PREVIRED

05/2016

Replicación

Replicación

- Archivos WAL: PostgreSQL utiliza los denominados ficheros WAL (Write Ahead Log / REDO) para guardar toda la información sobre las transacciones y cambios realizados en la base de datos.
- Los archivos WAL se utilizan para garantizar la integridad de los datos grabados en la base de datos. También se utilizan para reparar automáticamente posibles inconsistencias en la base de datos después de una caída súbita del servidor.
- Estos archivos tienen un nombre único y un tamaño por defecto de 16MB y se generan en el subdirectorio pg_xlog que se encuentra en el directorio de datos ($PGDATA) usado por PostgreSQL.
- El número de archivos WAL contenidos en pg_xlog dependerá del valor asignado al parámetro checkpoint_segments en el fichero de configuración postgresql.conf.
- Los archivos WAL generados en pg_xlog se reciclan continuamente y en un sistema muy ocupado solo tendremos disponibles en pg_xlog los últimos cambios ocurridos en la base de datos durante el periodo de tiempo registrado en los archivos WAL existentes en pg_xlog.
- Los archivos WAL se pueden archivar automáticamente como copia de seguridad ó para usarlos con PITR - Point in Time Recovery. Para activar el archivo automático de ficheros WAL hay que definir los parámetros wal_level, archive_mode y archive_command en postgresql.conf.
- Un fichero WAL se archivará antes que sea reciclado en pg_xlog, pero no antes de que tenga registrado 16MB de información en el mismo.

Replicación

- Archivos WAL: PostgreSQL utiliza los denominados ficheros WAL (Write Ahead Log / REDO) para guardar toda la información sobre las transacciones y cambios realizados en la base de datos.

- Los archivos WAL se utilizan para garantizar la integridad de los datos grabados en la base de datos. También se utilizan para reparar automáticamente posibles inconsistencias en la base de datos después de una caída súbita del servidor.

- Estos archivos tienen un nombre único y un tamaño por defecto de 16MB y se generan en el subdirectorio pg_xlog que se encuentra en el directorio de datos ($PGDATA) usado por PostgreSQL.

Replicación

- Transferencia de registros a nivel de archivos (file-based log shipping): O lo que es lo mismo, transferencia de archivos WAL completos (16MB de registros) entre servidores de bases de datos.

- Transferencia de registros a nivel de registros (record-based log shipping): Transferencia de registros WAL sobre la marcha entre servidores de bases de datos. Esto es lo que hace Streaming Replication.

