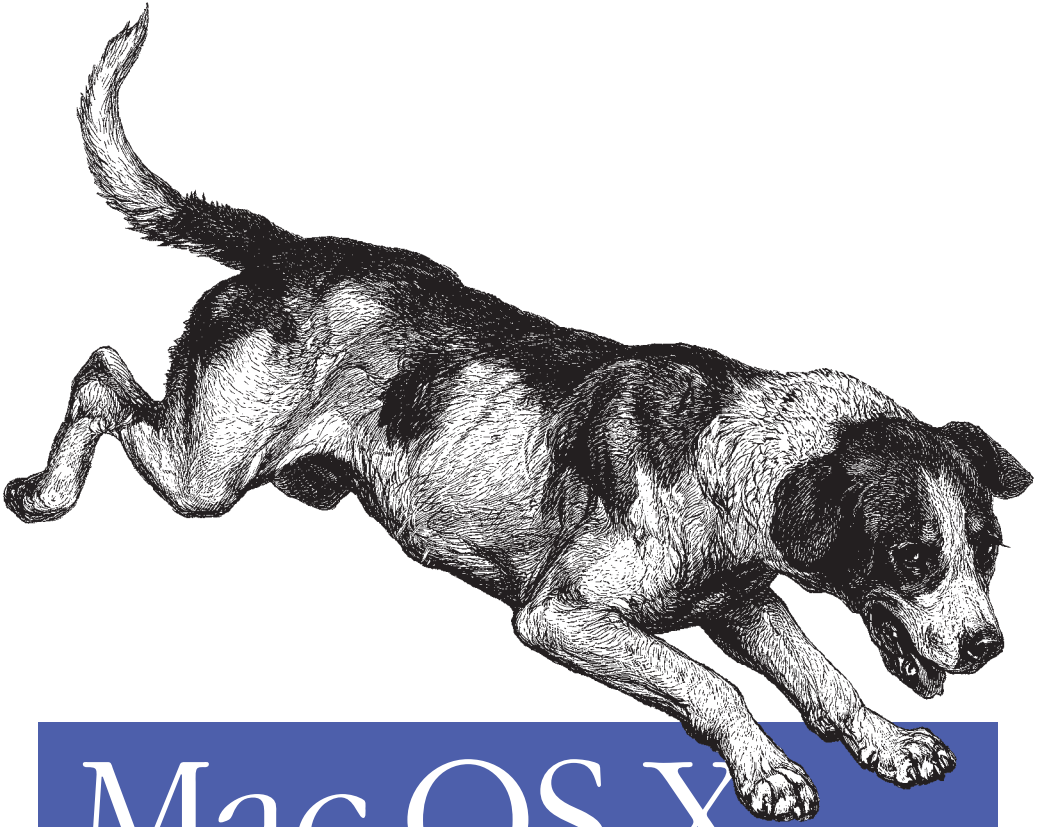




Developer
Connection

Recommended Title



Mac OS X Panther *for* Unix Geeks

O'REILLY®

Brian Jepson & Ernest E. Rothman

MySQL and PostgreSQL

Although there are some great binary distributions for MySQL and PostgreSQL, both build out of the box on Mac OS X. This chapter describes how to install them from source and get them set up so you can start playing with them. Fink is a good first stop for MySQL or PostgreSQL, since you can use it to install a binary build or compile from source.

You can also get MySQL as a binary package from MySQL AB (<http://www.mysql.com>), as well as Server Logistics (<http://www.serverlogistics.com/>). Server Logistics offers a selection of open source packages, one of which is Complete MySQL (<http://www.serverlogistics.com/mysql.php>), which includes the MySQL server, a System Preferences pane for MySQL, ODBC/JDBC drivers, and documentation.

MySQL

To get the source distribution of MySQL, download the latest tarball from <http://www.mysql.com/downloads/>. At the time of this writing, the latest production release was the 4.0.x series; we downloaded *mysql-4.0.16.tar.gz*.

