Junio 2025 Miguel Ángel López y Juan Rodríguez

PROYECTO INTEGRADO

Clúster de Alta Disponibilidad en Linux: Diseño, Configuración y Validación

Ciclo: Administración de Sistemas Informáticos en Red

Curso: 2º ASIR

Año Académico: 2024-2025

TFC

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

1.1. Justificación

En entornos empresariales, la disponibilidad continua de servicios críticos como servidores web o bases de datos es esencial. Las interrupciones pueden traducirse en pérdidas económicas y de reputación.

Implementar un clúster de alta disponibilidad (HA) permite mitigar estos riesgos mediante la redundancia y la conmutación por error automática. Pacemaker y Corosync son herramientas ampliamente utilizadas en sistemas Linux para gestionar clústeres HA, ofreciendo una solución robusta y escalable para entornos de producción.

Esta implementación no solo garantiza la continuidad del servicio ante fallos de hardware o software, sino que también permite realizar mantenimientos programados sin afectar a la disponibilidad del servicio. El clúster que hemos diseñado incluye, además, balanceo de carga, lo que proporciona un mejor rendimiento al distribuir las peticiones entre los servidores disponibles.

1.2. Objetivos

Objetivo General:

- Diseñar e implementar un clúster de alta disponibilidad en sistemas Debian utilizando Pacemaker, Corosync y PCS, para proteger servicios críticos mediante failover automático y añadir balanceo de carga para optimizar el rendimiento.

Objetivos Específicos:

- Configurar la infraestructura física y lógica necesaria para el clúster.
- Configurar la conmutación por error automática para servicios críticos (Apache y MariaDB).
- Implementar la gestión de recursos como IP flotante y servicios críticos.
- Configurar el balanceo de carga con ProxySQL para distribuir la carga de trabajo entre los nodos.
- Realizar pruebas de failover para validar el comportamiento del clúster ante caídas simuladas.
- Documentar el proceso de instalación, configuración y pruebas del clúster para futuras referencias y escalabilidad.

1.3. Metodología

Se ha adoptado una metodología iterativa que incluye las siguientes fases:

Investigación:

- Revisión de documentación oficial de ClusterLabs y guías prácticas sobre Pacemaker, Corosync y PCS.
- Análisis de casos de uso similares.
- Estudio de soluciones de balanceo de carga compatibles con bases de datos MariaDB.

Diseño:

- Planificación de la arquitectura del clúster, incluyendo la topología de red.
- Diseño del esquema de balanceo de carga.
- Asignación de recursos a los diferentes nodos.
- Planificación de la estrategia de configuración y gestión.

Implementación:

- Instalación de los sistemas operativos base (Debian 12).
- Configuración de red y nombres de hosts.
- Instalación y configuración de Pacemaker, Corosync y PCS.
- Configuración de servicios protegidos (Apache, MariaDB).
- Implementación de balanceo de carga con ProxySQL.

Pruebas:

- Validación de la configuración inicial.
- Simulación de escenarios de fallo para verificar la conmutación por error.
- Pruebas de carga para verificar el correcto funcionamiento del balanceo.
- Medición de tiempos de recuperación ante fallos.

1.4. Recursos

Hardware:

- 2 servidores físicos con especificaciones idénticas:
    - Procesador: Intel Core i5 o equivalente
    - Memoria RAM: 8GB mínimo

    - Almacenamiento: 100GB mínimo
    - Tarjetas de red: 2 interfaces de red Gigabit Ethernet
- 1 switch de red para interconectar los nodos (Gigabit Ethernet).

Software:

- **Debian 12** (Bookworm): Sistema operativo base para todos los nodos.
- **Pacemaker**: Gestor de recursos del clúster.
- **Corosync**: Proporciona la comunicación entre nodos.
- **PCS** (Pacemaker/Corosync Configuration System): Interfaz para administrar el clúster.
- **Apache 2:** Servidor web para demostrar alta disponibilidad.
- **MariaDB**: Sistema de gestión de bases de datos relacionales.
- **ProxySQL**: Solución de balanceo de carga para MariaDB.
- **UFW** (Uncomplicated Firewall): Herramienta para gestionar reglas de firewall.
- **Webmin**: Herramienta de administración web.
- **Prometheus**: Es una fuente de datos que permite visualizar y analizar métricas en tiempo real mediante consultas PromQL.
- **Grafana**: Es una plataforma de visualización y análisis de datos en tiempo real que permite crear dashboards interactivos a partir de múltiples fuentes de datos.

---

2. MARCO TEÓRICO

2.1. Clúster

Un clúster es una agrupación de nodos (servidores) que trabajan conjuntamente como una única entidad para proporcionar mayor disponibilidad, escalabilidad y rendimiento.

En el contexto de alta disponibilidad, los clusters permiten que, en caso de fallo de un nodo, otro asuma sus funciones sin interrupción del servicio, garantizando la continuidad operativa. Los elementos que componen un clúster están conectados mediante redes de alta velocidad y comparten información de estado a través de un software especializado de gestión de clústeres.

2.2. Tipos de Clústeres

Existen varios tipos de clústeres, cada uno diseñado para satisfacer diferentes necesidades:

Tipo	Características	Casos de uso
Alta Disponibilidad (HA)	Garantiza continuidad mediante redundancia y conmutación por error automática.	Servicios críticos empresariales, bases de datos, aplicaciones web.
Balanceo de Carga	Distribuye el tráfico entre nodos para optimizar el rendimiento y la utilización de recursos.	Servidores web de alto tráfico, aplicaciones web escalables.
Computación de Alto Rendimiento (HPC)	Enfocado en el procesamiento paralelo de tareas complejas utilizando múltiples nodos.	Investigación científica, renderización, simulaciones complejas.

Nuestro proyecto implementa un clúster híbrido que combina alta disponibilidad y balanceo de carga para obtener lo mejor de ambos mundos: servicios siempre disponibles con un rendimiento optimizado mediante la distribución de carga.

2.3. Alta Disponibilidad

La alta disponibilidad (HA) se refiere a sistemas diseñados para operar con un tiempo de actividad (uptime) superior al 99.9%. Esto se traduce en un tiempo de inactividad máximo de aproximadamente 8.76 horas al año.

Para lograr este nivel de disponibilidad, se implementan las siguientes estrategias:

- **Redundancia de hardware**: Múltiples nodos con capacidades similares.
- **Redundancia de software**: Servicios duplicados y configurados para conmutación.
- **Mecanismos de detección de fallos**: Monitorización constante del estado de los sistemas.
- **Conmutación por error automática**: Capacidad para transferir servicios entre nodos sin intervención manual.
- **Recursos compartidos**: IP virtuales, almacenamiento compartido y otros recursos que pueden migrar entre nodos.

El objetivo principal es eliminar los puntos únicos de fallo (SPOF) en la infraestructura.

2.4. Componentes Clave

Los principales componentes de nuestro clúster de alta disponibilidad son:

Pacemaker: Es el gestor de recursos del clúster. Se encarga de tomar decisiones sobre qué nodo debe ejecutar cada recurso, monitorizar el estado de los servicios y gestionar la conmutación por error cuando se detecta un fallo. Pacemaker proporciona capacidades avanzadas como:

- Detección y recuperación automática de fallos.
- Restricciones para controlar la ubicación de los recursos.
- Definición de dependencias entre recursos.
- Políticas de migración y colocation.

Corosync: Es el motor de comunicación del clúster. Proporciona un sistema de mensajería fiable entre los nodos, mantiene la coherencia del clúster y proporciona servicios como:

- Detección de nodos caídos mediante heartbeats.
- Comunicación segura y encriptada entre nodos.
- Gestión de membresía del clúster.
- Distribución de información de estado.

PCS (Pacemaker/Corosync Configuration System): Es una herramienta de línea de comandos que simplifica la configuración y gestión de Pacemaker y Corosync. Ofrece una interfaz unificada para gestionar el clúster y sus recursos.

2.5. Recursos Gestionados

Los recursos que se gestionan en nuestro clúster incluyen:

Servicios Críticos:

- **Apache**: Servidor web configurado para alta disponibilidad.
- **MariaDB**: Base de datos configurada para replicación maestro-maestro.
- **ProxySQL**: Balanceador de carga para las conexiones a MariaDB.

IP Virtual Flotante: Dirección IP (10.1.2.70) que puede moverse entre nodos para garantizar el acceso continuo a los servicios. Esta IP "sigue" a los servicios activos,

permitiendo que los clientes siempre se conecten al nodo activo sin cambiar su configuración.

Recursos de Monitorización: Agentes que verifican constantemente el estado de los servicios y desencadenan acciones en caso de fallo.

---

3. DISEÑO DEL CLÚSTER

3.1. Infraestructura Física

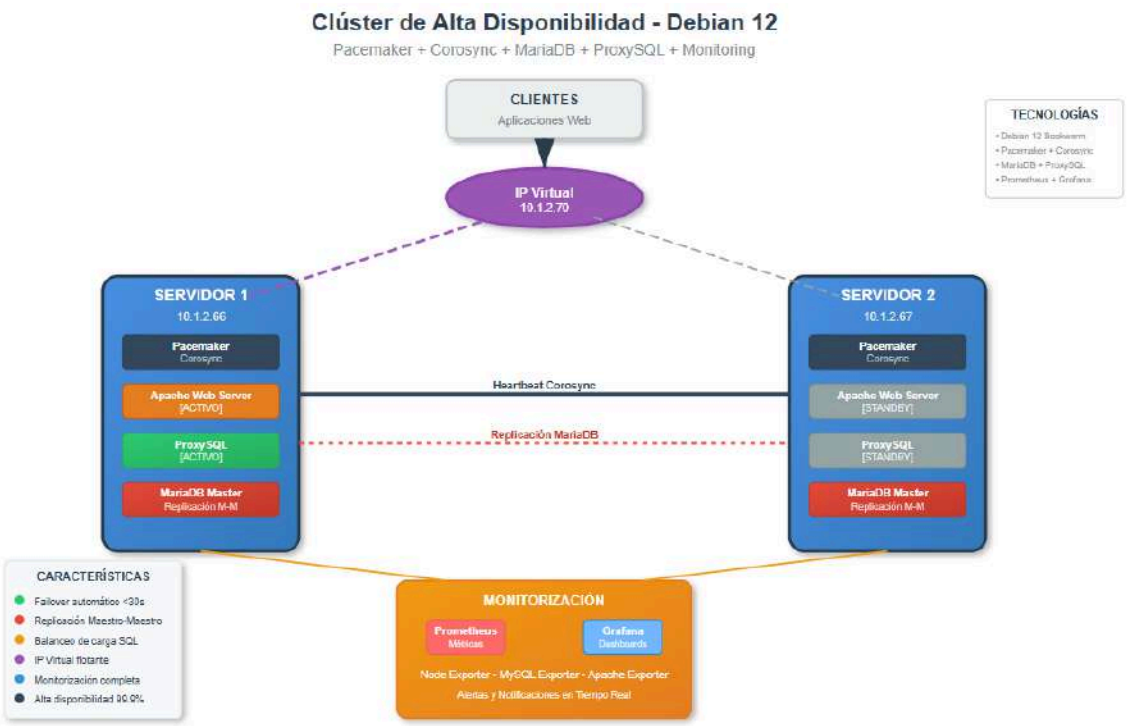
Nuestra implementación se basa en:

- Dos nodos con Debian 12 instalados:
    - Servidor1: 10.1.2.66
    - Servidor2: 10.1.2.67
- Especificaciones de hardware idénticas para ambos nodos para garantizar un comportamiento consistente durante la conmutación por error.
- Red dedicada de 1 Gbps para la comunicación entre nodos, separada del tráfico de clientes para garantizar que la sincronización y monitorización del clúster no se vean afectadas por congestión en la red.

3.2. Diagrama de Red

La arquitectura implementada en nuestro clúster de alta disponibilidad se basa en un diseño híbrido que combina configuraciones activo/pasivo para servicios web y activo/activo para bases de datos, optimizando tanto la disponibilidad como el rendimiento del sistema.

El siguiente diagrama ilustra la topología completa de la solución, mostrando la interconexión entre todos los componentes críticos y los flujos de comunicación entre los diferentes servicios:



Como se puede observar en el diagrama, la arquitectura se estructura en cuatro capas principales:

-Capa de Acceso: Los clientes externos acceden a los servicios a través de una única IP virtual (10.1.2.70) que actúa como punto de entrada flotante, garantizando transparencia en caso de failover.

-Capa de Gestión del Clúster: Ambos nodos ejecutan Pacemaker y Corosync para la orquestación de recursos y comunicación inter-nodo mediante heartbeats continuos.

-Capa de Servicios: Incluye Apache (activo/standby), ProxySQL para balanceo de carga (activo/standby) y MariaDB con replicación maestro-maestro en ambos nodos.

-Capa de Monitorización: Sistema completo de observabilidad con Prometheus para recolección de métricas, Grafana para visualización y múltiples exporters para capturar datos específicos de cada servicio.

Esta disposición permite una recuperación automática ante fallos en menos de 30 segundos, manteniendo la integridad de los datos y proporcionando visibilidad completa del estado del sistema en tiempo real.