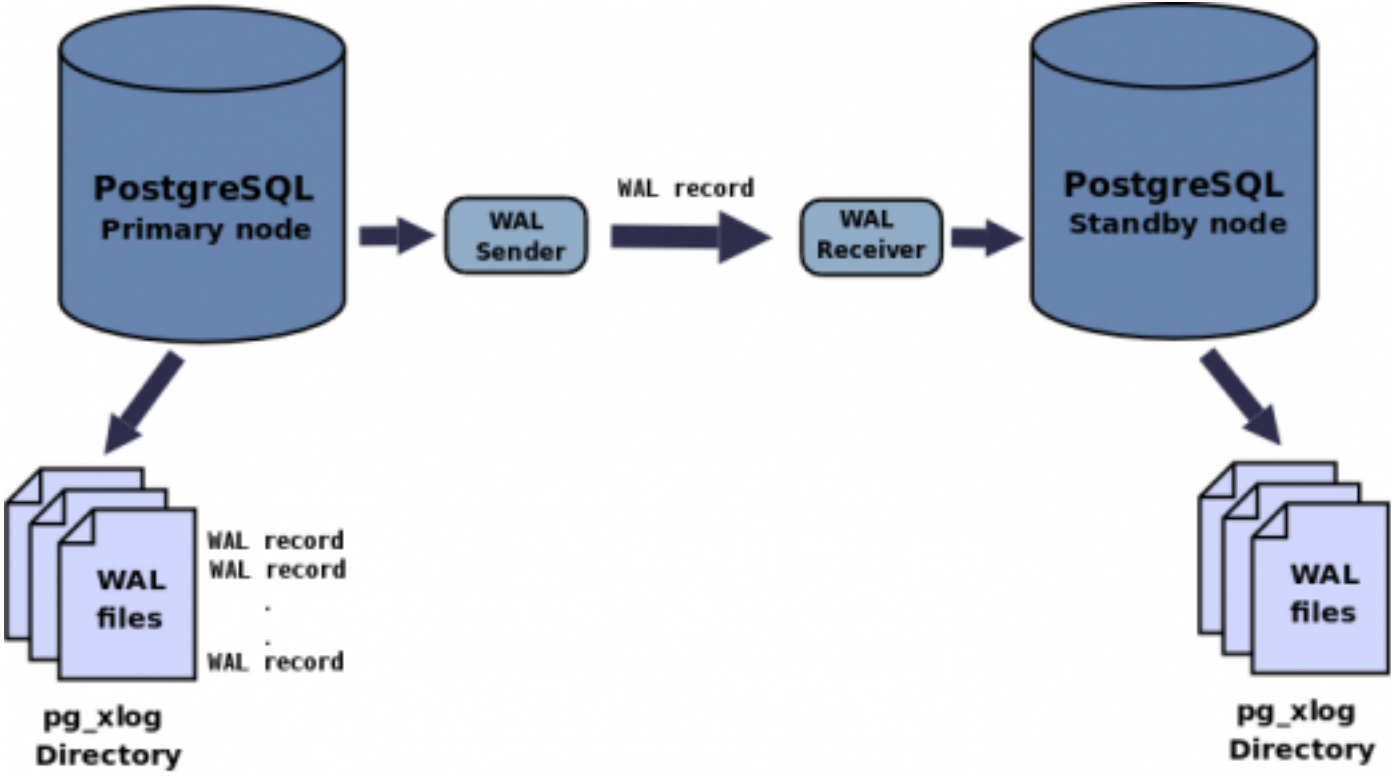
- Replicación/transferencia asincrónica: Cuando los datos se transfieren de un sistema A a otro B, sin esperar por el "acuse de recibo" de B antes de hacer disponibles en A los datos replicados . En un sistema de replicación asincrónico puede existir un cierto retraso ó demora en la disponibilidad de los datos en el sistema esclavo.

- Replicación/transferencia sincrónica: Cuando los datos se transfieren de un sistema A, a otro B, y A espera el "acuse de recibo" de B antes de hacer disponibles en A los datos replicados. En un sistema de replicación sincrónico, todos los datos disponibles en el maestro están disponibles en los esclavos.

