Installing a private **PostgreSQL** instance on WebFaction

By David Chin
@ tinybooks.briefnotes.net

Installing a private PostgreSQL instance on WebFaction

Yet another actionable, practical, to-the-point, and screencapture-heavy TinyBooks@BriefNotes.net publication that explains all the important stuff in 10 chapters or less.

David Chin

This book is for sale at http://leanpub.com/installing-a-private-postgresql-instance-on-webfaction

This version was published on 2016-02-23

Contents

Overview

This TinyBooks@BriefNotes.net[1] publication explains in fair detail the following steps to install a private instance of PostgreSQL on WebFaction. The procedure are based in part on the official PostgreSQL Installation Procedure[2]:

1. Create a WebFaction custom app in order to reserve a port number which allows you to communicate with a running instance of your installed PostgreSQL.
2. Update the file `.bash_profile` to have the `PATH` environment variable point at the to-be-installed PostgreSQL's `bin` directory.
3. Copy the download link of the compressed file containing the version of PostgreSQL that you wish to install.
4. Download the compressed file from Chapter 3 to WebFaction and extract its contents.
5. Install PostgreSQL with `configure`, `make` & `make install`.
6. Create and initialize a data directory to hold the PostgreSQL database storage files and logfile.
7. Start up the PostgreSQL server and monitor its status.
8. Create a default database named after your WebFaction username.
9. Run some simple database tests.
10. Learn how to shut down the database.

[1] tinybooks.briefnotes.net
[2] http://www.postgresql.org/docs/current/static/install-procedure.html