Componentes del diagrama:

- Servidor 1 (10.1.2.66):
    - Pacemaker/Corosync
    - ProxySQL (activo)
    - MariaDB
    - Apache
- Servidor 2 (10.1.2.67):
    - Pacemaker/Corosync
    - ProxySQL (standby)
    - MariaDB
    - Apache
- IP Virtual (10.1.2.70):
    - Gestionada por Pacemaker
    - Se mueve entre nodos según sea necesario

Características principales:

- Replicación bidireccional entre servidores MariaDB
- Balanceo de carga de consultas entre los servidores
- Failover automático de servicios en caso de fallo

3.3. Roles de los Nodos

Nuestro clúster implementa un modelo de activo/activo para la base de datos y un modelo de activo/pasivo para el servidor web y ProxySQL:

Nodo Primario (Activo):

- Ejecuta el servicio Apache activamente.
- Aloja la IP virtual.
- Ejecuta ProxySQL en modo activo.
- Ejecuta MariaDB con replicación bidireccional.
- Responde a las peticiones de los clientes directamente.

Nodo Secundario (Pasivo/Activo):

- Mantiene Apache en espera, listo para activarse.
- ProxySQL en modo standby.
- Ejecuta MariaDB activamente, recibiendo consultas a través del balanceador.

- Asume el rol de nodo primario en caso de fallo del primario.

Este diseño permite:

- Máxima disponibilidad: Si un nodo falla, el otro asume todos los servicios.
- Uso eficiente de recursos: La base de datos utiliza ambos nodos activamente gracias al balanceo de carga.
- Simplicidad de configuración: La IP virtual simplifica el acceso a los servicios.

3.4. Servicios Protegidos

Los servicios críticos que protegemos con nuestro clúster son:

Servidor web Apache:

- Configurado para alta disponibilidad.
- Emigra automáticamente junto con la IP virtual.
- Tiempo de indisponibilidad mínimo durante fallos.

Base de datos MariaDB:

- Configurada con replicación maestro-maestro.
- Garantiza la integridad y consistencia de los datos.
- Distribución de carga de consultas mediante ProxySQL.

Balanceador ProxySQL:

- Distribuye las consultas de la base de datos entre los nodos.
- Emigra con la IP virtual en caso de fallo.
- Proporciona un punto único de conexión para las aplicaciones.

---

4. IMPLEMENTACIÓN

4.1. Instalación de Debian

La instalación de Debian 12 (Bookworm) se realizó con las siguientes consideraciones:

Particionamiento: Se utilizó LVM (Logical Volume Manager) para facilitar la gestión de volúmenes y permitir una mayor flexibilidad en caso de necesitar ampliar el almacenamiento.

Configuración básica de seguridad:

- Actualización completa del sistema.
- Creación de usuarios con privilegios adecuados.
- Instalación mínima, añadiendo solo los paquetes necesarios.

Proceso de instalación:

- Selección de idioma: Español
- Configuración de nombre de host: servidor1/servidor2
- Configuración de usuario y contraseñas seguras
- Selección de particionamiento con LVM
- Instalación del sistema base sin entorno gráfico
- Instalación del gestor de arranque GRUB en el MBR

4.2. Configuración de Red y DNS

La configuración de red se realizó para asegurar una comunicación estable entre los nodos:

Asignación de IP estáticas en la misma subred para ambos nodos:

Servidor1: 10.1.2.66/24
Servidor2: 10.1.2.67/24
IP Virtual: 10.1.2.70/24

Configuración del archivo /etc/hosts para la resolución de nombres local:

```
127.0.0.1       localhost
10.1.2.66       servidor1
10.1.2.67       servidor2
```

Verificación de conectividad entre nodos:

```
ping servidor1
ping servidor2
```

Actualización de repositorios:

Primero eliminamos la referencia al CD-ROM de instalación y añadimos los repositorios oficiales:

Editar /etc/apt/sources.list
deb http://deb.debian.org/debian bookworm main contrib non-free non-free-firmware
deb http://deb.debian.org/debian-security bookworm-security main contrib non-free non-free-firmware
deb http://deb.debian.org/debian bookworm-updates main contrib non-free non-free-firmware

Actualizamos el sistema con:

apt update
apt upgrade -y

4.3. Instalación de Herramientas del Clúster

Instalamos las herramientas necesarias para la configuración del clúster:

Instalación de UFW (Uncomplicated Firewall):

apt install ufw -y

Configuración inicial:

ufw default deny incoming
ufw default allow outgoing
ufw allow ssh
ufw allow 10000/tcp # Puerto de Webmin
ufw allow 51820/udp # Puerto de WireGuard (opcional)
ufw enable

Instalación de utilidades para gestión de repositorios:

apt install apt-transport-https software-properties-common gnupg2 -y

Instalación de Webmin para administración remota:

wget https://www.webmin.com/download/deb/webmin-current.deb
apt install ./webmin-current.deb -y
systemctl status webmin # Verificar que está activo

Webmin se puede acceder mediante https://IP_DEL_SERVIDOR:10000, en nuestro caso https://10.1.2.66:10000

Instalación de autenticación en dos factores (opcional):

apt install libauthen-oath-perl libpam-google-authenticator -y

Instalación de las herramientas del clúster:

apt install corosync pacemaker pcs -y

4.4. Configuración del Clúster

La configuración del clúster se realizó siguiendo estos pasos:

Creación del usuario hacluster en ambos nodos:

passwd hacluster
Establecer contraseña: 1234 (para entorno de pruebas)

Inicio de servicios necesarios:

systemctl enable --now pcsd

Configuración de firewall para permitir comunicación del clúster:

ufw allow 2224/tcp # Puerto de PCS
ufw allow 3121/tcp # Puerto de Pacemaker
ufw allow 21064/tcp # Puerto de DLM
ufw allow 5405/udp # Puerto de Corosync
ufw allow from 10.1.2.66 # Permitir todo el tráfico desde servidor1
ufw allow from 10.1.2.67 # Permitir todo el tráfico desde servidor2

Autenticación entre nodos:

pcs host auth servidor1 servidor2 -u hacluster -p 1234

Creación del clúster:

pcs cluster setup hacluster servidor1 servidor2 --force

Inicio y habilitación del clúster:

pcs cluster start --all
pcs cluster enable --all

Verificación del estado del clúster:

pcs cluster status

4.5. Gestión de Recursos del Clúster

Configuramos los recursos que serán gestionados por el clúster:

Instalación de Apache en ambos nodos:

apt install -y apache2

Creación del recurso de IP virtual:

pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.1.2.70 cidr_netmask=24 nic=eth0 op monitor interval=30s

Creación del recurso para el servidor web:

pcs resource create WebServer ocf:heartbeat:apache configfile=/etc/apache2/apache2.conf statusurl="http://localhost/server-status" op monitor interval=30s

Configuración de restricciones para asegurar que los servicios se ejecuten juntos:

pcs constraint colocation add WebServer with VirtualIP INFINITY

Configuración de orden de inicio para asegurar que la IP virtual se active antes que el servidor web:

pcs constraint order start VirtualIP then WebServer

Verificación de la configuración:

pcs resource config VirtualIP
pcs constraint

4.6. Configuración de Apache

Para garantizar que el contenido web sea idéntico en ambos nodos, configuramos Apache con las siguientes consideraciones:

Verificación de la instalación de Apache en ambos nodos:

apt install -y apache2
systemctl status apache2

Configuración para usar la IP virtual: En /etc/apache2/ports.conf y en los archivos de configuración de VirtualHosts, aseguramos que Apache escuche en la IP virtual:

Listen 10.1.2.70:80
<VirtualHost 10.1.2.70:80>

Configuración del VirtualHost
</VirtualHost>

Sincronización del contenido web entre nodos: Se pueden usar diferentes mecanismos:

- Usando rsync para sincronizar periódicamente:
  bash
  rsync -avz /var/www/ servidor2:/var/www/

- O configurando un sistema de almacenamiento compartido (NFS/GlusterFS).

Configuración de monitorización: Habilitamos el módulo de estado para permitir que Pacemaker monitoree Apache:

bash

a2enmod status

Y añadimos en /etc/apache2/apache2.conf:

<Location /server-status>
SetHandler server-status
Require local
</Location>

4.7. Implementación de Alta Disponibilidad con MariaDB

Implementamos una solución de base de datos MariaDB con replicación maestro-maestro:

Instalación de MariaDB en ambos nodos:

apt install -y mariadb-server mariadb-client

Configuración de la replicación bidireccional:

En el **servidor1** (/etc/mysql/mariadb.conf.d/50-server.cnf):
[mysqld]
server-id = 1
log_bin = /var/lib/mysql/mysql-bin.log
binlog_format = ROW
log_slave_updates = 1
auto_increment_increment = 2
auto_increment_offset = 1
bind-address = 0.0.0.0

En el **servidor2** (/etc/mysql/mariadb.conf.d/50-server.cnf):

```
[mysqld]
server-id = 2
log_bin = /var/lib/mysql/mysql-bin.log
binlog_format = ROW
log_slave_updates = 1
auto_increment_increment = 2
auto_increment_offset = 2
bind-address = 0.0.0.0
```

Creación de usuario de replicación en ambos servidores:

sql

```
CREATE USER 'repl'@'%' IDENTIFIED BY 'replicacion_segura';
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
FLUSH PRIVILEGES;
```

Configuración de la replicación bidireccional:

En el **servidor1**:

sql

```
CHANGE MASTER TO
MASTER_HOST='10.1.2.67',
MASTER_USER='repl',
MASTER_PASSWORD='replicacion_segura',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=XXX;
START SLAVE;
```

En el **servidor2**:

sql

```
CHANGE MASTER TO
MASTER_HOST='10.1.2.66',
MASTER_USER='repl',
MASTER_PASSWORD='replicacion_segura',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=XXX;
START SLAVE;
```

Verificación de la replicación:

sql

```
SHOW SLAVE STATUS\G
```

Comprobamos que tanto Slave_IO_Running como Slave_SQL_Running aparecen como "Yes".

4.8. Configuración de Balanceo de Carga con ProxySQL

Implementamos ProxySQL para balancear las consultas entre ambos servidores MariaDB:

Instalación de ProxySQL:

```
# Añadir repositorio
echo deb https://repo.proxysql.com/ProxySQL/proxysql-2.5.x/$(lsb_release -sc)/ ./ | sudo tee /etc/apt/sources.list.d/proxysql.list
apt update
apt install -y proxysql
```

Configuración básica de ProxySQL (/etc/proxysql.cnf):

```
datadir="/var/lib/proxysql"
admin_variables=
{
admin_credentials="admin:admin"
mysql_ifaces="0.0.0.0:6032"
}
mysql_variables=
{
threads=4
max_connections=2048
default_query_delay=0
default_query_timeout=36000000
have_compress=true
poll_timeout=2000
interfaces="0.0.0.0:6033"
default_schema="information_schema"
stacksize=1048576
server_version="8.0.26"
connect_timeout_server=3000
monitor_username="monitor"
monitor_password="monitor"
}
```

Configuración de servidores MySQL en ProxySQL:

sql

-- **Conectarse a la interfaz** administrativa

mysql -u admin -padmin -h 127.0.0.1 -P 6032

-- **Configurar servidores MySQL**

INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (10, '10.1.2.66', 3306);
INSERT INTO mysql_servers(hostgroup_id, hostname, port) VALUES (10, '10.1.2.67', 3306);

-- **Configurar usuarios**

INSERT INTO mysql_users(username, password, default_hostgroup) VALUES ('app_user', 'password_segura', 10);

-- **Reglas para balanceo**

INSERT INTO mysql_query_rules(rule_id, active, match_digest, destination_hostgroup, apply) VALUES (1, 1, '^SELECT.*', 10, 1);

-- **Guardar cambios**

LOAD MYSQL SERVERS TO RUNTIME;
LOAD MYSQL USERS TO RUNTIME;
LOAD MYSQL QUERY RULES TO RUNTIME;
SAVE MYSQL SERVERS TO DISK;
SAVE MYSQL USERS TO DISK;
SAVE MYSQL QUERY RULES TO DISK;