Replicación

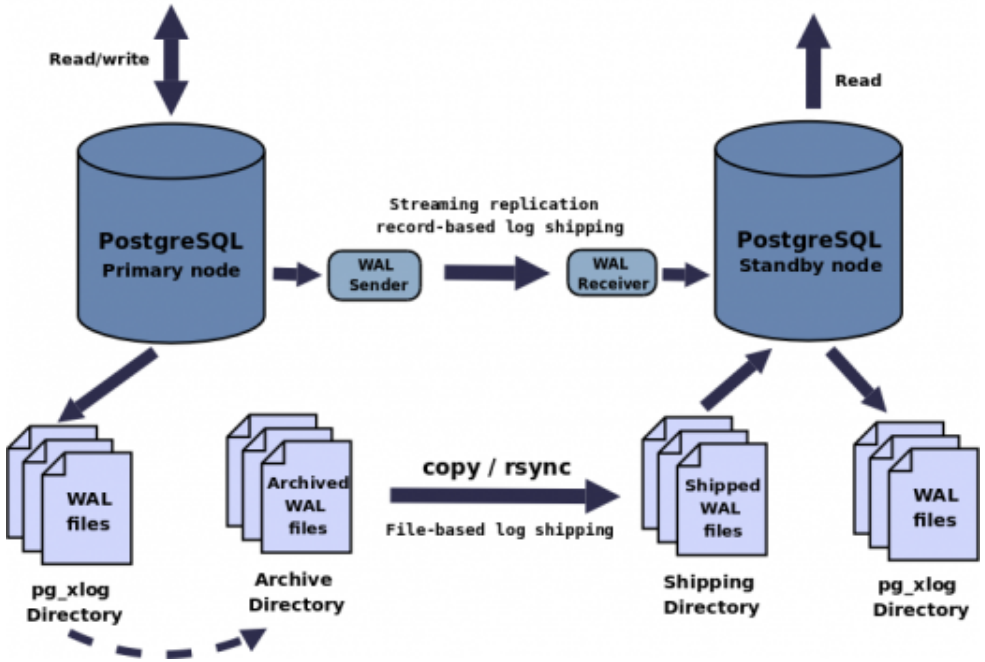
- **Streaming Replication (SR)**
  - –Esta nueva funcionalidad nos permite transferir asincrónicamente registros WAL sobre la marcha (record-based log shipping) entre un servidor maestro y uno/varios esclavos. SR se configura mediante los parámetros primary_conninfo en el fichero recovery.conf y max_wal_senders, wal_sender_delay y wal_keep_segments en postgresql.conf.En la práctica un proceso denominado receptor WAL (WAL receiver) en el servidor esclavo, se conecta mediante una conexion TCP/IP al servidor maestro. En el servidor maestro existe otro proceso denominado remitente WAL (WAL sender) que es el encargado de mandar los registros WAL sobre la marcha al servidor esclavo.

Replicación



Replicación

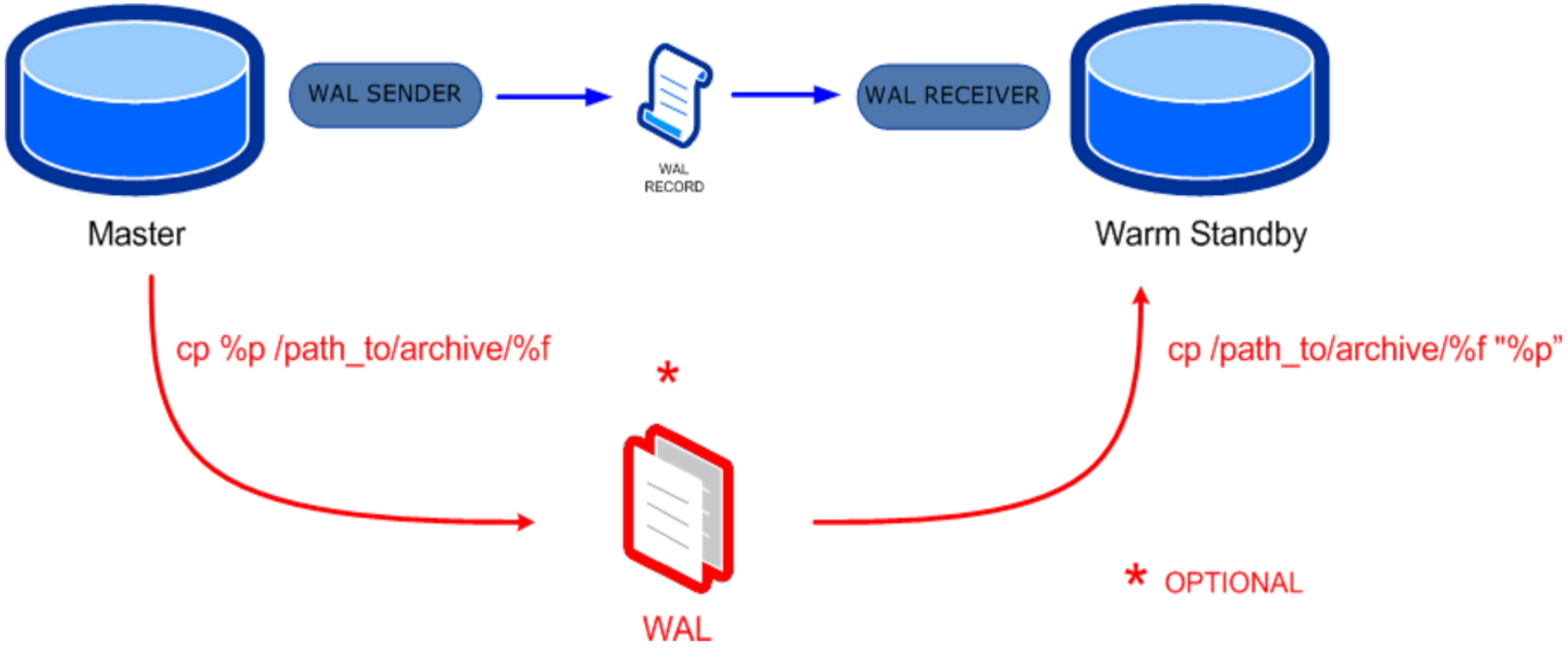
- **HS Hot Standby**
  - –Esta nueva funcionalidad nos permite acceder en modo de solo-lectura a todos los datos disponibles en el servidor esclavo a donde estamos replicando nuestras bases de datos. HS se configura mediante los parametros hot_standby y max_standby_delay en postgresql.conf



