−Table des matières

Installation de PostgreSQL

Cet article, écrit par Guillaume Lelarge, a été publié dans le hors-série 44 du magazine GNU/Linux Magazine France, hors-série dédié à PostgreSQL. Il est disponible maintenant sous licence Creative Commons.

Installer PostgreSQL n'est pas compliqué. Que ce soit par les sources ou par un paquet, c'est généralement une question d'une dizaine de minutes. Nous allons voir comment l'installer à partir des sources, à partir du paquet Debian et à partir d'un paquet RPM.

Installer à partir des sources

Il existe un paquet .tar.gz et un paquet .tar.bz2 pour chaque version disponible. Tout se trouve dans http://wwwmaster.postgresql.org/download/mirrors-ftp/source/. Si vous y jetez un œil, une fois un miroir sélectionné, vous verrez qu'il existe un répertoire pour les différentes versions disponibles. Si celle que vous cherchez ne s'y trouve, allez sur ftp://ftp-archives.postgresql.org. Toutes les versions, depuis la 1.0.8, s'y trouve pas. Néanmoins, en ce qui nous concerne, pas besoin d'y aller, nous allons installer la dernière version disponible au moment de l'écriture de cet article, c'est-à-dire la 8.4.0.

PostgreSQL a une particularité très sympathique : il refuse de s'exécuter en tant que root. Il faut nécessairement un utilisateur qui n'est pas un superutilisateur pour l'exécuter. Pour un vrai serveur, nous commencerions donc par créer un utilisateur simple, dont le nom sera généralement postgres par simplicité, mais tout autre nom ira bien. Notez que si vous ne pouvez pas créer un utilisateur, vous pouvez très exécuter le serveur PostgreSQL avec votre propre utilisateur personnel. C'est d'ailleurs ce que nous allons faire.

Autre point important. Pour compiler PostgreSQL, vous avez besoin d'un certain nombre de logiciels :

- la commande make;
- le compilateur C gcc;
- bison;
- flex;
- le paquet développement de la bibliothèque readline;
- le paquet développement de la bibliothèque zlib.

Vous pourriez avoir besoin de plus si vous activez certaines options, par exemple l'outil d'internationalisation des applications, gettext.

Tout ceci va vous donner la ligne suivante sous Debian:

```
aptitude install make gcc bison flex libreadline-dev libz-dev gettext
```

Comme nous n'avons pas à créer d'utilisateur, nous pouvons dès maintenant récupérer les sources à partir du compte qui exécutera le serveur PostgreSQL. Il suffit d'appeler wget avec la bonne URL pour récupérer les sources :

```
guillaume@debian1:~$ wget http://ftp4.fr.postgresql.org/pub/mirrors/postgresql/source/v8.4.0/postgresql-8.4.0.tar.bz2
--2009-07-25 19:17:06--  http://ftp4.fr.postgresql.org/pub/mirrors/postgresql/source/v8.4.0/postgresql-8.4.0.tar.bz2
Résolution de ftp4.fr.postgresql.org... 134.157.176.20
Connexion vers ftp4.fr.postgresql.org|134.157.176.20|:80... connecté.
requête HTTP transmise, en attente de la réponse... 200 OK
Longueur: 13839282 (13M) [application/x-bzip2]
Saving to: `postgresql-8.4.0.tar.bz2'

100%[======================================================>] 13 839 282   686K/s   in 25s

2009-07-25 19:17:31 (544 KB/s) - « postgresql-8.4.0.tar.bz2 » sauvegardé [13839282/13839282]
```

Notez la toute petite taille de ce paquet. Les sources, évidemment compressées, ne font que 14 Mo. De nos jours, c'est vraiment un tout petit paquet.