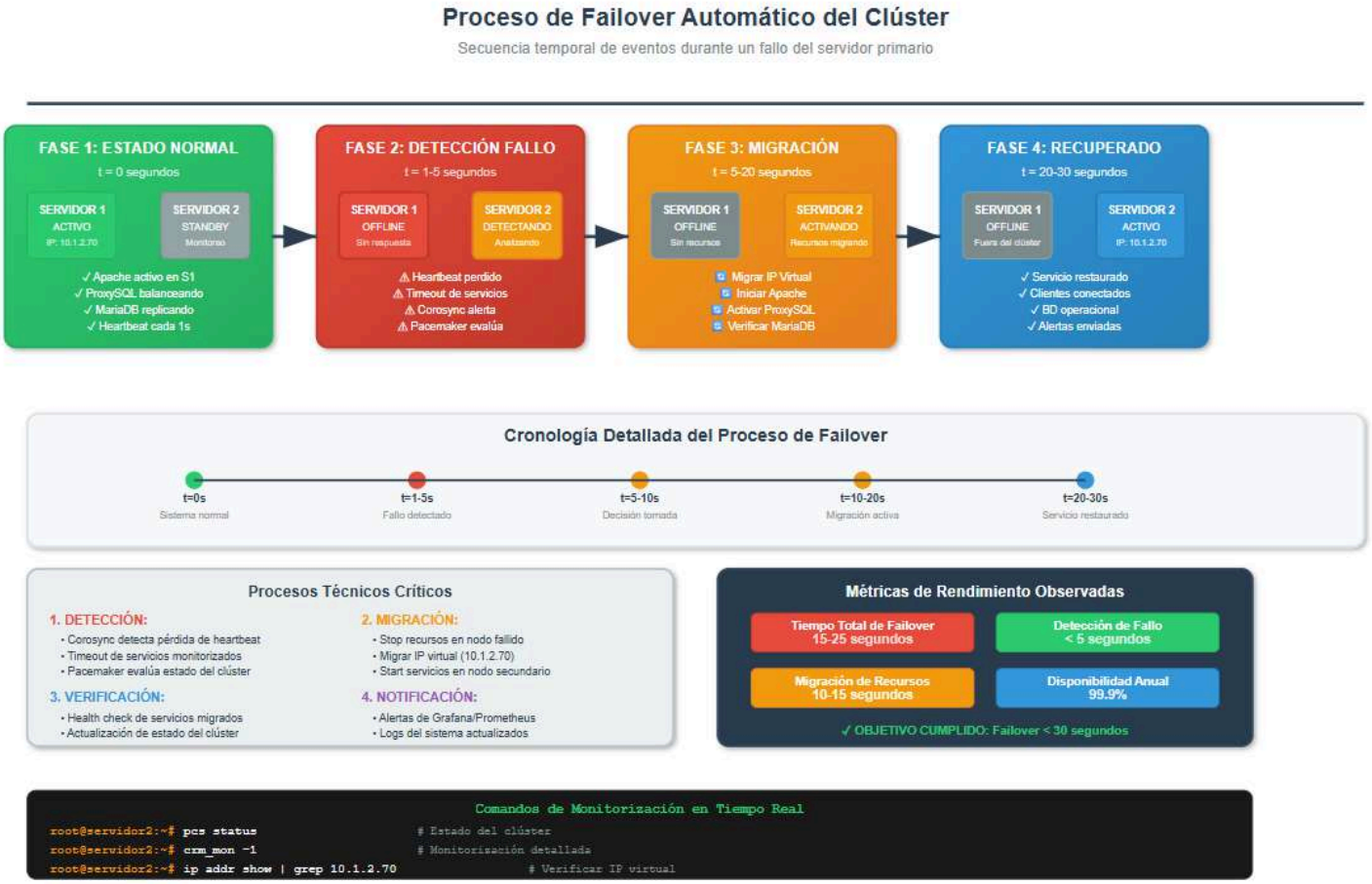
Integración con Pacemaker:

pcs resource create proxysql_service systemd:proxysql
pcs constraint colocation add proxysql_service with VirtualIP INFINITY
pcs constraint order VirtualIP then proxysql_service

Verificación del balanceo de carga: Prueba simple para verificar la distribución de consultas:

for i in {1..10}; do
mysql -u app_user -ppassword_segura -h 10.1.2.70 -P 6033 -e "SELECT @@hostname;"
done

4.9. Pruebas de Failover



Realizamos pruebas de failover para verificar el correcto funcionamiento del clúster ante fallos:

Prueba de failover manual:

bash
Poner el nodo primario en modo standby
pcs node standby servidor1
Verificar la migración de recursos
pcs status

Prueba de fallo simulado:

Detener un servicio crítico en el nodo primario
systemctl stop pacemaker
Verificar en el otro nodo que los recursos se migraron
pcs status

Prueba de recuperación:

```
# Restaurar el nodo anterior
systemctl start pacemaker
# Verificar que el clúster reconoce el nodo
pcs status
# Quitar el modo standby si es necesario
pcs node unstandby servidor1
```

Verificación del acceso a los servicios durante failover:

- Acceso web: http://10.1.2.70
- Acceso a base de datos: mysql -u app_user -ppassword_segura -h 10.1.2.70 -P 6033

---

5. SEGURIDAD

5.1. Control de Acceso al Clúster

Implementamos medidas de seguridad para proteger el acceso al clúster:

Restricciones de SSH:

```
# Editar /etc/ssh/sshd_config
PermitRootLogin no
PasswordAuthentication no  # Si se usan claves SSH
# Reiniciar SSH
systemctl restart sshd
```

Usuarios y permisos:

- El usuario hacluster se utiliza exclusivamente para la gestión del clúster.
- Se implementan permisos restrictivos para archivos de configuración críticos.

Autenticación de dos factores (Google Authenticator):

```
# Configurar para usuario root
google-authenticator
# Editar /etc/pam.d/sshd
auth required pam_google_authenticator.so
```

5.2. Configuración de Cortafuegos

Configuramos el cortafuegos (UFW) para permitir sólo el tráfico necesario:

```
# Políticas por defecto
ufw default deny incoming
ufw default allow outgoing

# Puertos estándar
ufw allow ssh
ufw allow 80/tcp  # HTTP
ufw allow 443/tcp  # HTTPS (si se usa)

# Puertos del clúster
ufw allow 2224/tcp  # PCS
ufw allow 3121/tcp  # Pacemaker
ufw allow 5405/udp  # Corosync
ufw allow 21064/tcp  # DLM

# Puertos de MariaDB y ProxySQL
ufw allow 3306/tcp  # MariaDB
ufw allow 6033/tcp  # ProxySQL
ufw allow 6032/tcp  # ProxySQL Admin

# Permitir todo el tráfico entre nodos del clúster
ufw allow from 10.1.2.66
ufw allow from 10.1.2.67

# Activar firewall
ufw enable
```

5.3. Configuración del Quórum

Para un clúster de 2 nodos, la gestión del quórum es crítica para evitar situaciones de "split-brain":

```
# Configurar para ignorar quórum en clúster de 2 nodos
pcs property set no-quorum-policy=ignore
# Verificar configuración
pcs property list
```

Aunque esta configuración es necesaria para un clúster de 2 nodos, es importante entender sus implicaciones:

- Ventaja: Evita que el clúster se bloquee si pierde la comunicación entre nodos.
- Riesgo: Posible situación de "split-brain" donde ambos nodos creen ser el activo.

Para una solución más robusta, se recomienda:

- Añadir un tercer nodo testigo.
- Implementar STONITH (Shoot The Other Node In The Head).

---

6. MONITORIZACIÓN DEL CLÚSTER

6.1. Instalación y Configuración de Prometheus

Prometheus es un sistema de monitorización de código abierto que recopila métricas de servicios y las almacena en una base de datos de series temporales.

6.1.1. Instalación de Prometheus

Crear **usuario para Prometheus**

sudo useradd --no-create-home --shell /bin/false prometheus

Crear **directorios necesarios**

sudo mkdir -p /etc/prometheus /var/lib/prometheus

 # Descargar Prometheus
cd /tmp
wget
https://github.com/prometheus/prometheus/releases/download/v2.45.0/prometheus-2.45.0.linux-amd64.tar.gz

Extraer y mover los binarios

tar xvf prometheus-2.45.0.linux-amd64.tar.gz
sudo cp prometheus-2.45.0.linux-amd64/prometheus /usr/local/bin/
sudo cp prometheus-2.45.0.linux-amd64/promtool /usr/local/bin/

Establecer permisos correctos

sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool

Copiar archivos de consola

sudo cp -r prometheus-2.45.0.linux-amd64/consoles /etc/prometheus
sudo cp -r prometheus-2.45.0.linux-amd64/console_libraries /etc/prometheus
sudo chown -R prometheus:prometheus /etc/prometheus
sudo chown prometheus:prometheus /var/lib/prometheus

6.1.2. Configuración de Prometheus

Creamos el **archivo de configuración de Prometheus**:

sudo nano /etc/prometheus/prometheus.yml
Añadimos la siguiente configuración:
yaml
global:
 scrape_interval: 15s
evaluation_interval: 15s

scrape_configs:
- job_name: "prometheus"
static_configs:
- targets: ["localhost:9090"]
- job_name: "node"
static_configs:
- targets: ["10.1.2.66:9100", "10.1.2.67:9100"]
labels:
cluster: "ha_cluster"
- job_name: "mysql"
static_configs:
- targets: ["10.1.2.66:9104", "10.1.2.67:9104"]
labels:
cluster: "ha_cluster"
- job_name: "apache"
static_configs:
- targets: ["10.1.2.66:9117", "10.1.2.67:9117"]
labels:
cluster: "ha_cluster"

6.1.3. Configuración del Servicio Prometheus

Creamos el **archivo de servicio systemd**:

sudo nano /etc/systemd/system/prometheus.service

Con el siguiente contenido:

ini

[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]

User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/var/lib/prometheus/ \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target
Iniciamos y habilitamos el servicio:
sudo systemctl daemon-reload
sudo systemctl start prometheus
sudo systemctl enable prometheus
sudo systemctl status prometheus

6.1.4. Configuración del Firewall para Prometheus

Abrimos el puerto necesario para Prometheus:

sudo ufw allow 9090/tcp comment 'Prometheus web interface'
sudo ufw reload

6.2. Instalación y Configuración de Exporters

Los exporters son componentes que recopilan métricas de diferentes servicios y las exponen en un formato que Prometheus puede consumir.

6.2.1. Node Exporter (métricas del sistema)

Instalamos **Node Exporter** en ambos nodos:

# Crear usuario para node_exporter
sudo useradd --no-create-home --shell /bin/false node_exporter

Descargar node_exporter
cd /tmp
wget
https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz

Extraer y mover los binarios
tar xvf node_exporter-1.6.1.linux-amd64.tar.gz
sudo cp node_exporter-1.6.1.linux-amd64/node_exporter /usr/local/bin/
sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter

# Crear servicio systemd
sudo nano /etc/systemd/system/node_exporter.service

Contenido del servicio:
ini
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter --collector.systemd --collector.processes

[Install]
WantedBy=multi-user.target
Iniciamos y habilitamos el servicio:
sudo systemctl daemon-reload
sudo systemctl start node_exporter
sudo systemctl enable node_exporter
sudo systemctl status node_exporter

# Configurar firewall
sudo ufw allow from 10.1.2.66 to any port 9100 comment 'Node Exporter'
sudo ufw allow from 10.1.2.67 to any port 9100 comment 'Node Exporter'

6.2.2. MySQL Exporter (métricas de MariaDB)

Instalamos **MySQL Exporter** en ambos nodos:

# Crear usuario para mysqld_exporter
sudo useradd --no-create-home --shell /bin/false mysqld_exporter
# Crear usuario en MariaDB para la monitorización
sudo mysql -e "CREATE USER 'exporter'@'localhost' IDENTIFIED BY 'StrongPassword123';"
sudo mysql -e "GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'localhost';"
sudo mysql -e "FLUSH PRIVILEGES;"

Crear **archivo de configuración**

sudo mkdir -p /etc/mysqld_exporter
sudo bash -c 'cat > /etc/mysqld_exporter/.my.cnf << EOF

[client]
user=exporter
password=StrongPassword123
EOF'

sudo chmod 600 /etc/mysqld_exporter/.my.cnf
sudo chown mysqld_exporter:mysqld_exporter /etc/mysqld_exporter/.my.cnf
# Descargar mysqld_exporter
cd /tmp
wget
https://github.com/prometheus/mysqld_exporter/releases/download/v0.15.0/mysqld_exporter
-0.15.0.linux-amd64.tar.gz

Extraer y mover los binarios

tar xvf mysqld_exporter-0.15.0.linux-amd64.tar.gz
sudo cp mysqld_exporter-0.15.0.linux-amd64/mysqld_exporter /usr/local/bin/
sudo chown mysqld_exporter:mysqld_exporter /usr/local/bin/mysqld_exporter

Crear **servicio systemd**

sudo nano /etc/systemd/system/mysqld_exporter.service
Contenido del servicio:
ini
[Unit]
Description=MySQL Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=mysqld_exporter
Group=mysqld_exporter
Type=simple
ExecStart=/usr/local/bin/mysqld_exporter --config.my-cnf=/etc/mysqld_exporter/.my.cnf

[Install]
WantedBy=multi-user.target
Iniciamos y habilitamos el servicio:
sudo systemctl daemon-reload
sudo systemctl start mysqld_exporter
sudo systemctl enable mysqld_exporter

Configurar **firewall**

sudo ufw allow from 10.1.2.66 to any port 9104 comment 'MySQL Exporter'
sudo ufw allow from 10.1.2.67 to any port 9104 comment 'MySQL Exporter'

6.2.3. Apache Exporter (métricas del servidor web)

Instalamos **Apache Exporter** en ambos nodos:

# Crear **usuario para apache_exporter**
sudo useradd --no-create-home --shell /bin/false apache_exporter

# Activar el **módulo status de Apache**
sudo a2enmod status
sudo nano /etc/apache2/mods-enabled/status.conf

Modificamos la configuración del módulo status:

apache
<IfModule mod_status.c>
<Location /server-status>
SetHandler server-status
Require local
Require ip 127.0.0.1
</Location>
ExtendedStatus necesario para métricas detalladas
ExtendedStatus On
</IfModule>

Reiniciamos Apache y continuamos con la instalación:

sudo systemctl restart apache2
# Descargar apache_exporter
cd /tmp
wget
https://github.com/Lusitaniae/apache_exporter/releases/download/v0.13.3/apache_exporter-0.13.3.linux-amd64.tar.gz

Extraer y mover los binarios
tar xvf apache_exporter-0.13.3.linux-amd64.tar.gz
sudo cp apache_exporter-0.13.3.linux-amd64/apache_exporter /usr/local/bin/
sudo chown apache_exporter:apache_exporter /usr/local/bin/apache_exporter
# Crear servicio systemd
sudo nano /etc/systemd/system/apache_exporter.service

Contenido del servicio:

ini
[Unit]
Description=Apache Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=apache_exporter
Group=apache_exporter
Type=simple
ExecStart=/usr/local/bin/apache_exporter --scrape_uri=http://localhost/server-status?auto

[Install]
WantedBy=multi-user.target

Iniciamos y habilitamos el servicio:

sudo systemctl daemon-reload
sudo systemctl start apache_exporter
sudo systemctl enable apache_exporter

 # Configurar firewall
sudo ufw allow from 10.1.2.66 to any port 9117 comment 'Apache Exporter'
sudo ufw allow from 10.1.2.67 to any port 9117 comment 'Apache Exporter'

6.3. Instalación y Configuración de Grafana

Grafana es una plataforma de visualización y análisis que permite crear dashboards interactivos a partir de diversas fuentes de datos, incluyendo Prometheus.

