#

Sistema de Alta Disponibilidad PostgreSQL

Grupo 3

Sistemas de Bases de Datos 2

Universidad San Carlos de Guatemala

Primer Semestre 2025

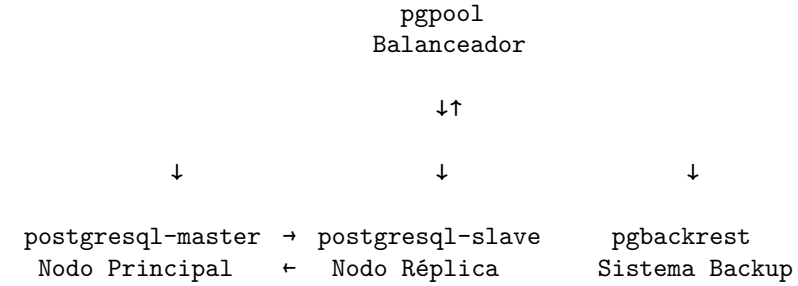
Índice

Arquitectura del Sistema

El sistema implementa una arquitectura robusta de alta disponibilidad para PostgreSQL con las siguientes características:

- **Arquitectura Maestro-Esclavo**: Configuración con un nodo principal y un nodo de réplica
- **Replicación Síncrona**: Garantiza consistencia de datos entre nodos
- **Failover Automático**: Detección de fallos y promoción de nodo esclavo
- **Estrategia de Backups Avanzada**: Respaldos completos, incrementales y diferenciales
- **Balanceo de Carga**: Distribución inteligente de consultas

Diagrama de Arquitectura



Componentes Principales

Componente	Descripción	Tecnología
postgresql-master	Nodo principal que gestiona todas las operaciones de escritura y lectura críticas. Configura el archivado WAL necesario para la replicación y backups.	PostgreSQL 15 + repmgr
postgresql-slave	Nodo secundario que replica en tiempo real el estado del master a través de streaming replication. Disponible para failover automático.	PostgreSQL 15 + repmgr
pgpool	Middleware que proporciona balanceo de carga, pooling de conexiones y detección de fallos para facilitar el failover automático.	pgpool-II
pgbackrest	Sistema avanzado de respaldo específico para PostgreSQL que permite backups full, incrementales y diferenciales con gestión sofisticada de WAL.	pgBackRest
repmgr	Sistema de gestión de replicación que supervisa el estado de los nodos y coordina la promoción de esclavos en caso de fallos.	repmgr 5

Configuración y Despliegue

El sistema completo se despliega mediante Docker Compose, utilizando imágenes personalizadas y configuraciones optimizadas para cada componente:

Estructura de Docker Compose

```yaml
services:
  postgresql-master:
    build:
      context: ./custom-postgres
    image: custom-postgresql-repmgr:15
    container_name: master
    environment:
      - POSTGRESQL_POSTGRES_PASSWORD=postgres
      - POSTGRESQL_USERNAME=replica
      - POSTGRESQL_PASSWORD=replica_pass
      - POSTGRESQL_DATABASE=app_db
      - REPMGR_PASSWORD=replica_pass
      - REPMGR_PRIMARY_HOST=postgresql-master
      - REPMGR_PARTNER_NODES=postgresql-master,postgresql-slave
      - REPMGR_NODE_NAME=master-1
      - REPMGR_NODE_NETWORK_NAME=postgresql-master
      - REPMGR_NODE_ID=1
      - REPMGR_PORT_NUMBER=5432
      - REPMGR_USERNAME=replica
    ports:
      - "5432:5432"
    volumes:
      - postgress_h_postgresql_slave_data:/bitnami/postgresql
      - ./configs/master/postgresql.conf:/custom-configs/postgresql.conf
      - ./configs/master/repmgr.conf:/custom-configs/repmgr.conf
      - ./configs/common/pg_hba.conf:/custom-configs/pg_hba.conf
      - ./scripts/init:/docker-entrypoint-initdb.d
      - ./configs/pgbackrest/pgbackrest.conf:/etc/pgbackrest/pgbackrest.conf:ro
      - ./pgbackrest_repo:/var/lib/pgbackrest
    networks:
      - postgres_net

  postgresql-slave:
    image: bitnami/postgresql-repmgr:15
    container_name: slave
    depends_on:
      - postgresql-master
    environment:
      - POSTGRESQL_POSTGRES_PASSWORD=postgres
      - POSTGRESQL_USERNAME=replica
      - POSTGRESQL_PASSWORD=replica_pass
      - POSTGRESQL_DATABASE=app_db
      - REPMGR_PASSWORD=replica_pass
      - REPMGR_PRIMARY_HOST=postgresql-master
```

```
      - REPMGR_PARTNER_NODES=postgresql-master,postgresql-slave
      - REPMGR_NODE_NAME=slave-2
      - REPMGR_NODE_NETWORK_NAME=postgresql-slave
      - REPMGR_NODE_ID=2
      - REPMGR_PORT_NUMBER=5432
      - REPMGR_USERNAME=replica
    ports:
      - "5433:5432"
    volumes:
      - ./configs/master/postgresql.conf:/custom-configs/postgresql.conf
      - ./configs/master/repmgr.conf:/custom-configs/repmgr.conf
      - ./configs/common/pg_hba.conf:/custom-configs/pg_hba.conf
      - ./scripts/init:/docker-entrypoint-initdb.d
    networks:
      - postgres_net

  pgpool:
    image: bitnami/pgpool:latest
    container_name: pgpool
    environment:
      - PGPOOL_SR_CHECK_USER=replica
      - PGPOOL_SR_CHECK_PASSWORD=replica_pass
      - PGPOOL_POSTGRES_USERNAME=postgres
      - PGPOOL_POSTGRES_PASSWORD=postgres
      - PGPOOL_ADMIN_USERNAME=admin
      - PGPOOL_ADMIN_PASSWORD=adminpass
      - PGPOOL_BACKEND_NODES=0:postgresql-master:5432,1:postgresql-slave:5432
      - PGPOOL_ENABLE_LOAD_BALANCING=yes
    ports:
      - "5431:5432"
    depends_on:
      - postgresql-master
      - postgresql-slave
    networks:
      - postgres_net

  pgbackrest:
    build:
      context: ./pgbackrest
      dockerfile: Dockerfile
    container_name: pgbackrest
    depends_on:
      - postgresql-slave
    environment:
      - PGPASSWORD=replica_pass
      - PGHOST=postgresql-master
```

```
  - PGPORT=5432
volumes:
  - ./configs/pgbackrest/pgbackrest.conf:/etc/pgbackrest/pgbackrest.conf:ro
  - ./pgbackrest_repo:/var/lib/pgbackrest
  - ./backup:/backup
  - ./scripts:/scripts
  - postgress_h_postgresql_slave_data:/bitnami/postgresql
networks:
  - postgres_net
```

Mecanismos de Replicación

El sistema utiliza streaming replication con repmgr para mantener sincronizados los nodos master y slave:

Configuración de repmgr

Los archivos `repmgr.conf` contienen las configuraciones específicas para cada nodo:

Master (`./configs/master/repmgr.conf`): - Define el nodo como primario (ID 1) - Configura los parámetros de conexión - Establece políticas de failover

Slave (`./configs/slave/repmgr.conf`): - Define el nodo como secundario (ID 2) - Apunta al master como origen de replicación - Configura parámetros de promoción automática

Flujo de Replicación

1. Los cambios se escriben primero en el master
2. Los Write-Ahead Logs (WAL) se generan en el master
3. Los WAL se transmiten en tiempo real al slave
4. El slave aplica estos WAL para mantenerse sincronizado

Estrategia de Respaldo y Recuperación

El sistema implementa una estrategia de respaldo robusta usando pgBackRest, que permite tres tipos de backups:

Tipos de Backups Soportados

Tipo	Descripción	Comando	Uso Recomendado
Full	Respaldo completo de toda la base de datos	`./scripts/backup_full.sh`	Semanalmente o tras cambios estructurales importantes
Incremental	Solo respalda los cambios desde el último backup (full o incremental)	`./scripts/backup_incr.sh`	Diariamente para minimizar tiempo y espacio
Diferencial	Respalda todos los cambios desde el último backup full	`./scripts/backup_diff.sh`	Alternativa al incremental cuando se requiere restauración más rápida

Configuración de pgBackRest

El archivo `pgbackrest.conf` contiene la configuración central del sistema de backup:

```
[global]
repo1-path=/var/lib/pgbackrest
repo1-retention-full=2

[demo]
pg1-path=/bitnami/postgresql/data
pg1-port=5432
pg1-host=postgresql-master
pg1-user=replica
```

Scripts de Backup

Los scripts están ubicados en el directorio `./scripts/` y permiten ejecutar de forma sencilla los distintos tipos de backup:

Full Backup (`backup_full.sh`):

```
#!/bin/bash
pgbackrest --stanza=demo --type=full backup
```

Incremental Backup (`backup_incr.sh`):

```
#!/bin/bash
pgbackrest --stanza=demo --type=incr backup
```

Differential Backup (`backup_diff.sh`):

```
#!/bin/bash
pgbackrest --stanza=demo --type=diff backup
```