Replicación

- **Consideraciones**
  - El tipo de replicación incluido en el núcleo de PostgreSQL está basado en la transferencia de registros WAL entre servidores. Esta transferencia se puede realizar registro a registro (record-based log shipping) ó en ficheros WAL completos (file-based log shipping).Si usamos solamente SR tendremos que tener cuidado que los ficheros WAL en el servidor maestro no sean reciclados antes de ser transferidos al servidor esclavo. Y si transferimos solamente ficheros WAL sin utilizar SR, perderemos las últimas transacciones registradas en el servidor maestro en caso de caida del servidor maestro. Es por esto que para tener un sistema más robusto, se suelen usar los dos métodos conjuntamente.Una replicación basada en la transferencia de registros WAL significa que se replicaran absolutamente todas las bases de datos y cambios que realicemos en el servidor maestro. Si lo que necesitamos es replicar solamente algunas de las bases de datos existentes en el maestro ó algunas tablas, tendremos que usar otro tipo de replicación.

Replicación



Replicación

- **Para configurar un sistema HS usando SR y transferencia de registros WAL, tendremos que realizar lo siguiente:**
  - –Configurar el acceso automático mediante llaves SSH entre el servidor maestro y el esclavo (Opcional)
  - –Configurar uno de los servidores como maestro
  - –Activar el archivo de ficheros WAL en el servidor maestro
  - –Activar la transferencia al servidor esclavo de ficheros WAL archivados en el maestro
  - –Configurar el acceso necesario en el maestro para que el servidor esclavo pueda conectarse via SR
  - –Arrancar el servidor maestro

Replicación

- **Para configurar un sistema HS usando SR y transferencia de registros WAL, tendremos que realizar lo siguiente:**
  - –Realizar una copia de seguridad 'base', mediante el mismo procedimiento que se utiliza con PITR
  - –Restaurar en el servidor esclavo la copia de seguridad 'base' realizada en el maestro
  - –Configurar el servidor esclavo en recuperación continua de registros WAL
  - –Configurar el servidor esclavo como nodo Hot Standby
  - –Crear un fichero recovery.conf en el esclavo
  - –Activar el uso de SR en el esclavo
  - –Activar el uso de ficheros WAL transferidos, en el proceso de restauración
  - –Arrancar el servidor esclavo

