Procedimientos habituales y tips en PostgreSQL 9.1

Objetivo

- Explicar, ejemplificar algunos conceptos y procedimientos utilizados en Postgresql 9.1 en Linux.
- Enumerar los procedimientos y tips para ser referenciados en otros documentos de esta asignatura.

Requisitos

- Que el alumno comprenda los conceptos de transacción y tenga experiencia en la programación de sistemas de bases de datos.
- Que el alumno comprenda la arquitectura conceptual de un SGBD.
- Que el alumno cuente con conocimientos de Networking, ssh.
- Que el alumno cuente con conocimientos básicos de arquitectura, Client-Server, N-capas.
- Que el alumno cuente con conocimientos básicos de sistemas operativos Unix / Linux: file system, network file system, shell scripts, etc.

Introducción

La implementación de algunas de las “Soluciones de Replicación”[2] en PostgreSQL 9.1 requieren de procedimientos, scripts, etc. que pretenden indicarse en este documento, para luego referenciarlos desde otros apuntes y evitar de esta forma describirlos más de una vez en distintos documentos. PostgreSQL se abreviará como PG. Los procedimientos y conceptos se enumerarán para su correcta identificación. Los procedimientos y tips han sido ejecutados y testeados en Linux Debian Wheezy amd64 kernel versión 3.2.0-4.

1. Instalación PostgreSQL 9.1, con usuario root:

$ aptitude install postgresql postgresql-contrib

o bien, para ser más precisos, porque postgresql es un paquete virtual que apuntaría a la última versión disponible en el repositorio Debian:

$ aptitude install postgresql-9.1
postgresql-contrib-9.1

en caso de haber problemas, desinstalar haciendo:

$ aptitude purge <paquete>

la version 9.1 previamente instalada y luego cualquier otra versión anterior de PG, luego volver a instalar la versión 9.1 como se indicó anteriormente.

Una vez que se ha instalado correctamente PG, se recomienda la instalación del cliente gráfico para la administración de PG:

$ aptitude install pgadmin3

2. Para realizar el arranque/detención/restart servidor PG, con usuario root:

$ /etc/init.d/postgresql start
$ /etc/init.d/postgresql stop
$ /etc/init.d/postgresql restart

otra forma de hacerlo:

$ service postgresql restart

3. Los archivos de configuración PG[1]:

archivo de configuración del cliente:

/etc/postgresql/9.1/main/pg_hba.conf

archivo de configuración del servidor:

/etc/postgresql/9.1/main/postgresql.conf

4. Cambio de la contraseña del usuario administrador de PG, durante la instalación se crea el usuario linux postgres; también existe el usuario postgres dentro del servidor PG, para cambiar la contraseña dentro de PG:

$ su - postgres
$ psql
postgres=# ALTER ROLE postgres WITH ENCRYPTED PASSWORD '<<CLAVE>>';
postgres=# \q

para cambiar la contraseña del usuario linux postgres, sabiendo la contraseña actual, desde el usuario postgres se puede hacer:

$ passwd

1 Los nombres de los archivos son los mismos para las distintas distribuciones Linux, pero la ubicación de los mismos suele ser esta en Debian y Ubuntu, pero no así en otras distribuciones Linux.

o bien, si no conoce la contraseña anterior, puede utilizar el usuario root:

$ su
$ passwd postgres

5. Verificar que el servidor esta corriendo y en qué puerto TCP esta escuchando:

$ netstat -nlp | grep 5432

```
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN
12191/postgres
unix  2      [ ACC ]     STREAM     LISTENING     362596   12191/postgres
/var/run/postgresql/.s.PGSQL.5432
```

se debería obtener una salida similar a la anterior.

6. Permitir el acceso local al servidor vía TCP/IP, modificar la configuración, con usuario root, hacer:

$ nano /etc/postgresql/9.1/main/postgresql.conf

buscar y modificar / descomentar / etc la linea:

listen_addresses = 'localhost'

también pueden agregarse otras direcciones ip o hosts separado por comas e incluso * para permitir conexión TCP/IP desde otras maquinas:

listen_addresses = 'localhost, 192.168.1.25, *'

grabar el archivo de configuración.

