P3. Cluster MySQL

ÍNDICE

INTRODUCCIÓN

La creación de un clúster de bases de datos con servidores MySQL replicados es fundamental para garantizar la alta disponibilidad y la resiliencia de los datos. Este proceso implica configurar distintos nodos: un nodo de administración (MGM) que gestiona el clúster, nodos de datos NDB que almacenan la información de forma distribuida, y nodos SQL que permiten el acceso a esos datos. A través de pruebas y scripts en Python, se podrá validar el funcionamiento del clúster, mientras que la incorporación de HAProxy como balanceador de carga optimiza la gestión del tráfico.

APARTADO 1

Mi elección ha sido Galera junto con MariaDB ya que proporcionan alta disponibilidad, una replicación síncrona y escalabilidad.

Alta Disponibilidad (HA)

- Todos los nodos están activos y aceptan lecturas y escritura por lo cual garantiza la redundancia. Si uno de los nodos falla, el otro sigue activo.
- También funciona como failover, es decir, si uno de los nodos dejan de funcionar busca otros nodos activas para mantener redundancia.

Replicación Síncrona y Multimaster

- Galera hace que las transacciones se repliquen de manera síncrona entre nodos, es decir, los cambios entre nodos se hacen al mismo tiempo.
- Todos los nodos pueden recibir tanto consultas de lectura como operaciones de escritura simultáneamente, lo cual mejora el rendimiento.

Escalabilidad Horizontal

- Las consultas se reparten entre los nodos, lo que reduce la carga de los nodos.

Seguridad y Tolerancia a fallos

- Galera hace que los conflictos de escritura se detecten en el momento. Por ejemplo, si dos nodos intentan actualizar el mismo dato, uno de los cambios no se hacen.
- Cuando uno de los nodos caídos se reincorpora, puede recuperar todos los datos que perdió.

Compatibilidad con MariaDB

- Galera está completamente integrado con MariaDB, lo que facilita su configuración.

Fácil Implementación y Administración

- Galera requiere menos configuraciones comparado con otras tecnologías de replicación. Es fácil la configuración y añadir o eliminar nodos.

Consistencia y Seguridad de Datos

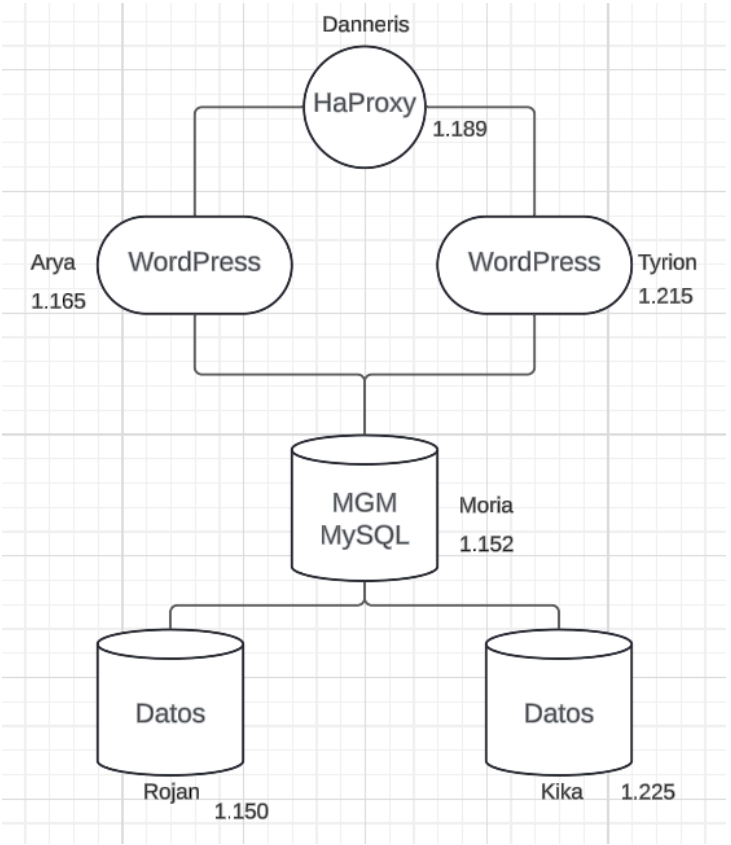
- Las transacciones son ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que asegura que los datos no son inconsistentes.
- Todos los nodos deben mantener el mismo esquema para evitar problemas de replicación.

Balanceo de Carga

- En Galera se puede integrar con un HaProxy para gestionar el balanceo de carga entre los nodos.

APARTADO 2

Creación de un cluster mysql. Mi infraestructura se basa en 4 máquinas una de nodo de administración, dos nodos de datos y otro de MySQL.



COSAS A TENER EN CUENTA

Antes de nada tenemos que tener bien configuradas las tarjetas de red para que nuestros nodos se puedan ver.

Cada vez que **APAGUEMOS** las máquinas los dos **nodos de datos** se van a poner en modo seguro, por lo cual, cada vez que iniciemos tendremos que poner la **variables a 1**.
Luego en el **nodo MGM** tendremos que hacer un **galera_new_cluster** y cuando nos vamos a los nodos datos iniciamos el mariadb (**systemctl start mariadb**) y ya tendríamos montado el clúster de nuevo.

MONTAJE CLÚSTER

Todo este proceso se hace en los 3 nodos.