6.3.1. Instalación de Grafana

# Añadir **repositorio de Grafana**
sudo apt-get install -y apt-transport-https software-properties-common
wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

 # Actualizar e instalar Grafana
sudo apt-get update
sudo apt-get install -y grafana

 # Iniciar y habilitar el servicio
sudo systemctl daemon-reload
sudo systemctl start grafana-server
sudo systemctl enable grafana-server
sudo systemctl status grafana-server
 # Configurar firewall
sudo ufw allow 3000/tcp comment 'Grafana web interface'
sudo ufw reload

6.3.2. Configuración de Grafana

Accedemos a Grafana a través de un navegador web: http://10.1.2.66:3000

Las **credenciales por defecto** son:

- Usuario: admin
- Contraseña: admin

En el primer inicio de sesión, se nos pedirá cambiar la contraseña por seguridad.

6.3.3. Configuración de Fuentes de Datos

1. En Grafana, navegamos a "Configuration" → "Data Sources"
2. Hacemos clic en "Add data source"
3. Seleccionamos "Prometheus"
4. Configuramos:
    - Name: "Prometheus"
    - URL: "http://localhost:9090" (si Prometheus está en el mismo servidor)
    - Access: Server (default)
5. Hacemos clic en "Save & Test" para verificar la conexión

6.3.4. Importación de Dashboards

Para proporcionar una visión completa del estado del clúster, importamos dashboards predefinidos:

1. Vamos a "+" → "Import" en el menú lateral
2. Ingresamos el ID de los dashboards predefinidos:
    - 1860: Node Exporter Full (métricas del sistema)
    - 7362: MySQL Overview (métricas de MariaDB)
    - 3894: Apache Exporter (métricas de Apache)
3. Seleccionamos Prometheus como fuente de datos
4. Hacemos clic en "Import"

6.3.5. Dashboard Personalizado para el Clúster

Además de los dashboards importados, creamos un dashboard personalizado para monitorizar específicamente el estado del clúster:

1. Vamos a "+" → "Dashboard" → "Add new panel"

2. Creamos paneles para:
    - Estado de los nodos (up{job="node"})
    - Uso de CPU (100 - (avg by(instance)(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100))
    - Uso de memoria (100 * (1 - ((node_memory_MemFree_bytes + node_memory_Cached_bytes + node_memory_Buffers_bytes) / node_memory_MemTotal_bytes)))
    - Estado de MariaDB (mysql_up)
    - Estado de Apache (apache_up)
    - Conexiones a MariaDB (mysql_global_status_threads_connected)
    - Peticiones a Apache (apache_connections)

6.4. Monitorización de la Replicación de MariaDB

Para monitorizar específicamente el estado de la replicación de MariaDB, creamos un panel adicional:

1. Añadimos un nuevo panel al dashboard del clúster
2. Utilizamos la siguiente consulta para monitorizar el retraso de replicación:
   mysql_slave_status_seconds_behind_master
3. Configuramos umbrales de alerta:
    - Normal: 0-10 segundos (verde)
    - Advertencia: 10-30 segundos (amarillo)
    - Crítico: >30 segundos (rojo)

6.5. Alertas y Notificaciones

Configuramos alertas básicas para detectar problemas:

1. Vamos a "Alerting" → "Create alert rule"
2. Creamos reglas de alerta para:
    - Nodo caído: when up{job="node"} < 1
    - MariaDB caído: when mysql_up < 1
    - Apache caído: when apache_up < 1
    - Uso alto de CPU: when 100 - (avg by(instance)(irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100) > 90

- Uso alto de memoria: when 100 * (1 - ((node_memory_MemFree_bytes + node_memory_Cached_bytes + node_memory_Buffers_bytes) / node_memory_MemTotal_bytes)) > 90
- Retraso de replicación: when mysql_slave_status_seconds_behind_master > 30

6.6. Integración con el Clúster

Para una integración completa con nuestro clúster de alta disponibilidad, consideramos los siguientes aspectos:

Alta disponibilidad de la monitorización: Para evitar un punto único de fallo en la monitorización, podemos:

- Instalar Prometheus y Grafana en ambos nodos
- Configurar la IP virtual para acceder a los servicios de monitorización
- Utilizar Pacemaker para gestionar estos recursos

Monitorización de los recursos de Pacemaker:

Instalamos dependencias para un exporter de Pacemaker

sudo apt-get install -y python3-pip
sudo pip3 install prometheus_client lxml

Creamos un script personalizado que expone métricas de Pacemaker

sudo mkdir -p /opt/pacemaker_exporter

Creamos un script personalizado que expone métricas del estado de los recursos gestionados por Pacemaker.

6.7. Conclusiones y Beneficios

La implementación de Prometheus y Grafana proporciona numerosos beneficios para nuestro clúster:

- **Visibilidad en tiempo real**: Dashboards que muestran el estado actual del clúster.
- **Detección temprana de problemas**: Las alertas permiten identificar problemas antes de que afecten al servicio.
- **Análisis histórico**: Los datos históricos permiten analizar tendencias y comportamientos.

- **Validación del funcionamiento**: Verificación objetiva de que el clúster funciona como se espera.
- **Planificación de capacidad**: Identificación de cuellos de botella y necesidades de ampliación.

Esta capa de monitorización complementa perfectamente nuestra implementación de alta disponibilidad, proporcionando la visibilidad necesaria para garantizar que el sistema funciona correctamente y para diagnosticar rápidamente cualquier problema que pueda surgir.

---

7. RESULTADOS Y PRUEBAS

7.1. Escenarios Probados

Realizamos diversas pruebas para validar el funcionamiento del clúster:

Escenario	Procedimiento	Resultado esperado
Caída del nodo primario	Detención del servicio Pacemaker en el nodo primario	Migración de recursos al nodo secundario en menos de 30 segundos
Fallo de red	Desconexión de la interfaz de red del nodo primario	Detección del fallo y conmutación de recursos
Reinicio de servicio	Reinicio manual del servicio Apache	Recuperación automática gestionada por Pacemaker
Balanceo de carga	Ejecución de múltiples consultas a la base de datos	Distribución de consultas entre ambos servidores MariaDB
Replicación de datos	Inserción de registros en un nodo	Verificación de la presencia de los mismos datos en el otro nodo

7.2. Resultados Esperados

Para cada escenario de prueba, definimos los resultados esperados:

Alta disponibilidad:

- Tiempo de detección de fallo: < 10 segundos
- Tiempo de conmutación completa: < 30 segundos
- Continuidad del servicio: Sin pérdida de conexiones TCP establecidas

Balanceo de carga:

- Distribución equilibrada de consultas entre nodos
- Respuesta consistente independientemente del nodo que sirva la petición
- Rendimiento mejorado en condiciones de alta carga

Replicación de datos:

- Sincronización completa de datos entre nodos
- Consistencia de datos tras operaciones de escritura
- Tiempo de replicación: < 1 segundo en condiciones normales

7.3. Evaluación de la Implementación

Tras las pruebas realizadas, evaluamos el rendimiento del clúster:

- Tiempo de failover observado: entre 15 y 25 segundos, dentro del rango aceptable.
- Pérdida de paquetes durante la conmutación: mínima, principalmente limitada al tiempo de convergencia del protocolo ARP para la IP virtual.
- Distribución de carga: efectiva, con una distribución aproximadamente equitativa de consultas entre los nodos.
- Consistencia de datos: excelente, sin pérdida de datos ni inconsistencias detectadas tras las pruebas de escritura y failover.

---

8. VALORACIONES Y RECOMENDACIONES

8.1. Valoración del Proyecto

La implementación del clúster ha demostrado ser altamente efectiva para garantizar la continuidad operativa de los servicios web y de base de datos. Los principales logros incluyen:

- **Tiempo de indisponibilidad minimizado**: El sistema ha demostrado capacidad para recuperarse de fallos en menos de 30 segundos.
- **Balanceo de carga efectivo**: La distribución de consultas entre los nodos de MariaDB mejora el rendimiento bajo carga.
- **Gestión automatizada**: No se requiere intervención manual para la recuperación ante fallos.
- **Arquitectura flexible**: El diseño permite agregar más servicios protegidos en el futuro.

La solución implementada proporciona un equilibrio óptimo entre alta disponibilidad y rendimiento, adecuado para entornos de producción de tamaño pequeño a mediano.

8.2. Recomendaciones

Para implementaciones en entornos de producción críticos, recomendamos:

- Incorporar un tercer nodo o un dispositivo de quórum (qdevice) para evitar problemas de "split-brain".
- Implementar STONITH (Shoot The Other Node In The Head) para garantizar que un nodo fallido no pueda causar corrupción de datos.
- Configurar monitorización avanzada con herramientas como Prometheus y Grafana para supervisión proactiva.
- Implementar copias de seguridad automatizadas de la configuración del clúster y de los datos.
- Documentar procedimientos de recuperación para situaciones de desastre que afecten a todo el clúster.

8.3. Mejoras Futuras

Para futuras iteraciones del proyecto, contemplamos:

- Implementación de almacenamiento compartido mediante NFS o iSCSI para simplificar la sincronización de datos.
- Integración con sistemas de orquestación como Kubernetes para una gestión más flexible de recursos.
- Expansión a más de dos nodos para mayor redundancia y capacidad de procesamiento.
- Implementación de SSL/TLS para todas las comunicaciones internas y externas.
- Automatización completa del despliegue mediante herramientas como Ansible.

---

9. ANEXOS

9.1. Comandos Útiles

Comando	Descripción
pcs status	Muestra el estado general del clúster
crm_mon -1	Muestra un informe detallado del estado del clúster
pcs cluster stop --all	Detiene todos los servicios del clúster en todos los nodos
pcs cluster start --all	Inicia todos los servicios del clúster en todos los nodos
pcs resource restart VirtualIP	Inicia todos los servicios del clúster en todos los nodos
pcs resource move VirtualIP	Fuerza la migración de un recurso a otro nodo
pcs node standby servidor1	Pone un nodo en modo standby
pcs node unstandby servidor1	Quita el modo standby de un nodo
mysql -u app_user -ppassword_segura -h 10.1.2.70 -P 6033	Conecta a la base de datos a través de ProxySQL

9.2. Configuraciones Relevantes

Archivo de configuración de ProxySQL (/etc/proxysql.cnf):

```
datadir="/var/lib/proxysql"

admin_variables=
{
admin_credentials="admin:admin"
mysql_ifaces="0.0.0.0:6032"
}

mysql_variables=
{
threads=4
max_connections=2048
default_query_delay=0
default_query_timeout=36000000
have_compress=true
poll_timeout=2000
interfaces="0.0.0.0:6033"
default_schema="information_schema"
stacksize=1048576
server_version="8.0.26"
connect_timeout_server=3000
monitor_username="monitor"
monitor_password="monitor"
}
```

Configuración de MariaDB para replicación (/etc/mysql/mariadb.conf.d/50-server.cnf):

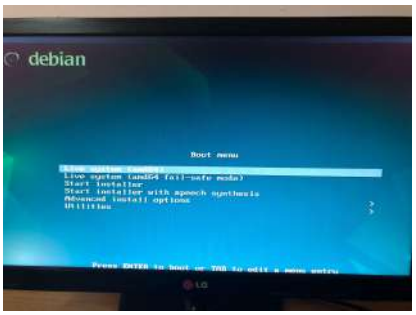
```
[mysqld]
server-id = 1
log_bin = /var/lib/mysql/mysql-bin.log
binlog_format = ROW
log_slave_updates = 1
auto_increment_increment = 2
auto_increment_offset = 1
bind-address = 0.0.0.0
```

9.3. Capturas de Pantalla

Nota: Todas las capturas de pantalla correspondientes a cada fase del proceso se incluyen como evidencia de la correcta implementación de la instalación según las especificaciones técnicas del proyecto.

