Installing PostgreSQL on Red Hat Enterprise Linux / Fedora Core

Last Update: Mon Sep 03, 2007

Written by
Devrim GÜNDÜZ (PGDG RPM Maintainer) devrim@PostgreSQL.org and
Lamar Owen (PGDG RPM Maintainer) lowen@pari.edu

(All contributions are welcome. Please feel free to drop us e-mail about your comments.)

This article will talk about the details of installing PostgreSQL Database Server on Red Hat Enterprise Linux / Fedora Core. RPMs are ready-to-install binary files for many Linux distributions. Please note that this article is written for 8.2 and above, so some information may not fit your version.

General information

PostgreSQL Global Development Group (PGDG) builds RPMs for various Linux distributions. At the time of this writing, we have RPMs and SRPMs for the following platforms[1]:

- Red Hat Enterprise Linux 3, 4 and 5
- Fedora Core 6,7,8

On PgFoundry, there is a project for building RPMs:

`http://pgfoundry.org/projects/pgsqlrpms`

Furthermore, there are -hackers, -general and -announce mailing lists. All discussions are open to everyone.

Obtaining the RPMs

The RPMs are available on PostgreSQL FTP site and all of its mirrors. For your convenience, you may try using web interface of PostgreSQL FTP site. That site will suggest you the suitable mirror(s) for you. The URL is:

`http://www.PostgreSQL.org/ftp`

1 For x86 and x86_64 architectures.

Which RPM for what purpose?

PGDG ships the following RPMs:

postgresql-libs	The postgresql-libs package provides the essential shared libraries for any PostgreSQL client program or interface. You will need to install this package to use any other PostgreSQL package or any clients that need to connect to a PostgreSQL server.
postgresql	If you want to manipulate a PostgreSQL database on a local or remote PostgreSQL server, you need this package. You also need to install this package if you're installing the postgresql-server package.
postgresql-contrib	The postgresql-contrib package contains contributed packages that are included in the PostgreSQL distribution.
postgresql-devel	The postgresql-devel package contains the header files and libraries needed to compile C or C++ applications which will directly interact with a PostgreSQL database management server and the ecpg Embedded C Postgres preprocessor. You need to install this package if you want to develop applications which will interact with a PostgreSQL server.
postgresql-docs	The postgresql-docs package includes the SGML source for the documentation as well as the documentation in PDF format and some extra documentation. Install this package if you want to help with the PostgreSQL documentation project, or if you want to generate printed documentation.
postgresql-server	The postgresql-server package includes the programs needed to create and run a PostgreSQL server, which will in turn allow you to create and maintain PostgreSQL databases. You should install postgresql-server if you want to create and maintain your own PostgreSQL databases and/or your own PostgreSQL server. You also need to install the postgresql package and its requirements.
postgresql-tcl	The postgresql-tcl package contains the Pgtcl client library and its documentation.
postgresql-jdbc	The postgresql-jdbc package includes the .jar files needed for Java programs to access a PostgreSQL database.
postgresql-plperl	The postgresql-plperl package contains the the PL/Perl procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-pltcl	The postgresql-plperl package contains PL/Tcl procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-plpython	The postgresql-plpython package contains the PL/Python procedural language for the backend. PL/pgSQL is part of the core server package.
postgresql-python	The postgresql-python package includes a module for developers to use when writing Python code for accessing a PostgreSQL database.
postgresql-test	The postgresql-test package includes the sources and pre-built binaries of various tests for the PostgreSQL database management system, including regression tests and benchmarks.

Which packages should I use?

If you feel lazy about reading the descriptions above, here is a shortcut for minimal scenarios:

- If you want to run a PostgreSQL server, install postgresql-libs, postgresql and postgresql-server.
- If you want to run a client, install postgresql-libs and postgresql rpms.

For some cases, you might just want to install postgresql-libs for some packages like PHP.

Package names also include version and architecture information. Official PostgreSQL Global Development Group RPM's have a 'PGDG' after the release number. Other RPMset's as distributed with Linux distributions may have a different release number and initials. The version numbering is the same as PostgreSQL.

It is preferable for the distribution-specific set to be the one used, as the PGDG set is intentionally generic. So, if your distro has a set of RPMs, use them in preference. If you want to stay up-to-date on the PostgreSQL core itself, use the PGDG generic set -- but understand that it is a GENERIC set.

