2025

Implementación de Replicación Homogénea MySQL Master-Slave con Docker

DISEÑO, CONFIGURACIÓN Y VALIDACIÓN TÉCNICA DE UNA ARQUITECTURA DE REPLICACIÓN DE BASES DE DATOS

YANDRI PISCOCAMA; WILSON PALMA

UNIVERSIDAD NACIONAL DE LOJA | Ing. René Guamán

IMPLEMENTACIÓN DE REPLICACIÓN HOMOGÉNEA MYSQL MASTER-SLAVE CON DOCKER

DISEÑO, CONFIGURACIÓN Y VALIDACIÓN TÉCNICA DE UNA ARQUITECTURA DE REPLICACIÓN DE BASES DE DATOS

YANDRI PISCOCAMA
WILSON PALMA
2025-11-19

Abstract

Este informe documenta la implementación completa de un entorno de replicación homogénea MySQL (arquitectura Master-Slave) utilizando contenedores Docker. Se detallan la configuración de infraestructura, la creación del usuario de replicación, la activación del binary log y la sincronización mediante binlog en formato ROW. Además se incluyen procedimientos de despliegue (docker compose), verificación de estado, pruebas de replicación en tiempo real, resolución de problemas comunes y comandos de referencia rápida. Los resultados muestran una sincronización efectiva con retraso mínimo, portabilidad del entorno y directrices para escalabilidad y mantenimiento.

1. INTRODUCCIÓN

1.1 CONTEXTO DEL PROYECTO

El presente informe documenta la implementación de un sistema de **replicación homogénea** de bases de datos MySQL utilizando la arquitectura Master-Slave. La replicación homogénea se refiere a la sincronización de datos entre servidores que utilizan el mismo sistema gestor de base de datos (en este caso, MySQL 8.0).

1.2 OBJETIVOS

- Configurar un entorno de replicación MySQL con arquitectura Master-Slave
- Implementar la infraestructura mediante contenedores Docker
- Validar la sincronización de datos en tiempo real
- Documentar el proceso completo incluyendo resolución de problemas

1.3 JUSTIFICACIÓN TÉCNICA

La replicación de bases de datos proporciona:

- **Alta disponibilidad:** Si el servidor master falla, el slave puede tomar el control
- **Distribución de carga:** Las consultas de lectura pueden dirigirse al slave
- **Respaldo en tiempo real:** Los datos se replican automáticamente
- **Escalabilidad horizontal:** Permite añadir múltiples slaves según necesidad

2. MARCO TEÓRICO

2.1 REPLICACIÓN HOMOGÉNEA

La replicación homogénea ocurre cuando tanto el servidor origen (master) como el destino (slave) utilizan el mismo sistema de gestión de base de datos. En este caso, ambos servidores ejecutan MySQL 8.0, lo que garantiza compatibilidad total en tipos de datos, funciones y sintaxis.

2.2 ARQUITECTURA MASTER-SLAVE

- **Master (Maestro):** Servidor principal que recibe todas las operaciones de escritura (INSERT, UPDATE, DELETE)
- **Slave (Esclavo):** Servidor secundario que replica automáticamente los cambios del master y puede atender consultas de lectura

2.3 FUNCIONAMIENTO DE LA REPLICACIÓN

1. El master registra todos los cambios en el **binary log** (binlog)
2. El slave se conecta al master mediante un usuario de replicación
3. El **IO Thread** del slave descarga los eventos del binlog
4. El **SQL Thread** del slave ejecuta esos eventos localmente

3. ENTORNO DE DESARROLLO

3.1 INFRAESTRUCTURA TECNOLÓGICA

Componente	Versión/Descripción
Sistema Operativo	Ubuntu Linux
Containerización	Docker Engine + Docker Compose
SGBD	MySQL 8.0
Cliente GUI	MySQL Workbench
Formato Binlog	ROW (registro a nivel de fila)