6.1 Permitir el acceso de un usuario determinado, a una base de datos determinada, desde una determinada IP:

modificar el archivo de configuración cliente:

$ nano /etc/postgresql/9.1/main/pg_hba.conf

debajo de la linea "# IPv4 local connections:", agregar la linea:

host <<nombre_base_datos>> <<usuario_postgresql>> 192.168.2.3/32 md5

reemplazar <<nombre_base_datos>> por la base de datos que corresponda; reemplazar <<usuario_postgresql>> por el usuario PG que corresponda; reemplazar 192.168.2.3/32 por el rango de IP's que correspondan. La encriptación debe ser md5 cuando no es dentro de la maquina local, para enviar las contraseñas encriptadas.

6.2 Luego de los cambios en la configuración del server y/o cliente, se debe hacer restart del server:

```
$ service postgresql restart
```

o bien

```
$ /etc/init.d/postgresql restart
```

7. Creación un nuevo usuario postgresql:

```
$ su postgres
$ createuser -D -S -R -l <<NUEVO_USUARIO>>
```

este nuevo usuario no puede crear una base de datos (-D), no puede ser superusuario (-S), no puede crear nuevos roles (-R), se puede logear en postgresql (-l). Ver las distintas opciones de createuser.

7.1 Otorgarle la contraseña 123456 al nuevo usuario PG (desde usuario postgres):

```
$ psql postgres
postgres=# ALTER USER <<NUEVO_USUARIO>> WITH ENCRYPTED PASSWORD '123456';
```

7.2 Para ver los usuarios creados:

```
$ psql postgres
postgres=# SELECT usename, passwd FROM pg_shadow;
```

para terminar la sesión psql:

```
postgres=# \q
```

8. Crear una base de datos PG. Usar createdb (siempre logeado con usuario linux postgres):

$ createdb -Ttemplate0 -O <<NUEVO_USUARIO>> -EUTF-8 <<BASE_DE_DATOS>>

crea <<BASE_DE_DATOS>> usando la plantilla template0 siendo el dueño (owner) de la nueva base de datos el usuario <<NUEVO_USUARIO>>, usando el esquema de codificación de caracteres UTF-8.

9. Otorgar todos los privilegios a un usuario PG:

$ su postgres
$ psql postgres
postgres=# GRANT ALL PRIVILEGES ON DATABASE <<BASE_DE_DATOS>> TO <<NUEVO_USUARIO>>;

10. Para ver las bases de datos creadas:

$ su postgres
$ psql postgres
postgres=# SELECT datname FROM pg_database;

11. Para ingresar a una base de datos con un determinado usuario.

11.1 Ingresando localmente, generalmente el administrador (postgres), porque esto implica peer authentication usando unix sockets TCP/IP locales:

$ psql -U <<USUARIO>> -d <<BASE_DE_DATOS>>

11.2 Ingresando remotamente, la mayoría de los usuarios, utilizando md5 authentication, TCP/IP sockets:

$ psql -h <<SERVIDOR>> -U <<USUARIO>> -d <<BASE_DE_DATOS>>

12. Para ejecutar el script script.sql con el usuario postgres:

$ su postgres
$ psql -U postgres -f script.sql
postgres=# exit

13. Ante errores, caídas del servidor, etc, para ver el log (con usuario root):

$ tail /var/log/postgresql/postgresql-9.1-main.log

el comando muestra las últimas líneas de log del servidor; también puede hacer:

$ tail -f /var/log/postgresql/postgresql-9.1-main.log

para quedar dentro de tail, mostrando permanentemente las últimas líneas del log a medida que éste crece.

Cabe aclarar que aquí nos referimos al log del servidor, el cual colecta todos los mensajes del servidor y su actividad, los errores que se generan, etc.; pero nada tiene que ver con el log de transacciones de PG.

14. El comando VACUUM sirve para liberar espacio en general o bien en una tabla determinada dentro de una base de datos PG, también se puede combinar con ANALYZE. Las tuplas borradas o modificadas que ya están fuera de línea, no son borradas de la base de datos, sino que están ocupando espacio que puede ser liberado por este comando, a través de un proceso de limpieza. VACUUM FULL <tabla> copiaría toda la <tabla> en un lugar nuevo y liberaría el espacio que ocupaba antes.

15. El comando ANALYZE recoge información estadística sobre una tabla o columna; la información estadística se almacena en el catalogo del sistema en pg_statistics. El planificador de queries utiliza esta información para determinar el plan de ejecución más eficiente. Si no se indica <tabla>, analiza toda la base de datos, tabla por tabla. Una estrategia común es correr VACUUM y ANALYZE una vez por día, cuando la carga de trabajo del sistema es baja. Aplica locks de read, así que puede correr en forma concurrente con otros queries.

