Setting up a Linux-based server for CollectiveAccess

Revised 3 March 2018 based on install_ca.sh v.7.0.4

This document goes through how to set up a Linux-based server for CollectiveAccess step-by-step. This process can take several hours to complete, but the actual duration depends heavily on the speed of your computer hardware and the speed of your internet connection.

If you want to not go through this tedious process,
there is a bash shell script that automates Steps 1-13 or this document
available at https://github.com/ChrisLitfin/install_ca.
Follow the instructions in README.md.

Note that this process will require the downloading of a LOT of data (on the order of several gigabytes), so make sure that that will not be a problem by consulting the appropriate authorities.

The process outlined in this document provides a reasonably secure installation, and is likely secure enough for a server that is inside a corporate/institutional firewall. If this server is going to be accessible from the Internet, you will need better security. Consult with somebody who knows these things. If you are a corporation/institution, there is likely somebody who is paid to know such things.

You should also consider how you will back up the data for your CollectiveAccess system. For a small system, you can probably go it alone; but for a large corporate/institutional system, consult with the aforementioned person who is paid to know such things.

Security and Backups, other than what is mentioned above, are outside of the scope of this document.

In terms of selecting server hardware, pretty much anything will work. However, more RAM is better, especially if you are importing large amounts of data at once. RAM is reasonably cheap, so better to have more than you ever think you will need. The actual operating system and other software needed to run CollectiveAccess, including CollectiveAccess itself, is quite small (less than 30 gigabytes). Storage is also quite inexpensive, but also easy to upgrade later. Choose a platform (motherboard, case, etcetera) which has room for expansion.

A note on passwords:

Passwords are extremely important when it comes to security. However, most passwords are laughably weak. Consider this: at the time of writing, passwords of up to 12 characters can be cracked using brute force (i.e. trying every possible combination) in less than a day on hardware available at the local computer shop for under 5000 US Dollars. Despite this, many corporations and institutions still allow short (8 character) passwords. (In fact, there is at least one major North American University which requires passwords to be exactly 8 characters!)

To counter this, try the following method of making a password: choose four words at random, string them together with spaces (writing the words in lower case) and use that for your password. Choose words that are reasonably infrequently used, like “unbecoming” or “ziggurat”. To make things even more secure, add a random underscore or other special character into a word (don’t substitute it for a letter, add it into the word). An example of a password created this way would be “unbecoming molten vale_ntine ziggurat”.

These instructions assume the following:

1. You have installed a Debian-derivative Linux Operating system (e.g. Debian, Ubuntu) already. (These instructions assume that this is Debian stretch, but they will work with small modifications on other OSes.)
2. That you do not have anything on the server already.
3. That you are logged in as whatever user you are going to install the system as, that that user IS NOT root, and that the user has sudo privileges. (these instructions assume that this user is `www`).

[STEP 0] GETTING NESCESSARY INFORMATION

Gather the necessary information to do this install. WRITE THIS INFORMATION DOWN SOMEWHERE SECURE. If you have a safe or vault, that is a good place for it. You will need:

a. The password for the user that you will install under (see passwords, above)
b. The ip address of the computer you are installing on
   i. First, we need to get the IP address of the computer. Run:
      `ip addr show eth0`
      (If your server has multiple Ethernet ports, change the "0" (zero) in "eth0" to another number to get the other IP addresses.)
   ii. The IPv4 address will be in the third line, between "`inet`" and the first "`/`".
c. The "Olson" Timezone that your computer is set to
   i. Run:
      `timedatectl`
   ii. The line labeled "Time zone" has the information you need, in the format "Continent"/"Location". Ignore the stuff in the (brackets) on the same line.
d. The system locale
   i. Run:
      `locale`
   ii. The first line has the information you need. The locale is in the format "aa_AA" (i.e. en_US). Ignore the stuff after those 5 characters, if any.
e. A DIFFERENT password for the MySQL root user (see passwords, above)
f. The app_name you want to use for CollectiveAccess (alphanumeric only)
g. YET ANOTHET DIFFERENT PASSWORD for the MySQL ca user (see passwords, above)
h. The display_name you want to use for CollectiveAccess (Can be whatever ASCII)

[STEP 1] ADD NEW PACKAGE SERVERS

Firstly, we need to add some new servers to the list of servers we can download packages from. This is because the default servers in the list don't have some packages that we need.

a. Run the following command, which opens the server source list in the text editor Nano:
   `sudo nano /etc/apt/sources.list`
   You will be prompted for the password for www. Enter it and press "Enter" to confirm.
b. At the bottom of the list, we need to add two more servers. (There should be five or so entries in the list already; blank lines are ignored.) Add:
   ```
   deb http://www.deb-multimedia.org stretch main non-free
   deb-src http://www.deb-multimedia.org stretch main non-free
   ```
   These two servers are specifically for media packages, which we will need when it comes time to install dependencies for CollectiveAccess
c. Press CTRL+X to exit. When prompted to save, type "`y`", and then "Enter" when asked to confirm the file name.

