:~/bbdd-ha$ mkdir -p data/{primary,replica1,replica2}
shen@debian:~/bbdd-ha$ mkdir -p scripts
shen@debian:~/bbdd-ha$ mkdir -p configs/{haproxy,prometheus}
```

Dentro del directorio se crea un archivo docker-compose.yml

```
shen@debian:~/bbdd-ha$ ls
configs  data  docker-compose.yml  scripts
```

Y dentro del directorio de scripts se crea 2 archivos .sh, uno para el init-primary y otro para init-replica

```
shen@debian:~/bbdd-ha$ nano scripts/init-primary.sh
shen@debian:~/bbdd-ha$ nano scripts/init-replica.sh
shen@debian:~/bbdd-ha$ ls scripts/
init-primary.sh  init-replica.sh
```

Se crea dentro del directorio de configs otro directorio llamado prometheus

```
shen@debian:~/bbdd-ha$ nano configs/prometheus/prometheus.yml
shen@debian:~/bbdd-ha$ ls configs/prometheus/
prometheus.yml
```

También se configura el archivo de de HAProxy con su directorio

```
shen@debian:~/bbdd-ha$ nano configs/haproxy/haproxy.cfg
shen@debian:~/bbdd-ha$ ls configs/haproxy/
haproxy.cfg
```

Se le otorga a los distintos archivos el poder ejecutarse además de dar permisos al usuario para no tener problemas en los distintos directorios

```
shen@debian:~/bbdd-ha$ chmod +x scripts/init-primary.sh scripts/init-replica.sh docker-compose.yml
```

```
shen@debian:~/bbdd-ha$ sudo chown -R $USER:$USER data configs scripts
[sudo] contraseña para shen:
```

4.5 Ejecucion del Docker compose

Se levanta los distintos contenedores del compose poco a poco para que no haya ningún error, aunque se podría levantar todos de a la vez

```
shen@debian:~/bbdd-ha$ docker-compose up -d postgres-primary
Creating network "bbdd-ha_postgres_network" with driver "bridge"
Creating volume "bbdd-ha_prometheus_data" with default driver
Creating volume "bbdd-ha_grafana_data" with default driver
Pulling postgres-primary (postgres:16)...
16: Pulling from library/postgres
254e724d7786: Pull complete
62999f964559: Pull complete
5ad015837561: Pull complete
4daf2c240273: Pull complete
79bbba0de13f: Pull complete
d51b66b14355: Pull complete
f5e640a97aed: Pull complete
8ff17c953578: Pull complete
```

```
shen@debian:~/bbdd-ha$ docker-compose up -d postgres-replica1 postgres-replica2
postgres-primary is up-to-date
Creating postgres-replica1 ... done
Creating postgres-replica2 ... done
```

```
shen@debian:~/bbdd-ha$ docker-compose up -d haproxy
Pulling haproxy (haproxy:latest)...
latest: Pulling from library/haproxy
254e724d7786: Already exists
7e9f826aa69a: Pull complete
e9655d50a840: Pull complete
c413db750e61: Pull complete
dfbc1d044541: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:a39b950837857b6a7b5349a1b8b5ee204bb55f1bd6a22468f0b426a936de2804
Status: Downloaded newer image for haproxy:latest
```

```
shen@debian:~/bbdd-ha$ docker-compose up -d prometheus grafana postgres_exporter
Pulling prometheus (prom/prometheus:latest)...
latest: Pulling from prom/prometheus
9fa9226be034: Pull complete
1617e25568b2: Pull complete
dd1e13a1db5e: Pull complete
85395c032c94: Pull complete
96a99fc8b470: Pull complete
6d243588c032: Pull complete
c66110f5fa59: Pull complete
759fe8dc37ae: Pull complete
d71c3f577a67: Pull complete
4c0194f7eb43: Pull complete
Digest: sha256:e2b8aa62b64855956e3ec1e18b4f9387fb6203174a4471936f4662f437f04405
Status: Downloaded newer image for prom/prometheus:latest
Pulling grafana (grafana/grafana:latest)...
```

Y se comprueba los distintos contenedores de Docker Compose

```
shen@debian:~/bbdd-ha$ docker-compose ps -a
      Name                     Command                  State                                      Ports
-----------------------------------------------------------------------------------------------------------------------
grafana             /run.sh                          Up             0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
haproxy             docker-entrypoint.sh hapro ...   Up             0.0.0.0:5435->5435/tcp,:::5435->5435/tcp,
                                                                    0.0.0.0:5436->5436/tcp,:::5436->5436/tcp,
                                                                    0.0.0.0:8404->8404/tcp,:::8404->8404/tcp
postgres-primary    docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5432->5432/tcp,:::5432->5432/tcp
postgres-replica1   docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5433->5432/tcp,:::5433->5432/tcp
postgres-replica2   docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5434->5432/tcp,:::5434->5432/tcp
postgres_exporter   /bin/postgres_exporter           Up             0.0.0.0:9187->9187/tcp,:::9187->9187/tcp
prometheus          /bin/prometheus --config.f ...   Up             0.0.0.0:9090->9090/tcp,:::9090->9090/tcp
```

4.6 Testeo de BBDD

Para el testeo de la BBDD se necesita el cliente de postgresql

```
shen@debian:~/bbdd-ha$ sudo apt install -y postgresql-client
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesaric
  libslirp0 openssh-sftp-server pigz runit-helper slirp4netns
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  libpq5 postgresql-client-15 postgresql-client-common
Paquetes sugeridos:
```

Se comprueba el nodo primario, si nos devuelve un f “false” significa que no esta en modo réplica.

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-primary psql -U shen -d prueba -c "SELECT pg_is_in_recovery();"
 pg_is_in_recovery
-------------------
 f
(1 row)
```

Ahora se comprueba los dos nodos de réplica, y nos retorna un valor t “true” indicando que esta en modo replicación

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-replica1 psql -U shen -d prueba -c "SELECT pg_is_in_recovery();"
 pg_is_in_recovery
-------------------
 t
(1 row)

shen@debian:~/bbdd-ha$ docker exec -it postgres-replica2 psql -U shen -d prueba -c "SELECT pg_is_in_recovery();"
 pg_is_in_recovery
-------------------
 t
(1 row)
```

Después comprueba los slots que hay en el primario de replicación

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-primary psql -U shen -d prueba -c "SELECT * FROM pg_replication_slots;"
   slot_name   | plugin | slot_type | datoid | database | temporary | active | active_pid | xmin | catalog_xmin | restart_lsn | confirmed_flush_lsn | wal_status | safe_wal_size | two_phase | conflicting
---------------+--------+-----------+--------+----------+-----------+--------+------------+------+--------------+-------------+---------------------+------------+---------------+-----------+-------------
 replica1_slot |        | physical  |        |          | f         | t      |         91 |  747 |              | 0/40473B8   |                     | reserved   |               | f         |
 replica2_slot |        | physical  |        |          | f         | t      |         90 |  747 |              | 0/40473B8   |                     | reserved   |               | f         |
(2 rows)
```