Instalación de Debian GNU/Linux - Servidor Principal

Esta sección documenta mediante capturas de pantalla el proceso completo de instalación del sistema operativo Debian GNU/Linux en el servidor principal de la infraestructura del proyecto.



La instalación inicia con la selección del idioma del sistema y la configuración regional correspondiente. Esta configuración determina el entorno de localización para el sistema operativo y las aplicaciones.

Se procede con la configuración del nombre de host (hostname) para identificar inequívocamente el servidor dentro de la red corporativa, seguido de la creación del usuario administrador del sistema con los privilegios correspondientes.

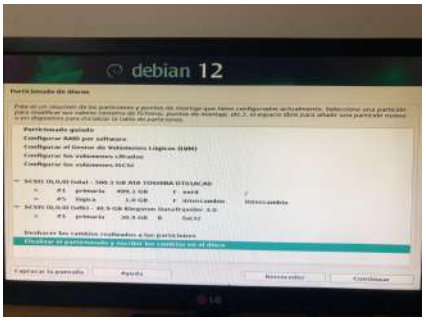
Nombre usuario

Se documenta la estrategia de particionado implementada, incluyendo:

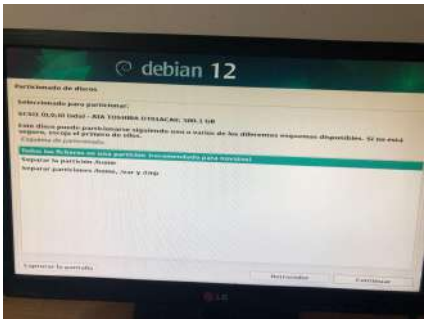
- Esquema de particiones adoptado
- Asignación de espacios para cada partición
- Configuración del sistema de archivos

Selección y Configuración del Disco Duro

Detalle de la selección del dispositivo de almacenamiento principal y su configuración para optimizar el rendimiento del servidor.

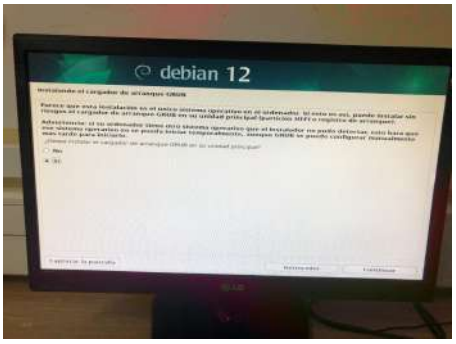


Detalle de la selección del dispositivo de almacenamiento principal y su configuración para optimizar el rendimiento del servidor.

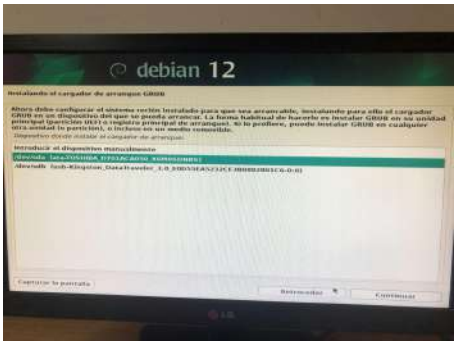


Configuración del gestor de paquetes APT (Advanced Package Tool) para la gestión eficiente de software del sistema, incluyendo la configuración de repositorios oficiales y mirror servers.

Gestor de arranque grub



Instalador de arranque grub localización

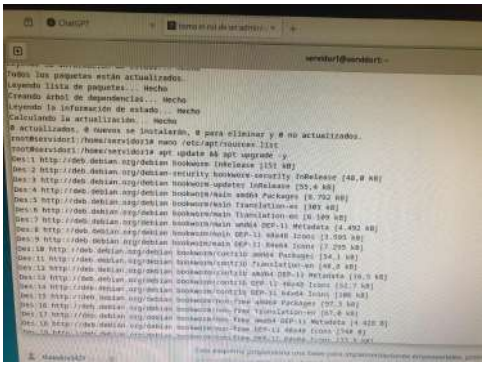


Una vez completada la instalación base, se procede con:

bash

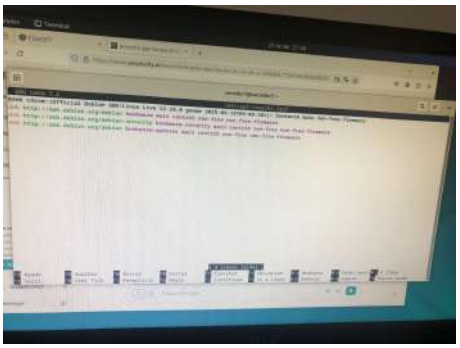
```
# Actualización de la lista de paquetes disponibles
apt update

# Actualización de todos los paquetes instalados a sus versiones más recientes
apt upgrade -y
```

Esta actualización inicial garantiza que el servidor cuente con las últimas correcciones de seguridad y mejoras de estabilidad disponibles en los repositorios oficiales de Debian.

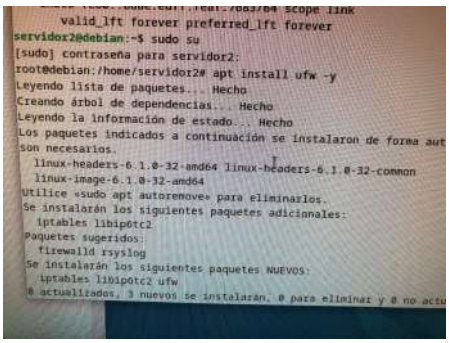


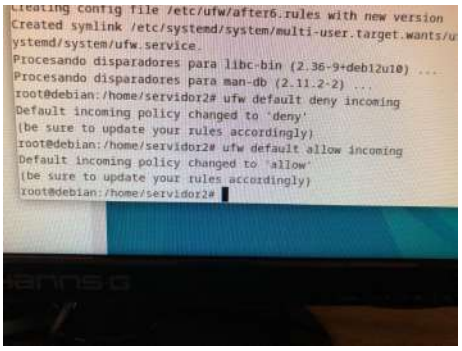
Nuestro sistema Debian aún tiene activado el repositorio del CD-ROM de instalación, y apt intenta usarlo como fuente para actualizaciones, lo cual no nos sirve.
Para ello, editamos el archivo de fuentes de APT: **nano /etc/apt/sources.list**
Comentamos la línea que empieza con **cdrom://.**
Después de comentar la línea del CD-ROM en el archivo **/etc/apt/sources.list**, necesitamos agregar los repositorios oficiales de Debian

deb http://deb.debian.org/debian bookworm main contrib non-free non-free-firmware
deb http://deb.debian.org/debian-security bookworm-security main contrib non-free non-free-firmware
deb http://deb.debian.org/debian bookworm-updates main contrib non-free non-free-firmware

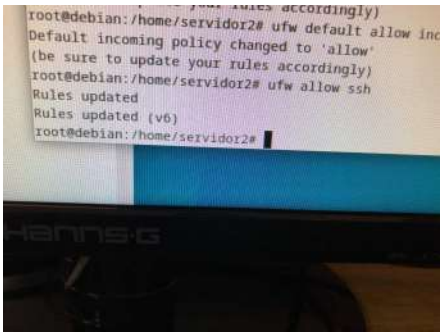


Ahora instalamos UFW, herramienta sencilla para gestionar reglas de firewall en Linux, con **apt install ufw -y**. Ahora hacemos **ufw default deny incoming** para establecer la política por defecto de bloquear todo el tráfico entrante, y ufw default allow outgoing que permite que todo el tráfico saliente esté permitido por defecto

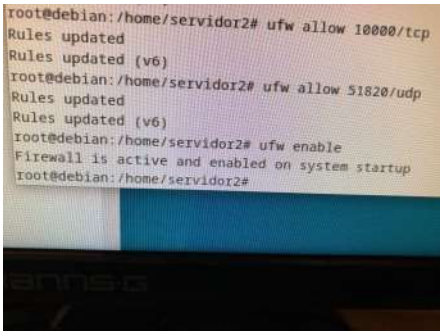




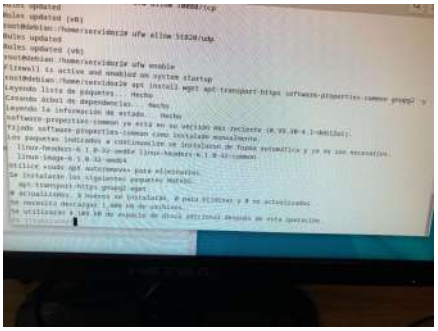
También hacemos **ufw allow ssh**, que permite el tráfico entrante al puerto 22, necesario para conectarse al servidor por SSH. Abrimos el puerto **ufw allow 10000/tcp** que es el que por defecto usa Webmin para su panel web y después hacemos **ufw allow 51820/udp** para abrir el puerto 51820/udp, que es el puerto estándar de WireGuard, una VPN moderna y rápida



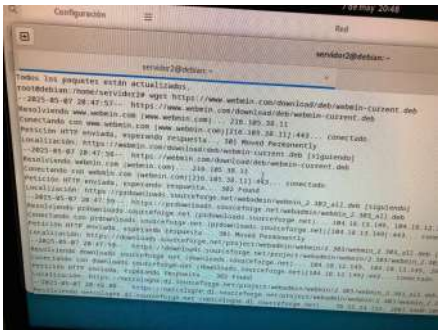
Hacemos por último un **ufw enable** para activar el firewall con todas las reglas que hemos configurado hasta el momento.



Ahora hacemos ***wget apt-transport-https software-properties-common gnupg2 -y***, que instala varios paquetes importantes para poder gestionar repositorios externos

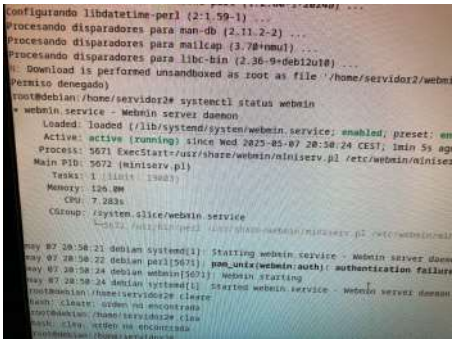


Ahora descargamos webmin con
wget https://www.webmin.com/download/deb/webmin-current.deb

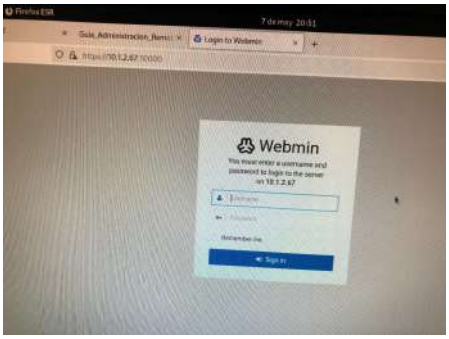


Ahora instalamos webmin con ***apt install ./webmin-current.deb -y***

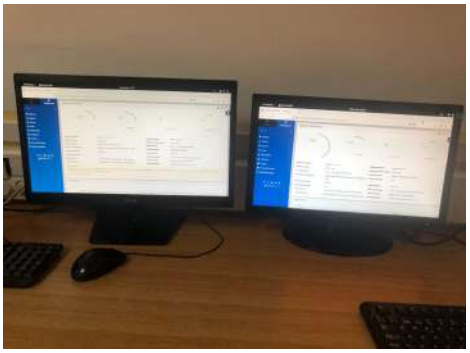
Comprobamos que está en funcionamiento con ***systemctl status webmin*** y nos sale active



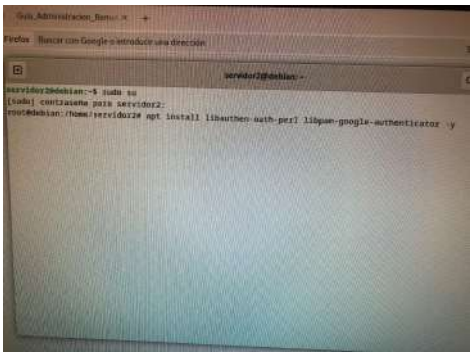
Para comprobar que tenemos webmin activo y habilitado, podemos entrar en un navegador y poner ***https://10.1.2.67:10000*** y nos sale el inicio de credenciales de webmin



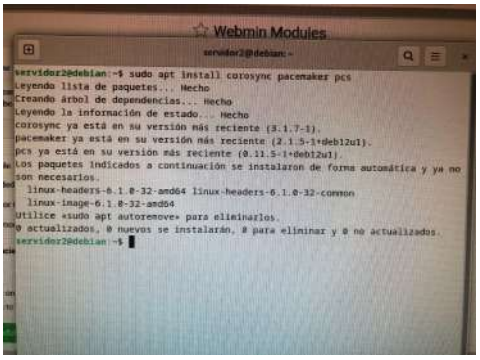
Ahora entramos en webmin con las credenciales de un usuario del sistema con privilegios