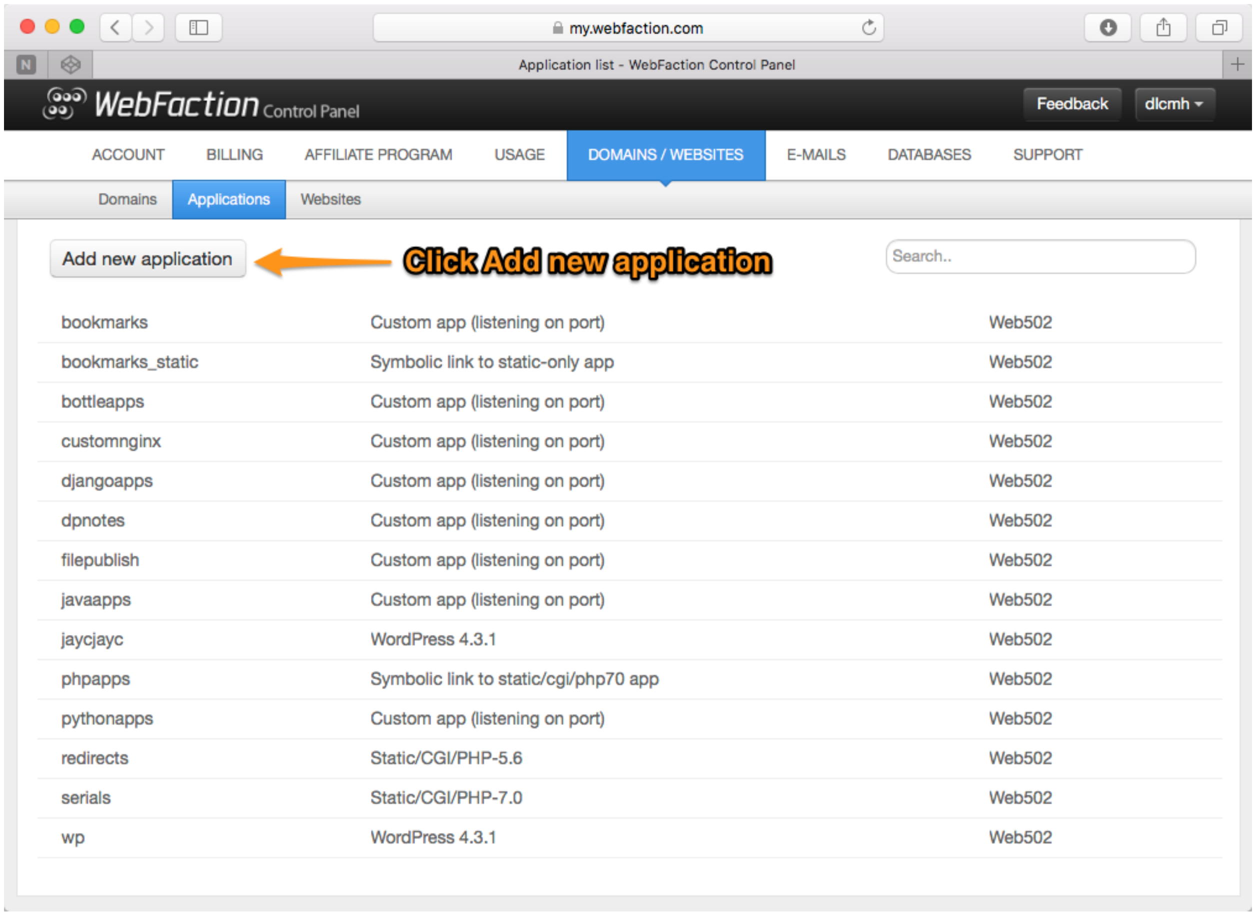
1. Create a WebFaction custom app

We create a custom application in order to reserve a **port number** from WebFaction.

This port will be required later in Chapter 5, and all communications with our PostgreSQL database will take place through this port.

1.1 Add a new application

1. Log in to the WebFaction Control Panel with your Username and Password at https://my.webfaction.com/new-application/[1].
2. Click **DOMAINS/WEBSITES** > **Applications** in the navigation bar.
3. Click the **Add new application** button.



bookmarks	Custom app (listening on port)	Web502
bookmarks_static	Symbolic link to static-only app	Web502
bottleapps	Custom app (listening on port)	Web502
customnginx	Custom app (listening on port)	Web502
djangoapps	Custom app (listening on port)	Web502
dpnotes	Custom app (listening on port)	Web502
filepublish	Custom app (listening on port)	Web502
javaapps	Custom app (listening on port)	Web502
jaycjayc	WordPress 4.3.1	Web502
phpapps	Symbolic link to static/cgi/php70 app	Web502
pythonapps	Custom app (listening on port)	Web502
redirects	Static/CGI/PHP-5.6	Web502
serials	Static/CGI/PHP-7.0	Web502
wp	WordPress 4.3.1	Web502