Y por último verificar el estado de la replicación

docker exec -it postgres-primary psql -U shen -d prueba -c "SELECT * FROM pg_stat_replication;"

```
pid | usesysid | usename | application_name | client_addr | client_hostname | client_port |         backend_start         | backend_xmin |   state   | sent_lsn  | write_lsn | flush_lsn | replay_ls
| write_lag | flush_lag | replay_lag | sync_priority | sync_state |          reply_time
-----+----------+---------+------------------+-------------+-----------------+-------------+-------------------------------+--------------+-----------+-----------+-----------+-----------+----------
+-----------+-----------+------------+---------------+------------+-------------------------------
  90 |    16385 | replica | walreceiver      | 172.20.0.4  |                 |       53584 | 2025-05-14 12:20:28.265836+00 |              | streaming | 0/40473B8 | 0/40473B8 | 0/40473B8 | 0/40473B8
|           |           |            |             0 | async      | 2025-05-14 12:33:12.548562+00
  91 |    16385 | replica | walreceiver      | 172.20.0.3  |                 |       53980 | 2025-05-14 12:20:32.317096+00 |              | streaming | 0/40473B8 | 0/40473B8 | 0/40473B8 | 0/40473B8
|           |           |            |             0 | async      | 2025-05-14 12:33:12.644116+00
(2 rows)
```

Se comprueba las tablas que tiene la bbdd primary y la información que tiene esa tabla

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-primary psql -U shen -d prueba -c "\dt"
          List of relations
 Schema |     Name     | Type  | Owner
--------+--------------+-------+-------
 public | test_table | table | shen
(1 row)
```

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-primary psql -U shen -d prueba -c "SELECT * FROM test_table;"
 id |      name      |          created_at
----+----------------+----------------------------
  1 | Test Record 1 | 2025-05-14 12:20:16.495343
  2 | Test Record 2 | 2025-05-14 12:20:16.495343
  3 | Test Record 3 | 2025-05-14 12:20:16.495343
(3 rows)
```

Una vez comprobado el primary, se hará lo mismo con las dos réplicas

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-replica1 psql -U shen -d prueba -c "\dt"
          List of relations
 Schema |     Name     | Type  | Owner
--------+--------------+-------+-------
 public | test_table | table | shen
(1 row)

shen@debian:~/bbdd-ha$ docker exec -it postgres-replica2 psql -U shen -d prueba -c "\dt"
          List of relations
 Schema |     Name     | Type  | Owner
--------+--------------+-------+-------
 public | test_table | table | shen
(1 row)
```

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-replica1 psql -U shen -d prueba -c "SELECT * FROM test_table;"
 id |     name      |         created_at
----+---------------+----------------------------
  1 | Test Record 1 | 2025-05-14 12:20:16.495343
  2 | Test Record 2 | 2025-05-14 12:20:16.495343
  3 | Test Record 3 | 2025-05-14 12:20:16.495343
(3 rows)

shen@debian:~/bbdd-ha$ docker exec -it postgres-replica2 psql -U shen -d prueba -c "SELECT * FROM test_table;"
 id |     name      |         created_at
----+---------------+----------------------------
  1 | Test Record 1 | 2025-05-14 12:20:16.495343
  2 | Test Record 2 | 2025-05-14 12:20:16.495343
  3 | Test Record 3 | 2025-05-14 12:20:16.495343
(3 rows)
```

Prueba de replicación

Se inserta datos en la bbdd primary

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-primary psql -U shen -d prueba -c "INSERT INTO test_table (name) VALUES ('prueba repli
ca manual'); SELECT * FROM test_table;"
INSERT 0 1
 id |         name          |         created_at
----+-----------------------+----------------------------
  1 | Test Record 1         | 2025-05-14 12:20:16.495343
  2 | Test Record 2         | 2025-05-14 12:20:16.495343
  3 | Test Record 3         | 2025-05-14 12:20:16.495343
  4 | prueba replica manual | 2025-05-14 12:38:47.794404
(4 rows)
```

Una vez insertado los datos en la primary, se comprueba en las réplicas, donde se replicará automáticamente

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-replica1 psql -U shen -d prueba -c "SELECT * FROM test_table;"
 id |         name          |         created_at
----+-----------------------+----------------------------
  1 | Test Record 1         | 2025-05-14 12:20:16.495343
  2 | Test Record 2         | 2025-05-14 12:20:16.495343
  3 | Test Record 3         | 2025-05-14 12:20:16.495343
  4 | prueba replica manual | 2025-05-14 12:38:47.794404
(4 rows)

shen@debian:~/bbdd-ha$ docker exec -it postgres-replica2 psql -U shen -d prueba -c "SELECT * FROM test_table;"
 id |         name          |         created_at
----+-----------------------+----------------------------
  1 | Test Record 1         | 2025-05-14 12:20:16.495343
  2 | Test Record 2         | 2025-05-14 12:20:16.495343
  3 | Test Record 3         | 2025-05-14 12:20:16.495343
  4 | prueba replica manual | 2025-05-14 12:38:47.794404
(4 rows)
```

Una vez comprobado que se hace bien la replica, lo que se hará es intentar escribir datos en las réplicas, donde nos retorna un error porque es de solo lectura

```
shen@debian:~/bbdd-ha$ docker exec -it postgres-replica1 psql -U shen -d prueba -c "INSERT INTO test_table (name) VALUES ('XD');"
ERROR:  cannot execute INSERT in a read-only transaction
```

4.7 HAProxy

Para probar el HAProxy, se conectará a través del puerto principal(5435)

```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5435 -U shen -d prueba -c "SELECT inet_server_addr();"
 inet_server_addr
------------------
 172.20.0.2
(1 fila)
```