Décompressons l'archive et déballons tous les fichiers :

```
guillaume@debian1:~$ tar xfj postgresql-8.4.0.tar.bz2
guillaume@debian1:~$ ll postgresql-8.4.0
total 1692
-rw-r--r--  1 guillaume guillaume    445 2004-04-23 20:15 aclocal.m4
drwxr-xr-x  2 guillaume guillaume   4096 2009-07-25 19:21 config
-rwxr-xr-x  1 guillaume guillaume 801991 2009-06-27 02:14 configure
-rw-r--r--  1 guillaume guillaume  61116 2009-06-27 02:14 configure.in
```

```
drwxr-xr-x 41 guillaume guillaume   4096 2009-07-25 19:21 contrib
-rw-r--r--  1 guillaume guillaume   1192 2009-01-01 18:23 COPYRIGHT
drwxr-xr-x  3 guillaume guillaume   4096 2009-07-25 19:21 doc
-rw-r--r--  1 guillaume guillaume   4331 2009-01-15 02:53 GNUmakefile.in
-rw-r--r--  1 guillaume guillaume 727455 2009-06-28 00:53 HISTORY
-rw-r--r--  1 guillaume guillaume  78427 2009-06-28 00:53 INSTALL
-rw-r--r--  1 guillaume guillaume   1423 2007-01-20 18:16 Makefile
-rw-r--r--  1 guillaume guillaume   1287 2008-05-07 00:02 README
drwxr-xr-x 14 guillaume guillaume   4096 2009-07-25 19:21 src
```

Le contenu est tout à fait standard : un script configure pour créer le fichier Makefile, un répertoire doc pour la documentation, un répertoire src pour les sources, les fichiers habituels (COPYRIGHT, HISTORY, INSTALL, README). Bref, rien de transcendant. Seule petite nouveauté, le répertoire contrib qui contient tous les modules optionnels qui ne font pas (encore ?) partie du cœur de PostgreSQL.

L'étape suivante est le configure. Cette étape permet, comme son nom l'indique, de configurer le fichier Makefile en prenant en compte les spécificités de votre serveur. Ce script dispose de différentes options permettant d'activer ou de désactiver certaines fonctionnalités, voire de modifier la valeur de certains paramètres. Les options les plus importantes sont indiquées dans le tableau 1.

Tableau 1 : options essentielles de configure.

Option	Signification
–prefix	permet de préciser le répertoire d'installation de PostgreSQL
–with-openssl	active l'utilisation d'OpenSSL
–with-libxml	active l'utilisation de la bibliothèque libxml
–with-libxslt	active l'utilisation de la bibliothèque libxslt
–enable-integer-datetimes	utilise des nombres entiers pour représenter les dates et les heures (en 8.4, cette option est activée par défaut)
–enable-nls	active les langues (éventuellement précisées) utilisables par les applications PostgreSQL

Lançons l'opération configure en précisant le répertoire d'installation et en activant la langue française.

```
./configure --prefix=/home/guillaume/postgresql --enable-nls=fr
```

Une fois cette opération terminée avec succès, il faut lancer la compilation:

```
guillaume@debian1:~$ make
```

Si tout se passe bien, le texte suivant doit apparaître tout à la fin:

```
All of PostgreSQL successfully made. Ready to install.
```

Enfin, il faut installer le logiciel :

```
guillaume@debian1:~$ make install
```

Attention, vous pourriez avoir besoin des droits administrateur pour la copie des fichiers si le répertoire d'installation se trouve à un endroit où vous n'avez pas le droit d'écrire. Dans ce cas, il faudra exécuter la commande suivante avec sudo, ce qui donne la ligne suivante:

```
guillaume@debian1:~$ sudo make install
```

Si tout se passe bien, vous devriez voir le texte suivant tout à la fin:

```
PostgreSQL installation made.
```

À ce moment-là, nous disposons des exécutables et des fichiers nécessaires à l'exécution d'un serveur PostgreSQL. Pour informations, tout ceci pèse seulement 31 Mo sur votre disque, dont 13 Mo de documentation. Les sous-répertoires sont là-aussi très habituels : bin pour les binaires, lib pour les bibliothèques, share pour les différents fichiers de données (comme les fichiers exemples de configuration, la documentation, les pages man, etc.) Le tableau 2 récapitule les exécutables disponibles et leur utilisation.