1.- Tenemos que hacer la instalación de Galera. Antes de nada vamos a actualizar el sistema. Luego si instalamos MariaDB tiene soporte para Galera por lo cual instalaremos el siguiente paquete *sudo apt install mariadb-server (**Este paquete se instalará en todos los nodos**)*.

```
root@Moria:/home/frodo# sudo apt install mariadb-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de f
son necesarios.
  cpp-10 gedit-plugin-commander gedit-plugin-find-in-fil
  gedit-plugin-translate guile-2.2-libs libappstream-gli
  libavfilter7 libavformat58 libavresample4 libcbor0 lib
  libdbus-glib-1-2 libdpkg-perl libextutils-pkgconfig-pe
  libfile-fcntllock-perl libfluidsynth2 libfwupdplugin1
  libigdgmm11 libilmbase25 libjim0.79 libjuh-java libjur
  liblibreoffice-java libllvm11 liblua5.2-0 libmalconten
  libmbedcrypto3 libmbedtls12 libmbedx509-0 libmms0 libm
```

2.- Ahora tenemos que habilitar el firewall para permitir las conexiones entre nodos. En mi caso tengo instalado el **ufw** por lo cual configurare las reglas para permitir las conexiones por los puertos **3306** (conexiones MySQL), **4567** (comunicación de Galera entre nodos), **4568** (para la replicación de flujo de estado) y **4444** (sincronización de estado).

```
root@Moria:/home/frodo# sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:3306
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:4567
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:4568
    0     0 ACCEPT     6    --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:4444

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@Moria:/home/frodo#
```

3.- Guardamos las reglas de firewall.

```
root@Moria:/home/frodo# sudo iptables-save > /etc/iptables/rules.v4
root@Moria:/home/frodo#
```

4.- Para verificar el estado de las reglas ufw.

```
root@Moria:/home/frodo# sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
80/tcp                     ALLOW       Anywhere
8888/tcp                   ALLOW       Anywhere
3306/tcp                   ALLOW       Anywhere
4567/tcp                   ALLOW       Anywhere
4568/tcp                   ALLOW       Anywhere
4444/tcp                   ALLOW       Anywhere
80/tcp (v6)                ALLOW       Anywhere (v6)
8888/tcp (v6)              ALLOW       Anywhere (v6)
3306/tcp (v6)              ALLOW       Anywhere (v6)
4567/tcp (v6)              ALLOW       Anywhere (v6)
4568/tcp (v6)              ALLOW       Anywhere (v6)
4444/tcp (v6)              ALLOW       Anywhere (v6)

root@Moria:/home/frodo#
```

5.- También tendremos que irnos al fichero **/etc/mysql/mariadb.conf.d/50-server.cnf** y configurar/añadir las siguientes líneas.

```
[server]

# this is only for the mysqld standalone daemon
[mysqld]
bind-address = 0.0.0.0

#Galera settings
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://192.168.1.152,192.168.1.150,192.168.1.225"

wsrep_node_address="192.168.1.152"
wsrep_node_name="mqm_cluster"

wsrep_sst_method=rsync

#MariaDB settings
binlog_format=row
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2

#
# * Basic Settings
#
```

6.- Una vez configurado el fichero hacemos un **systemctl stop mariadb**, luego creamos nuestro clúster con **galera_new_cluster**, iniciamos de nuevo el servicio de MariaDB con el comando **systemctl start mariadb** y finalmente accedemos a nuestro mysql con **mariadb -u root -p**.

```
root@Moria:/home/frodo# systemctl stop mariadb
root@Moria:/home/frodo# galera_new_cluster
root@Moria:/home/frodo# systemctl start mariadb
root@Moria:/home/frodo# mariadb -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> show status like ‘wsrep_cluster_size’
    ->
    -> ;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaD
tax to use near ''wsrep_cluster_size'' at line 1
MariaDB [(none)]> show status like 'wsrep_cluster_size'
    -> ;
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 1     |
+--------------------+-------+
1 row in set (0,000 sec)

MariaDB [(none)]>
```

7.- Si iniciamos uno de los nodos, y ejecutamos el comando **show status like 'wsrep_cluster_size';** para ver que se ha levantado.

```
MariaDB [(none)]> show status like 'wsrep_cluster_size'
    -> ;
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 2     |
+--------------------+-------+
1 row in set (0,001 sec)
```

Si iniciamos el otro.

```
MariaDB [(none)]> show status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0,001 sec)

MariaDB [(none)]>
```

8.- Hay que configurar el siguiente fichero para que al iniciar las máquinas el clúster no se caiga cada vez que apaguemos las máquinas. [Mirar apartado Cosas a tener en cuenta].

```
 GNU nano 7.2              /var/lib/mysql/grastate.dat *
# GALERA saved state
version: 2.1
uuid:    933a403f-87f6-11ef-b939-036c7ba15c34
seqno:   -1
safe_to_bootstrap: 1
```

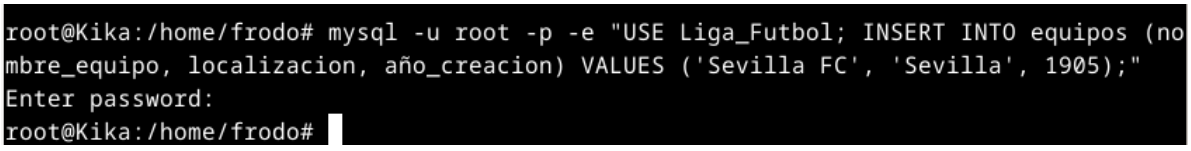
9.- Vamos a crear una base de datos.

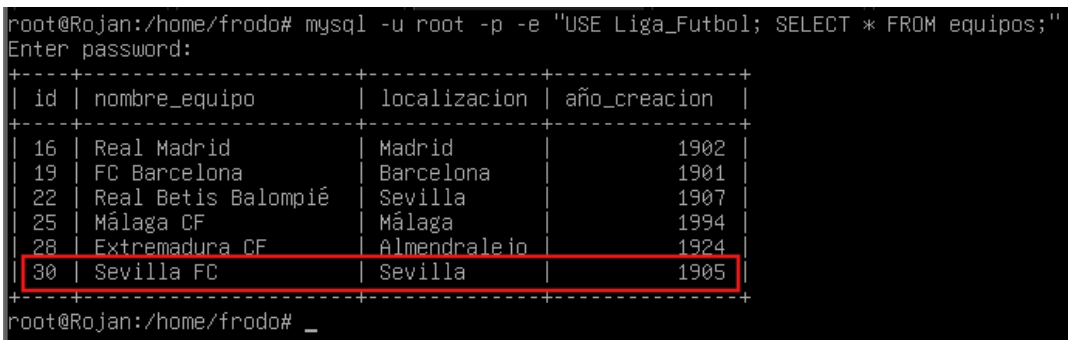
```
MariaDB [Liga_Futbol]> show tables;
+------------------------+
| Tables_in_Liga_Futbol |
+------------------------+
| equipos                |
+------------------------+
1 row in set (0,000 sec)

MariaDB [Liga_Futbol]> select * from equipos;
+----+----------------------+--------------+---------------+
| id | nombre_equipo        | localizacion | año_creacion  |
+----+----------------------+--------------+---------------+
| 16 | Real Madrid          | Madrid       |          1902 |
| 19 | FC Barcelona         | Barcelona    |          1901 |
| 22 | Real Betis Balompié  | Sevilla      |          1907 |
| 25 | Málaga CF            | Málaga       |          1994 |
| 28 | Extremadura CF       | Almendralejo |          1924 |
+----+----------------------+--------------+---------------+
5 rows in set (0,000 sec)

MariaDB [Liga_Futbol]>
```

