Welcome To My Blog Page

Tuesday, August 27, 2013

Installing PostgreSQL Database Server/Client on RedHat Linux Families (RedHat,Fedora,CentOS,SELinux)

Hello everyone,

Today, I would like to talk about a very very good open source Database Server called **PostgreSQL**. Recently, I've started to work with Postgresql and really love it. I would say it has all features of commercial databases such as DB2 and Oracle and even more. Some of core features are:

1. Object-Relational DBMS
2. Capable of handling complex routines and rules
3. Declarative SQL queries
4. Multi version concurrency control
5. Multi user support
6. Transactions

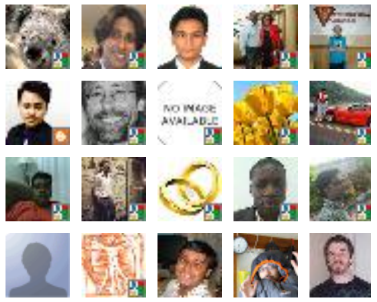
Total Pageviews

2 7 4 3 9 6

Followers

Join this site
with Google Friend Connect

Members (37) More »



Already a member? Sign in

7. Query optimization
8. Inheritance
9. Arrays
10. Highly extensible
11. Comprehensive SQL support (support SQL99, SQL92)
12. Referential integrity (insure the validity of database's data)
13. Flexible API (so many vendors such as Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, and Pike have deployment support for PostgreSQL RDBMS
14. Procedural Languages (it supports internal procedural native language called PL/pgSQL, which comparable to the Oracle procedural language PL/SQL, and also it has ability to use Perl, Python, and/or TCL as an embedded procedural language)
15. MVCC (Multi-version Concurrency Control is the technology that Posrgresql uses to avoid unnecessary locking)
16. Server/Client
17. Write Ahead Logging (WAL), ability to write the changes to log file before writing to database.(In case of unlikely crash, there will be a record of transaction to restore)

Installing PostgreSQL

The following shows how to install PostgreSQL from source code. Although you could install PostgreSQL server and client easily with yum command (***yum install postgresql-server postgresql-client***), *I would recommend to install from source code* because it is so flexible to adding/removing features and to customize it even after compiling source code. For example, you can add more features to Postgresql by reconfiguring and compiling the source code again without losing your data and databases.

1. Installing the required/optional packages:

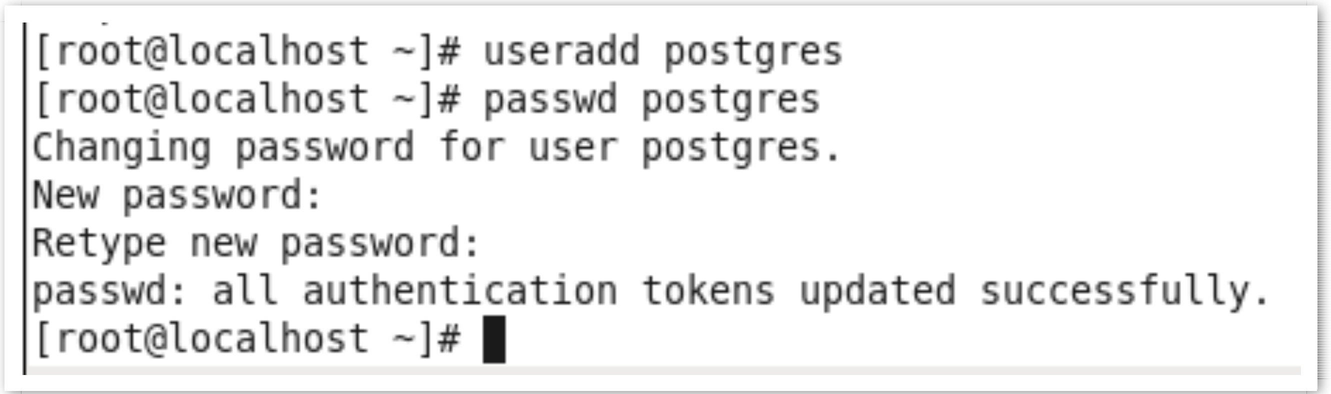
yum install gcc make kernel-devel perl-ExtUtils-MakeMaker perl-ExtUtils-Embed readline-devel zlib-devel openssl-devel pam-devel libxml2-devel openldap-devel tcl-devel python-devel flex bison

2. Download the source code from command line: (PostgreSQL-9.2.4 is the current stable version at the time of writing this)

wget http://ftp.postgresql.org/pub/source/v9.2.4/postgresql-9.2.4.tar.gz

3. Creating the "***postgres***" user:
It is always a good idea to create a PostgreSQL superuser to own and manage the PostgreSQL database files rather than using "root" account as the PostgreSQL superuser because of security purposes. This user can be named anything and I named it "posrgres":

su - --> switch to root account
useradd postgres --> create user
passwd postgres --> set password



```
[root@localhost ~]# useradd postgres
[root@localhost ~]# passwd postgres
Changing password for user postgres.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]# 
```

Figure 1

4. Move and unpack the Postgresql source package:

cp postgresql-9.2.4.tar.gz /usr/local/src/

About Me

Khosro Ta
Toronto, Ontari
Canada

I was graduate
from Seneca College in Comput
System Technology (CTY) majo
December 2011 and also I have
bachelor degree in Computer
Hardware Engineering since 19
am very interested in IT and ha
joined to Open Source Commu
since 2009. I believe Open Sou
and its power. I love Linux and i
opinion, Linux is the best OS in
world. More... click here:

View my complete profile

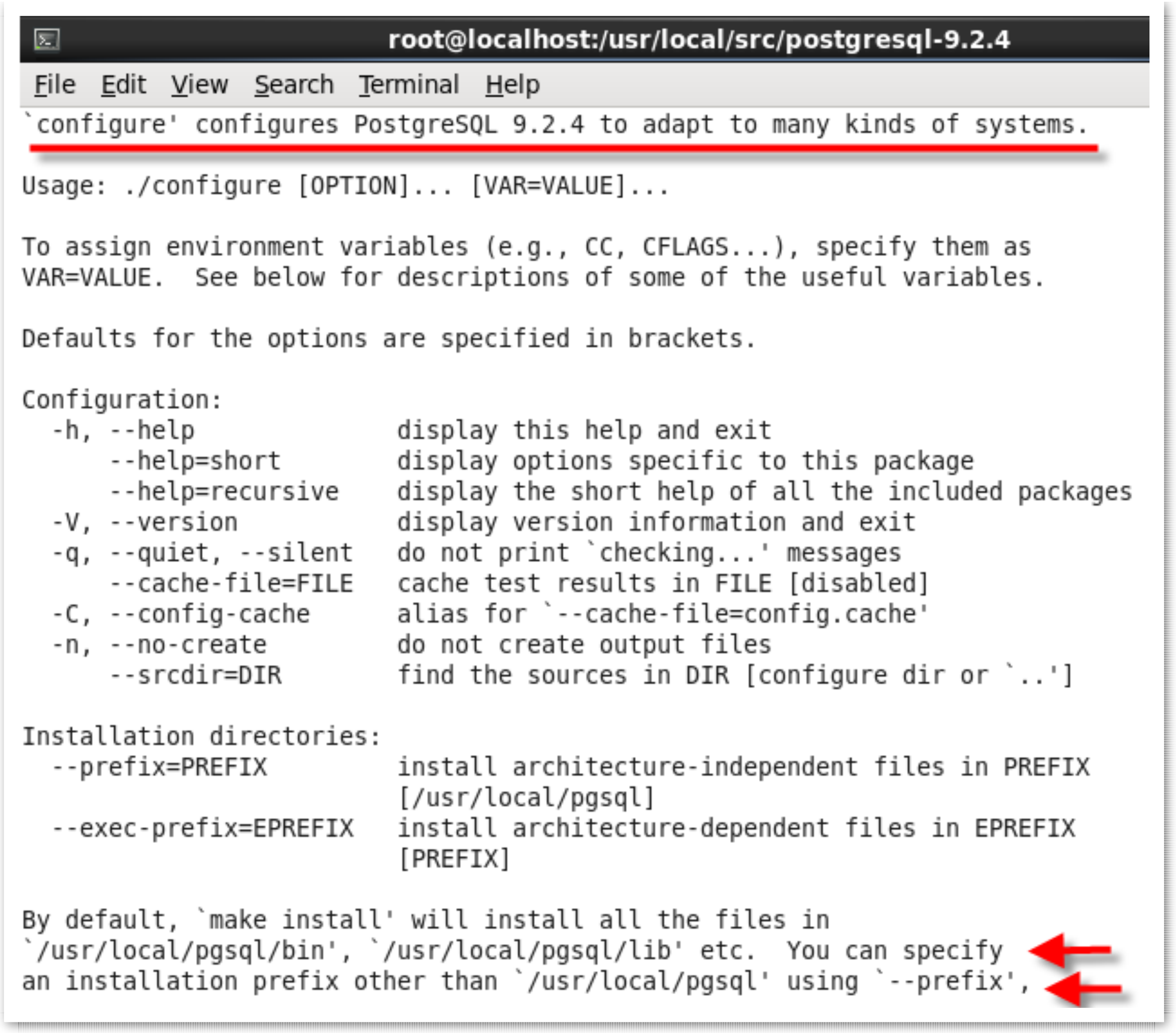
cd /usr/local/src/
tar -xzvf postgresql-9.2.4.tar.gz