Tableau 2 : liste des exécutables de PostgreSQL

Exécutable	Intérêts
clusterdb	exécute l'instruction SQL CLUSTER
createdb	ajoute une base de données
createlang	ajoute un langage de procédures dans une base de données
createuser	ajoute un utilisateur
dropdb	supprime une base de données
droplang	supprime un langage de procédures
dropuser	supprime un utilisateur
ecpg	préprocesseur de SQL embarqué dans un code source en C
initdb	initialise une instance PostgreSQL
pg_config	renvoie une liste de paramètres indiquant la configuration des exécutables
pg_ctl	contrôle le moteur de la base de données (démarrage, arrêt, statut, rechargement de la configuration)
pg_dumpall	sauvegarde de l'instance complète (donc toutes les bases de données, mais aussi les objets globaux)
pg_restore	restaure une base de données d'après une sauvegarde
pg_controldata	affiche les informations de contrôle du cluster
pg_dump	sauvegarde une base de données
pg_resetxlog	supprime les journaux de transactions
postgres	programme responsable de la discussion entre un client et le moteur de base de données
postmaster	de nos jours, un simple lien vers l'exécutable postgres ; auparavant le processus père de tous les processus d'une instance PostgreSQL
psql	console interactive (permet d'exécuter des requêtes SQL, de faire de la maintenance, de récupérer la liste des objets d'une base et leurs définitions)
reindexdb	exécute l'instruction SQL REINDEX
vacuumdb	exécute l'instruction SQL VACUUM

Il manque encore deux éléments pour terminer l'installation : l'initialisation de l'instance et l'ajout du script de démarrage.

Avant de pouvoir créer une base de données et y travailler, il faut tout d'abord créer une instance. Certains parlent aussi de cluster. Tout simplement, nous allons demander à PostgreSQL de créer les fichiers nécessaires au démarrage du serveur dans un répertoire que nous lui aurons préalablement indiqué. Ce répertoire peut être précisé soit en le donnant en argument de la commande initdb (avec l'option -D) soit en initialisant la variable d'environnement

PGDATA. Il est préférable de passer par ce deuxième cas, en prenant soin d'initialiser cette variable par un script de connexion (.profile, .bash_profile ou .bashrc pour les plus courants) de l'utilisateur qui exécute le serveur PostgreSQL. Préférable car cette variable est utilisée notamment par l'exécutable pg_ctl ainsi que par pg_controldata.

```
guillaume@debian1:~$ cat <<_EOF_ >>~/.bashrc
> export PGDATA=/home/guillaume/postgresql-data
> export PATH=/home/guillaume/postgresql/bin:\$PATH
> _EOF_
guillaume@debian1:~$ source ~/.bashrc
guillaume@debian1:~$ initdb
```

Voici l'affichage qui s'ensuit avec une explication pour chaque partie :

```
Les fichiers de ce cluster appartiendront à l'utilisateur « guillaume ».
Le processus serveur doit également lui appartenir.
```

Comme j'ai exécuté initdb en tant qu'utilisateur guillaume, tous les fichiers auront cet utilisateur comme propriétaire. Le serveur PostgreSQL devra aussi être exécuté par cet utilisateur.

```
Le cluster sera initialisé avec la locale fr_FR.UTF-8.
L'encodage par défaut des bases de données a été configuré en conséquence
avec UTF8.
```

L'instance utilise la locale par défaut du système (qui est surchargeable avec l'option -E). Les nouvelles bases de données seront aussi en UTF-8, sauf si l'utilisateur choisit de changer cela par un autre encodage qui soit compatible avec la locale UTF-8.

```
La configuration de la recherche plein texte a été initialisée à « french ».
```

La locale étant du français UTF-8, le français est utilisé comme configuration par défaut de la recherche par racine native.

```
création du répertoire /home/guillaume/postgresql-data... ok
création des sous-répertoires... ok
```

Les répertoires sont créés, le premier étant celui pointé par la variable PGDATA.

```
sélection de la valeur par défaut de max_connections... 100
sélection de la valeur par défaut pour shared_buffers... 28MB
création des fichiers de configuration... ok
```

Les fichiers de configuration sont ajoutés au répertoire de l'instance. Remarquons aussi que certaines variables d'un fichier de configuration sont testées pour s'adapter au matériel et au système d'exploitation du serveur.

```
création de la base de données template1 dans /home/guillaume/postgresql-data/base/1... ok
```