APARTADO 3

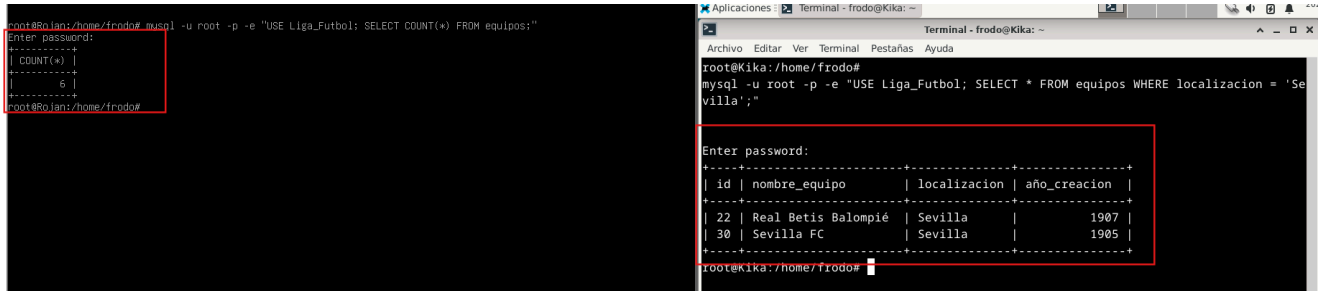
1.- Para verificar el estado del clúster podemos usar el siguiente comando **mysql -u <usuario> -p -e "SHOW STATUS LIKE 'wsrep_%';"** que nos da un montón de información.

2.- Ahora desde uno de los nodos nos vamos a conectar a la base de datos y vamos a insertar un nuevo equipo y desde otro nodo miramos los equipos y comprobamos que se ha actualizado.





3.- Ahora para comprobar que se pueden hacer consultas simultáneas vamos abrir dos terminales y vamos hacer consultas a la vez.



4.- Prueba de si se cae uno de los nodos el otro sigue en funcionamiento y cuando lo levantemos tiene todos los datos correspondientes.

```
root@Rojan:/home/frodo# systemctl stop mariadb
root@Rojan:/home/frodo#
```

```
root@Kika:/home/frodo# mysql -u root -p -e "USE Liga_Futbol; SELECT * FROM equipos;"
Enter password:
+----+----------------------+--------------+---------------+
| id | nombre_equipo        | localizacion | año_creacion  |
+----+----------------------+--------------+---------------+
| 16 | Real Madrid          | Madrid       |          1902 |
| 19 | FC Barcelona         | Barcelona    |          1901 |
| 22 | Real Betis Balompié  | Sevilla      |          1907 |
| 25 | Málaga CF            | Málaga       |          1994 |
| 28 | Extremadura CF       | Almendralejo |          1924 |
| 30 | Sevilla FC           | Sevilla      |          1905 |
+----+----------------------+--------------+---------------+
root@Kika:/home/frodo#
```

```
root@Kika:/home/frodo# mysql -u root -p -e "USE Liga_Futbol; INSERT INTO equipos (no
mbre_equipo, localizacion, año_creacion)
VALUES ('Cádiz CF', 'Cádiz', 1910);"
Enter password:
root@Kika:/home/frodo# S
```

Levantamos el otro nodo con systemctl start mariadb y vemos la base de datos.

```
root@Rojan:/home/frodo# mysql -u root -p -e "USE Liga_Futbol; SELECT * FROM equipos;"
Enter password:
+----+----------------------+--------------+---------------+
| id | nombre_equipo        | localizacion | año_creacion  |
+----+----------------------+--------------+---------------+
| 16 | Real Madrid          | Madrid       |          1902 |
| 19 | FC Barcelona         | Barcelona    |          1901 |
| 22 | Real Betis Balompié  | Sevilla      |          1907 |
| 25 | Málaga CF            | Málaga       |          1994 |
| 28 | Extremadura CF       | Almendralejo |          1924 |
| 30 | Sevilla FC           | Sevilla      |          1905 |
| 34 | Cádiz CF             | Cádiz        |          1910 |
+----+----------------------+--------------+---------------+
root@Rojan:/home/frodo#
```

APARTADO 4

NODO DE HAPROXY DANNERIS

1.- Lo primero que haremos será actualizar el sistema, **apt update** y **apt upgrade**.

```
root@Danneris:/home/frodo# apt update
Obj:1 http://deb.debian.org/debian bookworm InRelease
Obj:2 http://security.debian.org/debian-security bookworm-security InRelease
Obj:3 http://deb.debian.org/debian bookworm-updates InRelease
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 5 paquetes. Ejecute «apt list --upgradable» para verlos.
root@Danneris:/home/frodo# apt upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Calculando la actualización... Hecho
Se actualizarán los siguientes paquetes:
  firefox-esr firefox-esr-l10n-es-ar firefox-esr-l10n-es-cl
  firefox-esr-l10n-es-es firefox-esr-l10n-es-mx
5 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 0 B/71,9 MB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

2.- Instalamos el HaProxy. Usamos el comando **sudo apt install haproxy**.