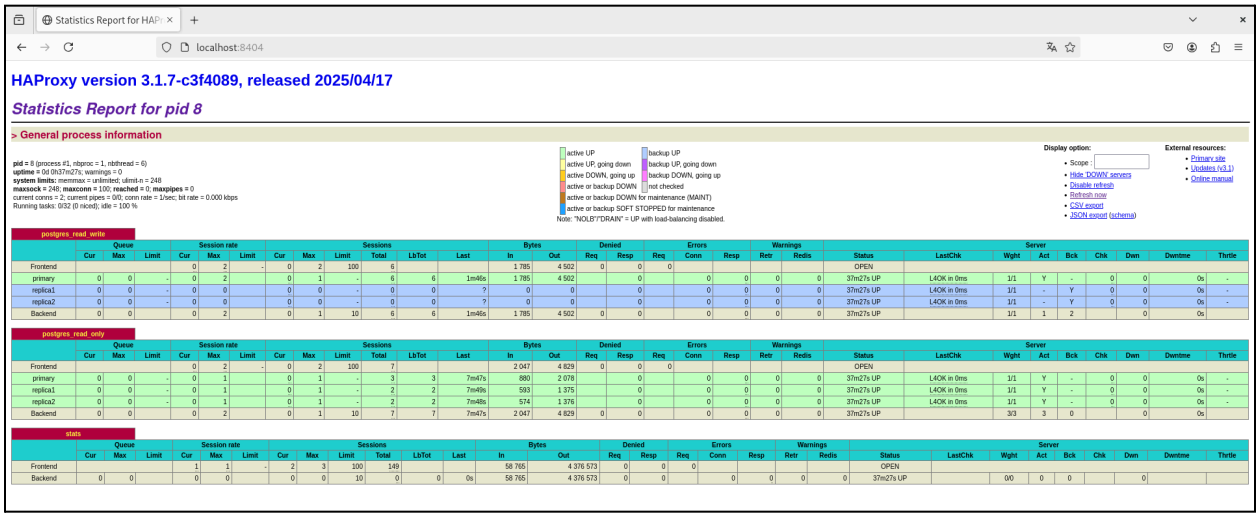
Para comprobar el balanceo de carga se harán varias consultas, estás se repartiran la carga mediante round robin

```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5436 -U shen -d prueba -c "SELECT inet_server_addr();"
 inet_server_addr
------------------
 172.20.0.2
(1 fila)

shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5436 -U shen -d prueba -c "SELECT inet_server_addr();"
 inet_server_addr
------------------
 172.20.0.3
(1 fila)

shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5436 -U shen -d prueba -c "SELECT inet_server_addr();"
 inet_server_addr
------------------
 172.20.0.4
(1 fila)

shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5436 -U shen -d prueba -c "SELECT inet_server_addr();"
 inet_server_addr
------------------
 172.20.0.2
(1 fila)
```

También se aprovecha para probar hacer un insert mediante el HAProxy y comprobar que no hay ningún problema

```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5435 -U shen -d prueba -c "INSERT INTO test_table (name) VALUES ('prueba
 manual haproxy'); SELECT * FROM test_table;"
INSERT 0 1
 id |         name          |         created_at
----+-----------------------+----------------------------
  1 | Test Record 1         | 2025-05-14 12:20:16.495343
  2 | Test Record 2         | 2025-05-14 12:20:16.495343
  3 | Test Record 3         | 2025-05-14 12:20:16.495343
  4 | prueba replica manual | 2025-05-14 12:38:47.794404
  5 | prueba manual haproxy | 2025-05-14 16:42:24.977541
(5 filas)
```

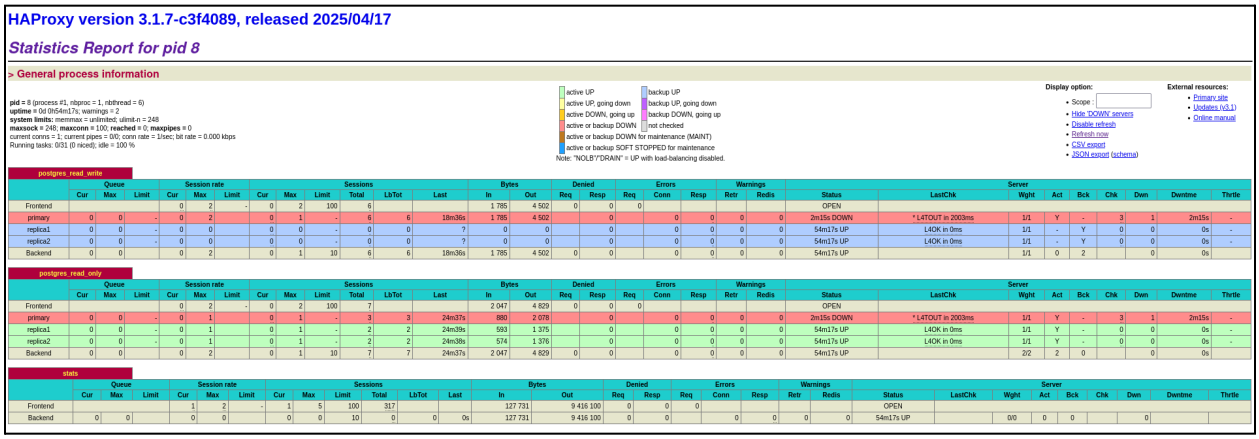
Se conecta a HAProxy mediante firefox http://localhost:8404



Para probar el failover, es simular una caída en el postgres-primary

```
shen@debian:~/bbdd-ha$ docker stop postgres-primary
postgres-primary
```

```
shen@debian:~/bbdd-ha$ docker ps -a | grep postgres
e88e3b5f27cf   quay.io/prometheuscommunity/postgres-exporter:latest   "/bin/postgres_expor…"   5 hours ago   Up 5 hours                0.0.0.0:9187->9187/tcp, :::9187->9187/tcp
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                
postgres_exporter
337f36a1e92d   postgres:16                                                "docker-entrypoint.s…"   5 hours ago   Up 5 hours (healthy)      0.0.0.0:5434->5432/tcp, :::5434->5432/tcp
postgres-replica2
5589fdae52d6   postgres:16                                                "docker-entrypoint.s…"   5 hours ago   Up 5 hours (healthy)      0.0.0.0:5433->5432/tcp, :::5433->5432/tcp
postgres-replica1
db0eecbbbaf3   postgres:16                                                "docker-entrypoint.s…"   5 hours ago   Exited (0) 2 minutes ago
postgres-primary
```



Ahora si se hace una consulta mediante el HAProxy por el puerto 5435(escritura, primary) nos redirige hacia las réplicas

```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5435 -U shen -d prueba -c "SELECT inet_server_addr(), * FROM test_table;"
 inet_server_addr | id |         name          |         created_at
------------------+----+-----------------------+----------------------------
 172.20.0.3       |  1 | Test Record 1         | 2025-05-14 12:20:16.495343
 172.20.0.3       |  2 | Test Record 2         | 2025-05-14 12:20:16.495343
 172.20.0.3       |  3 | Test Record 3         | 2025-05-14 12:20:16.495343
 172.20.0.3       |  4 | prueba replica manual | 2025-05-14 12:38:47.794404
 172.20.0.3       |  5 | prueba manual haproxy | 2025-05-14 16:42:24.977541
(5 filas)
```