Compiling MySQL

To compile MySQL from source:

1. Extract the tarball:

```
$ cd ~/src
$ tar xvfz ~/Desktop/mysql-4.0.16.tar.gz
```

2. Change to the top-level directory that *tar* created and run the configure script. We suggest specifying a prefix of */usr/local/mysql* so it stays out the way of any other binaries you have in */usr/local*.

```
$ cd mysql-4.0.16
$ ./configure --prefix=/usr/local/mysql
```

3. Next, type *make* to compile MySQL. Go get a few cups of coffee (compiling could take 30 minutes or more).

Installing MySQL

If the compilation succeeded, you're ready to install MySQL. If not, you should first search the MySQL mailing list archives (<http://lists.mysql.com>) to see if anyone has reported the same problem you experienced, and whether a fix is available (otherwise, you should submit a bug report). If you're having a lot of trouble here, you may want to install one of the binary packages. If everything went OK, you can now install MySQL:

1. Run *make install* as root:

```
$ sudo make install
```

2. Install the default databases:

```
$ sudo ./scripts/mysql_install_db
```

3. Set permissions on the MySQL directories:

```
$ sudo chown -R root /usr/local/mysql
```

```
$ sudo chown -R mysql /usr/local/mysql/var
```

```
$ sudo chgrp -R mysql /usr/local/mysql
```

4. Install a configuration file (*my-small.cnf*, *my-medium.cnf*, *my-large.cnf*, or *my-huge.cnf*):

```
$ sudo cp support-files/my-medium.cnf /etc/my.cnf
```

5. Now you're ready to install a startup script for MySQL. See "Startup Items" in Chapter 2 for a sample MySQL startup script. (For now, leave out the *--password=password* from the startup script. You can add it back in, with the appropriate password, after you set the MySQL root password.) After you've created the startup script, start MySQL:

```
$ sudo SystemStarter start MySQL
```

Configuring MySQL

Next, you need to configure MySQL. At a minimum, set the root user's password and create a user and a working database for that user. Before using MySQL, add the following line to your *.bash_profile* and start a new Terminal window to pick up the settings:

```
export PATH=$PATH:/usr/local/mysql/bin
```

To set the root password and create a new user:

1. Use *mysqladmin* to set a password for the root user (qualified as *root@localhost* and just plain old *root*). When you enter the second line,

there will be a root password in place, so you need to use `-p`, and you'll be prompted for the password you created on the first line:

```
$ mysqladmin -u root password 'password'
$ mysqladmin -u root -p -h localhost password 'password'
Enter password: *****
```

2. Create a database for your user (you'll be prompted for the *mysql* root user's password):

```
$ mysqladmin -u root -p create dbname
Enter password: *****
```

3. Log into the *mysql* shell as root, and grant full control over that database to your user, qualified as *user@localhost* and just the username alone (the `->` prompt indicates that you pressed return without completing the command, and the *mysql* shell is waiting for more input):

```
$ mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12 to server version: 4.0.16-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> GRANT ALL PRIVILEGES ON dbname.* TO username@localhost
-> IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.08 sec)

mysql> GRANT ALL PRIVILEGES ON dbname.* TO username
-> IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Playing with MySQL

You should be able to log in to MySQL as the user defined in the previous section, and do whatever you want within your database:

```
$ mysql -u username -p dbname
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16 to server version: 4.0.16-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE TABLE foo (bar CHAR(10));
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO foo VALUES('Hello');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO foo VALUES('World');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM foo;
+-----+
| bar   |
+-----+
| Hello |
| World |
+-----+
2 rows in set (0.00 sec)

mysql> quit
Bye
```

PostgreSQL

To get the source distribution of PostgreSQL, download the latest tarball from one of the mirrors at <http://www.postgresql.org/mirrors-ftp.html>. At the time of this writing, the latest release was 7.4, so we downloaded *postgresql-7.4.tar.bz2*.

Compiling PostgreSQL

Before installing PostgreSQL, you must install readline (<http://www.gnu.org/directory/readline.html>). This program enables support for command-line editing and history in the PostgreSQL shell (*psql*). Use *fink install readline* to install it, if you have Fink installed. You also need the most recent version of *bison* (<http://www.gnu.org/software/bison/bison.html>), which you can obtain with *fink install bison* (double check to ensure that */sw/bin* appears first in your *\$PATH*; this is the default after you've installed Fink).

To compile PostgreSQL from source:

1. Extract the tarball:

```
$ cd ~/src
$ tar xvfj ~/Desktop/postgresql-7.4.tar.bz2
```

2. Change to the top-level directory of the tar and run the *configure* script. We suggest specifying a prefix of */usr/local/pgsql* so it stays out the way of any other binaries you have in */usr/local*.

