**Gestión de WAL**

**FUENTE**

<http://www.postgresql.org.es/node/238>

**CÓDIGO**:

<https://github.com/rafaelma/pitr_scripts>

ANOTACIONES: Cambiando el **archive\_timeout** podemos configurar cada cuanto tiempo queremos realizar la copia del wal.

Usando los script descriptos debajo podremos realizar la gestión de los wal de una manera muy eficiente usando los .sh brindados.

### **Scripts de administración de PITR**

Para administrar PITR vamos a utilizar 3 scripts en BASH, pitr\_basebackup.sh, archive\_wal.sh y archive\_last\_wal.sh. Los podeis grabar en un directoria que este definido en vuestro PATH, por ejemplo /usr/local/bin

Estos 3 scripts han estado en uso en la Universidad de Oslo durante varios años realizando sus trabajo sin problemas. Para que os hagais una idea, durante el último año ejecutamos pitr\_basebackup.sh aproximádamente unas 3.650 veces, archive\_last\_wal.sh unas 5.256.000 y archive\_wal.sh unas 500.000 veces. Muchas de las comprobaciones que hacen son para asegurarse que todo funciona sin problemas y para mandar informes despues de haber descubierto problemas y fallos durante situaciones no planeadas (discos llenos, etc).

Al final de este artículo teneis los tres scripts a modo de ejemplos para que veais como están implementados. Si quereis utilizarlos vais a tener que modificar algunas de las variables para adaptarlos a vuestro sistema. Las versiones de estos scripts que nosotros tenemos en uso som más completas ya que están integradas en nuestro sistema de administración y entre otras cosas graban el estatus de las ejecuciones en una base de datos interna.

A continuación teneis una descripción de las tareas que estos scripts realizan:

#### **archive\_wal.sh**

* Comprobar que la partición para archivar ficheros WAL existe
* Comprobar que la partición donde se generan ficheros WAL existe
* Comprobar que el fichero WAL que se intenta archivar no está ya archivado
* Archivar el fichero WAL definido en la partición para archivar ficheros WAL

#### **archive\_last\_wal.sh**

* Borrar los archivos WAL obsoletos grabados en la partición para archivar el último fichero WAL en uso, residente en el directorio pg\_xlog
* Archivar el último fichero WAL en uso residente en el directorio pg\_xlog y que todavia no se ha archivado

#### **pitr\_basebackup.sh**

* Comprobar que la partición para archivar el fichero con copia de seguridad base existe
* Estimar el espacio necesario para crear copia de seguridad base y parar el script para evitar problemas si estimamos que la partición se va a llenar durante le ejecución de este script.
* Comprobar que no se esta generando una copia de seguridad base
* Registrar el comienzo de la copia de seguridad base con pg\_start\_backup()
* Crear un LVM-snapshot de la partición de datos
* Montar el volumen que contien el LVM-snapshot
* Crear un fichero tar de todo el contenido en el volumen LVM-snapshot
* Desmontar el volumen que contien el LVM-snapshot
* Destruir el contenido del LVM-snapshot
* Registrar el final de la copia de seguridad base con pg\_stop\_backup()
* Archivar el último fichero WAL en uso residente en el directorio pg\_xlog y que todavia no se ha archivado
* Borrar los archivos WAL obsoletos
* Borrar los ficheros con copia de seguridad base obsoletos
* Generar informe de la ejecución

### **Usando los scripts de administración de PITR**

archive\_last\_wal.sh se ejecutará cada minuto por el usuario 'postgres' desde cron. Para ello teneis que actualizar el fichero cron del usuario postgres con esta linea:

\* \* \* \* \* /usr/local/bin/archive\_last\_wal.sh -S *hostname*

archive\_wal.sh se ejecutará automaticamente por postgreSQL cuando lo necesite. Para que esto ocurra tendreis que activar PITR en el fichero postgresql.conf definiendo estas dos lineas:

archive\_mode = on   
archive\_command = '/usr/local/bin/archive\_wal.sh -P %p -F %f -S *hostname*'

pitr\_basebackup.sh se tiene que ejecutar tambien por el usuario 'postgres' desde cron. Dependiendo de lo grande que sea vuestra instalación PostgreSQL y lo mucho que los datos cambien en la misma, tendreis que ejecutar este script más ó menos a menudo. Si dejais pasar mucho tiempo entre cada ejecución y teneis una instalación que genere muchos ficheros WAL, tardareis mucho en restaurar el sistema si teneis que hacer esto alguna vez.