5. Grant the ownership of the Postgresql source directory to "postgres" user. It enables you to compile PostgreSQL as the "postgres" user.

chown -R postgres.postgres postgresql-9.2.4

6. Configuring the source.Now, switch to postgresql-9.2.4 directory:

cd postgresql-9.2.4
and run
./configure --help
to see all available options to customize your PostgreSQL


root@localhost:/usr/local/src/postgresql-9.2.4
File Edit View Search Terminal Help

```
`configure' configures PostgreSQL 9.2.4 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE.  See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help              display this help and exit
      --help=short        display options specific to this package
      --help=recursive    display the short help of all the included packages
  -V, --version           display version information and exit
  -q, --quiet, --silent   do not print `checking...' messages
      --cache-file=FILE   cache test results in FILE [disabled]
  -C, --config-cache      alias for `--cache-file=config.cache'
  -n, --no-create         do not create output files
      --srcdir=DIR        find the sources in DIR [configure dir or `..']

Installation directories:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local/pgsql]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [PREFIX]

By default, `make install' will install all the files in
`/usr/local/pgsql/bin', `/usr/local/pgsql/lib' etc.  You can specify
an installation prefix other than `/usr/local/pgsql' using `--prefix',
```

Figure 2

It's pretty self explanatory. For our purpose, I am going to use the options below and leave other options as default:

./configure --mandir=/usr/local/pgsql/man --with-tcl --with-perl --with-python --with-pam --with-ldap --with-openssl --with-libxml

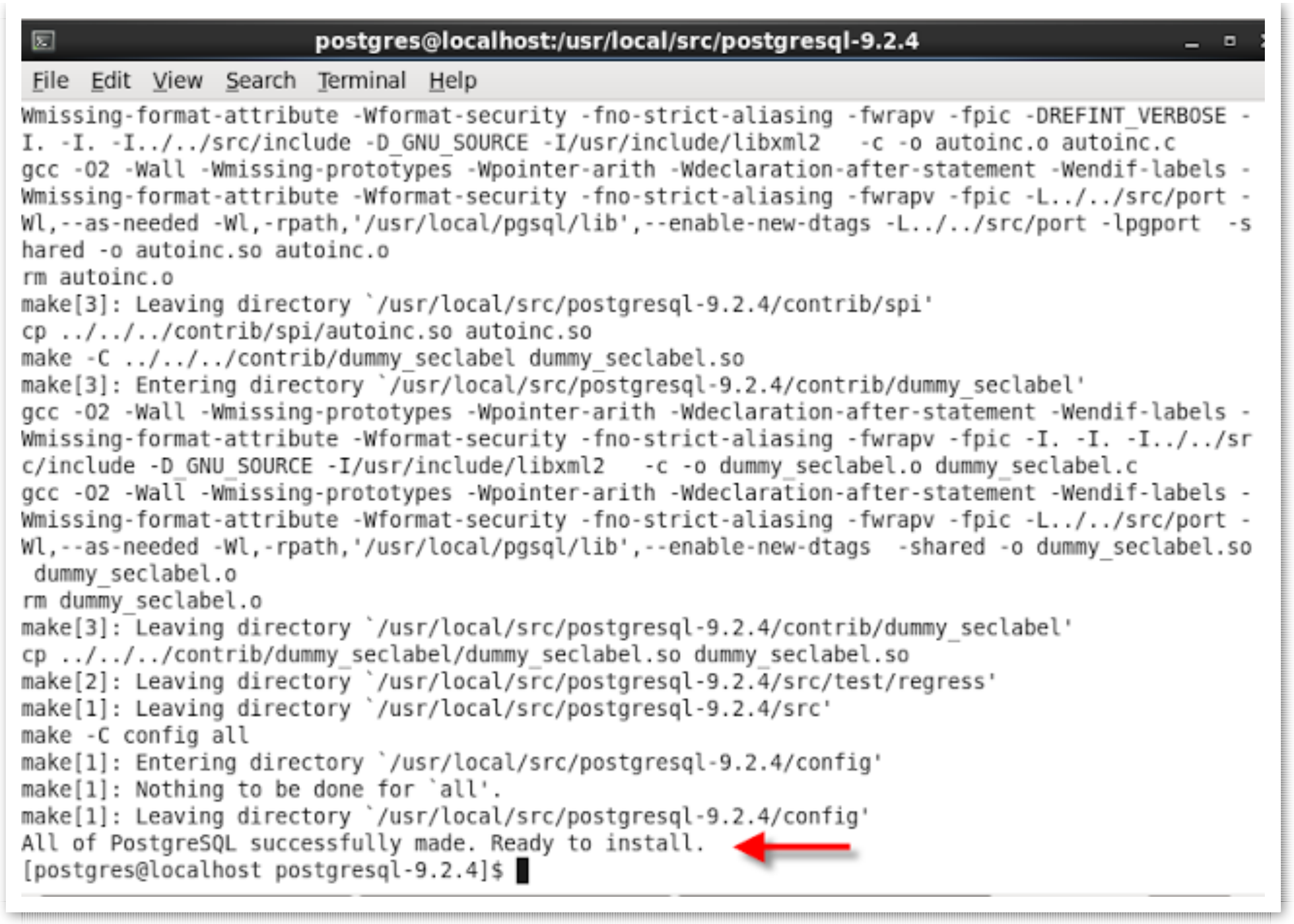
which

--mandir=DIR	is	man documentation [DATAROOTDIR/man]
--with-tcl	is	build Tcl modules (PL/Tcl); if you plan to use pl/Tcl procedural language
--with-perl	is	build Perl modules (PL/Perl); if you plan to use pl/Perl procedural language
--with-python	is	build Python modules (PL/Python); if you plan to use pl/Python procedural language
--with-pam	is	build with PAM support
--with-ldap	is	build with LDAP support
--with-openssl	is	build with OpenSSL support
--with-libxml	is	build with XML support

7. Now, run the "*make*" command after switching to "postgres" user:

su postgres
make

After compiling source, you should see the following message:
"*All of PostgreSQL successfully made. Ready to install.*"