```
$ cd postgresql-7.4
$ ./configure --prefix=/usr/local/pgsql \
> --with-includes=/sw/include --with-libs=/sw/lib
```

3. Next, type *make* to compile PostgreSQL. Go take a walk around the block while you wait (compiling could take 30 minutes or more).

Installing PostgreSQL

If everything went OK, you're ready to install. If it didn't go OK, check the PostgreSQL mail list archives (<http://www.postgresql.org/lists.html>) to see if anyone has reported the same problem you experienced and whether a fix is available (otherwise, you should submit a bug report).

1. Run *make install* as root:

```
$ sudo make install
```

2. Create the *postgres* group and user (this is the PostgreSQL superuser). Be sure to choose an unused group ID and user ID:

```
$ sudo nload group . <<EOF
> postgres:*:1001:
> EOF
$ sudo nload passwd . <<EOF
> postgres:*:1001:1001::0:0:PostgreSQL:/usr/local/pgsql:/bin/bash
> EOF
```

3. Create the data subdirectory and make sure that the *postgres* user is the owner of that directory:

```
$ sudo mkdir /usr/local/pgsql/data
$ sudo chown postgres /usr/local/pgsql/data
```

4. Use *sudo* to get a shell as the *postgres* user (supply your own password at this prompt):

```
$ sudo -u postgres -s
Password: *****
postgres$
```

5. Run the following commands to initialize the PostgreSQL installation:

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

6. You can now log out of the *postgres* user's shell.

Adding the Startup Item

Now you're ready to create a startup script for PostgreSQL (see "Adding Startup Items" in Chapter 2). First, create the script shown in Example 14-1, save it as */Library/StartupItems/PostgreSQL/PostgreSQL*, and mark it as an executable.

Example 14-1. Startup script for PostgreSQL

```
#!/bin/sh

# Source common setup, including hostconfig.
#
. /etc/rc.common
```

Example 14-1. Startup script for PostgreSQL (continued)

```
StartService( )
{
    # Don't start unless PostgreSQL is enabled in /etc/hostconfig
    if [ "${PGSQL:=-NO-}" = "-YES-" ]; then
        ConsoleMessage "Starting PostgreSQL"
        sudo -u postgres /usr/local/pgsql/bin/pg_ctl \
            -D /usr/local/pgsql/data \
            -l /usr/local/pgsql/data/logfile start
    fi
}

StopService( )
{
    ConsoleMessage "Stopping PostgreSQL"
    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop
}

RestartService( )
{
    # Don't restart unless PostgreSQL is enabled in /etc/hostconfig
    if [ "${PGSQL:=-NO-}" = "-YES-" ]; then
        ConsoleMessage "Restarting PostgreSQL"
        StopService
        StartService
    else
        StopService
    fi
}

RunService "$1"
```

Next, create the following file as */Library/StartupItems/PostgreSQL/StartupParameters.plist*:

```
{
    Description      = "PostgreSQL";
    Provides         = ("PostgreSQL");
    Requires         = ("Network");
    OrderPreference = "Late";
}
```

Then, add the following line to */etc/hostconfig*:

```
PGSQL=-YES-
```

Now PostgreSQL will start automatically when you reboot the system. If you want, you can start PostgreSQL right away with:

```
$ sudo SystemStarter start PostgreSQL
```

Configuring PostgreSQL

Before you proceed, you should add the following line to the *.bash_profile* and start a new Terminal window to pick up the settings (you should also add this to the *postgres* user's *.bash_profile*):

```
export PATH=$PATH:/usr/local/pgsql/bin
```

By default, PostgreSQL comes with weak permissions; any local user can connect to the database without authentication. Before changing anything, you must start a shell as the *postgres* user with *sudo* and stay in this shell until the end of this section:

```
$ sudo -u postgres -s
Password: *****
postgres$
```

To start locking things down and to set up a non-privileged user:

1. Create the *postgres* user's home database

```
$ createdb
```

2. Set a password for the PostgreSQL superuser:

```
postgres$ psql -U postgres -c \  
> "alter user postgres with password 'password';"
```

3. Under the default permissions, any local user can impersonate another user. So, even though you've set a password, it's not doing any good! You should edit */usr/local/pgsql/data/pg_hba.conf* to require MD5 passwords, give the *postgres* user control over all databases, and change the configuration so users have total control over databases that have the same name as their username. To do this, change *pg_hba.conf* to read:

```
# TYPE DATABASE USER IP-ADDR IP-MASK METHOD
local all postgres
local sameuser all md5
host all postgres 127.0.0.1 255.255.255.255 md5
host sameuser all 127.0.0.1 255.255.255.255 md5
host all postgres ::1 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff md5
host sameuser all ::1 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff md5
```

4. Once you've made this change, reload the configuration with *pg_ctl* (from here on in, you'll be prompted for a password when you run *psql* as the *postgres* user):

```
postgres$ pg_ctl -D /usr/local/pgsql/data reload
```

5. Now you're ready to add a normal user. Use the *psql* command to create the user and a database. Because the username and database name are the same, that user will be granted access to the database:

```
postgres$ psql -U postgres -c "create database username;"
Password: *****
CREATE DATABASE
```

```
postgres$ psql -U postgres -c \  
> "create user username with password 'password';"  
Password: *****  
CREATE USER
```

To give more than one user access to a database, create a group with the same name as the database (for example, *create group databasename*), and create users with the *create user* command as shown in step 5. Finally, add each user to the group with this command:

```
alter group databasename add user username
```

Playing with PostgreSQL

After configuring PostgreSQL's security and setting up an unprivileged user, you can log in as that user and play around with the database:

```
$ psql -U username  
Password: *****  
Welcome to psql 7.4, the PostgreSQL interactive terminal.  
  
Type: \copyright for distribution terms  
      \h for help with SQL commands  
      \? for help on internal slash commands  
      \g or terminate with semicolon to execute query  
      \q to quit  
  
username=> CREATE TABLE foo (bar CHAR(10));  
CREATE TABLE  
username=> INSERT INTO foo VALUES('Hello');  
INSERT 17148 1  
username=> INSERT INTO foo VALUES('World');  
INSERT 17149 1  
username=> SELECT * FROM foo;  
      bar  
-----  
      Hello  
      World  
(2 rows)  
  
username-> \q
```

For more information on building and using PostgreSQL, see *Practical PostgreSQL* by John C. Worsley and Joshua D. Drake (O'Reilly). *Practical PostgreSQL* covers installing, using, administrating, and programming PostgreSQL.

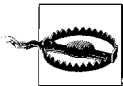
PHP and Perl

On Mac OS X Panther, MySQL support is built in to PHP. If you want PostgreSQL support, you must reinstall PHP from source.

You can install general database support in Perl by installing the DBI module with the *cpan* utility (see Chapter 10). After that, you can install the DBD::mysql module for MySQL-specific support, and DBD::Pg for PostgreSQL-specific support. Because there are some steps to these installations that the *cpan* utility can't handle, you should download the latest builds of these modules from <http://www.cpan.org/modules/by-module/DBD/> and install them manually. Be sure to check the *README* files, since some aspects of the configuration may have changed.

The DBD::mysql module requires a database in which to perform its tests. You can use the database and username/password that you set up earlier in *Configuring MySQL*. To install DBD::mysql, you must first generate the *Makefile*, compile the code, test it, and then install the module if the test run is successful. For example:

```
$ perl Makefile.PL --testdb=dbname --testuser=username \  
> --testpassword=password  
$ make  
$ make test  
$ sudo make install
```



At the time of this writing, DBD::mysql failed to compile on Panther. The short description of the fix is to replace all occurrences of *MACOSX* with *env MACOSX* in the *Makefile* (after generating it with *perl Makefile.PL*).

For a complete description, see the *Forwarding Address: OS X* weblog entry at http://www.saladwithsteve.com/osx/2003_11_01_archive.html#106802251200041735.

As with DBD::mysql, the DBD::Pg module needs a directory to perform its tests. If you'd like, you can use the database, username, and password that you set up earlier when configuring PostgreSQL.

You must first generate the *Makefile*, compile the code, set up environment variables that specify the database, username, and password, and then run the tests. If the tests run successfully, you can install DBD::Pg:

```
$ perl Makefile.PL  
$ make  
$ export DBI_DSN=dbi:Pg:dbname=username  
$ export DBI_USER=username  
$ export DBI_PASS=password  
$ make test  
$ sudo make install
```