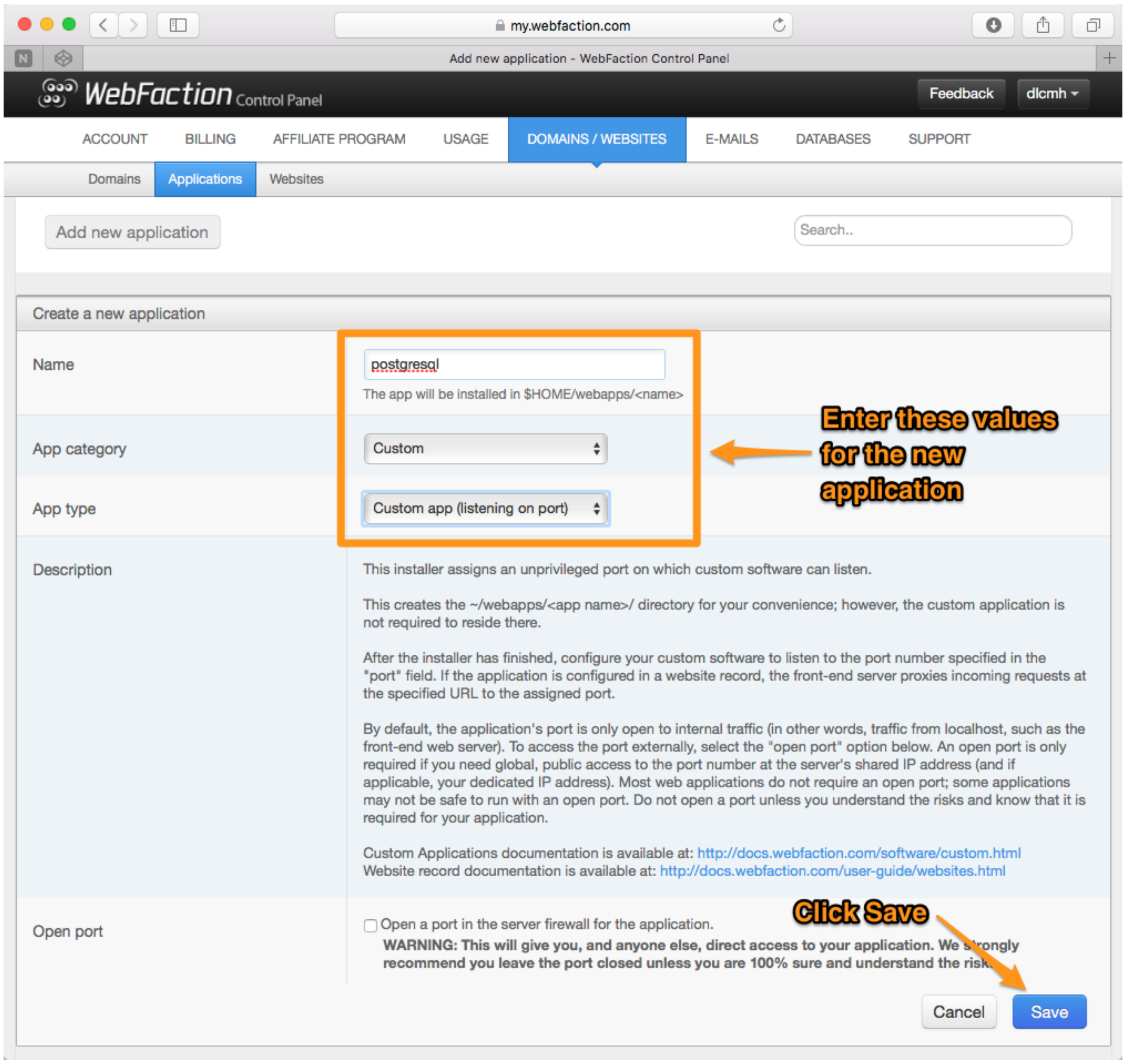
DOMAINS/WEBSITES > **Applications**

[1] https://my.webfaction.com/new-application/

1.2 Enter these values for the new custom application

- **Name**: postgresql (feel free to choose any meaningful name)
- **App category**: Custom
- **App type**: Custom app (listening on port)

Click the Save button when done.