Yo suelo ejecutarlo una vez al dia por la noche. Podeis por ejemplo actualizar el fichero cron del usuario postgres con esta linea:

01 03 \* \* \* /usr/local/bin/pitr\_basebackup.sh -S *hostname* -c t

Como veis, todos estos scripts se ejecutan por el usuario postgres. Por ello tendreis que actualizar el fichero /etc/sudoers con la siguiente linea:

postgres ALL = NOPASSWD: /usr/sbin/lvcreate, /usr/sbin/lvremove,   
/usr/sbin/lvdisplay, /usr/sbin/vgdisplay, /bin/mount, /bin/umount

El usuario postgres tiene que poder conectarse al cluster PostgreSQL via sockets y sin clave. Más información sobre como configurar la cuanta de administrador postgres, se puede encontrar en el artículo [Asegurando la cuenta de administrador "postgres"](http://www.postgresql.org.es/node/224).

### **Restaurar datos**

Si tuvieramos que restaurar los datos de nuestro cluster PostgreSQL a partir de las copias de seguridad realizadas por estos scripts tendriamos que realizar las siguientes operaciones:

* Si existe algún fichero en $PG\_DATA\_PARTITION, borrarlo.
* Si existe algún fichero en $PG\_ARCH\_PARTITION, borrarlo.
* Descomprimir en $PG\_DATA\_PARTITION la *copia de seguridad base* (fichero tar) grabado en $PG\_BACKUP\_PITR\_DATA
* Asegurarse que $PG\_DATA\_PARTITION y $PG\_ARCH\_PARTITION y sus contenidos tienen al usuario postgres como dueño
* Copiar a $PG\_BACKUP\_PITR\_WAL el fichero WAL grabado en $PG\_BACKUP\_PITR\_LAST
* Cambiar el nombre del directorio $PG\_BACKUP\_PITR\_WAL a ${PG\_BACKUP\_PITR\_WAL]\_restore
* Crear un directorio con el nombre $PG\_BACKUP\_PITR\_WAL
* Asegurarse que todos los ficheros WAL en ${PG\_BACKUP\_PITR\_WAL]\_restore tienen permisos de escritura para el usuario postgres
* Borrar los ficheros postmaster.pid y postmaster.opts en $PG\_DATA\_PARTITION
* Crear en $PG\_DATA\_PARTITION un fichero llamado recovery.conf con esta linea:
* restore\_command = 'cp ${PG\_BACKUP\_PITR\_WAL]\_restore/%f %p'
* Cambiar archive\_mode a 'off' y listen\_addresses a '' en el archivo postgresql.conf
* Arrancar PostgreSQL
* Esperar a que termine el proceso de restauración. Podeis comprobar cuando ha terminado cuando estas dos lineas aparezcan en el fichero log de PostgreSQL:
* LOG: archive recovery complete  
  LOG: database system is ready
* Cambiar archive\_mode a 'on' y listen\_addresses a su valor original
* Apagar y arrancar PostgreSQL

Los valores de las variables en esta lista están definidas en pitr\_globalconf.sh.

### **Actualización 10.2011**

* Versiones actualizadas de los scripts se encuentran disponibles en GITHUB
* A partir de la versión 9.0 hay que definir un parametro más (wal\_level) en el fichero postgresql.conf
* wal\_level = archive  
  archive\_mode = on   
  archive\_command = '/usr/local/bin/archive\_wal.sh -P %p -F %f -S *hostname*'
* Un nuevo script pitr\_globalconf.sh contiene todas las funciones y variables globales usadas en el resto de scripts. De esta manera solo hay que modificar este fichero.

### **Scripts**

En GITHUB teneis los tres scripts de los que hablamos en este artículo liberados bajo la licencia GPLv3. Los podeis encontrar en:

<https://github.com/rafaelma/pitr_scripts>

Tendreis que actualizar las variables definidas en pitr\_globalconf.sh para adaptarlas a vuestros sistemas. Es importante que tengais la configuración LVM de vuestro sistema correctamente configurada para que estos scripts funcionen. Mas información en el fichero README que encontrareis en GITHUB.