
# Operaciones de mantenimiento en clúster HA de MariaDB con *Zero-Downtime*

**Javier López López**
Grado de Ingeniería Informática
Administración de redes y sistemas operativos

**Jaume Jofre Bravo**
**David Bañeres Besora**
**Montse Serra Vizern**

Junio 2024

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *Operaciones de mantenimiento en clúster HA de MariaDB con zero-downtime* |
| **Nombre del autor:** | *Javier López López* |
| **Nombre del consultor/a:** | *Jaume Jofre Bravo* |
| **Nombre del PRA:** | *David Bañeres Besora* |
| **Fecha de entrega (mm/aaaa):** | 05/2024 |
| **Titulación:** | *Grado de Ingeniería Informática* |
| **Área del Trabajo Final:** | *Administración de redes y sistemas operativos* |
| **Idioma del trabajo:** | *Castellano* |
| **Palabras clave** | *MariaDB, Galera, MaxScale* |

**Resumen del Trabajo (máximo 250 palabras):** *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Actualmente las empresas, con motivo de sus actividades comerciales, la competencia y exposición pública, no pueden permitirse disponer de ventanas con periodos de no disponibilidad de sus sistemas. Este sería un grave contratiempo que tendría consecuencias en el negocio y, a su vez, es necesario para poder seguir ofreciendo un servicio adecuado, ya que estos sistemas y servicios evolucionan constantemente y requieren de mantenimiento.

Este proyecto tratará de poner solución a esta problemática, al menos en cuanto a uno de los sistemas fundamentales en todo negocio: las bases de datos y más concretamente, el SGBD MariaDB. Para ello, se ha montado toda la infraestructura necesaria y se han ejecutado con éxito y con la menor pérdida de servicio posible, un par de laboratorios que ponen solución a dicho problema.

Se ha montado, en *Google Cloud Platform,* una infraestructura en alta disponibilidad de MariaDB con Galera y reparto de peticiones de lectura y escritura de los clientes hacia los *backends*, con MaxScale y Keepalived, poniendo solución a las operaciones de mantenimiento del SGBD, como subidas de versión o mantenimientos menores y con pérdidas de servicio nulas o insignificantes. Por otro lado, también se ha propuesto una solución robusta para las copias de seguridad y restauración de las bases de datos, así como un remedio para la recuperación de los datos en cualquier momento del tiempo (PITR). Por último, se ha propuesto una futura línea de investigación para la

monitorización y el escalado horizontal de la plataforma.

**Abstract (in English, 250 words or less):**

Currently, companies, due to their commercial activities, competition, and public exposure, cannot afford to have periods of unavailability in their systems. This would be a serious setback with business consequences, and it is necessary to continue offering an adequate service, as these systems and services are constantly evolving and require maintenance.

This project aims to address this issue, at least concerning one of the fundamental systems in any business: databases, specifically the MariaDB DBMS. To achieve this, all necessary infrastructure has been set up, and a couple of labs have been successfully executed with minimal service loss, providing a solution to this problem.

A high-availability MariaDB infrastructure with Galera has been set up on Google Cloud Platform, distributing read and write requests from clients to the backends using MaxScale and Keepalived, addressing DBMS maintenance operations such as version upgrades or minor maintenance with zero or insignificant loss of service. Additionally, a robust solution for database backups and restores has been proposed, as well as a remedy for data recovery at any point in time (PITR). Finally, a future line of research has been proposed for monitoring and horizontal scaling of the platform.

# Índice de contenidos

# Índice de figuras

# Índice de tablas

# 1. Introducción

## 1.1. Contexto y justificación del Trabajo

Hoy en día las empresas y organizaciones, debido a la cada vez mayor exposición pública, la globalización y a sus actividades comerciales, tanto nacionales como internacionales, no pueden permitirse que sus sistemas y sus aplicaciones dejen de estar disponibles en ningún horario. Ya no existen las horas de baja actividad en la que poder hacer los mantenimientos de sus sistemas, necesarios por la cantidad de aplicaciones que mantienen y exponen públicamente. La continuidad del negocio se exige durante el mayor tiempo posible para no dañar la reputación de la empresa ni ocasionar pérdidas en la facturación y, es por ello, que el acceso a los datos debe tener una disponibilidad total o lo más cercano posible al 100%. Sin embargo, las aplicaciones que se nutren de estos datos están en continua evolución y, ya sea por requisitos de estas evoluciones o por vulnerabilidades de seguridad que afectan a los Sistemas Gestores de Bases de Datos (en adelante, SGBD), a veces hay que realizar mantenimientos e instalar parches o subidas de versión que mitiguen las vulnerabilidades encontradas por los departamentos de seguridad.

El objetivo de este trabajo de fin de grado (en adelante, TFG) es solucionar esta problemática, al menos en cuanto al SGBD se refiere, pudiendo realizar mantenimientos, incluso en horarios de máxima actividad, aunque siempre es recomendable abordarlos en horarios de menor impacto, y sin que la continuidad del negocio se resienta. Para ello, se plantea una arquitectura de alta disponibilidad para un SGBD de MariaDB, con redundancia de nodos multimáster, replicación síncrona, balanceo de carga, tolerancia a fallos y conmutación por *failover*, así como configuración manual de estados de mantenimiento sobre los nodos de un clúster en los que poder trabajar de forma individual y sin que afecte al servicio.

Tras la ejecución de la solución propuesta en este proyecto seremos capaces de realizar los siguientes pasos para el mantenimiento de un clúster de MariaDB sin que el servicio se resienta:

- Configuración del modo mantenimiento en uno de los nodos secundarios de un clúster, ya que el considerado como primario será el que siga recibiendo las operaciones de escritura.
- Liberación de las conexiones. Tras la activación del modo mantenimiento, este nodo dejará de recibir nuevas conexiones e irá finalizando habitualmente las operaciones que tuviera hasta ese momento.
- Tras la liberación de las conexiones se realizaría el mantenimiento oportuno o actualización del software con la seguridad de que no seguirá recibiendo operaciones de los clientes.
- Se desactiva el modo mantenimiento y se sincronizará automáticamente con otro de los nodos activos hasta alcanzar el mismo estado que el resto de los miembros del clúster.

- Se promociona a primario para que el nodo actualizado sea el que reciba las operaciones de escritura.
- Por último, se actuará de igual forma en el resto de los nodos del clúster hasta que éste esté completamente actualizado.

La solución propuesta se evaluará en un laboratorio, donde al mismo tiempo se estará en todo momento el estado del servicio para comprobar que efectivamente no se resiente, operando con normalidad. También se mostrarán métricas y evidencias de la disponibilidad del servicio durante las operaciones de mantenimiento.

Por último, se plantean también dos objetivos secundarios: en el primero de ellos se ofrecerá una solución para puntos de restauración en el tiempo (Point-in-Time Recovery, PITR) del SGBD con objeto de poder recuperar una base de datos justo antes del momento en que se produce algún desastre o en cualquier otro momento que se desee. El segundo tratará sobre la escalabilidad horizontal para ampliar el número de nodos de un clúster de forma automática en caso de saturación, como podría ser en horarios de máxima carga de los sistemas o debido a campañas puntuales sobre alguno de los servicios, que pueden llegar a provocar que el número de conexiones se sobrepase.

El proyecto se realizará por completo en un entorno virtual de laboratorio en la nube de *Google Cloud Platform* (en adelante, GCP) con los requisitos mínimos de hardware y con software libre, no obstante, será totalmente escalable a un entorno real de producción, ya sea con máquinas físicas o virtuales o en la nube.

## 1.2. Objetivos del Trabajo

Los objetivos principales que se pretenden alcanzar con la elaboración de este trabajo son los siguientes:

- Configuración de un clúster del sistema de gestión de bases de datos MariaDB con replicación síncrona multimáster.
- Balanceo de carga con repartición de las operaciones de lectura hacia los nodos *standby* del clúster y enrutamiento de las operaciones de escritura hacia un único máster, considerado como primario.
- Solución de *zero-downtime* del SGBD para una operación de subida de versión de los nodos del clúster de MariaDB.

Para alcanzar estos objetivos se deberán realizar las siguientes tareas asociadas:

1. Preparación de un entorno de laboratorio donde montar los servicios. Para el entorno de laboratorio se opta por una solución en la nube de Google Cloud, no obstante, será perfectamente escalable a cualquier otro entorno, ya sea *On-premise*, con virtualización local, contenedores Docker o cualquier otro entorno de *Cloud Computing* como AWS, Azure, etc.

2. Instalación y configuración de un mínimo de tres nodos del SGBD MariaDB sobre una distribución Linux, Debian GNU/Linux 12 (*bookworn*).
3. Implementación del clúster de alta disponibilidad de MariaDB con Galera.
4. Instalación y configuración de un proxy con MaxScale, que realizará el balanceo de la carga entre los nodos y distribuirá las operaciones de lectura y escritura entre los miembros del *pool*.
5. Clusterización del *software* de proxy con un nodo activo y otro en *standby* con mecanismo de *failover*.
6. Configuración de la IP flotante, o *Virtual* IP (en adelante, VIP) para la conexión de los clientes con Keepalived.

Otros objetivos secundarios, que se acometerán sólo en el caso de contar con el tiempo suficiente, son los que se detallan a continuación:

- Solución para *Point-in-Time Recovery* (en adelante, PITR), punto de restauración en el tiempo, con la herramienta Mariabackup y binlogs.
- Solución de escalabilidad horizontal automática en caso de saturación de las conexiones en los nodos del clúster.

### 1.3. Enfoque y método seguido

El objetivo principal del trabajo es el de mostrar el potencial de una arquitectura de clúster en alta disponibilidad para un SGBD como MariaDB con las herramientas Galera, MaxScale y Keepalived y atender a necesidades cotidianas del SGBD como labores de mantenimiento sin que el servicio se resienta con un porcentaje de disponibilidad lo más cercano posible al 100%.

Para ello, se ha realizado, en primer lugar, un profundo análisis a las herramientas mencionadas y teniendo en cuenta otras alternativas, llegando a la conclusión de que con una correcta implementación de éstas es posible alcanzar cada uno de los objetivos propuestos y dar una respuesta a las soluciones demandadas.

Tras el análisis de las soluciones que ofrece el mercado de software libre para atender la demanda, se ha realizado un estudio de las diferentes posibilidades con las que contamos para desarrollar un laboratorio en el que poder montar toda la infraestructura. Es por ello, por lo que se han valorado las siguientes opciones:

- Entorno de virtualización local con VirtualBox o Vmware.
- Contenedores Docker.
- Virtualización en la nube de Google (GCP), Amazon (AWS), Microsoft (Azure).

Descartadas las dos primeras opciones por falta de recursos en local, se ha optado finalmente por montar la infraestructura de máquinas en el *cloud* de

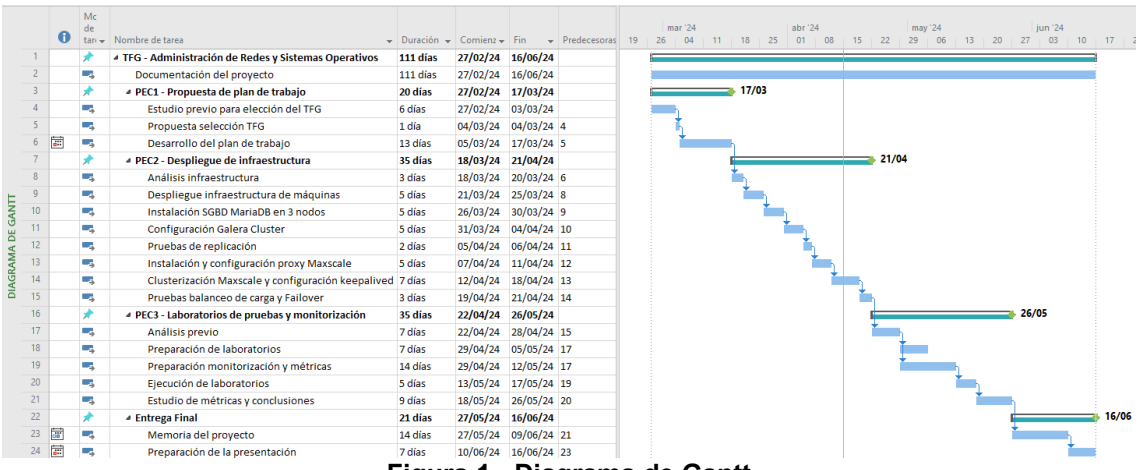
Google, tras una documentación previa y análisis de que la plataforma es capaz de atender todas las necesidades.

El método seguido ha sido el que se describe a continuación:

- **Análisis**, autoformación e investigación de la plataforma GCP.
- **Despliegue** de las instancias de máquinas virtuales (en adelante, VM, *virtual machine*).
- **Instalación** y configuración de la infraestructura de clúster en alta disponibilidad para MariaDB en las VM.
- **Instalación** y configuración de la infraestructura de proxy para el balanceo de carga y reparto de transacciones de lectura/escritura con MaxScale e IP flotante con Keepalived.
- **Copias de seguridad** de cada uno de los objetivos alcanzados.
- **Ejecución, testeo y monitorización** de las soluciones propuestas.
- **Documentación** de cada uno de los procedimientos llevados a cabo.

## 1.4. Planificación del Trabajo

El proyecto tendrá una duración de 111 días, dando comienzo el 27/02 con la propuesta del trabajo a desarrollar y finalizando el 16/06 con la entrega de la memoria y la presentación del proyecto. Para la planificación de este, se ha preparado el siguiente plan de trabajo representado por el diagrama de Gantt, que estará condicionado por las fechas de entrega de cada una de las PEC, que serán los puntos de control, y la entrega final establecidos como hitos.




| | Nombre de tarea | Duración | Comienz | Fin | Predecesoras |
|---|---|---|---|---|---|
| 1 | **TFG - Administración de Redes y Sistemas Operativos** | **111 días** | **27/02/24** | **16/06/24** | |
| 2 | Documentación del proyecto | 111 días | 27/02/24 | 16/06/24 | |
| 3 | **PEC1 - Propuesta de plan de trabajo** | **20 días** | **27/02/24** | **17/03/24** | |
| 4 | Estudio previo para elección del TFG | 6 días | 27/02/24 | 03/03/24 | |
| 5 | Propuesta selección TFG | 1 día | 04/03/24 | 04/03/24 | 4 |
| 6 | Desarrollo del plan de trabajo | 13 días | 05/03/24 | 17/03/24 | 5 |
| 7 | **PEC2 - Despliegue de infraestructura** | **35 días** | **18/03/24** | **21/04/24** | |
| 8 | Análisis infraestructura | 3 días | 18/03/24 | 20/03/24 | 6 |
| 9 | Despliegue infraestructura de máquinas | 5 días | 21/03/24 | 25/03/24 | 8 |
| 10 | Instalación SGBD MariaDB en 3 nodos | 5 días | 26/03/24 | 30/03/24 | 9 |
| 11 | Configuración Galera Cluster | 5 días | 31/03/24 | 04/04/24 | 10 |
| 12 | Pruebas de replicación | 2 días | 05/04/24 | 06/04/24 | 11 |
| 13 | Instalación y configuración proxy Maxscale | 5 días | 07/04/24 | 11/04/24 | 12 |
| 14 | Clusterización Maxscale y configuración keepalived | 7 días | 12/04/24 | 18/04/24 | 13 |
| 15 | Pruebas balanceo de carga y Failover | 3 días | 19/04/24 | 21/04/24 | 14 |
| 16 | **PEC3 - Laboratorios de pruebas y monitorización** | **35 días** | **22/04/24** | **26/05/24** | |
| 17 | Análisis previo | 7 días | 22/04/24 | 28/04/24 | 15 |
| 18 | Preparación de laboratorios | 7 días | 29/04/24 | 05/05/24 | 17 |
| 19 | Preparación monitorización y métricas | 14 días | 29/04/24 | 12/05/24 | 17 |
| 20 | Ejecución de laboratorios | 5 días | 13/05/24 | 17/05/24 | 19 |
| 21 | Estudio de métricas y conclusiones | 9 días | 18/05/24 | 26/05/24 | 20 |
| 22 | **Entrega Final** | **21 días** | **27/05/24** | **16/06/24** | |
| 23 | Memoria del proyecto | 14 días | 27/05/24 | 09/06/24 | 21 |
| 24 | Preparación de la presentación | 7 días | 10/06/24 | 16/06/24 | 23 |

**Figura 1 - Diagrama de Gantt**

En la tabla siguiente se describen los diferentes hitos que se pretenden alcanzar a lo largo de todo el TFG y se definen los objetivos, teniendo en cuenta la documentación como una tarea global que tendrá recorrido durante todo el proyecto:

| PEC1 – Propuesta del plan de trabajo | Inicio: 27/02/2024 \| Fin: 27/03/2024 |
|---|---|
| **Descripción** | |
| Valorar, analizar e investigar múltiples posibilidades para la elaboración del TFG y redactar la propuesta del plan de trabajo con el proyecto acordado con el tutor de la asignatura. | |
| **Objetivos** | |
| Se habrá decidido el proyecto objetivo del TFG y se habrá realizado un documento de plan de trabajo desglosado con las tareas a ejecutar en las próximas semanas hasta la fecha hito de finalización. | |
| **PEC2 – Despliegue de la infraestructura** | **Inicio: 18/03/2024 \| Fin: 21/04/2024** |
| **Descripción** | |
| Preparar la infraestructura de máquinas y recursos en Google Cloud para poder ejecutar los laboratorios objetivos del proyecto. | |
| **Objetivos** | |
| Tras la ejecución de este hito se dispondrá de un total de 6 instancias VM en GCP, 3 de ellas destinadas a formar el clúster multimáster de Galera MariaDB con replicación síncrona, otras 2 para el clúster máster/standby de MaxScale con conmutación de IP por *failover* automático con Keepalived y una última que hará las funciones de cliente. | |
| **PEC3 – Laboratorios de pruebas y monitorización** | **Inicio: 22/04/2024 \| Fin: 26/05/2024** |
| **Descripción** | |
| Se implementarán los laboratorios definidos para alcanzar los objetivos propuestos. | |
| **Objetivos** | |
| Se dispondrá de un laboratorio en el que se pueda realizar un *upgrade* de versión de todos los miembros del clúster Galera MariaDB con evidencias de continuidad del servicio, así como de otros laboratorios secundarios del proyecto como la escalabilidad horizontal y PITR. | |
| **Entrega final – Memoria y presentación** | **Inicio: 27/05/2024 \| Fin: 16/06/2024** |
| **Descripción** | |
| Últimos retoques a la memoria del TFG y preparación del video de presentación. | |
| **Objetivos** | |
| Disponer de una memoria del proyecto donde se ponga solución a todas las cuestiones planteadas y un video de presentación resumiendo cada uno de los pasos más importantes que se han dado a lo largo del TFG. | |

**Tabla 1 - Plan de trabajo**

### 1.5. Breve sumario de productos obtenidos

Tras la finalización del proyecto dispondremos de un entorno de laboratorio, con requisitos mínimos, en la nube de Google Cloud, totalmente funcional y

escalable a cualquier ámbito de producción, ajustando recursos y parametrización, así como una memoria que detalla todo el proceso de instalación y configuración de los servicios y ejecución de los laboratorios objetivos del TFG y de un video de presentación destacando los pasos más relevantes.

- Productos *hardware* en la nube:
  - 5 instancias VM (*servers*)
    - Tipo: E2-micro x86-64
    - CPU: 2 vCPU (Intel(R) Xeon(R) CPU @ 2.20GHz)
    - Memoria: 1 GB
    - Disco: 1 disco persistente balanceado de 10 GB
  - 1 instancia VM (*client*)
    - Tipo: E2-micro x86-64
    - CPU: 2 vCPU (Intel(R) Xeon(R) CPU @ 2.20GHz)
    - Memoria: 1 GB
    - Disco: 1 disco persistente balanceado de 10 GB
- Software (*servers*)
  - OS: Debian GNU/Linux 12 (*bookworn*)
  - mariadb  Ver 15.1 Distrib 10.11.7-MariaDB
  - galera-4 26.4.16-deb12
  - maxscale 23.08.5~bookworm-1
  - keepalived 1:2.2.7-1+b2
- Software (*client*)
  - OS: Debian GNU/Linux 12 (*bookworn*)
  - apache-jmeter-5.6.3

### 1.6. Breve descripción de los otros capítulos de la memoria

Los siguientes capítulos de la memoria en orden numérico tendrán el siguiente contenido:

- **Capítulo 2**: se detallarán los conceptos teóricos de cada uno de los componentes a los que se hará referencia a lo largo del proyecto.
- **Capítulos 3 y 4**: instalación y configuración de las infraestructuras que se van a montar para el desarrollo de los laboratorios, con Galera Clúster y MaxScale.

- **Capítulo 5**: se desarrollará un plan de contingencia de toda la infraestructura a la que poder recurrir en caso de desastre para restablecerla.
- **Capítulo 6**: ejecución de los laboratorios, que son el objetivo principal de la elaboración de este proyecto.
- **Capítulo 7**: en este capítulo se plasmarán las conclusiones del TFG y se analizarán posibles trabajos futuros que se podrían abordar en una continuación del proyecto.
- **Los últimos 3 capítulos** contienen el glosario de términos, la bibliografía que se ha consultado para el desarrollo del proyecto y los anexos de la memoria.

# 2. Conceptos previos

## 2.1. MariaDB

### 2.1.1. Qué es MariaDB

MariaDB, en su versión *Community*, es uno de los servidores de base de datos relacionales y de código abierto, bajo licencia GPL (*General Public License*), más populares del mundo, descargada miles de millones de veces y la predeterminada en todas las distribuciones Linux, aunque también compatible con otros sistemas operativos como Windows y macOS, entre otros. Creada por desarrolladores originales de MySQL, compatible con MySQL y Oracle y garantizado que siempre será de código abierto. Existe también una versión *premium* denominada MariaDB *Enterprise Server* [1].

Entre sus características tenemos las siguientes:

- Compatibilidad con funciones modernas de SQL y cuyo dialecto es compatible con MySQL, pero ampliado con almacenamiento en columnas y otras funciones.
- Motores de almacenamiento como InnoDB, Aria, ColumnStore, MyRocks y otros motores de terceros.
- Características avanzadas como JSON, XML y consultas geoespaciales, entre otros.
- En cuanto a seguridad ofrece cifrado de datos en reposo y en tránsito, control de acceso a nivel de columna y autenticación PAM (*Pluggable Authentication Modules*).
- *Clustering* con Galera para la replicación y transferencias de estado.
- Alta disponibilidad con MaxScale como proxy para la conmutación por error.

### 2.1.2. Diferencias entre MariaDB y MySQL

Ambos son tecnologías de bases de datos de código abierto, utilizados para almacenar los datos en forma tabular con filas y columnas, lo que se conoce como un RDBMS (*Relational DataBase Management System*), un SGBD mejorado. MariaDB es una versión modificada de MySQL, desmarcada de su predecesor tras problemas de licencia y distribución con la compra de este último por Oracle Corporation en 2009. No obstante, aun llevando caminos de desarrollo separados desde entonces, tienen un alto grado de compatibilidad entre ambos, lo que facilita la migración entre ambos sistemas.

Sus diferencias más significativas quedan resumidas en la siguiente tabla por categorías:

| | MySQL | MariaDB |
|---|---|---|
| **Licencia** | GPL y comercial | GNU (GPL) |
| **Mantenimiento y desarrollo** | Oracle Corporation | Fundación MariaDB, comunidad |
| **Compatibilidad** | Menor compatibilidad con MariaDB | Alta compatibilidad con MySQL |
| **Motores de almacenamiento** | InnoDB, CSV, Federated, MyISAM, Merge y Federated | XtraDB, Aria, InnoDB, MariaDB ColumnStore, Memory, Cassandra y Connect |
| **Rendimiento y velocidad** | Menos consultas por segundo que MariaDB | Más rápido procesando consultas y replicación |
| **Otras características** | - No compatible con PL/SQL de Oracle<br>- InnoDB y AES para cifrado de datos en reposo<br>- Agrupación de subprocesos en ed. empresarial<br>- JSON como objetos binarios | - Compatible con PL/SQL de Oracle<br>- Cifrado de registro temporal y binario<br>- 200.000 conexiones a la vez (+ que MySQL)<br>- JSON en cadenas (LONGTEXT) |

**Tabla 2 - Comparativa MySQL y MariaDB** [2]

En términos de uso, acudiendo a las gráficas de búsquedas web globales de Google Trends [3], vemos como MySQL alcanzó su punto máximo entre 2004 y 2005 y luego fue decayendo.


**Figura 2 - Google Trends MySQL**

En contraposición vemos el crecimiento de MariaDB en cuanto al interés global, alcanzando el punto máximo en 2022.


**Figura 3 - Google Trends MariaDB**

## 2.2. Galera clúster

MariaDB Galera Cluster es un clúster multimáster síncrono para bases de datos relacionales como MariaDB. Disponible únicamente para sistemas operativos Linux y entre sus limitaciones destacamos que tan sólo es compatible con el motor de almacenamiento InnoDB.

Entre las características fundamentales que lo convierten en una solución robusta para la alta disponibilidad se podrían señalar las siguientes:

- **Replicación síncrona multimáster**: las actualizaciones en la base de datos se replican de forma síncrona entre todos los nodos que forman el clúster, actuando como maestros.
- **Lecturas y escrituras en cualquier nodo del clúster**, mejorando el rendimiento y la escalabilidad.
- **Tolerancia a errores**, detectando automáticamente y recuperándose de fallos en los *backends* sin interrumpir el servicio.
- **Integridad de los datos**, garantizando que las transacciones se han completado en todos los nodos o en ninguno.
- **Reparación automática de nodos desincronizados**. Galera Cluster tratará de sincronizar un nodo desactualizado por cualquier motivo sin intervención manual.
- **Topología flexible**. Permite configuración de topologías en anillo, árboles o mallas, adaptándose a cualquier tipo de infraestructura.

A continuación, en la figura 4, se observa una topología de arquitectura básica para un clúster de Galera con 3 nodos de MariaDB.

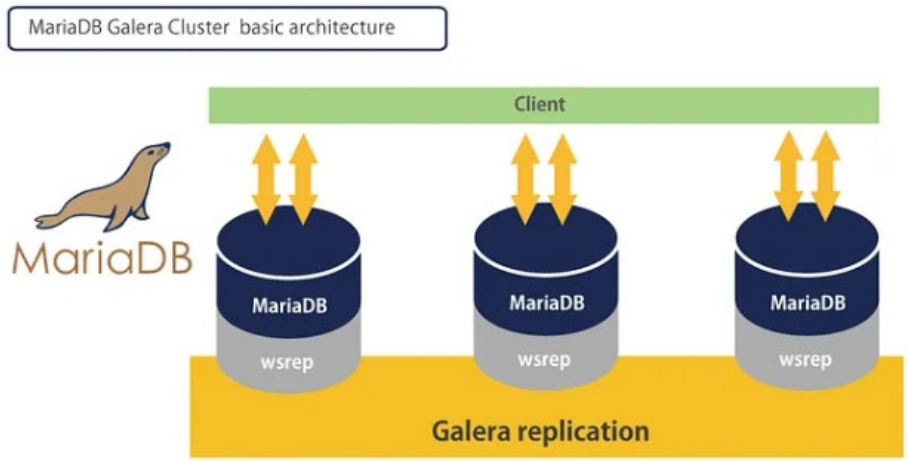



**Figura 4 - Arquitectura básica de MariaDB Galera Cluster** [5]

Galera se basa en el protocolo de replicación WSREP (*Write Set Replication*) para sincronizar los datos entre los nodos del clúster de forma síncrona. "*Write Set*" es el conjunto de operaciones de escritura que se ejecutan en uno de los nodos del clúster en un intervalo de tiempo configurado. Este conjunto de operaciones debe confirmarse en el mismo orden en todos los nodos del clúster antes de darse por completado.

Para entender como funciona un clúster de MariaDB con Galera es necesario comprender una serie de conceptos clave que detallaremos a continuación como Galera-Cache (gcache), *state transfers* (IST, SST), quorum (con Galera Arbitrator) y bootstrapping [6].

### 2.2.1. Galera-Cache

Galera-Cache desempeña un papel fundamental en la replicación de datos y para prevenir situaciones de "*split-brain*", caso en el que dos o más nodos de un clúster actúan por separado, dividiéndose en grupos aislados como partes de un componente primario y permaneciendo operativos.

Es una memoria caché compartida entre todos los nodos del clúster. Cada uno la utiliza para almacenar los registros binarios de las transacciones (*Write Sets*), que han sido recibidas y van a ser enviados al resto de nodos para su replicación. Cada nodo miembro tendrá un registro completo y actualizado de todas las transacciones que han sido replicadas en el clúster.

Esta caché es la que se utilizará también cuando un nodo se ha desconectado temporalmente del clúster y al unirse nuevamente necesita sincronizar su estado con el resto de los nodos.

El tamaño de la caché es configurable, determinando cuantos registros binarios de transacciones se almacenarán en la memoria, siendo muy importante un tamaño adecuado que garantice un rendimiento óptimo para evitar la pérdida de datos en caso de desconexiones de los nodos.

### 2.2.2. Estados de transferencia (SST, IST)

El tamaño del caché comentado en el punto anterior determinará el estado de transferencia (*State Transfer*) o aprovisionamiento automático de los nuevos nodos. Si el tiempo de desconexión del nodo que se une al clúster es inferior al número de registros almacenados en la memoria, entonces se realizará una transferencia incremental IST (*Incremental State Transfer*). En el caso de que el nodo haya permanecido desconectado durante más tiempo o se trate de un nuevo miembro y la GCache no tenga los registros necesarios para que éste alcance el estado del resto de nodos del clúster, se realizará una sincronización mediante *snapshot* SST (*State Snapshot Transfer*) del directorio de datos. Una vez sincronizado con el clúster se volverá operacional.

Al nuevo miembro, que realiza la petición de estado de transferencia, se le denomina "*joiner*" y al nodo que recepciona la petición se le conoce como donante o "*donor*".

El método de transferencia ideal, en cuanto a impacto en el rendimiento del clúster será el IST, ya que éste reducirá considerablemente el tiempo de sincronización y es menos intrusivo. Una transferencia SST es más exigente en términos de recursos y tiempo, en este caso, el nodo donante experimentará una carga adicional que puede afectar al rendimiento del clúster, aunque en ningún momento se desincronizará para realizar la transferencia del estado.

El nodo *joiner*, que solicita el SST, va a recibir una copia del estado de la base de datos desde otro nodo *donor*. En Galera hay varios métodos disponibles para realizar el SST, entre los que podemos destacar los siguientes:

- **mysqldump**: crea una copia de seguridad de la base de datos y luego importa esa copia en el nodo *joiner*. Este es un método fácil de entender y funciona bien para bases de datos pequeñas, en contra, puede ser lento y provocar bloqueos prolongados, ya que utiliza un bloqueo global para garantizar la consistencia durante el volcado de datos.
- **rsync**: copia el directorio de datos de un nodo a otro. Este método es rápido y eficiente y sólo usa los recursos necesarios para la transferencia. No obstante, al igual que el método anterior también realiza un bloqueo global durante la transferencia, por lo que ocasiona interrupciones.
- **xtrabackup-v2**: crea una copia de seguridad en caliente, sin interrupciones, y se transfiere al nodo *joiner*. Este método es más eficiente para bases de datos grandes, en contra, requiere tener instalado XtraBackup y configurado correctamente en todos los nodos.
- **mariabackup**: versión derivada de XtraBackup y optimizada para MariaDB. El funcionamiento es muy similar al del anterior, realiza SST sin bloqueos y es eficiente para bases de datos grandes, en contra, al igual que en el caso anterior, requiere que mariabackup esté instalado y configurado en todos los nodos del clúster.

### 2.2.3. Cifrado de datos

Para dotar de seguridad y privacidad, tanto los datos en reposo (GCache), como los datos en tránsito (tráfico de replicación de Galera y SST), MariaDB proporciona el cifrado desde la versión 10.4 para el fichero de GCache y los datos almacenados permanentemente en los estados de tablas. Del mismo modo y, desde la versión 10.6 de MariaDB, se incorpora el cifrado de los datos en tránsito con TLS de forma predeterminada.

La encriptación de las comunicaciones entre los nodos del clúster de Galera se habilitará automáticamente cuando se configure correctamente en los nodos MariaDB y se generen correctamente los certificados SSL/TLS en cada uno de los miembros del clúster, tal y como veremos en el punto 3.1 de este documento.

### 2.2.4. Quorum (Galera Arbitrator)

El concepto de quorum se utiliza en Galera clúster para determinar cierto tipo de decisiones importantes como pueden ser la elección de que nodo se va a encargar de recepcionar y replicar las transacciones de escritura, desconexión del clúster de un nodo con fallos o prevenir situaciones de *split-brain.* Esta situación puede darse en clústeres con un número par de nodos, perdiéndose, por ejemplo, la conectividad de red y quedando dividido el número de miembros exactamente en la mitad, donde ninguna de las particiones será capaz de retener el quorum y entrando ambas en un estado no primario, tal y como puede verse en la figura 5.

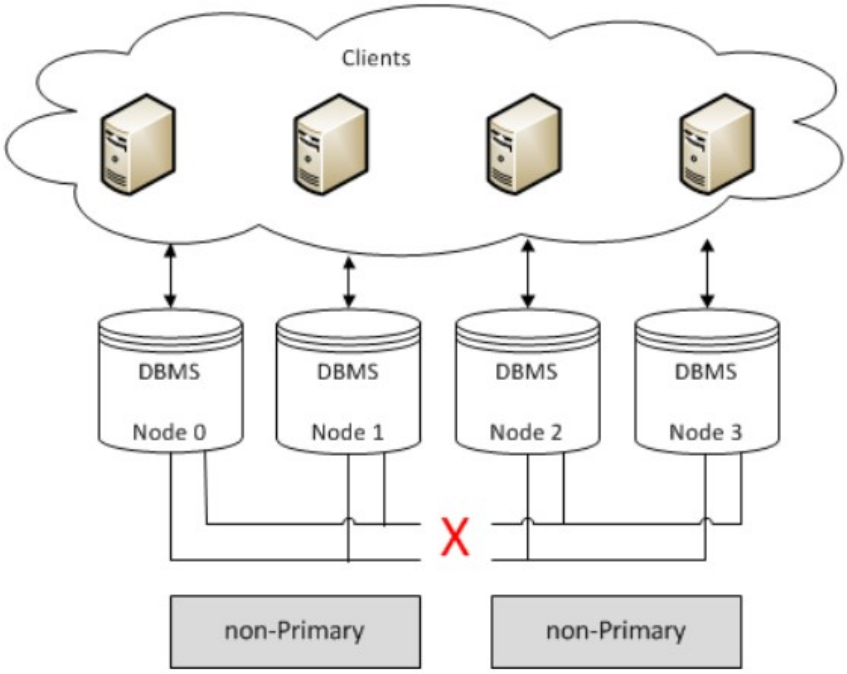



**Figura 5 - Situación de *split-brain*** [7]

Para que un clúster tenga quorum debe haber una mayoría de votos. Idealmente, para garantizar dicha mayoría el clúster deberá estar compuesto por un número impar de miembros, no obstante, podría darse el caso de que, aún contando con un número impar, uno de ellos se desconectara o fallase, poniendo en peligro el quorum y pudiendo detener operaciones críticas, provocar corrupción de datos o degradación del rendimiento. Es en estos casos donde entra en juego el componente de Galera Arbitrator.

Galera Arbitrator es un nodo ligero e independiente que tan sólo se utiliza para proporcionar un voto adicional a las decisiones críticas. No almacena datos ni participa en replicación de transacciones, pero se considera útil en escenarios como el descrito anteriormente y sin ser necesario adicionar un nuevo nodo de datos para resolver el conflicto.