Create a new application

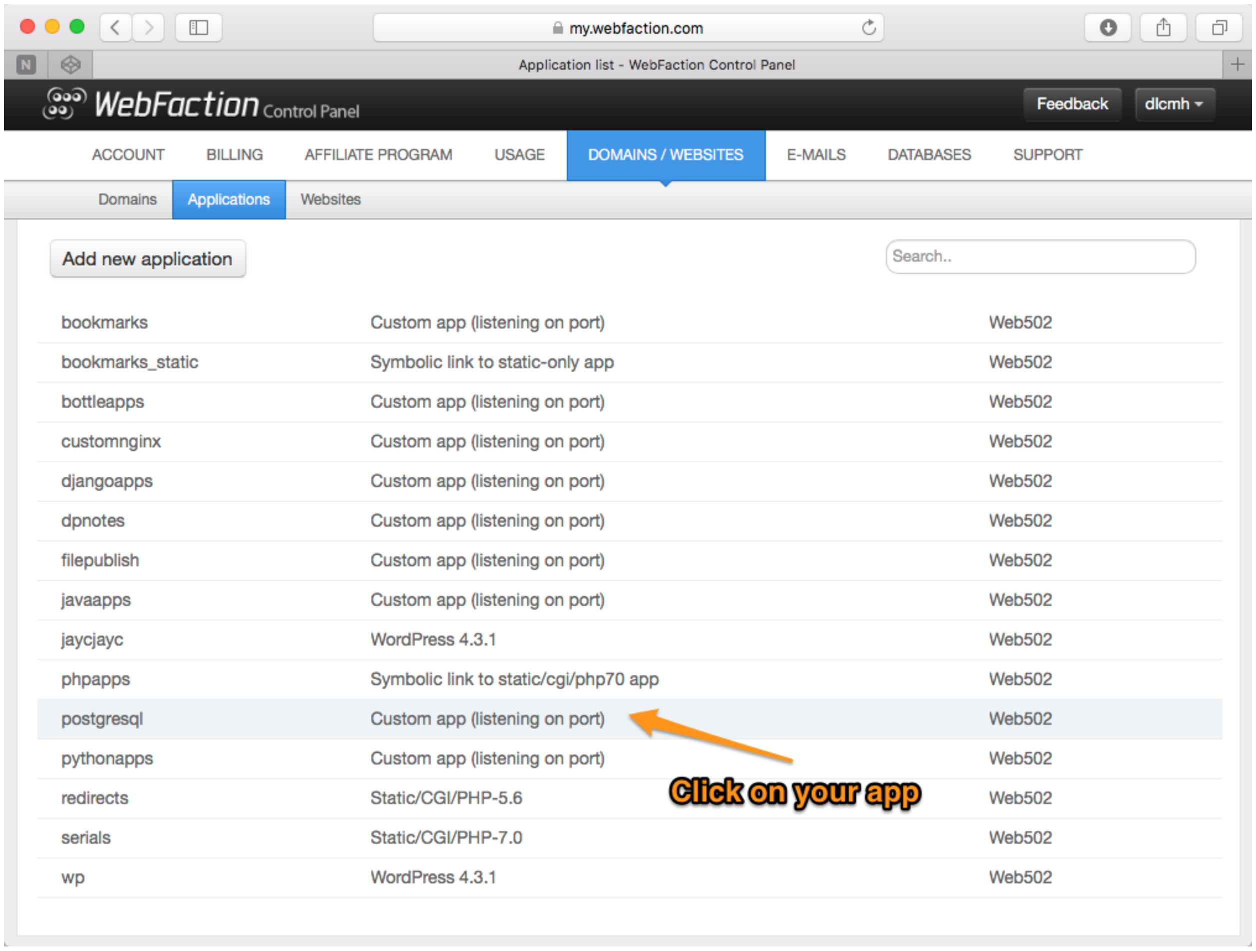
Name	postgresql The app will be installed in $HOME/webapps/<name>
App category	Custom
App type	Custom app (listening on port)
Description	This installer assigns an unprivileged port on which custom software can listen. This creates the ~/webapps/<app name>/ directory for your convenience; however, the custom application is not required to reside there. After the installer has finished, configure your custom software to listen to the port number specified in the "port" field. If the application is configured in a website record, the front-end server proxies incoming requests at the specified URL to the assigned port. By default, the application's port is only open to internal traffic (in other words, traffic from localhost, such as the front-end web server). To access the port externally, select the "open port" option below. An open port is only required if you need global, public access to the port number at the server's shared IP address (and if applicable, your dedicated IP address). Most web applications do not require an open port; some applications may not be safe to run with an open port. Do not open a port unless you understand the risks and know that it is required for your application. Custom Applications documentation is available at: http://docs.webfaction.com/software/custom.html Website record documentation is available at: http://docs.webfaction.com/user-guide/websites.html
Open port	☐ Open a port in the server firewall for the application. **WARNING: This will give you, and anyone else, direct access to your application. We strongly recommend you leave the port closed unless you are 100% sure and understand the risk.**

Values for the new application

1.3 Write down the assigned port number

WebFaction assigned a port number at the point of creating the new custom app.