```
root@Danneris:/home/frodo# apt install haproxy
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libopentracing-c-wrapper0 libopentracing1
Paquetes sugeridos:
  vim-haproxy haproxy-doc
Se instalarán los siguientes paquetes NUEVOS:
  haproxy libopentracing-c-wrapper0 libopentracing1
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 2.125 kB de archivos.
Se utilizarán 4.628 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

3.- Verificamos la instalación del haproxy.

```
root@Danneris:/home/frodo# haproxy -v
HAProxy version 2.6.12-1+deb12u1 2023/12/16 - https://haproxy.org/
Status: long-term supported branch - will stop receiving fixes around Q2 2027.
Known bugs: http://www.haproxy.org/bugs/bugs-2.6.12.html
Running on: Linux 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-0
9-30) x86_64
root@Danneris:/home/frodo# S
```

4.- Nos vamos al fichero de configuración **/etc/haproxy/haproxy.cfg** y ponemos lo siguiente en el fichero:

```
  GNU nano 7.2                        /etc/haproxy/haproxy.cfg
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

#Define frontend
frontend mysql_cluster
    bind 0.0.0.0:9000
    mode tcp
    option tcplog
    default_backend mysql_nodes

# Define backend with the nodes to balance the load
backend mysql_nodes
    mode tcp
    balance roundrobin
    #option mysql-check user haproxy_check
    server rojan 192.168.1.150:3306 check
    server kika 192.168.1.225:3306 check

listen stats
    bind :9000
    mode http
    stats enable
    stats uri /haproxy?stats
    stats refresh 10s
    stats admin if LOCALHOST
    stats auth admin:password
```

Web

5.- Ahora nos vamos a los nodos Rojan y Kika para crear el usuario haproxy_check. Una vez dentro de las máquinas y ponemos **sudo mysql -u root -p**. También le daremos privilegios.

```
root@Kika:/home/frodo# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 36
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> CREATE USER 'haproxy_check'@'%' IDENTIFIED BY 'bolsom';
Query OK, 0 rows affected (0,003 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,002 sec)
```

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'haproxy_check'@'%';
Query OK, 0 rows affected (0,002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,002 sec)
```

Si nos fijamos cuando vamos a crear el usuario en el la otra máquina nos saldrá un error porque como tenemos las máquinas conectadas como cluster ya tenemos el usuario creado.

```
MariaDB [(none)]> CREATE USER 'haproxy_check'@'%' IDENTIFIED BY 'bolsom';
ERROR 1396 (HY000): Operation CREATE USER failed for 'haproxy_check'@'%'
```

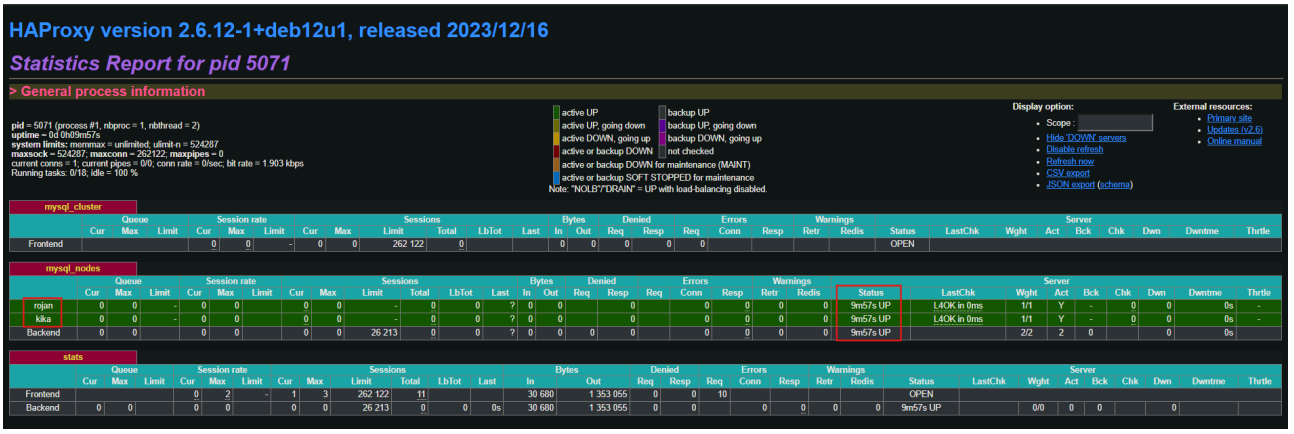
6.- Reiniciamos el servicio del haproxy, con el comando **systemctl restart haproxy**.

```
root@Danneris:/home/frodo# sudo systemctl restart haproxy
root@Danneris:/home/frodo#
```

7.- Iniciamos el haproxy con el comando **systemctl start haproxy** y luego hacemos **systemctl enable haproxy**.

```
root@Danneris:/home/frodo# sudo systemctl enable haproxy
Synchronizing state of haproxy.service with SysV service script with /lib/systemd/systemd-s
ysv-install.
Executing: /lib/systemd/systemd-sysv-install enable haproxy
root@Danneris:/home/frodo#
```

8.- Si configuramos el fichero del haproxy podemos conectarnos mediante la web poniendo la contraseña y el usuario y monitorear desde ahí nuestro clúster



9.- Ahora si nos vamos a otro equipo de la red y queremos acceder a nuestros nodos de datos a través de nuestro balanceador ponemos el comando **mysql -h <IP_HaProxy> -P <puerto_configrado> -u <usuario> -p**. Accederemos a nuestro nodo de datos y para ver a que nodo nos ha redirigido el Haproxy ejecutamos el comando **SELECT CONNECTION_ID(), @@hostname;**

```
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 56
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SELECT CONNECTION_ID(), @@hostname;
+-----------------+------------+
| CONNECTION_ID() | @@hostname |
+-----------------+------------+
|              56 | Rojan      |
+-----------------+------------+
1 row in set (0,001 sec)

MariaDB [(none)]> exit;
Bye
root@Moria:/home/frodo# mysql -h 192.168.1.189 -P 9000 -u haproxy_check -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 58
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> SELECT CONNECTION_ID(), @@hostname;
+-----------------+------------+
| CONNECTION_ID() | @@hostname |
+-----------------+------------+
|              58 | Kika       |
+-----------------+------------+
1 row in set (0,001 sec)

MariaDB [(none)]>
```

APARTADO 5

1.- Comprobaciones con Scripts de Python y/o PHP. Script para ver las bases de datos y las tablas.

