PostgreSQL - установка и базовая настройка

Опубликовано muff в Вс, 2012-08-26 22:12



PostgreSQL - мощная свободная объектно-реляционная система управления базами данных (СУБД). Базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2003.

Основные преимущества **PostgreSQL**:

- поддержка БД практически неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования;
- масштабируемость.

Выполним установку из системы портов:

```
# cd /usr/ports/databases/postgresql92-server && make install clean && rehash
```

Опции сборки все оставил по умолчанию.

По завершению установки, порт вывел короткие рекомендации по дальнейшей настройке:

```
For procedural languages and postgresql functions, please note thatyou might have to
 update them when updating the server.

If you have many tables and many clients running, consider raisingkern.maxfiles usin
g sysctl(8), or reconfigure your kernelappropriately.

The port is set up to use autovacuum for new databases, but you mightalso want to va
cuum and perhaps backup your database regularly. Thereis a periodic script, /usr/loc
al/etc/periodic/daily/502.pgsql, thatyou may find useful. You can use it to backup a
nd perfom vacuum on alldatabases nightly. Per default, it perfoms `vacuum analyze'.
See thescript for instructions. For autovacuum settings, please review~pgsql/data/po
stgresql.conf.

To allow many simultaneous connections to your PostgreSQL server, youshould raise th
e SystemV shared memory limits in your kernel. Here areexample values for allowing u
p to 180 clients (configurations inpostgresql.conf also needed, of course):  options
        SYSVSHM  options         SYSVSEM  options         SYSVMSG  options
SHMMAXPGS=65536  options         SEMMNI=40  options         SEMMNS=240  options
    SEMUME=40  options         SEMMNU=120

If you plan to access your PostgreSQL server using ODBC, pleaseconsider running the
SQL script /usr/local/share/postgresql/odbc.sqlto get the functions required for ODB
C compliance.
```

```
Please note that if you use the rc script,/usr/local/etc/rc.d/postgresql, to initial
ize the database, unicode(UTF-8) will be used to store character data by default.  S
etpostgresql_initdb_flags or use login.conf settings described below toalter this be
haviour. See the start rc script for more info.

To set limits, environment stuff like locale and collation and otherthings, you can
set up a class in /etc/login.conf before initializingthe database. Add something sim
ilar to this to /etc/login.conf:---postgres:\        :lang=en_US.UTF-8:\        :set
env=LC_COLLATE=C:\        :tc=default:---and run `cap_mkdb /etc/login.conf'.Then add
 'postgresql_class="postgres"' to /etc/rc.conf.

=====================================================================

To initialize the database, run

  /usr/local/etc/rc.d/postgresql initdb

You can then start PostgreSQL by running:

  /usr/local/etc/rc.d/postgresql start

For postmaster settings, see ~pgsql/data/postgresql.conf

NB. FreeBSD's PostgreSQL port logs to syslog by default    See ~pgsql/data/postgresq
l.conf for more info

=====================================================================

To run PostgreSQL at startup, add'postgresql_enable="YES"' to /etc/rc.conf

===> Installing rc.d startup script(s)===> Correct pkg-plist sequence to create grou
p(s) and user(s)===>   Registering installation for postgresql-server-9.2.b4===> SEC
URITY REPORT:      This port has installed the following files which may act as netw
ork      servers and may therefore pose a remote security risk to the system./usr/lo
cal/bin/postgres

      This port has installed the following startup scripts which may cause      the
se network services to be started at boot time./usr/local/etc/rc.d/postgresql

      If there are vulnerabilities in these programs there may be a security      ri
sk to the system. FreeBSD makes no guarantee about the security of      ports includ
ed in the Ports Collection. Please type 'make deinstall'      to deinstall the port
if this is a concern.

      For more information, and contact details about the security      status of th
is software, see the following webpage:http://www.postgresql.org/[1]
```