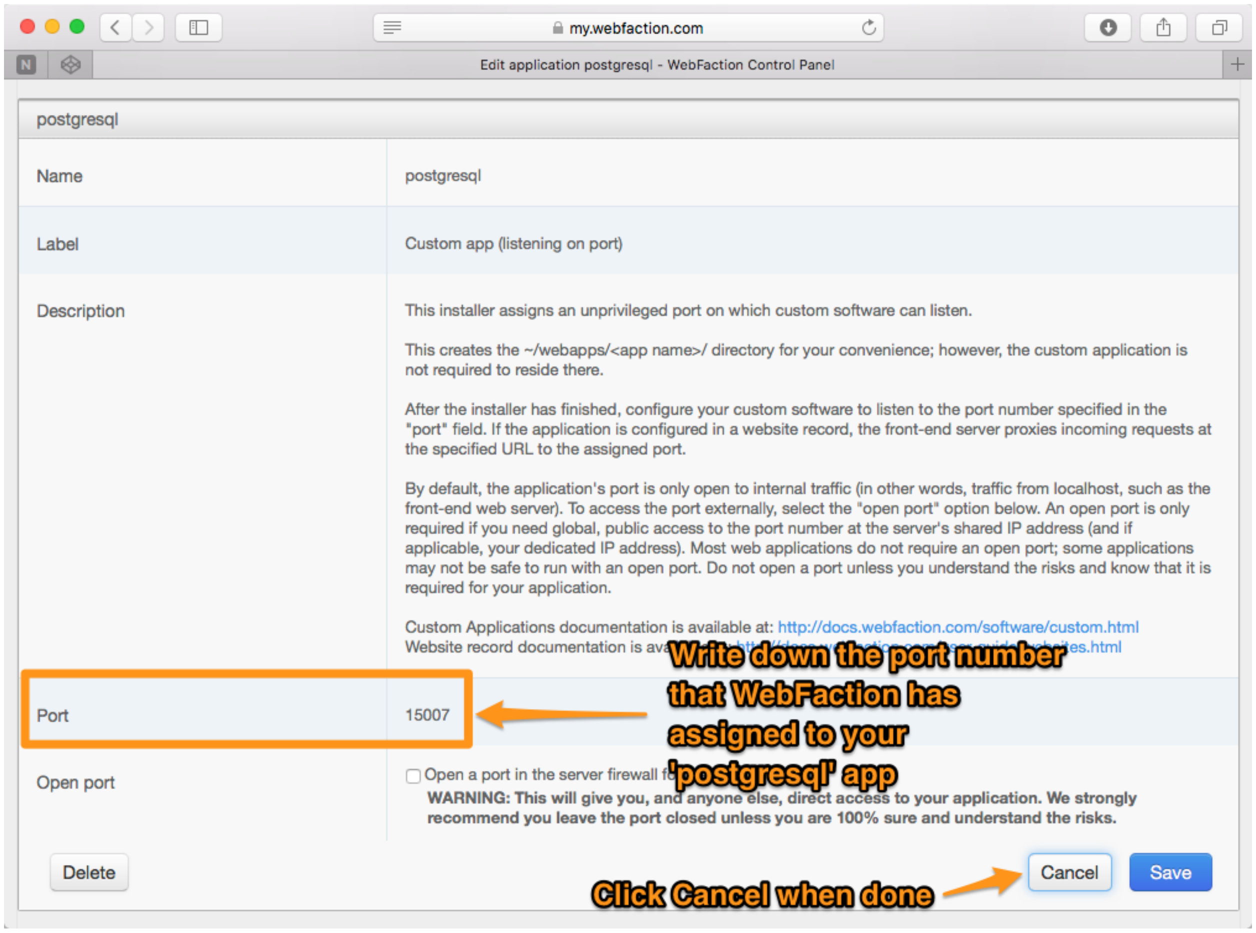
You should now see your app 'postgresql' listed in the list of applications. Click on it.



bookmarks	Custom app (listening on port)	Web502
bookmarks_static	Symbolic link to static-only app	Web502
bottleapps	Custom app (listening on port)	Web502
customnginx	Custom app (listening on port)	Web502
djangoapps	Custom app (listening on port)	Web502
dpnotes	Custom app (listening on port)	Web502
filepublish	Custom app (listening on port)	Web502
javaapps	Custom app (listening on port)	Web502
jaycjayc	WordPress 4.3.1	Web502
phpapps	Symbolic link to static/cgi/php70 app	Web502
postgresql	Custom app (listening on port)	Web502
pythonapps	Custom app (listening on port)	Web502
redirects	Static/CGI/PHP-5.6	Web502
serials	Static/CGI/PHP-7.0	Web502
wp	WordPress 4.3.1	Web502

Click the postgresql app

Take note of the port number and write it down somewhere. In the example below, the port number is 15007.


my.webfaction.com
Edit application postgresql - WebFaction Control Panel
Write down the port number that WebFaction has assigned to your 'postgresql' app
Click Cancel when done
Delete
Cancel
Save

postgresql

Name	postgresql
Label	Custom app (listening on port)
Description	This installer assigns an unprivileged port on which custom software can listen. This creates the ~/webapps/<app name>/ directory for your convenience; however, the custom application is not required to reside there. After the installer has finished, configure your custom software to listen to the port number specified in the "port" field. If the application is configured in a website record, the front-end server proxies incoming requests at the specified URL to the assigned port. By default, the application's port is only open to internal traffic (in other words, traffic from localhost, such as the front-end web server). To access the port externally, select the "open port" option below. An open port is only required if you need global, public access to the port number at the server's shared IP address (and if applicable, your dedicated IP address). Most web applications do not require an open port; some applications may not be safe to run with an open port. Do not open a port unless you understand the risks and know that it is required for your application. Custom Applications documentation is available at: http://docs.webfaction.com/software/custom.html Website record documentation is ava
Port	15007
Open port	☐ Open a port in the server firewall f **WARNING: This will give you, and anyone else, direct access to your application. We strongly recommend you leave the port closed unless you are 100% sure and understand the risks.**

Write down the assigned port number

2. Create and initialize a PostgreSQL data directory

We've already completed installing the PostgreSQL software in Chapter 5.