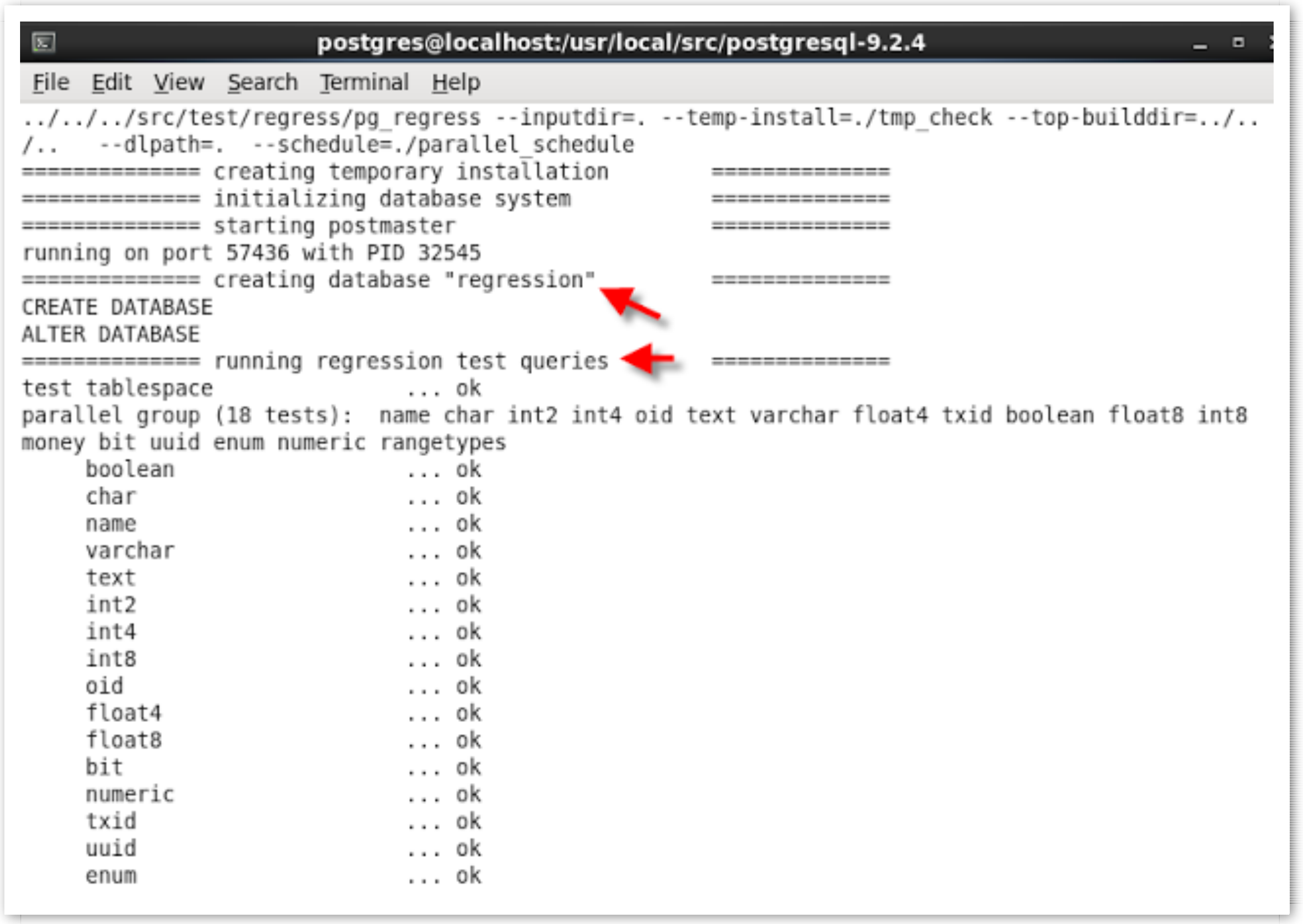

postgres@localhost:/usr/local/src/postgresql-9.2.4
File Edit View Search Terminal Help

```
Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fpic -DREFINT_VERBOSE -
I. -I. -I../../src/include -D_GNU_SOURCE -I/usr/include/libxml2   -c -o autoinc.o autoinc.c
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -
Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fpic -L../../src/port -
Wl,--as-needed -Wl,-rpath,'/usr/local/pgsql/lib',--enable-new-dtags -L../../src/port -lpgport  -s
hared -o autoinc.so autoinc.o
rm autoinc.o
make[3]: Leaving directory `/usr/local/src/postgresql-9.2.4/contrib/spi'
cp ../../../contrib/spi/autoinc.so autoinc.so
make -C ../../../contrib/dummy_seclabel dummy_seclabel.so
make[3]: Entering directory `/usr/local/src/postgresql-9.2.4/contrib/dummy_seclabel'
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -
Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fpic -I. -I. -I../../sr
c/include -D_GNU_SOURCE -I/usr/include/libxml2   -c -o dummy_seclabel.o dummy_seclabel.c
gcc -O2 -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-labels -
Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -fpic -L../../src/port -
Wl,--as-needed -Wl,-rpath,'/usr/local/pgsql/lib',--enable-new-dtags  -shared -o dummy_seclabel.so
 dummy_seclabel.o
rm dummy_seclabel.o
make[3]: Leaving directory `/usr/local/src/postgresql-9.2.4/contrib/dummy_seclabel'
cp ../../../contrib/dummy_seclabel/dummy_seclabel.so dummy_seclabel.so
make[2]: Leaving directory `/usr/local/src/postgresql-9.2.4/src/test/regress'
make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/src'
make -C config all
make[1]: Entering directory `/usr/local/src/postgresql-9.2.4/config'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/config'
All of PostgreSQL successfully made. Ready to install.
[postgres@localhost postgresql-9.2.4]$ █
```

Figure 3

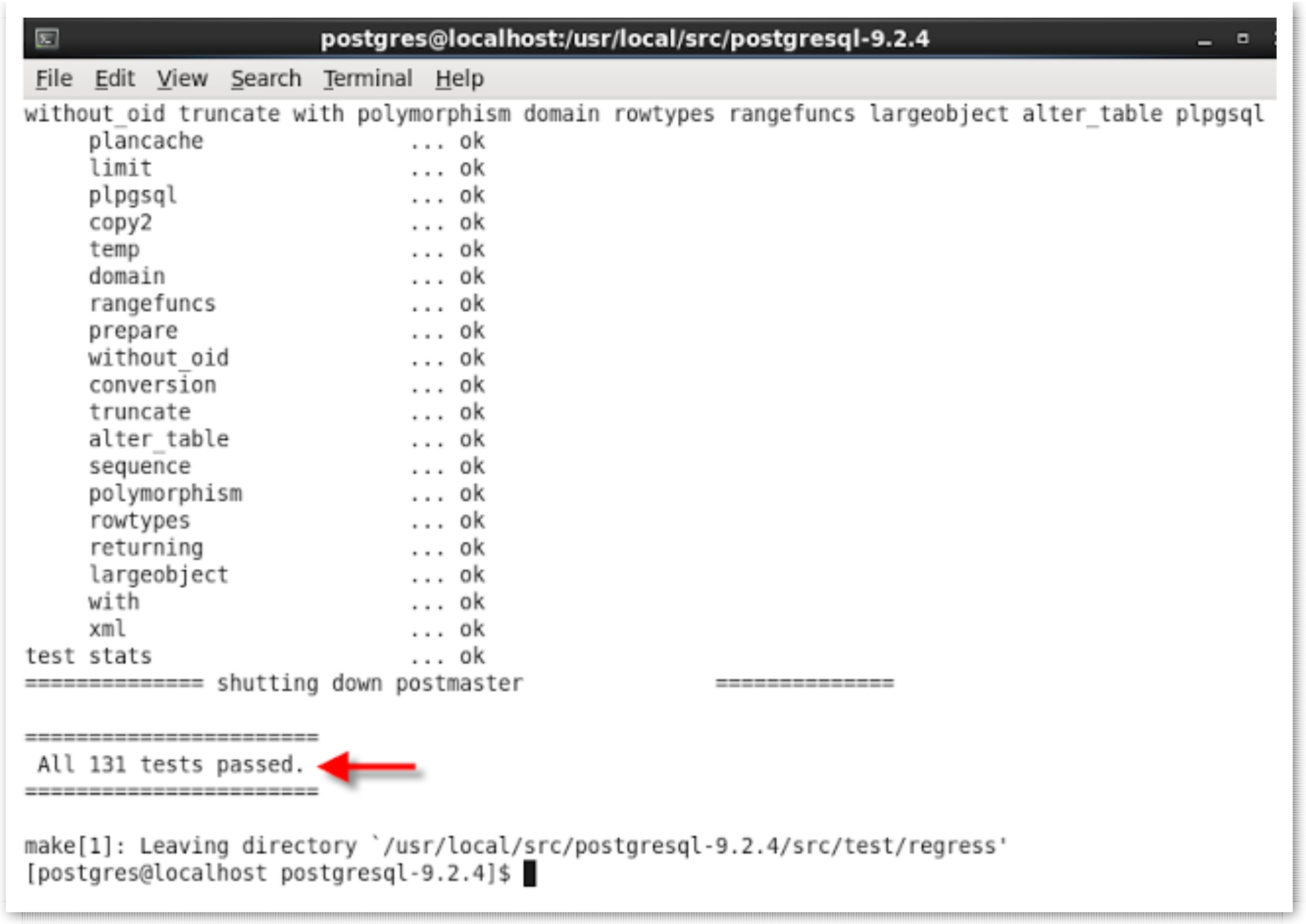
8. We need to do regression test. This is optional but really recommended it.