These RPMs no longer support any sort of upgrading process other than that documented in the regular documentation. That is, you must dump, upgrade, initdb, and restore your data. You must remove the old server subpackage, install the new package and restore the data from dump.

RPM File Locations

To be in compliance with the Linux FHS, the PostgreSQL PGDG RPMs install files in a manner not consistent with most of the PostgreSQL documentation. According to the standard PostgreSQL documentation, PostgreSQL is installed under the directory /usr/local/pgsql, with executables, source, and data existing in various subdirectories.

Different distributions have different ideas of some of these file locations. In particular, the documentation directory can be /usr/doc, /usr/doc/packages, /usr/share/doc, /usr/share/doc/packages, or some other similar path. The Red Hat / Fedora Core locations are listed below:

Executables	`/usr/bin`
Libraries	`/usr/lib`
Documentation	`/usr/share/doc/postgresql-x.y.z` `/usr/share/doc/postgresql-x.y.z/contrib`
Contrib	`/usr/share/pgsql/contrib`
Data	`/var/lib/pgsql/data`
Backup area	`/var/lib/pgsql/backup`
Templates	`/usr/share/pgsql`
Procedural Languages	`/usr/lib/pgsql`
Development Headers	`/usr/include/pgsql`
Other shared data	`/usr/share/pgsql`
Regression tests	`/usr/lib/pgsql/test/regress` (in the -test package)

Executables	`/usr/bin`
Documentation SGML	`/usr/share/doc/postgresql-docs-x.y.z`

The above list references the Red Hat / Fedora Core structure. These locations may change for other distributions. Use of 'rpm -ql' for each package is recommended as the 'Official' location source.

These RPMs are designed to be LSB-compliant -- if you find this not to be the case, please let us know by way of the pgsqlrpms-hackers@PgFoundry.org mailing list.

Installing and Upgrading PostgreSQL RPMs

Installing PGDG RPMs are as easy as installing any RPMs

```
rpm –ivh package_name.version.arch.rpm
```

Unless specified, on minor release upgrades (i.e., upgrading from 8.2.0 to 8.2.1 or 8.2.4, etc[2]), you may use usual RPM upgrade process:

```
rpm –Uvh package_name.version.arch.rpm
```

Please note that on every new major version upgrade, you have to follow dump/reload sequence. However please ***don't forget to read the Release Notes before upgrading because on some cases minor versions may need a dump/reload process. At those times if you use -U switch, then you will probably lose data! Please refer to "How do I perform a major upgrade?" section.***

Many of the RPMs are signed with the PGP key of the package builder. Directories contain a CURRENT_MAINTAINER file which includes the name/email of the package builder and link to their PGP key. You might want to import the signature before you'll install the RPM:

```
rpm –import http://link/to/the/pgp/key
```

Unless this is not done, you can install/upgrade the RPMs but you'll be thrown a warning.

You may subscribe to pgsqlrpms-general@pgfoundry.org list for more details.

Removing RPMS

You might want to take a full dump (and possibly a filesystem level backup) before removing RPMs. Removing an PostgreSQL RPM is as easy as removing any RPM:

```
rpm –e package_name
```

Before the removal of server package, if there are any running processes, all are stopped. You don't need to stop it.

Starting PostgreSQL for the first time and init script

Red Hat Linux uses the System V Init package. A startup script for PostgreSQL is provided in the server package, as /etc/rc.d/init.d/postgresql. To start the postmaster, with sanity checking, as root, run

```
service postgresql start
```

To shut the postmaster down,

```
service postgresql stop
```

There are other parameters to this script -- execute 'service postgresql' for a listing.

On some cases you might want to edit this init file. For example, you might want to pass a –locale=... parameter to initdb, etc.

2 8.1 is the major version number, and 0,1,3, etc are minor version numbers

This script does the following sets defaults for configuration variables and then performs the given action (stop, start, etc). During the startup, first the script checks whether the database cluster has been initialized or not. If not, then you must run:

```
service postgresql initdb
```

This option is new as of 8.2 . Before 8.2, `service postgresql start` was calling initdb when the cluster was not initalized.

Then the service is started as usual.