```
(mi_entorno) root@Moria:/home/frodo# ./prueba_balanceador.py
Conectado a MySQL Server version 5.5.5-10.11.6-MariaDB-0+deb12u1
Estás conectado a la base de datos: Liga_Futbol
Tablas en la base de datos:
equipos
```

2.- Script que permite ver las bases de datos disponibles para un usuario y si están activos los nodos WordPress.

```
(mi_entorno) root@Moria:/home/frodo# ./balanceador.py
Verificando servidores de WordPress...
Servidor WordPress activo: http://192.168.1.165
Servidor WordPress activo: http://192.168.1.215

Verificando conexión a la base de datos...
Conexión exitosa a la base de datos.
Versión de MySQL: ('10.11.6-MariaDB-0+deb12u1',)
Bases de datos disponibles:
- db_wordpress
- information_schema
(mi_entorno) root@Moria:/home/frodo#
```

3.- Para ver el estado del clúster.

```
(mi_entorno) root@Moria:/home/frodo# ./cluster_estado.py
Conexión exitosa a la base de datos.
Tamaño del clúster: 3
No se pudo obtener el estado del clúster.
No se pudo obtener la dirección del nodo actual.
Estado de conexión de nodos: Conectado
(mi_entorno) root@Moria:/home/frodo#
```

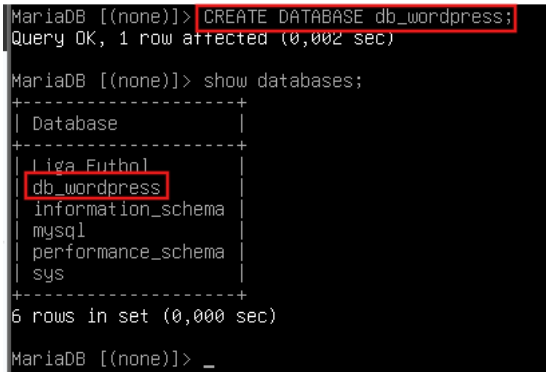
4.- Para hacer consultas a la base de datos.

```
(mi_entorno) root@Moria:/home/frodo# ./consulta_bd.py
Bienvenido al sistema de consultas de la base de datos.
Introduce tu consulta SQL (o 'salir' para terminar): select * from equipos;
Conexión exitosa a la base de datos.
Resultados de la consulta:
(16, 'Real Madrid', 'Madrid', 1902)
(19, 'FC Barcelona', 'Barcelona', 1901)
(22, 'Real Betis Balompié', 'Sevilla', 1907)
(25, 'Málaga CF', 'Málaga', 1994)
(28, 'Extremadura CF', 'Almendralejo', 1924)
(30, 'Sevilla FC', 'Sevilla', 1905)
(34, 'Cádiz CF', 'Cádiz', 1910)
Introduce tu consulta SQL (o 'salir' para terminar): salir
(mi_entorno) root@Moria:/home/frodo#
```

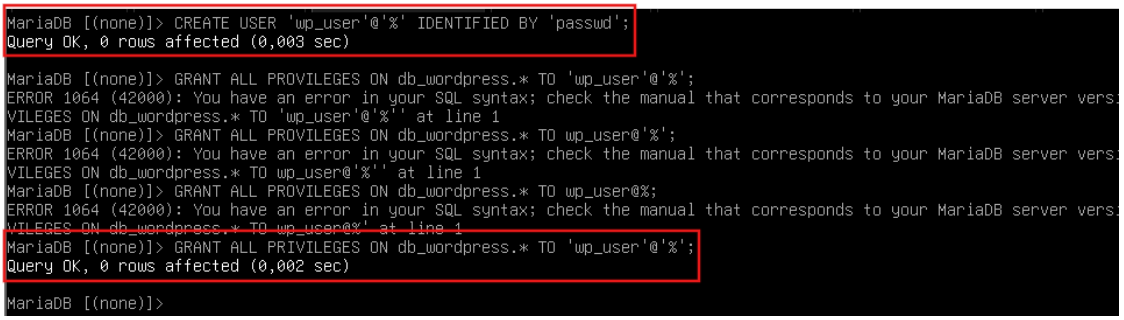
APARTADO 6

PREPARATIVOS INICIALES

1.- Accedemos con mariadb a las bases de datos y vamos a crear una base de datos específica para WordPress.

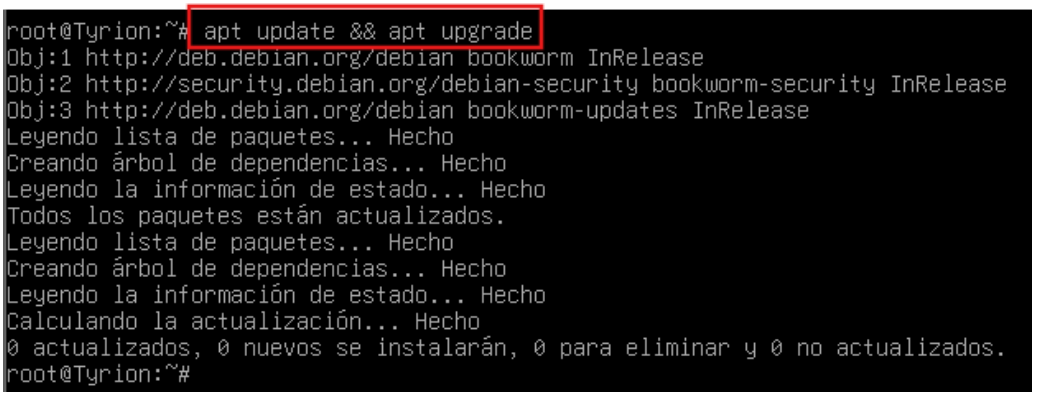


2.- Ahora vamos a crear un usuario específico que va acceder a la base de datos **db_wordpress** y le vamos a dar privilegios solo a la base de datos creada anteriormente.



NODOS WORDPRESS TYRION + ARYA

1.- Lo primero que haremos es actualizar el sistema.



2.- Una vez actualizado nuestro sistema, tendremos que instalar una pila LAMP.