#### 2.2.5. Bootstrapping

*Bootstrap* es el proceso que se conoce para iniciar un clúster de Galera en uno de los nodos, el cual será el componente principal o primario, y al que el resto de los miembros verán automáticamente al iniciarse como punto de referencia con el que sincronizarse. La sincronización se realizará con el componente principal a través de una transferencia de estado IST o SST, tal y como se detalló anteriormente.

Este proceso de *bootstrapping* sólo debe realizarse cuando se requiera iniciar un nuevo clúster, por desconexión completa de éste o porque no existe ningún nodo en estado primario, debiendo tener especial cuidado de no iniciar dos clústeres por separado, ya que podría incurrir en una situación de *split-brain* o pérdida de datos.

### 2.3. MaxScale

MariaDB Maxscale es un proxy para bases de datos MariaDB, que se sitúa, en una arquitectura de red, por delante del clúster de Galera o de los servidores MariaDB para realizar el balanceo de carga y el enrutado de las peticiones desde los clientes hasta los servidores *backend* de base de datos, extendiendo la alta disponibilidad, la escalabilidad y seguridad al mismo tiempo.

La siguiente tabla con los casos de uso de MaxScale nos servirá para entender el propósito de esta herramienta:

| Concepto | Definición |
|---|---|
| Proxy de base de datos | Servidor que actúa como intermediario entre el cliente y los *backends* de bases de datos, de forma que el cliente no se ve afectado por fallos en el servidor o cambios en |

| | |
|---|---|
| | la topología. |
| Balanceo de carga | Envío de consultas a distintos servidores de bases de datos con objeto de que un único servidor no tenga que manejar toda la carga. |
| Failover | Conmutación por error. Se configura un nuevo servidor con el rol de principal cuando falla el existente. |
| Switchover | Configuración de un nuevo servidor como primario cuando se requieren operaciones de mantenimiento sobre el existente. |
| Failback | Devolución del rol de primario a su nodo original. |
| Consistencia causal | Acto de garantizar que las operaciones internas dependientes unas de otras mantengan la coherencia realizándose en el mismo orden en todos los servidores. |
| Lecturas causales | Acto de garantizar la coherencia de que una consulta de lectura se realiza posteriormente a otra de escritura confirmada con anterioridad. |

**Tabla 3 - Casos de uso de MaxScale** [9]

A continuación, en la figura 6, vemos una topología básica para un servidor de MaxScale centralizado, cuyo *backend* será un clúster de Galera con tres nodos de MariaDB. MaxScale se posiciona como proxy o *frontend* para las aplicaciones clientes.

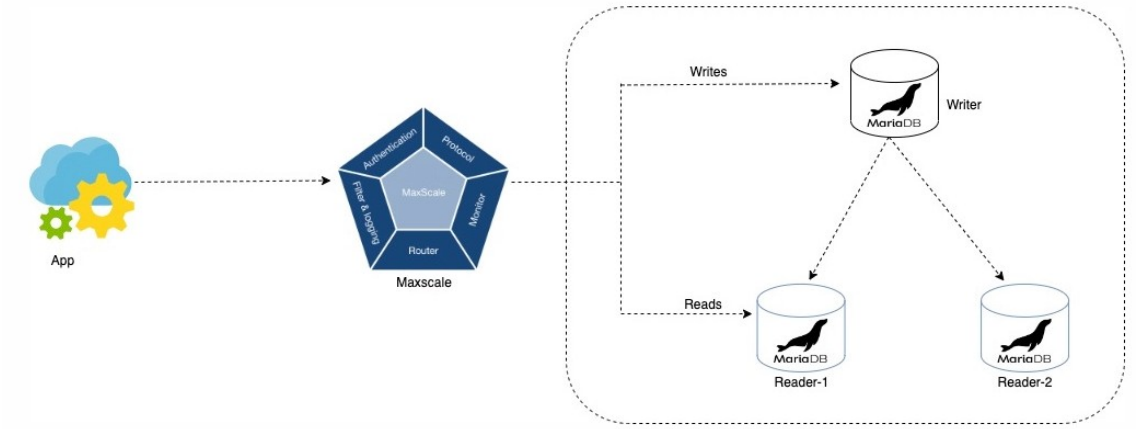



**Figura 6 - Topología básica de un servidor de MaxScale centralizado** [10]

Tal y como se puede observar en la figura anterior, MaxScale se basa en una arquitectura modular para facilitar su configuración: *Authentication, Protocol, Monitor, Router, Filter & logging*. Los tipos de módulos más utilizados son los de protocolo, enrutadores y filtro. Los del tipo protocolo se configuran para para la comunicación entre los clientes y MaxScale (protocolo MariaDBClient) y entre MaxScale y los *backend* MariaDB (protocolo MariaDBBackend). Los

enrutadores se encargan de inspeccionar las consultas de los clientes y decidir el *backend* de destino, dichas decisiones se basan generalmente en las reglas de enrutamiento definidas y el estado del servidor *backend*. Los filtros trabajan con los datos al pasar por MaxScale y se utilizan para el registro de las consultas, con objeto de auditorías, almacenamiento en caché o para modificar las respuestas del servidor.

En los siguientes apartados detallaremos un poco más los módulos de *Monitor* (galeramon) y *Router* (readwritesplit, readconnroute), ya que serán objeto de los laboratorios de este proyecto, así como profundizar en el mecanismo de *failover*, *switchover* y *failback* de MaxScale.

### 2.3.1. Galera Monitor

Galera Monitor (galeramon) es un módulo de MaxScale que monitoriza el estado del clúster de Galera, detectando los nodos miembros y su estado de sincronización, así como la latencia de replicación, tiempo de inactividad y otros indicadores de salud. Permite la configuración de notificaciones de alerta a los administradores del servicio, ya sea mediante el envío de correos electrónicos, mensajes de texto u otros medios.

De forma predeterminada, Galera Monitor elige el nodo con el valor “wsrep_local_index” más bajo como máster. Esto significará que si existe más de un MaxScale que se ejecutan en servidores diferentes todos elegirán el mismo nodo como principal.

Por último, hay que indicar que el uso de los monitores no afecta al rendimiento de los enrutadores, ya que se ejecutan dentro de subprocesos separados en MaxScale.

### 2.3.2. Read-write splitter

Este enrutador se encarga de dividir las operaciones de los clientes a la base de datos, según sean de lectura (SELECT, SHOW, etc.) o de escritura (INSERT, UPDATE, DELETE, etc.) y enrutarlas hacia los *backends* de Galera. Las de escritura se enrutarán hacia el considerado máster y las de lectura hacia los standby o de réplica. El objetivo será el de mejorar el rendimiento, aliviando la carga del nodo máster y liberándolo de recepcionar la carga de solo lectura.

Por otro lado, también es capaz de realizar balanceo de carga entre los nodos standby mediante un reparto equitativo utilizando algoritmos de balanceo de carga configurables.

El servicio de readwritesplit se configura con el protocolo MariaDBClient y un *listener*, que generalmente utilizará el puerto por defecto de MariaDB, 3306, de manera que el paso por el proxy sea transparente para los clientes.

### 2.3.3. Read connections route (readconnroute)

El enrutador para conexiones de lectura (readconnroute) proporciona un equilibrio de carga simple de las conexiones y bastante más liviano que el anterior. Igualmente, el balanceo de carga es configurable en función de parámetros de ponderación definidos en la sección del servidor.

Hay que tener en cuenta que este enrutador balancea las conexiones y no las declaraciones como en el caso anterior. Cuando un cliente se conecta, se selecciona un servidor *backend* según la configuración del enrutador y la carga del nodo, dicha conexión creada es única y no se cambiará durante la sesión. Si la conexión entre MaxScale y el nodo se interrumpe no se podrá restablecer y se cerrará la sesión.

El enrutador readconnroute no impide que se realicen peticiones de escritura, siendo la aplicación cliente la responsable de que tan sólo se realicen este tipo de peticiones. Tal y como se ha comentado anteriormente, este es un enrutador con un diseño simple, se selecciona un *backend* para cada conexión de cliente y enruta allí todas las consultas.

Al igual que en el caso anterior, el servicio requerirá de un *listener* con un puerto configurado que utilizará el protocolo MariaDBClient.

### 2.3.4. Mecanismos de *failover* y *switchover*

Ambos mecanismos son procesos que se refieren a conmutación de nodos de respaldo a principal, diferenciándose sobre todo en la naturaleza de éstos:

- **Failover** es el proceso automático de conmutación por error de un nodo standby a principal cuando el actual experimenta un fallo. MaxScale monitorea constantemente los *backends* y, en caso de detectar un fallo en el nodo considerado como máster promocionará el standby más actualizado a principal, redirigiendo a éste el tráfico de consultas automáticamente y garantizando la continuidad del servicio.
- **Switchover** es el proceso manual de conmutación de la actividad de un nodo principal a otro, generalmente con el objetivo de realizar un mantenimiento planificado sobre el clúster. Una vez completado el mantenimiento se reactiva el nodo manualmente y MaxScale reanuda automáticamente el enrutamiento de consultas de escritura hacia él nuevo primario en caso de estar así configurado.

Nota[1]: si tenemos configurado más de un servidor MaxScale en HA, es importante tener en cuenta que al activar el mantenimiento para uno de los *backends* MariaDB en el MaxScale principal, dicha configuración de mantenimiento no se hereda al otro nodo secundario al promocionar. Significa que la configuración temporal en un nodo principal no se transmite al secundario, perdiéndose y volviendo a su estado original en el caso de conmutar. Esta casuística puede verse como se configura en el punto 4.1 y a prueba en el punto 4.4.1.

## 2.4. Keepalived

Keepalived es una herramienta de software libre de enrutamiento para el balanceo de carga y alta disponibilidad. El objetivo de uso de la herramienta en este proyecto será el de proporcionar alta disponibilidad al servicio de MaxScale mediante la conmutación por error de una IP virtual en una arquitectura de máster/*standby*. La VIP o IP flotante conmutará hacia el MaxScale de respaldo en caso de detección de fallo en el servidor principal. Una vez restablecido el servicio la VIP podrá volver a estar disponible en el servidor original si así se configura o permanecer como está (*failback*).

Keepalived se configura en cada uno de los servidores *backends* a los que presta servicio, considerándose uno de ellos como el principal. Los nodos Keepalived transmiten continuamente su estado a la red y se escuchan entre sí y si un nodo no recibe el mensaje de estado de otro con mayor prioridad que él, reclamará la VIP y se convertirá en el principal. Del mismo modo, este procedimiento puede forzarse parando el servicio de Keepalived manualmente en el nodo principal.

En el escenario planteado en este proyecto, MaxScale no tiene conocimiento de la existencia del servicio de Keepalived y ambos nodos continúan operando como primarios monitoreando los *backends* de Galera y escuchando las conexiones de los clientes. Los clientes se conectan a la VIP, por tanto, sólo la maquina que la reclama será la que reciba las conexiones entrantes. Esta topología de alta disponibilidad para MaxScale se representa en la figura 7.

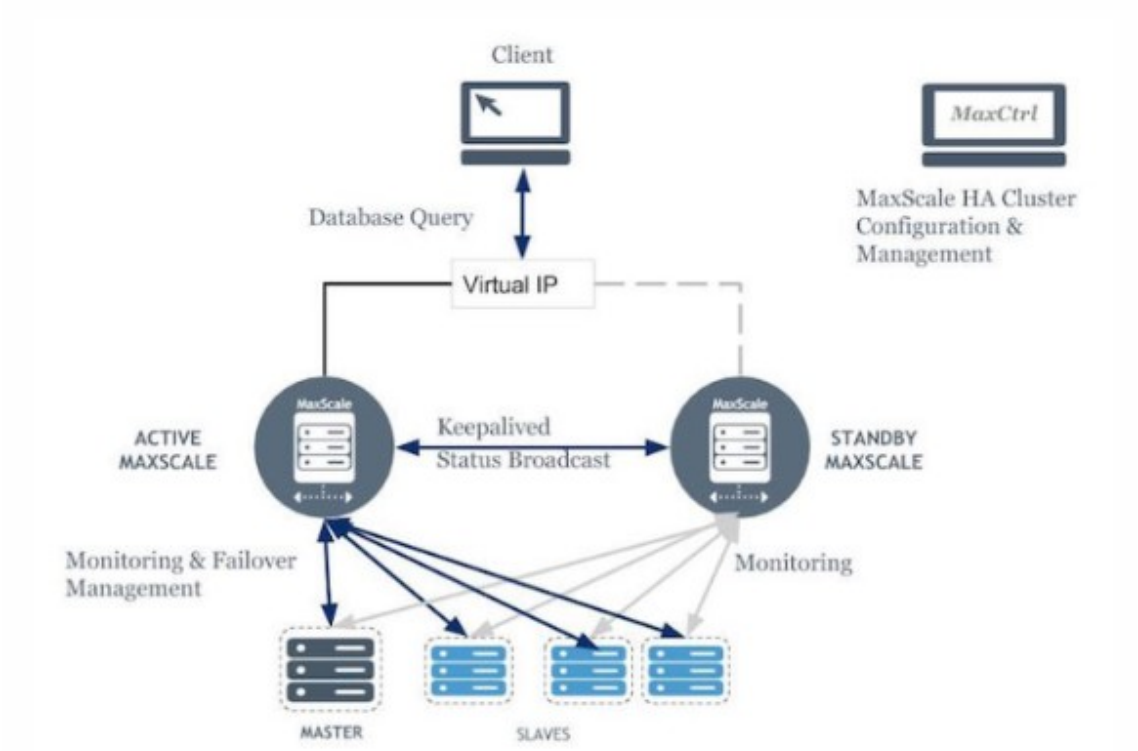



**Figura 7 - Topología de alta disponibilidad para MaxScale con Keepalived** [11]

### 2.4.1. Protocolo VRRP

Keepalived utiliza el protocolo VRRP (*Virtual Router Redundancy Protocol*) para proporcionar alta disponibilidad y *failover* en servidores Linux. Cada servidor que ejecuta Keepalived configura una instancia VRRP con un identificador único y comparte una IP virtual que actúa como Gateway predeterminado para los clientes. Entre los servidores que comparten la VIP, uno de ellos es elegido el maestro mediante algoritmos basados en la prioridad y el resto como respaldo. En el caso de fallo del servidor principal, los nodos de respaldo detectan la falta de anuncios VRRP eligiendo un nuevo maestro [12].

# 3. Despliegue de la infraestructura de Galera Cluster para MariaDB

## 3.1. Instalación y configuración de MariaDB y Galera

La infraestructura de MariaDB que desplegaremos en GCP consistirá en tres nodos idénticos con la configuración de Galera Cluster para la alta disponibilidad y replicación síncrona multimáster, tal y como se mostró anteriormente en la figura 4.

Dado que ya contamos con las tres instancias de VM desplegadas, con la configuración de red y visibilidades a nivel de *firewall*, detallado en el anexo 1, nos centraremos en la instalación de MariaDB y su configuración de clúster. En primer lugar, añadiremos las siguientes entradas al fichero "/etc/hosts" para el mapeo de *hosts* a direcciones IP de las tres instancias, caso de no contar con un servicio DNS en entorno corporativo:

```
# Galera Mariadb nodes
10.10.10.11     galera-mariadb-01 galera-mariadb-01.cluster.lab
10.10.10.12     galera-mariadb-02 galera-mariadb-02.cluster.lab
10.10.10.13     galera-mariadb-03 galera-mariadb-03.cluster.lab
```

Añadimos el repositorio oficial de MariaDB en la versión 10.11 a nuestras fuentes:

```
$ curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --mariadb-server-version="mariadb-10.11"
# [info] Checking for script prerequisites.
# [info] MariaDB Server version 10.11 is valid
# [info] Repository file successfully written to /etc/apt/sources.list.d/mariadb.list
# [info] Adding trusted package signing keys...
# [info] Running apt-get update...
# [info] Done adding trusted package signing keys
```

Instalamos el paquete mariadb-server y sus dependencias:

```
$ sudo apt update
$ sudo apt install mariadb-server
```

Securizamos la instalación con el siguiente comando, en el que nos solicitará una contraseña para el usuario root. Dicha securización elimina el usuario anónimo, deshabilita el *login* remoto para el usuario root, elimina la base de datos de pruebas y recarga privilegios:

```
$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] n
 ... skipping.

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Antes de continuar, realizamos las verificaciones oportunas para confirmar que la instalación es correcta y nos podemos logar satisfactoriamente.

```
$ sudo systemctl status mariadb

● mariadb.service - MariaDB 10.11.7 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-05-05 19:15:16 UTC; 3min 45s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 366 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld
(code=exited, status=0/SUCCESS)
    Process: 386 ExecStartPre=/bin/sh -c systemctl unset-environment
_WSREP_START_POSITION (code=exited, status=0/SUCCESS)
    Process: 391 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||
VAR=`cd /usr/bin/..; /usr/bin/galera_recovery`; [ $? -eq 0 ]   && systemctl set-environment
_WSREP_START_POSITION=$VAR || >
    Process: 591 ExecStartPost=/bin/sh -c systemctl unset-environment
_WSREP_START_POSITION (code=exited, status=0/SUCCESS)
    Process: 595 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
   Main PID: 495 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 9 (limit: 1141)
     Memory: 241.9M
        CPU: 939ms
     CGroup: /system.slice/mariadb.service
             └─495 /usr/sbin/mariadbd

May 05 19:15:16 prueba.cluster.lab mariadbd[495]: 2024-05-05 19:15:16 0 [Note] InnoDB:
Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
May 05 19:15:16 prueba.cluster.lab mariadbd[495]: 2024-05-05 19:15:16 0 [Warning] You need
to use --log-bin to make --expire-logs-days or --binlog-expire-logs-seconds work.
May 05 19:15:16 prueba.cluster.lab mariadbd[495]: 2024-05-05 19:15:16 0 [Note] InnoDB:
Buffer pool(s) load completed at 240505 19:15:16
May 05 19:15:16 prueba.cluster.lab mariadbd[495]: 2024-05-05 19:15:16 0 [Note] Server
socket created on IP: '127.0.0.1'.
May 05 19:15:16 prueba.cluster.lab mariadbd[495]: 2024-05-05 19:15:16 0 [Note]
/usr/sbin/mariadbd: ready for connections.
May 05 19:15:16 prueba.cluster.lab mariadbd[495]: Version: '10.11.7-MariaDB-0+deb12u1'
socket: '/run/mysqld/mysqld.sock'  port: 3306  Debian 12
May 05 19:15:16 prueba.cluster.lab systemd[1]: Started mariadb.service - MariaDB 10.11.7
database server.

$ sudo mariadb -u root -p

Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.7-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)]>
```

Antes de configurar Galera Cluster y replicar al resto de nodos, vamos a securizar las conexiones para el tráfico de replicación entre los nodos de

Galera (SST e IST). Para ello se generará un certificado de servidor autofirmado con openssl:

- Generamos certificado y clave privada de la autoridad certificadora

```
$ openssl genrsa 2048 > ca-key.pem

$ openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca-cert.pem

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
Organizational Unit Name (eg, section) []:TFG
Common Name (e.g. server FQDN or YOUR name) []:galera.cluster.lab
Email Address []:javilops@uoc.edu
```

- Creamos clave privada y certificado autofirmado

```
$ openssl req -newkey rsa:2048 -days 365000 -nodes -keyout mariadb-key.pem -out mariadb-req.pem

Ignoring -days without -x509; not generating a certificate
.+...+.....................+...+..++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++*..........+......+....+......+.........+.....+......+...+....+..+.+..+..........+......+............+...
+.........+...+..+.......+.........++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++*....++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++
..+...+..........+..+.......+.....+....+..+...++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++*................+...+......+.........+...+....+......+...+..............+.+.................+...+.
+.........+...+...+....+......+.....+......+....+......+.....+..........++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++*.............+......+...+..++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Sevilla
Locality Name (eg, city) []:Sevilla
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UOC
Organizational Unit Name (eg, section) []:TFG
Common Name (e.g. server FQDN or YOUR name) []:mariadb.galera.cluster.lab
Email Address []:javilops@uoc.edu
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

$ openssl rsa -in mariadb-key.pem -out mariadb-key.pem
writing RSA key

$ openssl x509 -req -in mariadb-req.pem -days 365000 -CA ca-cert.pem -
CAkey ca-key.pem -set_serial 01 -out mariadb-cert.pem
Certificate request self-signature ok
subject=C = ES, ST = Sevilla, L = Sevilla, O = UOC, OU = TFG, CN =
mariadb.galera.cluster.lab, emailAddress = javilops@uoc.edu
```

- Verificamos que se ha creado correctamente el certificado de servidor autofirmado, generado con la CA que también acabamos de crear:

```
$ ls -l
total 20
-rw-r--r-- 1 root root 1440 May  6 18:28 ca-cert.pem
-rw-r--r-- 1 root root 1704 May  6 18:27 ca-key.pem
-rw-r--r-- 1 root root 1306 May  6 18:33 mariadb-cert.pem
-rw------- 1 root root 1704 May  6 18:32 mariadb-key.pem
-rw-r--r-- 1 root root 1066 May  6 18:32 mariadb-req.pem

$ openssl verify -CAfile ca-cert.pem mariadb-cert.pem
mariadb-cert.pem: OK
```

Se configura el certificado en MariaDB, en la siguiente sección del fichero "/etc/mysql/mariadb.conf.d/50-server.cnf".

```
…
[sst]
encrypt=4
ssl-key=/etc/mysql/mariadb.conf.d/certificates/mariadb-key.pem
ssl-cert=/etc/mysql/mariadb.conf.d/certificates/mariadb-cert.pem
ssl-ca=/etc/mysql/mariadb.conf.d/certificates/ca-cert.pem
…
```

Con la propiedad "*encrypt*=4" estaremos indicando que se va a utilizar el cifrado para todas las conexiones: SST, IST y wsrep replication. El resto de los valores de *encrypt* (0-3) son para cifrar uno u otro tipo de conexiones, siendo 0 la opción por defecto, sin cifrar.

El resto de la configuración para Galera clúster la aplicaremos al fichero "/etc/mysql/mariadb.conf.d/60-galera.cnf". Se muestra la configuración para uno de los nodos (en negrita los valores que deben cambiar para otros nodos):

```
[galera]
wsrep_on              = ON
wsrep_provider        = /usr/lib/galera/libgalera_smm.so
wsrep_cluster_name    = "MariaDB Galera Cluster"
wsrep_cluster_address = "gcomm://10.10.10.11,10.10.10.12,10.10.10.13"
binlog_format         = row
```

```
default_storage_engine   = InnoDB
innodb_autoinc_lock_mode = 2
innodb_force_primary_key = 1
innodb_doublewrite       = 1

bind-address = 0.0.0.0

wsrep_slave_threads           = 1
innodb_flush_log_at_trx_commit = 0
wsrep_node_name               = galera-mariadb-01
wsrep_node_address            = "10.10.10.11"

server_id = 1
log_bin = /var/log/mysql/binlogs/mariadb-bin
log_slave_updates = ON

log_error = /var/log/mysql/error.log

wsrep_sst_method = mariabackup
wsrep_sst_auth = mariabackup:mariadb
```

En la siguiente tabla se explica el significado de cada parámetro configurado con sus posibles valores:

| PARÁMETRO | VALORES | DEFINICIÓN |
|---|---|---|
| **WSREP_ON** | ON / OFF | Habilita el soporte de Galera. |
| **WSREP_PROVIDER** | N/A | Biblioteca utilizada por Galera para manejar la replicación. |
| **WSREP_CLUSTER_NAME** | N/A | Nombre que se le da al clúster de Galera para diferenciarlo de otros. |
| **WSREP_CLUSTER_ADDRESS** | N/A | Direcciones de los nodos que forman parte del clúster. |
| **BINLOG_FORMAT** | ***statement***: declaraciones SQL. Menos espacio, problemas con declaraciones no deterministas.<br>***row***: cambios en las filas. Más detallado y replicación más precisa.<br>***mixed***: mezcla de ambos. Equilibrio entre eficiencia y precisión. | Formato del registro binario que guarda que guarda las declaraciones SQL que modifican datos. Utilizado para la replicación. |
| **DEFAULT_STORAGE_ENGINE** | InnoDB, MyISAM, MEMORY, ARCHIVE | Motor de almacenamiento por defecto para las nuevas tablas. |
| **INNODB_AUTOINC_LOCK_MODE** | **'0' (Tradicional)**: bloqueo de tabla. El más seguro y restrictivo.<br>**'1' (Consecutivo)**: bloqueo de inserción. Equilibrio entre rendimiento y secuencialidad.<br>**'2' (Intercalado)**: Sin bloqueos, máx. concurrencia. Núm. No consecutivos y con espacios. Más rápido. | Tipo de bloqueo utilizado para las columnas 'AUTO_INCREMENT' en InnoDB. |
| **INNODB_FORCE_PRIMARY_KEY** | **'0'**: tablas sin clave primaria.<br>**'1'**: Clave primaria en todas las tablas InnoDB. | Requiere que todas las tablas de InnoDB tengan clave primaria cuando está habilitado. Útil para mejorar el rendimiento e integridad de los datos. |

| | | |
|---|---|---|
| **INNODB_DOUBLEWRITE** | 1 / 0 | Controla si InnoDB utiliza o no el buffer de doble escritura para proteger los datos contra posibles corrupciones. |
| **BIND-ADDRESS** | N/A | Dirección IP en la que se vincula el servicio y escucha conexiones. |
| **WSREP_SLAVE_THREADS** | N/A | Núm. de subprocesos de replicación en paralelo utilizados en el nodo actual. |
| **INNODB_FLUSH_LOG_AT_TRX_COMMIT** | **'0'**: registro a buffer, sin vaciado ni escritura a disco. Más rápido.<br>**'1'**: escribe registro de transacciones y vaciado a disco al final de cada transacción. Más seguro.<br>**'2'**: registro de transacciones a buffer, sin vaciado a disco. Híbrido entre 0 y 1. | Comportamiento de InnoDB respecto a la escritura y vaciado del registro de transacciones. |
| **WSREP_NODE_NAME** | N/A | Nombre del nodo en el clúster. |
| **WSREP_NODE_ADDRESS** | N/A | IP/[Puerto] del nodo en el clúster. |
| **SERVER_ID** | N/A | Identificador único de servidor en el entorno de replicación. Evita conflictos y ciclos infinitos de replicación. |
| **LOG_BIN** | 1 / 0 | Registro binario (binlog) habilitado o no. Necesario para PITR. |
| **LOG_SLAVE_UPDATES** | ON / OFF | Controla que los servidores *slave* guarden o no las actualizaciones recibidas desde el máster. Necesario para PITR desde cualquier nodo. |
| **LOG_ERROR** | N/A | *Path* del fichero de registro de errores del servidor. |
| **WSREP_SST_METHOD** | rsync, mysqldump, xtrabackup, mariabackup. | Método utilizado para realizar el SST. |
| **WSREP_SST_AUTH** | N/A | Credenciales de autenticación *donor/joiner* en la replicación SST. |

**Tabla 4 - Configuración Galera Cluster**

Esta sería una configuración básica de Galera para poder levantar un clúster de tres nodos multimáster con replicación síncrona. También se han configurado otra serie de parámetros, que nos permitirán realizar la replicación y *backup* con mariabackup y realizar restauraciones en el tiempo (PITR) con binlogs.

Una vez configurado el primero de los nodos es posible la clonación de los otros dos dependiendo del sistema de virtualización y ajustando los parámetros que hacen referencia a cada nodo vistos más arriba. En nuestro caso clonaremos desde GCP.

Por último, se levanta el clúster, teniendo en cuenta el comando de inicialización del clúster cuando éste se encuentre completamente parado (ver 2.2.5 Bootstrapping).

### 3.2. Inicio de un clúster Galera desde cero

Se ejecutará el siguiente comando en cualquiera de los nodos, al arrancar el clúster por primera vez para inicializar el clúster, o en el último de los nodos en apagarse en un apagado ordenado:

```
$ sudo galera_new_cluster

2024-05-07 20:30:12 0 [Note] Starting MariaDB 10.11.7-MariaDB-1:10.11.7+maria~deb12-log source revision 87e13722a95af5d9378d990caf48cb6874439347 as process 100554
2024-05-07 20:30:12 0 [Note] WSREP: Loading provider /usr/lib/galera/libgalera_smm.so initial position: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18369
2024-05-07 20:30:12 0 [Note] WSREP: wsrep_load(): loading provider library '/usr/lib/galera/libgalera_smm.so'
2024-05-07 20:30:12 0 [Note] WSREP: wsrep_load(): Galera 26.4.16(r7dce5149) by Codership Oy <info@codership.com> loaded successfully.
2024-05-07 20:30:12 0 [Note] WSREP: Initializing allowlist service v1
2024-05-07 20:30:12 0 [Note] WSREP: CRC-32C: using 64-bit x86 acceleration.
2024-05-07 20:30:12 0 [Note] WSREP: Found saved state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18369, safe_to_bootstrap: 1
2024-05-07 20:30:12 0 [Note] WSREP: GCache DEBUG: opened preamble:
Version: 2
UUID: fb6d29ed-ee13-11ee-b506-eb954fb352d1
Seqno: 1 - 18369
Offset: 1280
Synced: 1
2024-05-07 20:30:12 0 [Note] WSREP: Recovering GCache ring buffer: version: 2, UUID: fb6d29ed-ee13-11ee-b506-eb954fb352d1, offset: 1280
2024-05-07 20:30:12 0 [Note] WSREP: GCache::RingBuffer initial scan...  0.0% (        0/134217752 bytes) complete.
2024-05-07 20:30:12 0 [Note] WSREP: GCache::RingBuffer initial scan...100.0% (134217752/134217752 bytes) complete.
2024-05-07 20:30:12 0 [Note] WSREP: Recovering GCache ring buffer: found gapless sequence 1-18369
2024-05-07 20:30:12 0 [Note] WSREP: GCache::RingBuffer unused buffers scan...  0.0% (      0/6466344 bytes) complete.
2024-05-07 20:30:12 0 [Note] WSREP: Recovering GCache ring buffer: found 58/18427 locked buffers
2024-05-07 20:30:12 0 [Note] WSREP: Recovering GCache ring buffer: free space: 127763232/134217728
2024-05-07 20:30:12 0 [Note] WSREP: GCache::RingBuffer unused buffers scan...100.0% (6466344/6466344 bytes) complete.
2024-05-07 20:30:12 0 [Note] WSREP: Passing config to GCS: base_dir = /var/lib/mysql/; base_host = 10.10.10.11; base_port = 4567; cert.log_conflicts = no; cert.optimistic_pa = yes; debug = no; evs.auto_evict = 0; evs.delay_margin = PT1S; evs.delayed_keep_period = PT30S; evs.inactive_check_period = PT0.5S; evs.inactive_timeout = PT15S; evs.join_retrans_period = PT1S; evs.max_install_timeouts = 3; evs.send_window = 4; evs.stats_report_period = PT1M; evs.suspect_timeout = PT5S; evs.user_send_window = 2; evs.view_forget_timeout = PT24H; gcache.dir = /var/lib/mysql/; gcache.keep_pages_size = 0; gcache.keep_plaintext_size = 128M; gcache.mem_size = 0; gcache.name = galera.cache; gcache.page_size = 128M; gcache.recover = yes; gcache.size = 128M; gcomm.thread_prio = ; gcs.fc_debug = 0; gcs.fc_factor = 1.0; gcs.fc_limit = 16; gcs.fc_master_slave = no; gcs.fc_single_primary = no; gcs.max_packet_size = 64500; gcs.max_throttle = 0.25; gcs.recv_q_hard_limit = 9223372036854775807; gcs.recv_q_soft_limit = 0.25; gcs.sync_donor = no; gmcast.segment = 0;
2024-05-07 20:30:12 0 [Note] WSREP: Service thread queue flushed.
2024-05-07 20:30:12 0 [Note] WSREP: ####### Assign initial position for certification: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18369, protocol version: -1
2024-05-07 20:30:12 0 [Note] WSREP: Start replication
```

```
2024-05-07 20:30:12 0 [Note] WSREP: Connecting with bootstrap option: 1
2024-05-07 20:30:12 0 [Note] WSREP: Setting GCS initial position to fb6d29ed-ee13-11ee-
b506-eb954fb352d1:18369
2024-05-07 20:30:12 0 [Note] WSREP: protonet asio version 0
2024-05-07 20:30:12 0 [Note] WSREP: Using CRC-32C for message checksums.
2024-05-07 20:30:12 0 [Note] WSREP: backend: asio
2024-05-07 20:30:12 0 [Note] WSREP: gcomm thread scheduling priority set to other:0
2024-05-07 20:30:12 0 [Note] WSREP: access file(/var/lib/mysql//gvwstate.dat) failed(No such
file or directory)
2024-05-07 20:30:12 0 [Note] WSREP: restore pc from disk failed
2024-05-07 20:30:12 0 [Note] WSREP: GMCast version 0
2024-05-07 20:30:12 0 [Note] WSREP: (d72aeac4-99b0, 'tcp://0.0.0.0:4567') listening at
tcp://0.0.0.0:4567
2024-05-07 20:30:12 0 [Note] WSREP: (d72aeac4-99b0, 'tcp://0.0.0.0:4567') multicast: , ttl: 1
2024-05-07 20:30:12 0 [Note] WSREP: EVS version 1
2024-05-07 20:30:12 0 [Note] WSREP: gcomm: bootstrapping new group 'MariaDB Galera
Cluster'
2024-05-07 20:30:12 0 [Note] WSREP: start_prim is enabled, turn off pc_recovery
2024-05-07 20:30:12 0 [Note] WSREP: EVS version upgrade 0 -> 1
2024-05-07 20:30:12 0 [Note] WSREP: PC protocol upgrade 0 -> 1
2024-05-07 20:30:12 0 [Note] WSREP: Node d72aeac4-99b0 state prim
2024-05-07 20:30:12 0 [Note] WSREP: view(view_id(PRIM,d72aeac4-99b0,1) memb {
        d72aeac4-99b0,0
} joined {
} left {
} partitioned {
})
2024-05-07 20:30:12 0 [Note] WSREP: save pc into disk
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr without UUID:
tcp://10.10.10.11:4567
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr proto entry 0x558a0d1b5050
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr without UUID:
tcp://10.10.10.12:4567
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr proto entry 0x558a0d1b3450
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr without UUID:
tcp://10.10.10.13:4567
2024-05-07 20:30:12 0 [Note] WSREP: discarding pending addr proto entry 0x558a0d1b3210
2024-05-07 20:30:12 0 [Note] WSREP: gcomm: connected
2024-05-07 20:30:12 0 [Note] WSREP: Changing maximum packet size to 64500, resulting
msg size: 32636
2024-05-07 20:30:12 0 [Note] WSREP: Shifting CLOSED -> OPEN (TO: 0)
2024-05-07 20:30:12 0 [Note] WSREP: Opened channel 'MariaDB Galera Cluster'
2024-05-07 20:30:12 0 [Note] WSREP: New COMPONENT: primary = yes, bootstrap = no,
my_idx = 0, memb_num = 1
2024-05-07 20:30:12 1 [Note] WSREP: Starting rollbacker thread 1
2024-05-07 20:30:12 0 [Note] WSREP: STATE_EXCHANGE: sent state UUID: d72c8378-0c9f-
11ef-9fed-0f2cb98d4b07
2024-05-07 20:30:12 0 [Note] WSREP: STATE EXCHANGE: sent state msg: d72c8378-0c9f-
11ef-9fed-0f2cb98d4b07
2024-05-07 20:30:12 0 [Note] WSREP: STATE EXCHANGE: got state msg: d72c8378-0c9f-
11ef-9fed-0f2cb98d4b07 from 0 (galera-mariadb-01)
2024-05-07 20:30:12 0 [Note] WSREP: Quorum results:
        version    = 6,
        component  = PRIMARY,
        conf_id    = 0,
        members    = 1/1 (joined/total),
        act_id     = 18369,
        last_appl. = 18369,
        protocols  = 2/10/4 (gcs/repl/appl),
        vote policy= 0,
```