make check


postgres@localhost:/usr/local/src/postgresql-9.2.4
File Edit View Search Terminal Help

```
../../../src/test/regress/pg_regress --inputdir=. --temp-install=./tmp_check --top-builddir=../..
/..   --dlpath=.  --schedule=./parallel_schedule
============== creating temporary installation        ==============
============== initializing database system           ==============
============== starting postmaster                    ==============
running on port 57436 with PID 32545
============== creating database "regression"         ==============
CREATE DATABASE
ALTER DATABASE
============== running regression test queries        ==============
test tablespace               ... ok
parallel group (18 tests):  name char int2 int4 oid text varchar float4 txid boolean float8 int8
money bit uuid enum numeric rangetypes
     boolean                  ... ok
     char                     ... ok
     name                     ... ok
     varchar                  ... ok
     text                     ... ok
     int2                     ... ok
     int4                     ... ok
     int8                     ... ok
     oid                      ... ok
     float4                   ... ok
     float8                   ... ok
     bit                      ... ok
     numeric                  ... ok
     txid                     ... ok
     uuid                     ... ok
     enum                     ... ok
```

Figure 4

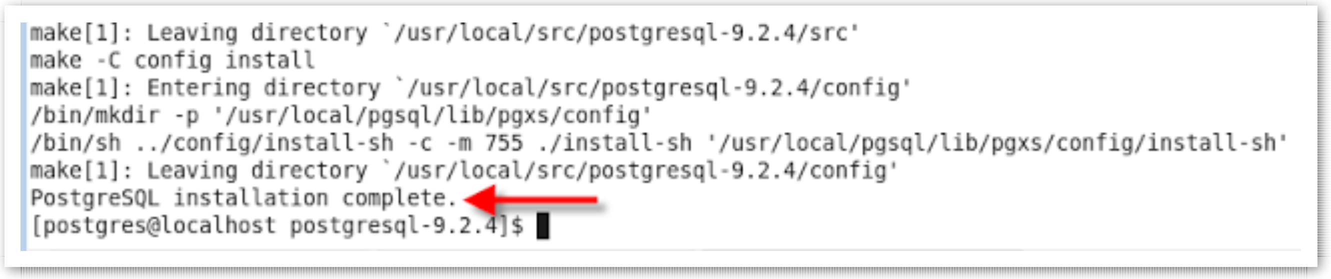

postgres@localhost:/usr/local/src/postgresql-9.2.4
File Edit View Search Terminal Help

```
without_oid truncate with polymorphism domain rowtypes rangefuncs largeobject alter_table plpgsql
     plancache                ... ok
     limit                    ... ok
     plpgsql                  ... ok
     copy2                    ... ok
     temp                     ... ok
     domain                   ... ok
     rangefuncs               ... ok
     prepare                  ... ok
     without_oid              ... ok
     conversion               ... ok
     truncate                 ... ok
     alter_table              ... ok
     sequence                 ... ok
     polymorphism             ... ok
     rowtypes                 ... ok
     returning                ... ok
     largeobject              ... ok
     with                     ... ok
     xml                      ... ok
test stats                    ... ok
============== shutting down postmaster               ==============

=======================
 All 131 tests passed.
=======================

make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/src/test/regress'
[postgres@localhost postgresql-9.2.4]$ 
```

Figure 5

8. Now, you need to install compiled programs and libraries and "**su -**" command save your time to log in as root user for command's execution:

su -c "make install"



```
make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/src'
make -C config install
make[1]: Entering directory `/usr/local/src/postgresql-9.2.4/config'
/bin/mkdir -p '/usr/local/pgsql/lib/pgxs/config'
/bin/sh ../config/install-sh -c -m 755 ./install-sh '/usr/local/pgsql/lib/pgxs/config/install-sh'
make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/config'
PostgreSQL installation complete.
[postgres@localhost postgresql-9.2.4]$ 
```

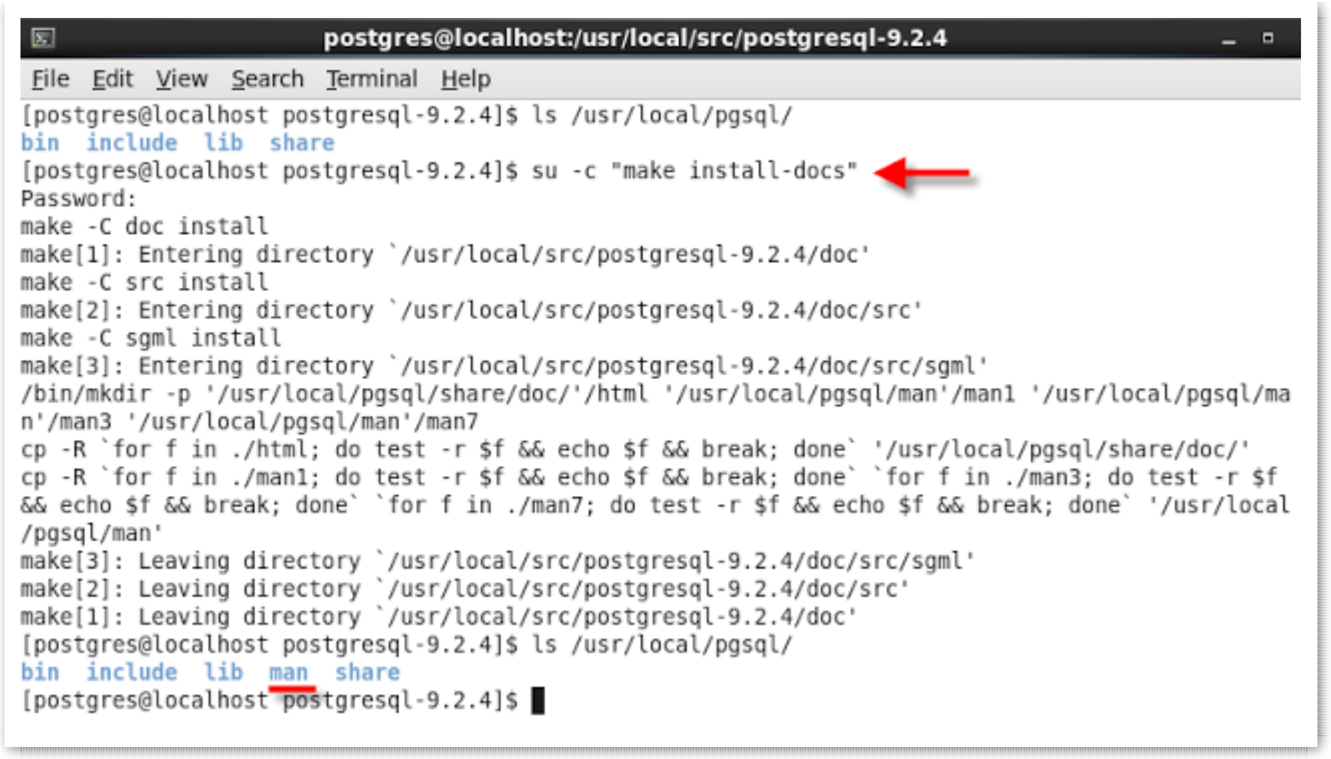
Figure 6

Don't forget to change the owner of PostgreSQL installation directory, in this case /usr/local/pgsql, to "postgres" user:

su -c "chown -R postgres.postgres /usr/local/pgsql"