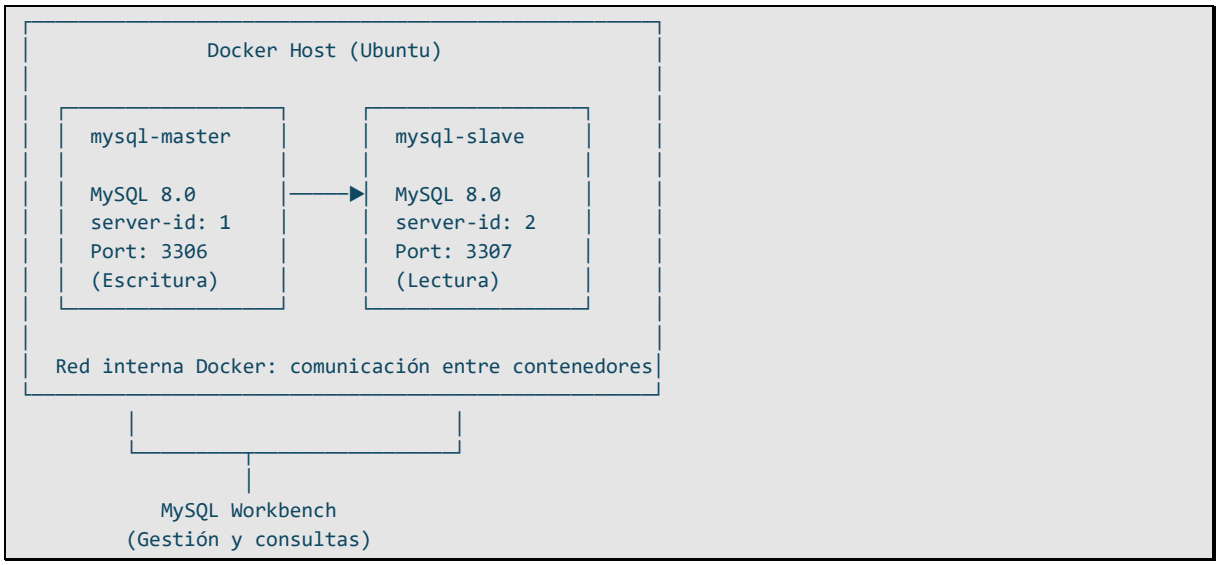
3.2 CREDENCIALES CONFIGURADAS

```
Root Password: Password@54321
Usuario Replicación: replication_user
Password Replicación: Password@54321
Base de Datos: employee
```

Nota de seguridad: Estas contraseñas son para propósitos educativos. En un entorno de producción deben utilizarse contraseñas robustas y gestión segura de secretos.

4. ARQUITECTURA DE LA SOLUCIÓN

4.1 DIAGRAMA DE COMPONENTES



4.2 CONFIGURACIÓN DE PUERTOS

- **Master:** Puerto 3306 (host) → 3306 (contenedor)
- **Slave:** Puerto 3307 (host) → 3306 (contenedor)

Esta configuración permite acceder a ambos servidores desde el host sin conflictos de puertos.

5. IMPLEMENTACIÓN

5.1 ARCHIVO DOCKER-COMPOSE.YML

Este archivo define toda la infraestructura de contenedores:

```yaml
version: "3"
services:
  mysql-master:
    image: mysql:8.0
    container_name: mysql-master
    command: --server-id=1 --log-bin=mysql-bin --binlog-format=row
    environment:
      MYSQL_ROOT_PASSWORD: Password@54321
      MYSQL_DATABASE: employee
      MYSQL_USER: replication_user
      MYSQL_PASSWORD: Password@54321
    ports:
      - "3306:3306"
    networks:
      - mysql-network

  mysql-slave:
    image: mysql:8.0
```

```
    container_name: mysql-slave
    depends_on:
      - mysql-master
    command: --server-id=2 --log-bin=mysql-bin --binlog-format=row
    environment:
      MYSQL_ROOT_PASSWORD: Password@54321
      MYSQL_DATABASE: employee
      MYSQL_USER: replication_user
      MYSQL_PASSWORD: Password@54321
    ports:
      - "3307:3306"
    networks:
      - mysql-network

networks:
  mysql-network:
    driver: bridge
```

Explicación de parámetros clave:

- `--server-id`: Identificador único para cada servidor en la topología de replicación
- `--log-bin=mysql-bin`: Activa el binary log necesario para la replicación
- `--binlog-format=row`: Define que se registran cambios a nivel de fila (más seguro y completo)
- `depends_on`: Garantiza que el master inicie antes que el slave
- `networks`: Red interna para comunicación entre contenedores por nombre

5.2 DESPLIEGUE DE CONTENEDORES

```
# Crear y levantar los servicios en segundo plano
docker compose up -d

# Verificar que ambos contenedores estén ejecutándose
docker ps

# Resultado esperado:
CONTAINER ID   IMAGE        PORTS                    NAMES
abc123...      mysql:8.0    0.0.0.0:3306->3306/tcp   mysql-master
def456...      mysql:8.0    0.0.0.0:3307->3306/tcp   mysql-slave
```

