Circular Multilateral Barter Installation Guide

Description: Installation guide for the CMB server-side software
Date: 2016-04-10
Author: Evgeni Pandurksi
Contact: epandurski@gmail.com

Contents

Installation on a dedicated server

Using a dedicated server (or servers) is the preferred way of installing *Circular Multilateral Barter* (CMB). This method gives the maximum amount of control, flexibility, and privacy. It is the recommended approach in all cases where maintaining dedicated server(s) is feasible.

Before you start the installation, you should download and copy CMB's source code in your */usr/local/share/* directory:

```
# cd /usr/local/share/
# wget http://sourceforge.net/projects/cmb/files/tarballs/\
cmbarter-1.14.tar.gz/download -O cmbarter-1.14.tar.gz
...

# tar -xzf cmbarter-1.14.tar.gz
# mv cmbarter-1.14 cmbarter
```

Also, make sure a *Python 2.7* interpreter is installed on your server.

Application installation

Here are the installation steps that you should perform:

1. Install the *Python 2* versions of the following software packages:
   - *Django* (Versions from 1.4 to 1.10 have been tested; chances are that newer versions will work too.)
   - *psycopg2* (version 2.3 at least)
   - *Pillow* (or "Python Imaging Library")
   - *pycrypto* [1]
   - *pytz*

   For example [2]:

```
# apt-get install python-django
...

# apt-get install python-psycopg2
...

# apt-get install python-imaging
...

# apt-get install python-crypto
...

# apt-get install python-tz
...
```

2. Create *cmbarter* unix user. For example:

```
# adduser cmbarter --system --group
...
```

3. Change the owner of */usr/local/share/cmbarter* to *cmbarter*:

```
# chown cmbarter -R /usr/local/share/cmbarter
# chgrp cmbarter -R /usr/local/share/cmbarter
```

4. Restrict access to those source files that may contain sensitive information:

```
# chmod og-r /usr/local/share/cmbarter/cmbarter/settings.py
```

5. Create the directory that will contain django's session files:

```
# mkdir /var/tmp/cmbarter
# chown cmbarter /var/tmp/cmbarter
# chgrp cmbarter /var/tmp/cmbarter
# chmod og-rx /var/tmp/cmbarter
```

Web server installation

Although CMB should work well with all kinds of web servers, this document focuses specifically on running CMB with *Apache*. Therefore, having prior experience with administering Apache would be of help.

Here are the installation steps that you should perform:

1. Install Apache and make sure the following modules are active:
    - *mod_mime*
    - *mod_dir*
    - *mod_alias*
    - *mod_expires*
    - *mod_ssl*
    - *mod_headers*
    - *mod_rewrite*
    - *mod_include*
    - *mod_reqtimeout*
    - *mod_authz_host*

    Make sure *mod_deflate* is NOT ACTIVE.
2. Install and activate *mod_wsgi* [3]. Make sure mod_wsgi uses the *Python 2.7* interpreter, and not the *Python 3.x* one.

    For example:

    ```
    # apt-get install libapache2-mod-wsgi
    ...
    ```
3. Set the variable "PYTHONHASHSEED" to "random" in Apache's execution environment.

    For example:

    ```
    # echo "export PYTHONHASHSEED=random" >> /etc/apache2/envvars
    ```
4. Obtain a proper SSL certificate for your server.
5. Use the prototype configuration in */usr/local/share/cmbarter/apache/httpd.conf* to configure your Apache. Notice that you will have to make some changes to the prototype configuration in order to adapt it to your specific setup. In particular, do not forget to replace "yourdomainname.foo" with your real domain name.
6. Consider installing and configuring a firewall (*iptables* for example) to protect your web-server from denial-of-service attacks. [4]

Mail server installation

CMB needs a mail server only for sending outgoing e-mails. Therefore, you may install whatever server is most convenient for you [5]. The only requirement is that the server is configured to accept anonymous connections at *localhost:25*.

Database server installation

CMB relies on the *PostgreSQL* open source database server to hold its data. You do not need to know very much about PostgreSQL to install CMB, but you definitely should obtain some experience in administering PostgreSQL databases in order to keep your users' data safe and secure.

Here are the installation steps that you should perform:

1. Install PostgreSQL (version 8.3 at least).

Keep in mind that the default PosgreSQL configuration is not very well suited for large database servers. So, you will probably need to edit your PosgreSQL configuration files at some point [6]. See PostgreSQL's documentation for more info.