```
         group UUID = fb6d29ed-ee13-11ee-b506-eb954fb352d1
2024-05-07 20:30:12 0 [Note] WSREP: Flow-control interval: [16, 16]
2024-05-07 20:30:12 0 [Note] WSREP: Restored state OPEN -> JOINED (18370)
2024-05-07 20:30:12 0 [Note] WSREP: Member 0.0 (galera-mariadb-01) synced with
group.
2024-05-07 20:30:12 0 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 18370)
2024-05-07 20:30:12 2 [Note] WSREP: Starting applier thread 2
2024-05-07 20:30:12 2 [Note] WSREP: ####### processing CC 18370, local, ordered
2024-05-07 20:30:12 2 [Note] WSREP: Process first view: fb6d29ed-ee13-11ee-b506-
eb954fb352d1 my uuid: d72aeac4-0c9f-11ef-99b0-72222026e304
2024-05-07 20:30:12 2 [Note] WSREP: Server galera-mariadb-01 connected to cluster at
position fb6d29ed-ee13-11ee-b506-eb954fb352d1:18370 with ID d72aeac4-0c9f-11ef-
99b0-72222026e304
2024-05-07 20:30:12 2 [Note] WSREP: Server status change disconnected -> connected
2024-05-07 20:30:12 2 [Note] WSREP: ####### My UUID: d72aeac4-0c9f-11ef-99b0-
72222026e304
2024-05-07 20:30:12 2 [Note] WSREP: Cert index reset to 00000000-0000-0000-0000-
000000000000:-1 (proto: 10), state transfer needed: no
2024-05-07 20:30:12 0 [Note] WSREP: Service thread queue flushed.
2024-05-07 20:30:12 2 [Note] WSREP: ####### Assign initial position for certification:
00000000-0000-0000-0000-000000000000:-1, protocol version: -1
2024-05-07 20:30:12 2 [Note] WSREP: REPL Protocols: 10 (5)
2024-05-07 20:30:12 2 [Note] WSREP: ####### Adjusting cert position: -1 -> 18370
2024-05-07 20:30:12 0 [Note] WSREP: Service thread queue flushed.
2024-05-07 20:30:12 2 [Note] WSREP:
=================================================
View:
  id: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18370
  status: primary
  protocol_version: 4
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY,
ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED, PREORDERED,
STREAMING, NBO
  final: no
  own_index: 0
  members(1):
          0: d72aeac4-0c9f-11ef-99b0-72222026e304, galera-mariadb-01
=================================================
2024-05-07 20:30:12 2 [Note] WSREP: Server status change connected -> joiner
2024-05-07 20:30:12 2 [Note] WSREP: Server status change joiner -> initializing
2024-05-07 20:30:12 0 [Note] InnoDB: Compressed tables use zlib 1.2.13
2024-05-07 20:30:12 0 [Note] InnoDB: Using transactional memory
2024-05-07 20:30:12 0 [Note] InnoDB: Number of transaction pools: 1
2024-05-07 20:30:12 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-05-07 20:30:12 0 [Note] InnoDB: Using liburing
2024-05-07 20:30:12 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk
size = 2.000MiB
2024-05-07 20:30:12 0 [Note] InnoDB: Completed initialization of buffer pool
2024-05-07 20:30:12 0 [Note] InnoDB: File system buffers for log disabled (block size=4096
bytes)
2024-05-07 20:30:12 0 [Note] InnoDB: End of log at LSN=8019481
2024-05-07 20:30:12 0 [Note] InnoDB: 128 rollback segments are active.
2024-05-07 20:30:12 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically
writing the file full; Please wait ...
2024-05-07 20:30:12 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
2024-05-07 20:30:12 0 [Note] InnoDB: log sequence number 8019481; transaction id 37253
2024-05-07 20:30:12 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
2024-05-07 20:30:12 0 [Note] Plugin 'FEEDBACK' is disabled.
2024-05-07 20:30:12 0 [Note] InnoDB: Buffer pool(s) load completed at 240507 20:30:12
2024-05-07 20:30:12 0 [Note] Server socket created on IP: '0.0.0.0'.
```

```
2024-05-07 20:30:12 0 [Note] WSREP: wsrep_init_schema_and_SR (nil)
2024-05-07 20:30:12 0 [Note] WSREP: Server initialized
2024-05-07 20:30:12 0 [Note] WSREP: Server status change initializing -> initialized
2024-05-07 20:30:12 2 [Note] WSREP: Bootstrapping a new cluster, setting initial position to
00000000-0000-0000-0000-000000000000:-1
2024-05-07 20:30:12 6 [Note] WSREP: Recovered cluster id fb6d29ed-ee13-11ee-b506-
eb954fb352d1
2024-05-07 20:30:12 2 [Note] WSREP: Server status change initialized -> joined
2024-05-07 20:30:12 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
2024-05-07 20:30:12 0 [Note] /usr/sbin/mariadbd: ready for connections.
Version: '10.11.7-MariaDB-1:10.11.7+maria~deb12-log'  socket: '/run/mysqld/mysqld.sock'
port: 3306  mariadb.org binary distribution
2024-05-07 20:30:12 2 [Note] WSREP: Lowest cert index boundary for CC from group: 18370
2024-05-07 20:30:12 2 [Note] WSREP: Min available from gcache for CC from group: 1
2024-05-07 20:30:12 2 [Note] WSREP: Server galera-mariadb-01 synced with group
2024-05-07 20:30:12 2 [Note] WSREP: Server status change joined -> synced
2024-05-07 20:30:12 2 [Note] WSREP: Synchronized with group, ready for connections
2024-05-07 20:30:12 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
```

En el log de inicio del clúster se han destacado una serie de líneas que nos indican lo siguiente:

- Se ha encontrado un estado almacenado anteriormente, con su identificador UUID y número de secuencia, *seqno*.
- *Safe_to_bootstrap*: 1 *-> nos indica que es seguro inicializar el clúster.*
- Recuperando el *GCache* para el UUID y *seqno*.
- Inicialización del clúster con el nombre establecido en la configuración (MariaDB Galera Clúster)
- Resultado del Quorum, que reconoce el nodo como primario, UUID y número de miembros del clúster actualmente.
- Nodo (galera-mariadb-01) se encuentra sincronizado con el grupo y conectado al clúster con la posición indicada e identificador.
- No necesario SST.
- Confirmación de estado sincronizado y a la espera de conexiones.

Arrancamos el nodo 2 con el siguiente comando:

```
$ sudo systemctl start mariadb.service
```

Y analizamos la parte más relevante del log de arranque de los nodos *joiner* (galera-mariadb-02) y *donor* (galera-mariadb-01).

```
### donor (galera-mariadb-01) ###

2024-05-07 20:57:59 0 [Note] WSREP: (d72aeac4-99b0, 'tcp://0.0.0.0:4567') connection
established to b8c9ecc6-8e78 tcp://10.10.10.12:4567
2024-05-07 20:57:59 0 [Note] WSREP: (d72aeac4-99b0, 'tcp://0.0.0.0:4567') turning message
relay requesting on, nonlive peers:
2024-05-07 20:57:59 0 [Note] WSREP: declaring b8c9ecc6-8e78 at tcp://10.10.10.12:4567
stable
2024-05-07 20:57:59 0 [Note] WSREP: Node d72aeac4-99b0 state prim
2024-05-07 20:57:59 0 [Note] WSREP: view(view_id(PRIM,b8c9ecc6-8e78,2) memb {
        b8c9ecc6-8e78,0
        d72aeac4-99b0,0
```

```
} joined {
} left {
} partitioned {
})
2024-05-07 20:57:59 0 [Note] WSREP: save pc into disk
2024-05-07 20:57:59 0 [Note] WSREP: New COMPONENT: primary = yes, bootstrap = no,
my_idx = 1, memb_num = 2
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: Waiting for state UUID.
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: sent state msg: b916ffd4-
0ca3-11ef-9b02-aa1f847c3f65
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: got state msg: b916ffd4-0ca3-
11ef-9b02-aa1f847c3f65 from 0 (galera-mariadb-02)
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: got state msg: b916ffd4-0ca3-
11ef-9b02-aa1f847c3f65 from 1 (galera-mariadb-01)
2024-05-07 20:57:59 0 [Note] WSREP: Quorum results:
        version    = 6,
        component  = PRIMARY,
        conf_id    = 1,
        members    = 1/2 (joined/total),
        act_id     = 18370,
        last_appl. = 18369,
        protocols  = 2/10/4 (gcs/repl/appl),
        vote policy= 0,
        group UUID = fb6d29ed-ee13-11ee-b506-eb954fb352d1
2024-05-07 20:57:59 0 [Note] WSREP: Flow-control interval: [23, 23]
2024-05-07 20:57:59 2 [Note] WSREP: ####### processing CC 18371, local, ordered
2024-05-07 20:57:59 2 [Note] WSREP: ####### My UUID: d72aeac4-0c9f-11ef-99b0-
72222026e304
2024-05-07 20:57:59 2 [Note] WSREP: Skipping cert index reset
2024-05-07 20:57:59 2 [Note] WSREP: REPL Protocols: 10 (5)
2024-05-07 20:57:59 2 [Note] WSREP: ####### Adjusting cert position: 18370 -> 18371
2024-05-07 20:57:59 0 [Note] WSREP: Service thread queue flushed.
2024-05-07 20:57:59 2 [Note] WSREP:
=================================================
View:
  id: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18371
  status: primary
  protocol_version: 4
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY,
ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED,
PREORDERED, STREAMING, NBO
  final: no
  own_index: 1
  members(2):
        0: b8c9ecc6-0ca3-11ef-8e78-1ba8b1b60b5c, galera-mariadb-02
        1: d72aeac4-0c9f-11ef-99b0-72222026e304, galera-mariadb-01
=================================================
2024-05-07 20:57:59 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
2024-05-07 20:57:59 2 [Note] WSREP: Lowest cert index boundary for CC from group: 18371
2024-05-07 20:57:59 2 [Note] WSREP: Min available from gcache for CC from group: 1
2024-05-07 20:58:00 0 [Note] WSREP: Member 0.0 (galera-mariadb-02) requested state
transfer from '10.10.10.13,'. Selected 1.0 (galera-mariadb-01)(SYNCED) as donor.
2024-05-07 20:58:00 0 [Note] WSREP: Shifting SYNCED -> DONOR/DESYNCED (TO: 18371)
2024-05-07 20:58:00 2 [Note] WSREP: Detected STR version: 1, req_len: 133, req: STRv1
2024-05-07 20:58:00 2 [Note] WSREP: IST request: fb6d29ed-ee13-11ee-b506-
eb954fb352d1:18367-18371|tcp://10.10.10.12:4568
2024-05-07 20:58:00 2 [Note] WSREP: Server status change synced -> donor
2024-05-07 20:58:00 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
2024-05-07 20:58:00 0 [Note] WSREP: Donor monitor thread started to monitor
2024-05-07 20:58:00 0 [Note] WSREP: Running: 'wsrep_sst_mariabackup --role 'donor' --
```

```
address '10.10.10.12:4444/xtrabackup_sst//1' --local-port 3306 --socket
'/run/mysqld/mysqld.sock' --progress 0 --datadir '/var/lib/mysql/' --gtid 'fb6d29ed-ee13-
11ee-b506-eb954fb352d1:18367' --gtid-domain-id 0 --binlog
'/var/log/mysql/binlogs/mariadb-bin' --bypass --mysqld-args --wsrep-new-cluster --
wsrep_start_position=fb6d29ed-ee13-11ee-b506-eb954fb352d1:18369'
2024-05-07 20:58:00 2 [Note] WSREP: sst_donor_thread signaled with 0
2024-05-07 20:58:00 0 [Note] WSREP: async IST sender starting to serve
tcp://10.10.10.12:4568 sending 18368-18371, preload starts from 18371
2024-05-07 20:58:00 0 [Note] WSREP: IST sender 18368 -> 18371
WSREP_SST: [INFO] mariabackup IST started on donor (20240507 20:58:00.189)
WSREP_SST: [INFO] SSL configuration: CA='/etc/mysql/mariadb.conf.d/certificates/ca-
cert.pem', CAPATH='', CERT='/etc/mysql/mariadb.conf.d/certificates/mariadb-cert.pem',
KEY='/etc/mysql/mariadb.conf.d/certificates/mariadb-key.pem', MODE='DISABLED',
encrypt='4' (20240507 20:58:00.313)
WSREP_SST: [INFO] Logging all stderr of SST/mariadb-backup to syslog (20240507
20:58:00.674)
2024-05-07 20:58:00 0 [Note] WSREP: SST sent: fb6d29ed-ee13-11ee-b506-
eb954fb352d1:18367
2024-05-07 20:58:00 0 [Note] WSREP: Server status change donor -> joined
2024-05-07 20:58:00 0 [Note] WSREP: 1.0 (galera-mariadb-01): State transfer to 0.0
(galera-mariadb-02) complete.
2024-05-07 20:58:00 0 [Note] WSREP: Shifting DONOR/DESYNCED -> JOINED (TO: 18371)
2024-05-07 20:58:00 0 [Note] WSREP: Processing event queue:... -nan% (0/0 events)
complete.
2024-05-07 20:58:00 0 [Note] WSREP: Member 1.0 (galera-mariadb-01) synced with
group.
2024-05-07 20:58:00 0 [Note] WSREP: Processing event queue:...100.0% (1/1 events)
complete.
2024-05-07 20:58:00 0 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 18371)
2024-05-07 20:58:00 2 [Note] WSREP: Server galera-mariadb-01 synced with group
2024-05-07 20:58:00 2 [Note] WSREP: Server status change joined -> synced
2024-05-07 20:58:00 2 [Note] WSREP: Synchronized with group, ready for connections
2024-05-07 20:58:00 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
2024-05-07 20:58:00 0 [Note] WSREP: Donor monitor thread ended with total time 0 sec
2024-05-07 20:58:01 0 [Note] WSREP: async IST sender served
2024-05-07 20:58:01 0 [Note] WSREP: 0.0 (galera-mariadb-02): State transfer from 1.0
(galera-mariadb-01) complete.
2024-05-07 20:58:01 0 [Note] WSREP: Member 0.0 (galera-mariadb-02) synced with
group.
2024-05-07 20:58:02 0 [Note] WSREP: (d72aeac4-99b0, 'tcp://0.0.0.0:4567') turning message
relay requesting off
```

- Detectada nueva conexión tcp://IP:PORT
- El nuevo componente considerado como primario y 'bootstrap = no'.
- Información del identificador de estado del *donor* y del nodo *joiner* coinciden.
- Resultado del quorum, que identifica el nodo *joiner* como primario. Información de miembros: 1/2 (unidos/total).
- Información de versión del protocolo de replicación (4 - mariabackup).
- Se realiza la petición de estado de transferencia del nodo nuevo y selección del nodo01 como DONOR.
- Estado de transferencia se realizará con IST. Información de la configuración para el IST en el nodo donante.
- IST *sender* 18368 -> 18371 (*seqno* que se va a alcanzar).
- Comienza IST, información de las *keys* públicas y privadas para la transferencia cifrada.

- Finaliza la transferencia y confirmación del *donor* unido y sincronizado con el grupo
- Confirmación de sincronización OK en *joiner*.

```
### joiner (galera-mariadb-02) ###

…
2024-05-07 20:57:59 0 [Note] Starting MariaDB 10.11.7-MariaDB-1:10.11.7+maria~deb12-log source revision 87e13722a95af5d9378d990caf48cb6874439347 as process 19505
2024-05-07 20:57:59 0 [Note] WSREP: Loading provider /usr/lib/galera/libgalera_smm.so initial position: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18367
…
2024-05-07 20:57:59 0 [Note] WSREP: Found saved state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18367, safe_to_bootstrap: 0
2024-05-07 20:57:59 0 [Note] WSREP: GCache DEBUG: opened preamble:
Version: 2
UUID: fb6d29ed-ee13-11ee-b506-eb954fb352d1
Seqno: 18337 - 18367
Offset: 6457464
Synced: 1
2024-05-07 20:57:59 0 [Note] WSREP: Recovering GCache ring buffer: version: 2, UUID: fb6d29ed-ee13-11ee-b506-eb954fb352d1, offset: 6457464
2024-05-07 20:57:59 0 [Note] WSREP: GCache::RingBuffer initial scan...  0.0% (        0/134217752 bytes) complete.
2024-05-07 20:57:59 0 [Note] WSREP: GCache::RingBuffer initial scan...100.0% (134217752/134217752 bytes) complete.
2024-05-07 20:57:59 0 [Note] WSREP: Recovering GCache ring buffer: found gapless sequence 18337-18367
2024-05-07 20:57:59 0 [Note] WSREP: GCache::RingBuffer unused buffers scan...  0.0% ( 0/10912 bytes) complete.
2024-05-07 20:57:59 0 [Note] WSREP: Recovering GCache ring buffer: found 4/35 locked buffers
2024-05-07 20:57:59 0 [Note] WSREP: Recovering GCache ring buffer: free space: 134207528/134217728
2024-05-07 20:57:59 0 [Note] WSREP: GCache::RingBuffer unused buffers scan...100.0% (10912/10912 bytes) complete.
…
2024-05-07 20:57:59 0 [Note] WSREP: gcomm: connecting to group 'MariaDB Galera Cluster', peer '10.10.10.11:,10.10.10.12:,10.10.10.13:'
2024-05-07 20:57:59 0 [Note] WSREP: (b8c9ecc6-8e78, 'tcp://0.0.0.0:4567') Found matching local endpoint for a connection, blacklisting address tcp://10.10.10.12:4567
2024-05-07 20:57:59 0 [Note] WSREP: (b8c9ecc6-8e78, 'tcp://0.0.0.0:4567') connection established to d72aeac4-99b0 tcp://10.10.10.11:4567
…
2024-05-07 20:57:59 0 [Note] WSREP: Opened channel 'MariaDB Galera Cluster'
2024-05-07 20:57:59 0 [Note] WSREP: New COMPONENT: primary = yes, bootstrap = no, my_idx = 0, memb_num = 2
…
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: got state msg: b916ffd4-0ca3-11ef-9b02-aa1f847c3f65 from 0 (galera-mariadb-02)
2024-05-07 20:57:59 0 [Note] WSREP: STATE EXCHANGE: got state msg: b916ffd4-0ca3-11ef-9b02-aa1f847c3f65 from 1 (galera-mariadb-01)
2024-05-07 20:57:59 0 [Note] WSREP: Quorum results:
        version    = 6,
        component  = PRIMARY,
        conf_id    = 1,
        members    = 1/2 (joined/total),
        act_id     = 18370,
```

```
         last_appl. = 18369,
         protocols  = 2/10/4 (gcs/repl/appl),
         vote policy= 0,
         group UUID = fb6d29ed-ee13-11ee-b506-eb954fb352d1
…
2024-05-07 20:57:59 2 [Note] WSREP: Process first view: fb6d29ed-ee13-11ee-b506-eb954fb352d1 my uuid: b8c9ecc6-0ca3-11ef-8e78-1ba8b1b60b5c
2024-05-07 20:57:59 2 [Note] WSREP: Server galera-mariadb-02 connected to cluster at position fb6d29ed-ee13-11ee-b506-eb954fb352d1:18371 with ID b8c9ecc6-0ca3-11ef-8e78-1ba8b1b60b5c
2024-05-07 20:57:59 2 [Note] WSREP: Server status change disconnected -> connected
2024-05-07 20:57:59 2 [Note] WSREP: ####### My UUID: b8c9ecc6-0ca3-11ef-8e78-1ba8b1b60b5c
…
2024-05-07 20:57:59 2 [Note] WSREP: State transfer required:
         Group state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18371
         Local state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18367
2024-05-07 20:57:59 2 [Note] WSREP: Server status change connected -> joiner
2024-05-07 20:57:59 0 [Note] WSREP: Joiner monitor thread started to monitor
2024-05-07 20:57:59 0 [Note] WSREP: Running: 'wsrep_sst_mariabackup --role 'joiner' --address '10.10.10.12' --datadir '/var/lib/mysql/' --parent 19505 --progress 0 --binlog '/var/log/mysql/binlogs/mariadb-bin' --mysqld-args --wsrep_start_position=fb6d29ed-ee13-11ee-b506-eb954fb352d1:18367'
WSREP_SST: [INFO] mariabackup SST started on joiner (20240507 20:57:59.658)
WSREP_SST: [INFO] SSL configuration: CA='/etc/mysql/mariadb.conf.d/certificates/ca-cert.pem', CAPATH='', CERT='/etc/mysql/mariadb.conf.d/certificates/mariadb-cert.pem', KEY='/etc/mysql/mariadb.conf.d/certificates/mariadb-key.pem', MODE='DISABLED', encrypt='4' (20240507 20:57:59.764)
WSREP_SST: [INFO] Logging all stderr of SST/mariadb-backup to syslog (20240507 20:57:59.954)
2024-05-07 20:58:00 2 [Note] WSREP: ####### IST uuid:fb6d29ed-ee13-11ee-b506-eb954fb352d1 f: 18368, l: 18371, STRv: 3
2024-05-07 20:58:00 2 [Note] WSREP: IST receiver addr using tcp://10.10.10.12:4568
2024-05-07 20:58:00 2 [Note] WSREP: Prepared IST receiver for 18368-18371, listening at: tcp://10.10.10.12:4568
2024-05-07 20:58:00 0 [Note] WSREP: Member 0.0 (galera-mariadb-02) requested state transfer from '10.10.10.13,'. Selected 1.0 (galera-mariadb-01)(SYNCED) as donor.
2024-05-07 20:58:00 0 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 18371)
2024-05-07 20:58:00 2 [Note] WSREP: Requesting state transfer: success, donor: 1
2024-05-07 20:58:00 0 [Note] WSREP: 1.0 (galera-mariadb-01): State transfer to 0.0 (galera-mariadb-02) complete.
2024-05-07 20:58:00 0 [Note] WSREP: Member 1.0 (galera-mariadb-01) synced with group.
2024-05-07 20:58:00 3 [Note] WSREP: SST received
…
2024-05-07 20:58:01 3 [Note] WSREP: Recovered position from storage: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18367
2024-05-07 20:58:01 3 [Note] WSREP: Recovered view from SST:
  id: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18343
  status: primary
  protocol_version: 4
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY, ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED, PREORDERED, STREAMING, NBO
  final: no
  own_index: -1
  members(3):
         0: a8be63f9-0bed-11ef-9e40-2a09825c1d81, galera-mariadb-01
         1: e6e6a9fe-0bed-11ef-ae6b-7345baeaef72, galera-mariadb-02
         2: ed247459-0bed-11ef-b152-1fe88bfefcc5, galera-mariadb-03
…
```

```
2024-05-07 20:58:01 0 [Note] WSREP: ####### IST applying starts with 18368
2024-05-07 20:58:01 0 [Note] WSREP: ####### IST current seqno initialized to 18368
2024-05-07 20:58:01 0 [Note] WSREP: Receiving IST...  0.0% (0/4 events) complete.
…
2024-05-07 20:58:01 0 [Note] WSREP: Receiving IST...100.0% (4/4 events) complete.
2024-05-07 20:58:01 2 [Note] WSREP:
=================================================
View:
  id: fb6d29ed-ee13-11ee-b506-eb954fb352d1:18371
  status: primary
  protocol_version: 4
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY,
ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED, PREORDERED,
STREAMING, NBO
  final: no
  own_index: 0
  members(2):
          0: b8c9ecc6-0ca3-11ef-8e78-1ba8b1b60b5c, galera-mariadb-02
          1: d72aeac4-0c9f-11ef-99b0-72222026e304, galera-mariadb-01
=================================================
2024-05-07 20:58:01 2 [Note] WSREP: Server status change initialized -> joined
…
2024-05-07 20:58:01 2 [Note] WSREP: Server galera-mariadb-02 synced with group
2024-05-07 20:58:01 2 [Note] WSREP: Server status change joined -> synced
2024-05-07 20:58:01 2 [Note] WSREP: Synchronized with group, ready for connections
```

- Se realiza la comprobación de la GCache existente en el nodo joiner.
- Información de la conexión al grupo y su información de nodos.
- Comienza la transferencia del estado de transferencia desde el nodo donante y se realiza la comprobación de la conexión SSL. Se comprueba la clave del servidor donante y se establece la confianza en el joiner.
- Completada la transferencia IST al 100%.
- Confirmación de nodo galera-mariadb-02 unido, sincronizado con el grupo y a la espera de conexiones.

El tercer nodo lo arrancamos con el mismo comando que el segundo y las trazas que obtendremos serán muy similares a las anteriores, en este caso el nodo donante podrá ser cualquiera de los dos que ya están unidos y sincronizados con el grupo, se decidirá en el procedimiento de Quorum del clúster.

### 3.3. Comandos útiles

A continuación, una lista de variables útiles para comprobar el estado de un clúster de Galera [16]:

- **wsrep_local_send_queue_avg** → promedio de número del tamaño de la cola de envío local. Útil para monitorear el rendimiento y capacidad de procesamiento en el nodo.

- **wsrep_local_send_queue_avg** → tamaño promedio de la cola de recepción local. Muestra cuántas transacciones están esperando ser aplicadas en el nodo actual.
- **wsrep_flow_control_paused** → porcentaje de tiempo que el nodo ha estado en estado de flujo controlado o pausado debido a problemas de sincronización con otros nodos.
- **wsrep_cert_deps_distance** → mide la distancia entre transacciones consecutivas certificadas en el mismo nodo. Se refiere al número de transacciones que no tuvieron conflictos de certificación antes de que se aplicara una transacción con conflictos.
- **wsrep_local_state_comment** → descripción del estado del nodo. Estado en que se encuentra un nodo en relación con el clúster.
- **wsrep_cluster_conf_id** → variable de estado en Galera que indica el ID de la configuración actual del clúster. Se actualiza cada vez que se produce un cambio de configuración en el clúster. Todos los nodos deben tener el mismo valor.
- **wsrep_cluster_size** → indica el número de nodos activos en el clúster.
- **wsrep_cluster_state_uuid** → esta variable nos devolverá el identificador de estado del clúster. Cada nodo del clúster debe devolver el mismo valor, indicando así que todos los nodos están sincronizados, tal y como podemos comprobar en las tres capturas siguientes de cada uno de los nodos.
- **wsrep_cluster_status** → indica si el nodo es o no tipo primario en el clúster. Otros estados – 'Disconnected': desconectado del clúster; 'Primary': nodo primario en el clúster, estado deseado; 'Non-Primary': vista no primaria, clúster dividido y el nodo no puede replicar; 'Undefined': estado indefinido, generalmente cuando el nodo está arrancando.
- **wsrep_connected** → indica si el nodo está conectado a otros nodos del clúster.
- **wsrep_ready** → indica si el nodo está listo para replicar transacciones.

Las siguientes capturas con el resultado de las variables en cada uno de los nodos de Galera evidencian el buen estado de salud del clúster:

**galera-mariadb-01**

```
MariaDB [(none)]> SHOW GLOBAL STATUS WHERE Variable_name =
'wsrep_cluster_state_uuid' OR Variable_name = 'wsrep_cluster_conf_id' OR Variable_name =
'wsrep_cluster_status' OR Variable_name = 'wsrep_cluster_size' OR Variable_name =
'wsrep_ready' OR Variable_name = 'wsrep_connected' OR Variable_name =
'wsrep_local_state_comment' OR Variable_name = 'wsrep_local_recv_queue_avg' OR
Variable_name = 'wsrep_flow_control_paused' OR Variable_name =
'wsrep_cert_deps_distance' OR Variable_name = 'wsrep_local_send_queue_avg'; OR
Variable_name = 'wsrep_local_state_comment'
```

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_send_queue_avg | 0                                    |
| wsrep_local_recv_queue_avg | 0.142857                             |
| wsrep_flow_control_paused  | 0                                    |
| wsrep_cert_deps_distance   | 0                                    |
| wsrep_local_state_comment  | Synced                               |
| wsrep_cluster_conf_id      | 3                                    |
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_state_uuid   | fb6d29ed-ee13-11ee-b506-eb954fb352d1 |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
```

**galera-mariadb-02**

```
MariaDB [(none)]> SHOW GLOBAL STATUS WHERE Variable_name =
'wsrep_cluster_state_uuid' OR Variable_name = 'wsrep_cluster_conf_id' OR Variable_name =
'wsrep_cluster_status' OR Variable_name = 'wsrep_cluster_size' OR Variable_name =
'wsrep_ready' OR Variable_name = 'wsrep_connected' OR Variable_name =
'wsrep_local_state_comment' OR Variable_name = 'wsrep_local_recv_queue_avg' OR
Variable_name = 'wsrep_flow_control_paused' OR Variable_name =
'wsrep_cert_deps_distance' OR Variable_name = 'wsrep_local_send_queue_avg'; OR
Variable_name = 'wsrep_local_state_comment
```

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_send_queue_avg | 0.333333                             |
| wsrep_local_recv_queue_avg | 0                                    |
| wsrep_flow_control_paused  | 0                                    |
| wsrep_cert_deps_distance   | 0                                    |
| wsrep_local_state_comment  | Synced                               |
| wsrep_cluster_conf_id      | 3                                    |
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_state_uuid   | fb6d29ed-ee13-11ee-b506-eb954fb352d1 |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
```

galera-mariadb-03

```
MariaDB [(none)]> SHOW GLOBAL STATUS WHERE Variable_name =
'wsrep_cluster_state_uuid' OR Variable_name = 'wsrep_cluster_conf_id' OR Variable_name =
'wsrep_cluster_status' OR Variable_name = 'wsrep_cluster_size' OR Variable_name =
'wsrep_ready' OR Variable_name = 'wsrep_connected' OR Variable_name =
'wsrep_local_state_comment' OR Variable_name = 'wsrep_local_recv_queue_avg' OR
Variable_name = 'wsrep_flow_control_paused' OR Variable_name =
'wsrep_cert_deps_distance' OR Variable_name = 'wsrep_local_send_queue_avg'; OR
Variable_name = 'wsrep_local_state_comment
```

```
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_send_queue_avg | 0                                    |
| wsrep_local_recv_queue_avg | 0                                    |
| wsrep_flow_control_paused  | 0                                    |
| wsrep_cert_deps_distance   | 0                                    |
| wsrep_local_state_comment  | Synced                               |
| wsrep_cluster_conf_id      | 3                                    |
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_state_uuid   | fb6d29ed-ee13-11ee-b506-eb954fb352d1 |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
```

**Reinicio del Quorum**[17]

Si tras comprobar el estado de la variable 'wsrep_cluster_status' en todos los nodos del clúster, ninguno de ellos devuelve el valor de 'Primary', podría deberse a una situación de split-brain o problemas de red, lo cual puede ocasionar un problema importante y será necesario un reseteo del Quorum. Los pasos que se deberán seguir son los siguientes:

1. **Encontrar el nodo más avanzado del clúster**. El siguiente comando devolverá el número de secuencia, el nodo que devuelva el valor más alto será por tanto el más avanzado y el que deberá utilizarse para el nuevo componente 'Primary'.

```
$ mariadb -u root -pmariadb -r -t -e "SHOW STATUS LIKE
'wsrep_last_committed';"

+----------------------------+---------+
| Variable_name              | Value |
+----------------------------+---------+
| wsrep_last_committed | 18372 |
+----------------------------+---------+
```

2. **Establecer nuevo 'Primary'**. Cuando se hace el reinicio del quorum, lo que estamos haciendo es un *bootstrapping* a componente primario del nodo más avanzado disponible y el resto de los nodos se sincronizarán posteriormente con éste. Existe la posibilidad de realizar este procedimiento de forma automática o manual, siendo la automática la opción recomendada, ya que conserva la GCache en cada nodo, pudiendo así sincronizarse con IST, en lugar de SST.

   - **Bootstrap automático** → habilitando pc.bootstrap a través de la variable 'wsrep_provider_options' desde el cliente de base de datos, en lugar del archivo de configuración. Una vez configurado el nodo se convierte en el nuevo componente principal.

```
SET GLOBAL wsrep_provider_options='pc.bootstrap=YES';
```

- **Bootstrap manual** → se parará el servicio de MariaDB en todos los nodos y a continuación se arrancará en primer lugar el identificado como más avanzado con el comando 'galera_new_cluster'. En caso de error al inicializar el clúster, indicando que no es seguro, podemos forzarlo, estableciendo el parámetro '**safe_to_bootstrap = 1**' en el fichero '/var/lib/mysql/grastate.dat':

```
$ sudo cat /var/lib/mysql/grastate.dat

# GALERA saved state
version: 2.1
uuid:    fb6d29ed-ee13-11ee-b506-eb954fb352d1
seqno:   -1
safe_to_bootstrap: 0
```

# 4. Despliegue de la infraestructura de proxy con MaxScale y Keepalived

## 4.1. Instalación y configuración de MaxScale

Una vez instalado y configurado el clúster de Galera con MariaDB, procedemos a montar la infraestructura de proxy con MaxScale, el cual distribuirá las peticiones de los clientes a los nodos *backend* de Galera.

A continuación, se va a configurar una infraestructura de proxy con MaxScale y dos nodos en alta disponibilidad, como la vista en la figura 7, gestionando y balanceando las peticiones de los clientes hacia Galera Cluster. De los dos nodos de MaxScale, uno será el que actúe como primario y el otro como secundario o standby y a la espera de conmutar a primario en caso de necesidad.

Empezaremos agregando las dos máquinas nuevas en el fichero "/etc/hosts" para el mapeo de hosts en todas las instancias que tenemos hasta ahora:

```
# Galera Mariadb nodes
10.10.10.11     galera-mariadb-01 galera-mariadb-01.cluster.lab
10.10.10.12     galera-mariadb-02 galera-mariadb-02.cluster.lab
10.10.10.13     galera-mariadb-03 galera-mariadb-03.cluster.lab

# MaxScale Proxy nodes
10.10.10.101    maxscale-lb-01 maxscale-lb-01.cluster.lab
10.10.10.102    maxscale-lb-02 maxscale-lb-02.cluster.lab
```

Descargamos e instalamos la paquetería necesaria para MaxScale, en este caso, instalaremos la versión 23.08.5.

```
$ sudo apt install libltdl7 libodbc2
$ wget
https://dlm.mariadb.com/3773315/MaxScale/23.08.5/packages/debian/bookworm/x86_64/maxscale-23.08.5-1.debian.bookworm.x86_64.deb
$ sudo dpkg -i maxscale-23.08.5-1.debian.bookworm.x86_64.deb
```

La configuración de MaxScale es exactamente la misma en todos los nodos que forman el clúster y los cambios en la configuración se replicarán de uno a otro a través de la tabla 'mysql.maxscale_config' en los *backends* de Galera. Dicha tabla se creará cuando se realice la primera modificación en la configuración de MaxScale y será accesible mediante un usuario y contraseña que tendremos que configurar en MaxScale.

Empezaremos por la creación del usuario y asignación de permisos en cualquiera de los nodos de MariaDB, que se propagará al resto de miembros que forman el clúster de Galera.