Para comprobarlo de forma definitiva, sería intentar hacer una escritura al puerto 5435 de HAProxy que debería ser el primary, pero como nos redirige hacia las réplicas no debería de poder escribir en ella

```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5435 -U shen -d prueba -c "INSERT INTO test_table (name) VAL
UES ('prueba de escritura que deberia de fallar');"
ERROR:  cannot execute INSERT in a read-only transaction
```

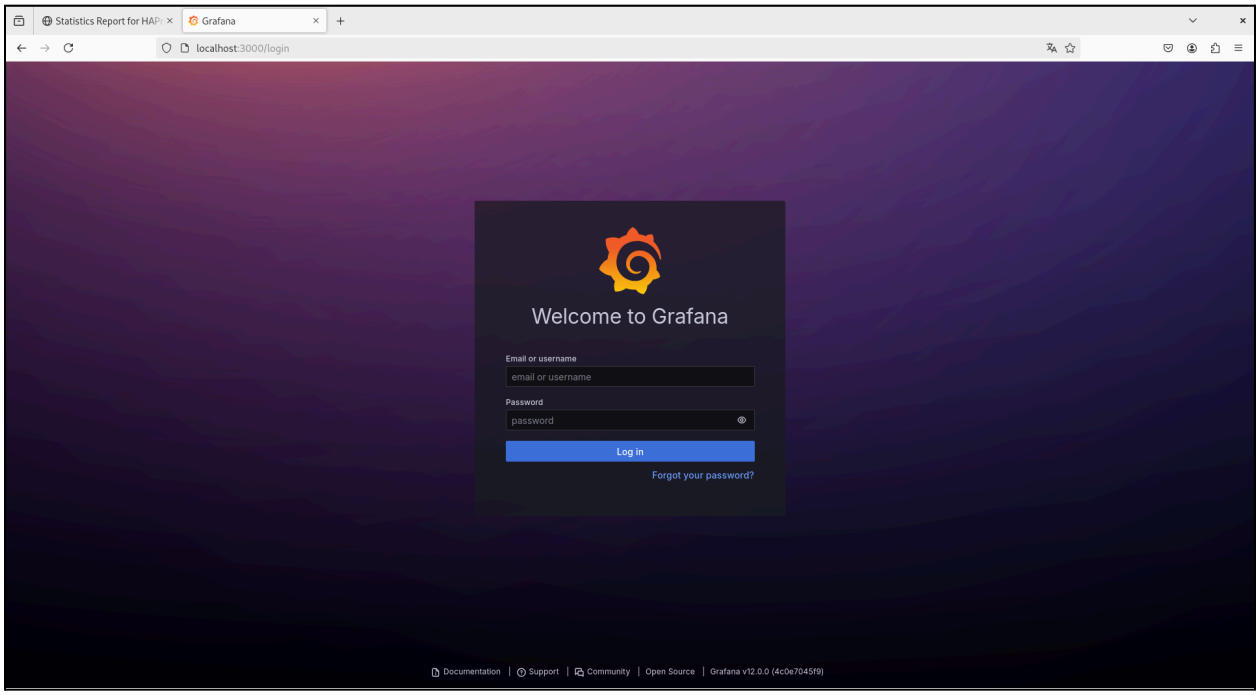
Para comprobar que funciona correctamente el primary al volver a levantarlo, lo que se hace es insertar un dato y hacer un select de ella

```
shen@debian:~/bbdd-ha$ docker start postgres-primary
postgres-primary
```

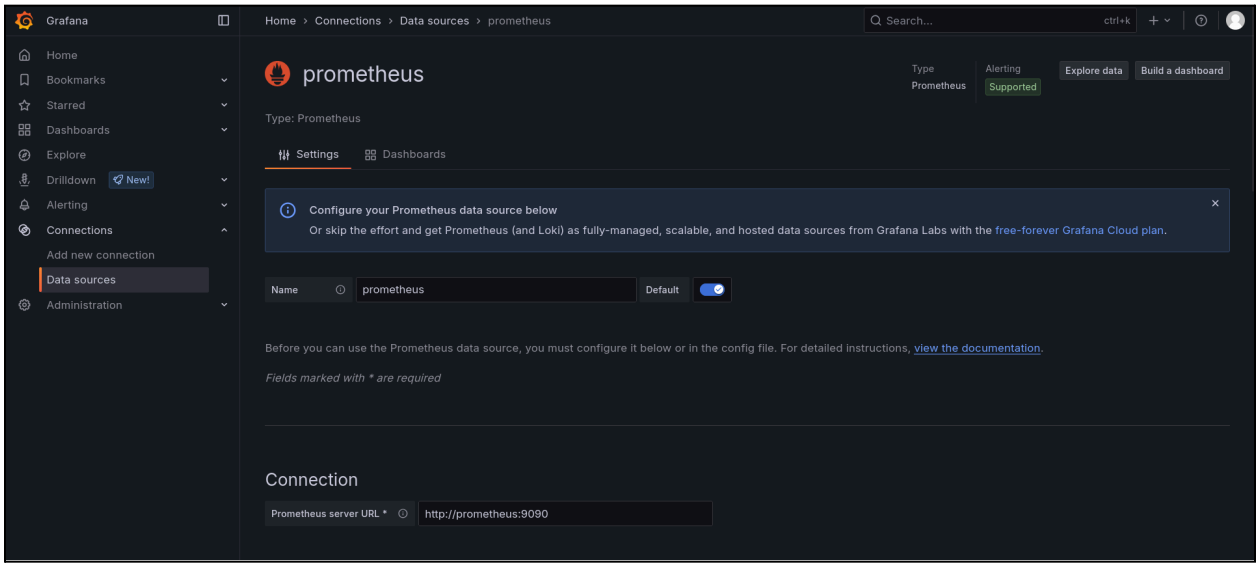
```
shen@debian:~/bbdd-ha$ PGPASSWORD=sdfsdf psql -h localhost -p 5435 -U shen -d prueba -c "INSERT INTO test_table (name) VAL
UES ('prueba de primary restaurado'); SELECT * FROM test_table ;"
INSERT 0 1
 id |             name              |         created_at
----+-------------------------------+----------------------------
  1 | Test Record 1                 | 2025-05-14 12:20:16.495343
  2 | Test Record 2                 | 2025-05-14 12:20:16.495343
  3 | Test Record 3                 | 2025-05-14 12:20:16.495343
  4 | prueba replica manual         | 2025-05-14 12:38:47.794404
  5 | prueba manual haproxy         | 2025-05-14 16:42:24.977541
  6 | prueba de primary restaurado  | 2025-05-14 17:07:37.976478
(6 filas)
```