9. Then, install documentation:

su -c "make install-docs"


postgres@localhost:/usr/local/src/postgresql-9.2.4
File Edit View Search Terminal Help

```
[postgres@localhost postgresql-9.2.4]$ ls /usr/local/pgsql/
bin  include  lib  share
[postgres@localhost postgresql-9.2.4]$ su -c "make install-docs"
Password:
make -C doc install
make[1]: Entering directory `/usr/local/src/postgresql-9.2.4/doc'
make -C src install
make[2]: Entering directory `/usr/local/src/postgresql-9.2.4/doc/src'
make -C sgml install
make[3]: Entering directory `/usr/local/src/postgresql-9.2.4/doc/src/sgml'
/bin/mkdir -p '/usr/local/pgsql/share/doc/'/html '/usr/local/pgsql/man'/man1 '/usr/local/pgsql/ma
n'/man3 '/usr/local/pgsql/man'/man7
cp -R `for f in ./html; do test -r $f && echo $f && break; done` '/usr/local/pgsql/share/doc/'
cp -R `for f in ./man1; do test -r $f && echo $f && break; done` `for f in ./man3; do test -r $f
&& echo $f && break; done` `for f in ./man7; do test -r $f && echo $f && break; done` '/usr/local
/pgsql/man'
make[3]: Leaving directory `/usr/local/src/postgresql-9.2.4/doc/src/sgml'
make[2]: Leaving directory `/usr/local/src/postgresql-9.2.4/doc/src'
make[1]: Leaving directory `/usr/local/src/postgresql-9.2.4/doc'
[postgres@localhost postgresql-9.2.4]$ ls /usr/local/pgsql/
bin  include  lib  man  share
[postgres@localhost postgresql-9.2.4]$
```

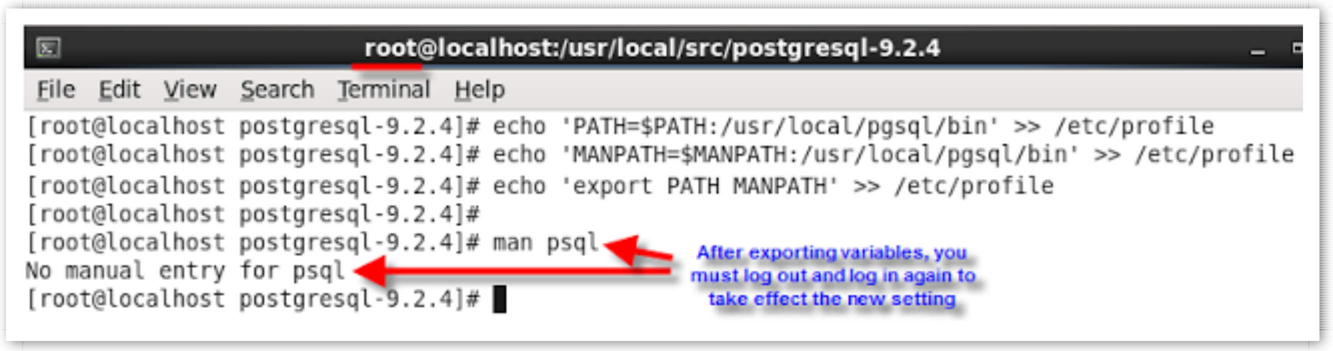
Figure 7

10. Next, we need to set environment variables. I am going to set environment variables for man page and bin directory. In order to do that, add the following lines to the end of */etc/profile*

echo 'PATH=$PATH:/usr/local/pgsql/bin' >> /etc/profile
echo 'MANPATH=$MANPATH:/usr/local/pgsql/bin' >> /etc/profile
echo 'export PATH MANPATH' >> /etc/profile

Don't forget to log out and log in again to take effect the new variables.

Now try "man psql"



```
root@localhost:/usr/local/src/postgresql-9.2.4
[root@localhost postgresql-9.2.4]# echo 'PATH=$PATH:/usr/local/pgsql/bin' >> /etc/profile
[root@localhost postgresql-9.2.4]# echo 'MANPATH=$MANPATH:/usr/local/pgsql/bin' >> /etc/profile
[root@localhost postgresql-9.2.4]# echo 'export PATH MANPATH' >> /etc/profile
[root@localhost postgresql-9.2.4]#
[root@localhost postgresql-9.2.4]# man psql
No manual entry for psql
[root@localhost postgresql-9.2.4]#
```

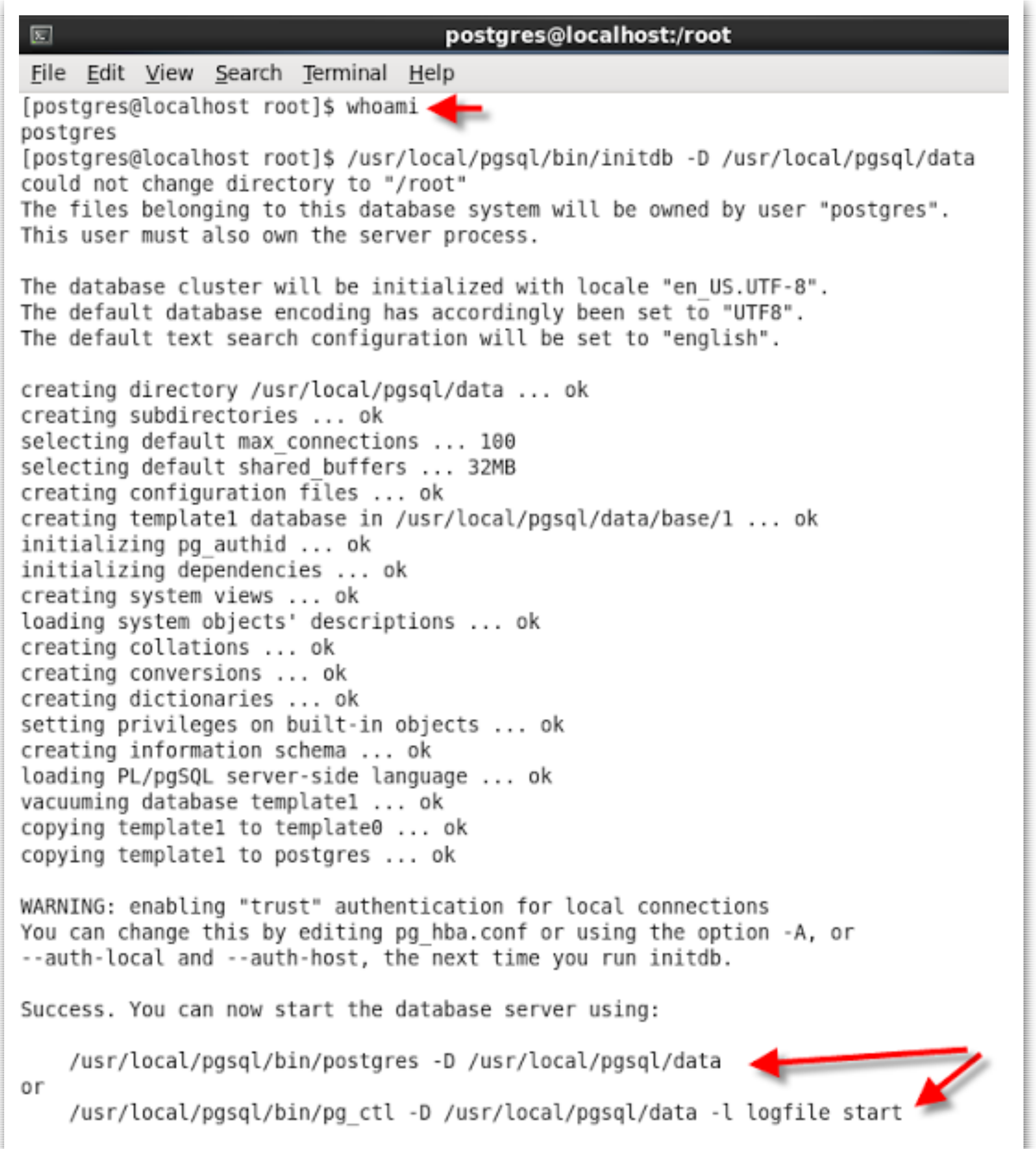
File Edit View Search Terminal Help
After exporting variables, you must log out and log in again to take effect the new setting

Figure 8

11. Now, we need to initialize and start PostgreSQL. Make sure you logged in as postgres user. Then run the following command:

/usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data

The path after the **-D** option is up to you. You can put any path **BUT** make sure on that path the user "postgres" has write access on it.



```
[postgres@localhost root]$ whoami
postgres
[postgres@localhost root]$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
could not change directory to "/root"
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

creating directory /usr/local/pgsql/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 32MB
creating configuration files ... ok
creating template1 database in /usr/local/pgsql/data/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating collations ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
loading PL/pgSQL server-side language ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

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

Success. You can now start the database server using:

    /usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data
or
    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start
```