```
MariaDB [(none)]> CREATE USER 'maxscale_sync'@'%' IDENTIFIED BY
'maxscale';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT SELECT, INSERT, UPDATE, CREATE ON
`mysql`.`maxscale_config` TO maxscale_sync@'%';
Query OK, 0 rows affected (0.009 sec)
```

Aprovechamos también para crear el usuario 'maxscaleuser' y los GRANT necesarios, que nos servirá para la monitorización del clúster y el servicio de 'readwritesplit' de MaxScale:

```
MariaDB [(none)]> CREATE USER 'maxscaleuser'@'%' IDENTIFIED BY
'maxscale';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT SELECT ON mysql.user TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.db TO 'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.tables_priv TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.roles_mapping TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SHOW DATABASES ON *.* TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT REPLICATION CLIENT on *.* to
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.columns_priv TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.procs_priv TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
MariaDB [(none)]> GRANT SELECT ON mysql.proxies_priv TO
'maxscaleuser'@'%';
Query OK, 0 rows affected (0.009 sec)
```

Y ahora ya podemos empezar con la configuración de MaxScale en uno de los nodos, que realizaremos en el fichero "/etc/maxscale.cnf" con una configuración básica y suficiente para la ejecución de los laboratorios:

```
[maxscale]
threads = auto
config_sync_cluster="Galera-Monitor"
config_sync_user=maxscale_sync
config_sync_password=maxscale
admin_secure_gui=false
log_augmentation = 1
```

```
ms_timestamp = 1
syslog = 1


[server1]
type=server
address=10.10.10.11
port=3306
protocol=MariaDBBackend
priority=1

[server2]
type=server
address=10.10.10.12
port=3306
protocol=MariaDBBackend
priority=2

[server3]
type=server
address=10.10.10.13
port=3306
protocol=MariaDBBackend
priority=3


[Galera-Monitor]
type=monitor
module=galeramon
servers=server1,server2,server3
user=maxscaleuser
password=maxscale
monitor_interval=2000ms
use_priority=yes
available_when_donor=true


[Read-Write-Service]
type=service
router=readwritesplit
targets=server1,server2,server3
user=maxscaleuser
password=maxscale
use_sql_variables_in=all
connection_keepalive=300000ms
master_reconnection=true
slave_selection_criteria=least_current_operations
master_failure_mode=fail_on_write
master_accept_reads=false
retry_failed_reads=true

[Read-Write-Listener]
type=listener
service=Read-Write-Service
protocol=MariaDBClient
port=3306
```

En la tabla 5 se describen los parámetros más destacados de la configuración divididos por secciones:

| Parámetro | Descripción |
|---|---|
| Sección maxscale | |
| threads | Se establece el número de hilos en ejecución para procesar las solicitudes entrantes. En auto, maxscale decidirá el valor óptimo en base al núm. de CPUs. |
| config_sync_cluster | Habilita la sincronización automática de los cambios en la configuración entre los nodos en un clúster de MaxScale. El valor debe ser el mismo que el establecido en la sección del monitor y debe coincidir en todos los nodos. |
| config_sync_user | Usuario utilizado para la sincronización, dado de alta anteriormente en MariaDB. |
| config_sync_password | Password del usuario para la sincronización. |
| admin_secure_gui | Securización de la interfaz gráfica de administración del clúster. |
| Sección serverX | |
| type | **server**: se definen y configuran cada uno de los nodos de bases de datos a los que va a servir como proxy MaxScale. |
| address | Se especifica la IP o nombre de host del tipo server. |
| port | Puerto donde escucha el servidor de MariaDB. |
| protocol | Protocolo que utiliza MaxScale para con los servidores de MariaDB. |
| priority | Prioridad que se le asigna al servidor para tomar el rol de master. |
| Sección Galera-Monitor | |
| type | **monitor**: define y configura el tipo monitor, que se encargará de revisar periódicamente el estado de los servidores de bases de datos, proporcionando información sobre su salud y accesibilidad. |
| module | **galeramon**: monitor específico para interactuar y monitorizar el clúster de Galera. |
| servers | Se especifican los servidores que forman el clúster Galera. |
| user | Usuario para la conexión a los servidores de base de datos. Definido anteriormente como ‘maxscaleuser’. |
| password | Contraseña del usuario de conexión. |
| monitor_interval | Intervalo de tiempo en ms en que se revisará el estado de los *backends*. |
| use_priority | Habilitar o deshabilitar el uso de la prioridad en los servidores. |
| available_when_donor | Permite normalidad cuando en el nodo máster cuando adquiera el rol de donante en operaciones de SST non-blocking (Ej: mariabackup). En este caso no habrá conmutación de nodo. |
| Sección Read-Write-Service | |
| type | ***service***: se indica que es un tipo servicio. Cada *router* definido requiere de un servicio y un *listener*. |
| router | **readwritesplit**: *router* específico para la división de la lectura y escritura y el balanceo de carga de las operaciones de lectura. |
| targets | Servidores asociados al servicio. |
| user | Usuario para la conexión en la base de datos (‘maxscaleuser’). |
| password | Contraseña del usuario de conexión. |
| use_sql_variables_in | Especifica dónde deben enrutarse las consultas que leen la variable de sesión (all/master). |
| connection_keepalive | Tiempo máximo para mantener las conexiones de cliente en estado *idle* (inactivas). |
| master_reconnection | Permite que el servidor máster cambie a mitad de una sesión. |
| slave_selection_criteria | Controla cómo el enrutador elige los esclavos a los que se |

| | |
|---|---|
| | conecta y cómo se realiza el equilibrio de carga:<br>• LEAST_CURRENT_OPERATION, el esclavo con menos operaciones activas.<br>• ADAPTIVE_ROUTING, basado en los tiempos de respuesta promedio del servidor.<br>• LEAST_BEHIND_MASTER, el esclavo con el menor retraso de replicación.<br>• LEAST_GLOBAL_CONNECTIONS, el esclavo con menos conexiones de MariaDB MaxScale.<br>• LEAST_ROUTER_CONNECTIONS, el esclavo con menos conexiones de este servicio. |
| master_failure_mode | Controla cómo se maneja el fallo en un servidor maestro:<br>• **fail_instantly** Cuando se detecta el fallo del servidor maestro, la conexión se cerrará inmediatamente.<br>• **fail_on_write** La conexión del cliente se cierra si se recibe una consulta de escritura cuando no hay ningún maestro disponible.<br>• **error_on_write** Si no hay ningún maestro disponible y se recibe una consulta de escritura, se devuelve un error indicando que la conexión está en modo de solo lectura. |
| master_accept_reads | Permite al máster recibir operaciones de lectura. |
| retry_failed_reads | Intenta ejecutar la operación de lectura en otro servidor en caso de fallo. De esta forma el fallo en un nodo standby será transparente para el cliente. |
| Sección Read-Write-Listener | |
| type | ***listener*:** se indica que es un tipo *listener*. Cada *router* definido requiere de un servicio y un *listener*. |
| service | Nombre asociado al servicio de este *listener*. |
| protocol | Protocolo utilizado en el *listener*. MariaDBClient es el optimizado para MariaDB. |
| **port** | **Puerto en el que escucha el *listener* y al que se conectarán los clientes.** |

**Tabla 5 - Configuración de MaxScale**

Una vez configurado MaxScale en el primero de los nodos, podemos arrancar el servicio para comprobar que todo está correctamente configurado con el siguiente comando y revisamos el log en “/var/log/maxscale/maxscale.log”:

```
$ sudo systemctl start maxscale.service

MariaDB MaxScale  /var/log/maxscale/maxscale.log  Thu May  9 19:54:45 2024
------------------------------------------------------------------------------
2024-05-09 19:54:45   notice : (is_directory): /etc/maxscale.cnf.d does not exist, not reading.
2024-05-09 19:54:45   notice : (load_module): Module 'galeramon' loaded from
'/usr/lib/x86_64-linux-gnu/maxscale/libgaleramon.so'.
2024-05-09 19:54:45   notice : (load_module): Module 'readwritesplit' loaded from
'/usr/lib/x86_64-linux-gnu/maxscale/libreadwritesplit.so'.
2024-05-09 19:54:45.049   notice : (mxb_log_set_highprecision_enabled): highprecision
logging is enabled.
2024-05-09 19:54:45.050   notice : (configure): Using up to 146.1MiB of memory for query
classifier cache
2024-05-09 19:54:45.050   notice : (main): syslog logging is enabled.
2024-05-09 19:54:45.050   notice : (main): maxlog logging is enabled.
2024-05-09 19:54:45.050   notice : (maxscale_log_info_blurb): Host: 'maxscale-lb-
01.cluster.lab' OS: Linux@6.1.0-21-cloud-amd64, #1 SMP PREEMPT_DYNAMIC Debian
6.1.90-1 (2024-05-03), x86_64 with 2 processor cores (2.00 available).
2024-05-09 19:54:45.051   notice : (maxscale_log_info_blurb): Total main memory:
```

```
973.98MiB (973.98MiB usable).
2024-05-09 19:54:45.051   notice : (maxscale_log_info_blurb): MaxScale is running in process 2183
2024-05-09 19:54:45.051   notice : (maxscale_log_info_blurb): MariaDB MaxScale 23.08.5 started (Commit: 4bb6b679d2be75ba84226baea656db7a0f330eb4)
2024-05-09 19:54:45.051   notice : (maxscale_log_info_blurb): Transparent hugepages are set to 'always', MaxScale may end up using more memory than it needs. To disable it, set '/sys/kernel/mm/transparent_hugepage/enabled' to 'madvise'
2024-05-09 19:54:45.051   notice : (main): Configuration file: /etc/maxscale.cnf
2024-05-09 19:54:45.051   notice : (main): Log directory: /var/log/maxscale
2024-05-09 19:54:45.051   notice : (main): Data directory: /var/lib/maxscale
2024-05-09 19:54:45.051   notice : (main): Module directory: /usr/lib/x86_64-linux-gnu/maxscale
2024-05-09 19:54:45.051   notice : (main): Service cache: /var/cache/maxscale
2024-05-09 19:54:45.051   notice : (change_cwd): Working directory: /var/log/maxscale
2024-05-09 19:54:45.052   notice : (main): Query classification results are cached and reused. Memory used per thread: 73.05MiB
2024-05-09 19:54:45.052   notice : (load_encryption_keys): Password encryption key file '/var/lib/maxscale/.secrets' not found, using configured passwords as plaintext.
2024-05-09 19:54:45.053   notice : (WatchdogNotifier): The systemd watchdog is Enabled. Internal timeout = 30s\n
2024-05-09 19:54:45.054   notice : (load_module): Module 'pp_sqlite' loaded from '/usr/lib/x86_64-linux-gnu/maxscale/libpp_sqlite.so'.
2024-05-09 19:54:45.054   notice : [MariaDBProtocol] (module_init): Parser plugin loaded.
2024-05-09 19:54:45.056   warning: (ConfigManager); (load_cached_config): Found cached configuration for cluster 'MariaDB-Monitor' when configured to use cluster 'Galera-Monitor', ignoring the cached configuration: /var/lib/maxscale/maxscale-config.json
2024-05-09 19:54:45.058   notice : (auto_detect_algorithm): Using HS256 for JWT signatures
2024-05-09 19:54:45.060   notice : (operator()): Started REST API on [127.0.0.1]:8989
2024-05-09 19:54:45.066   notice : (set_version): server1 sent version string '10.11.7-MariaDB-1:10.11.7+maria~deb12-log'. Detected type: MariaDB, version: 10.11.7.
2024-05-09 19:54:45.074   notice : (mxs_update_server_charset): Server 'server1' charset: utf8mb4_general_ci
2024-05-09 19:54:45.086   notice : (set_version): server2 sent version string '10.11.7-MariaDB-1:10.11.7+maria~deb12-log'. Detected type: MariaDB, version: 10.11.7.
2024-05-09 19:54:45.093   notice : (mxs_update_server_charset): Server 'server2' charset: utf8mb4_general_ci
2024-05-09 19:54:45.105   notice : (set_version): server3 sent version string '10.11.7-MariaDB-1:10.11.7+maria~deb12-log'. Detected type: MariaDB, version: 10.11.7.
2024-05-09 19:54:45.112   notice : (mxs_update_server_charset): Server 'server3' charset: utf8mb4_general_ci
2024-05-09 19:54:45.117   notice : [galeramon] (post_tick): Found cluster members
2024-05-09 19:54:45.122   notice : (launch_all): Starting a total of 1 services...
2024-05-09 19:54:45.122   notice : (Read-Write-Listener); (listen): Listening for connections at [::]:3306
2024-05-09 19:54:45.122   notice : (launch_all): Service 'Read-Write-Service' started (1/1)
2024-05-09 19:54:45.124   notice : (operator()): MaxScale started with 2 worker threads.
2024-05-09 19:54:46.075   notice : (update_users): Read 7 user@host entries from 'server1' for service 'Read-Write-Service'.
```

- Carga las librerías de ‘galeramon’ y ‘readwritesplit’ definidas en la configuración.
- Escaneo de los recursos de la máquina, SO, Host y arranque del proceso.
- Detección de los *backends* de MariaDB definidos en la configuración.
- Arranque de los servicios y de los *listeners* configurados.
- Inicio de MaxScale con el número de hilos en ejecución en base al número de CPUs detectados (2).

### 4.2. Instalación y configuración de Keepalived

Instalaremos a continuación el servicio de Keepalived en la misma máquina y antes de clonar en un segundo nodo para no tener que repetir toda la configuración.

Lo primero será incorporar el mapeo de la IP flotante o Virtual-IP (VIP) al fichero "/etc/hosts" de todas las máquinas, quedando finalmente así configurado:

```
# Galera Mariadb nodes
10.10.10.11     galera-mariadb-01 galera-mariadb-01.cluster.lab
10.10.10.12     galera-mariadb-02 galera-mariadb-02.cluster.lab
10.10.10.13     galera-mariadb-03 galera-mariadb-03.cluster.lab

# MaxScale Proxy nodes
10.10.10.101    maxscale-lb-01 maxscale-lb-01.cluster.lab
10.10.10.102    maxscale-lb-02 maxscale-lb-02.cluster.lab

# Keepalived Virtual IP
10.10.10.100    mariadb.cluster.lab
```

A continuación, se instalará la paquetería de netcat-openbsd y así poder levantar el servicio nc, el cual expondrá el puerto 6000 para el *health-check*, necesario para que funcione correctamente el balanceo en GCP (ver Anexo 2). La monitorización constante del balanceador de carga de GCP sobre este puerto en las dos instancias de MaxScale será el que redirija la VIP hacia un nodo u otro del clúster, el cual se convertirá en el máster de MaxScale.

```
$ sudo apt install netcat-openbsd

$ cat /etc/systemd/system/nc.service
[Unit]
Description=Netcat service for health check

[Service]
ExecStart=/bin/nc -k -l 6000

$ sudo systemctl daemon-reload
$ sudo systemctl start nc.service
$ sudo netstat -putan |grep 6000
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN      905/nc
```

Para el monitor de Keepalived vamos a configurar el siguiente script, el cual comprobará mediante el comando pidof a maxscale si el servicio está o no levantado.

```
$ cat /etc/keepalived/pidof.sh

#!/bin/bash
/usr/bin/pidof maxscale
exit $?
```

```
$ /usr/bin/pidof maxscale
383
```

Básicamente el script lo que devolverá será un 0 o un 1 según encuentre o no PID para maxscale, tal y como se puede comprobar en el comando anterior, donde devolvería un 0 y en caso contrario devolverá un 1, que será el que desencadene la conmutación de la VIP al nodo standby.

El siguiente paso es la instalación de la paquetería necesaria para KeepAlived y su configuración en el fichero "/etc/keepalived/keepalived.conf", destacando la parte de la configuración que diferirá en el nodo standby que clonaremos más adelante:

```
$ sudo apt install keepalived

$ cat /etc/keepalived/keepalived.conf

vrrp_script maxscale {
    script "/etc/keepalived/pidof.sh"
    interval 1
}

vrrp_instance floating_ip {
    state MASTER
    interface ens4
    track_script {
            maxscale
    }
    unicast_src_ip 10.10.10.101
    unicast_peer {
            10.10.10.102
    }
    virtual_router_id 50
    priority 100
    authentication {
            auth_type PASS
            auth_pass maxscale
    }

    notify_master "sudo systemctl start nc"
    notify_backup "sudo systemctl stop nc"
    notify_fault "sudo systemctl stop nc"
}
```

En la sección 'vrrp_script' definimos el monitor comentado anteriormente, con una frecuencia de chequeo de 1 segundo y al que hemos llamado 'maxscale'

En la sección 'vrrp_instance' definimos la instancia de router virtual que vamos a utilizar y a la que hemos llamado 'floating_ip'. Entre sus parámetros destacamos los siguientes:

- **state**: tendrá los valores MASTER o BACKUP, según corresponda. En el otro nodo tendrá el valor BACKUP.
- **interface**: interfaz de red que se va a utilizar

- **track_script**: script que monitoriza la salud del servicio, definido en la sección anterior (maxscale).
- **unicast_src_ip**: IP de origen que utiliza el *router* para enviar los anuncios VRRP, garantizando que los mensajes enviados desde este *router* sean identificados y aceptados por los otros *routers* en la configuración *unicast*.
- **unicast_peer**: IP del *router* BACKUP. Indica los otros *routers* que recibirán los anuncios VRRP desde este *router*.
- **virtual_router_id**: Identificador único para esta instancia de VRRP dentro de una red local. Todos los *routers* VRRP que comparten este ID deben tener la misma configuración de IP virtual.
- **priority**: prioridad del *router*. El *router* con la prioridad más alta será el elegido como maestro. En este caso hemos configurado el maestro con prioridad 100 y el BACKUP con 50.
- **authentication**: configuramos la autenticación tipo PASS y la contraseña que deberá utilizar el resto de los miembros.
- **notify**: script que se ejecutará al cambiar de estado, útil para realizar acciones automáticas dependientes del estado. En nuestro caso, lo que hacemos es levantar o parar el puerto 6000 del servicio nc para el balanceador del GCP, según corresponda. **'notify_master**' arranca el servicio nc, '**notify_backup**' para el servicio nc y '**notify_fault**' para el servicio nc cuando la instancia VRRP de KeepAlived detecta un fallo (por ej. error en la ejecución del script de monitorización o pérdida de conectividad en la interfaz de red).

Tras finalizar con la instalación y configuración de los servicios de MaxScale y Keepalived en el primero de los nodos, clonaremos en un segundo nodo según el software de virtualización utilizado, en nuestro caso, clonamos instancia en GCP. Para el caso de Keepalived habrá que adaptar la configuración según lo comentado en los pasos anteriores.

Por último, arrancamos los servicios de Keepalived en los nodos máster y standby y vemos qué sucede, por defecto el log del servicio Keepalived se puede ver en el syslog del sistema '/var/log/syslog':

```
### maxscale-lb-01 ###

2024-05-10T21:22:00.054198+00:00 localhost systemd[1]: Starting keepalived.service - Keepalive Daemon (LVS and VRRP)...
2024-05-10T21:22:00.067850+00:00 localhost Keepalived[3047]: Starting Keepalived v2.2.7 (01/16,2022)
2024-05-10T21:22:00.068474+00:00 localhost Keepalived[3047]: Running on Linux 6.1.0-21-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.90-1 (2024-05-03) (built for Linux 5.19.11)
2024-05-10T21:22:00.068613+00:00 localhost Keepalived[3047]: Command line: '/usr/sbin/keepalived' '--dont-fork'
2024-05-10T21:22:00.068928+00:00 localhost Keepalived[3047]: Configuration file /etc/keepalived/keepalived.conf
2024-05-10T21:22:00.070522+00:00 localhost Keepalived[3047]: NOTICE: setting config option max_auto_priority should result in better keepalived performance
2024-05-10T21:22:00.071797+00:00 localhost Keepalived[3047]: Starting VRRP child process, pid=3048
```

```
2024-05-10T21:22:00.072376+00:00 localhost systemd[1]: keepalived.service: Got notification message from PID 3048, but reception only permitted for main PID 3047
2024-05-10T21:22:00.073318+00:00 localhost Keepalived_vrrp[3048]: Script user 'keepalived_script' does not exist
2024-05-10T21:22:00.073473+00:00 localhost Keepalived_vrrp[3048]: SECURITY VIOLATION - scripts are being executed but script_security not enabled.
2024-05-10T21:22:00.073552+00:00 localhost Keepalived_vrrp[3048]: (floating_ip) No VIP specified; at least one is sensible
2024-05-10T21:22:00.073621+00:00 localhost Keepalived_vrrp[3048]: (floating_ip) No VIP specified; at least one is sensible
2024-05-10T21:22:00.073843+00:00 localhost Keepalived[3047]: Startup complete
2024-05-10T21:22:00.079675+00:00 localhost systemd[1]: Started keepalived.service - Keepalive Daemon (LVS and VRRP).
2024-05-10T21:22:00.086559+00:00 localhost Keepalived_vrrp[3048]: VRRP_Script(maxscale) succeeded
2024-05-10T21:22:00.086722+00:00 localhost Keepalived_vrrp[3048]: (floating_ip) Entering BACKUP STATE
2024-05-10T21:22:03.697904+00:00 localhost Keepalived_vrrp[3048]: (floating_ip) Entering MASTER STATE
2024-05-10T21:22:03.733032+00:00 localhost systemd[1]: Started nc.service - Netcat service for health check.
```

- Detectado fichero de configuración.
- Arranque del servicio de Keepalived.
- Ejecución del script de pidof correctamente.
- Entra por defecto en estado BACKUP.
- Conmutando a nodo Master por prioridad y al no detectar otro Master.
- Existe cambio de estado, por lo que ejecuta el script 'notify_master' y levanta el servicio nc, que a su vez levanta el puerto 6000 para el balanceo de GCP.

Y comprobamos, a continuación, como el puerto 6000 es accesible a través del nodo (máster) y el 3306 de maxscale a través de la VIP:

```
$ nc -zv maxscale-lb-01 6000
Connection to maxscale-lb-01 (10.10.10.101) 6000 port [tcp/x11] succeeded!

$ nc -zv mariadb.cluster.lab 3306
Connection to mariadb.cluster.lab (10.10.10.100) 3306 port [tcp/mysql] succeeded!
```

Levantamos Keepalived en el nodo standby:

```
### maxscale-lb-02 ###

2024-05-10T21:35:51.825413+00:00 localhost systemd[1]: Starting keepalived.service - Keepalive Daemon (LVS and VRRP)...
2024-05-10T21:35:51.836809+00:00 localhost Keepalived[3643]: Starting Keepalived v2.2.7 (01/16,2022)
2024-05-10T21:35:51.837393+00:00 localhost Keepalived[3643]: Running on Linux 6.1.0-21-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.90-1 (2024-05-03) (built for Linux 5.19.11)
2024-05-10T21:35:51.837524+00:00 localhost Keepalived[3643]: Command line: '/usr/sbin/keepalived' '--dont-fork'
2024-05-10T21:35:51.837608+00:00 localhost Keepalived[3643]: Configuration file /etc/keepalived/keepalived.conf
2024-05-10T21:35:51.838073+00:00 localhost Keepalived[3643]: NOTICE: setting config
```

```
option max_auto_priority should result in better keepalived performance
2024-05-10T21:35:51.838888+00:00 localhost Keepalived[3643]: Starting VRRP child process, pid=3644
2024-05-10T21:35:51.840126+00:00 localhost Keepalived_vrrp[3644]: Script user 'keepalived_script' does not exist
2024-05-10T21:35:51.840285+00:00 localhost Keepalived_vrrp[3644]: SECURITY VIOLATION - scripts are being executed but script_security not enabled.
2024-05-10T21:35:51.840366+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) No VIP specified; at least one is sensible
2024-05-10T21:35:51.840435+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) No VIP specified; at least one is sensible
2024-05-10T21:35:51.840762+00:00 localhost systemd[1]: keepalived.service: Got notification message from PID 3644, but reception only permitted for main PID 3643
2024-05-10T21:35:51.840959+00:00 localhost Keepalived[3643]: Startup complete
2024-05-10T21:35:51.841355+00:00 localhost systemd[1]: Started keepalived.service - Keepalive Daemon (LVS and VRRP).
2024-05-10T21:35:51.852639+00:00 localhost Keepalived_vrrp[3644]: VRRP_Script(maxscale) succeeded
2024-05-10T21:35:51.852913+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) Entering BACKUP STATE
```

- Detección del fichero de configuración.
- Arranque correcto del servicio.
- Ejecución del comando de pidof correctamente.
- Entra por defecto en estado BACKUP.
- No hay conmutación de estado, no se ejecutan los scripts de notify.

Y en efecto comprobamos como el servicio nc no está arrancado y, por tanto, el puerto 6000 no está disponible, luego GCP, lo detectará como caído y no activará la VIP para este nodo:

```
$ nc -zv maxscale-lb-02 6000
nc: connect to maxscale-lb-02 (10.10.10.102) port 6000 (tcp) failed: Connection refused
```

### 4.3. Comandos útiles

En este punto vamos a presentar una serie de comandos útiles de MaxScale, que nos van a servir para la comprobación de la correcta configuración de toda la infraestructura que se ha montado:

**Listado de los servidores** (maxctrl list servers)

```
jlopez@maxscale-lb-01:~$ maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 6 | Slave, Synced, Running | 0-3-35 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 6 | Slave, Synced, Running | 0-3-35 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 6 | Master, Synced, Running | 0-3-35 | Galera-Monitor |

Vemos cada uno de los nodos *backends* de Galera con su configuración definida en maxscale y su estado, tipo Slave o Master y su estado de

sincronización. También el número de conexiones establecidas a cada uno de los servidores.

**Listado de los servicios** (maxctrl list services)

```
jlopez@maxscale-lb-01:~$ maxctrl list services
```

| Service | Router | Connections | Total Connections | Targets |
|---|---|---|---|---|
| Read-Write-Service | readwritesplit | 18 | 36 | server1, server2, server3 |

Listado de los servicios configurado en MaxScale, en nuestro caso el servicio de Read-Write-Service, encargado de la distribución de las operaciones de escritura al nodo Máster y lecturas balanceadas a los nodos Slave. También el número de conexiones establecidas.

**Listado de sesiones activas** (maxctrl list sessions)

```
jlopez@maxscale-lb-01:~$ maxctrl list sessions
```

| Id | User | Host | Connected | Idle | Service | Memory | I/O-Activity |
|---|---|---|---|---|---|---|---|
| 68 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:28 PM | 4.6 | Read-Write-Service | 199328 | 13 |
| 72 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:32 PM | 0.4 | Read-Write-Service | 199328 | 13 |
| 60 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:20 PM | 11.9 | Read-Write-Service | 199328 | 0 |
| 66 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:26 PM | 6.7 | Read-Write-Service | 199328 | 13 |
| 64 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:24 PM | 8.8 | Read-Write-Service | 199328 | 0 |
| 70 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:30 PM | 2.5 | Read-Write-Service | 199328 | 13 |
| 62 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:21 PM | 10.9 | Read-Write-Service | 199328 | 0 |
| 58 | pruebas | 10.10.10.4 | 5/10/2024, 7:56:15 PM | 16.9 | Read-Write-Service | 199328 | 0 |

Lista todas las sesiones activas en MaxScale, donde se puede ver el identificador de la sesión, el usuario conectado, el origen, la hora de la conexión, tiempo desde la última actividad de la sesión, el servicio utilizado de MaxScale, la memoria utilizada y la actividad de entrada y salida.

**Listado y actividad de hilos** (maxctrl list threads)

```
jlopez@maxscale-lb-01:~$ maxctrl list threads
```

| Id | Current FDs | Total FDs | Load (1s) | Load (1m) | Load (1h) |
|---|---|---|---|---|---|
| 0 | 32 | 96 | 0 | 0 | 0 |
| 1 | 64 | 140 | 0 | 0 | 0 |

Muestra la información de cada uno de los hilos en ejecución, descriptores de ficheros actuales y en total, estadísticas de carga de la CPU.

**Mostrar información de MaxScale** (maxctrl show maxscale)

```
jlopez@maxscale-lb-01:~$ maxctrl show maxscale
```

| | |
|---|---|
| Version | 23.08.5 |
| Commit | 4bb6b679d2be75ba84226baea656db7a0f330eb4 |
| Started At | Fri, 10 May 2024 17:58:06 GMT |
| Activated At | Fri, 10 May 2024 17:58:06 GMT |
| Uptime | 8370 |
| Config Sync | { "status": "No configuration changes", "version": 0 } |
| Parameters | { "admin_audit": false, "admin_audit_exclude_methods": [], "admin_audit_file": "/var/log/maxscale/admin_audit.csv", "admin_auth": true, "admin_enabled": true, "admin_gui": true, "admin_host": "127.0.0.1", "admin_jwt_algorithm": "auto", "admin_jwt_issuer": "maxscale", "admin_jwt_key": null, "admin_jwt_max_age": "86400000ms", "admin_log_auth_failures": true, "admin_oidc_url": null, "admin_pam_readonly_service": null, "admin_pam_readwrite_service": null, "admin_port": 8989, "admin_secure_gui": false, "admin_ssl_ca": null, "admin_ssl_cert": null, "admin_ssl_key": null, "admin_ssl_version": "MAX", "admin_verify_url": null, "auth_connect_timeout": "10000ms", "auth_read_timeout": "10000ms", "auth_write_timeout": "10000ms", "auto_tune": [], "cachedir": "/var/cache/maxscale", "config_sync_cluster": "Galera-Monitor", "config_sync_db": "mysql", "config_sync_interval": "5000ms", "config_sync_password": "*****", "config_sync_timeout": "10000ms", "config_sync_user": "maxscale_sync", "connector_plugindir": "/usr/lib/x86_64-linux-gnu/maxscale/plugin", "datadir": "/var/lib/maxscale", "debug": null, "dump_last_statements": "never", "execdir": "/usr/bin", "key_manager": "none", "language": "/var/lib/maxscale", "libdir": "/usr/lib/x86_64-linux-gnu/maxscale", "load_persisted_configs": true, "local_address": null, "log_debug": false, "log_info": false, "log_notice": true, "log_throttling": { |

```
            "count": 10,
            "suppress": 10000,
            "window": 1000
        },
        "log_warn_super_user": false,
        "log_warning": true,
        "logdir": "/var/log/maxscale",
        "max_auth_errors_until_block": 10,
        "max_read_amount": 0,
        "maxlog": true,
        "module_configdir": "/etc/maxscale.modules.d",
        "ms_timestamp": true,
        "passive": false,
        "persist_runtime_changes": true,
        "persistdir": "/var/lib/maxscale/maxscale.cnf.d",
        "piddir": "/var/run/maxscale",
        "query_classifier_cache_size": 153194496,
        "query_retries": 1,
        "query_retry_timeout": "5000ms",
        "rebalance_period": "0ms",
        "rebalance_threshold": 20,
        "rebalance_window": 10,
        "retain_last_statements": 0,
        "session_trace": 0,
        "session_trace_match": null,
        "skip_name_resolve": false,
        "sql_mode": "default",
        "syslog": true,
        "threads": 2,
        "threads_max": 256,
        "users_refresh_interval": "0ms",
        "users_refresh_time": "30000ms",
        "writeq_high_water": 65536,
        "writeq_low_water": 1024
    }
```


System


```
    {
        "machine": {
            "cores_available": 2,
            "cores_physical": 2,
            "cores_virtual": 2,
            "memory_available": 1021296640,
            "memory_physical": 1021296640
        },
        "maxscale": {
            "query_classifier_cache_size": 153194496,
            "threads": 2
        },
        "os": {
            "machine": "x86_64",
            "nodename": "localhost",
            "release": "6.1.0-21-cloud-amd64",
            "sysname": "Linux",
            "version": "#1 SMP PREEMPT_DYNAMIC Debian 6.1.90-1 (2024-05-03)"
        }
    }
```


Vemos toda la información de MaxScale, versión, tiempo arrancado, configuración de sincronización, parámetros, tanto personalizados como por defecto e información del sistema detectada por MaxScale.

**maxctrl show <command>**

Con ‘maxctrl show <command>’ podemos obtener mucha más información detallada sobre los componentes de maxscale: *servers, services, monitors, sessions, filters, listeners modules, threads*. Información en la que no vamos a profundizar más puesto que se haría demasiado extenso y queda fuera del ámbito de este proyecto.

## 4.4. Pruebas de sincronización y failover

### 4.4.1. Sincronización de cambios en configuración

Comprobamos la correcta configuración del mecanismo de sincronización de MaxScale, realizando un cambio cualquiera en la configuración en alguno de los nodos y comprobando en el log del otro nodo que ha detectado y actualizado un cambio nuevo y la versión:

```
### maxscale-lb-01 ###
$ MAXCTRL_WARNINGS=0 maxctrl alter monitor Galera-Monitor monitor_interval=1500ms
OK

### maxscale-lb-02 ###
$ tail -f /var/log/maxscale/maxscale.log
…
2024-05-10 20:48:10.069   notice : (ConfigManager); (sync): Updating to configuration version 3
```

Vemos como el cambio de configuración realizado en el nodo01 lo ha detectado enseguida el nodo02 al comprobar que ha habido un cambio de versión. De igual forma, si capturamos el valor de la variable actualmente en el nodo02, vemos cómo se ha actualizado al valor configurado:

```
jlopez@maxscale-lb-02:~$ maxctrl show monitor Galera-Monitor | grep monitor_interval
    "monitor_interval": "1500ms",
```

Para detectar el cambio en la configuración, MaxScale compara el valor de la versión que tiene almacenado en local, en el fichero “/var/lib/maxscale/maxscale-config.json”, con el que existe en la tabla ‘mysql.maxscale_config’. En caso de no coincidir, se descargará de nuevo toda la configuración de la base de datos al fichero local, que a su vez servirá como caché e incluso para funcionar cuando por algún motivo no pueda establecer comunicación con el clúster Galera.

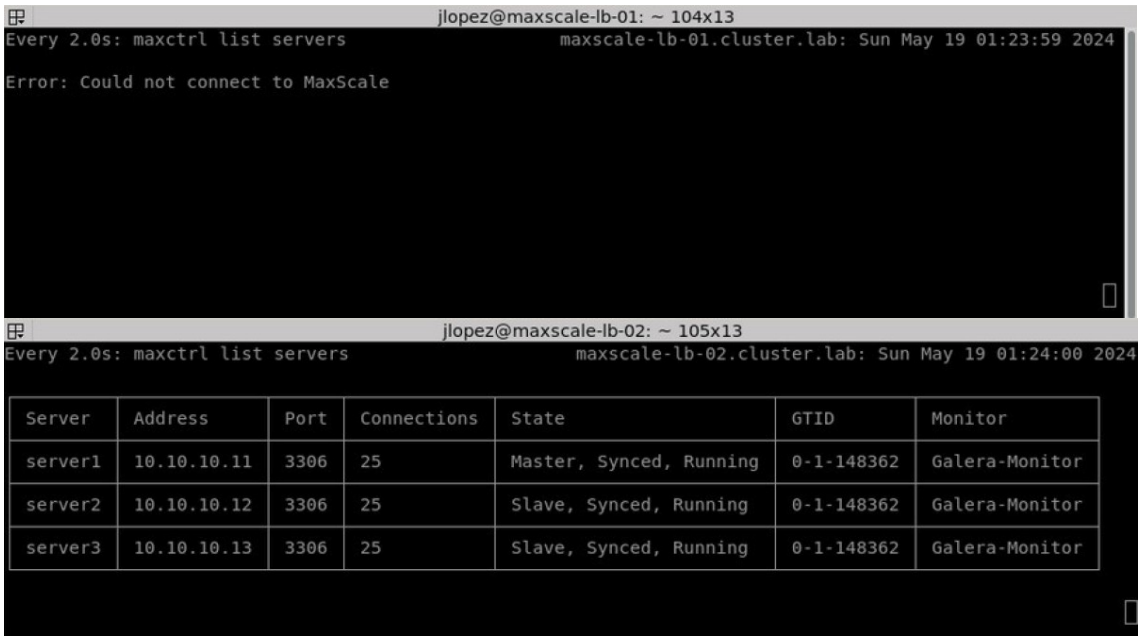
Se puede comprobar también el estado de la configuración con el siguiente comando y en la sección “Config Sync”:

```
$ maxctrl show maxscale
    Config Sync   {
                      "checksum": "022ffe9d105bc65ac970c504b1279794ce9d179b",
                      "nodes": {
                          "maxscale-lb-01.cluster.lab": "OK",
                          "maxscale-lb-02.cluster.lab": "OK"
                      },
                      "origin": "maxscale-lb-01.cluster.lab",
                      "status": "OK",
                      "version": 296
                  }
```

### 4.4.2. Pruebas de conmutación: failover y failback

Para comprobar el correcto funcionamiento del mecanismo de *failover* y conmutación de nodo máster de MaxScale podemos hacer la siguiente prueba.