16. WAL=Write-Ahead-Logging=[referencia capítulo 29.2 documentación PG 9.1] se refiere a un método standard para asegurar la integridad de los datos de una base de datos PG. Los cambios sobre las tablas primero se escriben en el log, éste se "flushea"[2] en disco y luego, mas tarde, los cambios son hechos en las tablas. Esto evita que se deba hacer un flush de cada pagina modificada en la base de datos por cada commit. Si el servidor se cae, el mismo se puede recuperar a partir de la información del log, se pueden aplicar sobre las tablas todos los cambios que están en el log y que no fueron aplicados sobre las tablas; a esto se lo llama REDO, roll-forward recovery.

WAL reduce mucho las grabaciones a disco, solo el log necesita grabarse para asegurar el commit de una transacción. El log se graba secuencialmente, con lo cual tiene un costo mucho menor a grabar bloques de la base de datos. El log también hace posible soportar backup on-line y PITR (point-in-time-recovery). Manteniendo los datos históricos del log, partiendo de una copia de la base de datos de algún momento pasado, podemos llevar a la base de datos al punto en el tiempo que querramos, aplicando los registros del log. Incluso, podemos partir de un backup no consistente de la base de datos y aplicando/re-aplicando los

2 Se graba al disco lo más pronto posible, no se hace buffering del mismo.

registros de log, llevaremos a la base de datos a un estado consistente. Físicamente, el log se va grabando en el sub-directorio pg_xlog.

17. “base backup”. Hacer un “base backup” permite clonar un servidor PG, por ejemplo, como carga inicial de un servidor esclavo, se puede hacer un base backup en servidor maestro y transferirlo a servidor esclavo. Copia todo el cluster en un estado consistente:

La función pg_start_backup() fuerza un checkpoint de la base de datos, escribe una etiqueta de "backup" en disco para que ésta pueda ser utilizada como punto de partida para futuras recuperaciones del servidor, a partir de hacer REDO sobre segmentos WAL posteriores a esta marca y lograr alcanzar un estado consistente. pg_start_backup() termina de escribir el segmento WAL actual y fuerza un “switch” por otro nuevo WAL. Se debe esperar a que termine la función pg_start_backup() para comenzar a copiar los archivos del cluster y se debe terminar la copia del cluster antes de invocar a pg_stop_backup().

Procedimiento: start backup - copiar - stop backup; luego transferir copia a servidor esclavo. Con usuario postgres, hacer:

```
$ cd ~
$ psql -c "select pg_start_backup('base_backup');"
$ tar -czvf base_backup.tar.gz /var/lib/postgresql/9.1/main/
$ psql -c "select pg_stop_backup();"
```

en este caso, se utiliza el comando tar para comprimir los datos del cluster, pero también podría usarse otros métodos, como por ejemplo, utilizar rsync para transferir (via ssh) los datos del cluster a otro servidor (IP 192.168.0.2) :

```
$ rsync -av --exclude pg_xlog --exclude postgresql.conf data/* 192.168.0.2:/var/lib/postgresql/data/
```

18. ssh. El traslado de datos entre servidores PG muchas veces debe hacerse a través de un canal de comunicación seguro, cifrado por claves, para ello suele utilizarse ssh. Los servidores deben tener instalado y configurado ssh de forma tal, de poder establecer una comunicación segura entre ellos, basados en la generación de claves públicas y privadas seguras. Suele utilizarse la herramienta rsync, la cual trabaja sobre ssh. La configuración de PG implica escribir scripts o indicar la ejecución de los mismos dentro de los archivos de configuración PG, no es deseable indicar passwords de usuarios dentro de los archivos y rsync no pide usuario y/o contraseña, por lo tanto, debemos configurar ssh para que la comunicación segura entre servidores pueda hacerse sin indicar contraseñas. PG no hace ningún tipo de interacción interactiva entre servidores; solo ejecuta scripts o comandos shell y nada más. A continuación veremos como instalar y configurar ssh para transferir datos desde un servidor origen a un servidor destino. Este instructivo, si deseamos transferir desde A a B y viceversa, debe hacerse en A y en B, ambas veces. Se asume servidor origen en IP 192.168.1.34 y servidor destino en IP 192.168.1.36 [3].