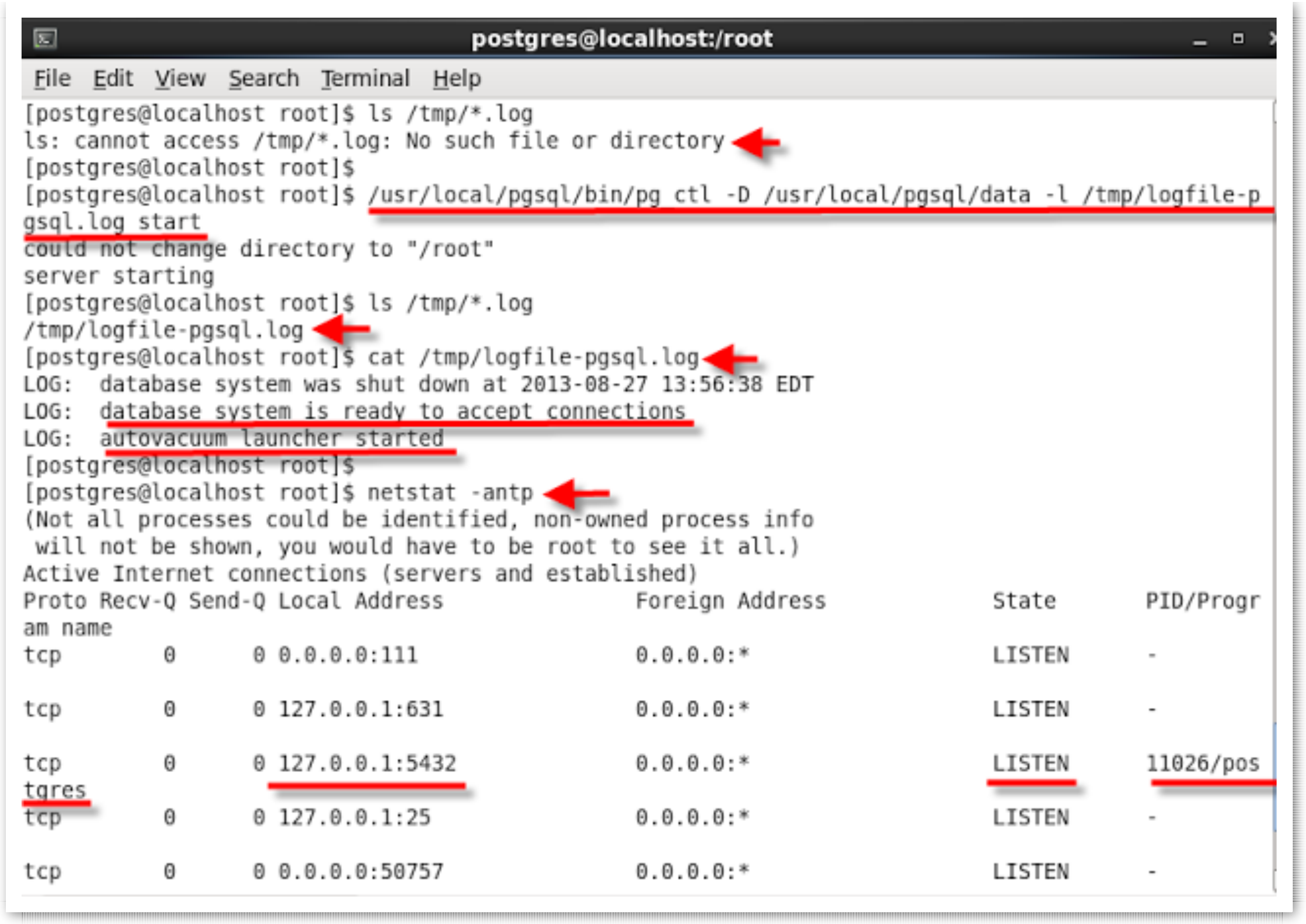
Figure 9

12. To start the database server in the background, run the following command:

```
/usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l /tmp/logfile-pgsql.log start
```

to make sure that server is running, use the following commands:

```
cat /tmp/logfile-pgsql.log
netstat -antp
```



```
[postgres@localhost root]$ ls /tmp/*.log
ls: cannot access /tmp/*.log: No such file or directory
[postgres@localhost root]$
[postgres@localhost root]$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l /tmp/logfile-p
gsql.log start
could not change directory to "/root"
server starting
[postgres@localhost root]$ ls /tmp/*.log
/tmp/logfile-pgsql.log
[postgres@localhost root]$ cat /tmp/logfile-pgsql.log
LOG:  database system was shut down at 2013-08-27 13:56:38 EDT
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
[postgres@localhost root]$
[postgres@localhost root]$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Progr
am name
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN      -

tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN      -

tcp        0      0 127.0.0.1:5432              0.0.0.0:*                   LISTEN      11026/pos
tgres
tcp        0      0 127.0.0.1:25                0.0.0.0:*                   LISTEN      -

tcp        0      0 0.0.0.0:50757               0.0.0.0:*                   LISTEN      -
```

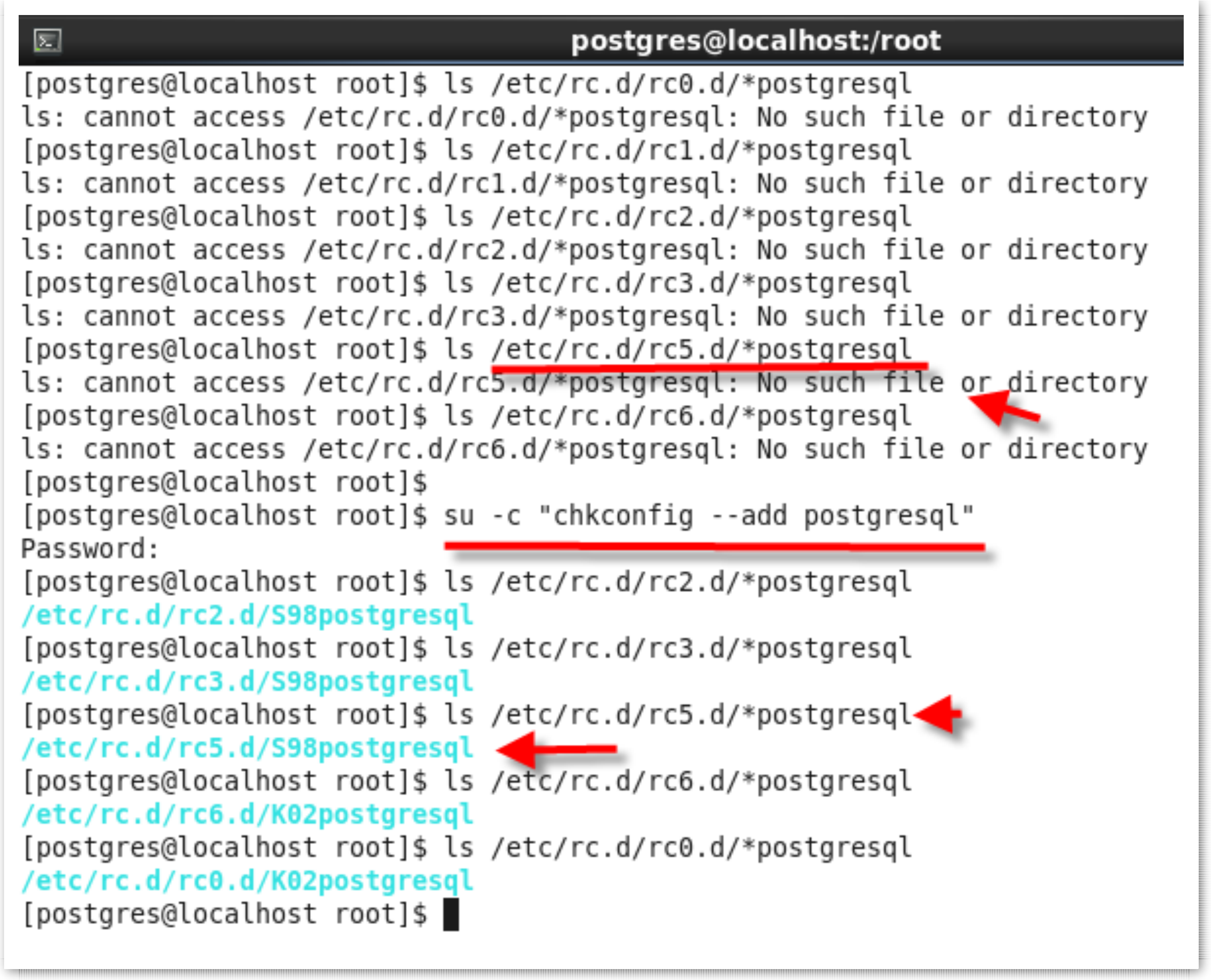
Figure 10

13. Next, we need to configure PostgreSQL in *SysV* Script so that we can gracefully control PostgreSQL database though the use of SysV runlevel system. In order to do that, we need to copy a script called "*linux*" to init.d directory. I also renamed it to "postgresql" to be more meaningful. Run the following commands:

su -c "cp /usr/local/src/postgresql-9.2.4/contrib/start-scripts/linux /etc/rc.d/init.d/postgresql"
su -c "chmod a+x /etc/rc.d/init.d/postgresql" --> make the script executable

If you wish for the script to startup PostgreSQL automatically when the machine boots up, run the following command:

su -c "chkconfig --add postgresql"

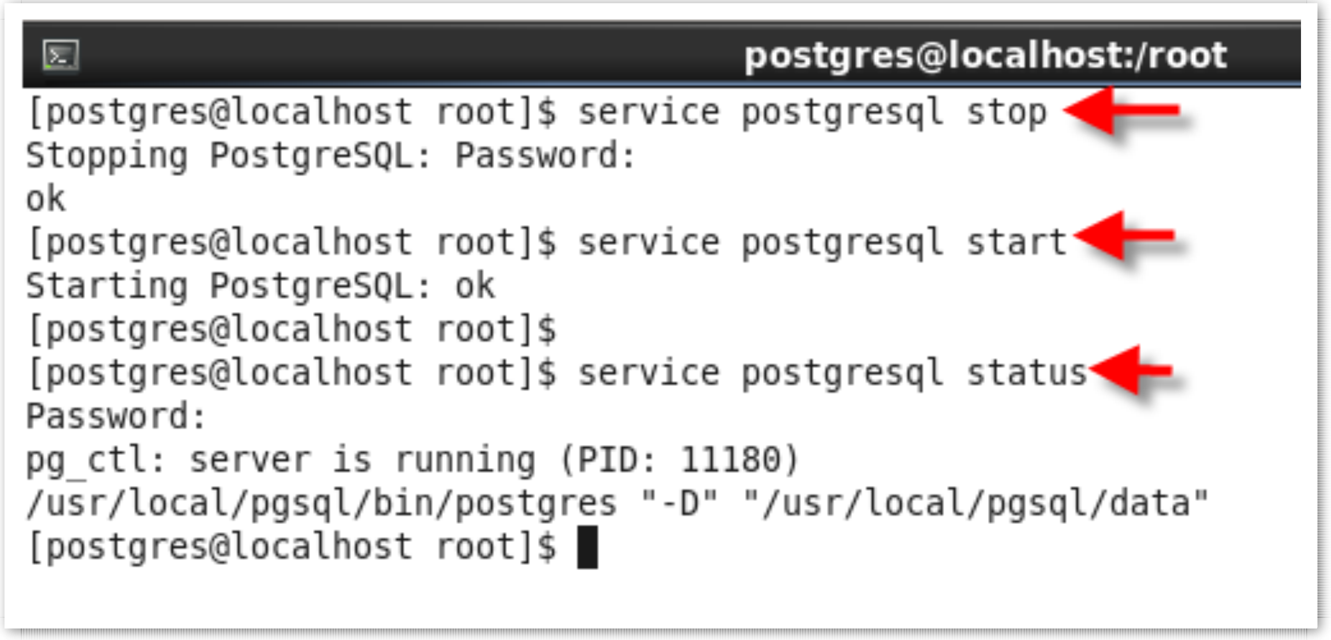


```
postgres@localhost:/root
[postgres@localhost root]$ ls /etc/rc.d/rc0.d/*postgresql
ls: cannot access /etc/rc.d/rc0.d/*postgresql: No such file or directory
[postgres@localhost root]$ ls /etc/rc.d/rc1.d/*postgresql
ls: cannot access /etc/rc.d/rc1.d/*postgresql: No such file or directory
[postgres@localhost root]$ ls /etc/rc.d/rc2.d/*postgresql
ls: cannot access /etc/rc.d/rc2.d/*postgresql: No such file or directory
[postgres@localhost root]$ ls /etc/rc.d/rc3.d/*postgresql
ls: cannot access /etc/rc.d/rc3.d/*postgresql: No such file or directory
[postgres@localhost root]$ ls /etc/rc.d/rc5.d/*postgresql
ls: cannot access /etc/rc.d/rc5.d/*postgresql: No such file or directory
[postgres@localhost root]$ ls /etc/rc.d/rc6.d/*postgresql
ls: cannot access /etc/rc.d/rc6.d/*postgresql: No such file or directory
[postgres@localhost root]$
[postgres@localhost root]$ su -c "chkconfig --add postgresql"
Password:
[postgres@localhost root]$ ls /etc/rc.d/rc2.d/*postgresql
/etc/rc.d/rc2.d/S98postgresql
[postgres@localhost root]$ ls /etc/rc.d/rc3.d/*postgresql
/etc/rc.d/rc3.d/S98postgresql
[postgres@localhost root]$ ls /etc/rc.d/rc5.d/*postgresql
/etc/rc.d/rc5.d/S98postgresql
[postgres@localhost root]$ ls /etc/rc.d/rc6.d/*postgresql
/etc/rc.d/rc6.d/K02postgresql
[postgres@localhost root]$ ls /etc/rc.d/rc0.d/*postgresql
/etc/rc.d/rc0.d/K02postgresql
[postgres@localhost root]$
```

Figure 11

Now, to start and stop PostgreSQL, run the following commands:

service postgresql stop
service postgresql start

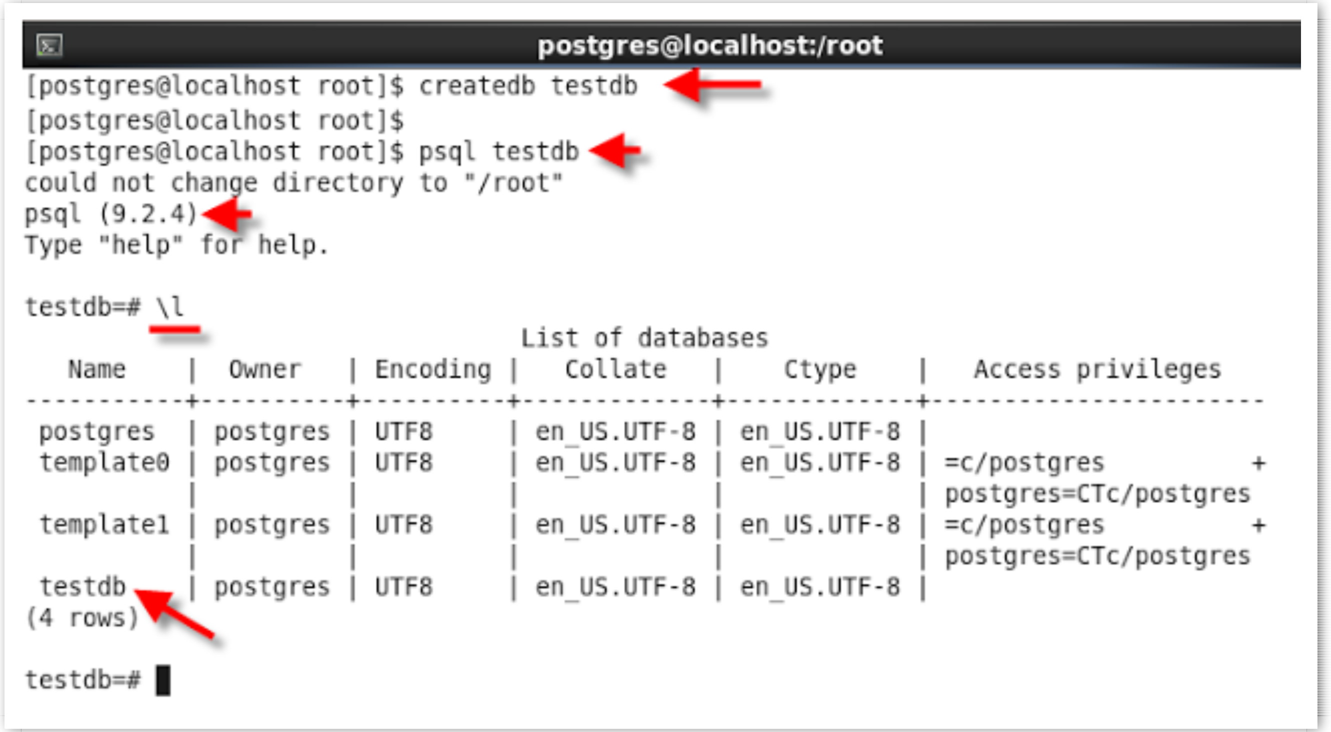


```
postgres@localhost:/root
[postgres@localhost root]$ service postgresql stop
Stopping PostgreSQL: Password:
ok
[postgres@localhost root]$ service postgresql start
Starting PostgreSQL: ok
[postgres@localhost root]$
[postgres@localhost root]$ service postgresql status
Password:
pg_ctl: server is running (PID: 11180)
/usr/local/pgsql/bin/postgres "-D" "/usr/local/pgsql/data"
[postgres@localhost root]$
```