You may also reload the server for some changes to take effect; but please look at the PostgreSQL documentation for the conditions that the database server needs a restart or a reload.

Starting PostgreSQL automatically at system startup

To get this script to run at system startup run:

```
chkconfig postgresql on
```

and the proper symlinks will be created. See the chkconfig man page for more information. Note that this is manual -- while the startup script can include tags to allow chkconfig to automatically perform the symlinking, this is not done at this time.

How do I perform a major upgrade?

Currently, PostgreSQL RPMs does not provide a data upgrade feature among major releases (or clearly, upgrades that require an initdb.) This work in under progress. In order to upgrade to a major version, you should follow the following steps:

- Take a full dump using pg_dumpall
- You might want to take a filesystem-level backup also. This is intentional.
- Check the backups! (Do it again!)
- Now, stop the database server:

```
/sbin/service postgresql stop
```

- Remove all postgresql rpms. Please note that you'll probably need a --nodeps switch at the end:

```
/bin/rpm -e `/bin/rpm -qa|grep postgresql^``
/bin/rpm -e `/bin/rpm -qa|grep postgresql^`` --nodeps
```

During the removal of packages, some scripts will be run to remove postgres user and to uninstall postgresql service, etc.

Please note that removing of postgresql-server RPM will not also remove /var/lib/pgsql

- Remove[3] database cluster:

```
/bin/rm -rf /var/lib/pgsql
```

- Install new RPM sets
- Start database server

```
/sbin/service postgresql start
```

- Edit conf files, if you need.
- Reload the data to the new server (You may need to edit your data).

... and you're done!

3 Better: /bin/rm -rf ~postgres or best take a filesystem-level backup of this directory by using /bin/mv for example

Rebuilding from Source RPM

If your distribution is not supported by the binary RPM's from PostgreSQL.org, you will need to rebuild from the source RPM. Download the .src.rpm for this release. You will need to be root to rebuild, unless you have already set up a non-root build environment.

Install the source RPM with rpm -i, then CD to the rpm building area (on Red Hat or Fedora Core this is /usr/src/redhat by default). You will have to have a full development environment to rebuild the full RPM set.

This release of the RPMset includes the ability to conditionally build sets of packages. The parameters, their defaults, and the meanings are:

beta	0	#build with cassert and do not strip the binaries
pltcl	1	#build the postgresql-pltcl package.
jdbc	1	#build the postgresql-jdbc package.
plperl	1	#build the postgresql-plperl package.
test	1	#build the postgresql-test package.
plpython	1	#build the postgresql-plpython package.
pltcl	1	#build the pltcl portion of the postgresql-pl package.
plperl	1	#build the plperl portion of the postgresql-pl package.
ssl	1	#use OpenSSL support.
kerberos	1	#use Kerberos 5 support.
nls	1	#build with national language support.
pam	1	#build with PAM support.
runselftest	1	#do "make check" during the build.
xml	1	#build contrib/xml2
pgfts	0	#Build with –enable-thread-safety

To use these defines, invoke a rebuild like this:

```
rpm --rebuild --define 'plperl 0' --define 'pltcl 0' \
      --define 'test 0' --define 'runselftest 1' \
      --define 'kerberos 0'   postgresql-8.2.4-1PGDG.src.rpm
```

This line would disable the plperl, pltcl, and test subpackages, enable the regression test run during build, and disable kerberos support. You might need to disable runselftest if there is an installed version of PostgreSQL that is a different major version from what you are trying to build. The self test tends to pick up the installed libpq.so shared library in place of the one being built :-(, so if that isn't compatible the test will fail. Also, you can't use runselftest when doing the build as root.

More of these conditionals may be added/removed in the future.

Contrib Files

The contents of the contrib tree are packaged into the -contrib subpackage and are processed with make and make install. There is documentation in /usr/share/doc/postgresql-contrib-VERSION for these modules. Most of the modules are in /usr/lib/pgsql for loadable modules, and binaries are in /usr/bin. In the future these files may be split out, depending upon function and dependencies.

More Information

You can get more information at http://www.postgresql.org

Please help make this packaging better -- let us know if you find problems, or better ways of doing things. You can reach us by e-mail at pgsqlrpms-hackers@PgFoundry.org.