Simulando un entorno real con conexiones, tenemos el siguiente estado en los dos servidores de MaxScale:




Vemos como todas las conexiones las está atendiendo el nodo01 de MaxScale, mientras que el nodo02 permanece a la espera.

Paramos el servicio de maxscale en el nodo configurado actualmente como máster:

**### maxscale-lb-01 ###**

**jlopez@maxscale-lb-01:$ sudo systemctl stop maxscale.service**

**$ tail -f /var/log/syslog**

2024-05-10T21:52:54.374356+00:00 localhost systemd[1]: Stopping maxscale.service - MariaDB MaxScale Database Proxy...
2024-05-10T21:52:54.478537+00:00 localhost maxscale[1901]: MaxScale is shutting down.
2024-05-10T21:52:54.478852+00:00 localhost maxscale[1901]: Stopped MaxScale REST API
2024-05-10T21:52:54.478979+00:00 localhost maxscale[1901]: All workers have shut down.
**2024-05-10T21:52:54.479284+00:00 localhost maxscale[1901]: MaxScale shutdown completed.**
2024-05-10T21:52:54.479882+00:00 localhost maxscale[1901]: MaxScale received signal SIGTERM. Exiting.
2024-05-10T21:52:54.483896+00:00 localhost systemd[1]: maxscale.service: Deactivated successfully.
2024-05-10T21:52:54.484426+00:00 localhost systemd[1]: Stopped maxscale.service - MariaDB MaxScale Database Proxy.
2024-05-10T21:52:54.485254+00:00 localhost systemd[1]: maxscale.service: Consumed 7.562s CPU time.

```
2024-05-10T21:52:55.324274+00:00 localhost Keepalived_vrrp[3048]: Script `maxscale` now returning 1
2024-05-10T21:52:55.325000+00:00 localhost Keepalived_vrrp[3048]: VRRP_Script(maxscale) failed (exited with status 1)
2024-05-10T21:52:55.325171+00:00 localhost Keepalived_vrrp[3048]: (floating_ip) Entering FAULT STATE
2024-05-10T21:52:55.341117+00:00 localhost systemd[1]: Stopping nc.service - Netcat service for health check...
2024-05-10T21:52:55.342769+00:00 localhost systemd[1]: nc.service: Deactivated successfully.
2024-05-10T21:52:55.343338+00:00 localhost systemd[1]: Stopped nc.service - Netcat service for health check.
```

- Parada correcta de MaxScale
- El script 'maxscale', que ejecuta el pidof ahora devuelve un 1, luego desencadena error.
- Instancia entra en estado FAULT, se ejecuta 'notify_fault'.
- Se para el servicio nc y por tanto el puerto 6000 ya no estará disponible

Comprobamos qué ha ocurrido en el standby:

```
### maxscale-lb-02 ###

$ tail -f /var/log/syslog

2024-05-10T21:52:56.130240+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) Entering MASTER STATE
2024-05-10T21:52:56.169604+00:00 localhost systemd[1]: Started nc.service - Netcat service for health check.
```

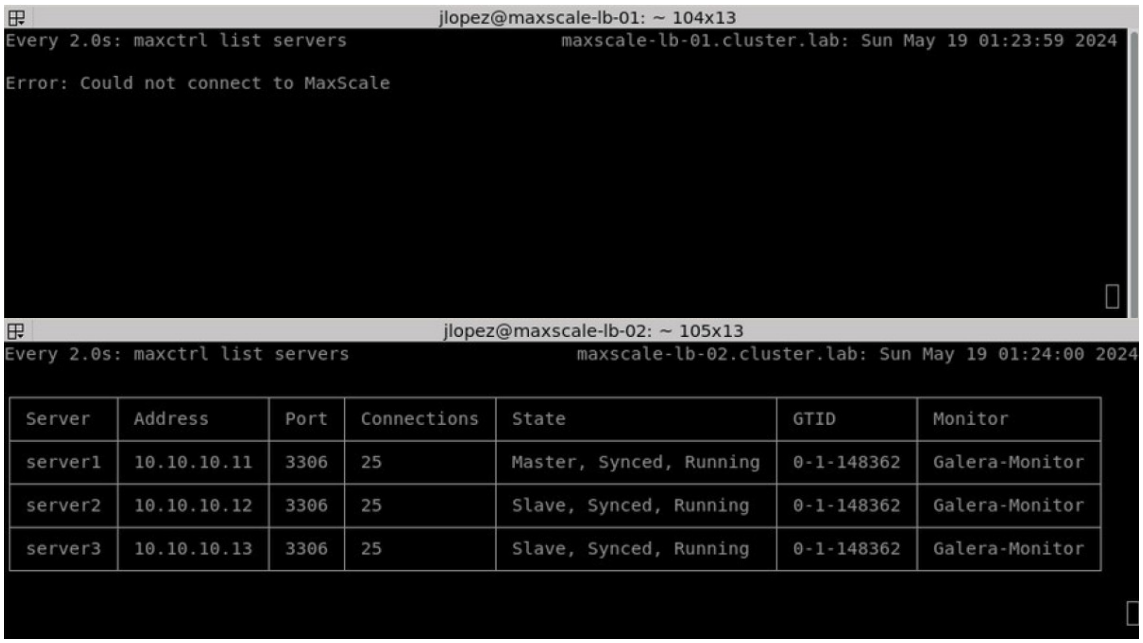
- Nodo entra en estado MASTER, ha habido conmutación de BACKUP a MASTER, por tanto, se ejecuta script 'notify_master'
- Se levanta el servicio nc con el puerto 6000 para GCP.

Comprobamos conectividades:

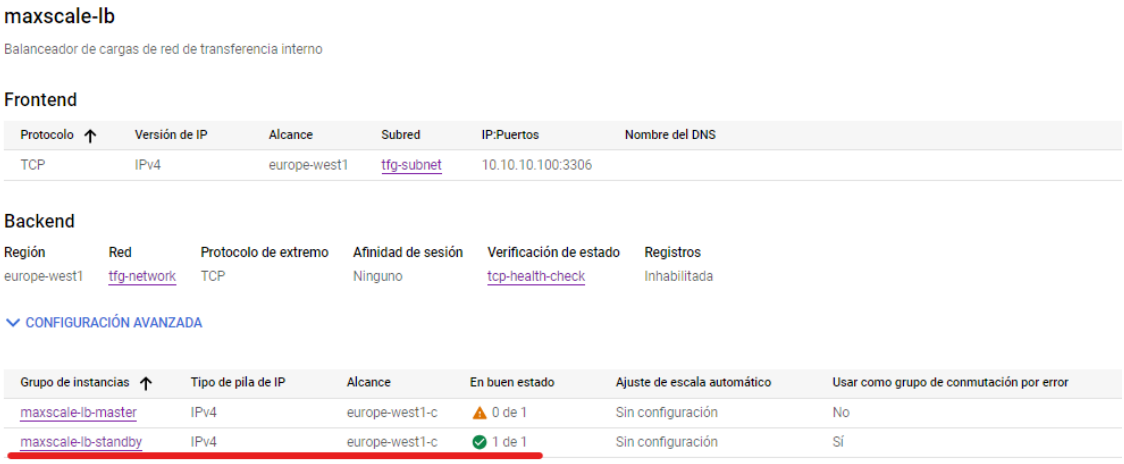
```
$ nc -zv maxscale-lb-01 6000
nc: connect to maxscale-lb-01 (10.10.10.101) port 6000 (tcp) failed: Connection refused

$ nc -zv maxscale-lb-02 6000
Connection to maxscale-lb-02 (10.10.10.102) 6000 port [tcp/x11] succeeded!

$ nc -zv mariadb.cluster.lab 3306
Connection to mariadb.cluster.lab (10.10.10.100) 3306 port [tcp/mysql] succeeded!
```

- maxscale-lb-01 (anterior nodo máster) rechaza la conexión al puerto 6000.
- maxscale-lb-02 (anterior nodo standby) acepta la conexión al puerto 6000.
- La VIP continúa respondiendo y la conexión al puerto 3306 es correcta, pero esta vez es el nodo maxscale.lb-02 el que la ofrece.

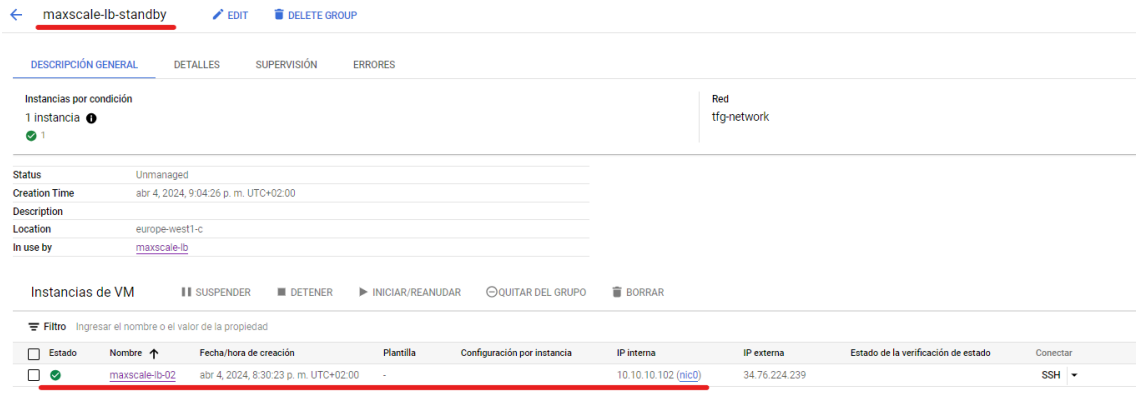
Y comprobamos el entorno simulado, donde vemos que todas las conexiones se han recreado en el nodo maxscale-lb-02. En este caso, al tratarse de una irrupción en el servicio, las conexiones se han cerrado abruptamente con el nodo01, dando error de conexión en el cliente y ocasionando pérdida de servicio durante el proceso de conmutación de nodo.




Hemos comprobado de esta forma como el servicio de Keepalived ha conmutado de nodo la VIP y el balanceo de GCP redirige correctamente las peticiones de la IP flotante hacia la instancia maxscale-lb-02. Podemos verlo también en la siguiente captura en GCP:




Donde el grupo de instancias “maxscale-lb-standby” es el que se corresponde con el nodo maxscale-lb-02 e IP 10.10.10.102.




Para terminar, comprobamos también qué ocurre en el procedimiento de *failback*, en el cual volvemos a levantar el servicio de maxscale en el nodo maxscale-lb-01, antiguo nodo máster:

```
### maxscale-lb-01 ###

$ sudo systemctl start maxscale.service

…
2024-05-10T22:27:27.982573+00:00 localhost maxscale[11329]: MariaDB MaxScale
23.08.5 started (Commit: 4bb6b679d2be75ba84226baea656db7a0f330eb4)
…
2024-05-10T22:27:27.985362+00:00 localhost maxscale[11329]: Using cached
configuration for cluster 'Galera-Monitor', version 3: /var/lib/maxscale/maxscale-
config.json
…
2024-05-10T22:27:28.027977+00:00 localhost maxscale[11329]: Found cluster members
2024-05-10T22:27:28.628947+00:00 localhost Keepalived_vrrp[3048]: Script `maxscale`
now returning 0
2024-05-10T22:27:28.629494+00:00 localhost Keepalived_vrrp[3048]:
VRRP_Script(maxscale) succeeded
2024-05-10T22:27:28.629597+00:00 localhost Keepalived_vrrp[3048]: (floating_ip)
Entering BACKUP STATE
2024-05-10T22:27:28.995877+00:00 localhost maxscale[11329]: Read 7 user@host entries
from 'server3' for service 'Read-Write-Service'.
2024-05-10T22:27:29.435064+00:00 localhost Keepalived_vrrp[3048]: (floating_ip)
received lower priority (50) advert from 10.10.10.102 - discarding
2024-05-10T22:27:30.435354+00:00 localhost Keepalived_vrrp[3048]: (floating_ip)
received lower priority (50) advert from 10.10.10.102 - discarding
2024-05-10T22:27:31.435577+00:00 localhost Keepalived_vrrp[3048]: (floating_ip)
received lower priority (50) advert from 10.10.10.102 - discarding
2024-05-10T22:27:32.238447+00:00 localhost Keepalived_vrrp[3048]: (floating_ip)
Entering MASTER STATE
2024-05-10T22:27:32.272621+00:00 localhost systemd[1]: Started nc.service - Netcat
service for health check.
```

- Maxscale arrancado
- No detecta cambio de versión en la configuración, luego utiliza la configuración almacenada localmente en caché en el fichero '/var/lib/maxscale/maxscale-config.json'
- El script 'maxscale' ahora devuelve un 0, se ha encontrado PID para maxscale.
- Se entra por defecto en estado BACKUP.

- El nodo tiene mayor prioridad (100) que el actual MASTER y advierte por 3 veces.
- Conmuta a estado MASTER y por tanto adquiere la VIP.
- Arranca el servicio nc para informar al balanceador de GCP que las peticiones a la VIP las atenderá este nodo.

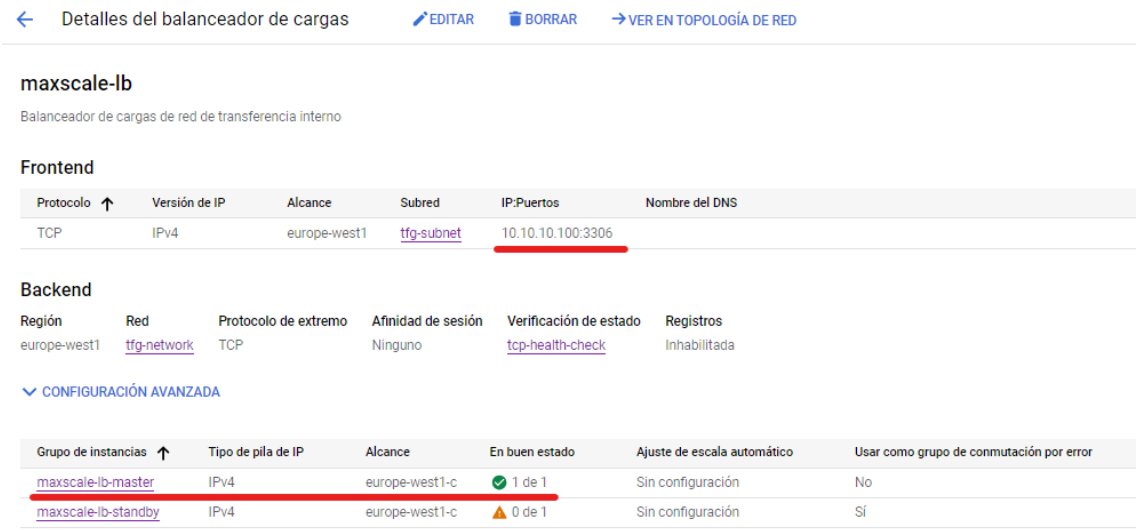
```
### maxscale-lb-02 ###

2024-05-10T22:27:32.239844+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) Master received advert from 10.10.10.101 with higher priority 100, ours 50
2024-05-10T22:27:32.240106+00:00 localhost Keepalived_vrrp[3644]: (floating_ip) Entering BACKUP STATE
2024-05-10T22:27:32.257358+00:00 localhost systemd[1]: Stopping nc.service - Netcat service for health check...
2024-05-10T22:27:32.259101+00:00 localhost systemd[1]: nc.service: Deactivated successfully.
2024-05-10T22:27:32.259280+00:00 localhost systemd[1]: Stopped nc.service - Netcat service for health check.
```

- Nodo recibe advertencia de otra IP con mayor prioridad (100 sobre 50).
- Entra en estado BACKUP.
- Hay conmutación a BACKUP, en este caso se ejecuta ‘notify_backup’ y se para el servicio nc, por tanto, el puerto 6000 deja de estar disponible y el nodo pierde la VIP debido al balanceo de GCP.

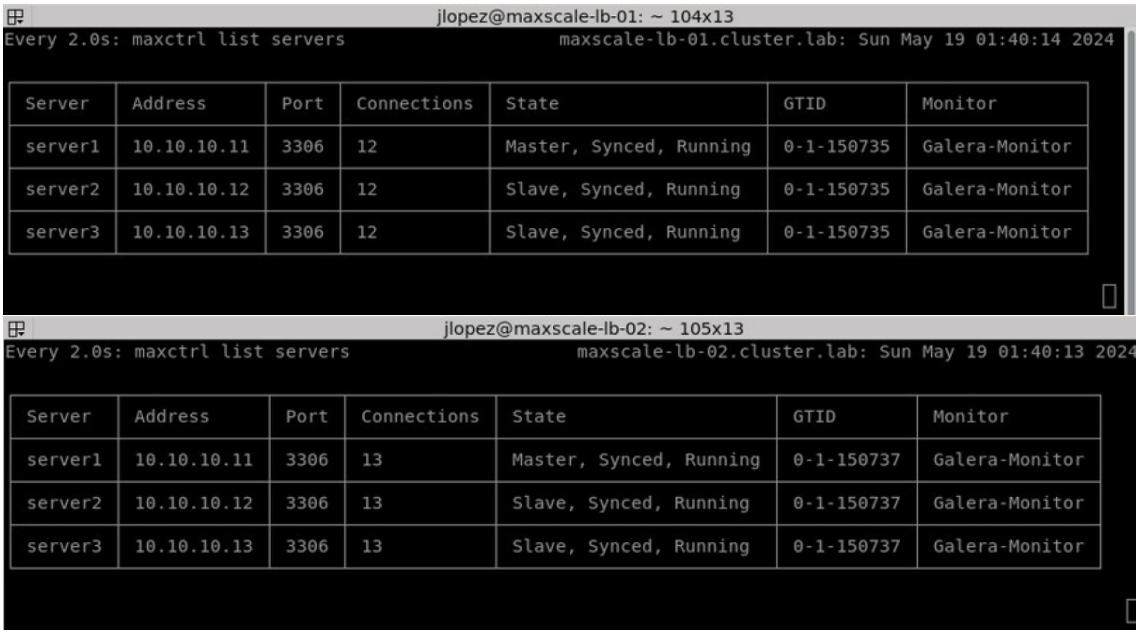
Comprobamos conectividades

```
$ nc -zv maxscale-lb-02 6000
nc: connect to maxscale-lb-02 (10.10.10.102) port 6000 (tcp) failed: Connection refused

$ nc -zv maxscale-lb-01 6000
Connection to maxscale-lb-01 (10.10.10.101) 6000 port [tcp/x11] succeeded!

$ nc -zv mariadb.cluster.lab 3306
Connection to mariadb.cluster.lab (10.10.10.100) 3306 port [tcp/mysql] succeeded!
```

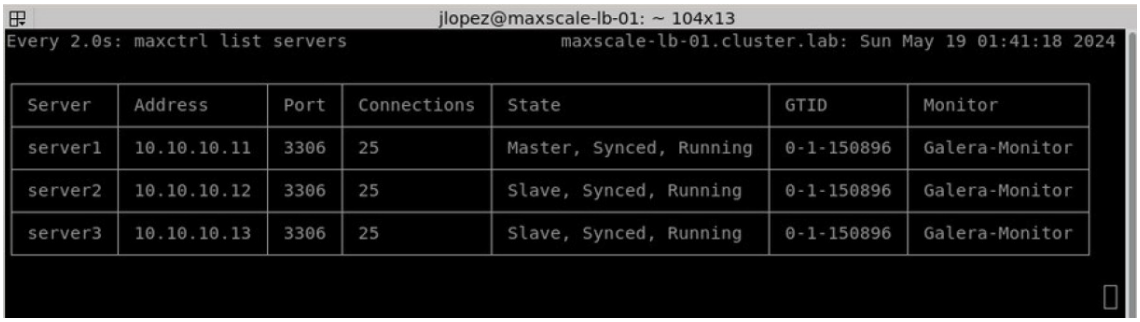
- maxscale-lb-02 rechaza conexiones al puerto 6000.
- maxscale-lb-01 acepta conexiones al puerto 6000.
- La VIP continúa disponible y esta vez atendida por maxscale-lb-01, tal y como podemos comprobar en la siguiente captura del balanceo en GCP:

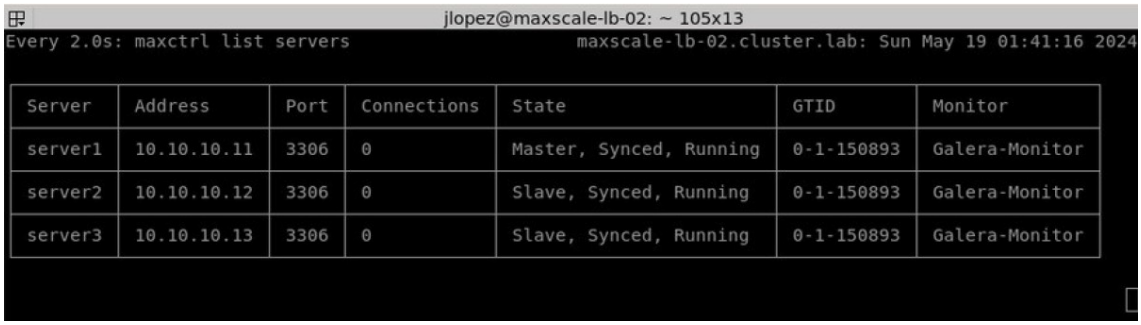



En cuanto al entorno simulado, comprobamos como el procedimiento de *failback* y la conmutación de nodo se produce de forma más suave que en el *failover*, sin ocasionar pérdida de servicio. Las conexiones van finalizando correctamente y sin errores en maxscale-lb-02 y recreándose en el nuevo nodo máster, maxscale-lb-01.




Hasta llegar a cero conexiones en maxscale-lb-02 y volver así a entrar en estado de *standby*.




jlopez@maxscale-lb-02: ~ 105x13
Every 2.0s: maxctrl list servers
maxscale-lb-02.cluster.lab: Sun May 19 01:41:16 2024
Server | Address | Port | Connections | State | GTID | Monitor
server1 | 10.10.10.11 | 3306 | 0 | Master, Synced, Running | 0-1-150893 | Galera-Monitor
server2 | 10.10.10.12 | 3306 | 0 | Slave, Synced, Running | 0-1-150893 | Galera-Monitor
server3 | 10.10.10.13 | 3306 | 0 | Slave, Synced, Running | 0-1-150893 | Galera-Monitor

# 5. Plan de contingencias

El plan de contingencias va a fundamentarse, principalmente, en una robusta y fiable **estrategia de copias de seguridad** del SGBD, la cual poder restaurar sin problema y en el menor tiempo posible en caso de necesidad o desastre en el SGBD.

De la misma forma se habrá de contar con una política de **retención de backups**, en la que se incluye la frecuencia de las copias de seguridad y la duración del almacenamiento. En nuestro caso se creará una copia diaria y en horario de mínima actividad laboral, la madrugada del lunes se realizará una copia completa y el resto de los días incremental. La retención de los *backups* será de un mes en local y a partir de entonces, pasando a cinta y armario ignífugo con una retención máxima de dos años.

Se cuenta también con un **plan de recuperación** bien detallado con todas las políticas y procedimientos y fácilmente accesible para todo el equipo responsable. El plan de recuperación se debe actualizar cada vez que haya algún cambio en la infraestructura.

Por último, una vez al año, se realizará una **prueba de restauración** completa y planificada de toda la infraestructura para confirmar que el plan es efectivo y no nos encontraremos con ninguna “sorpresa” cuando haya que recurrir a él.

## 5.1. Estrategia de backup y restauración.

Para diseñar una estrategia de *backup* adecuada para nuestra infraestructura de Galera Cluster, se dispondrá de un servidor dedicado y con conectividad por ssh (puerto 22) de los tres nodos que forman el clúster de MariaDB. Utilizaremos la compartición de claves públicas entre el servidor de *backup* y los nodos Galera para mayor seguridad y las copias de seguridad se almacenarán sobre un volumen NFS, accesible desde los tres nodos con *automount*.

Se realizarán *backups*, tanto incrementales como de tipo full. Los de tipo full tendrán una frecuencia semanal y los incrementales se realizarán diariamente. La herramienta escogida para realizar las copias de seguridad será Mariabackup, ya que no bloquea las tablas durante el proceso de copia de seguridad y tiene soporte para *backups* incrementales. No obstante, Mariabackup tiene un inconveniente y es que no se puede ejecutar desde un servidor remoto, ese es el motivo por el que necesitaremos conectividad por ssh desde el servidor de *backup*, así que habrá que adaptar el script para que los comandos se ejecuten localmente, en los nodos de Galera, a través de ssh.

Empezaremos por comprobar la conectividad ssh entre el servidor de *backup* y los nodos de Galera:

```
jlopez@client:~$ nc -zv galera-mariadb-01.cluster.lab 22
Connection to galera-mariadb-01.cluster.lab (10.10.10.11) 22 port [tcp/mysql] succeeded!

jlopez@client:~$ nc -zv galera-mariadb-02.cluster.lab 22
Connection to galera-mariadb-02.cluster.lab (10.10.10.12) 22 port [tcp/ssh] succeeded!

jlopez@client:~$ nc -zv galera-mariadb-03.cluster.lab 22
Connection to galera-mariadb-03.cluster.lab (10.10.10.13) 22 port [tcp/ssh] succeeded!!
```

Vamos a generar el par de claves, pública y privada, en el servidor de *backup*:

```
root@client:/root# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:YKzGytZKcKvRrfY8ExcL/UVmPBM6n4Rg6ok5PeVzOgg root@client.cluster.lab
The key's randomart image is:
+---[RSA 4096]----+
|      o  ...     |
|     + . oB      |
|    ..= ++.o     |
|   *.*o. +..     |
|. E O.o+S.o      |
| = O.oo+.        |
|. B +oo          |
| =.+o  .         |
|..o.oo           |
+----[SHA256]-----+
```

En los servidores remotos de MariaDB, nos aseguramos de que los siguientes parámetros estén configurados en el fichero de configuración de ssh '/etc/ssh/sshd_config', caso de hacer algún cambio habrá que reiniciar el servicio:

```
PermitRootLogin yes
PasswordAuthentication yes
PubkeyAuthentication yes
```

Y copiamos la clave pública a los tres nodos remotos utilizando usuario/password:

```
root@client:/root# ssh-copy-id -i ~/.ssh/id_rsa.pub root@galera-mariadb-01
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@galera-mariadb-01's password:

Number of key(s) added: 1
```

```
Now try logging into the machine, with:   "ssh 'root@galera-mariadb-01'"
and check to make sure that only the key(s) you wanted were added.
```

Comprobamos la conexión con la clave pública y repetiremos la operación para los otros dos nodos de MariaDB:

```
root@client:/root# ssh root@galera-mariadb-01
Linux galera-mariadb-01.cluster.lab 6.1.0-21-cloud-amd64 #1 SMP PREEMPT_DYNAMIC
Debian 6.1.90-1 (2024-05-03) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Mar 31 13:36:59 2024 from 10.10.10.5
root@galera-mariadb-01:~#
```

A continuación, vamos a configurar en los tres nodos de MariaDB el cliente para montar el volumen NFS donde almacenar los backups. Dicho volumen ya lo tenemos desde “10.10.10.4:/var/backups/galera” y se montará con autofs según necesidad. Tras un periodo de inactividad de 600 segundos el volumen se desmontará automáticamente.

Empezamos por instalar autofs en los MariaDB:

```
$ sudo apt install autofs
```

Creamos el directorio donde se montará el *shareddir*, configuramos autofs en los ficheros ‘/etc/auto.master’ y ‘/etc/auto.nfs’ y arrancamos el servicio:

```
$ sudo mkdir /mnt/backups

$ tail -1 /etc/auto.master
/mnt/backups    /etc/auto.nfs --timeout=600

$ cat /etc/auto.nfs
galera  -fstype=nfs,rw  10.10.10.4:/var/backups/galera

$ sudo systemctl start autofs.service
```

En ‘/mnt/backups’ especificamos donde montar el *filesystem*, le indicamos los detalles en el fichero ‘/etc/auto.nfs’ y el *timeout* para desmontar automáticamente en 600 segundos. En el fichero ‘/etc/auto.nfs’ especificamos el nombre del *shareddir* a montar (galera), el tipo de *filesystem* con lectura/escritura y el origen de la conexión.

Comprobamos, accediendo al *shareddir*, para que se auto monte y con el comando *mount* observamos que el volumen NFS está correctamente montado y accesible con lectura/escritura:

```
$ cd /mnt/backups/galera
```

```
$ mount | grep nfs
/etc/auto.nfs on /mnt/backups type autofs
(rw,relatime,fd=6,pgrp=3908,timeout=600,minproto=5,maxproto=5,indirect,pipe_ino=24069)
10.10.10.4:/var/backups/galera on /mnt/backups/galera type nfs4
(rw,relatime,vers=4.2,rsize=131072,wsize=131072,namlen=255,hard,proto=tcp,timeo=600,retra
ns=2,sec=sys,clientaddr=10.10.10.12,local_lock=none,addr=10.10.10.4)
```

Al cabo de 600 segundos de inactividad veremos en el syslog del sistema la siguiente línea, que indica que el *filesystem* se ha desmontado correctamente:

```
2024-05-14T20:39:10.740310+02:00 localhost systemd[1]: mnt-backups-
galera.mount: Deactivated successfully.
```

Continuamos con la creación de un usuario específico para realizar las copias de seguridad sobre una de las instancias de MariaDB, que se replicará al resto de nodos del clúster automáticamente. Este usuario deberá tener los permisos de RELOAD, LOCK TABLES, REPLICATION CLIENT, PROCESS y SELECT.

```
MariaDB [(none)]> CREATE USER 'backupuser'@'localhost' IDENTIFIED
BY 'backuppass';
Query OK, 0 rows affected (0.011 sec)

MariaDB [(none)]> GRANT RELOAD, LOCK TABLES, REPLICATION
CLIENT, PROCESS, SELECT on *.* TO 'backupuser'@'localhost';
Query OK, 0 rows affected (0.010 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.009 sec)
```

Para mayor seguridad y no tener que disponer de datos sensibles en plano en el script que lanzará las copias de seguridad, se va a generar un fichero oculto en el home del usuario root de los tres nodos de Galera con la información necesaria para la conexión. Al encontrarse dentro del directorio “/root”, oculto y con permisos de 600 nos aseguramos de que tan sólo el usuario root es capaz de leer dicha información, por lo que no se considera necesario encriptar ningún dato:

```
$ sudo cat /root/.my.cnf
[mariabackup]
user=backupuser
password=backuppass
host=localhost
port=3306

$ sudo chmod 600 /root/.my.cnf

$ sudo ls -l /root/.my.cnf
-rw------- 1 root root 75 May 11 21:56 /root/.my.cnf
```

Para lanzar la copia de seguridad full o incremental en base al día de la semana, se ha creado el siguiente script en bash, que hemos colocado dentro del volumen NFS en la ruta ‘/mnt/backups/galera/scripts/backup_galera.sh’:

```
#!/bin/bash

# Configuración de las rutas
BACKUP_DIR="/mnt/backups/galera"
FULL_BACKUP_DIR="${BACKUP_DIR}/full"
INCREMENTAL_BACKUP_DIR="${BACKUP_DIR}/incremental"
LAST_FULL_BACKUP="${BACKUP_DIR}/last_full"

# Crear directorios si no existen
mkdir -p "${FULL_BACKUP_DIR}"
mkdir -p "${INCREMENTAL_BACKUP_DIR}"

# Determinar si hoy es el día para un backup completo
DAY_OF_WEEK=$(date +%u)  # %u es el día de la semana (1=lunes, 7=domingo)

# Se realiza backup full la madrugada del domingo al lunes, incremental el resto.
if [ "$DAY_OF_WEEK" -eq 1 ]; then
    # Realizar un backup completo
    mariabackup --backup --target-dir="${FULL_BACKUP_DIR}/$(date +%F)"
    # Actualizar el puntero al último backup completo
    echo "$(date +%F)" > "${LAST_FULL_BACKUP}"
else
    # Realizar un backup incremental
    if [ -f "${LAST_FULL_BACKUP}" ]; then
        LAST_DATE=$(cat "${LAST_FULL_BACKUP}")
        mariabackup --backup --target-dir="${INCREMENTAL_BACKUP_DIR}/$(date +%F)" --
incremental-basedir="${FULL_BACKUP_DIR}/${LAST_DATE}"
    else
        echo "No se encontró un backup full previo. Abortando backup incremental."
        exit 1
    fi
fi
```

El script comienza por configurar las rutas, separando los directorios para las copias de seguridad full e incremental. Determina el día de la semana y, si el día es 1 lanza un *backup* completo, si no, lanzará una copia incremental siempre que exista un full previo. Las copias se guardarán en su directorio correspondiente estructuradas por fecha.

Desde el servidor de backup se ha creado el siguiente cron, que se ejecutará todos los días a las 02:00h AM.

```
$ sudo crontab -l
0 2 * * * /var/backups/galera/scripts/exec_backup.sh >
/var/backups/galera/logs/backup_galera_$(date +\%F).log 2>&1
```

Dicho cron ejecuta el script “/var/backups/galera/scripts/exec_backup.sh” en el servidor de backup, redirigiendo la salida al fichero de log:

```
$ sudo cat /var/backups/galera/scripts/exec_backup.sh
#!/bin/bash

# Lista de servidores a verificar
servers=("galera-mariadb-01.cluster.lab" "galera-mariadb-02.cluster.lab" "galera-mariadb-
03.cluster.lab")
```

```
# Timeout para la conexión nc en segundos
timeout=5

# Función para comprobar la conectividad SSH usando nc
check_ssh() {
    local server=$1
    if nc -z -w $timeout "$server" 22; then
#        echo "$server"
        return 0
    else
        return 1
    fi
}

# Comprobar la conectividad de cada servidor
for server in "${servers[@]}"; do
    if check_ssh "$server"; then
        echo "Se va a realizar backup en el servidor: $server"
            /usr/bin/ssh -q -o "StrictHostKeyChecking no" root@$server
/mnt/backups/galera/scripts/backup_galera.sh
            exit 0
    fi
done

echo "No hay conectividad a ninguno de los servidores SSH, no se realizará backup."
exit 1
```

El script comprueba la conectividad por ssh de una lista con los tres servidores de MariaDB y en el primero que encuentra con visibilidad al puerto 22, lanzará en remoto, mediante ssh y claves públicas, el script de *backup* configurado anteriormente.

Finalmente, y tras la ejecución del cron de *backup* durante tres días, tendremos un árbol de directorios en el volumen NFS como el siguiente:

```
$ sudo tree -L 2 /var/backups/galera/
/var/backups/galera/
├── full
│   └── 2024-05-13
├── incremental
│   ├── 2024-05-14
│   └── 2024-05-15
├── last_full
├── logs
│   ├── backup_galera_2024-05-13.log
│   ├── backup_galera_2024-05-14.log
│   └── backup_galera_2024-05-15.log
└── scripts
    ├── backup_galera.sh
    └── exec_backup.sh

8 directories, 6 files
```

El procedimiento para la restauración se puede ver en el laboratorio del punto 6.2 Point-in-Time Recovery (PITR).

