Explicación del proyecto

Explicación del Proyecto

El proyecto consistió en la implementación de una base de datos de utilizando el modelo de replicación maestro - esclavo complementado con mecanismos automáticos de respaldo y Redis como sistema de caché en memoria.

El propósito fue simular un entorno empresarial real, donde la base de datos principal (osea el maestro) atiende operaciones de escritura y las réplicas (los esclavos) garantizan la disponibilidad y lectura continua de los datos, incluso ante fallos o mantenimientos de la principal.

Por qué se utilizó maestro - esclavo

Se eligió este modelo por su simplicidad, consistencia y confiabilidad. En entornos como el que se planteó para el proyecto donde la prioridad es mantener la integridad de los datos y reducir el tiempo de recuperación ante fallos, el esquema maestro - esclavo ofrece ventajas como:

- La centralización de escritura: Con esto conseguimos que solo el nodo maestro pueda modificar los datos, lo que evita conflictos de sincronización.
- La escalabilidad de la lectura: los nodos esclavos se configuran para atender peticiones de lectura, distribuyendo la carga y reduciendo la presión sobre el maestro para trabajos de peticiones. Esto mejora el rendimiento general del sistema bajo cargas elevadas, simulando cómo grandes plataformas manejan millones de consultas.
- La tolerancia a fallos: en caso de que el nodo maestro se caiga, un esclavo puede volverse automáticamente maestro mediante failover, garantizando continuidad en el sistema. Una vez restablecido el nodo original, se aplica failback, devolviendo el rol principal sin pérdida de datos y regresando la jerarquia a como estaba antes.
- Simplicidad frente a maestro - maestro: aunque el modelo maestro - maestro permite escritura simultánea, también aumenta la complejidad de resolución de conflictos. En cambio, maestro–esclavo mantiene un sistema más estable, predecible y fácil de monitorear dentro del entorno Dockerizado gracias a su unica escritura por un nodo.

Implementación

1. Se creó un entorno Docker con dos contenedores PostgreSQL, uno actuando como maestro y otro como esclavo).
2. Se configuró la replicación streaming con autenticación mediante replication slots.
3. Se diseñó un script de backups automáticos que genera copias completas, diferenciales e incrementales, registrando cada evento en Redis junto con el tamaño del archivo, la hora y el destino del respaldo.
4. Redis funcionó como un almacenamiento temporal en memoria, donde se guardaban los metadatos de los backups antes de consolidarlos en la base de datos principal.
5. Finalmente, se probaron escenarios de failover y failback, asegurando que la réplica asumiera el control correctamente y se restableciera la sincronización una vez reactivado el nodo maestro.