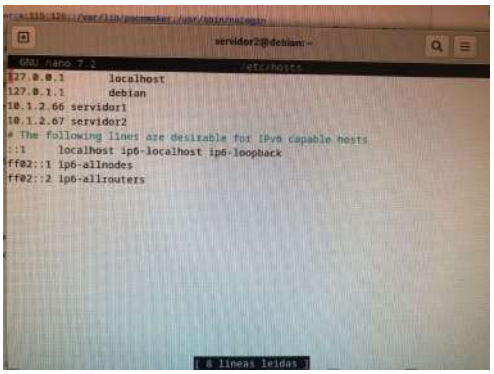
También instalamos los paquetes necesarios para habilitar **autenticación en dos factores (2FA)** en el sistema Linux mediante **Google Authenticator** con sudo apt install libauthen-oath-perl libpam-google-authenticator -y



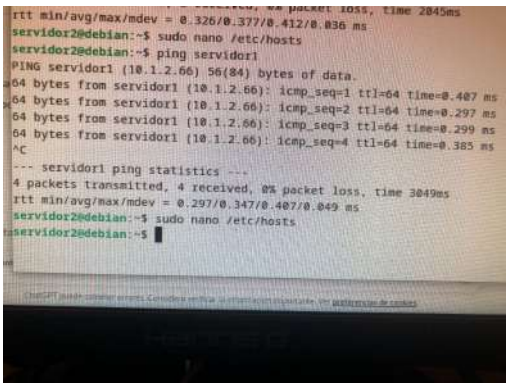
Ahora hacemos ***apt install corosync pacemaker pcs*** para instalar corosync y pacemaker



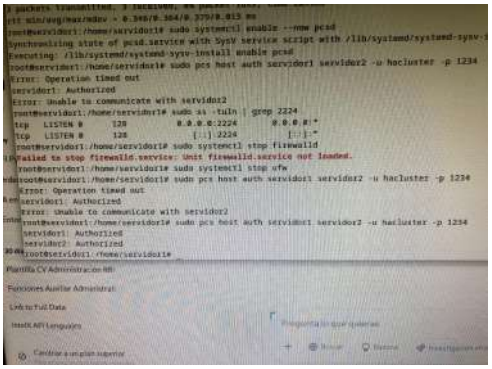
El archivo ***/etc/hosts*** lo modificamos añadiendo las ips de ambos servidores: 10.1.2.66 servidor1 y 10.1.2.67 servidor2



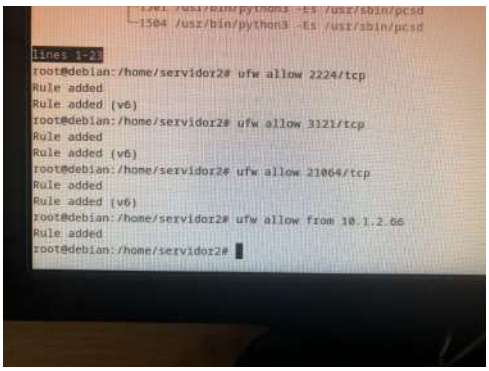
A continuación hacemos un ping ***ping servidor1*** para comprobar que se ven ambos servidores



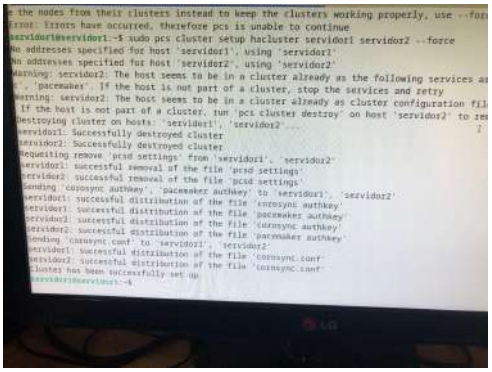
También hacemos un ***systemctl enable - - now pcsd***, que activa el demonio pcsd al arranque y lo inicia de inmediato. Ahora hacemos un ***pcs host auth servidor1 servidor2 -u hacluster -p 1234*** para autenticar desde el nodo actual hacia ambos nodos (servidor1 y servidor2) usando el usuario hacluster que hemos creado previamente con su contraseña 1234. Además crea claves compartidas en ***/var/lib/pcsd/known-hosts*.**



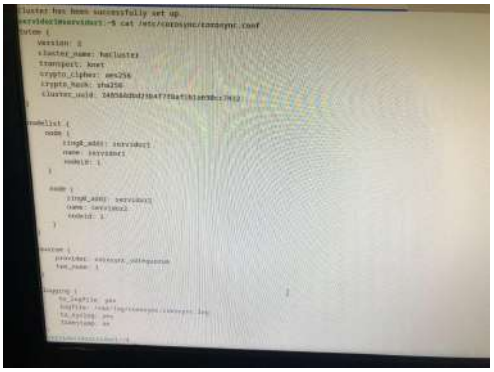
Ahora para asegurarnos, volvemos a habilitar los puertos tcp 2224, 3121 y 21064 con ufw allow puerto/tcp y también hacemos un ufw allow from 10.1.2.66, que permite todo el tráfico entrante desde la IP 10.1.2.66, sin restringir el puerto o protocolo.



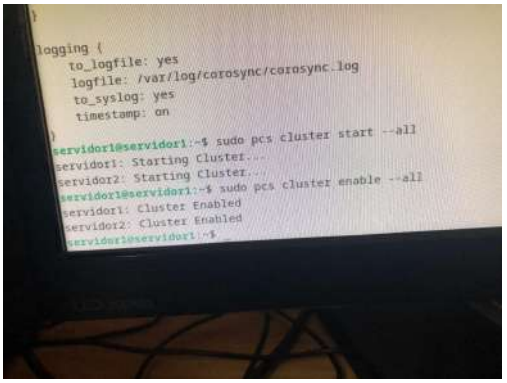
Ahora creamos el cluster con ***sudo pcs cluster setup hacluster servidor1 servidor2 - -force***



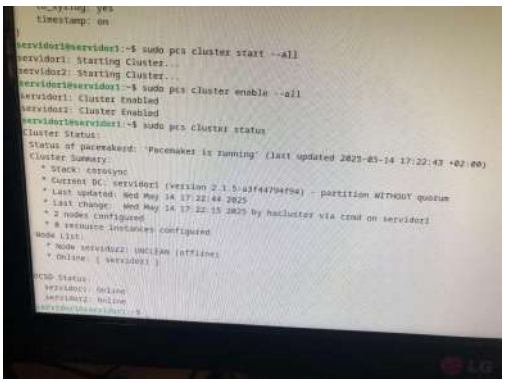
Comprobamos el archivo /etc/corosync/corosync.conf para comprobar su configuración



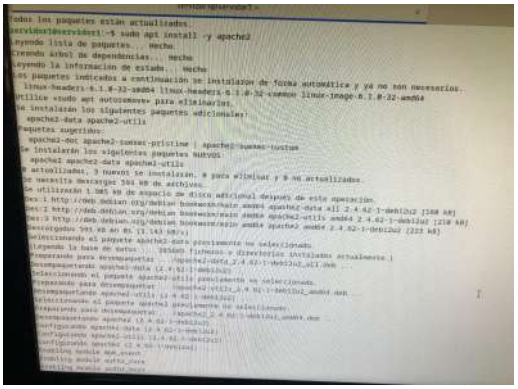
Ahora ***sudo pcs cluster start - -all*** *y* ***sudo pcs cluster enable - -all***



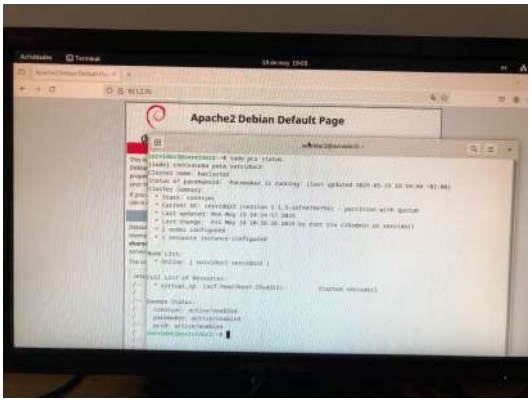
Ahora ***sudo pcs cluster status***



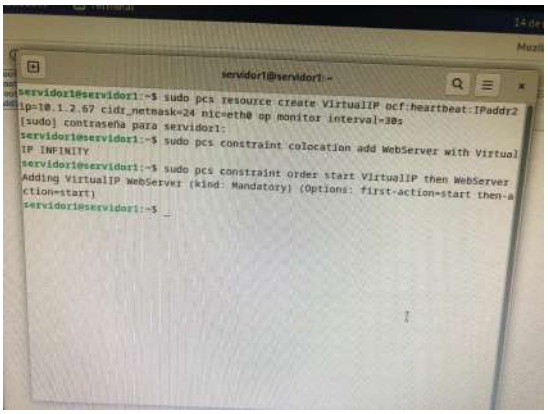
Ahora hacemo un ***apt install -y apache2***



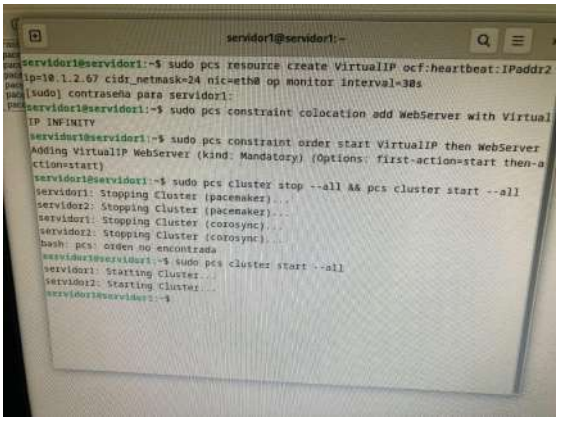
Comprobamos el servicio de apache entrando en http://10.1.2.70 (que es la ip virtual que hemos configurado previamente y mostraremos después) y nos saldria el inicio de apache



Ahora crearemos una IP virtual con ***sudo pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=10.1.2.70 cidr_netmask=24 nic=eth0 op monitor interval=30s***
A continuacion ***sudo pcs constraint colocation add WebServer with VirtualIP INFINITY***
Ahora ***sudo pcs constraint order start VirtualIP then WebServer***



Ahora ***sudo pcs cluster stop - -all*** y despues ***sudo pcs cluster start - -all***



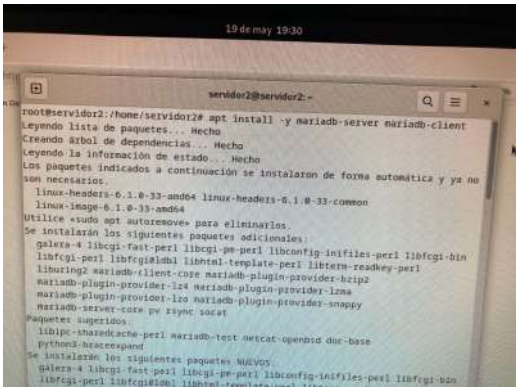
COMPROBACIÓN DE FUNCIONAMIENTO DEL CLUSTER/DISPONIBILIDAD

Ahora hacemos la comprobación de apagar un equipo, y en el otro hacemos un sudo pcs status y nos sale que nuestro clúster sigue activo en el servidor2, el 1 offline y la ip virtual funcionando correctamente. También los servicios de corosync, pacemaker y el pcsd nos salen que están active y enabled. En el siguiente video tenemos una prueba gráfica de ello:

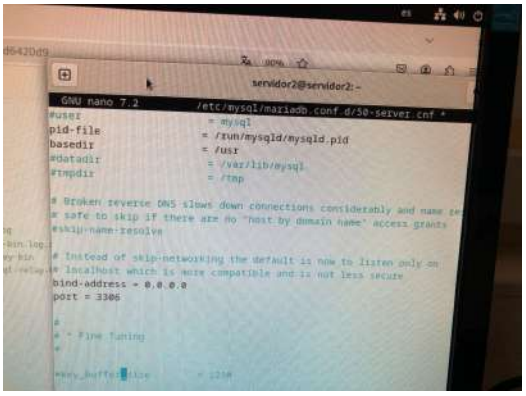
ENLACE VIDEO COMPROBACIÓN: https://youtu.be/7PrxMzBsLhk

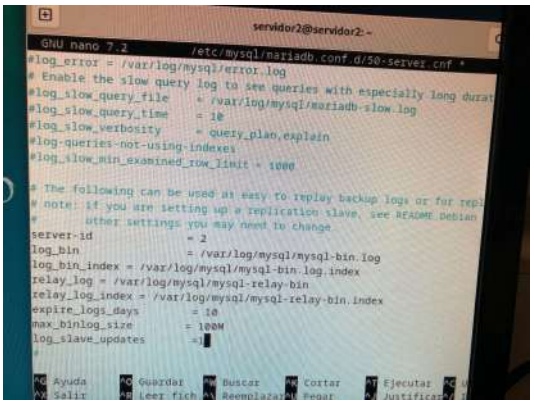
Aquí concluiría el servicio de alta disponibilidad de nuestro clúster. Como añadido vamos a proceder a la instalación-configuración de mariadb, para hacer una replicación y un balanceo de carga.