```
root@Tyrion:~# apt install apache2 && apt install mariadb-server mariadb-client && ap
t install php php-mysql php-xml php-curl php-gd php-cli php-zip libapache2-mod-php
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 libgdbm-compat4 libgdbm6
  libldap-2.5-0 libldap-common liblua5.3-0 libnghttp2-14 libperl5.36 libpsl5
  librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh2-1
  media-types perl perl-modules-5.36 publicsuffix ssl-cert
Paquetes sugeridos:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom gdbm-l10n
  libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal
  libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql perl-doc
  libterm-readline-gnu-perl | libterm-readline-perl-perl make
  libtap-harness-archive-perl
Se instalarán los siguientes paquetes NUEVOS:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libcurl4 libgdbm-compat4 libgdbm6
  libldap-2.5-0 libldap-common liblua5.3-0 libnghttp2-14 libperl5.36 libpsl5
  librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libsqlite3-0 libssh2-1
  media-types perl perl-modules-5.36 publicsuffix ssl-cert
0 actualizados, 28 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 11,8 MB de archivos.
Se utilizarán 62,1 MB de espacio de disco adicional después de esta operación.
```

3.- Ahora vamos a instalar WordPress. Para ello accedemos a la ruta **/var/www/html** y ahí descargamos el paquete del WordPress.

```
root@Tyrion:/var/www/html# wget https://wordpress.org/latest.tar.gz
--2024-10-12 17:30:57--  https://wordpress.org/latest.tar.gz
Resolviendo wordpress.org (wordpress.org)... 198.143.164.252
Conectando con wordpress.org (wordpress.org)[198.143.164.252]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 24640061 (23M) [application/octet-stream]
Grabando a: «latest.tar.gz»

latest.tar.gz         100%[=====================>]  23,50M  9,68MB/s    en 2,4s

2024-10-12 17:31:00 (9,68 MB/s) - «latest.tar.gz» guardado [24640061/24640061]
```

4.- Descomprimimos el paquete **tar -xzvf latest.tar.gz**, y damos permisos a la ruta.

```
root@Tyrion:/var/www/html# chown -R www-data:www-data /var/www/html/wordpress
root@Tyrion:/var/www/html# chmod -R 755 /var/www/html/wordpress
```

5.- Ahora vamos a configurar el Apache para poder alojar WordPress. Para ello vamos a crear un fichero en la ruta **nano /etc/apache2/sites-available/wordpress.conf**. Una vez creado vamos a poner lo siguiente en el archivo:

```
  GNU nano 7.2          /etc/apache2/sites-available/wordpress.conf *
<VirtualHost *:80>
    DocumentRoot /var/www/html/wordpress
    ServerName 192.168.1.215

    <Directory /var/www/html/wordpress>
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/wordpress_error.log
    CustomLog ${APACHE_LOG_DIR}/wordpress_access.log combined
</VirtualHost>
```

6.- Habilitamos la configuración del sitio y los módulos necesarios:

```
root@Tyrion:/var/www/html# systemctl restart apache2
root@Tyrion:/var/www/html# a2ensite wordpress.conf
Site wordpress already enabled
root@Tyrion:/var/www/html# a2enmod rewrite
Module rewrite already enabled
root@Tyrion:/var/www/html#
```

7.- Ahora tenemos que configurar el archivo **wp-config.php**, por lo que copiamos el archivo de configuración:

```
root@Tyrion:/var/www/html/wordpress# cp wp-config-sample.php wp-config.php
root@Tyrion:/var/www/html/wordpress#
```

8.- Editamos el archivo wp-config.php y ponemos los datos de nuestra base de datos.

```
/** The name of the database for WordPress */
define( 'DB_NAME', 'db_wordpress' );

/** Database username */
define( 'DB_USER', 'wp_user' );

/** Database password */
define( 'DB_PASSWORD', 'passwd' );

/** Database hostname */
define( 'DB_HOST', '192.168.1.225' );
```

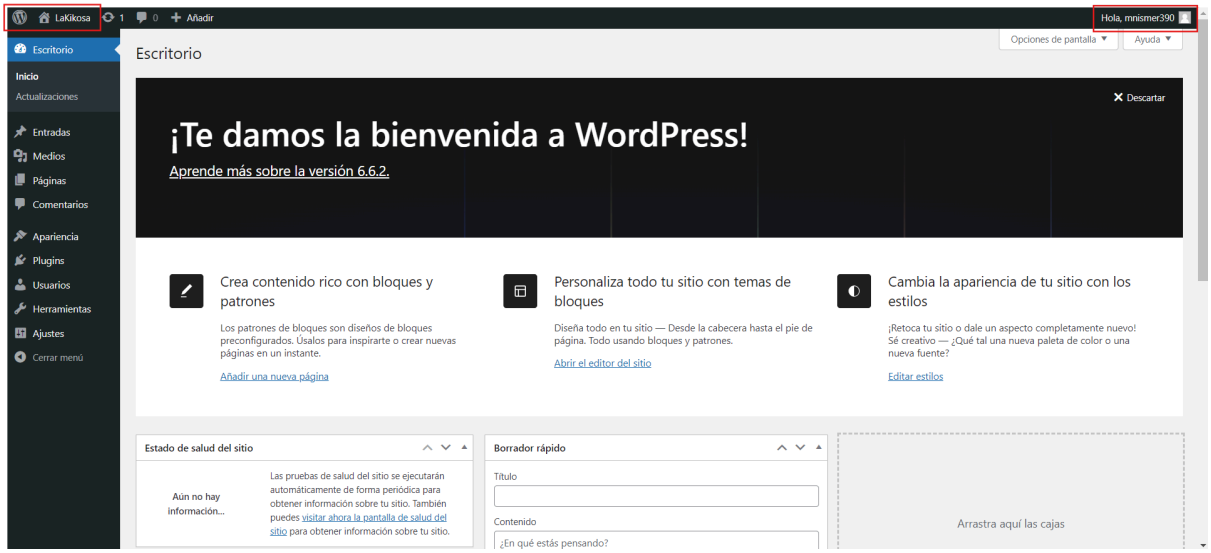
9.- Tenemos que verificar que existe el siguiente fichero **/var/www/html/wordpress/.htaccess**, en el caso de que no existe lo creamos y le añadimos lo siguiente al fichero:

```
  GNU nano 7.2                 /var/www/html/wordpress/.htaccess
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>
```

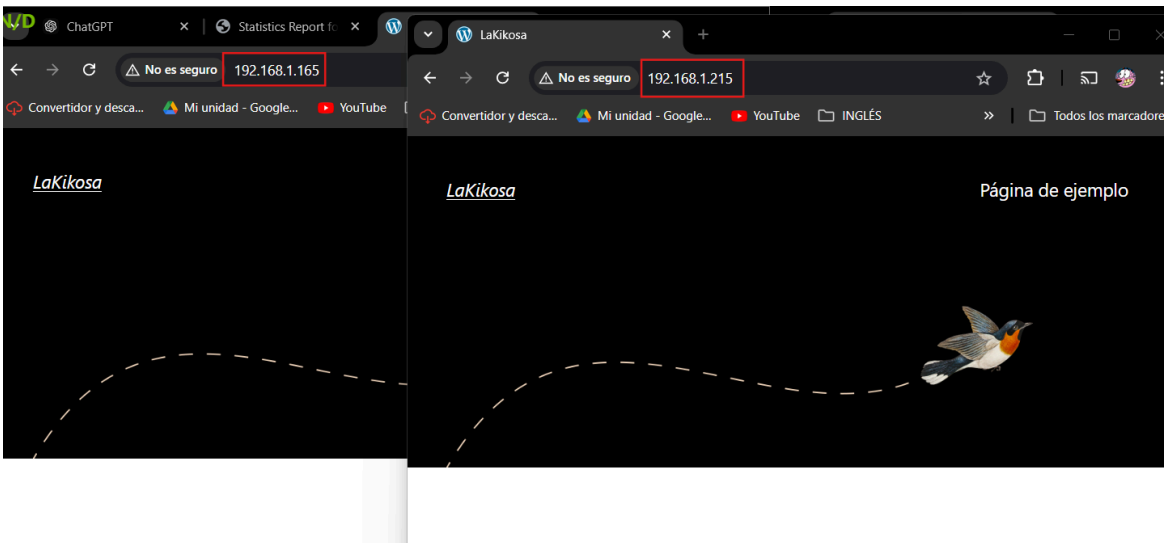
10.- Le damos permisos al fichero creado anteriormente.

```
root@Tyrion:/var/www/html/wordpress# chown www-data:www-data /var/www/html/wordpress/
.htaccess
root@Tyrion:/var/www/html/wordpress# chmod 644 /var/www/html/wordpress/.htaccess
```

11.- Y si nos vamos a un navegador web y ponemos la IP de nuestro servidor WordPress ya lo podremos configurar.



12.- Una vez realizado este proceso en ambos servidores podremos acceder de forma simultánea a ambos.

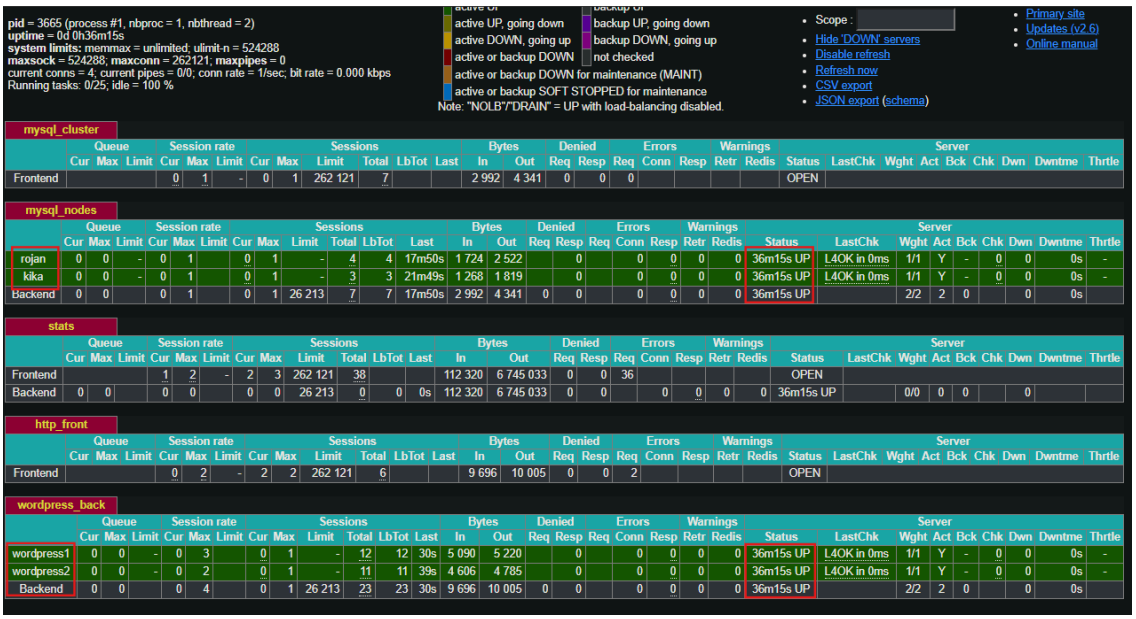


13.- Por último lo que tendremos que hacer es irnos a nuestro servidor HaProxy también para que balancee la carga de nuestro WordPress, tendremos que añadir las siguientes líneas al archivo de configuración del HaProxy (**/etc/haproxy/haproxy.cfg**).

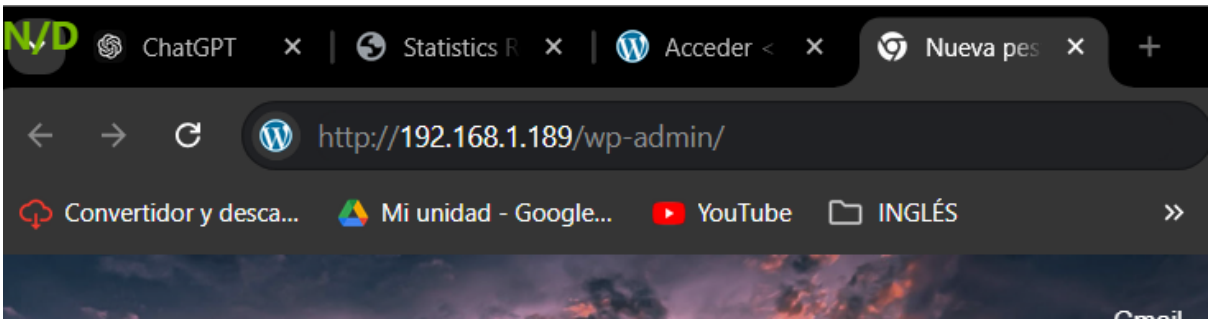
```
#CONFIGURACIÓN DEL WORDPRESS