18.1 Instalación de la la implementación de ssh más popular (openssh), con usuario root hacer:

$ aptitude install openssh-server openssh-client

18.2 Instalación rsync, con usuario root, hacer:

$ aptitude install rsync

18.3 Generamos clave ssh en máquina origen, usar el usuario linux que va a utilizar para realizar la transferencia (en el caso de PG, por lo general, se trata de un nuevo usuario linux creado a tal efecto, o bien se trata del usuario linux administrador de PG. No es seguro ni recomendable utilizar el usuario root para esta tarea); en este caso, asumimos utilizar el usuario postgres:

$ su - postgres

$ cd ~

$ pwd

```
/var/lib/postgresql
```

estamos verificando el directorio home del usuario postgres. Ahora verificamos en ambas máquinas (origen y destino), la versión openssh que estamos usando (en ambas máquinas, se debe tratar de la misma versión, caso contrario, hacer la instalación/re-instalación que corresponda para compatibilizar ambas versiones):

$ ssh -V

verificar si existe el archivo oculto .ssh; en tal caso, borre su contenido en ambas máquinas (esto eliminará todas las claves ssh y configuraciones antes almacenadas allí; si ya esta haciendo tranferencias ssh con otras maquinas previamente, no haga esta acción):

$ ls -la
(verificar si existe .ssh, en tal caso, borrarlo con $ rmdir, borrar previamente el contenido de la carpeta con $ rm) una vez borrado .ssh, generamos una nueva clave ssh (siempre utilizando usuario postgres):

$ ssh-keygen

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/var/lib/postgresql/.ssh/id_rsa):
( tipear enter, asuma este archivo)

Enter passphrase (empty for no passphrase): (tipear enter, no ingrese frase
para uso batch)

Enter same passphrase again: (tipear enter nuevamente)

Your identification has been saved in /var/lib/postgresql/.ssh/id_rsa.

Your public key has been saved in /var/lib/postgresql/.ssh/id_rsa.pub.

The key fingerprint is:

a1:5f:8c:ae:7b:8f:0b:6b:7c:21:37:1e:c2:40:26:c6 postgres@debian64

The key's randomart image is:

+--[ RSA 2048]----+

| .               |

|  E o            |

| . +    .        |

|    .  . +       |

|     o. S o      |

|      +o=.       |

|     ..=o+       |

|      o++.       |

|     .++oo.      |

+-----------------+
```

Asumiendo utilizar el usuario postgres en ambas máquinas, asumiendo que es un nueva comunicación ssh (que no hay previas conexiones ssh, sino, en tal caso, se debe agregar la nueva clave al archivo authorized_keys en vez de reemplazarlo como haremos a continuación), con usuario postgres, hacer lo siguiente:

$ scp ~/.ssh/id_rsa.pub postgres@192.168.1.36:~/.ssh/authorized_keys

este comando copia la clave publica del equipo origen en el equipo destino.

Modificar los permisos en la maquina destino, utilizando usuario postgres, en maquina destino, hacer:

$ chmod 755 ~/.ssh

$ chmod 644 ~/.ssh/authorized_keys

Ahora en maquina origen, siempre utilizando usuario postgres, pruebe si puede conectar a la maquina destino sin necesidad de ingresar frase o contraseña:

$ ssh 192.168.1.36

debería ingresar en servidor destino sin ingresar contraseña ni frase ni nada, caso contrario, revisar los pasos anteriores, no puede continuar avanzando sin resolver este problema; ya que, no habrá replicación que funcione, porque el comando rsync no funcionará si el mismo no esta configurado para andar sin
clave.

Habiendo superado la prueba anterior, ahora se puede probar el comando rsync, creamos el archivo prueba.txt y lo transferimos a maquina destino, sin usar contraseñas, siempre utilizando usuario postgres, desde maquina origen, hacer:

$ echo "hola" > prueba.txt

$ cat prueba.txt

