

postgresql clustering and Debian

Anand Kumria

wildfire@progsoc.org

1. Introduction

1.1. What

Each postgresql installation can contain multiple, independent, databases. This is known as a postgresql cluster. The databases within the cluster all share configuration (such as how authentication occurs, what port the postmaster is listening on) as well as any configured storage areas (e.g. tablespaces).

1.2. Why

You may want to have the set of databases but in different instances (e.g. *production*, *test*, *dev*). Rather than having to manipulate the database name across your application, you can connect to each cluster just by specifying a particular port.

This kind of setup can also allow you to use the same binaries for testing as you are for production and also know that the hardware you have set aside for development is also capable of handling the load that a testing places on it.

2. Source or Distribution?

Invariably when installing postgresql at a client's site you'll need to choose between a source-based approach, or using what the distribution provides.

2.1. Source

Positives

- You can choose what version to deploy
- You can choose when (if) to upgrade

- You can choose which optional languages should be supported

Negatives

- You have to keep up to date with security issues
- It is a custom installation which needs to be documented for others
- You might mis-specify compilation options with effects not apparent for a while

2.2. Binary Distributions

Positives

- Known to work (reasonably) well
- Integrates well with other parts of distribution
- Security updates are *free*
- Generally all optional languages are available

Negatives

- Have to deploy available version of distribution
- Required to update / upgrade when the distribution does
- May have too many unfamiliar components (e.g. TLS, IPv6)

3. How

All (future) versions of Debian (testing, unstable) have the package *postgresql-common*, which is installed when postgresql-8.1 is brought in. So `apt-get install <postgresql>-<version>`, where version is either 7.4 or 8.1 to get access to the clustering system.

3.1. Commands

All standard binary commands are wrapped with *pg_cluster/PgCommon.pm*

This is a perl script (library), which interprets a pseudo argument to the standard commands `--cluster [version]/cluster`.

For example, to connect to postgresql 7.4's main cluster:

```
psql --cluster 7.4/main -l
```

If no cluster is specified, the version of postgresql listening on 5432 and the *main* cluster are the defaults.

As well as wrapping the logical commands (such as `psql`, `createdb`, etc.), some additional commands are provided to manage things at the cluster level.

For example, on a machine with both 7.4 and 8.1 installed:

```
$ pg_lsclusters
Version Cluster  Port Status Owner    Data directory          Log file
7.4      main      5432 online postgres /var/lib/postgresql/7.4/main /var/log/postgresql/pos
8.1      main      5433 online postgres /var/lib/postgresql/8.1/main /var/log/postgresql/pos
```

As you would expect, clusters can be created, upgraded, dropped and control with the commands `pg_createcluster`, `pg_upgradecluster`, `pg_dropcluster` and `pg_ctlcluster` respectively.

3.2. Paths

Configuration information and the actual data need to be stored per-cluster instance. Since we are storing both version and cluster name, the path generated is, unsurprisingly enough, composed of both.

For configuration files:

```
/etc/postgresql/<version>/<cluster>/start.conf
/etc/postgresql/<version>/<cluster>/pg_hba.conf
```

and likewise, for data areas:

```
/var/lib/postgresql/<version>/<cluster>/PG_VERSION
/var/lib/postgresql/<version>/<cluster>/base
```

4. What about Right Now?

While you could the testing and unstable versions of Debian on your production machine — you probably don't want to. For that you can turn to [backports.org](http://www.backports.org) (<http://www.backports.org/>).

[backports.org](http://www.backports.org/) (<http://www.backports.org/>) only allows uploads only from Debian developers and it rebuilds packages across all the supported architectures of a stable release. Thus by using [backports.org](http://www.backports.org/)

(<http://www.backports.org/>) you can get the features of testing with the stability (security updates, etc.) of stable.

Place into `/etc/apt/source.list` the following:

```
deb http://www.backports.org/debian/ sarge-backports main
```

And then into `/etc/apt/preferences`:

```
Package: *  
Pin: release a=sarge-backports  
Pin-Priority: 200
```

Then perform an `apt-get update`, followed by `apt-get install <backport package>` which will download and install the package for you.

If you are running Debian 3.1 and want to migrate to the clustering system herein, then you would do `apt-get install postgresql-7.4 postgresql-client-7.4 postgresql-contrib-7.4`. If you want to utilise either Perl, Python, R, Ruby or TCL then you would also do `apt-get install postgresql-plperl-7.4 postgresql-plpython-7.4 postgresql-7.4-plr postgresql-7.4-plruby postgresql-pltcl7.4`.

You can then upgrade to *postgresql-8.1* at your own pace by installing the same packages but substituting 8.1 instead of 7.4.