frontend http_front
    bind *:80
    default_backend wordpress_back

backend wordpress_back
    balance roundrobin
    #option httpchk GET /wp-login.php
    server wordpress1 192.168.1.215:80 check
    server wordpress2 192.168.1.165:80 check
```

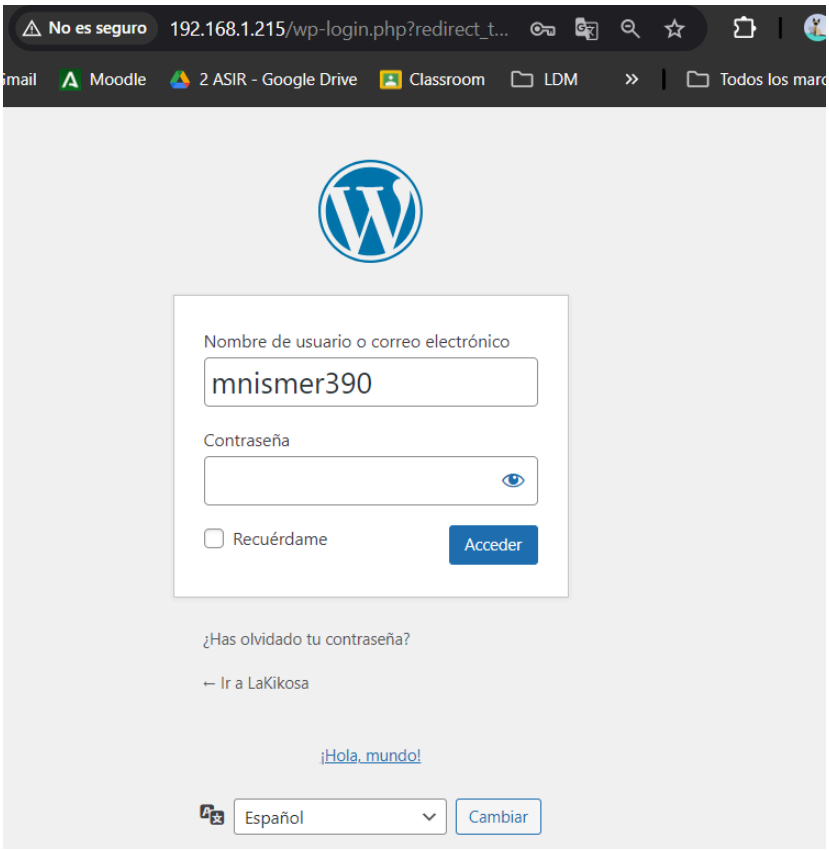
14.- Si ejecutamos el siguiente comando podemos ver que nos balancea a un servidor u otro. La IP que podremos en el comando es la del HaProxy.

```
root@Moria:/home/frodo# for i in {1..10}; do curl -I http://192.168.1.189/; sleep 1; done
HTTP/1.1 200 OK
date: Sat, 12 Oct 2024 17:17:57 GMT
server: Apache/2.4.62 (Debian)
last-modified: Sat, 12 Oct 2024 15:24:25 GMT
etag: "29cd-6244932ee23cf" WP1
accept-ranges: bytes
content-length: 10701
vary: Accept-Encoding
content-type: text/html

HTTP/1.1 200 OK
date: Sat, 12 Oct 2024 17:17:58 GMT
server: Apache/2.4.62 (Debian)
last-modified: Sat, 12 Oct 2024 16:38:57 GMT
etag: "29cd-6244a3d79fb9c" WP2
accept-ranges: bytes
content-length: 10701
vary: Accept-Encoding
content-type: text/html

HTTP/1.1 200 OK
date: Sat, 12 Oct 2024 17:17:59 GMT
server: Apache/2.4.62 (Debian)
last-modified: Sat, 12 Oct 2024 15:24:25 GMT
etag: "29cd-6244932ee23cf" WP1
```



15.- Si ponemos la IP del HaProxy y le damos a enter nos va redirigir a uno de los nodos.





CONCLUSIÓN

En resumen, la implementación de un clúster de bases de datos MySQL ofrece una solución eficiente para asegurar la disponibilidad y el rendimiento de los datos. La clara división de roles entre los nodos garantiza la integridad y accesibilidad de la información. Las pruebas realizadas y el uso de scripts en Python confirman la operatividad del sistema, mientras que HAProxy mejora la distribución del tráfico, preparando la infraestructura para enfrentar futuras demandas.

APARTADO 7

WEBGRAFÍA

Clúster MySQL

- Manual Clúster MySQL
- Vídeo instalación clúster
- Vídeo Cluster MariaDB + Galera
- Clúster MariaDB

Instalación HaProxy

Configuración Firewall

- Enlace Firewall 2

Instalación WordPress

- Enlace WordPress 2
- Enlace WordPress 3

Info Adicional

FALLOS + SOLUCIONES

1.- Puertos → Firewall.
2.- Ficheros de configuración → Parámetros mal.
3.- Creación de ficheros → Crearlos y configurarlos.
4.- Rutas ficheros mal puestas → Poner la ruta correcta