Поскольку **PostgreSQL** использует кодировку **UTF-8** по умолчанию, для избежания проблем в дальнейшем, последуем совету и для пользователя **pgsql** (в новых версиях **PostgreSQL** работает от имени пользователя **pgsql**, а не **postgres**) немного подкорректируем локаль. Для этого в внесем такой блок в файл **/etc/login.conf**:

```
pgsql:\
     :lang=en_US.UTF-8:\
     :setenv=LC_COLLATE=C:\
     :tc=default:
```

Далее:

```
# cap_mkdb /etc/login.conf
```

Базы данных будем размещать в каталоге **/var/db/pgsql**. Соответственно создадим необходимый каталог и выставим права доступа:

```
# mkdir /var/db/pgsql
 # chown pgsql:pgsql /var/db/pgsql
```

Вносим изменения в **rc.conf**, указывая каталог хранения данных. Ну и добавим запуск **PostgreSQL** при старте системы:

```
# echo '# PostgreSQL' >> /etc/rc.conf
# echo 'postgresql_enable="YES"' >> /etc/rc.conf
 # echo 'postgresql_data="/var/db/pgsql"' >> /etc/rc.conf
 # echo 'postgresql_class="postgres"' >> /etc/rc.conf
```

После этих манипуляций выполним инициализацию базы данных:

```
# /usr/local/etc/rc.d/postgresql initdb
```

Скрипт по ходу своей работы будет выводить сообщения на консоль. Пример такого вывода:

```
The files belonging to this database system will be owned by user "pgsql".This user
must also own the server process.

The database cluster will be initialized with locale "C".The default text search con
figuration will be set to "english".

fixing permissions on existing directory /var/db/pgsql ... okcreating subdirectories
 ... okselecting default max_connections ... 100selecting default shared_buffers ...
 32MBcreating configuration files ... okcreating template1 database in /var/db/pgsql
/base/1 ... okinitializing pg_authid ... okinitializing dependencies ... okcreating
system views ... okloading system objects' descriptions ... okcreating collations ..
. not supported on this platformcreating conversions ... okcreating dictionaries ...
 oksetting privileges on built-in objects ... okcreating information schema ... oklo
ading PL/pgSQL server-side language ... okvacuuming database template1 ... okcopying
 template1 to template0 ... okcopying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connectionsYou can change this by
 editing pg_hba.conf or using the option -A, or--auth-local and --auth-host, the nex
t time you run initdb.

Success. You can now start the database server using:

    /usr/local/bin/postgres -D /var/db/pgsqlor    /usr/local/bin/pg_ctl -D /var/db/p
gsql -l logfile start
```

После инициализации БД запускаем **PostgreSQL** сервер:

```
# sh /usr/local/etc/rc.d/postgresql start
```

После запуска проверяем состояние сервера:

```
# sh /usr/local/etc/rc.d/postgresql status
pg_ctl: server is running (PID: 30752)
/usr/local/bin/postgres "-D" "/var/db/pgsql"
```

Что ж, **PostgreSQL** сервер запущен и работает... Однако, в настройках по умолчанию сервер сконфигурирован так, что к нему можно подключиться с локального хоста без пароля. Это явная дыра в безопасности, поэтому попытаемся исправить ситуацию.

Сервер PosgreSQL предоставляет два варианта управления пользователями и базами данных:

- утилиты коммандной строки (**createuser**, **createdb**, **dropuser**, **dropdb**...)
- интерактивный терминал

Воспользуемся интерактивным терминалом для настройки прав доступа. Подключимся от имени пользователся **pgsql**. Пароль не нужен.

```
# psql -U pgsql template1

psql (9.2beta4)
Введите "help", чтобы получить справку.

template1=#
```

Наберем команду help и ознакомимся со справкой:

```
template1=# help


?? ?????????? psql - ????????? ????????? ?????? ? PostgreSQL.???:   \copyright - ??
????? ???????????????       \h - ??????? ?? ?????????? SQL       \? - ??????? ?? ???
????? psql       \g ??? ; ? ????? ?????? - ?????????? ???????       \q - ?????
```