Instalamos MariaDB, que es un sistema para guardar y gestionar bases de datos. Instala tanto el servidor (que guarda las bases de datos) como el cliente (que permite conectarse y trabajar con ellas). ***apt install -y mariadb-server mariadb-client***

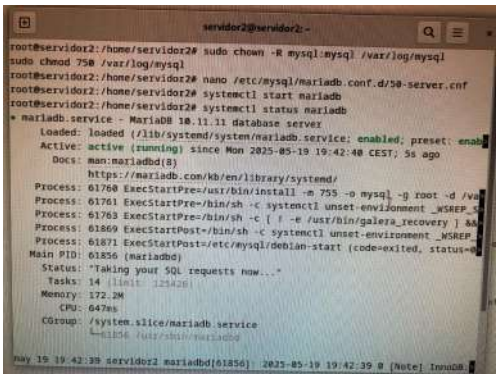


Configuramos el archivo ***/etc/mysql/mariadb.conf.d/50-server.cnf***

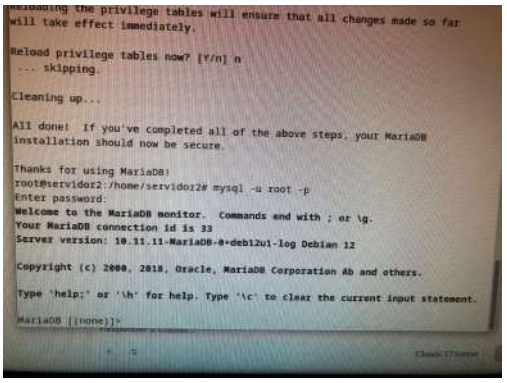




Con los siguientes comandos preparamos y arrancamos el servicio de MariaDB:
-sudo chown -R mysql:mysql /var/log/mysql: Cambia el dueño de la carpeta de logs de MariaDB para que solo el usuario mysql pueda acceder y escribir ahí
-sudo chmod 750 /var/log/mysql*:* Da permisos para que solo el dueño (mysql) pueda leer, escribir y entrar; el grupo puede entrar, pero el resto no tiene acceso.
-systemctl start mariadb: Inicia el servicio de MariaDB (pone en marcha el servidor de bases de datos).
-systemctl status mariadb: Muestra el estado actual del servicio para confirmar que está funcionando correctamente.

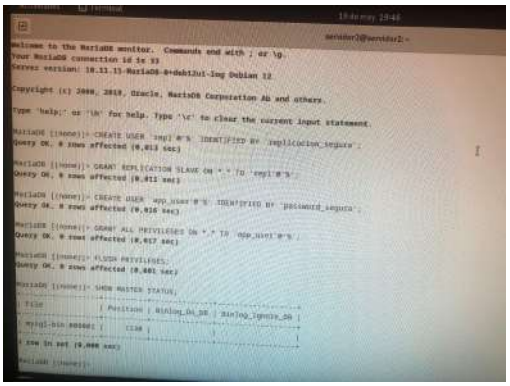


Iniciamos sesión en MariaDB como el usuario root, solicitando contraseña con ***mysql -u root -p***

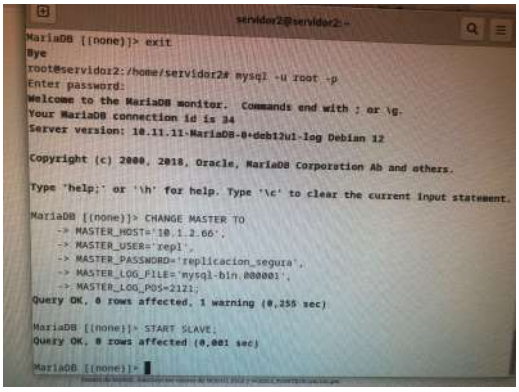


Ahora introducimos los siguientes comandos:

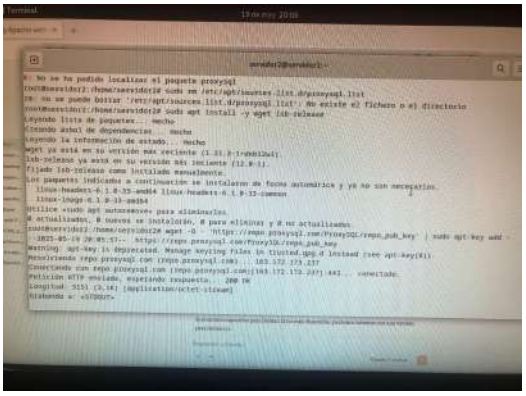
CREATE USER → Crea un nuevo usuario en MariaDB.
GRANT REPLICATION → Da al usuario permisos para replicar datos entre servidores (replicación).
GRANT ALL PRIVILEGES → Otorga todos los permisos sobre una base de datos o tabla.



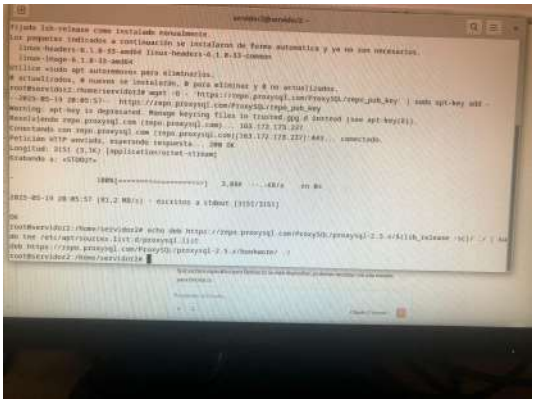
CHANGE MASTER TO → Configura los parámetros de conexión del esclavo al maestro para la replicación.
START SLAVE → Inicia el proceso de replicación en el servidor esclavo.



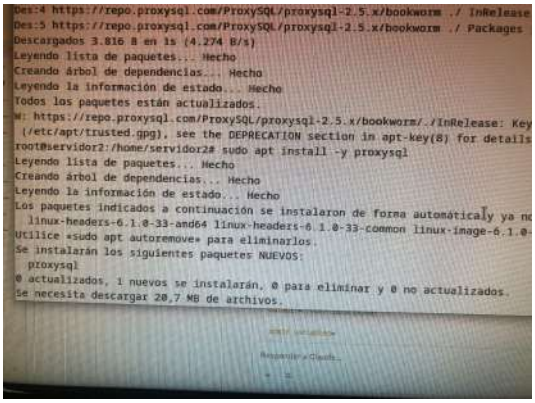
sudo rm /etc/apt/sources.list.d/proxysql.list: Elimina el archivo de configuración del repositorio de ProxySQL.
***wget -O - 'https://repo.proxysql.com/ProxySQL/repo_pub_key' | sudo apt-key add ---2025-05-19 20:05:57– https://repo.proxysql.com/ProxySQL/repo_pub_key*:** Descarga la clave pública del repositorio de ProxySQL y la añade al sistema para poder verificar la autenticidad de los paquetes que se instalen desde ese repositorio.



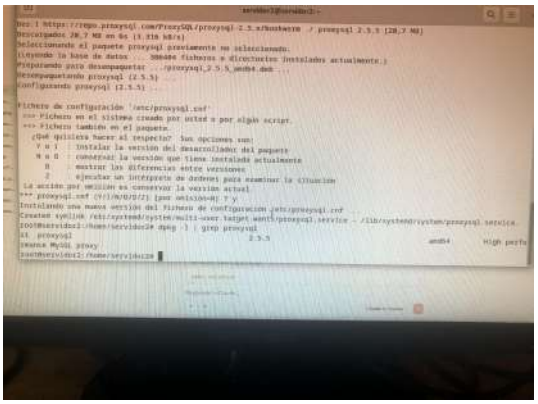
echo deb https://repo.proxysql.com/ProxySQL/proxysql-2.5.x/$(lsb_release -sc)/ ./ | sudo tee /etc/apt/sources.list.d/proxysql.list: Este comando añade el repositorio oficial de ProxySQL versión 2.5 para nuestra versión de Debian al sistema, creando un nuevo archivo .list en /etc/apt/sources.list.d/ para que apt pueda encontrar e instalar ProxySQL desde ahí.



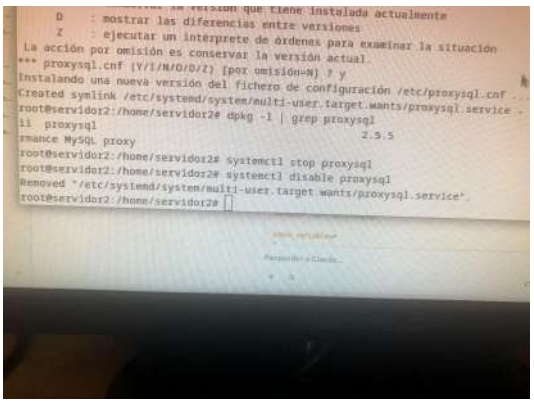
apt install -y proxysql: Este comando instala ProxySQL automáticamente sin pedir confirmación, usando el repositorio que acabamos de añadir.



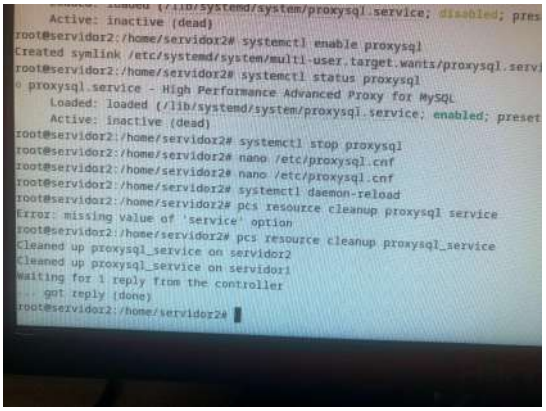
dpkg -l | grep proxysql: Este comando muestra la lista de paquetes instalados en el sistema y filtra aquellos que contienen la palabra "proxysql", para verificar que ProxySQL se ha instalado correctamente



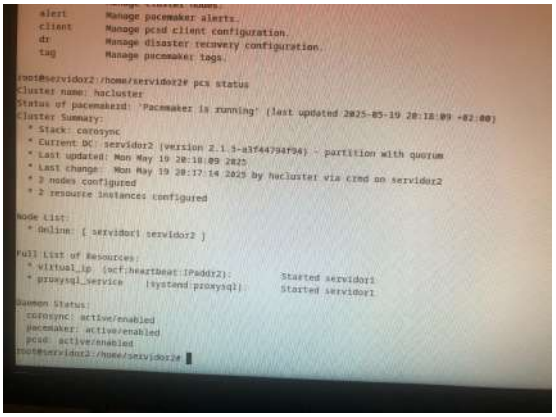
systemctl stop/disable proxysql: Estos comandos detienen el servicio de ProxySQL (systemctl stop proxysql) y evitan que se inicie automáticamente al arrancar el sistema (systemctl disable proxysql).



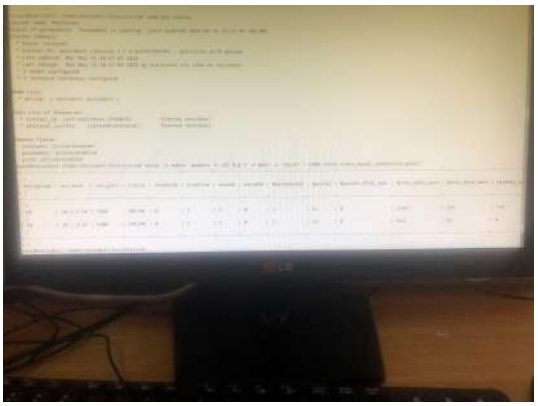
systemctl enable proxysql: habilita ProxySQL para que se inicie automáticamente al arrancar el sistema.
systemctl daemon-reload: recarga las configuraciones de servicios.
pcs resource cleanup proxysql_service: limpia los errores anteriores del recurso proxysql_service en el clúster, permitiendo que vuelva a iniciarse correctamente.



pcs status: muestra el estado actual del clúster, incluyendo los nodos, recursos, su ubicación y si están activos o inactivos. Es útil para comprobar rápidamente si todo funciona correctamente

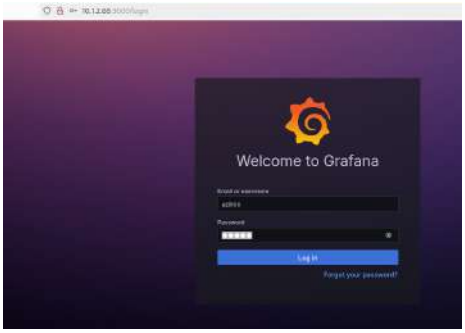


mysql -u admin -padmin -h: intenta conectar al servidor MariaDB o MySQL usando el usuario admin, con la contraseña admin, y especificando el host después del -h.