4.8 Grafana

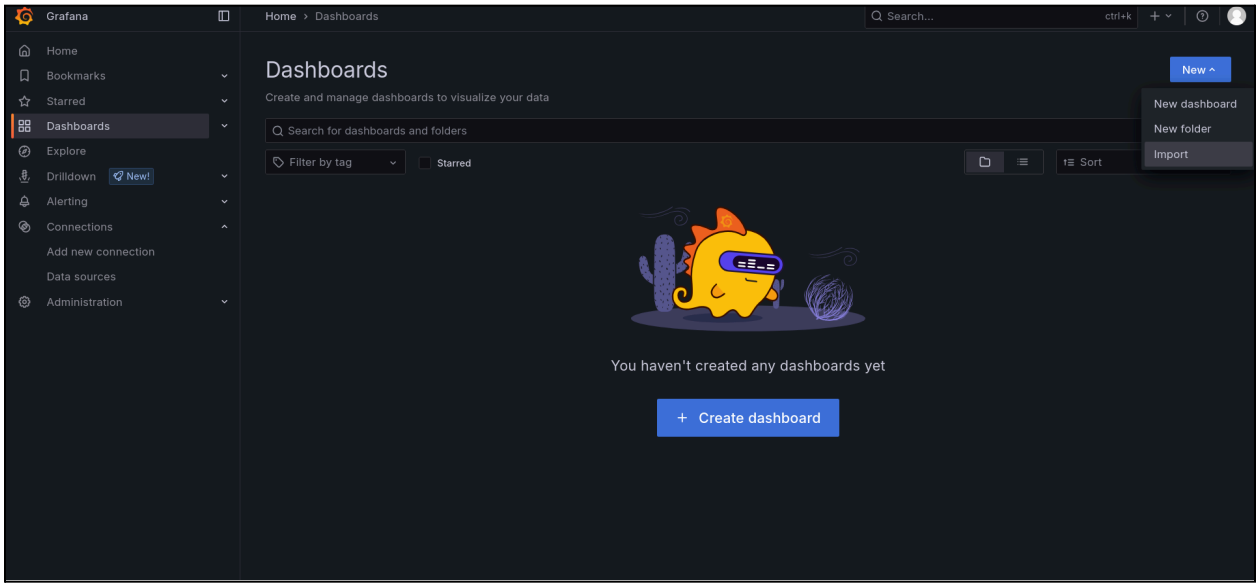
Mediante grafana se puede monitorear el cómo se está desarrollando el bbdd haciendo conexión con prometheus, se tendrá que ir a un navegador de web y entrar en la dirección de http://localhost:3000 , el usuario y la contraseña es admin admin,



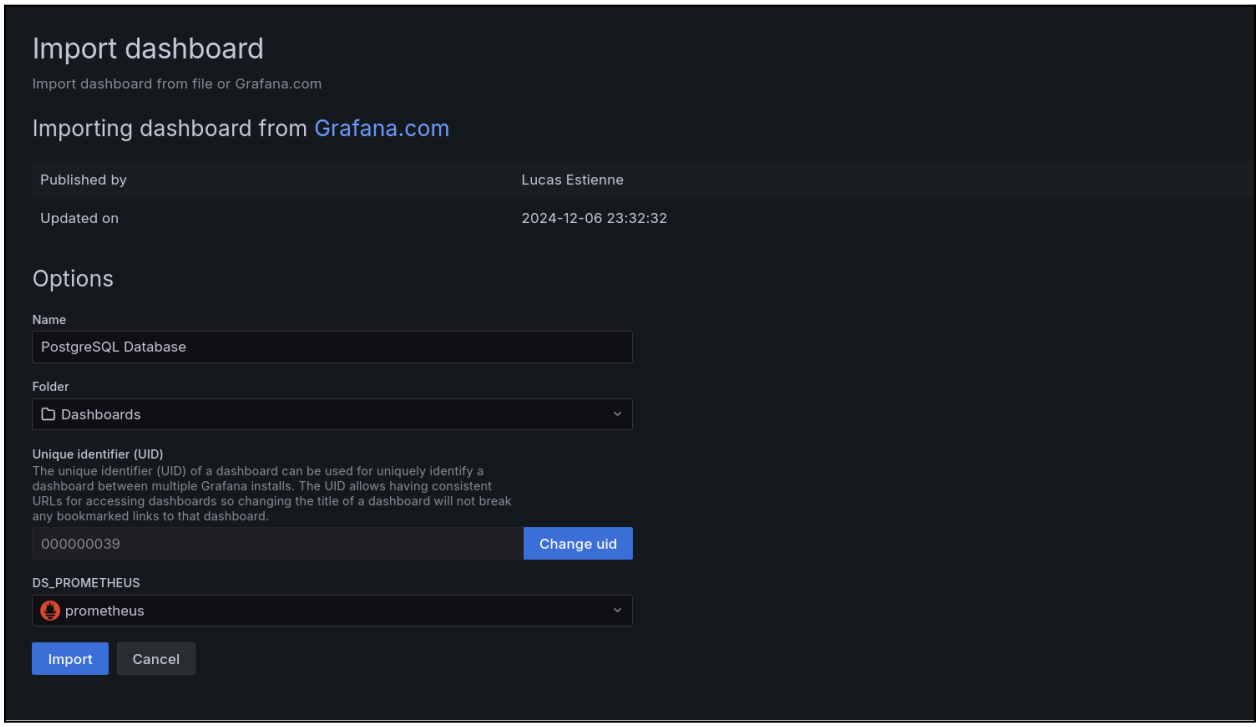
Para configurar la fuente de datos de prometheus lo que se hará es ir a connections > Data sources > prometheus y luego en connection añadir la dirección de prometheus http://prometheus:9090 y guardar la configuración