2. Create a database user *cmbarter* and a database *cmbarter* belonging to this user. For example:

```
# su postgres

$ createuser cmbarter
...

$ createdb --owner=cmbarter cmbarter "The CMB database"
...

$ exit
```

3. Create the necessary objects in the database schema. Make sure they are all owned by the *cmbarter* database user:

```
# sudo -u cmbarter psql -d cmbarter

cmbarter=> \cd /usr/local/share/cmbarter/pgsql
cmbarter=> create language plpgsql;
...

cmbarter=> \i schema.sql
...

cmbarter=> \i triggers.sql
...

cmbarter=> \i views.sql
...

cmbarter=> \i sprocs.sql
...
```

4. If your users' primary language is other than English, you should create a new default text-search configuration for that language. (The out-of-the-box configuration works well for English.)

   For example, to create a default text-search configuration for Bulgarian, you should extract the file */usr/local/share/cmbarter/pgsql/tsearch_data/bulgarian.tar.gz* and copy its content to your PostgreSQL "tsearch_data" directory. Then you should execute the following commands in *psql*:

```
CREATE TEXT SEARCH DICTIONARY bulgarian_ispell (
  TEMPLATE = ispell,
  DICTFILE = bulgarian,
  AFFFILE = bulgarian,
  STOPWORDS = bulgarian
);

CREATE TEXT SEARCH CONFIGURATION public.bulgarian (
  COPY = pg_catalog.russian
);

ALTER TEXT SEARCH CONFIGURATION bulgarian
  ALTER MAPPING FOR word, hword, hword_part
```

```
  WITH bulgarian_ispell, simple;

ALTER DATABASE cmbarter
  SET default_text_search_config TO 'public.bulgarian';
```

See PostgreSQL's documentation for more info.

Application configuration

Here are the configuration steps that you should perform:

1. Review and edit your */usr/local/share/cmbarter/cmbarter/settings.py* file — the CMB configuration is held there.
2. Add the following lines to your system *crontab*:

```
0,10,20,30,40,50 * * * * sudo -u cmbarter python /usr/local/share...
  /cmbarter/execute_turn.py

* * * * * sudo -u cmbarter python /usr/local/share/cmbarter/check...
  _sessions.py --repeat-interval=5

* * * * * sudo -u cmbarter python /usr/local/share/cmbarter/proce...
  ss_emails.py
```

Call each script [7] with "--help" to see its full list of accepted parameters.

Notice that the system crontab format might be slightly different on your system. Also, make sure "python" runs the *Python 2.7* interpreter, and not the *Python 3.x* one.

Installation on a shared server

Although using a dedicated server is the preferred way of installing *Circular Multilateral Barter* (CMB), sometimes maintaining your own server is not feasible. In such cases, an installation on a shared server is a perfectly acceptable, and fully functional alternative.

Choosing a hosting provider

Choosing appropriate hosting provider for your installation is probably the most difficult step. Here are some important things that you should look for:

- They must have real experience in hosting *Python* web applications. Many providers are narrowly specialized in hosting PHP applications. You should avoid them.
- They must support *PostgreSQL* databases.
- They must support *FastCGI*.
- They must use *Apache*. Although you could configure CMB to work with other web servers, Apache is the safe bet.
- They must give you *SSH* access to your user account. Although you could manage to install CMB without SSH access, not having it is a huge obstacle.

CMB is quite efficient in using system resources, so you probably will not need more than few gigabytes of disk and database space.

Application installation

Here are the installation steps that you should perform:

1. Make sure a *Python 2.7* interpreter is installed on the hosting server. Also, make sure the *Python 2* versions of the following software packages are installed:

   - *Django* (Versions from 1.4 to 1.10 have been tested; chances are that newer versions will work too.)
   - *psycopg2* (version 2.3 at least)
   - *Pillow* (or "Python Imaging Library")
   - *pycrypto*
   - *pytz*
   - *flup*

   Often some of the packages will be missing on the hosting server. Therefore you should either be able to convince administrators to install them for you, or be able to set up a customized local Python environment for yourself (using *virtualenv* for example).

2. Download and copy CMB's source code in your home directory. For example:

```
$ cd ~
$ wget http://sourceforge.net/projects/cmb/files/tarballs/\
cmbarter-1.14.tar.gz/download -O cmbarter-1.14.tar.gz
...

$ tar -xzf cmbarter-1.14.tar.gz
$ mv cmbarter-1.14 cmbarter
```

3. Restrict access to those source files that may contain sensitive information:

```
$ chmod og-r ~/cmbarter/cmbarter/settings.py
```

4. Create the directory that will contain django's session files:

```
$ mkdir ~/tmp/cmbarter
$ chmod og-rx ~/tmp/cmbarter
```

Apache configuration

Here are the configuration steps that you should perform:

1. Obtain and install a proper SSL certificate for your site. To do this, you will probably have to buy a dedicated IP address from your hosting provider.
2. Copy or link *~/cmbarter/static/* and *~/cmbarter/doc/* to your web root directory. For example:

```
$ ln -s ~/cmbarter/static/ ~/public_html/static
$ ln -s ~/cmbarter/doc/ ~/public_html/doc
```

3. Copy *~/cmbarter/apache/cmbarter.fcgi* to your web root directory. Make sure it is executable. For example:

```
$ cp ~/cmbarter/apache/cmbarter.fcgi ~/public_html/
$ chmod a+x ~/public_html/cmbarter.fcgi
```

You will need to make some changes in the newly created *cmbarter.fcgi* file in order to adapt it to your specific setup. Do not forget to replace "yourusername" with your real username.

4. Add the content of *~/cmbarter/apache/htaccess* to your *.htaccess* file in the web root directory. For example:

```
$ cat ~/cmbarter/apache/htaccess >> ~/public_html/.htaccess
```

You will need to make some changes in the *.htaccess* file in order to adapt it to your specific setup. Do not forget to replace "yourdomainname.foo" with your real domain name.

Email configuration

CMB needs a mail server only for sending outgoing e-mails. Therefore, the only needed configuration is to create one email account on the outgoing mail server: "noreply@yourdomainname.foo".

Database configuration

Here are the configuration steps that you should perform:

1. Create a new PostgreSQL database. To do this, you will probably have use whatever tools have been given to you by your hosting provider.
2. Create the necessary objects in the database schema. For example:

```
$ cd ~/cmbarter/pgsql/
$ psql -d yourdatabase

yourdatabase=> create language plpgsql;
...

yourdatabase=> \i schema.sql
...

yourdatabase=> \i triggers.sql
...

yourdatabase=> \i views.sql
...

yourdatabase=> \i sprocs.sql
...
```

Application configuration

Here are the configuration steps that you should perform:

1. Review and edit your *~/cmbarter/cmbarter/settings.py* file — the CMB configuration is held there.
2. Add the following lines to your *cron* jobs:

```
0,10,20,30,40,50 * * * * python ~/cmbarter/execute_turn.py

* * * * * python ~/cmbarter/check_sessions.py

* * * * * python ~/cmbarter/process_emails.py --smtp='mailserve...
  rname' --smtp-username='noreply@yourdomainname.foo' --smtp-pa...
```

```
  ssword='yourpassword'

0 * * * * touch ~/public_html/cmbarter.fcgi
```

Do not forget to replace the dummy data with your real data. Call each script with "--help" to see its full list of accepted parameters.

Maintenance

In CMB, all application-level maintenance tasks are automated. There are, however, at least two important tasks, that you should figure out how to do by yourself. Those are:

- Database backups
- Traffic, and system load data analysis

You will be able to find lots of good open source tools performing these tasks, though.

---

1 CMB tries its best to support PyPy. If *pycrypto* is not available, CMB will use its own pure-python AES cipher implementation. Also, if *psycopg2* is not available CMB will try to use *psycopg2cffi* instead.

2 The examples given here are for *Debian*. If you use another operating system, the exact commands that do the work might be different.

3 You can not use *mod_wsgi* with PyPy. To use Apache with PyPy, you may configure *mod_proxy* to operate as a reverse proxy for a separate PyPy-compatible WSGI server (*gunicorn* for example). You may consider using *nginx* as well.

4 Here is a simplistic, but informative *iptables* configuration that limits the number of new connections from one IP address over a time period:

```
# iptables -A INPUT -p tcp -m conntrack \
--ctstate RELATED,ESTABLISHED -j ACCEPT

# iptables -A INPUT -p tcp -m recent \
--update --seconds 60 --hitcount 60 -j DROP

# iptables -A INPUT -p tcp -m recent --set
```

Also, while CMB is working, it maintains a whitelist of "good" IP addresses. You can use this auto-generated whitelist to configure *iptables* to further protect your web-server (see the *show_whitelist.py* command-line tool).

5 Debian uses the *exim* mail server by default. You can reconfigure it with this command:

```
# dpkg-reconfigure exim4-config
...
```

6 You may consider changing the following parameters: *maintainance_work_mem, shared_buffers , effective_cache_size, checkpoint_segments, checkpoint_timeout, checkpoint_completion_target, effective_io_concurrency, random_page_cost*

7 The important commands are: *execute_turn.py*, *process_emails.py*, *check_sessions.py*, *generate_regkeys.py*, *schedule_turns.py*, *show_whitelist.py*, *show_emails.py*. "manage.py" is Django's standard administration tool.