Failover o conmutación por error.

La replicación es un proceso que consiste en crear y mantener copias idénticas de los servidores de las bases de datos, este proceso se puede realizar de dos formas distintas, cargando los datos y cambios en tiempo real, o determinando un parámetro de un número de estos para realizar el proceso, esto con el propósito de que dichas copias se mantengan sincronizadas y actualizadas. Este proceso garantiza el acceso, la disponibilidad y la integridad de la información de la base de datos en caso de un evento que provoque la desconexión del servidor principal.

Cuando se presentan estos fallos se implementa el proceso de failover o conmutación por error, en donde el servidor espejo toma releva el papel del servidor principal.

Antes de implementar un failover es necesario que el archivo postgresql.conf en el servidor principal tenga las siguientes configuraciones:

```
wal_level = replica
max_wal_senders = 10
synchronous_standby_names = 'servidor_espejo'
```

y por otro lado el archivo postgresql.conf en el servidor espejo debe tener habilitada esta configuración:

```
hot_standby = on
```

Conmutación por error (manual)

`pg_promote()` es una función que puedes llamar desde una sesión de SQL para promover un servidor de réplica a principal.

Tal sentencia se ejecuta desde el servidor espejo de la siguiente manera:

```
SELECT pg_promote();
```

o con psql

```
psql -c "SELECT pg_promote();"
```

Conmutación por error (Automatizada)
Primero es necesario que el archivo **`repmgr.conf`** esté configurado correctamente en ambos servidores.

Configuración en el Servidor Principal:

1. **Archivo** `repmgr.conf`:

Plaintext — Copiar

```
node_id=1
node_name='servidor_principal'
conninfo='host=servidor_principal user=repmgr dbname=repmgr'
data_directory='/ruta/a/data_directory_principal'
log_file='/ruta/a/repmgr_principal.log'
failover='automatic'
promote_command='pg_ctl promote -D /ruta/a/data_directory_principal'
follow_command='repmgr standby follow -f /ruta/a/repmgr.conf --log-to-file'
```

Configuración en el Servidor Espejo:

1. **Archivo** `repmgr.conf`:

Plaintext — Copiar

```
node_id=2
node_name='servidor_espejo'
conninfo='host=servidor_espejo user=repmgr dbname=repmgr'
data_directory='/ruta/a/data_directory_espejo'
log_file='/ruta/a/repmgr_espejo.log'
failover='automatic'
promote_command='pg_ctl promote -D /ruta/a/data_directory_espejo'
follow_command='repmgr standby follow -f /ruta/a/repmgr.conf --log-to-file'
```

una vez bien configurados ejecutamos la siguiente línea en ambos servidores
`repmgrd -f /ruta/a/tu/repmgr.conf`

pg_rewind

Sincroniza un directorio de datos de PostgreSQL con otro directorio de datos que se bifurcó de él.

Básicamente, se copian los bloques modificados de los archivos de relación existentes; logra esto examinando los historiales de la línea de tiempo de los clústeres de origen y destino para determinar el punto en el que divergieron y espera encontrar WAL en el `pg_wal` directorio del clúster de destino que llegue hasta el punto de divergencia.

Así que cuando el servidor de destino se inicia nuevamente, ingresará en la recuperación de archivo y reproducirá todo el WAL generado en el servidor de origen desde el último punto de control antes del punto de divergencia.

Notas:

pg_rewind requiere que el servidor de destino tenga habilitada la opción wal_log_hints `postgresql.conf` o las sumas de comprobación de datos habilitadas cuando se inicializó el clúster con initdb . Ninguna de estas opciones está activada actualmente de manera predeterminada. full_page_writes también debe configurarse en `on`, pero está habilitado de manera predeterminada.

Si pg_rewind falla durante el procesamiento, es probable que la carpeta de datos del destino no esté en un estado que permita recuperarla. En tal caso, se recomienda realizar una nueva copia de seguridad.

stonith (Shoot The Other Node In The Head)

Es un mecanismo utilizado para evitar situaciones en las que ambos sistemas piensan que son el servidor principal, lo que provoca pérdida de datos. Ya que se debe informar al servidor principal anterior que ya no es el principal. STONITH apaga automáticamente un nodo que no funciona correctamente, para que no cause ningún daño.

Referencias

https://www.postgresql.org/docs/current/warm-standby-failover.html

https://www.postgresql.org/docs/current/app-pgrewind.html

https://www.postgresql.org/docs/current/warm-standby-failover.html

https://www.techtarget.com/searchitoperations/definition/STONITH-Shoot-The-Other-Node-In-The-Head

https://cloud.google.com/compute/docs/tutorials/high-availability-linux-pacemaker?hl=es-419#setup-stonith-fence