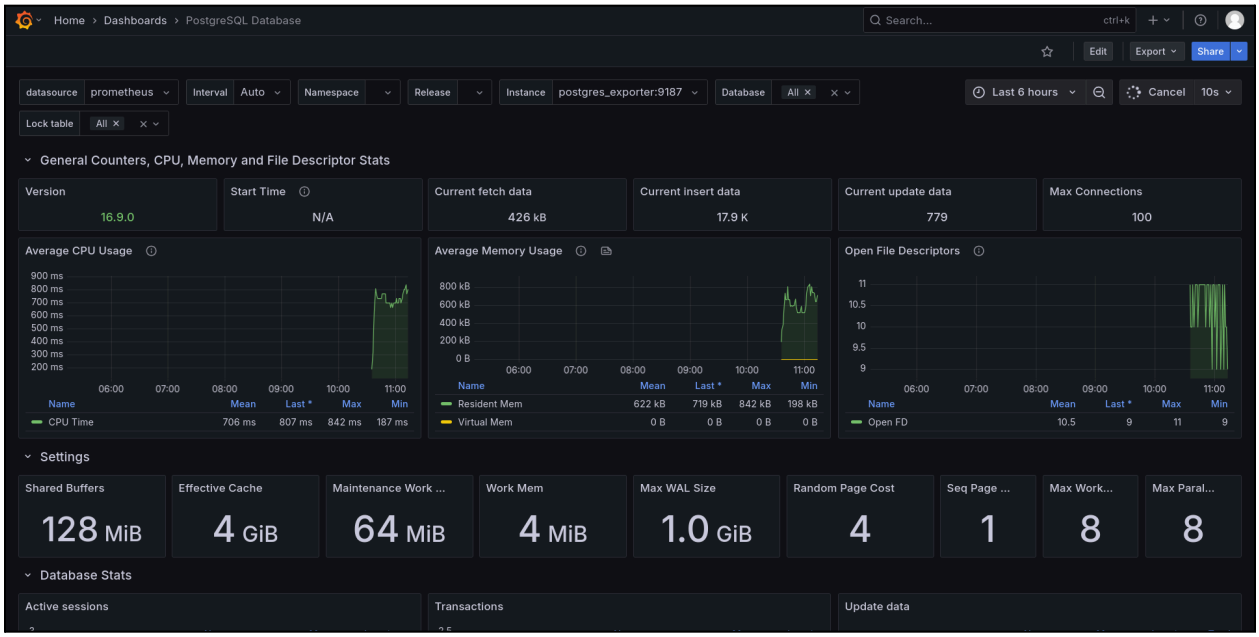
Para tener una dashboard que nos muestre de forma gráfica los distintos datos, nos digimos a Dashboards > New > Import



Para añadir un dashboard se puede ir a https://grafana.com/grafana/dashboards/ para ver los distintos tipos y buscar el que se necesite, en este caso para PostgreSQL (9628) , se introduce el id y selecciona la fuente de prometheus



Una vez completado todo, al ir a Dashboards se podrá ver las distintas métricas de la bbdd, como uso de cpu, memoria, select, insert etc de la bbdd



5. CONCLUSIONES

La implementación del sistema de PostgreSQL con alta disponibilidad utilizando tecnologías de código abierto me ha permitido desarrollar y aprender los siguientes componentes:

- Virtualización con VMware: Se implementó una máquina virtual con Debian 12 que sirve como base para todo el sistema
- Docker Compose: Se implementó múltiples contenedores coordinados mediante scripts (docker-compose.yml, init-primary.sh, init-replica.sh) que permiten el despliegue automatizado de una base de datos distribuida en réplica con alta disponibilidad
- Base de datos PostgreSQL: Se configuró un sistema master-slave con replicación asincrónica que garantiza la consistencia de datos entre el nodo primario y las réplicas
- Balanceador de carga HAProxy: Se implementó un sistema de balanceo que optimiza el rendimiento distribuyendo las consultas y un failover para garantizar la disponibilidad del servicio
- Sistema de monitorización: Se estableció un stack completo de monitorización con Prometheus, Postgres Exporter y Grafana que permite el análisis en tiempo real del rendimiento del sistema

BIBLIOGRAFÍA

PostgreSQL Global Development Group. (2024). PostgreSQL 16 documentation.
https://www.postgresql.org/docs/16/

Docker Inc. (2024). Docker documentation.

https://docs.docker.com/

Docker Inc. (2024). Docker Compose documentation.

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

HAProxy Technologies. (2024). HAProxy configuration manual - Version 2.8.
http://docs.haproxy.org/2.8/configuration.html

Prometheus Authors. (2024). Prometheus documentation.

https://prometheus.io/docs/

Grafana Labs. (2024). Grafana documentation.

https://grafana.com/docs/grafana/latest/

Docker Inc. (2024). Install Docker Engine on Debian.
https://docs.docker.com/engine/install/debian/

Dimensiona. (2024). ¿Qué es Docker y cuáles son sus ventajas?
https://www.dimensiona.com/es/que-es-docker-y-cuales-son-sus-ventajas/

Kinsta. (2024). PostgreSQL replicación: Guía completa para principiantes.
https://kinsta.com/es/blog/postgresql-replicacion/

PostgreSQL Configuration. (2024). PostgreSQL configuration parameters.
https://postgresqlco.nf/doc/en/param/

Todo PostgreSQL. (2024). Replicación PostgreSQL.
https://www.todopostgresql.com/replicacion-postgresql/

VMware Inc. (2024). VMware Workstation and Fusion desktop hypervisor products.
https://www.vmware.com/products/desktop-hypervisor/workstation-and-fusion

6. MANUAL DE USUARIO

Introducción

Este es un manual de instrucciones que está dirigido a los usuarios que necesiten interactuar con el sistema de PostgreSQL de alta disponibilidad.

Será un manual básico para hacer las distintas funciones como conectarse a la bbdd, realizar consultar, gestionar datos a través de HAProxy

Este manual está dirigido a los usuarios finales que necesitan interactuar con el sistema PostgreSQL de alta disponibilidad desplegado. Describe las operaciones básicas para conectarse a la base de datos, realizar consultas y gestionar datos a través de la interfaz de HAProxy.

URLs y Puertos

Aquí dejo un esquema de los distintos servicio que hay además de su URL/puerto y una pequeña descripción rápida

Servicio	URL/Puerto	Descripción
PostgreSQL Primario	5432	Acceso directo al nodo primario
PostgreSQL Réplica 1	5433	Acceso directo a la réplica 1
PostgreSQL Réplica 2	5434	Acceso directo a la réplica 2
HAProxy Lectura/Escritura	5435	Punto de entrada para operaciones de lectura/escritura
HAProxy Solo Lectura	5436	Punto de entrada para operaciones de solo lectura (balanceado)
HAProxy Stats	http://localhost:8404	Panel de estadísticas de HAProxy
Prometheus	http://localhost:9090	Interfaz de Prometheus
Grafana	http://localhost:3000	Dashboards de monitorización
Postgres Exporter	http://localhost:9187/metrics	Métricas de PostgreSQL en formato Prometheus

Conexión a la Base de Datos

El sistema tiene dos puntos de acceso principales a través de HAProxy:

- **Puerto 5435:** Para operaciones de lectura/escritura (conecta al nodo primario)
- **Puerto 5436:** Para operaciones de solo lectura (balanceo entre todos los nodos)

Para conectarnos utilizare el cliente de PostgreSQL

Conexión para operaciones de lectura/escritura

PGPASSWORD=sdfsdf psql -h <dirección_ip_servidor> -p 5435 -U shen -d prueba

Conexión para operaciones de solo lectura

PGPASSWORD=sdfsdf psql -h <dirección_ip_servidor> -p 5436 -U shen -d prueba

Los parámetros se pueden cambiar en el archivo Docker Compose y significa :

-h: <dirección_ip_servidor> es la dirección IP del servidor donde está instalado HAProxy