Replicación

- **listen_addresses**: Con este parámetro definimos la IP por la que podremos acceder via TCP/IP a postgreSQL.
- **wal_level**: Este parámetro define cuanta información se grabará en los ficheros WAL generados. Se pueden utilizar tres valores, minimal, archive y hot_standby. En nuestro caso utilizamos el valor hot_standby, porque vamos a utilizar esta funcionalidad.
- **archive_mode**: Con este parámetro activamos el archivo de ficheros WAL en el maestro.
- **archive_command**: Con el comando definido en este parámetro, copiamos los ficheros WAL a archivar al directorio /var/pgsql/wal_arch en el servidor maestro y transferimos los ficheros WAL archivados, al directorio /var/pgsql/wal_shipped en el servidor esclavo.
- **max_wal_senders**: Con este parámetro definimos el número máximo de conexiones que se pueden realizar desde servidores esclavos al servidor maestro via SR (1 por servidor esclavo)
- **wal_keep_segments**: Este parámetro define el número máximo de ficheros WAL que mantendremos sin reciclar en el servidor maestro en el caso que SR se retrase en la replicación de datos. Si utilizamos ademas de SR, transferencia de ficheros WAL, este parámetro no es tan importante de configurar.

Replicación

- **postgresql.conf Master**

```
listen_addresses = '10.1.1.100'
wal_level = hot_standby
archive_mode = on
archive_command = '/usr/local/pgsql9.0/bin/archive_wal.sh -P %p -F %f'
max_wal_senders = 5
wal_keep_segments = 10
```

- **El valor de archive_command puede ser más simple**

```
archive_command = 'cp %p ../archive/%f'
# o remoto
archive_command = 'rsync -av %p postgres@192.168.0.10:archive/%f'
```

Replicación

•Archive_wal.sh

```
#!/bin/bash
CHMOD="/bin/chmod”
COPY="/bin/cp”
SCP="/usr/bin/scp”
# Directorio usado por PostgreSQL para generar los ficheros WAL
PG_XLOG_DIR="/var/pgsql/data/pg_xlog”
# Directorio usado por PostgreSQL para archivar los ficheros WAL
PG_WAL_ARCH_DIR="/var/pgsql/wal_arch”
# Servidor PostgreSQL esclavo
STANDBY_SERVER="10.1.1.101”
# Directorio en servidor esclavo donde transferimos los ficheros
# WAL archivados en el servidor maestro.
PG_WAL_SHIPPED="/var/pgsql/wal_shipped”
NO_ARGS=0
E_OPTERROR=65
# ########################################
# ########################################
# Function archive_wal()#
# ########################################
# ########################################
archive_wal(){
if  $COPY -dp $ABSOLUTE_PATH $PG_WAL_ARCH_DIR/$WAL_FILE
then
        $CHMOD 400 $PG_WAL_ARCH_DIR/$WAL_FILE
        $SCP $PG_WAL_ARCH_DIR/$WAL_FILE $STANDBY_SERVER:$PG_WAL_SHIPPED
```

Replicación

•Archive_wal.sh

```
else
        sleep 1
        exit 1
fi
}
# ########################################
# Script invoked with no command-line args?
# ########################################
if [ $# -eq "$NO_ARGS" ]
then
        help
exit
        $E_OPTERROR
fi
# ########################################
# Getting command options
# ########################################
while getopts "P:F:" Option
do
        case $Option in
                P)
                        ABSOLUTE_PATH=$OPTARG;;
                 F)
                        WAL_FILE=$OPTARG;;
        esac
done
shift $(($OPTIND - 1))# ########################################
######################################## Sanity check# ########################################
########################################if [ -z $ABSOLUTE_PATH ]then    echo "Error: Absolute
path not defined"    echo    exit $E_OPTERROR    fiif [ -z $WAL_FILE ]then    echo "Error: WAL
```

Replicación

•Archive_wal.sh

```
shift $(($OPTIND - 1))
# ########################################
# ########################################
# Sanity check
# ########################################
# ########################################
if [ -z $ABSOLUTE_PATH ]
then
        echo "Error: Absolute path not defined"
        echo    exit $E_OPTERROR
Fi
if [ -z $WAL_FILE ]
then
        echo "Error: WAL filename not defined"
        echo    exit
        $E_OPTERROR
fi
archive_wal

exit 0

#
# EOF
```

Replicación

- pg_hba.conf

```
host    replication     all     10.1.1.101 255.255.255.255         trust
```

- Iniciar el servidor maestro

```
/usr/local/pgsql-9.0/bin/pg_ctl -D /var/pgsql/data -l /var/pgsql/data/logserver.log start
```