La première base de données, template 1, est créée.

```
initialisation de pg_authid... ok
initialisation des dépendances... ok
création des vues système... ok
chargement de la description des objets système... ok
création des conversions... ok
création des dictionnaires... ok
initialisation des droits sur les objets internes... ok
création du schéma d'informations... ok
```

Un certain nombre d'objets globaux et de catalogues systèmes sont créés.

```
lancement du vacuum sur la base de données template1... ok
copie de template1 vers template0... ok
copie de template1 vers postgres... ok
```

Après un VACUUM sur la première base, les bases template0 et postgres sont créées en copiant la base initiale.

```
ATTENTION : active l'authentification « trust » pour les connexions
locales.
Cela peut être modifié par l'édition de pg_hba.conf ou en utilisant l'option
-A au prochain lancement d'initdb.
```

L'authentification trust est activée pour les connexions locales, ce qui signifie que n'importe qui peut se connecter au serveur PostgreSQL sans avoir à connaître de mot de passe. Heureusement, cela n'est vrai que pour les connexions locales (par socket de domaine Unix et par TCP/IP via l'adresse IP 127.0.0.1).

```
Succès. Vous pouvez maintenant lancer le serveur de bases de données par :

    postgres -D /home/guillaume/postgresql-data
ou
    pg_ctl -D /home/guillaume/postgresql-data -l journal_applicatif start
```

Nous sommes arrivés à la fin du message d'initdb. Il indique le succès de l'opération d'initialisation et précise comment lancer le serveur PostgreSQL.

L'option -D permet d'indiquer le répertoire de l'instance. Cela n'est pas nécessaire si la variable d'environnement PGDATA a bien été positionnée. Nous pouvons donc lancer la commande :

```
guillaume@debian1:~$ pg_ctl start
serveur en cours de démarrage
LOG:  le système de bases de données a été arrêté à 2009-07-26 09:15:50 CEST
LOG:  lancement du processus autovacuum
LOG:  le système de bases de données est prêt pour accepter les connexions
```

Parfait, PostgreSQL démarre sans problème.

Il ne reste plus qu'à écrire un script de démarrage. En fait, il en existe un déjà écrit dans les modules contrib de PostgreSQL. Il suffit de le copier et d'ajuster certains paramètres pour qu'il soit fonctionnel.

```
guillaume@debian1:~$ cd postgresql-8.4.0/contrib/start-scripts
guillaume@debian1:~$ sudo cp linux /etc/init.d/postgresql
guillaume@debian1:~$ sudo vim /etc/init.d/postgresql
```

La partie à modifier se trouve dans ce bloc de texte (ligne 29 à 43) :

```
## EDIT FROM HERE

# Installation prefix
prefix=/usr/local/pgsql

# Data directory
PGDATA="/usr/local/pgsql/data"

# Who to run the postmaster as, usually "postgres".  (NOT "root")
PGUSER=postgres

# Where to keep a log file
PGLOG="$PGDATA/serverlog"

## STOP EDITING HERE
```

J'ai modifié la variable préfix par /home/guillaume/postgresql, la variable PGDATA par /home/guillaume/postgresql-data et la variable PGUSER par guillaume. Évidemment, il faut revoir cela par rapport à votre contexte.

Testons le script :

```
guillaume@debian1:~$ sudo /etc/init.d/postgresql status
pg_ctl : le serveur est en cours d'exécution (PID : 2178)
/home/guillaume/postgresql/bin/postgres
guillaume@debian1:~$ sudo /etc/init.d/postgresql stop
Stopping PostgreSQL: LOG: a reçu une demande d'arrêt rapide
LOG:  annulation des transactions actives
LOG:  arrêt du processus autovacuum
LOG:  arrêt en cours
LOG:  le système de base de données est arrêté
guillaume@debian1:~$ /etc/init.d/postgresql start
Starting PostgreSQL: ok
```

Reste à s'assurer que PostgreSQL démarre au lancement du serveur. Sous Debian, il faut utiliser l'outil update-rc.d ainsi :

```
guillaume@debian1:~$ sudo update-rc.d postgresql defaults
```

Et voilà, PostgreSQL est installé et fonctionnel !