-p: 5435/5436 es el puerto para escritura o lectura

-U: shen es el nombre de usuario predeterminado

PGPASSWORD: sdfsdf es la contraseña predeterminada

-d: prueba es el nombre de la base de datos

Operaciones Básicas

Select Datos

Para realizar consultas de select :

Consultar todos los registros de la tabla test_table (o cualquier otra tabla)

SELECT * FROM test_table;

Consultar con filtros

SELECT * FROM test_table WHERE id > 10;

Consultar con ordenamiento

SELECT * FROM test_table ORDER BY created_at DESC;

Insertar Datos

Para insert de nuevos datos (solo a través del puerto 5435):

Insertar un solo registro

INSERT INTO test_table (name) VALUES ('Nuevo registro');

Insertar múltiples registros

INSERT INTO test_table (name) VALUES ('Registro 1'), ('Registro 2'), ('Registro 3');

Update Datos

Para update de datos existentes (sólo a través del puerto 5435):

Actualizar un registro

UPDATE test_table SET name = 'Nombre actualizado' WHERE id = 1;

Actualizar múltiples registros

UPDATE test_table SET name = 'Actualización masiva' WHERE id BETWEEN 5 AND 10;

Delete Datos

Para delete de datos (solo a través del puerto 5435):

Eliminar un registro específico

DELETE FROM test_table WHERE id = 2;

Eliminar múltiples registros

DELETE FROM test_table WHERE created_at < '2023-01-01';

Hay muchas formas diferentes de poder hacer las operaciones de sql, recomiendo ir al manual oficial de postgresql ante cualquier duda ya que esto es una pequeña guia basica de sentencias sql

https://www.postgresql.org/files/documentation/pdf/16/postgresql-16-A4.pdf

Panel de Estadísticas de HAProxy

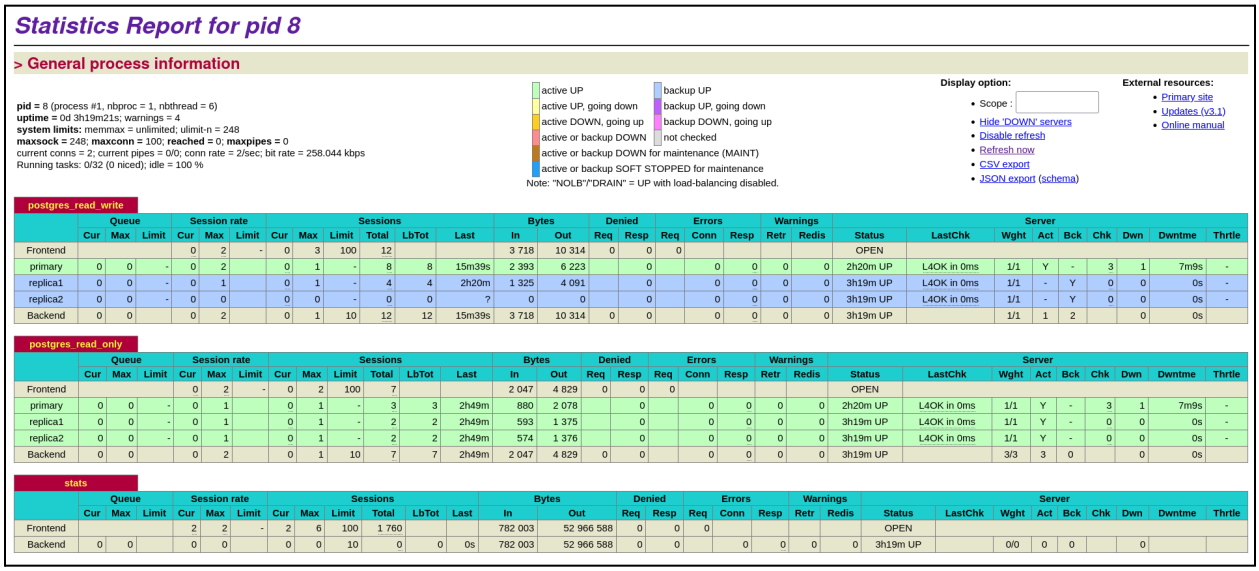
Para acceder al panel de estadísticas de HAProxy:

Abre un navegador web (Firefox, Chrome, Edge) utilizando la dirección:

http://<dirección_ip_servidor>:8404

Este panel muestra el estado de todos los servidores PostgreSQL, mostrando información como:

Estado actual (UP/DOWN), tiempos de respuesta, número de conexiones, balanceo de carga



Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 6)
uptime = 0d 3h19m21s; warnings = 4
system limits: memmax = unlimited; ulimit-n = 248
maxsock = 248; **maxconn =** 100; **reached =** 0; **maxpipes =** 0
current conns = 2; current pipes = 0/0; conn rate = 2/sec; bit rate = 258.044 kbps
Running tasks: 0/32 (0 niced); idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export
- JSON export (schema)

External resources:
- Primary site
- Updates (v3.1)
- Online manual

postgres_read_write

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	0	3	100	12			3 718	10 314	0	0	0					OPEN								
primary	0	0	-	0	2		0	1	-	8	8	15m39s	2 393	6 223		0		0	0	0	0	2h20m UP	L4OK in 0ms	1/1	Y	-	3	1	7m9s	-
replica1	0	0	-	0	1		0	1	-	4	4	2h20m	1 325	4 091		0		0	0	0	0	3h19m UP	L4OK in 0ms	1/1	-	Y	0	0	0s	-
replica2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	3h19m UP	L4OK in 0ms	1/1	-	Y	0	0	0s	-
Backend	0	0		0	2		0	1	10	12	12	15m39s	3 718	10 314	0	0		0	0	0	0	3h19m UP		1/1	1	2		0	0s	

postgres_read_only

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	0	2	100	7			2 047	4 829	0	0	0					OPEN								
primary	0	0	-	0	1		0	1	-	3	3	2h49m	880	2 078		0		0	0	0	0	2h20m UP	L4OK in 0ms	1/1	Y	-	3	1	7m9s	-
replica1	0	0	-	0	1		0	1	-	2	2	2h49m	593	1 375		0		0	0	0	0	3h19m UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
replica2	0	0	-	0	1		0	1	-	2	2	2h49m	574	1 376		0		0	0	0	0	3h19m UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
Backend	0	0		0	2		0	1	10	7	7	2h49m	2 047	4 829	0	0		0	0	0	0	3h19m UP		3/3	3	0		0	0s	

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				2	2	-	2	6	100	1 760			782 003	52 966 588	0	0	0					OPEN								
Backend	0	0		0	0		0	0	10	0	0	0s	782 003	52 966 588	0	0		0	0	0	0	3h19m UP		0/0	0	0		0		

Grafana

Para acceder a los dashboards de Grafana:

Abre un navegador web (Firefox, Chrome, Edge) utilizando la dirección:

http://<dirección_ip_servidor>:3000

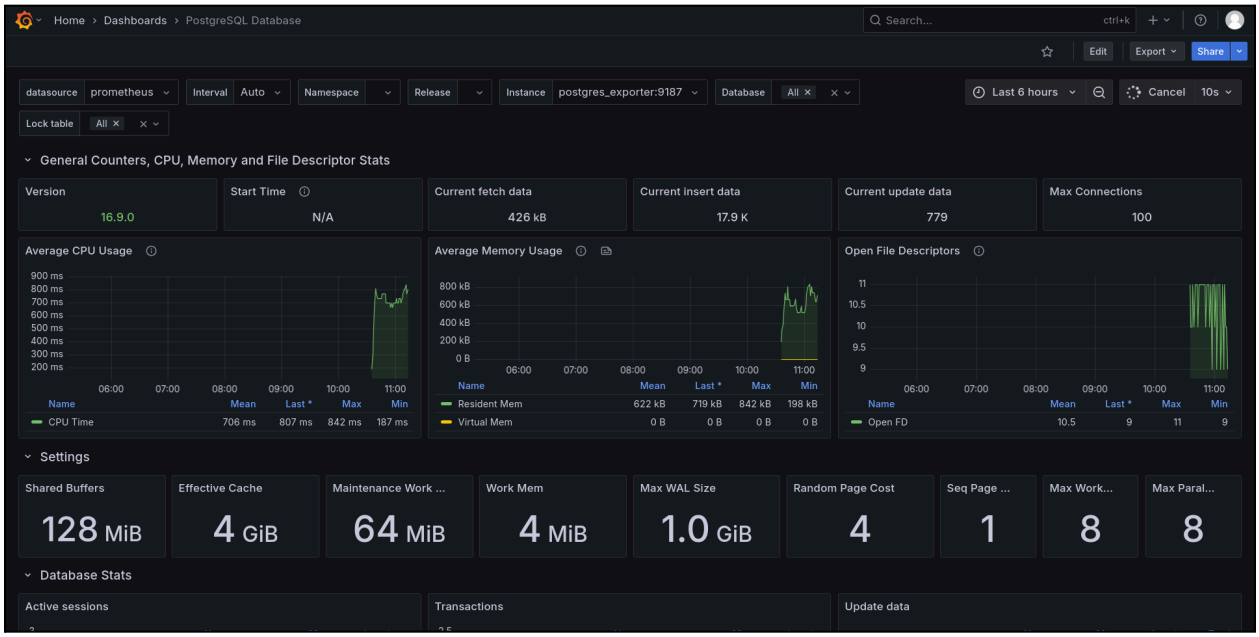
Inicie sesión con(valores definidos en el docker compose):

Usuario: admin

Contraseña: admin

Los dashboards preconfigurados incluyen (ID: 9628):

Rendimiento general de PostgreSQL, estadísticas de replicación, uso de recursos del sistema, métricas de HAProxy



7. MANUAL DE INSTALACIÓN

Despliegue de infraestructura

Para hacer el despliegue de forma más fácil sin tener que crear los archivos uno a uno, lo que haré es clonar un repositorio de github con todo el proyecto y sus scripts, además de poder personalizar las distintas configuraciones

https://github.com/a2158068171/Proyecto-TFG-ASIR.git

```
shen@debian:~$ git clone https://github.com/a2158068171/Proyecto-TFG-ASIR.git
Clonando en 'Proyecto-TFG-ASIR'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 25 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
Recibiendo objetos: 100% (25/25), 22.70 KiB | 540.00 KiB/s, listo.
Resolviendo deltas: 100% (6/6), listo.
shen@debian:~$ cd Proyecto-TFG-ASIR
shen@debian:~/Proyecto-TFG-ASIR$ ls
configs  docker-compose.yml  LICENSE  README.md  scripts
```

Creo el directorio data para las distintas los contenedores primary, replica1 y replica2

```
shen@debian:~/Proyecto-TFG-ASIR$ mkdir -p data/{primary,replica1,replica2}
shen@debian:~/Proyecto-TFG-ASIR$ ls
configs  data  docker-compose.yml  LICENSE  README.md  scripts
```

Otorgo permisos de ejecución para los scripts y docker-compose

```
shen@debian:~/Proyecto-TFG-ASIR$ chmod +x scripts/init-primary.sh scripts/init-replica.sh docker-compose.yml
```

Una vez que hecho todo, lo que haré es iniciar los distintos contenedores en 4 fases para que no haya error

```
shen@debian:~/Proyecto-TFG-ASIR$ docker-compose up -d postgres-primary
Creating network "proyecto-tfg-asir_postgres_network" with driver "bridge"
Creating volume "proyecto-tfg-asir_prometheus_data" with default driver
Creating volume "proyecto-tfg-asir_grafana_data" with default driver
Creating postgres-primary ... done
shen@debian:~/Proyecto-TFG-ASIR$ docker-compose up -d postgres-replica1 postgres-replica2
postgres-primary is up-to-date
Creating postgres-replica1 ... done
Creating postgres-replica2 ... done
shen@debian:~/Proyecto-TFG-ASIR$ docker-compose up -d haproxy
postgres-primary is up-to-date
postgres-replica1 is up-to-date
postgres-replica2 is up-to-date
Creating haproxy ... done
shen@debian:~/Proyecto-TFG-ASIR$ docker-compose up -d prometheus grafana postgres_exporter
Creating prometheus        ... done
Creating postgres_exporter ... done
Creating grafana           ... done
```

Una vez levantado todo los contenedores compruebo que está en funcionamiento

```
shen@debian:~/Proyecto-TFG-ASIR$ docker-compose ps -a
      Name                     Command                  State                                          Ports
-----------------------------------------------------------------------------------------------------------------------
grafana             /run.sh                          Up             0.0.0.0:3000->3000/tcp,:::3000->3000/tcp
haproxy             docker-entrypoint.sh hapro ...   Up             0.0.0.0:5435->5435/tcp,:::5435->5435/tcp,
                                                                    0.0.0.0:5436->5436/tcp,:::5436->5436/tcp,
                                                                    0.0.0.0:8404->8404/tcp,:::8404->8404/tcp
postgres-primary    docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5432->5432/tcp,:::5432->5432/tcp
postgres-replica1   docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5433->5432/tcp,:::5433->5432/tcp
postgres-replica2   docker-entrypoint.sh postgres    Up (healthy)   0.0.0.0:5434->5432/tcp,:::5434->5432/tcp
postgres_exporter   /bin/postgres_exporter           Up             0.0.0.0:9187->9187/tcp,:::9187->9187/tcp
prometheus          /bin/prometheus --config.f ...   Up             0.0.0.0:9090->9090/tcp,:::9090->9090/tcp
```