In this chapter, we'll create a directory named 'pgdata' in your home folder.

The purpose of this directory is to hold all the files that PostgreSQL needs for data storage.

2.1 Create the data directory

`cd ~`
: Switch to the home directory, if you aren't already in it

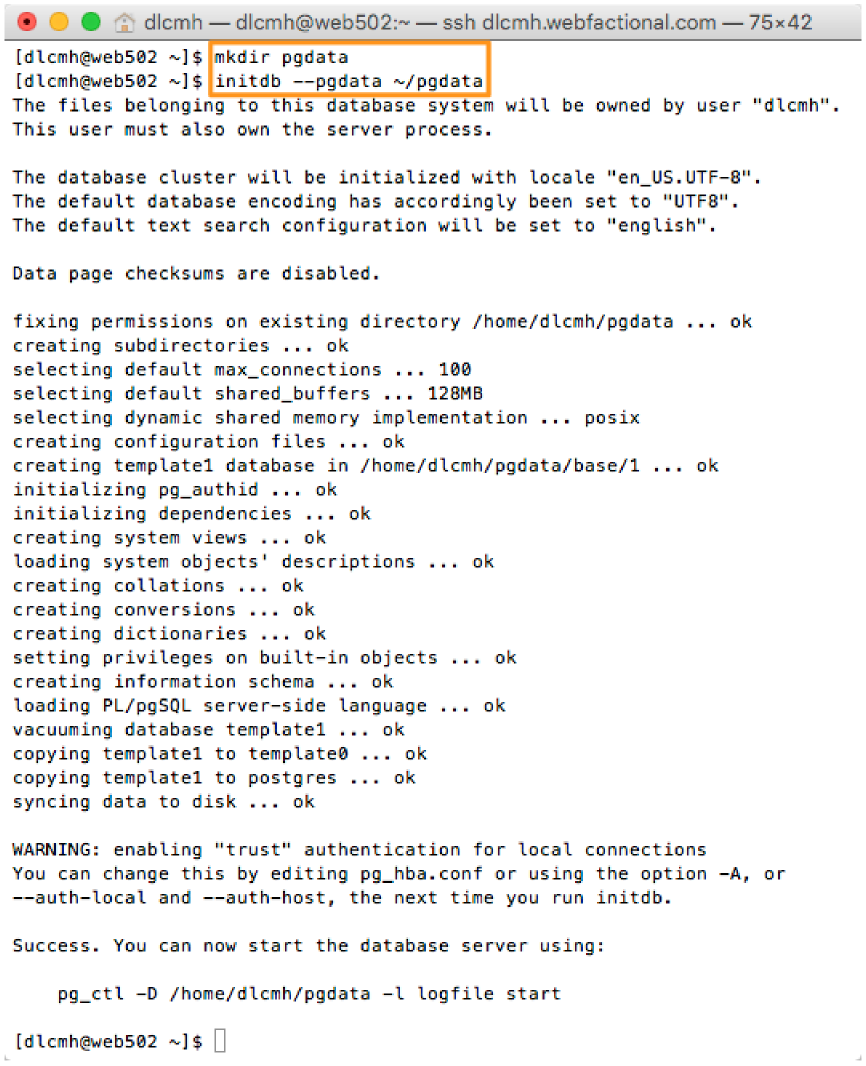
`mkdir pgdata`
: Create a directory named 'pgdata' that PostgreSQL will use for data storage

2.2 Initialize the data directory with PostgreSQL data files and sub-folders

`initdb --pgdata ~/pgdata`
: The following is an extract of the documentation at the webpage PostgreSQL: Documentation: 9.5: initdb[1] for this command:

> `initdb` creates a new PostgreSQL database cluster. A database cluster is a collection of databases that are managed by a single server instance.
>
> Creating a database cluster consists of creating the directories in which the database data will live, generating the shared catalog tables (tables that belong to the whole cluster rather than to any particular database), and creating the template1 and postgres databases. When you later create a new database, everything in the template1 database is copied. (Therefore, anything installed in template1 is automatically copied into each database created later.) The postgres database is a default database meant for use by users, utilities and third party applications.