# 6. Laboratorio de soluciones propuestas

## 6.1. Operaciones de mantenimiento con zero-downtime

Antes de comenzar con el laboratorio, es necesario indicar que, por la naturaleza de funcionamiento de MaxScale con el enrutador 'readwritesplit' (*máster*/*standby*), tan sólo las operaciones de lectura serán balanceadas y enrutadas a todos los nodos disponibles del clúster de Galera, de manera que en todo momento existirá un nodo disponible para atender estas operaciones. En cambio, las operaciones de escritura sólo cuentan con un nodo máster, que será el que reciba todas las operaciones de este tipo, de forma que es posible que, en el breve periodo de transición durante la conmutación de un nodo máster a otro, es posible que se pierda alguna operación de escritura, debido al cierre de las conexiones en el antiguo máster y apertura en el nuevo máster. Esta restricción de MaxScale es necesaria para mantener la consistencia de los datos, evitando así conflictos de escritura, donde diferentes nodos podrían aceptar cambios contradictorios.

Para tratar de reducir al máximo este tipo de errores se utilizarán, en tiempo de ejecución, los parámetros de *priority* de los servidores y los modos de mantenimiento (*maintenance*) y liberación de conexiones (*drain*) para controlar en todo momento cuándo y hacia que nodo conmutará el rol de máster. Para alterar estos valores en tiempo de ejecución se utilizará el comando 'maxctrl' con *alter* y *set*, tal y como veremos a continuación.

Este primer laboratorio va a consistir en una subida de versión (*major upgrade*) de todos los nodos *backend* del clúster de Galera, sin pérdida de servicio (*zero-downtime*) o con la menor pérdida posible. Se partirá de una versión 10.11.7 de MariaDB y se actualizará a una versión 11.0.6, anunciada como *short-term release* y mantenida por un año.

Indicar también que las evidencias y monitorización para comprobar el estado del servicio durante todo el proceso de actualización se han realizado con el software Apache JMeter, su configuración se puede ver en el Anexo 3.

Una vez establecidas todas las premisas, comenzamos con el laboratorio. En primer lugar, vamos a realizar un chequeo de salud de todas las bases de datos en los tres nodos:

```
$ sudo mysqlcheck -u root -pmariadb --all-databases
mysql.column_stats                 OK
mysql.columns_priv                 OK
mysql.db                           OK
mysql.event                        OK
mysql.func                         OK
mysql.global_priv                  OK
mysql.gtid_slave_pos               OK
mysql.help_category                OK
mysql.help_keyword                 OK
```

```
mysql.help_relation                 OK
mysql.help_topic                    OK
mysql.index_stats                   OK
mysql.innodb_index_stats               OK
mysql.innodb_table_stats               OK
mysql.maxscale_config                  OK
mysql.plugin                        OK
mysql.proc                          OK
mysql.procs_priv                    OK
mysql.proxies_priv                  OK
mysql.roles_mapping                    OK
mysql.servers                       OK
mysql.table_stats                   OK
mysql.tables_priv                   OK
mysql.time_zone                     OK
mysql.time_zone_leap_second                OK
mysql.time_zone_name                    OK
mysql.time_zone_transition              OK
mysql.time_zone_transition_type            OK
mysql.transaction_registry              OK
mysql.wsrep_allowlist                   OK
mysql.wsrep_cluster                     OK
mysql.wsrep_cluster_members                OK
mysql.wsrep_streaming_log               OK
sys.sys_config                      OK
test.test_table                     OK
testdb.sbtest1                      OK
testdb.sbtest2                      OK
testdb.sbtest3                      OK
testdb.sbtest4                      OK
testdb.sbtest5                      OK
```

Y comprobamos, en MaxScale, el estado de sincronización de Galera con el siguiente comando, donde se puede ver en la captura como todos los nodos Galera están sincronizados y trabajando correctamente:

```
jlopez@maxscale-lb-01:~$ maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 17 | Master, Synced, Running | 0-1-152086 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 17 | Slave, Synced, Running | 0-1-152086 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 17 | Slave, Synced, Running | 0-1-152086 | Galera-Monitor |

Continuamos comprobando la versión de MariaDB previa a la actualización:

```
$ mariadb --version
mariadb  Ver 15.1 Distrib 10.11.7-MariaDB, for debian-linux-gnu (x86_64) using  EditLine wrapper
```

Y actualizamos los repositorios con las fuentes oficiales de MariaDB:

```
$ curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash -s -- --mariadb-server-version="mariadb-11.0"
# [info] Checking for script prerequisites.
# [info] MariaDB Server version 11.0 is valid
# [info] Repository file successfully written to /etc/apt/sources.list.d/mariadb.list
```

```
# [info] Adding trusted package signing keys...
# [info] Running apt-get update...
# [info] Done adding trusted package signing keys
```

Comprobamos el listado de paquetes actualizables y vemos como ya nos aparecen como candidatos la versión 11.0 de los paquetes de mariadb:

```
$ sudo apt list --upgradable
Listing... Done
galera-4/unknown,unknown 26.4.18-deb12 amd64 [upgradable from: 26.4.16-deb12]
google-cloud-cli/cloud-sdk-bookworm 476.0.0-0 amd64 [upgradable from: 468.0.0-0]
google-cloud-packages-archive-keyring/google-compute-engine-bookworm-stable 1.2-
629101324 all [upgradable from: 1.2-614705963]
google-compute-engine-oslogin/google-compute-engine-bookworm-stable 1:20240415.00-
g1+deb12 amd64 [upgradable from: 1:20231004.00-g1+deb12]
google-osconfig-agent/google-compute-engine-bookworm-stable 1:20240501.03-g1 amd64
[upgradable from: 1:20231207.01-g1]
libmariadb3/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mariadb-backup/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mariadb-client-core/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mariadb-client/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mariadb-common/unknown,unknown 1:11.0.6+maria~deb12 all [upgradable from:
1:10.11.7+maria~deb12]
mariadb-server-core/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mariadb-server/unknown 1:11.0.6+maria~deb12 amd64 [upgradable from:
1:10.11.7+maria~deb12]
mysql-common/unknown,unknown 1:11.0.6+maria~deb12 all [upgradable from:
1:10.11.7+maria~deb12]
```

Antes de comenzar con el proceso de actualización vamos a asegurarnos de que tenemos un respaldo actualizado para apoyarnos en caso de necesidad. Contamos con las copias de seguridad diarias configuradas en el punto 5.1, no obstante, vamos a lanzar otra copia con mysqldump en cualquiera de los tres nodos para disponer también de un *backup* lo más reciente posible.

```
$ mysqldump -u root -pmariadb --all-databases --single-transaction >
/mnt/backups/galera/LAB1/mariadb_backup_$(date +\%F).sql
```

También realizamos copia de los ficheros de configuración de los tres nodos:

```
$ cd /etc/mysql/
$ tar zcvf /mnt/backups/galera/LAB1/$(hostname --fqdn)_conf_$(date +\
%F).tgz .
```

Desde un terminal, lanzaremos también *snapshot* de los discos de las tres instancias GCP de MariaDB, para poder volver rápidamente al estado de inicio del laboratorio en caso necesario.

```
$ for i in galera-mariadb-01 galera-mariadb-02 galera-mariadb-03
> do
```

```
> gcloud compute \
> --project "quixotic-skill-418320" \
> disks snapshot $i \
> --snapshot-names snap-$i-lab1-$(date +\%F) \
> --zone "europe-west1-b" \
> --description "Snapshot de $i previo a LAB1."
> done
Creating snapshot(s) snap-galera-mariadb-01-lab1-2024-05-19...done.
Creating snapshot(s) snap-galera-mariadb-02-lab1-2024-05-19...done.
Creating snapshot(s) snap-galera-mariadb-03-lab1-2024-05-19...done.
```

Comprobamos también, en GCP, como los *snapshots* se han creado correctamente.


Comenzaremos por el nodo 'galera-mariadb-03', el cual hemos visto en la captura anterior de MaxScale que tiene el rol de standby, por lo que no será disruptivo y se puede actualizar tranquilamente. Empezamos por marcarlo como no candidato a conmutar a nodo máster, asignándole prioridad "-1" en MaxScale:

```
$ MAXCTRL_WARNINGS=0 maxctrl alter server server3 priority=-1
OK
```

A continuación, lanzamos el comando para liberar de conexiones el server3:

```
$ maxctrl set server server3 drain
OK
```

Y comprobamos como efectivamente va liberando conexiones. El estado del servidor cambia a "*Draining*":

```
Every 2.0s: maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-152465 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-152465 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 15 | Draining, Slave, Synced, Running | 0-1-152465 | Galera-Monitor |

Lo dejamos hasta que alcance las 0 conexiones, en este momento el servidor ya ha procesado todas las transacciones que tenía pendientes y ya no aceptará más conexiones. El estado ha cambiado a "*Drained*":

```
Every 2.0s: maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-153034 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-153034 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 0 | Drained, Slave, Synced, Running | 0-1-153034 | Galera-Monitor |

Nos aseguramos de que no vaya a recibir más conexiones durante el proceso de actualización activando el modo mantenimiento:

```
$ maxctrl set server server3 maintenance
OK
```

El estado cambia a “Maintenance” y continua con cero conexiones:

```
Every 2.0s: maxctrl list servers                maxscale-lb-01.cluster.lab: Sun May 19 12:35:29 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-153416 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-153416 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 0 | Maintenance, Running | 0-1-153416 | Galera-Monitor |

En paralelo, podemos ir viendo en el fichero de log como las configuraciones que hemos estado aplicando en el nodo master de MaxScale se han estado sincronizando con el nodo standby. Este nodo se mantiene a la espera y no tiene conexiones:

```
==> /var/log/maxscale/maxscale.log <==

2024-05-19 12:27:35.216   notice : (ConfigManager); (sync): Updating to configuration version
301
2024-05-19 12:28:40.856   notice : (ConfigManager); (sync): Updating to configuration version
302
2024-05-19 12:28:41.393   notice : (apply_status_requests): Server 'server3' is being drained.
2024-05-19 12:35:21.599   notice : (ConfigManager); (sync): Updating to configuration version
303
```

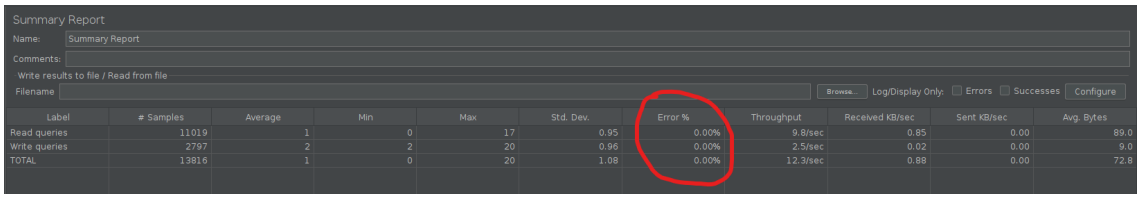
Estado de los servidores Galera en nodo standby (maxscale-lb-02):

```
Every 2.0s: maxctrl list servers                maxscale-lb-02.cluster.lab: Sun May 19 12:41:09 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 0 | Master, Synced, Running | 0-1-154268 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 0 | Slave, Synced, Running | 0-1-154268 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 0 | Maintenance, Running | 0-1-154268 | Galera-Monitor |

También podemos ir observando en tiempo real, en el “Summary Report” del plan de pruebas que hemos lanzado con JMeter, que las operaciones de los

clientes conectados se continúan ejecutando correctamente y no tenemos ningún error:




Por último, vamos a activar el nodo máster en MaxScale para que pueda recibir

Una vez tenemos el server3 de Galera en modo mantenimiento y totalmente liberado de conexiones, podemos parar el servicio de MariaDB en el backend:

```
$ sudo systemctl stop mariadb.service
```

Y comprobamos en MaxScale como el estado del backend ha cambiado a “Down”:


Lanzamos ahora la instalación de la versión 11.0:

```
$ sudo apt install mariadb-server

…
Configuration file '/etc/mysql/mariadb.conf.d/50-server.cnf'
 ==> Modified (by you or by a script) since installation.
 ==> Package distributor has shipped an updated version.
   What would you like to do about it ?  Your options are:
    Y or I  : install the package maintainer's version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
*** 50-server.cnf (Y/I/N/O/D/Z) [default=N] ? N
Setting up mariadb-server-compat (1:11.0.6+maria~deb12) ...
Processing triggers for man-db (2.11.2-2) ...
Processing triggers for libc-bin (2.36-9+deb12u7) ...
```

Durante la instalación, si no hay ningún problema, hay poca información que destacar, salvo que nos indicará que ya existe un fichero de configuración modificado por nosotros y respondemos ‘N’ para que mantenga la versión actualmente instalada.

Tras la instalación, el servicio se arrancará automáticamente y se deberá sincronizar de nuevo con el clúster, lo comprobamos en el log de MariaDB y analizamos la información más relevante:

```
2024-05-19 12:53:34 0 [Note] Starting MariaDB 11.0.6-MariaDB-deb12-log source revision 466ae1cf81f54b729058357bb19c4cf3982e1367 as process 18887
…
2024-05-19 12:53:34 0 [Note] WSREP: gcomm: connecting to group 'MariaDB Galera Cluster', peer '10.10.10.11:,10.10.10.12:,10.10.10.13:'
…
2024-05-19 12:53:35 0 [Note] WSREP: gcomm: connected
…
2024-05-19 12:53:35 0 [Note] WSREP: STATE EXCHANGE: Waiting for state UUID.
2024-05-19 12:53:35 0 [Note] WSREP: STATE EXCHANGE: sent state msg: 0a67b789-15ce-11ef-af04-f6dd310c9021
2024-05-19 12:53:35 0 [Note] WSREP: STATE EXCHANGE: got state msg: 0a67b789-15ce-11ef-af04-f6dd310c9021 from 0 (galera-mariadb-02)
2024-05-19 12:53:35 0 [Note] WSREP: STATE EXCHANGE: got state msg: 0a67b789-15ce-11ef-af04-f6dd310c9021 from 2 (galera-mariadb-01)
2024-05-19 12:53:35 1 [Note] WSREP: Starting rollbacker thread 1
2024-05-19 12:53:35 2 [Note] WSREP: Starting applier thread 2
2024-05-19 12:53:35 0 [Note] WSREP: STATE EXCHANGE: got state msg: 0a67b789-15ce-11ef-af04-f6dd310c9021 from 1 (galera-mariadb-03)
…
2024-05-19 12:53:35 2 [Note] WSREP: Server galera-mariadb-03 connected to cluster at position fb6d29ed-ee13-11ee-b506-eb954fb352d1:174690 with ID 0a194c00-15ce-11ef-8985-bad7b22e4304
…
2024-05-19 12:53:35 2 [Note] WSREP: State transfer required:
        Group state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:174690
        Local state: fb6d29ed-ee13-11ee-b506-eb954fb352d1:174227
2024-05-19 12:53:35 2 [Note] WSREP: Server status change connected -> joiner
2024-05-19 12:53:35 0 [Note] WSREP: Joiner monitor thread started to monitor
2024-05-19 12:53:35 0 [Note] WSREP: Running: 'wsrep_sst_mariabackup --role 'joiner' --address '10.10.10.13' --datadir '/var/lib/mysql/' --parent 18887 --progress 0 --binlog '/var/log/mysql/binlogs/mariadb-bin' --mysqld-args --wsrep_start_position=fb6d29ed-ee13-11ee-b506-eb954fb352d1:174227'
WSREP_SST: [INFO] mariabackup SST started on joiner (20240519 12:53:35.915)
WSREP_SST: [INFO] SSL configuration: CA='/etc/mysql/mariadb.conf.d/certificates/ca-cert.pem', CAPATH='', CERT='/etc/mysql/mariadb.conf.d/certificates/mariadb-cert.pem', KEY='/etc/mysql/mariadb.conf.d/certificates/mariadb-key.pem', MODE='DISABLED', encrypt='4' (20240519 12:53:36.050)
WSREP_SST: [INFO] Logging all stderr of SST/mariadb-backup to syslog (20240519 12:53:36.401)
2024-05-19 12:53:36 2 [Note] WSREP: ####### IST uuid:fb6d29ed-ee13-11ee-b506-eb954fb352d1 f: 174228, l: 174690, STRv: 3
2024-05-19 12:53:36 2 [Note] WSREP: IST receiver addr using tcp://10.10.10.13:4568
2024-05-19 12:53:36 2 [Note] WSREP: Prepared IST receiver for 174228-174690, listening at: tcp://10.10.10.13:4568
2024-05-19 12:53:36 0 [Note] WSREP: Member 1.0 (galera-mariadb-03) requested state transfer from '10.10.10.11,10.10.10.12'. Selected 2.0 (galera-mariadb-01)(SYNCED) as donor.
2024-05-19 12:53:36 0 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 174693)
2024-05-19 12:53:36 2 [Note] WSREP: Requesting state transfer: success, donor: 2
2024-05-19 12:53:37 0 [Note] WSREP: 2.0 (galera-mariadb-01): State transfer to 1.0 (galera-mariadb-03) complete.
2024-05-19 12:53:37 0 [Note] WSREP: Member 2.0 (galera-mariadb-01) synced with group.
…
2024-05-19 12:53:38 0 [Note] WSREP: ####### IST applying starts with 174228
```

```
2024-05-19 12:53:38 0 [Note] WSREP: ####### IST current seqno initialized to 174228
2024-05-19 12:53:38 0 [Note] WSREP: Receiving IST...  0.0% (  0/463 events) complete.
…
2024-05-19 12:53:38 0 [Note] WSREP: Receiving IST...100.0% (463/463 events) complete.
2024-05-19 12:53:38 2 [Note] WSREP:
=================================================
View:
  id: fb6d29ed-ee13-11ee-b506-eb954fb352d1:174690
  status: primary
  protocol_version: 4
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY,
ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED, PREORDERED,
STREAMING, NBO
  final: no
  own_index: 1
  members(3):
          0: 04394ca7-1536-11ef-bd62-de8153917b43, galera-mariadb-02
          1: 0a194c00-15ce-11ef-8985-bad7b22e4304, galera-mariadb-03
          2: cf2887fc-1535-11ef-8e2a-aecfb685d42a, galera-mariadb-01
=================================================
…
2024-05-19 12:53:38 2 [Note] WSREP: Server galera-mariadb-03 synced with group
2024-05-19 12:53:38 2 [Note] WSREP: Server status change joined -> synced
2024-05-19 12:53:38 2 [Note] WSREP: Synchronized with group, ready for connections
2024-05-19 12:53:38 2 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
```

El nodo arranca mostrando la nueva versión, encuentra el cluster 'MariaDB Galera Cluster' y se conecta; reclama estado y comienza la transferencia IST desde el donante galera-mariadb-01, finaliza la transferencia, se sincroniza con el grupo e indica que está preparado para recibir conexiones.

Comprobamos también en MaxScale el estado de los servidores tras la instalación. Vemos como de nuevo server3 aparece en estado "Running" y continua en mantenimiento con cero conexiones:

```
Every 2.0s: maxctrl list servers                maxscale-lb-01.cluster.lab: Sun May 19 13:07:51 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-158269 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-158269 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 0 | Maintenance, Running | 0-1-158269 | Galera-Monitor |

En el log de MaxScale observamos como server3 ya ha informado de su nueva versión de MariaDB:

```
==> /var/log/maxscale/maxscale.log <==
2024-05-19 12:53:39.000   notice : (set_version): server3 sent version string '11.0.6-
MariaDB-deb12-log'. Detected type: MariaDB, version: 11.0.6.
```

En este momento, podemos desactivar el modo mantenimiento

```
$ maxctrl clear server server3 maintenance
OK
```

```
Every 2.0s: maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-161138 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-161138 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 0 | Drained, Slave, Synced, Running | 0-1-161138 | Galera-Monitor |

Comprobamos como el nodo está sincronizado con el clúster, en *standby* y ejecución, pero permanece en estado “Drained”, así que lo liberamos para que vuelva a aceptar conexiones de los clientes, pasando a su estado original:

```
$ maxctrl clear server server3 drain
OK
```

```
Every 2.0s: maxctrl list servers                maxscale-lb-01.cluster.lab: Sun May 19 13:31:12 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-161764 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-161764 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 20 | Slave, Synced, Running | 0-1-161764 | Galera-Monitor |

Tras comprobar que todo el proceso de actualización sobre este nodo ha concluido correctamente y ya no va a ser necesario reiniciarlo más, vamos a forzar la conmutación a nuevo máster el server3 de forma controlada(*switchover*) y así, poder repetir el mismo procedimiento de actualización en los otros dos servidores. **Indicar que este es el procedimiento más delicado de todo el laboratorio, puesto que al conmutar el nodo máster podría haber algún error en los clientes durante el cierre de conexiones en el antiguo máster y transición al nuevo máster, si algo no estuviera bien configurado o coincidiera el cierre de conexiones con alguna operación de escritura en el *backend*.**

Empezamos por asignarle la prioridad “1”, que será compartida con el server1, actual máster, por lo que no habrá conmutación de momento:

```
$ MAXCTRL_WARNINGS=0 maxctrl alter server server3 priority=1
OK
```

A continuación, asignamos prioridad “-1” a los server1 y server2 para indicar que ya no son candidatos a máster, por lo que en este momento sí se activará el procedimiento para conmutar el máster al server3:

```
$ MAXCTRL_WARNINGS=0 maxctrl alter server server2 priority=-1
OK
$ MAXCTRL_WARNINGS=0 maxctrl alter server server1 priority=-1
OK
```

Comprobamos como la conmutación de máster se ha realizado correctamente al server3:

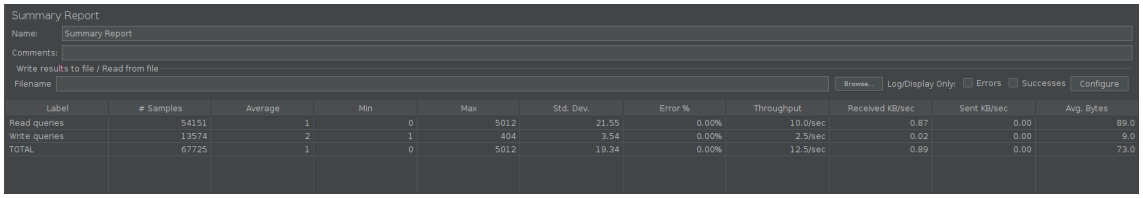
```
Every 2.0s: maxctrl list servers                maxscale-lb-01.cluster.lab: Sun May 19 13:54:46 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 21 | Slave, Synced, Running | 0-3-165307 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-3-165307 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 25 | Master, Synced, Running | 0-3-165307 | Galera-Monitor |

En el log de MaxScale también comprobamos la transición del server3 a nuevo máster:

```
==> /var/log/maxscale/maxscale.log <==
2024-05-19 13:53:54.074   notice : (log_state_change): Server changed state:
server1[10.10.10.11:3306]: new_slave. [Master, Synced, Running] -> [Slave, Synced, Running]
2024-05-19 13:53:54.074   notice : (log_state_change): Server changed state:
server3[10.10.10.13:3306]: new_master. [Slave, Synced, Running] -> [Master, Synced,
Running]
```

Y en JMeter continuamos sin errores desde los clientes, por lo que confirmamos que el paso de actualización de versión de MariaDB en el primero de los servidores y posterior conmutación controlada del nodo máster ha finalizado correctamente:




Una vez concluida la actualización en el primero de los nodos, ya podemos actuar de la misma forma en los otros dos, ejecutando los siguientes pasos, tal y como hemos hecho en server3 y sin riesgo de conmutación de máster. Ejecutaremos los pasos siguientes, primero en un nodo y luego en el otro para no saturar de conexiones el nodo máster y disponiendo en todo momento de un nodo standby:

1. [maxscale]: $ **maxctrl set server server1 drain** → esperamos que se liberen las conexiones.
2. [maxscale]: $ **maxctrl set server server1 maintenance**
3. [server]: $ **sudo systemctl stop mariadb.service**
4. [server]: $ **sudo apt install mariadb-server** → comprobamos que el servicio arranca correctamente, muestra la nueva versión y se sincroniza con el resto de nodos.
5. [maxscale] $ **maxctrl clear server server1 maintenance**
6. [maxscale] $ **maxctrl clear server server1 drain** → comprobamos que se crean nuevas conexiones desde los clientes.

Finalizado el proceso en los otros dos nodos, habremos conseguido actualizar de versión el clúster de Galera sin parada del servicio. Comprobamos el estado final de los servidores en MaxScale:

```
Every 2.0s: maxctrl list servers                    maxscale-lb-01.cluster.lab: Sun May 19 14:42:05 2024
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Slave, Synced, Running | 0-3-172408 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-3-172408 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 25 | Master, Synced, Running | 0-3-172408 | Galera-Monitor |

Vemos como todos los servidores están recibiendo conexiones y en su estado normal, sin embargo, server3 es ahora el nodo máster, debido a la prioridad 1 que le asignamos anteriormente. Podemos, en este momento, dejarlo tal y como está, pero decidimos realizar el proceso de *failback* para devolver el nodo maestro al server1 de forma controlada, tal y como estaba al comienzo del laboratorio.

Volvemos a asignar las prioridades originales a los servidores y, al asignar la prioridad 1 a server1, se deberá lanzar automáticamente la conmutación de nodo máster a server1.

```
$ MAXCTRL_WARNINGS=0 maxctrl alter server server2 priority=2
OK
$ MAXCTRL_WARNINGS=0 maxctrl alter server server1 priority=1
OK
$ MAXCTRL_WARNINGS=0 maxctrl alter server server3 priority=3
OK
```

En el log de MaxScale observamos como efectivamente ha conmutado el nodo al asignar la nueva prioridad:

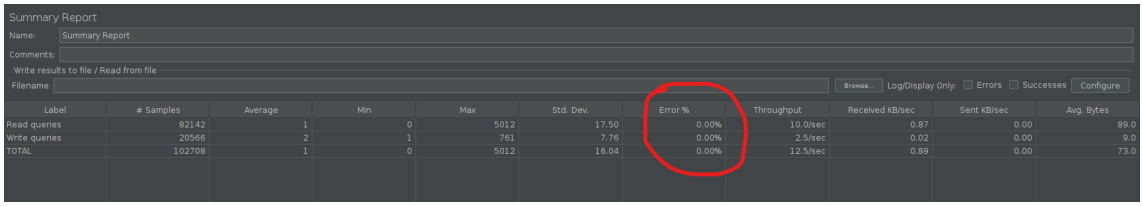
```
==> /var/log/maxscale/maxscale.log <==
2024-05-19 14:45:05.961   notice : (log_state_change): Server changed state:
server1[10.10.10.11:3306]: new_master. [Slave, Synced, Running] -> [Master, Synced,
Running]
2024-05-19 14:47:18.961   notice : (log_state_change): Server changed state:
server3[10.10.10.13:3306]: new_slave. [Master, Synced, Running] -> [Slave, Synced,
Running]
```

Quedando finalmente el estado de los servidores en MaxScale como al comienzo del laboratorio:

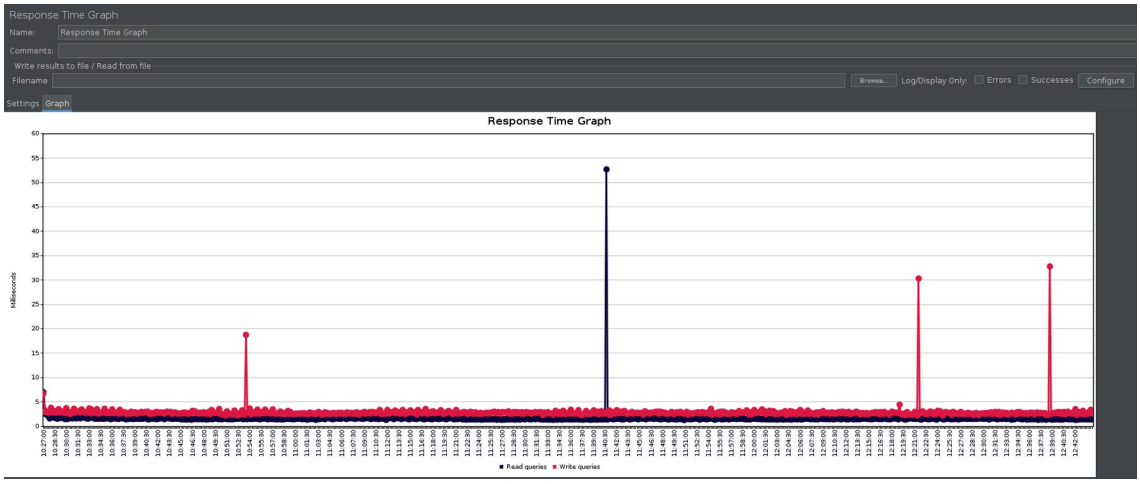
| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 25 | Master, Synced, Running | 0-1-175635 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 25 | Slave, Synced, Running | 0-1-175635 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 25 | Slave, Synced, Running | 0-1-175635 | Galera-Monitor |

Paramos en este momento la simulación del plan de pruebas en JMeter para comprobar el resultado final de la experiencia del cliente y si el proceso ha sido totalmente transparente.

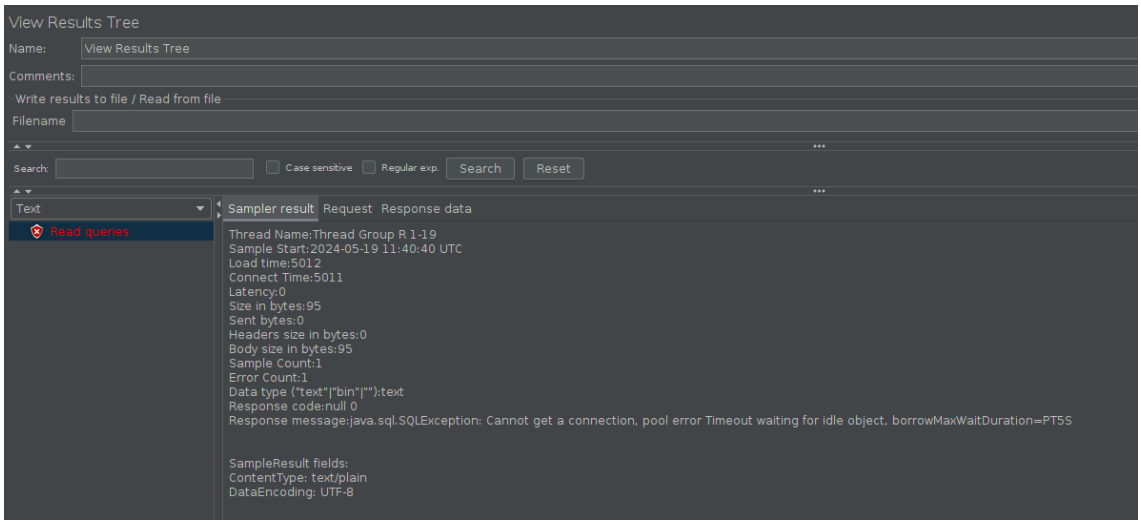
El informe del *Summary Report* nos devuelve un porcentaje de error del 0,00% sobre el total de peticiones de lectura y escrituras ejecutadas en este periodo, que podemos ver en la columna *Samples*:




Sin embargo, si observamos la gráfica del *Response Time Graph*, vemos como durante todo el procedimiento los tiempos de respuesta se han mantenido estables por debajo de los 5ms, salvo algunos picos en la respuesta a alguna operación de escritura(en rojo) y otro pico sobre las 11:40h en una operación de lectura(en azul) con un tiempo de respuesta especialmente alto, por encima de los 50ms. Esa hora coincide más o menos con el de la primera conmutación de nodo máster, pero en los servidores no se ha registrado ningún error.




Observamos ahora el informe de *View Results Tree*, donde teníamos filtrado por errores y vemos que tan sólo se ha producido un error en una operación de lectura de las 82.142 ejecutadas, que coincide en hora con la del pico en el tiempo de respuesta de la gráfica anterior.




El error devuelto al cliente es que el pool de conexiones no ha podido obtener ninguna conexión en 5 segundos (*timeout*). Probablemente este error se haya producido durante la conmutación de máster en MaxScale, no obstante, es un error insignificante, al tratarse de una única operación de lectura de entre todas las ejecutadas y que, probablemente, se podría haber evitado ampliando el valor del *timeout* en JMeter.

En el informe del *Generate Summary Results* observamos también el único error en esta operación durante todo el tiempo de ejecución del laboratorio, que como ya hemos comentado, lo consideramos insignificante.

```
# Generate Summary Results #
…
2024-05-19 11:40:30,038 INFO o.a.j.r.Summariser: Generate Summary Results +    375 in
00:00:30 =   12.5/s Avg:     1 Min:     1 Max:     7 Err:     0 (0.00%) Active: 25 Started: 25
Finished: 0
2024-05-19 11:40:30,038 INFO o.a.j.r.Summariser: Generate Summary Results =  54836 in
01:13:29 =   12.4/s Avg:     1 Min:     0 Max:   404 Err:     0 (0.00%)
2024-05-19 11:41:00,065 INFO o.a.j.r.Summariser: Generate Summary Results +    373 in
00:00:30 =   12.4/s Avg:    15 Min:     1 Max:  5012 Err:     1 (0.27%) Active: 25 Started: 25
Finished: 0
2024-05-19 11:41:00,066 INFO o.a.j.r.Summariser: Generate Summary Results =  55209 in
01:13:59 =   12.4/s Avg:     1 Min:     0 Max:  5012 Err:     1 (0.00%)
2024-05-19 11:41:30,027 INFO o.a.j.r.Summariser: Generate Summary Results +    374 in
00:00:30 =   12.5/s Avg:     1 Min:     1 Max:     6 Err:     0 (0.00%) Active: 25 Started: 25
Finished: 0
2024-05-19 11:41:30,028 INFO o.a.j.r.Summariser: Generate Summary Results =  55583 in
01:14:29 =   12.4/s Avg:     1 Min:     0 Max:  5012 Err:     1 (0.00%)
…
```

Concluimos, por tanto, que el resultado del laboratorio y objetivo principal del proyecto ha sido satisfactorio y se ha conseguido subir de versión *major* un clúster de Galera con 3 nodos MariaDB, sin interrupción del servicio y con zero-downtime.

### 6.2. Point-in-Time Recovery (PITR)

El siguiente laboratorio consistirá en la recuperación de una base de datos MariaDB en entorno clusterizado con Galera y tres nodos en cualquier punto de restauración en el tiempo (Point-in-Time Recovery o PITR). Para lograrlo nos apoyaremos sobre copias de seguridad completas con la herramienta Mariabackup, copias incrementales y los registros binarios (binlogs) hasta el punto en el que se necesite restaurar la base de datos.

Esta práctica es muy útil cuando, tal y como vimos en el punto 5.1, disponemos de copias de seguridad completas semanales, incrementales diarias y, debido a un problema con los datos, necesitamos recuperar un *backup*, así que tenemos que recurrir al último incremental realizado previo al problema, pero no queremos perder todo el trabajo posterior realizado durante el día siguiente. Para este tipo de situaciones es fundamental conocer la fecha y hora en la que se produjo el problema. Se restaurará el último *backup* full realizado, seguido de todos los incrementales que se dispongan previos al problema y, a continuación, los registros binarios hasta la fecha y hora que se desee.