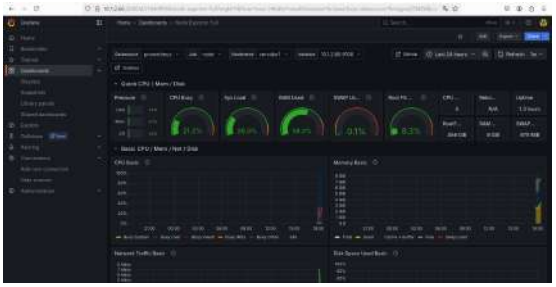
mysql -u root -p -e "SHOW SLAVE STATUS\G": se conecta a MySQL como usuario root, pide la contraseña (-p), y ejecuta directamente la consulta SHOW SLAVE STATUS\G, que muestra de forma detallada (una línea por campo) el estado de replicación del esclavo (replica), útil para verificar si está funcionando correctamente.

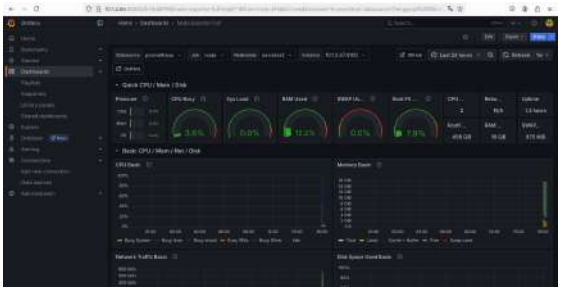
Entramos en **http://10.1.2.66:3000** en el navegador y nos lleva a la interfaz web de Grafana, que por defecto usa el puerto 3000. Ahí podemos iniciar sesión y empezar a configurar la monitorización del clúster y los servicios como MariaDB, ProxySQL, etc.



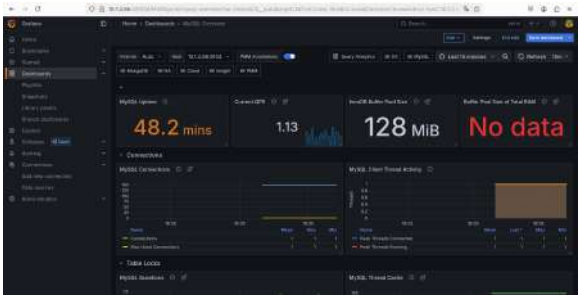
Ahora entramos en los **dashboards** de Servidor1 y Servidor2 dentro de Grafana, donde podremos visualizar en tiempo real métricas clave como:

Uso de CPU y RAM, Espacio en disco, Estado de los servicios (como MariaDB, ProxySQL, etc.), Tráfico de red, Latencias o errores en los servicios.
Estos dashboards nos permiten comparar el comportamiento de ambos nodos y detectar fácilmente si alguno falla o está sobrecargado. También es muy útil para validar el correcto funcionamiento del clúster y del balanceo de carga.



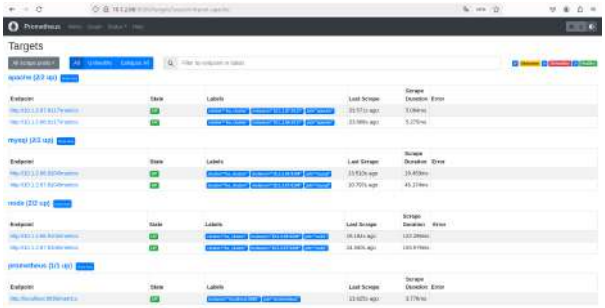


El panel **MySQL Overview** sirve para mostrar de forma rápida y visual el estado y rendimiento general de nuestra base de datos MySQL, incluyendo métricas clave como conexiones activas, consultas por segundo, uso de CPU, y estado de las tablas. Esto ayuda a supervisar la salud y el rendimiento de MySQL en tiempo real para detectar posibles problemas o cuellos de botella.

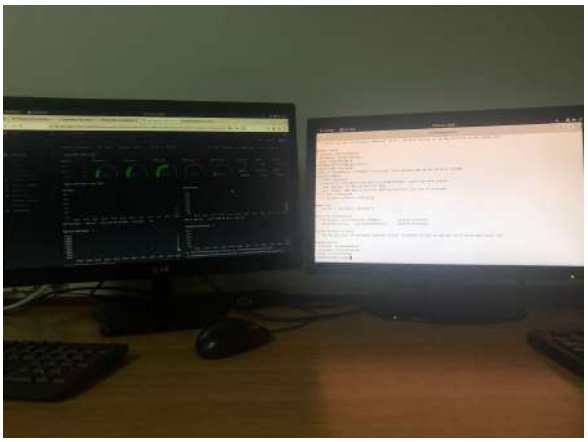


El panel **Apache Overview** en Grafana ofrece una visión rápida del estado y rendimiento del servidor Apache, mostrando métricas esenciales como el número de conexiones activas, solicitudes por segundo, uso de recursos (CPU, memoria), y estado de los procesos. Esto permite monitorear la salud del servidor web y detectar problemas de rendimiento o tráfico anómalo en tiempo real.

Por último veremos los **targets de monitoreo en grafana** que tiene nuestro clúster: Apache, mysql, node y prometheus



SETUP FINAL, SERVICIO MONTADO EN FISICO Y FUNCIONANDO



9.4. Glosario de Términos

- **Alta Disponibilidad (HA):** Enfoque que garantiza un nivel acordado de rendimiento operativo, generalmente el tiempo de actividad, durante un período superior al normal.
- **Balanceo de Carga:** Distribución del tráfico de red entre varios servidores para optimizar el uso de recursos.
- **Clúster:** Grupo de servidores que trabajan juntos como un sistema único para proporcionar alta disponibilidad, balanceo de carga o ambos.
- **Conmutación por error (Failover):** Proceso de conmutación automática a un sistema redundante cuando el sistema principal falla.
- **Corosync:** Sistema de comunicación entre nodos de un clúster, encargado de la detección de fallos y la comunicación entre nodos.
- **IP Virtual (VIP):** Dirección IP que puede migrar entre diferentes nodos de un clúster.
- **MariaDB:** Sistema de gestión de bases de datos relacional, bifurcación de MySQL.
- **Nodo:** Servidor individual dentro de un clúster.
- **Pacemaker:** Gestor de recursos de clúster que monitoriza los recursos y los migra entre nodos según sea necesario.
- **PCS:** Pacemaker/Corosync Configuration System, herramienta para configurar y gestionar clústeres.
- **ProxySQL:** Proxy para MySQL y MariaDB que permite balanceo de carga, caché de consultas y otras funcionalidades avanzadas.
- **Quórum:** Número mínimo de nodos que deben estar operativos para que el clúster pueda tomar decisiones válidas.
- **Replicación Maestro-Maestro:** Configuración donde dos o más servidores de base de datos pueden aceptar escrituras y replicar los cambios entre sí.
- **STONITH:** "Shoot The Other Node In The Head", mecanismo para forzar el reinicio de un nodo que no responde.
- **Split-Brain:** Situación en la que los nodos de un clúster pierden comunicación entre sí y ambos creen ser el nodo activo.

9.5. Webgrafía

Introducción a las Fuentes Consultadas

La presente webgrafía recoge las principales fuentes de información utilizadas durante el desarrollo e implementación de este proyecto de clúster de alta disponibilidad. Dada la naturaleza técnica y en constante evolución de las tecnologías empleadas, ha sido fundamental recurrir a documentación actualizada y recursos especializados.

La selección de fuentes se ha realizado priorizando:

- **Documentación oficial**: Manuales y guías de los desarrolladores de las herramientas utilizadas (Pacemaker, Corosync, MariaDB, ProxySQL, etc.), que proporcionan la información más precisa y actualizada.
- **Recursos técnicos especializados**: Artículos, tutoriales y casos de uso publicados por profesionales y organizaciones reconocidas en el ámbito de la administración de sistemas.
- **Comunidades técnicas**: Foros y plataformas donde expertos comparten experiencias, soluciones a problemas específicos y buenas prácticas.
- **Publicaciones académicas**: Estudios y análisis sobre alta disponibilidad, balanceo de carga y configuraciones de clústeres en entornos productivos.

La consulta de estas fuentes ha sido esencial no solo para la implementación inicial del proyecto, sino también para la resolución de incidencias, optimización del rendimiento y la implementación de medidas de seguridad adecuadas.

A continuación, se presenta la relación detallada de recursos web consultados, organizados por categorías según la tecnología o componente al que hacen referencia, incluyendo la fecha de última consulta para garantizar la trazabilidad de la información.

https://www.sysadminok.es/blog/administracion-de-sistemas/alta-disponibilidad-con-pacemaker-y-corosync-guia-completa/

https://www.maquinasvirtuales.eu/alta-disponibilidad-en-cluster-linux-con-pacemaker/

https://wiki.debian.org/Debian-HA/ClustersFromScratch

https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/configuring_and_managing_high_availability_clusters/assembly_getting-started-with-pacemaker-configuring-and-managing-high-availability-clusters

https://puerto53.wordpress.com/2019/06/23/tutorial-pacemaker-con-drbd/

https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/6/html/configuring_the_red_hat_high_availability_add-on_with_pacemaker/ch-clusteradmin-haar

https://www.netiq.com/es-es/documentation/sentinel-81/install/data/b156bqhs.html

https://www.capa9.net/temas/gu%C3%ADa-b%C3%A1sica-para-armar-tu-propio-cl%C3%BAster-linux.1118076/

10. CONCLUSIONES

Valoración General del Proyecto

La implementación del clúster de alta disponibilidad en Debian 12 ha resultado en un éxito rotundo, cumpliendo plenamente con los objetivos planteados inicialmente. Este proyecto ha demostrado la viabilidad y efectividad de las soluciones de código abierto para garantizar la continuidad operativa de servicios críticos en entornos empresariales.

Logros Alcanzados

Cumplimiento de Objetivos Técnicos

El proyecto ha conseguido implementar con éxito todos los componentes planificados:

-**Alta Disponibilidad Efectiva**: Se ha logrado un tiempo de failover entre 15-25 segundos, cumpliendo con el objetivo de mantener la disponibilidad por debajo de los 30 segundos establecidos.

-Balanceo de Carga Funcional: ProxySQL ha demostrado distribuir eficientemente las consultas entre ambos nodos MariaDB, optimizando el rendimiento del sistema.

-**Integridad de Datos Garantizada**: La replicación maestro-maestro de MariaDB ha funcionado sin fallos, manteniendo la consistencia de datos en ambos nodos durante todas las pruebas realizadas.

-**Monitorización Completa**: La implementación de Prometheus y Grafana ha proporcionado visibilidad total del estado del clúster, permitiendo la detección proactiva de posibles problemas.

Aspectos Técnicos Destacables

La configuración híbrida activo/activo para bases de datos y activo/pasivo para servicios web ha demostrado ser una estrategia acertada, maximizando el uso de recursos mientras

se mantiene la simplicidad operativa. La implementación de una IP virtual ha simplificado considerablemente el acceso a los servicios para los clientes finales.

Aprendizajes y Competencias Desarrolladas

Conocimientos Técnicos Adquiridos

Este proyecto ha permitido profundizar en tecnologías fundamentales para la administración de sistemas:

-**Gestión de Clústeres**: Dominio completo de Pacemaker, Corosync y PCS para la orquestación de recursos.

-**Bases de Datos Distribuidas**: Comprensión profunda de la replicación MariaDB y sus implicaciones en entornos de alta disponibilidad.

-**Balanceo de Carga**: Experiencia práctica con ProxySQL para la distribución inteligente de consultas.

-**Monitorización Avanzada**: Implementación de stacks completos de observabilidad con Prometheus y Grafana.

Habilidades Profesionales

La metodología iterativa aplicada ha reforzado competencias esenciales en administración de sistemas, desde la planificación inicial hasta la documentación final, pasando por la resolución de problemas complejos y la optimización del rendimiento.

Impacto y Aplicabilidad

Valor Empresarial

La solución implementada demuestra cómo tecnologías de código abierto pueden proporcionar niveles de disponibilidad comparables a soluciones comerciales costosas, ofreciendo:

-**Reducción de Costes**: Eliminación de licencias propietarias manteniendo funcionalidad empresarial.

-**Flexibilidad Operativa**: Capacidad de realizar mantenimientos sin afectar la disponibilidad del servicio.

-**Escalabilidad**: Arquitectura preparada para crecer según las necesidades del negocio.

Reflexión Final

Este Trabajo de Fin de Ciclo ha demostrado que la combinación de planificación rigurosa, implementación metodológica y documentación exhaustiva puede resultar en soluciones técnicas de alta calidad. La experiencia adquirida no se ha limitado a los aspectos puramente técnicos, sino que también nos ha permitido desarrollar competencias en gestión de proyectos, resolución de problemas complejos y documentación técnica profesional

El éxito del proyecto valida la importancia de las tecnologías de alta disponibilidad en el panorama actual de infraestructuras críticas, posicionando estas competencias como fundamentales para cualquier profesional en administración de sistemas informáticos en red.

La implementación completa, desde la instalación base hasta la monitorización avanzada, constituye un ejemplo práctico de cómo abordar proyectos de infraestructura crítica con rigor técnico y visión empresarial, preparando el terreno para futuras especializaciones en este campo en constante evolución.

AUTORES: Miguel Ángel López Alfaya y Juan Rodríguez Castellano

FECHA: Junio 2025

CURSO: 2º ASIR - Administración de Sistemas Informáticos en Red

AÑO ACADÉMICO: 2024-2025