5.3 CONFIGURACIÓN DEL SERVIDOR MASTER

5.3.1 ACCESO AL CONTENEDOR MASTER

```
docker exec -it mysql-master mysql -uroot -p
# Ingresar password: Password@54321
```

5.3.2 CREACIÓN DEL USUARIO DE REPLICACIÓN

```
-- Eliminar usuario previo si existe (para evitar conflictos)
DROP USER IF EXISTS 'replication_user'@'%';

-- Crear usuario con plugin de autenticación nativo (compatible con slave
s)
CREATE USER 'replication_user'@'%'
IDENTIFIED WITH mysql_native_password BY 'Password@54321';

-- Otorgar privilegios necesarios para replicación
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replication_user'@
'%';

-- Aplicar cambios inmediatamente
FLUSH PRIVILEGES;
```

Nota importante: El plugin `mysql_native_password` es necesario porque algunos clientes de replicación no soportan el plugin de autenticación por defecto de MySQL 8.0 (`caching_sha2_password`).

5.3.3 OBTENER INFORMACIÓN DEL ESTADO DEL MASTER

```
SHOW MASTER STATUS;
```

Resultado obtenido:

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000003 |     1733 |              |                  |
+------------------+----------+--------------+------------------+
```

Estos valores son críticos para configurar el slave:

- **File:** Nombre del archivo binlog actual del master
- **Position:** Posición exacta en el binlog desde donde el slave debe comenzar a leer

5.4 CONFIGURACIÓN DEL SERVIDOR SLAVE

5.4.1 ACCESO AL CONTENEDOR SLAVE

```
# Conectar al slave usando el puerto 3307 desde el host
docker exec -it mysql-slave mysql -uroot -p
```

5.4.2 RESETEAR CONFIGURACIÓN PREVIA (SI EXISTE)

```
-- Detener cualquier proceso de replicación activo
STOP SLAVE;
```

```
-- Limpiar configuración anterior de replicación
RESET SLAVE ALL;
```

5.4.3 CONFIGURAR CONEXIÓN AL MASTER

```
CHANGE MASTER TO
  MASTER_HOST='mysql-master',            -- Nombre del contenedor master (
resuelto por red Docker)
  MASTER_PORT=3306,                      -- Puerto interno del master
  MASTER_USER='replication_user',        -- Usuario creado en el master
  MASTER_PASSWORD='Password@54321',      -- Contraseña del usuario
  MASTER_LOG_FILE='mysql-bin.000003',    -- Archivo binlog obtenido de SHO
W MASTER STATUS
  MASTER_LOG_POS=1733;                   -- Posición exacta en el binlog

-- Iniciar los threads de replicación
START SLAVE;
```

5.4.4 VERIFICAR ESTADO DE LA REPLICACIÓN

```
SHOW SLAVE STATUS\G
```

Parámetros clave a verificar:

```
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: mysql-master
                  Master_User: replication_user
                  Master_Port: 3306
             Slave_IO_Running: Yes    ← DEBE SER "Yes"
            Slave_SQL_Running: Yes    ← DEBE SER "Yes"
          Seconds_Behind_Master: 0    ← Indica sincronización
```

Interpretación: - `Slave_IO_Running: Yes` → El thread de E/S está conectado y descargando eventos del master - `Slave_SQL_Running: Yes` → El thread SQL está ejecutando los eventos replicados - `Seconds_Behind_Master: 0` → El slave está completamente sincronizado (no hay retraso)

6. PRUEBAS DE VALIDACIÓN

6.1 CREACIÓN DE BASE DE DATOS DE PRUEBA (EN MASTER)

Desde MySQL Workbench conectado al master (puerto 3306):

```
-- Crear tabla de ejemplo
CREATE TABLE DISCIPLINA (
    codigo_disciplina VARCHAR(10) PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    tipo VARCHAR(20),
```

```
    cupo_maximo INT,
    horario VARCHAR(50),
    entrenador VARCHAR(10)
);

-- Insertar datos iniciales
INSERT INTO DISCIPLINA VALUES
('DIS001', 'Natación', 'Individual', 20, 'Lun,Mié,Vie 07:00-08:00', 'ENT0
01'),
('DIS002', 'Fútbol', 'Colectivo', 22, 'Mar,Jue 16:00-18:00', 'ENT002'),
('DIS003', 'Tenis', 'Individual', 10, 'Lun,Mié 15:00-17:00', 'ENT003');
```

6.2 VERIFICACIÓN DE REPLICACIÓN AUTOMÁTICA (EN SLAVE)

Conectar al slave (puerto 3307) y verificar que los datos existen:

```
SELECT * FROM DISCIPLINA;
```

Resultado esperado: Las tres filas deben aparecer automáticamente, confirmando que la replicación inicial funciona.

6.3 PRUEBA DE REPLICACIÓN EN TIEMPO REAL

EN EL MASTER:

```
-- Insertar un nuevo registro
INSERT INTO DISCIPLINA VALUES
('DIS011', 'MMA', 'Colectivo', 25, 'Lun,Mié 18:00-20:00', 'ENT001');
```

EN EL SLAVE (INMEDIATAMENTE DESPUÉS):

```
-- Verificar que el registro se replicó
SELECT * FROM DISCIPLINA WHERE codigo_disciplina='DIS011';
```

Resultado: La fila aparece en el slave sin intervención manual, confirmando que la replicación en tiempo real funciona correctamente.

6.4 PRUEBA DE MODIFICACIÓN Y ELIMINACIÓN

```
-- En Master: Actualizar registro
UPDATE DISCIPLINA
SET cupo_maximo = 30
WHERE codigo_disciplina = 'DIS011';

-- En Master: Eliminar registro
DELETE FROM DISCIPLINA WHERE codigo_disciplina = 'DIS003';
```

Verificar en el slave que ambos cambios se replicaron correctamente.

7. RESOLUCIÓN DE PROBLEMAS

Durante la implementación se encontraron y resolvieron los siguientes problemas:

7.1 CONFLICTO: CONTENEDORES CON NOMBRES DUPLICADOS

Síntoma: Error al ejecutar `docker compose up`

```
Error response from daemon: Conflict. The container name "/mysql-master" is already in use
```

Causa: Contenedores previos con los mismos nombres no fueron eliminados.

Solución:

```
# Detener contenedores existentes
docker stop mysql-master mysql-slave

# Eliminar contenedores
docker rm mysql-master mysql-slave

# Volver a levantar servicios
docker compose up -d
```

7.2 PUERTO 3306 OCUPADO POR MYSQL DEL SISTEMA

Síntoma: El contenedor master no puede iniciarse

```
Error starting userland proxy: listen tcp 0.0.0.0:3306: bind: address already in use
```

Causa: MySQL instalado nativamente en el sistema operativo está usando el puerto 3306.

Solución:

```
# Detener el servicio MySQL del sistema
sudo systemctl stop mysql

# Desactivar inicio automático (opcional)
sudo systemctl disable mysql

# Recargar configuración de systemd
sudo systemctl daemon-reload

# Reintentar levantar contenedores
docker compose up -d
```

7.3 RESOLUCIÓN DE NOMBRES DE HOST FALLIDA

Síntoma: El slave no puede conectarse al master

```
Error: Can't connect to MySQL server on 'mysql-master' (Name or service not known)
```

Causa: Los contenedores no están en la misma red de Docker o la red no tiene DNS interno configurado.

Solución: Añadir la sección `networks` al archivo `docker-compose.yml` (ver sección 5.1) y recrear los contenedores:

```
docker compose down
docker compose up -d
```

7.4 VOLÚMENES Y REDES CORRUPTAS

Síntoma: Comportamiento inconsistente o errores de inicialización.

Solución: Limpiar recursos no utilizados:

```
# Eliminar volúmenes huérfanos
docker volume prune -f

# Eliminar redes no utilizadas
docker network prune -f

# Recrear contenedores desde cero
docker compose down
docker compose up -d
```

7.5 ERROR DE AUTENTICACIÓN DEL USUARIO DE REPLICACIÓN

Síntoma:

```
Slave_IO_Running: Connecting
Last_IO_Error: error connecting to master 'replication_user@mysql-master:3306'
- retry-time: 60 retries: 1 message: Authentication plugin 'caching_sha2_password'
```

Causa: MySQL 8.0 usa por defecto el plugin `caching_sha2_password`, que no es totalmente compatible con clientes de replicación antiguos.

Solución: Recrear el usuario con el plugin `mysql_native_password`:

```
-- En el Master
DROP USER IF EXISTS 'replication_user'@'%';
CREATE USER 'replication_user'@'%'
IDENTIFIED WITH mysql_native_password BY 'Password@54321';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'replication_user'@
'%';
FLUSH PRIVILEGES;
```

Luego reiniciar la replicación en el slave:

```
STOP SLAVE;
START SLAVE;
SHOW SLAVE STATUS\G  -- Verificar que ahora funcione
```

8. COMANDOS DE REFERENCIA RÁPIDA

GESTIÓN DE CONTENEDORES

```
# Iniciar servicios
docker compose up -d

# Detener servicios
docker compose down

# Ver logs en tiempo real
docker logs -f mysql-master
docker logs -f mysql-slave

# Acceder a shell del contenedor
docker exec -it mysql-master bash
docker exec -it mysql-slave bash
```

GESTIÓN DE REPLICACIÓN

En el Master:

```
SHOW MASTER STATUS;                    -- Ver estado del binlog
SHOW SLAVE HOSTS;                      -- Ver slaves conectados
SHOW BINARY LOGS;                      -- Listar archivos binlog
```

En el Slave:

```
SHOW SLAVE STATUS\G                    -- Ver estado completo de replica
ción
START SLAVE;                           -- Iniciar replicación
STOP SLAVE;                            -- Detener replicación
RESET SLAVE ALL;                       -- Resetear configuración
```

9. VENTAJAS Y LIMITACIONES DE LA SOLUCIÓN

9.1 VENTAJAS

Alta disponibilidad: El slave puede promovido a master en caso de fallo
Escalabilidad de lectura: Múltiples slaves pueden distribuir consultas SELECT
Backup en caliente: El slave mantiene una copia actualizada sin detener el servicio

Portabilidad: Docker permite replicar el entorno en cualquier sistema
Aislamiento: Cada servidor corre en su propio contenedor sin interferencias

9.2 LIMITACIONES

Replicación asíncrona: Existe un pequeño delay entre master y slave
No hay failover automático: Requiere intervención manual o herramientas adicionales
Escrituras solo en master: El slave es de solo lectura por defecto
Requiere monitoreo: Los errores de replicación deben detectarse y corregirse

10. CONCLUSIONES

10.1 LOGROS OBTENIDOS

Se implementó exitosamente un sistema de **replicación homogénea MySQL Master-Slave** utilizando contenedores Docker. El sistema cumple con todos los objetivos planteados:

1. Replicación bidireccional funcional verificada con datos reales
2. Sincronización en tiempo real (delay < 1 segundo)
3. Infraestructura portable y reproducible mediante Docker Compose
4. Documentación completa del proceso incluyendo troubleshooting

10.2 APRENDIZAJES CLAVE

- La importancia de configurar correctamente el `server-id` y `binlog-format`
- El uso de plugins de autenticación compatibles en MySQL 8.0
- La necesidad de redes Docker para comunicación entre contenedores por nombre
- El proceso de obtener y aplicar la posición exacta del binlog

10.3 APLICACIONES PRÁCTICAS

Este tipo de arquitectura es fundamental en:

- Sistemas con alta demanda de consultas (separar lectura/escritura)
- Aplicaciones que requieren disaster recovery
- Entornos de desarrollo/testing que necesitan réplicas de producción
- Distribución geográfica de datos para reducir latencia

11. REFERENCIAS BIBLIOGRÁFICAS

- MySQL 8.0 Reference Manual - Replication. Oracle Corporation. **https://dev.mysql.com/doc/refman/8.0/en/replication.html**
- Docker Documentation - Compose Specification. Docker Inc. **https://docs.docker.com/compose/**
- High Performance MySQL, 4th Edition. Schwartz, B., Zaitsev, P., Tkachenko, V. O'Reilly Media. 2021.

ANEXOS

ANEXO A: ESTRUCTURA COMPLETA DEL PROYECTO

```
proyecto-replicacion-mysql/
│
├── docker-compose.yml          # Definición de servicios
├── master-config/
│   └── Master-Server.sql       # Scripts de configuración master
├── slave-config/
│   └── Slave-Server.sql        # Scripts de configuración slave
└── README.md                   # Documentación del proyecto
```

ANEXO B: COMANDOS DE MONITOREO CONTINUO

```
# Monitorear estado de replicación cada 5 segundos
watch -n 5 'docker exec mysql-slave mysql -uroot -pPassword@54321 -e "SHO
W SLAVE STATUS\G" | grep -E "Slave_IO_Running|Slave_SQL_Running|Seconds_B
ehind"'
```