Para poder comprobar el éxito del laboratorio, se ha modificado la tabla de la base de datos que se creó para el laboratorio anterior, añadiendo una columna nueva de tipo *timestamp*. Esta columna se rellenará automáticamente con el valor de la fecha y hora actual cada vez que se inserte un registro en la base de datos.

```
MariaDB [test]> ALTER TABLE test_table ADD COLUMN timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP;
Query OK, 0 rows affected (0.023 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [test]> SELECT * FROM test_table ORDER BY timestamp DESC LIMIT 10;
+---------+---------------------------------------+---------------------+
| id      | msg                                   | timestamp           |
+---------+---------------------------------------+---------------------+
| 105539 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:58 |
| 105530 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:57 |
| 105533 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:57 |
| 105536 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:57 |
| 105524 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:56 |
| 105527 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:56 |
| 105515 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:55 |
| 105518 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:55 |
| 105521 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:55 |
| 105509 | galera-mariadb-01.cluster.lab | 2024-05-21 21:07:54 |
+---------+---------------------------------------+---------------------+
10 rows in set (0.011 sec)
```

```
MariaDB [test]> desc test.test_table;
+-----------+-----------+--------+------+---------------------+----------------+
| Field     | Type      | Null   | Key | Default             | Extra          |
+-----------+-----------+--------+------+---------------------+----------------+
| id        | int(11)   | NO     | PRI | NULL                | auto_increment |
| msg       | text      | YES    |     | NULL                |                |
| timestamp | timestamp | YES    |     | current_timestamp() |                |
+-----------+-----------+--------+------+---------------------+----------------+
3 rows in set (0.002 sec)
```

Antes de comenzar con el laboratorio, volveremos a hacer instantáneas en GCP de los discos de las instancias implicadas, que nos servirán como respaldo para poder volver al estado previo en caso de necesidad.

```
$ for i in galera-mariadb-01 galera-mariadb-02 galera-mariadb-03
> do
> gcloud compute \
> --project "quixotic-skill-418320" \
> disks snapshot $i \
> --snapshot-names snap-$i-lab2-$(date +\%F) \
> --zone "europe-west1-b" \
> --description "Snapshot de $i previo a LAB2."
> done
Creating snapshot(s) snap-galera-mariadb-01-lab2-2024-05-21...done.
Creating snapshot(s) snap-galera-mariadb-02-lab2-2024-05-21...done.
Creating snapshot(s) snap-galera-mariadb-03-lab2-2024-05-21...done.
```

Antes de parar el servicio, consultamos los últimos registros que tenemos en la base de datos con su *timestamp*.

```
MariaDB [test]> SELECT * FROM test_table ORDER BY timestamp DESC
LIMIT 10;
+---------+---------------------------------------+---------------------+
| id      | msg                                   | timestamp           |
+---------+---------------------------------------+---------------------+
| 120911 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:11 |
| 120914 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:11 |
| 120905 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:10 |
| 120908 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:10 |
| 120896 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:09 |
| 120899 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:09 |
| 120902 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:09 |
| 120890 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:08 |
| 120893 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:08 |
| 120881 | galera-mariadb-01.cluster.lab | 2024-05-21 21:42:07 |
+---------+---------------------------------------+---------------------+
10 rows in set (0.012 sec)
```

Queremos restaurar el estado de la base de datos a fecha "**2024-05-21 20:00:00**", así que consultamos, en primer lugar, las copias que disponemos:

```
$ sudo ls -ltr /mnt/backups/galera/full/ /mnt/backups/galera/incremental/
/mnt/backups/galera/full/:
total 8
drwx------ 6 nobody nogroup 4096 May 15 22:09 2024-05-15
```

```
drwx------ 7 nobody nogroup 4096 May 20 02:00 2024-05-20

/mnt/backups/galera/incremental/:
total 20
drwx------ 6 nobody nogroup 4096 May 16 18:30 2024-05-16
drwx------ 7 nobody nogroup 4096 May 17 02:00 2024-05-17
drwx------ 7 nobody nogroup 4096 May 18 02:00 2024-05-18
drwx------ 7 nobody nogroup 4096 May 19 02:00 2024-05-19
drwx------ 7 nobody nogroup 4096 May 21 02:00 2024-05-21
```

Tal y como vemos arriba, las copias más próximas y, anteriores a la fecha de restauración, son "**2024-05-20**" para el *full* y "**2024-05-21**" para la incremental, así que estas serán las candidatas para restaurar. Si dispusiéramos de más incrementales entre el *full* y la fecha de restauración habría que restaurarlas también en orden secuencial.

Anotamos también el **fichero de binlog y la posición actual**:

```
MariaDB [test]> show master status;
+--------------------------+------------+------------------+------------------------+
| File                     | Position   | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------------------+------------+------------------+------------------------+
| mariadb-bin.000045 |  6904328 |                  |                        |
+--------------------------+------------+------------------+------------------------+
1 row in set (0.001 sec)
```

Y, por otro lado, extraemos la misma información en la copia incremental para obtener el **punto de partida de los ficheros tipo binlog** que se van a restaurar:

```
$ sudo cat
/mnt/backups/galera/incremental/2024-05-21/xtrabackup_binlog_info
mariadb-bin.000045    2786709         0-1-177920
```

En nuestro caso, el fichero binlog de partida coincide con el actual, luego sabemos que toda la información de los binlog que queremos restaurar se encuentra en este fichero. En caso de disponer de más de uno, habrá que restaurar todos los intermedios, desde la fecha de inicio a la de fin.

Ya disponemos de toda la información que necesitamos, así que resumimos y a continuación podremos parar el servicio y comenzar con las restauraciones:

| Fecha restauración: **2024-05-21 20:00:00** | | |
|---|---|---|
| Posición inicial: **2786709** | | |
| Posición final: **6904328** | | |
| Tipo copia | Nombre (orden secuencial) | Fecha |
| FULL | 2024-05-20 | 2024-05-20 |
| INCREMENTALES | 2024-05-21 | 2024-05-21 |
| BINLOGS | mariadb-bin.000045 | 2024-05-21 |

**Tabla 6 - Resumen restauración PITR**

Comenzamos parando el servicio de MariaDB en los tres nodos de Galera, anotando el último en parar, que será el nodo en que restauraremos e iniciaremos el clúster:

```
$ sudo systemctl stop mariadb.service
```

A continuación, preparamos el *full base backup*. Este comando es necesario para sincronizar la copia de seguridad *full* con los cambios contenidos en los redo log de InnoDB que se recopilaron mientras se hizo la copia de seguridad.

```
$ sudo /usr/bin/mariadb-backup --prepare
--target-dir=/mnt/backups/galera/full/2024-05-20
/usr/bin/mariadb-backup based on MariaDB server 11.0.6-MariaDB debian-linux-gnu (x86_64)
[00] 2024-05-21 22:57:49 cd to /mnt/backups/galera/full/2024-05-20/
[00] 2024-05-21 22:57:49 open files limit requested 0, set to 1024
[00] 2024-05-21 22:57:49 This target seems to be not prepared yet.
[00] 2024-05-21 22:57:49 mariabackup: using the following InnoDB configuration for recovery:
[00] 2024-05-21 22:57:49 innodb_data_home_dir = .
[00] 2024-05-21 22:57:49 innodb_data_file_path = ibdata1:12M:autoextend
[00] 2024-05-21 22:57:49 innodb_log_group_home_dir = .
[00] 2024-05-21 22:57:49 InnoDB: Using liburing
[00] 2024-05-21 22:57:49 Starting InnoDB instance for recovery.
[00] 2024-05-21 22:57:49 mariabackup: Using 104857600 bytes for buffer pool (set by --use-
memory parameter)
2024-05-21 22:57:49 0 [Note] InnoDB: Compressed tables use zlib 1.2.13
2024-05-21 22:57:49 0 [Note] InnoDB: Using transactional memory
2024-05-21 22:57:49 0 [Note] InnoDB: Number of transaction pools: 1
2024-05-21 22:57:49 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-05-21 22:57:49 0 [Note] InnoDB: Using liburing
2024-05-21 22:57:49 0 [Note] InnoDB: Initializing buffer pool, total size = 100.000MiB, chunk
size = 100.000MiB
2024-05-21 22:57:49 0 [Note] InnoDB: Completed initialization of buffer pool
2024-05-21 22:57:49 0 [Note] InnoDB: Buffered log writes (block size=512 bytes)
2024-05-21 22:57:49 0 [Note] InnoDB: Starting crash recovery from checkpoint
LSN=791585867
2024-05-21 22:57:49 0 [Note] InnoDB: End of log at LSN=795012871
2024-05-21 22:57:49 0 [Note] InnoDB: To recover: 294 pages
2024-05-21 22:57:49 0 [Note] InnoDB: Last binlog file '/var/log/mysql/binlogs/mariadb-
bin.000045', position 2183035
[00] 2024-05-21 22:57:49 Last binlog file /var/log/mysql/binlogs/mariadb-bin.000045, position
2183035
[00] 2024-05-21 22:57:49 mariabackup: Recovered WSREP position: fb6d29ed-ee13-11ee-
b506-eb954fb352d1:194279 domain_id: 0
```

Luego aplicamos los incrementales al *full base backup* para sincronizar la copia de seguridad *full* con los cambios contenidos en el incremental. En nuestro caso, solo lo realizaremos con uno puesto que sólo tenemos que restaurar un incremental:

```
$ sudo /usr/bin/mariadb-backup --prepare
--target-dir=/mnt/backups/galera/full/2024-05-20
--incremental-dir=/mnt/backups/galera/incremental/2024-05-21
/usr/bin/mariadb-backup based on MariaDB server 11.0.6-MariaDB debian-linux-gnu (x86_64)
[00] 2024-05-21 23:13:38 incremental backup from 791585867 is enabled.
[00] 2024-05-21 23:13:38 cd to /mnt/backups/galera/full/2024-05-20/
[00] 2024-05-21 23:13:38 open files limit requested 0, set to 1024
```

```
[00] 2024-05-21 23:13:38 This target seems to be already prepared.
[00] 2024-05-21 23:13:38 mariabackup: using the following InnoDB configuration for recovery:
[00] 2024-05-21 23:13:38 innodb_data_home_dir = .
[00] 2024-05-21 23:13:38 innodb_data_file_path = ibdata1:12M:autoextend
[00] 2024-05-21 23:13:38 innodb_log_group_home_dir =
/mnt/backups/galera/incremental/2024-05-21/
[00] 2024-05-21 23:13:38 InnoDB: Using liburing
[00] 2024-05-21 23:13:38 mariabackup: Generating a list of tablespaces
[00] 2024-05-21 23:13:38 page size for
/mnt/backups/galera/incremental/2024-05-21//undo002.delta is 16384 bytes
[00] 2024-05-21 23:13:38 Applying
/mnt/backups/galera/incremental/2024-05-21//undo002.delta to ./undo002...
[00] 2024-05-21 23:13:38 page size for
/mnt/backups/galera/incremental/2024-05-21//undo003.delta is 16384 bytes
…
[00] 2024-05-21 23:13:38 mariabackup: using the following InnoDB configuration for recovery:
[00] 2024-05-21 23:13:38 innodb_data_home_dir = .
[00] 2024-05-21 23:13:38 innodb_data_file_path = ibdata1:12M:autoextend
[00] 2024-05-21 23:13:38 innodb_log_group_home_dir =
/mnt/backups/galera/incremental/2024-05-21/
[00] 2024-05-21 23:13:38 InnoDB: Using liburing
[00] 2024-05-21 23:13:38 Starting InnoDB instance for recovery.
[00] 2024-05-21 23:13:38 mariabackup: Using 104857600 bytes for buffer pool (set by --use-
memory parameter)
2024-05-21 23:13:38 0 [Note] InnoDB: Compressed tables use zlib 1.2.13
2024-05-21 23:13:38 0 [Note] InnoDB: Using transactional memory
2024-05-21 23:13:38 0 [Note] InnoDB: Number of transaction pools: 1
2024-05-21 23:13:38 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2024-05-21 23:13:38 0 [Note] InnoDB: Using liburing
2024-05-21 23:13:38 0 [Note] InnoDB: Initializing buffer pool, total size = 100.000MiB, chunk
size = 100.000MiB
2024-05-21 23:13:38 0 [Note] InnoDB: Completed initialization of buffer pool
2024-05-21 23:13:38 0 [Note] InnoDB: Buffered log writes (block size=512 bytes)
2024-05-21 23:13:38 0 [Note] InnoDB: Starting crash recovery from checkpoint
LSN=791585867
2024-05-21 23:13:38 0 [Note] InnoDB: End of log at LSN=796068806
2024-05-21 23:13:38 0 [Note] InnoDB: To recover: 305 pages
2024-05-21 23:13:38 0 [Note] InnoDB: Last binlog file '/var/log/mysql/binlogs/mariadb-
bin.000045', position 2786709
[00] 2024-05-21 23:13:38 Last binlog file /var/log/mysql/binlogs/mariadb-bin.000045,
position 2786709
[00] 2024-05-21 23:13:38 mariabackup: Recovered WSREP position: fb6d29ed-ee13-11ee-
b506-eb954fb352d1:196478 domain_id: 0

[01] 2024-05-21 23:13:39 Copying
/mnt/backups/galera/incremental/2024-05-21/test/test_table.frm to ./test/test_table.frm
[01] 2024-05-21 23:13:39         ...done
…
[00] 2024-05-21 23:13:41 completed OK!
```

Este el punto límite que tenemos para mantener la base de datos arrancada. En nuestro caso paramos el servicio antes de la preparación, pero se podría haber mantenido disponible hasta este momento. En caso de haber optado por mantenerla arrancada hasta ahora, el fichero binlog y posición actual que tendríamos que anotar sería el que aparece más arriba, en verde, en la salida por pantalla del comando de preparación del último incremental.

Borramos el contenido del datadir, en nuestro caso en la ruta “/var/lib/mysql”:

```
$ sudo rm -rf /var/lib/mysql/
```

Y restauramos desde el backup *full*:

```
$ sudo /usr/bin/mariadb-backup --copy-back
--target-dir=/mnt/backups/galera/full/2024-05-20
/usr/bin/mariadb-backup based on MariaDB server 11.0.6-MariaDB debian-linux-gnu (x86_64)
[01] 2024-05-21 23:50:28 Copying ./aria_log.00000001 to /var/lib/mysql/aria_log.00000001
[01] 2024-05-21 23:50:29         ...done
[01] 2024-05-21 23:50:29 Copying ./aria_log_control to /var/lib/mysql/aria_log_control
[01] 2024-05-21 23:50:29         ...done
…
[00] 2024-05-21 23:50:30 completed OK!
```

Restauramos permisos y comprobamos el contenido del datadir para comprobar que la restauración es correcta:

```
$ sudo chown -R mysql:mysql /var/lib/mysql
$ sudo ls -l /var/lib/mysql/
total 43496
  424 -rw-r----- 1 mysql mysql   434176 May 21 23:50 aria_log.00000001
    4 -rw-r----- 1 mysql mysql       52 May 21 23:50 aria_log_control
   16 -rw-r----- 1 mysql mysql    12304 May 21 23:50 ib_logfile0
12288 -rw-r----- 1 mysql mysql 12582912 May 21 23:50 ibdata1
    4 drwx------ 2 mysql mysql     4096 May 21 23:50 mysql
    4 drwx------ 2 mysql mysql     4096 May 21 23:50 performance_schema
   12 drwx------ 2 mysql mysql    12288 May 21 23:50 sys
    4 drwx------ 2 mysql mysql     4096 May 21 23:50 test
    4 drwx------ 2 mysql mysql     4096 May 21 23:50 testdb
10240 -rw-r----- 1 mysql mysql 10485760 May 21 23:50 undo001
10240 -rw-r----- 1 mysql mysql 10485760 May 21 23:50 undo002
10240 -rw-r----- 1 mysql mysql 10485760 May 21 23:50 undo003
    4 -rw-r----- 1 mysql mysql       42 May 21 23:50 xtrabackup_binlog_pos_innodb
    4 -rw-r----- 1 mysql mysql       45 May 21 23:50 xtrabackup_galera_info
    4 -rw-r----- 1 mysql mysql       45 May 21 23:50 xtrabackup_galera_info_SST
    4 -rw-r----- 1 mysql mysql      610 May 21 23:50 xtrabackup_info
```

Arrancamos el cluster en el nodo donde hemos restaurado

```
$ sudo galera_new_cluster
```

Y comprobamos los últimos registros en la base de datos, que deben corresponderse en este momento con los del último incremental:

```
MariaDB [test]> SELECT * FROM test_table ORDER BY timestamp DESC
LIMIT 10;
+---------+-------------------------------------+---------------------+
| id      | msg                                 | timestamp           |
+---------+-------------------------------------+---------------------+
| 76646 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:25 |
| 76649 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:25 |
| 76637 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:24 |
| 76640 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:24 |
| 76643 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:24 |
```

```
| 76631 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:23 |
| 76634 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:23 |
| 76622 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:22 |
| 76625 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:22 |
| 76628 | galera-mariadb-01.cluster.lab | 2024-05-21 01:43:22 |
+----------+---------------------------------------+---------------------------+
10 rows in set (0.115 sec)
```

Así que aplicamos los binlogs hasta la fecha en la que decidimos que queríamos restaurar “2024-05-21 20:00:00”:

```
$ sudo mysqlbinlog --start-position="2786709" --stop-datetime="2024-05-21 20:00:00" /var/log/mysql/binlogs/mariadb-bin.000045 | sudo mysql -u root -pmariadb
```

Y volvemos a comprobar los últimos registros:

```
MariaDB [test]> SELECT * FROM test_table ORDER BY timestamp DESC LIMIT 10;
+----------+---------------------------------------+---------------------------+
| id       | msg                                   | timestamp                 |
+----------+---------------------------------------+---------------------------+
| 81389 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:59 |
| 81392 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:59 |
| 81380 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:58 |
| 81383 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:58 |
| 81386 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:58 |
| 81374 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:57 |
| 81377 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:57 |
| 81365 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:56 |
| 81368 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:56 |
| 81371 | galera-mariadb-01.cluster.lab | 2024-05-21 19:59:56 |
+----------+---------------------------------------+---------------------------+
10 rows in set (0.008 sec)
```

Comprobando que, efectivamente, se ha restaurado la base de datos hasta la fecha y hora justa en la que queríamos restaurar previo al problema.

Nos queda levantar los otros nodos y comprobar la sincronización y el estado final del sistema:

```
$ sudo systemctl start mariadb.service
==> /var/log/maxscale/maxscale.log <==
2024-05-22  0:32:50 0 [Note] WSREP: 0.0 (galera-mariadb-02): State transfer from 1.0 (galera-mariadb-01) complete.
2024-05-22  0:32:50 2 [Note] WSREP: Server galera-mariadb-02 synced with group
2024-05-22  0:32:50 2 [Note] WSREP: Server status change joined -> synced
2024-05-22  0:32:50 2 [Note] WSREP: Synchronized with group, ready for connections
```

```
MariaDB [(none)]> show status like 'wsrep_cluster_size';
+--------------------------+--------+
| Variable_name      | Value |
+--------------------------+--------+
| wsrep_cluster_size | 3       |
+--------------------------+--------+
1 row in set (0.001 sec)
```

Y por último comprobamos en MaxScale como volvemos a tener el clúster de Galera con los tres nodos correctamente sincronizados, corriendo y recibiendo conexiones:

```
jlopez@maxscale-lb-01:~$ maxctrl list servers
```

| Server | Address | Port | Connections | State | GTID | Monitor |
|---|---|---|---|---|---|---|
| server1 | 10.10.10.11 | 3306 | 9 | Master, Synced, Running | 0-1-192683 | Galera-Monitor |
| server2 | 10.10.10.12 | 3306 | 9 | Slave, Synced, Running | 0-1-179508 | Galera-Monitor |
| server3 | 10.10.10.13 | 3306 | 9 | Slave, Synced, Running | 0-1-179508 | Galera-Monitor |

Con estas últimas comprobaciones damos por concluido el laboratorio, habiendo conseguido con éxito restaurar el sistema de Galera hasta un punto en el tiempo (PITR), contando con un sistema de copias de seguridad robusto y fiable, tal y como se definió en el punto 5.1 de la memoria. Por último, indicar la importancia de tener habilitado la generación de log binarios en formato ROW, de no ser así, no habría sido posible la restauración de la base de datos más allá de la última copia de seguridad *full* o incremental.

# 7. Conclusiones

## 7.1. Conclusiones

En líneas generales, todas las conclusiones que puedo sacar del trabajo realizado son realmente satisfactorias y gratificantes. En lo personal, es una temática que siempre me ha resultado especialmente interesante y tenía ganas de abordar y, gracias al TFG, me ha permitido profundizar bastante más en los conocimientos que ya tenía de dicha tecnología, así como aprender de otras de las que no tenía conocimiento alguno y que me han servido para ejecutar con éxito los laboratorios objetivos en esta memoria, como por ejemplo *Google Cloud Platform* y Apache Jmeter. En dichas tecnologías he obtenido unas bases, que antes no tenía, y me servirán para poder aplicarlas en mi entorno laboral, al igual que las técnicas aprendidas con Galera y MaxScale, de las que no era consciente en un principio de todo el potencial que tenían.

En cuanto a la planificación, se han podido finalizar todos los hitos en tiempo y forma, por lo que se concluye que en líneas generales ha sido la adecuada con la siguiente excepción. Tan sólo se ha dejado sin realizar uno de los objetivos secundarios, el referente al escalado horizontal, por motivos de tiempo, extensión de la memoria y fruto de una planificación original un poco optimista. La elección de la plataforma de virtualización en GCP y la autoformación no prevista, me llevó más tiempo del planificado en un principio y me obligó a tener que descartarla y posponerlo para un plan futuro en una versión extendida del proyecto. No obstante, se sustituyó por un punto que no tenía planificado en un principio, plan de contingencias, que vi necesario realizar para poder ejecutar el laboratorio secundario de Point-in-Time Recovery (PITR).

Por último, quiero comentar también que, por motivos de extensión, hay ciertos puntos, considerados menos prioritarios, que no se han podido abordar en el grueso de la memoria y se han documentado en los tres anexos finales.

## 7.2. Planes futuros

En cuanto a los planes futuros, tal y como ya he adelantado en el punto anterior, tengo una serie de ideas que sería interesante abordar más adelante en una continuación del trabajo:

- **Entorno de monitorización de la infraestructura**, para vigilar el buen estado de salud de todos los componentes, disparando alertas en caso de incidentes y permitiendo la ejecución de *event handlers* que permitan reaccionar de forma automática a los eventos para solucionar un problema. El entorno de monitorización podría montarse con *software* tipo Zabbix, Nagios, o la propia monitorización de GCP.
- **Solución para el escalado horizontal**, que podría ser **proactiva** y de forma manual, adelantándose a próximos eventos conocidos que

podrían alterar el consumo de recursos de la plataforma o **reactiva** y de forma automática, actuando a través de los *event handlers* lanzados desde la plataforma de monitorización.

En ambos casos, al igual que se ha realizado con los otros laboratorios y como base del trabajo, las soluciones se aplicarían con zero-downtime o con la menor pérdida de servicio posible.

# Glosario

**AWS** – Amazon web services.

**Backend** – parte trasera, en segundo plano.

**Bash** - intérprete de comandos y lenguaje de scripting para sistemas operativos basados en Unix.

**Bootstrapping** – Arranque

**CA** - Certification Authority, autoridad certificadora.

**Cloud computing** – computación en la nube.

**Cluster** – sistema de procesamiento paralelo o distribuido.

**Donor** – donante

**Event handler** – manejador de eventos.

**Failback** – devolución a su estado original antes del error.

**Failover** – conmutación por error.

**Frontend** – parte frontal, en primer plano.

**GCP** – Google Cloud Platform.

**GB** – gigabyte

**GHz** – gigahercio

**GNU** - GNU's Not Unix, GNU No es Unix.

**GPL** - General public license, licencia pública general.

**HA** – High availability, alta disponibilidad.

**IST** – Incremental state transfer, estado de transferencia incremental.

**Joiner** – el que se une

**JSON** - JavaScript Object Notation, notación de objeto de JavaScript. Formato de texto sencillo para el intercambio de datos.

**Listener** – oyente

**Multimaster** – múltiples nodos maestros o principales.

**On-premise** – en las propias instalaciones, en local.

**OS** – Operating system, sistema operativo.

**PAM** - Pluggable authentication modules, módulos de autenticación flexibles.

**PITR** – Point-in-time Recovery, punto de restauración en el tiempo.

**Quorum** – Consenso

**RDBMS** - Relational database management system, sistema gestor de bases de datos relacional.

**Seqno** - Sequence number, número de secuencia.

**SGBD** – Sistema gestor de bases de datos, en inglés, DBMS, Database management system.

**Split-Brain** – cerebro dividido, cada uno por su cuenta.

**Splitter** – divisor

**SST** – State snapshot transfer, instantánea del estado de transferencia.

**Standby** – referido a nodo de apoyo, a la espera.

**Switchover** – pasar a otra cosa, conversión, conmutación, transición.

**Upgrade** – mejora

**vCPU** – virtual CPU, unidad central de procesamiento virtual.

**VIP** – Virtual IP, IP virtual, flotante.

**VM** – Virtual machine, máquina virtual.

**VRRP** - Virtual router redundancy protocol, protocolo de redundancia de enrutador virtual.

**WSREP** - Write set replication, replicación de conjunto de operaciones de escritura.

**XML** - eXtensible markup language, lenguaje de marcas extensible.

**Zero-Downtime** – método para la implementación de cambios en una aplicación sin interrupción del servicio.

# Bibliografía

[1] MariaDB Community Server [consulta: abril 2024] https://mariadb.com/products/community-server/

[2] Amazon Web Services [consulta: abril 2024] https://aws.amazon.com/es/compare/the-difference-between-mariadb-vs-mysql/

[3] Google Trends [consulta: abril 2024] https://trends.google.com/trends/

[4] MariaDB Galera Cluster [consulta: abril 2024] https://mariadb.com/kb/en/galera-cluster/

[5] Medium.com DBA Jungle [consulta: abril 2024] https://medium.com/dba-jungle/make-mariadb-galera-cluster-auto-recovery-fb1ce1d89f09

[6] Architecture of MariaDB Enterprise Cluster, Powered by Galera [consulta: abril 2024] https://mariadb.com/docs/server/architecture/components/enterprise-server/enterprise-cluster/

[7] Galera Cluster [consulta: abril 2024] https://galeracluster.com/library/documentation/weighted-quorum.html

[8] MariaDB MaxScale [consulta: abril 2024] https://mariadb.com/docs/server/products/mariadb-maxscale/

[9] MaxScale Use Cases [consulta: abril 2024] https://mariadb.com/docs/server/architecture/components/maxscale/use-cases

[10] Maria MaxScale 6.0 Native Clustering [consulta: abril 2024] https://mariadb.com/es/resources/blog/mariadb-maxscale-6-0-native-clustering/

[11] rasyiqul.wordpress.com [consulta: abril 2024] https://rasyiqul.wordpress.com/2019/03/08/setup-maxscale-ha-using-keepalived-and-maxctrl/

[12] Wikipedia - Virtual Router Redundancy Protocol [consulta: abril 2024] https://es.wikipedia.org/wiki/Virtual_Router_Redundancy_Protocol

[13] TecMint [consulta: marzo 2024] https://www.tecmint.com/install-mariadb-in-debian/

[14] MariaDB [consulta marzo 2024] https://mariadb.com/kb/en/securing-communications-in-galera-cluster/

[15] MariaDB - server system variables [consulta: marzo 2024] https://mariadb.com/kb/en/server-system-variables/

[16] GaleraCluster [consulta: mayo 2024] https://galeracluster.com/library/documentation/monitoring-cluster.html

[17] GaleraCluster [consulta: mayo 2024] https://galeracluster.com/library/documentation/quorum-reset.html

[18] MariaDB [consulta: mayo 2024] https://mariadb.com/es/resources/blog/configuring-mariadb-databases-for-disaster-recovery-for-self-hosted-servicenow-deployments/

[19] severalnines [consulta: mayo 2024] https://severalnines.com/blog/maxscale-basic-management-using-maxctrl-mariadb-cluster/

[20] Fromdual [consulta: mayo 2024] https://fromdual.com/maxscale-configurations-synchronisation

[21] MariaDB readwritesplit [consulta: mayo 2024] https://mariadb.com/kb/en/mariadb-maxscale-6-readwritesplit

[22] MariaDB [consulta: mayo 2024] https://mariadb.com/kb/en/upgrading-from-mariadb-10-11-to-mariadb-11-0/

[23] github.com – GoogleCloudPlatform [consulta: marzo 2024] https://github.com/GoogleCloudPlatform/solutions-floating-ip-patterns-terraform/tree/main/3-ilb-keepalived

[24] Google Cloud – Cloud Architecture Center [consulta: marzo 2024] https://cloud.google.com/architecture/patterns-for-floating-ip-addresses-in-compute-engine?hl=es-419

[25] Google Cloud – Cloud Load Balancing [consulta: marzo 2024] https://cloud.google.com/load-balancing/docs/health-check-concepts?hl=es-419#ip-ranges

[26] Apache JMeter [consulta: mayo 2024] https://jmeter.apache.org/usermanual/build-db-test-plan.html

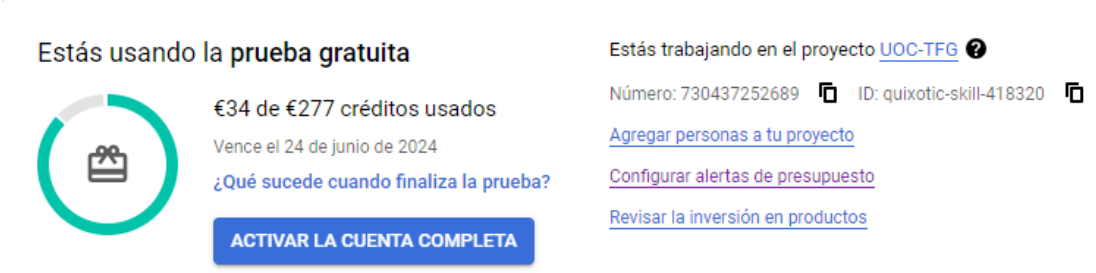
[27] MariaDB [consulta: mayo 2024] https://mariadb.com/kb/en/incremental-backup-and-restore-with-mariabackup/

[28] MariaDB [consulta: mayo 2024] https://mariadb.com/kb/en/full-backup-and-restore-with-mariabackup/

# Anexos

## Anexo 1: Despliegue de la infraestructura en Google Cloud

Una vez completado el registro y activado la cuenta de prueba gratuita, que nos proporciona un crédito de 300$ y un periodo de validez de 90 días, habrá que crear un proyecto, en nuestro caso lo hemos llamado UOC-TFG (quixotic-skill-418320)



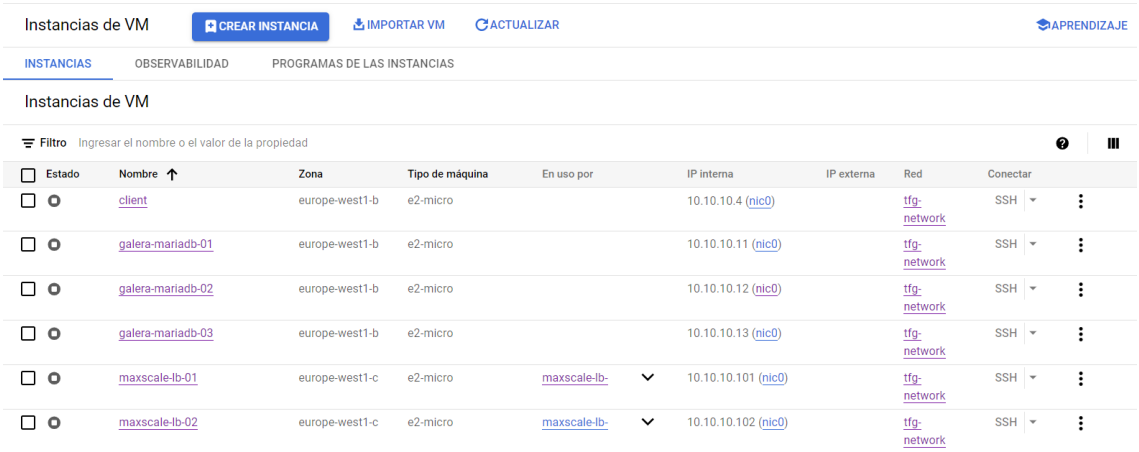

A continuación, vamos a crear una red de VPC (*Virtual Private Cloud*) con nombre “tfg-network” y una subred “tfg-subnet”, asociadas al proyecto y en el rango 10.10.10.0/24 para dar conectividad a las instancias que se crearán más adelante.

Comandos gcloud para la creación de la red y subred:

```
$ gcloud compute networks create tfg-network --project=quixotic-skill-418320 --subnet-mode=custom --mtu=1460 --bgp-routing-mode=regional

$ gcloud compute networks subnets create tfg-subnet --project=quixotic-skill-418320 --range=10.10.10.0/24 --stack-type=IPV4_ONLY --network=tfg-network --region=europe-west1
```

Aprovechamos también para la creación de un par de reglas de firewall para permitir el acceso externo al puerto 22 (ssh) de las máquinas y otra (intranet) que permitirá todo el tráfico interno entre las máquinas.


Lo siguiente será la creación de instancias de VM. Se han dado de alta 6 máquinas con las siguientes características y direccionamiento e imagen de sistema operativo Debian GNU/Linux 12 (bookworm).




Comando gcloud para la creación de instancias:

```
$ gcloud compute instances create galera-mariadb-01 \
    --project=quixotic-skill-418320 \
    --zone=europe-west1-b \
    --machine-type=e2-micro \
                --network-interface=network-tier=PREMIUM,private-network-
ip=10.10.10.11,stack-type=IPV4_ONLY,subnet=tfg-subnet \
          --hostname=galera-mariadb-01.cluster.lab \
    --maintenance-policy=MIGRATE \
    --provisioning-model=STANDARD \
                                                --service-account=730437252689-
compute@developer.gserviceaccount.com \
           --scopes=https://www.googleapis.com/auth/devstorage.read_only,https://
www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/
monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://
www.googleapis.com/auth/service.management.readonly,https://
www.googleapis.com/auth/trace.append \
             --create-disk=auto-delete=yes,boot=yes,device-name=galera-mariadb-
01,image=projects/debian-cloud/global/images/debian-12-bookworm-
v20240312,mode=rw,size=10,type=projects/quixotic-skill-418320/zones/europe-
west1-b/diskTypes/pd-balanced \
    --no-shielded-secure-boot \
    --shielded-vtpm \
    --shielded-integrity-monitoring \
    --labels=goog-ec-src=vm_add-gcloud \
    --reservation-affinity=any
```

## Anexo 2: Solución para el uso de direcciones IP flotantes en GCE [24]

Para poder configurar VRRP (Virtual Router Redundancy Protocol) y la IP flotante para Keepalived, que pueda conmutar de una VM a otra en caso de fallo en el servicio de MaxScale, es necesario realizar la siguiente configuración previa para Google Compute Engine (en adelante, GCE), dado que éste no soporta los mecanismos de implementación típicos de los entornos locales.

En los entornos on-premise, al conmutar una IP virtual de un servidor a otro, éste anuncia la toma de control a otros dispositivos de la capa 2 a través del envío de un marco de protocolo de resolución de direcciones ARP gratuito, el cual es ignorado en las redes VPC de GCE

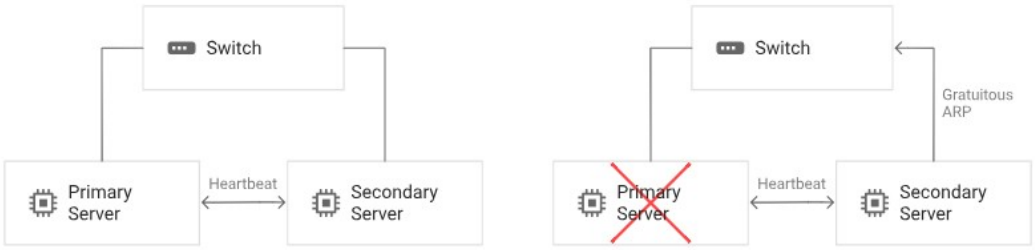



**Figura 8 - Configuración típica de IP flotante en entorno On-premise [25]**

En la imagen siguiente se podrá ver la arquitectura de IP flotante para una red de VPC en Google Cloud para el caso concreto del balanceo de carga con el patrón de conmutación por error y verificaciones de estado expuestas por una señal de monitoreo (socket 6000), que será el que utilicemos para nuestro laboratorio.