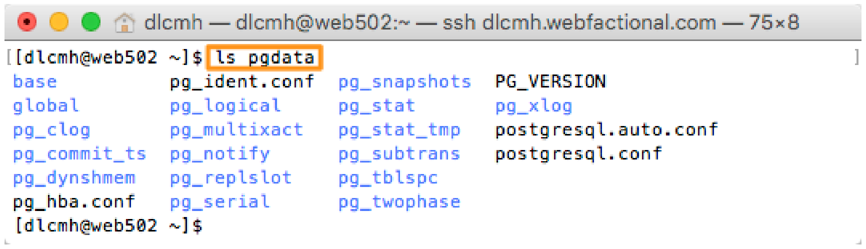
[1] http://www.postgresql.org/docs/current/static/app-initdb.html



```
[dlcmh@web502 ~]$ mkdir pgdata
[dlcmh@web502 ~]$ initdb --pgdata ~/pgdata
The files belonging to this database system will be owned by user "dlcmh".
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".

Data page checksums are disabled.

fixing permissions on existing directory /home/dlcmh/pgdata ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
creating template1 database in /home/dlcmh/pgdata/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
syncing data to disk ... 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:

    pg_ctl -D /home/dlcmh/pgdata -l logfile start

[dlcmh@web502 ~]$ 
```

`mkdir` and `initdb` the `pgdata` data directory

To see the files and sub-folders that `initdb` created, type the command `ls pgdata`:



```
[dlcmh@web502 ~]$ ls pgdata
base           pg_ident.conf  pg_snapshots  PG_VERSION
global         pg_logical     pg_stat       pg_xlog
pg_clog        pg_multixact   pg_stat_tmp   postgresql.auto.conf
pg_commit_ts   pg_notify      pg_subtrans   postgresql.conf
pg_dynshmem    pg_replslot    pg_tblspc
pg_hba.conf    pg_serial      pg_twophase
[dlcmh@web502 ~]$
```

`ls pgdata`

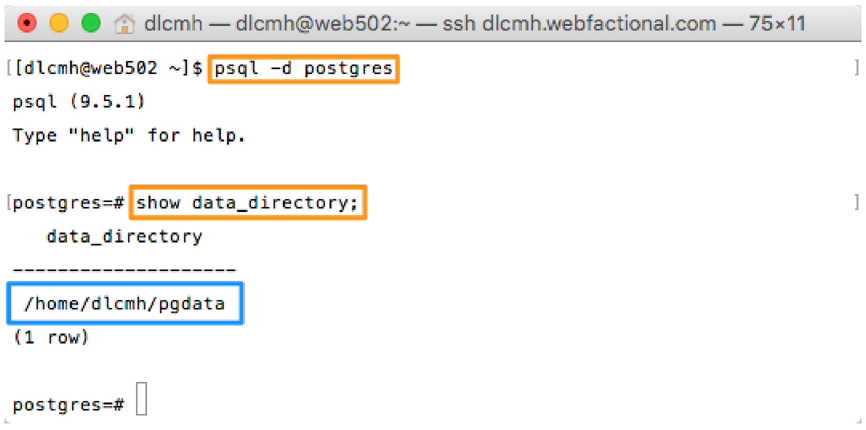
2.3 Optional: Find out the location of the data directory with `psql` command `show data_directory`

Assuming PostgreSQL is already running (see Chapter 7 for details on how to start up PostgreSQL and determine if it is running), enter the following sequence of commands to find out the location of the data directory:

`psql --dbname=postgres`
: Starts the PostgreSQL interactive terminal and connects to the default database named `postgresql`. If you'd already created a database named after your WebFaction username (say, `dlcmh` in my case, as explained in Chapter 8), then you'd just need to issue the command `psql` without needing to specify `--dbname=yourusername`.

`show data_directory;`
: Don't forget to end the command with a semicolon. This will display the full path to the data directory for this running instance of PostgreSQL.

In the screen capture below, the data directory is revealed to be `/home/dlcmh/pgdata`.



```
[dlcmh@web502 ~]$ psql -d postgres
psql (9.5.1)
Type "help" for help.

postgres=# show data_directory;
   data_directory
--------------------
 /home/dlcmh/pgdata
(1 row)

postgres=#
```

`show data_directory`