```
hola
```

$ rsync -av prueba.txt postgres@192.168.1.36:/var/lib/postgresql

ingresar en la maquina destino, con usuario postgres y verificar que exista en /var/lib/postgresql el archivo prueba.txt transferido, en tal caso, borrarlo. Caso contrario, no puede avanzar, debe resolver el problema de transferencia antes de poder continuar.

19. VirtualBox (VB)[4]. Muchas veces para armar un entorno de desarrollo o prueba de replicación, utilizamos máquinas virtuales dentro de una misma máquina física, para ello existe una herramienta gratuita que podemos utilizar: VirtualBox. Llamamos máquina host o anfitrión a la máquina que tiene instalado Virtualbox y maquina virtual (MV) o máquina huésped a la máquina virtualizada utilizando VB. Aquí enumeramos algunos conceptos y tips sobre VB:

19.1. Configuración de red

19.1.1 Adaptador conectado a NAT**

Es la mejor forma para que la MV pueda ver una red externa. Es decir, con NAT puedo hacer ping
a la ip de la maquina host, pero la maquina host no puede ver la MV. Es ideal para navegar por la
web, etc usando la conexión de la maquina host, no requiere ninguna configuración de la MV.

19.1.2 Adaptador conectado a red NAT (net nat)

Requiere de configurar y levantar servicio de red usando comando VBoxManage natnetwork ; hace
trabajar a la MV como si fuera el router de nuestra casa, permite que los servidores dentro de la
misma se comuniquen, también salir fuera la misma, via tcp/ip ; previene que sistemas externos a la MV ingresen
a la misma.

19.1.3 Adaptador conectado a puente (bridge)

Para simulaciones de red mas avanzadas, correr servidores dentro de la MV, cuando arranca la MV,
VB conectara la MV a una de las placas de red del host como si fuera otra maquina, “bypaseando” al host y obteniendo su propia ip dentro de la misma red del host. La MV y el host serán visibles en la red. Conectar la interfase host (eth0, wlan0, etc) que corresponda para que la MV pase a través de ella para obtener su ip[3].

19.1.4 Red interna

Las MV's pueden comunicarse entre ellas, dependientes del mismo host, no hay comunicación con el
exterior. Poniendo IP's estáticas en las MV's se ha podido comprobar la conexión entre MV's pero no

3 En general, no se han encontrado mayores inconvenientes utilizando esta opción muy utilizada para nuestros propósitos. No obstante, verifique que la MV obtenga correctamente su IP dentro del rango de las IP's del anfitrión; caso contrario, es posible que se trate de una restricción de seguridad de su red, consulte con el administrador de red. Pruebe la MV en otra red y verifique si obtiene su IP correctamente.

entre cualquiera de ellas y la maquina host. Tampoco es posible utilizar la conexión de internet de
la maquina host.

19.1.5 Adaptador sólo anfitrión (host-only networking)

No se require adaptador físico, la MV no puede salir fuera del host, VB crea software interfase
en el host, se puede usar el dhcp server incluido en VB, se pueden crear N host-only networks.

19.2 Montar carpetas compartidas entre maquina host linux y huésped linux
:

19.2.1 Crear carpeta compartida en VB, supongamos nombre de carpeta compartida "descargas" con ruta /home/grchere/Descargas

19.2.2 En máquina huésped, con usuario root, para montar carpeta compartida:

$ mkdir /media/Descargas

se crea carpeta Descargas para montar allí la carpeta /home/grchere/Descargas , luego montar como:

$ mount -t vboxsf descargas /media/Descargas

si desea montar esta carpeta en forma permanente, deberá editar el archivo /etc/fstab utilizando el usuario root (deberá asegurarse de que la carpeta compartida estará siempre disponible desde VB, si exporta la MV deberá indicar que, previo a su uso, se deberá crear en VB la carpeta "descargas"); por ejemplo, montar el recurso descargas en /media/Descargas, y que nuestro usuario se llama grchere y pertenece al grupo grchere. Si queremos utilizar los permisos por defecto de Debian/Ubuntu, tenemos que especificar al montador de VB las siguientes opciones:

```
uid=grchere
gid=grchere
dmode=775
fmode=664
```

editamos /etc/fstab (utilizando usuario root):

$ su
$ nano /etc/fstab

agregamos la siguiente línea al final, recalcamos que las opciones se separan con comas:

```
descargas /media/Descargas vboxsf
uid=grchere,gid=grchere,dmode=775,fmode=664 0 0
```

guardamos y listo. Después de reiniciar, debería de aparecer montada nuestra carpeta descargas automáticamente.
Cada vez que Ud. copie archivos desde VB, es posible que requiera hacer un "refresh" desde el navegador de archivos de Linux (nautilus) para ver los cambios o nuevos archivos copiados.

Referencias

[1] PostgreSQL 9.1.14 Documentation, disponible en http://www.postgresql.org/docs/9.1/static/
[2] Apunte de Asignatura 11078, Base de Datos II, UNLu, "Soluciones de Replicación de PostgreSQL 9.1", Cherencio, G., disponible en http://www.grch.com.ar/docs/bdd/apuntes/unidad.iii/11078-Soluciones%20de%20Replicacion.pdf
[3] http://www.thegeekstuff.com/2008/06/perform-ssh-and-scp-without-entering-password-on-openssh/
[4] https://www.virtualbox.org/manual/ch06.html

Atte. Guillermo Cherencio.
11078 BD II UNLu