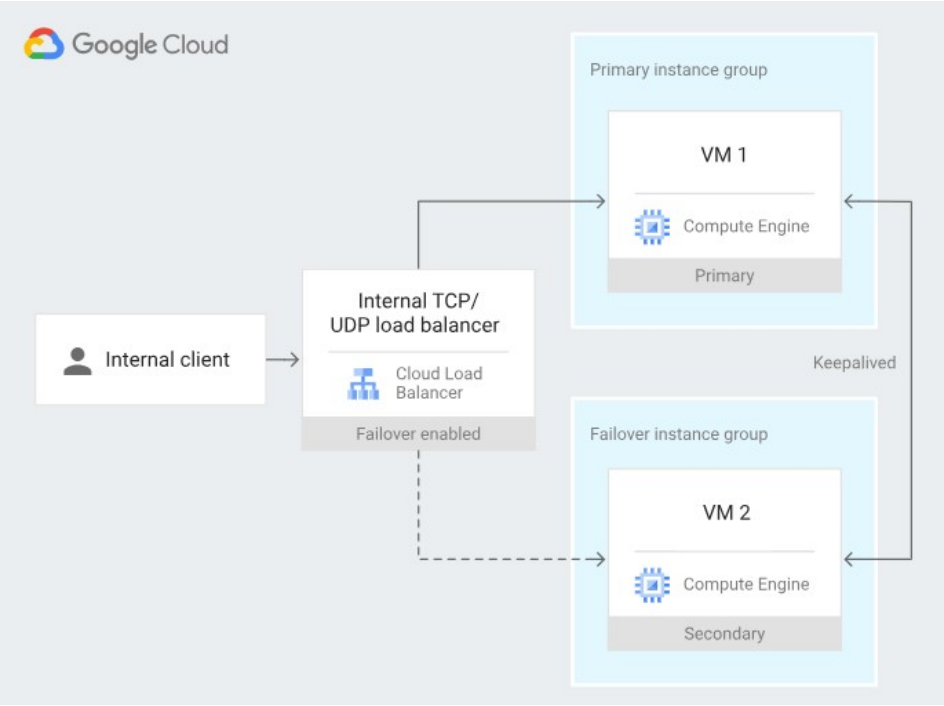


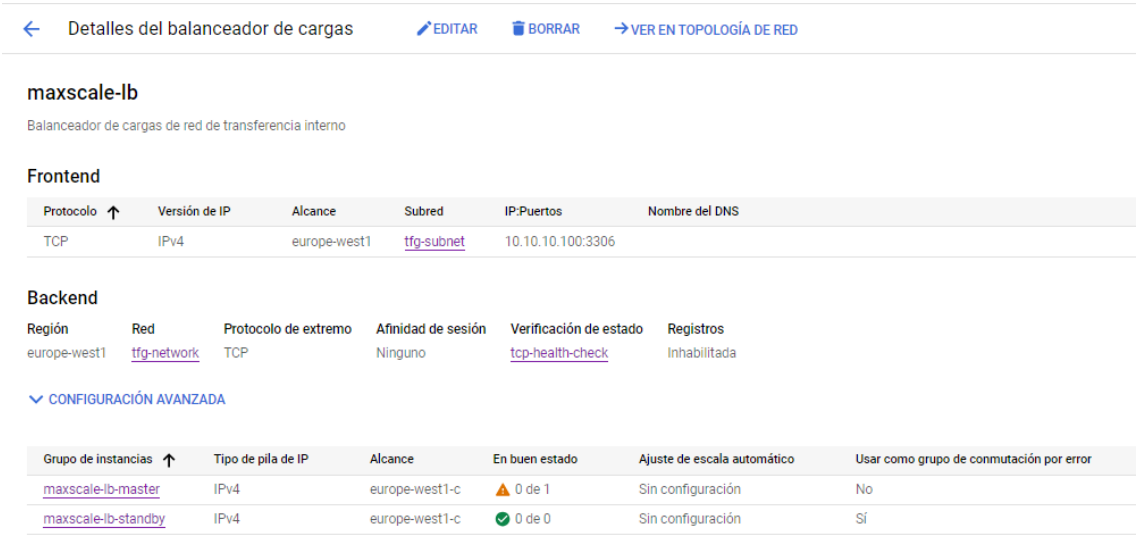

**Figura 9 - Configuración de IP flotante para entorno GCE [21]**

En primer lugar, se crea la regla de firewall para exponer el puerto (socket 6000) a los rangos de IP de sondeo de Google Cloud [26], que utilizaremos como *health-check* para activar el mecanismo de failover con Keepalived para MaxScale.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| failover-hc | Regla de firewall de entrada | Global | 1000 | Aplicar a todas | Rangos de IPv4: 130.211.0.0/22, 35.191.0.0/16 | – | tcp:6000 | Permitir |

Crearemos dos grupos de instancia para los servidores *máster* y *standby* de MaxScale

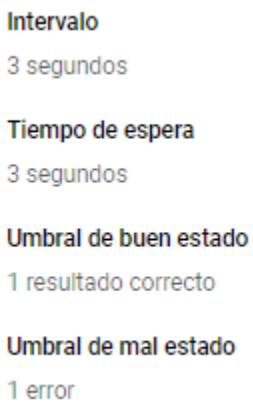

En servicios de red, crearemos un balanceador de cargas (maxscale-lb), cuyo *frontend* será la VIP y puerto de MaxScale (10.10.10.100:3306) y los *backend*, los dos grupos de instancia asignados a los servidores máster y *standby* de MaxScale con el puerto 6000 utilizado para el chequeo de salud, el cual será el que compruebe el estado el balanceador de carga de GCE para redirigir la VIP hacia un nodo u otro de MaxScale.




Nota[1]: igualmente se podría haber configurado como puerto de *health-check* el 3306 de MaxScale, pero en este caso habría que exponer dicho puerto hacia fuera de la intranet (IPs de sondeo de Google Cloud) y no nos interesa por seguridad.

Nota[2]: se ha configurado “maxscale-lb-standby” como grupo de conmutación por error para que la VIP siempre retorne al nodo máster cuando éste responda correctamente al chequeo de salud.

Las verificaciones de estado para el chequeo de salud se han configurado con los siguientes valores de *timeout* e *interval* para evitar falsos positivos, por lo que provocará interrupciones del servicio mínimas durante las conmutaciones por *failover*, tal y como se comprobará en el laboratorio.



## Anexo 3: Instalación y configuración de Apache JMeter

Para comprobar el estado del servicio en todo momento y durante el periodo de ejecución de los laboratorios, se han configurado unas pruebas de carga básicas con usuarios concurrentes sobre JMeter, realizando operaciones de lectura y escritura sobre MariaDB, simulando un periodo de trabajo ordinario sobre las bases de datos.

En primer lugar, se ha creado una base de datos de prueba en MariaDB, con una única tabla y dos campos, tal y como puede verse en la siguiente descripción de la tabla:

```
MariaDB [(none)]> desc test.test_table;
+-------+---------+-------+------+----------+--------------------+
| Field | Type    | Null  | Key | Default | Extra              |
+-------+---------+-------+------+----------+--------------------+
| id    | int(11) | NO    | PRI | NULL    | auto_increment |
| msg   | text    | YES   |     | NULL    |                    |
+-------+---------+------+-----+-----------+--------------------+
2 rows in set (0.006 sec)
```

Y un usuario para la conexión con los siguientes permisos:

```
MariaDB [(none)]> SHOW GRANTS FOR 'pruebas'@'%';
+-----------------------------------------------------------------------------------------------------------------+
| Grants for pruebas@%                                                                                            |
+-----------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `pruebas`@`%` IDENTIFIED BY PASSWORD '*6BB4837EB74329105EE4568DDA7DC67ED2CA2AD9' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `test`.* TO `pruebas`@`%` |
+-----------------------------------------------------------------------------------------------------------------+
```

En el equipo ‘client.cluster.lab’, se ha instalado desde paquetería el cliente para MariaDB y el entorno para java con OpenJDK con las siguientes versiones:

```
$ dpkg --list | grep -iE 'mariadb-client|openjdk'
ii  mariadb-client   1:10.11.7-0+deb12u1   amd64   MariaDB database client binaries
ii  mariadb-client-core   1:10.11.7-0+deb12u1   amd64   MariaDB database core client binaries
ii  openjdk-17-jdk-headless:amd64  17.0.11+9-1~deb12u1   amd64   OpenJDK Development Kit (JDK) (headless)
ii  openjdk-17-jre:amd64   17.0.11+9-1~deb12u1   amd64   OpenJDK Java runtime, using Hotspot JIT
ii  openjdk-17-jre-headless:amd64   17.0.11+9-1~deb12u1   amd64   OpenJDK Java runtime, using Hotspot JIT (headless)
```

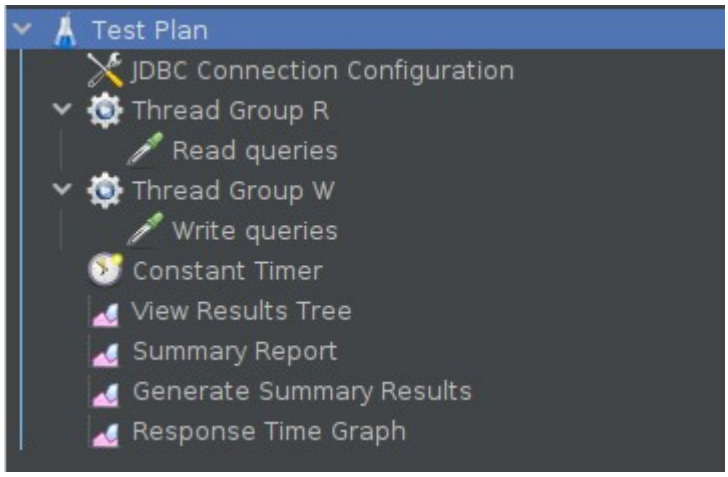
También se ha descargado la versión 5.6.3 de Apache JMeter y el conector java para mariadb:

```
$ wget https://dlcdn.apache.org//jmeter/binaries/apache-jmeter-5.6.3.zip
$ unzip apache-jmeter-5.6.3.zip

$ cd /opt/apache-jmeter-5.6.3/lib/
```

```
$ wget https://dlm.mariadb.com/3752081/Connectors/java/connector-java-3.3.3/mariadb-java-client-3.3.3.jar
```

Tras toda esta configuración previa, ya podemos ejecutar el binario de JMeter (/opt/apache-jmeter-5.6.3/bin/jmeter) y empezar con la configuración para las pruebas de carga.

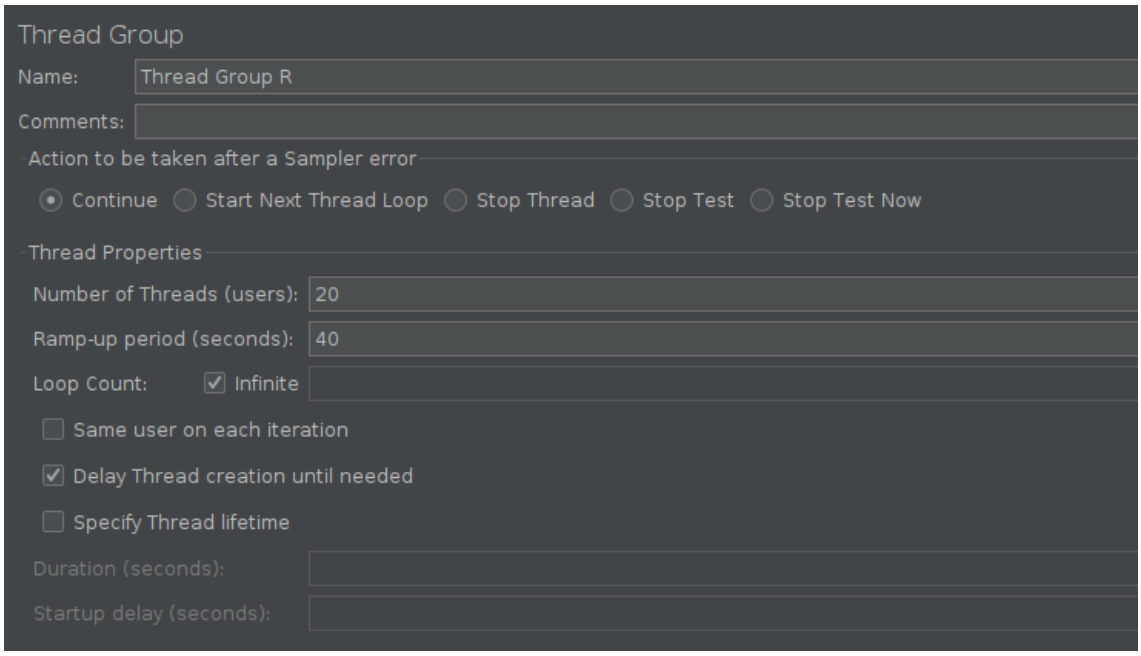
Se ha configurado el *Test Plan* siguiente y que se describe a continuación:



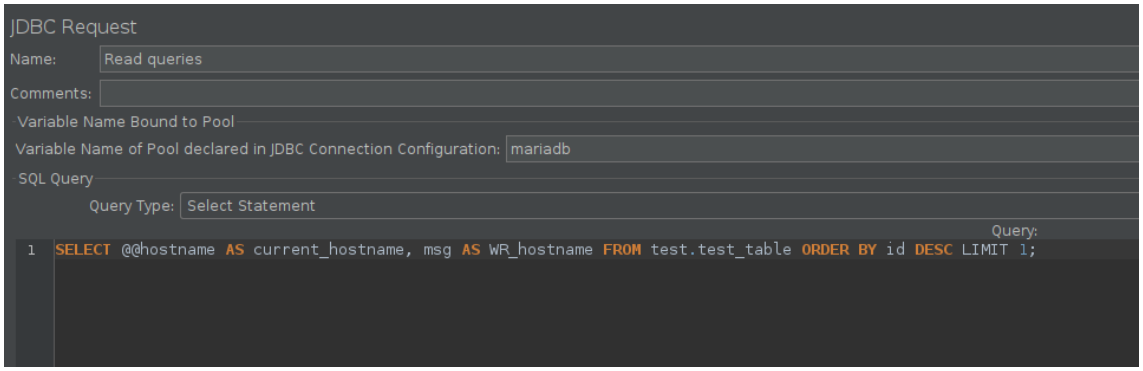

En primer lugar, se añade un elemento de configuración de tipo 'JDBC Connection Configuration' para la conexión con la base de datos, donde se configuran los datos de conexión a la Virtual IP de KeepAlived (10.10.10.100) y la clase del driver JDBC que descargamos previamente. El nombre de la variable lo hemos llamado 'mariadb':


Se añade un elemento Thread Group para las operaciones de lectura, que hemos llamado 'Thread Group R'. Definimos en 20 el número de usuarios concurrentes que van a ejecutar operaciones de lectura y un periodo de despliegue de 40 segundos para que no se ejecuten todos los hilos a la vez, es decir, se abrirá un hilo cada 40/20=2 segundos. El contador de bucles en

infinito y distintos usuarios en cada iteración del bucle y se configura para que continue el plan de pruebas en caso de haber un error.



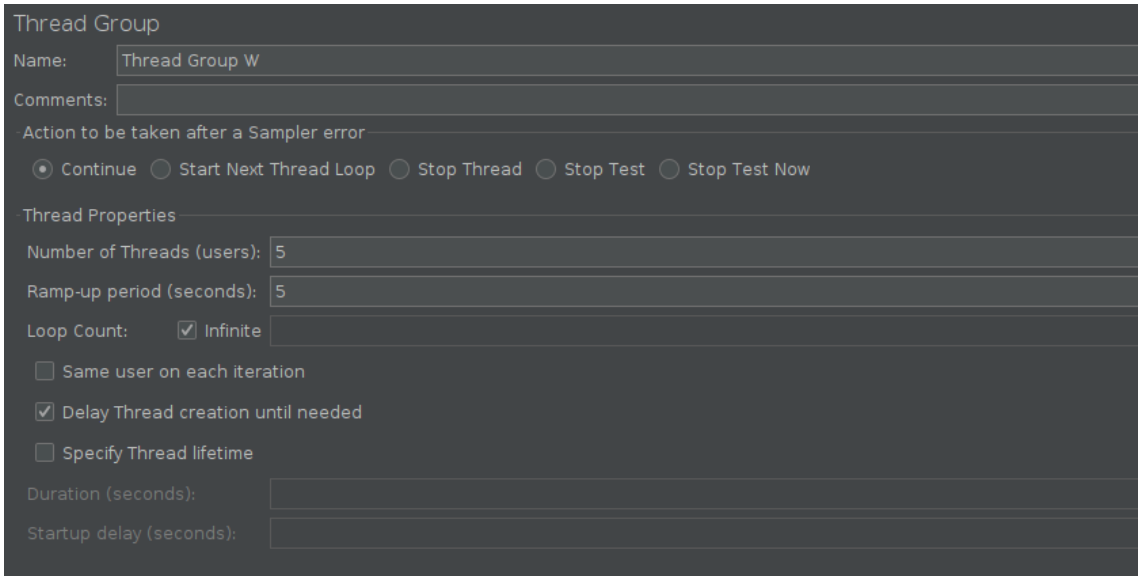

Dentro del Thread Group, añadimos un sampler del tipo 'JDBC Request', donde se configura la *query* que ejecutará la transacción de lectura sobre la base de datos en el *pool* 'mariadb' configurado anteriormente en la conexión JDBC. Lo hemos llamado 'Read queries' y el tipo de petición será 'Select Statement'.



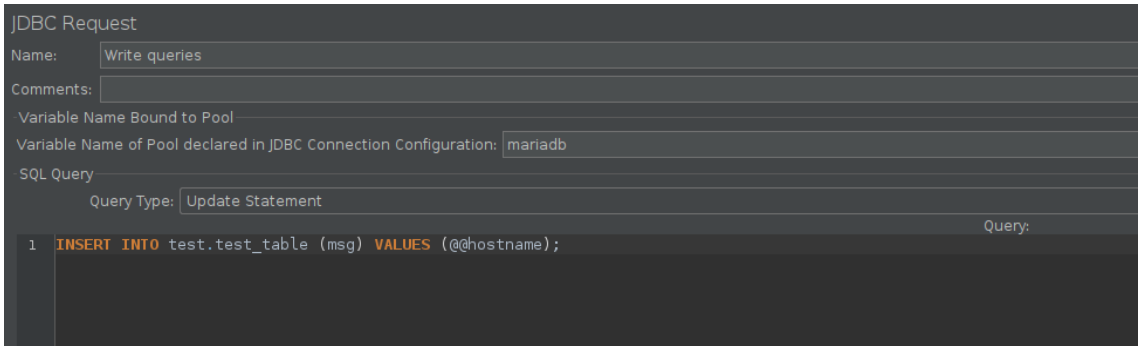

Dicha query devolverá el siguiente resultado cada vez que se ejecute, donde se puede ver el host de MariaDB que ejecuta la consulta y el último nodo que ha ejecutado un INSERT, de esta forma podemos comprobar como la operación de lectura se realiza sobre uno de los nodos standby y la de escritura sobre el nodo master.

```
MariaDB [test]> SELECT @@hostname AS current_hostname, msg AS
WR_hostname FROM test.test_table ORDER BY id DESC LIMIT 1;
+-----------------------------------+-----------------------------------+
| current_hostname                  | WR_hostname                       |
+-----------------------------------+-----------------------------------+
| galera-mariadb-02.cluster.lab | galera-mariadb-01.cluster.lab |
+-----------------------------------+-----------------------------------+
1 row in set (0.001 sec)
```

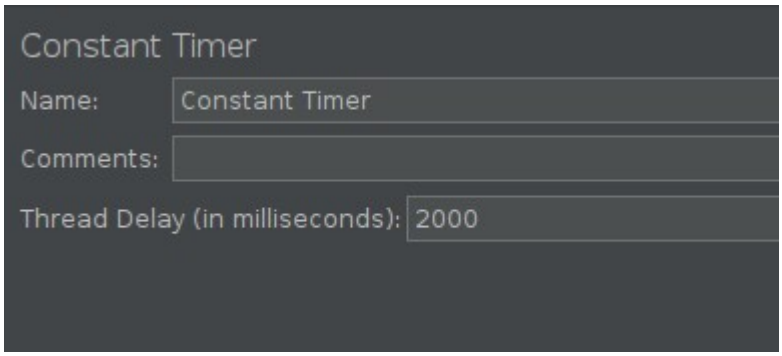
A continuación, se crea otro elemento *Thread Group* para las operaciones de escritura, que hemos llamado 'Thread Group W'. En este caso definimos 5 usuarios concurrentes y un periodo de despliegue de 5 segundos, es decir, se abrirá un hilo cada segundo y se irá ejecutando en un bucle infinito. El resto de configuración igual que en el *Thread Group* de lectura.




De nuevo, al igual que en el caso anterior, creamos un sampler de tipo 'JDBC Request' para la query de escritura dentro del Thread Group y que hemos llamado 'Write queries'. Se ejecuta sobre el mismo pool 'mariadb' de la conexión JDBC anterior y en este caso, el tipo de petición será 'Update Statement'.

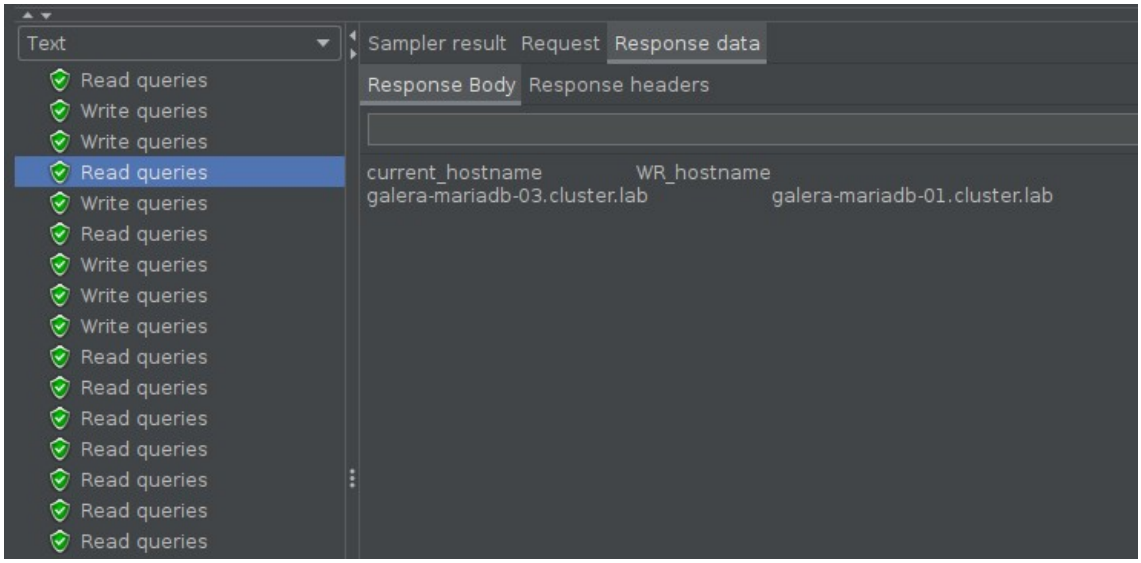



Se crea también un *timer* tipo 'Constant Timer' con un retraso entre iteraciones del bucle de 2000ms para no saturar los servidores, ya que al ser un laboratorio y no un entorno de producción cuenta con muy pocos recursos.

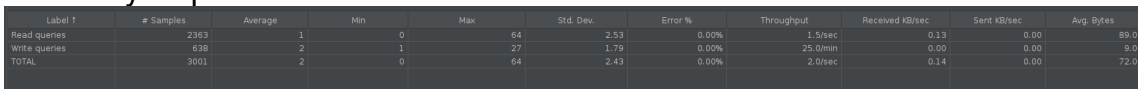



Por último, se han añadido cuatro Listeners para poder ir comprobando los resultados, tanto en tiempo real como al finalizar el laboratorio con informes y gráficas.

View Results Tree

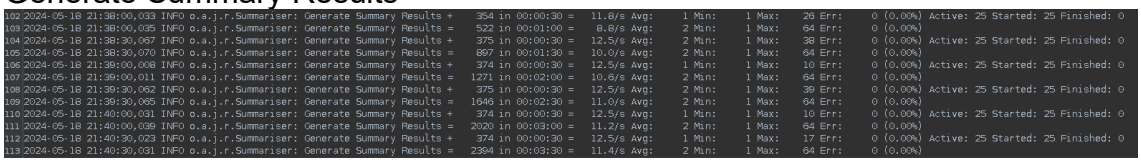



Summary Report

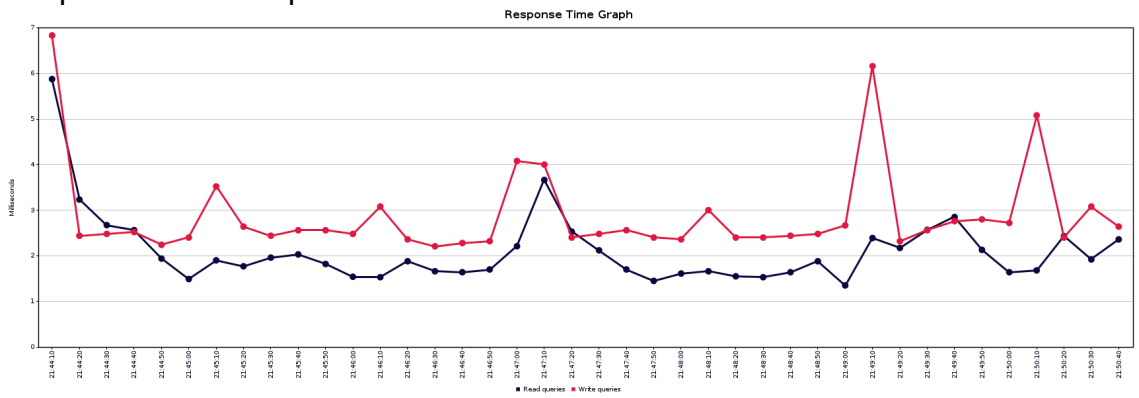


| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| Read queries | 2363 | 1 | 0 | 64 | 2.53 | 0.00% | 1.5/sec | 0.13 | 0.00 | 89.0 |
| Write queries | 638 | 2 | 1 | 27 | 1.79 | 0.00% | 25.0/min | 0.00 | 0.00 | 9.0 |
| TOTAL | 3001 | 2 | 0 | 64 | 2.43 | 0.00% | 2.0/sec | 0.14 | 0.00 | 72.0 |

Generate Summary Results



Response Time Graph




Contenido final del fichero jmx con el plan de pruebas configurado:

```
<?xml version="1.0" encoding="UTF-8"?>
<jmeterTestPlan version="1.2" properties="5.0" jmeter="5.6.3">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test
Plan">
      <elementProp name="TestPlan.user_defined_variables"
elementType="Arguments" guiclass="ArgumentsPanel" testclass="Arguments"
testname="User Defined Variables">
        <collectionProp name="Arguments.arguments"/>
      </elementProp>
      <boolProp name="TestPlan.functional_mode">false</boolProp>
      <boolProp name="TestPlan.serialize_threadgroups">false</boolProp>
    </TestPlan>
    <hashTree>
      <JDBCDataSource guiclass="TestBeanGUI" testclass="JDBCDataSource"
testname="JDBC Connection Configuration">
        <boolProp name="autocommit">true</boolProp>
        <stringProp name="checkQuery">select 1 from dual</stringProp>
        <stringProp name="connectionAge">1000</stringProp>
        <stringProp name="connectionProperties"></stringProp>
        <stringProp name="dataSource">mariadb</stringProp>
        <stringProp
name="dbUrl">jdbc:mariadb://10.10.10.100:3306/test</stringProp>
        <stringProp name="driver">org.mariadb.jdbc.Driver</stringProp>
        <stringProp name="initQuery"></stringProp>
        <boolProp name="keepAlive">true</boolProp>
        <stringProp name="password">123456</stringProp>
        <stringProp name="poolMax">0</stringProp>
        <boolProp name="preinit">false</boolProp>
        <stringProp name="timeout">5000</stringProp>
        <stringProp name="transactionIsolation">DEFAULT</stringProp>
        <stringProp name="trimInterval">60000</stringProp>
        <stringProp name="username">pruebas</stringProp>
      </JDBCDataSource>
      <hashTree/>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
```

```
testname="Thread Group R" enabled="true">
      <boolProp name="ThreadGroup.delayedStart">true</boolProp>
      <intProp name="ThreadGroup.num_threads">20</intProp>
      <intProp name="ThreadGroup.ramp_time">40</intProp>
      <boolProp
name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
      <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller">
        <intProp name="LoopController.loops">-1</intProp>
        <boolProp name="LoopController.continue_forever">false</boolProp>
      </elementProp>
    </ThreadGroup>
    <hashTree>
      <JDBCSampler guiclass="TestBeanGUI" testclass="JDBCSampler"
testname="Read queries" enabled="true">
        <stringProp name="dataSource">mariadb</stringProp>
        <stringProp name="query">SELECT @@hostname AS
current_hostname, msg AS WR_hostname FROM test.test_table ORDER BY id
DESC LIMIT 1;</stringProp>
        <stringProp name="queryArguments"></stringProp>
        <stringProp name="queryArgumentsTypes"></stringProp>
        <stringProp name="queryTimeout"></stringProp>
        <stringProp name="queryType">Select Statement</stringProp>
        <stringProp name="resultSetHandler">Store as String</stringProp>
        <stringProp name="resultSetMaxRows"></stringProp>
        <stringProp name="resultVariable"></stringProp>
        <stringProp name="variableNames"></stringProp>
      </JDBCSampler>
      <hashTree/>
    </hashTree>
    <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Thread Group W" enabled="true">
      <boolProp name="ThreadGroup.delayedStart">true</boolProp>
      <intProp name="ThreadGroup.num_threads">5</intProp>
      <intProp name="ThreadGroup.ramp_time">5</intProp>
      <boolProp
name="ThreadGroup.same_user_on_next_iteration">false</boolProp>
      <stringProp
name="ThreadGroup.on_sample_error">continue</stringProp>
      <elementProp name="ThreadGroup.main_controller"
elementType="LoopController" guiclass="LoopControlPanel"
testclass="LoopController" testname="Loop Controller">
        <intProp name="LoopController.loops">-1</intProp>
        <boolProp name="LoopController.continue_forever">false</boolProp>
      </elementProp>
    </ThreadGroup>
    <hashTree>
```

```
      <JDBCSampler guiclass="TestBeanGUI" testclass="JDBCSampler"
testname="Write queries" enabled="true">
        <stringProp name="dataSource">mariadb</stringProp>
        <stringProp name="queryType">Update Statement</stringProp>
        <stringProp name="query">INSERT INTO test.test_table (msg) VALUES
(@@hostname);</stringProp>
        <stringProp name="queryArguments"></stringProp>
        <stringProp name="queryArgumentsTypes"></stringProp>
        <stringProp name="variableNames"></stringProp>
        <stringProp name="resultVariable"></stringProp>
        <stringProp name="queryTimeout"></stringProp>
        <stringProp name="resultSetMaxRows"></stringProp>
        <stringProp name="resultSetHandler">Store as String</stringProp>
      </JDBCSampler>
      <hashTree/>
    </hashTree>
    <ConstantTimer guiclass="ConstantTimerGui" testclass="ConstantTimer"
testname="Constant Timer" enabled="true">
      <stringProp name="ConstantTimer.delay">2000</stringProp>
    </ConstantTimer>
    <hashTree/>
    <ResultCollector guiclass="ViewResultsFullVisualizer"
testclass="ResultCollector" testname="View Results Tree" enabled="true">
      <boolProp name="ResultCollector.error_logging">false</boolProp>
      <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
          <time>true</time>
          <latency>true</latency>
          <timestamp>true</timestamp>
          <success>true</success>
          <label>true</label>
          <code>true</code>
          <message>true</message>
          <threadName>true</threadName>
          <dataType>true</dataType>
          <encoding>false</encoding>
          <assertions>true</assertions>
          <subresults>true</subresults>
          <responseData>false</responseData>
          <samplerData>false</samplerData>
          <xml>false</xml>
          <fieldNames>true</fieldNames>
          <responseHeaders>false</responseHeaders>
          <requestHeaders>false</requestHeaders>
          <responseDataOnError>false</responseDataOnError>

<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMes
sage>
          <assertionsResultsToSave>0</assertionsResultsToSave>
```

```
          <bytes>true</bytes>
          <sentBytes>true</sentBytes>
          <url>true</url>
          <threadCounts>true</threadCounts>
          <idleTime>true</idleTime>
          <connectTime>true</connectTime>
        </value>
      </objProp>
      <stringProp name="filename"></stringProp>
    </ResultCollector>
    <hashTree/>
    <ResultCollector guiclass="SummaryReport" testclass="ResultCollector"
testname="Summary Report" enabled="true">
      <boolProp name="ResultCollector.error_logging">false</boolProp>
      <objProp>
        <name>saveConfig</name>
        <value class="SampleSaveConfiguration">
          <time>true</time>
          <latency>true</latency>
          <timestamp>true</timestamp>
          <success>true</success>
          <label>true</label>
          <code>true</code>
          <message>true</message>
          <threadName>true</threadName>
          <dataType>true</dataType>
          <encoding>false</encoding>
          <assertions>true</assertions>
          <subresults>true</subresults>
          <responseData>false</responseData>
          <samplerData>false</samplerData>
          <xml>false</xml>
          <fieldNames>true</fieldNames>
          <responseHeaders>false</responseHeaders>
          <requestHeaders>false</requestHeaders>
          <responseDataOnError>false</responseDataOnError>

<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMes
sage>
          <assertionsResultsToSave>0</assertionsResultsToSave>
          <bytes>true</bytes>
          <sentBytes>true</sentBytes>
          <url>true</url>
          <threadCounts>true</threadCounts>
          <idleTime>true</idleTime>
          <connectTime>true</connectTime>
        </value>
      </objProp>
      <stringProp name="filename"></stringProp>
    </ResultCollector>
```

```
      <hashTree/>
      <Summariser guiclass="SummariserGui" testclass="Summariser"
testname="Generate Summary Results" enabled="true"/>
      <hashTree/>
      <ResultCollector guiclass="RespTimeGraphVisualizer"
testclass="ResultCollector" testname="Response Time Graph" enabled="true">
        <boolProp name="ResultCollector.error_logging">false</boolProp>
        <objProp>
          <name>saveConfig</name>
          <value class="SampleSaveConfiguration">
            <time>true</time>
            <latency>true</latency>
            <timestamp>true</timestamp>
            <success>true</success>
            <label>true</label>
            <code>true</code>
            <message>true</message>
            <threadName>true</threadName>
            <dataType>true</dataType>
            <encoding>false</encoding>
            <assertions>true</assertions>
            <subresults>true</subresults>
            <responseData>false</responseData>
            <samplerData>false</samplerData>
            <xml>false</xml>
            <fieldNames>true</fieldNames>
            <responseHeaders>false</responseHeaders>
            <requestHeaders>false</requestHeaders>
            <responseDataOnError>false</responseDataOnError>

<saveAssertionResultsFailureMessage>true</saveAssertionResultsFailureMes
sage>
            <assertionsResultsToSave>0</assertionsResultsToSave>
            <bytes>true</bytes>
            <sentBytes>true</sentBytes>
            <url>true</url>
            <threadCounts>true</threadCounts>
            <idleTime>true</idleTime>
            <connectTime>true</connectTime>
          </value>
        </objProp>
        <stringProp name="filename"></stringProp>
      </ResultCollector>
      <hashTree/>
    </hashTree>
  </hashTree>
</jmeterTestPlan>
```