Советую ознакомиться со справкой по операторам **SQL** и командам **psql**. Теперь создадим тестового пользователя и базу данных, к которой он будет иметь доступ.

```
template1=# CREATE USER username;
 CREATE ROLE
```

Проверим список ролей:

```
template1-# \du


                                  ?????? ????? ??? ???? |                          ???
?????                          | ???? ?????----------+------------------------------
-------------------------+------------ pgsql    | ???????????????, ??????? ????,
??????? ??, ?????????? | {} username |                              |{}
```

Повысим созданному пользователю права доступа:

```
template1=# ALTER USER username superuser createrole createdb;
ALTER ROLE
```

Повторно проверим список ролей:

```
template1=# \du


                                  ?????? ????? ??? ???? |                          ???
?????                          | ???? ?????----------+------------------------------
-------------------------+------------ pgsql    | ???????????????, ??????? ????,
??????? ??, ?????????? | {} username | ???????????????, ??????? ????, ??????? ??
           | {}
```

Обычному пользователю такие права ни к чему, их стоит выдавать только администратору системы. Отменим определенные права доступа:

```
template1=# ALTER USER username nosuperuser nocreaterole nocreatedb;
ALTER ROLE
```

А вот пароли для пользователя - вещь обязательная. Зададим пароль для созданного пользователя:

```
template1=# \password username
 Введите новый пароль:
Повторите его:
```

При вводе пароль не отображается! Также стоит установить пароль для пользователя **pgsql**. После этого включаем проверку пароля при подключении к **PostgreSQL**-серверу. Неоходимо внести изменения в файл **/var/db/pgsql/pg_hba.conf**. Метод подключения trust необходимо изменить на md5 (также полностью удаляем строку для IPv6). В результате имеем такой результат:

```
# TYPE   DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections onlylocal   all            all
                                md5# IPv4 local connections:host    all
all             127.0.0.1/32            md5
```

Для того, чтобы изменения вступили в силу, необходимо перезапустить **PostgreSQL**-сервер:

```
# sh /usr/local/etc/rc.d/postgresql restart
```

Повторно подключаемся к интерактивной оболочке:

```
# psql -U pgsql template1
Пароль пользователя pgsql:
 psql (9.2beta4)
Введите "help", чтобы получить справку.

template1=#
```

Теперь сервер при подключении запрашивает пароль. Ну и напоследок создадим базу данных и дадим на нее права пользователю **username**. Сделать это можно двумя способами.

Первый способ:

```
template1=# CREATE DATABASE userdb owner=username;
CREATE DATABASE
```

Второй способ:

```
template1=# CREATE DATABASE userdb;
 CREATE DATABASE
template1=# ALTER DATABASE userdb owner to username;
ALTER DATABASE
```

Просмотрим список имеющихся баз данных:

```
template1=# \l


                              ?????? ??? ??????    ???    | ???????? | ????????? | LC
_COLLATE | LC_CTYPE |  ????? ???????-----------+----------+-----------+------------+
----------+----------------- postgres  | pgsql    | UTF8      | C           | C
  | template0 | pgsql    | UTF8      | C           | C         | =c/pgsql       +
      |           |            |             |           | pgsql=CTc/pgsql template1 | p
gsql    | UTF8      | C           | C         | =c/pgsql       +            |
|           |            |           | pgsql=CTc/pgsql userdb    | username | UTF8
   | C           | C         |(4 ??????)
```

В принципе, все интуитивно понятно. Используйте в своей работе документацию и подсказки, и работа с сервером **PostgreSQL** не должна вызывать больших проблем.