Proceso de Restauración

El script `restore.sh` facilita la recuperación de datos:

```
#!/bin/bash
# Detener PostgreSQL si está en ejecución
# ...
# Restaurar desde el backup más reciente
pgbackrest --stanza=demo restore
# Reiniciar PostgreSQL
# ...
```

Configuración de Failover Automático

El sistema implementa failover automático a través de la integración de `repmgr` y `pgpool`:

Detección de Fallos

- `repmgr` monitorea constantemente la disponibilidad del nodo master
- Si detecta que el master no está disponible, inicia el proceso de failover

Proceso de Failover

1. `repmgr` promueve automáticamente el nodo slave a nuevo master
2. `pgpool` detecta el cambio y redirige el tráfico al nuevo master
3. Las aplicaciones conectadas continúan funcionando transparentemente

Failback

El script `failback.sh` permite restaurar la configuración original después de un failover:

```
#!/bin/bash
# Este script coordina la vuelta del nodo original como master
# una vez que se ha recuperado después de un fallo

# 1. Registrar el nodo caído como nuevo slave
# 2. Restablecer la replicación
# 3. Actualizar configuración en pgpool
# ...
```

Estructura de Archivos

```
.
  backup                    # Directorio para backups exportados
  backup.ps1                # Script PowerShell para iniciar backups
  configs                   # Archivos de configuración
     common                 # Configuraciones comunes
        pg_hba.conf         # Control de acceso a PostgreSQL
     master                 # Configuraciones del nodo master
        postgresql.conf     # Parámetros de PostgreSQL para master
        repmgr.conf         # Configuración de repmgr para master
     pgbackrest             # Configuración del sistema de backup
        pgbackrest.conf     # Parámetros de pgBackRest
     slave                  # Configuraciones del nodo slave
         postgresql.conf    # Parámetros de PostgreSQL para slave
         repmgr.conf        # Configuración de repmgr para slave
  custom-postgres           # Personalización de imagen PostgreSQL
     Dockerfile             # Definición de imagen personalizada
  docker-compose.yml        # Definición de servicios y dependencias
  failback.sh               # Script para recuperar configuración original
  pgbackrest                # Configuración del contenedor pgBackRest
     Dockerfile             # Definición de imagen pgBackRest
  pgbackrest_repo           # Repositorio de datos para pgBackRest
     archive                # Archivos WAL archivados
     backup                 # Datos de backups
  scripts                   # Scripts de operación
      backup_diff.sh        # Ejecuta backup diferencial
      backup_full.sh        # Ejecuta backup completo
      backup_incr.sh        # Ejecuta backup incremental
      init                  # Scripts de inicialización
         00-override-configs.sh  # Sobrescribe configuraciones iniciales
      restore.sh            # Restaura desde backup
      setup_stanza.sh       # Configura el "stanza" de pgBackRest
```

Guía de Operación

Iniciar el Sistema

```
# Crear la red si no existe
docker network create postgres_net

# Iniciar todos los servicios
docker-compose up -d
```

Verificar Estado de Replicación

```
# Verificar estado de los nodos desde master
docker exec -it master repmgr cluster show

# Verificar lag de replicación
docker exec -it master psql -U postgres -c "SELECT * FROM pg_stat_replication;"
```

Ejecutar Backups

```
# Backup completo
docker exec -it pgbackrest /scripts/backup_full.sh

# Backup incremental
docker exec -it pgbackrest /scripts/backup_incr.sh

# Backup diferencial
docker exec -it pgbackrest /scripts/backup_diff.sh
```

Listar Backups Disponibles

```
docker exec -it pgbackrest pgbackrest info
```

Simular Failover Manual

```
# Detener el master para provocar failover
docker stop master

# Verificar promoción del slave
docker exec -it slave repmgr cluster show
```

Realizar Failback

```
# Una vez recuperado el nodo master original
bash ./failback.sh
```

Restaurar desde Backup

```
# Restaurar al punto más reciente
docker exec -it pgbackrest /scripts/restore.sh
```

Diferencias entre Estrategias de Backup

Característica	Backup Full	Backup Incremental	Backup Diferencial
Tamaño	Mayor (copia completa)	Mínimo (solo cambios desde último backup)	Medio (cambios desde último full)
Tiempo de Ejecución	Mayor	Mínimo	Medio
Tiempo de Restauración	Rápido (un solo archivo)	Lento (último full + todos los incrementales)	Medio (último full + el diferencial)
Complejidad de Restauración	Simple	Compleja	Media
Uso Recomendado	Semanal	Diario	Cada 2-3 días

Desarrollado por Grupo 3 - Universidad San Carlos de Guatemala