- Copia de seguridad PITR

```
postgres@server01:~$ /usr/local/pgsql-9.0/bin/psql template1 -c "SELECT pg_start_backup('copia base inicial')”

pg_start_backup
----------------- 0/1000020
(1 row)

postgres@server01:~$ tar -cvf /home/postgres/pg_base_backup.tar /var/pgsql/data/

postgres@server01:~$ /usr/local/pgsql-9.0/bin/psql template1 -c "SELECT pg_stop_backup()"

pg_stop_backup ---------------- 0/10000D8
(1 row)

postgres@server01:~$ scp /home/postgres/pg_base_backup.tar 10.1.1.101:~/tmp/

postgres@server01:~$ ssh postgres@10.1.1.101 "cd / && tar -xvf /tmp/pg_base_backup.tar”

postgres@server01:~$ ssh postgres@10.1.1.101 "rm /var/pgsql/data/postmaster.pid"
```

Replicación

- Otra opción es apagar el maestro y copiar todos los archivos del directorio data a excepión de los directorios pg_log y pg_xlog a los esclavos.
- Es importante que los esclavos tengan los directorios pg_log y pg_xlog, pero vacíos
- Otra opción es la utilida pg_basebackup

Replicación

- **En el esclavo**
- **postgresql.conf**

```
listen_addresses = '10.1.1.101'
hot_standby = on
```

- **recovery.conf**
  - –standby_mode: Este parámetro define que el servidor no saldrá del modo de recuperación y continuará probando la restauración continua de información WAL
  - –primary_conninfo: Este parámetro define el servidor maestro usado por SR para recoger registros WAL
  - –trigger_file: Con este parámetro se define un fichero que en caso de crearse/existir sacará al servidor esclavo del modo "hot standby" y de recuperación continua.
  - –restore_command: Con este parámetro definimos el comando a utilizar, si es necesario, para restaurar los ficheros WAL que se encuentran en /var/pgsql/wal_shipped

```
standby_mode = 'on'
primary_conninfo = 'host=10.1.1.100 port=5432 user=postgres'
# antiguo trigger_file = '/var/pgsql/data/pg_failover_trigger'
trigger_file = 'failover.now'
#SI WAL no es lo suficientemente rápido tenemos un directorio de caché
restore_command = 'cp /var/pgsql/wal_shipped/%f "%p"'
```

Replicación

- **Siempre tratar de iniciar primero los esclavos**
- **Luego el maestro**
- **Para liberar un esclavo:**
  - Crear un archivo en blanco llamado failover.now en la carpeta data del esclavo
  - PostgreSQL completará la reproducción del WAL
  - Renombra el archivo recover.conf a recover.done.
  - Con eso el esclavo está libre
- **Para volver a la “esclavitud” se debe repetir el proceso**

Replicación

- **Tareas de Mantenimiento**
- **Una vez que todo está funcionando, tendremos que mantener el sistema y administrarlo en caso de fallo del servidor maestro.**
- **Las tareas que tendremos que implementar/realizar serán:**
  - Limpiar el directorio donde se archivan los ficheros WAL en el servidor maestro, borrando los ficheros WAL antiguos que no se necesiten.
  - Limpiar el directorio a donde se transfieren los ficheros WAL en el servidor esclavo, borrando los ficheros WAL antiguos que no se necesiten.
  - Activar automáticamente el servidor esclavo como nuevo servidor maestro en caso de fallo del servidor maestro en uso.
  - Monitorear el estado de la replicación para saber el retraso del servidor esclavo en relación al maestro

Replicación

- **Cascading replication**
  - –A partir de la versión 9.2, los servidores esclavos pueden recibir logs de otros esclavosy no tan solo del master. Esto permite replicar. Los esclavos siguen siendo Read-only. Cuando un esclavo recibe y replica se le denomina cascading standby.
- **Remastering**
  - –Es el proceso mediante el cual un esclavo es promovido a maestro. A partir de la versión 9.3 esto se puede hacer con replicación, sin embargo, requiere de reinicio.