Последний штрих - тюнинг **PostgreSQL**-сервера. Все основные настройки хранятся в файле

/var/db/pgsql/postgresql.conf. Отметим опции, на которые стоит обратить внимание (перевод описания опций взят здесь [2]):

- **max_connections** - максимальное количество одновременных подключений к серверу БД. Используйте этот параметр, чтобы не допустить запуска большого количества процессов сервера. Иначе есть вероятность, что серер БД исчерпает весь объем ОЗУ и будет активно использоваться подкачка, что в свою очередь повлечет за собой падение производительности.
- **shared_buffers** - объем разделяемой памяти, используемый сервером БД. По умолчанию - 32 Мб, но может быть меньше, если настройки вашего ядра не позволяют выделить такой объем памяти, определяется при инициализации кластера БД утилитой initdb. Большие значения этого параметра положительно сказываются на производительности сервера БД.
  Если у вас выделенный компьютер с ОЗУ 1 Гб и больше под сервер БД, то хорошим значением этой переменной будет 25% от объема памяти. При большой нагрузке даже большие значения этого параметра могут быть эффективными, но так как PostgreSQL полагается на кэш ОС, то выделение более 40% от объема памяти вряд ли имеет смысл. Для больших значений этого параметра требуется так же увеличить значение параметра checkpoint_segments.
  На системах с ОЗУ меньше 1 Гб правильнее будет использовать меньший объем памяти (чем 25%), чтобы не исчерпать весь объем оперативной памяти. На винде большие значения этого параметра могут оказаться эффективными. Вы можете добиться большей производительности сохраняя это значение маленьким и больше используя средства кэширования ОС. Хорошим диапазоном значений для Windows является 64-512 Мб.
- **temp_buffers** - максимальный размер временных буферов для каждой сессии. Эта память используется только локально в сессии для временных таблиц. По умолчанию - 8 МБ. Значение может быть изменено во время сессии, но только до первого использования этой памяти.
- **max_prepared_transactions** - максимальное количество "prepared" транзакций (смотрите описание SQL команды PREPARE TRANSACTION в документации). Чтобы отключить эту фишку, поставьте значение в 0.
- **work_mem** - определяет объем памяти, который будет использоваться внутренними операциями сортировки и хэш-таблицами прежде, чем переключиться на временные дисковые файлы. Учтите, что для сложных запросов несколько внутренних операций сортировки и работа с хэш-таблицами могут работать параллельно (одновременно). Кроме того, несколько сессий могут делать такие операции одновременно. В итоге необходимый объем памяти для этих операций может в несколько раз превышать значение параметра work_mem. Учтите это при выборе значения для этого параметра. Под внутренними операциями сортировки подразумевается - ORDER BY, DISTINCT и слияния.
- **maintenance_work_mem** - максимальный объем памяти, используемый для внутренних операций, таких как VACUUM, CREATE INDEX и ALTER TABLE ADD FOREIGN KEY. По умолчанию - 16 Мб. Эти команды выполняются только во во время сессии, так что можно выбирать большие значения для этого параметра, чем для параметра work_mem. Большие значения могут положительно сказаться на производительности vacuuming и скорости восстановления БД из дампа. Только учтите, что процесс autovacuum запускается autovacuum_max_workers раз, поэтому может потребоваться больше свободной памяти.
- **max_stack_depth** - максимальная глубина стека. Хорошим значением этого параметра является максимально разрешенная глубина стека в системе.
- **max_fsm_pages** - с помощью этого параметра можно управлять картой свободного пространства. Когда что-то удаляется из таблицы, то место занимаемое этим что-то не освобождается на диске, вместо этого занимаемое место просто помечается как "свободно" в карте свободного пространства. Потом это место используется для новых записей. Если на вашем сервере очень много удаляется/добавляется данных из/в таблицы, то большие значения этого параметра могут положительно сказаться на производительности.

Источник (получено *2026-11-12 05:56*):
http://muff.kiev.ua/content/postgresql-ustanovka-i-bazovaya-nastroika

Ссылки:
[1] http://www.postgresql.org/
[2] http://www.info-x.org/node/71