Para configurar SIGENU

1. Instalando PostgreSql en CentOS 6.2 / 6.3

La instalación de PostgreSql y el soporte para php en CentOS es bastante fácil. Lo dividiremos en los siguientes pasos:

1. Instalación de los paquetes.

2. Cambiando contraseñas de PostgreSql.

3. Abriendo el cortafuego.

4. Activando el servicio al inicio.

 1. Instalación de los paquetes

$ sudo yum -y install postgresql postgresql-server php-pgsql

2. Cambiando contraseñas de PostgreSql

$ sudo passwd postgres

Aquí escribimos la contraseña nueva

Una vez cambiada la contraseña del usuario postgres debemos acceder a la línea de comandos de la base de datos para cambiar la contraseña del usuario postgres de la base de datos. Pero antes debemos iniciar la base de datos. Lo hacemos con el siguiente comando:

$ sudo service postgresql initdb

Iniciando la base de datos: [ OK ]

$ sudo service postgresql start

Iniciando servicios postgresql:

Una vez iniciado el servicio de la base de datos de postgresql nos conectamos a la base de datos a través del usuario postgres del sistema y una vez dentro cambiamos la contraseña de acceso del usuario postgres de la base de datos. Primero cambiamos al usuario postgres y después ejecutamos la consola de la base de datos PostgreSql.

$ su postgres

Contraseña:

bash-4.1

$ psql

Ya una vez dentro de la consola debemos ejecutar el comando que cambiará la contraseña del usuario postgre y que es el siguiente.

postgres=# ALTER ROLE postgres WITH PASSWORD 'sigenu\*postgres';

ALTER ROLE postgres=

Evidentemente, habrá que sustituir password por la contraseña que quieras utilizar con dicho usuario de la base de datos. Cuando la orden se ha ejecutado correctamente la consola devuelve ALTER ROLE tal como se puede ver en el código de arriba. Para salir de la consola psql y de la sesión del usuario postgre ejecutamos los siguientes comandos

postgres=# \q

bash-4.1

$ exit

Ya hemos cambiado las contraseñas.

Nota: Por defecto el servidor de base de datos escucha todas las conexiones a través del puerto 5432. Si se desea conﬁgurar éste y otros parámetros de la propia base de datos, sólo hay que editar el ﬁchero de conﬁguración situado en /var/lib/pgsql/data/postgresql.conf.

Una vez instalado el servidor reiniciamos el servidor apache con el comando:

$ sudo /etc/init.d/httpd restart

Parando httpd: [ OK ]

 Iniciando httpd: [ OK ]

Para comprobar si el módulo de PostgreSql para PHP se ha instalado correctamente el módulo con extensión .so debe estar listado en el siguiente directorio siguiente directorio /usr/lib/php/modules, lo que podemos comprobar con el siguiente comando

$ sudo ls -la /usr/lib/php/modules/ | grep pgsql

 -rwxr-xr-x. 1 root root 32424 jul 3 18:44 pdo\_pgsql.so

 -rwxr-xr-x. 1 root root 111220 jul 3 18:44 pgsql.so

Además el módulo debe aparecer listado cuando mostramos un phpinfo();

3. Abriendo el cortafuego

Ahora para permitir conexiones remotas al servidor hay que decirle a PostgreSql que queremos que escuche conexiones desde cualquier ip y que queremos que permita la autenﬁcación de todos los usuarios de la base de datos desde cualquier ip. conexiones desde cualquier.

Para que escuche conexiones desde cualquier dirección ip debemos ejecuat el siguiente comando para editar el archivo de conﬁguración

$ sudo vim /var/lib/pgsql/data/postgresql.conf

$ sudo nano /var/lib/pgsql/data/postgresql.conf

Y descomentar y cambiar la línea 59 dejándola tal como sigue:

listen\_addresses = '\*' # what IP address(es) to listen on

Ahora indicamos los usuarios a los que queremos permitir el acceso al servidor postgre y sus bases de datos. Lo hacemos editando el siguiente ﬁchero.

$ sudo vim /var/lib/pgsql/data/pg\_hba.conf

$ nano vim /var/lib/pgsql/data/pg\_hba.conf

Al ﬁnal del ﬁchero, cambiamos todos los métodos de autenﬁcación por md5 lo que le dará un poco de más seguridad y añadimos la última línea, quedando esa parte del ﬁchero tal como sigue

# TYPE DATABASE USER CIDR-ADDRESS METHOD

# "local" is for Unix domain socket connections only

local all all md5

# IPv4 local connections:

host all all 127.0.0.1/32 md5

# IPv6 local connections:

host all all ::1/128 md5

#Para todos los hosts.

host all all 0.0.0.0 0.0.0.0 md5

La última línea hace que el servidor PostgreSql acepte todos los usuarios para todas las bases de datos desde todas las ips posibles. Evidentemente esto se puede modiﬁcar a nuestro antojo permitiendo sólo conexiones desde un rango de ips, para un usuario en concreto, para una base de datos en concreto, etc. En el caso que nos ocupa como el destino del servidor es para un servidor de desarrollo lo dejaremos tal como aparece arriba aunque podríamos haberlo limitado a que se conecten únicamente ips de la red local.

Ahora activaremos el servicio de postgresql para que se arranque durante el inicio del sistema.

4. Activando el servicio al inicio

Primero comprobamos si está ya activado para que arranque al inicio con el siguiente comando:

$ sudo chkconfig --list | grep postgre

Postgresql 0:desactivado 1:desactivado 2:desactivado 3:desactivado 4:desactivad

Como evidentemente no lo estará, ya que acabamos de instalar la base de datos Postgre, lo activamos con el siguiente comando:

$ sudo chkconfig postgresql on

Ya podemos volver a comprobar si ya está activo con el primer comando.

Conclusión

La instalación de PostgreSql, como podeis ver es bastante sencilla y no tiene demasiadas complicaciones en CentOS. Una buena opción para administrar las bases de datos, permisos, roles y etc… de nuestro servidor postgre es mediante la aplicación pgAdmin III, dicha aplicación la , dicha aplicación la instalaremos en las máquinas desde las cuales administraremos el servidor postgre. Se puede instalar con el siguiente comando:

$ sudo yum install pgadmin3