Installer à partir d'un paquet Debian

Pour éviter tous les (légers) tracats relatifs à l'installation après compilation, il est possible d'utiliser les paquets de sa distribution.

Sous Debian, il existe un groupe de paquets par version majeure. Cela permet d'installer plusieurs versions de PostgreSQL sur une même machine. L'installation se fait comme d'habitude avec les outils apt-get ou aptitude.

```
guillaume@debian1:~$ aptitude install postgresql-8.3
```

Cela peut vous installer une petite dizaine de paquets sur une Debian tout juste installée. Néanmoins, d'autres paquets peuvent vous intéresser comme les modules contrib (postgresql-contrib-8.3), la documentation (postgresql-doc-8.3) ou encore un langage de procédures stockées (postgresql-plperl-8.3 pour le Perl).

La version installée par la commande suivante concerne la version majeure 8.3. Or la 8.4 existe. Elle ne sera disponible qu'en backport pour Lenny. Vous devez tout d'abord modifier votre fichier /etc/apt/sources.list pour intégrer ce nouveau dépôt en insérant cette ligne :

```
deb http://www.backports.org/debian lenny-backports main contrib non-free
```

Puis, il faut mettre à jour la liste des paquets :

```
aptitude update
```

et installer la version 8.4 :

```
aptitude install postgresql-8.4
```

La particularité du paquet Debian est que l'installation du paquet cause l'initialisation de l'instance. Cette dernière est installée par défaut dans le répertoire /var/lib/postgresql/8.3/main (remplacez le 8.3 par la version majeure que vous avez installé). L'autre importante particularité de Debian est que les fichiers de configuration ne sont pas stockés dans le répertoire de l'instance. Il se trouve dans le répertoire /etc/postgresql/8.3/main (là-aussi, remplacez le 8.3 par la version majeure que vous avez installé). Expliquons un peu ces chemins. Le sous-répertoire 8.3 indique la version majeure, le sous-répertoire main est le nom de l'instance. Il est en effet possible avec le paquet Debian de créer plusieurs instances de PostgreSQL pour la même version stable. L'intérêt est limité dans ce cas.

Pour permettre une gestion simple des différentes instances et des différentes versions des exécutables, Debian fournit un ensemble de wrappers et de scripts. Le tableau 3 en liste les principaux.

Tableau 3 : liste des exécutables Debian supplémentaires

Exécutable	**Intérêts**
pg_createcluster	Ajout d'une instance
pg_dropcluster	Suppression d'une instance
pg_upgradecluster	Mise à jour d'une instance
pg_lscluster	Liste des instances et de leur statut
pg_ctlcluster	Contrôle des instances

Installer à partir d'un paquet RPM

Sous fedora, il existe aussi des paquets pour PostgreSQL. Il vous suffit d'un

```
yum install postgresql postgresql-server
```

Et vous vous trouvez avec un PostgreSQL 8.3 (pour une fedora 11).

La première différence avec les paquets Debian, c'est qu'aucune instance n'est créée automatiquement. Vous devez exécuter le script de démarrage avec l'option initdb de cette façon :

```
/etc/init.d/postgresql initdb
Initializing database: ok
```

Attention à SELinux qui peut causer bien des soucis.

L'instance est créée dans le répertoire `/var/lib/pgsql/data`. Les fichiers de configuration sont aussi dans ce répertoire. On peut donc dire que le paquet RPM de fedora est bien plus proche de ce qu'on obtiendrait en compilant PostgreSQL.

À noter qu'il est possible d'avoir la dernière version majeure (la 8.4) grâce aux paquets disponibles sur le site http://yum.pgsqlrpms.org/.

Conclusion

Comme vous avez pu le constater vous-même, l'installation de PostgreSQL est réellement simple, d'autant plus si vous choisissez l'option des paquets de votre distribution. C'est d'ailleurs ce qui est généralement conseillé. Vous bénéficiez des mises à jour de votre distribution, et vous n'avez pas à gérer les options à intégrer dans le programme. C'est autant de temps gagné que vous devriez plutôt passer sur la configuration de PostgreSQL.