Figure 12

14. Let's create a test database, we need it when we want to try to connect from client side(another machine) to database server:

createdb testdb
psql testdb



```
postgres@localhost:/root
[postgres@localhost root]$ createdb testdb
[postgres@localhost root]$
[postgres@localhost root]$ psql testdb
could not change directory to "/root"
psql (9.2.4)
Type "help" for help.

testdb=# \l
                                  List of databases
   Name    |  Owner   | Encoding |  Collate    |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 testdb    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(4 rows)

testdb=#
```

Figure 13

15. In order to connect to database server through network from client side, we need to do the followings:

1. Insatll PostgreSQL Client with yum command in the client machine(it's different than server machine):
   yum install postgresql-client
2. Go back to server, and open pg_hba.conf in vi:
   vi /usr/local/pgsql/data/pg_hba.conf
   then, change this line:

```
host    all        all        127.0.0.1/32        trust
```

to whatever your client's ip address is. Or you can say the whole subnet. In this case:

```
host    all        all        192.168.0.2/24        trust
```

3. Next, open postgresql.conf in vi:

```
vi /usr/local/pgsql/data/postgresql.conf
```

and uncomment this line:

```
#listen_addresses = 'localhost'
```

and change 'localhost' to the ip address of server, in this case:

```
listen_addresses = '192.168.0.1'
```

4. Open the PostgreSQL server port, run the below command:

```
su -c "iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT"
```

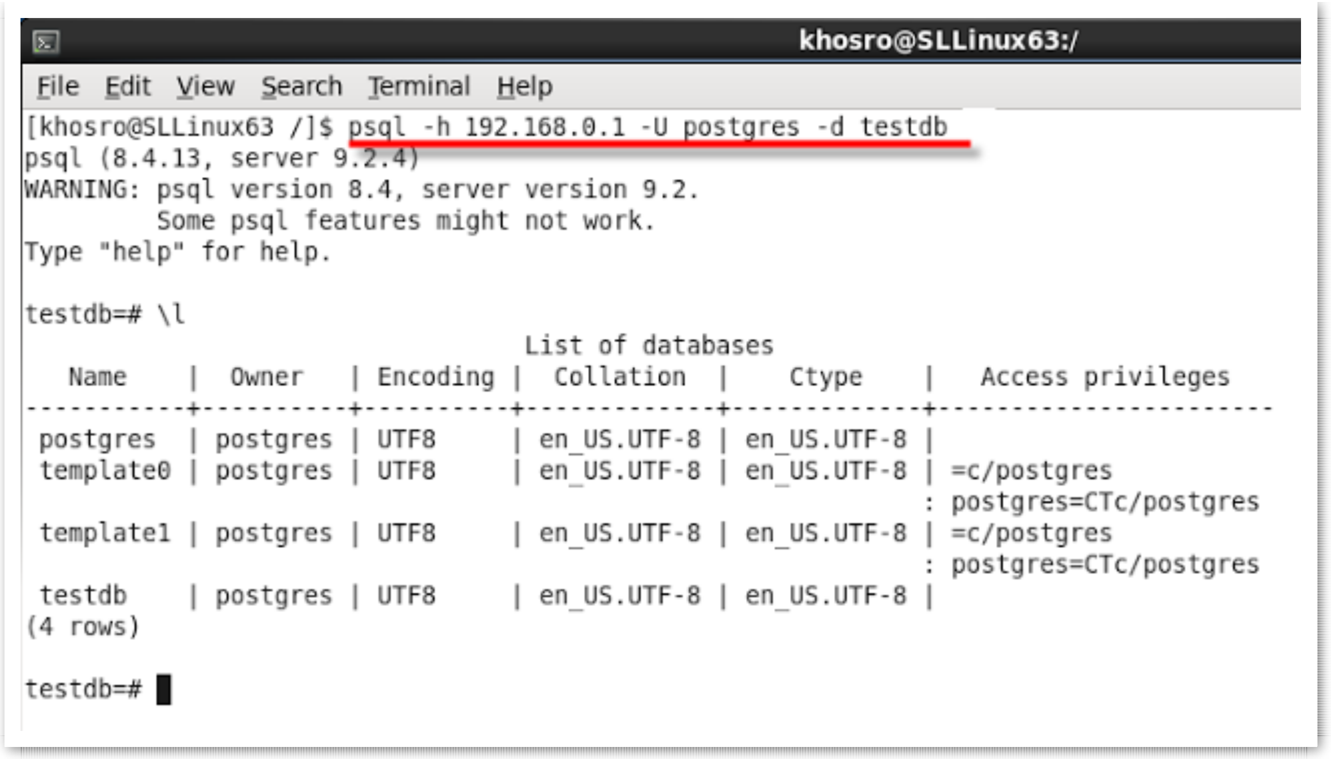
5. Restart postgresql service:

```
service postgresql stop
service postgresql start
```

6. Finally, try to connect to server by the following command in clinet machine:

```
psql -h 192.168.0.1 -U postgres -d testdb
```

which
-h means host
-U means user
-d means database name



```
[khosro@SLLinux63 /]$ psql -h 192.168.0.1 -U postgres -d testdb
psql (8.4.13, server 9.2.4)
WARNING: psql version 8.4, server version 9.2.
         Some psql features might not work.
Type "help" for help.

testdb=# \l
                                  List of databases
   Name    |  Owner   | Encoding |  Collation  |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
                                                             : postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
                                                             : postgres=CTc/postgres
 testdb    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(4 rows)

testdb=#
```

Figure 14

And that's all. I am going to post more blog about PostgreSQL since I've been liked it so far. Hope you enjoyed.

Khosro Taraghi

Posted by Khosro Taraghi at 6:14 PM

G+1 Recommend this on Google

6 comments:

Zahid Rahman September 2, 2013 at 4:06 AM

Waoou... What a tutorial you made? I think any body will be clear after seen this. Just not only looking nice , also
more helpful.
Emergency Exchange Support

Reply

krizna July 29, 2014 at 3:07 AM

Thanks a lot Centos tutorials

Reply

Mehul Patel August 31, 2014 at 9:02 PM

nice post. I will try this and I hope it help me. Thanks.

Reply

Muthu April 21, 2015 at 3:15 AM

This is excellent tutorial. It helped to setup Postgres on deferent versions of Linuxes.

Reply

pavuluri santhi June 23, 2015 at 4:17 AM

Well explained the matter and you can find more relevant content on the same here below.

Reading data from CSV file and writing it into the PostgreSQl database in C++

Reply

derty77 November 18, 2015 at 3:18 PM

werwert

Reply

Enter your comment...

Comment as: Select profile...

Publish **Preview**

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)