[STEP 2] UPDATE PACKAGE LISTS

We now need to update the package lists.

i. Run the following command, which should throw an error:
`sudo apt-get update`
If it does not throw an error, go back to the beginning of this step and make sure that you have correctly edited sources.list.

ii. We need to fix the error by explicitly trusting the media package servers:
`apt-get install deb-multimedia-keyring`
Press "y" to accept changes if prompted.

iii. We then run the package update again, which should now be happy:
`sudo apt-get update`

[STEP 3] UPGRADE PRE-INSTALLED PACKAGES

We now need to upgrade the pre-installed packages to the latest versions. Run the following commands, one at a time. They will update the lists of available packages and update any existing packages already installed (some are installed during the install of the OS).

`sudo apt-get dist-upgrade` (Note that this will likely appear to freeze, but it is actually waiting for human intervention. Type 'q' to accept.)

```
sudo apt-get autoremove
sudo apt-get upgrade
sudo apt-get update
```

If prompted to enter the password for `www`, enter it. If prompted to confirm changes, type "`y`".

[STEP 4] INSTALL APACHE

We are now ready to do some actual installing. We start with Apache.

a. Run:
`sudo apt-get install apache2`
Confirm changes with "`y`". This installs Apache and its dependencies.

b. We can now check to make sure that Apache has been installed correctly. To check if the server is working properly, enter the IP address of the server into the web browser of another computer on the same network. You should get the Apache default page.

[STEP 5] INSTALL PHP7.0

Now we install PHP 7.0.

a. Run (all on one line):
`sudo apt-get install php7.0 php7.0-gd libapache2-mod-php7.0 php7.0-mcrypt php7.0-curl php7.0-dev`
Confirm changes with "`y`". This installs PHP 7.0 and its dependencies, some under-the-hood stuff to make PHP and Apache play nice, and some packages that CollectiveAccess will need to work, and some packages that we will need to install other CollectiveAccess dependencies.

b. We need to change some default settings about how php behaves. Run the following commands (on one line):

i. `sudo cp /etc/php7.0/apache2/php.ini /etc/php7.0/apache2/php.ini-old`
Which creates a backup copy of php.ini so we can revert if it breaks things; and

ii. `sudo nano /etc/php7.0/apache2/php.ini`
Which opens the php.ini file in Nano for editing.

iii. In Nano, we want to change four variables in php.ini. The easiest way to find the variables we want to change (the file is nearly 2000 lines) is to press CTRL+W to bring up a search bar at the bottom of the screen, and then search the COMPLETE variable name. We want to change the following variables:

1. post_max_size - sets maximum size a POST-style HTTP request can be. The default value is 8 megabytes. If you are uploading large media files (and most CollectiveAccess

users are) you will need to raise this to a value larger than the largest file size you are likely to encounter. 32M is good.
2. upload_max_filesize - sets the maximum size of an uploaded file. Set this to the same value as post_max_size.
3. memory_limit - sets the maximum amount of memory a PHP script may consume. The default is 128 megabytes which should be enough for many systems, unless you are (a) uploading large images (b) reindexing the search index of a large database or (c) importing data. Even if you have not received memory limit exceeded errors, you may want to increase this limit to 196 or 256 megabytes. For importing large amounts of data, this may need to be raised to several thousand megabytes.
4. display_errors - determines whether errors are printed to the screen or not. In some installation this is set to "off" by default. While this is a good security decision for public-facing systems, it can make debugging installation problems difficult. It is therefore suggested that while installing and testing CollectiveAccess you set this option to "On". You can always change it back later.

iv. When you are done, press CTRL+X to exit. When prompted to save, type "`y`", and then "Enter" when asked to confirm the file name.

c. To make sure that the changes that we made take effect, we have to restart Apache:
```
sudo service apache2 restart
```
If it fails to restart (it will give a message if that happens), php.ini probably got messed up somehow. Restore it using:
```
sudo cp /etc/php7.0/apache2/php.ini-old
/etc/php7.0/apache2/php.ini
```
And try this section again.

d. We now need to tell Apache to look for PHP files first, instead of html files.

i. Run:
```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

ii. You should get a file that looks something like this:
```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php
index.xhtml index.htm
</IfModule>
```

iii. We want to delete the existing "`index.php`" and add a new "`index.php`" between "`DirectoryIndex`" and "`index.html`". Make sure that each entry on the "`DirectoryIndex`" line is separated from the adjacent text with exactly one space, otherwise it gets angry. After making the changes, you should get something like this (note the location of "`index.php`"):
```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl
index.xhtml index.htm
</IfModule>
```

iv. Press CTRL+X to exit. When prompted to save, type "`y`", and then "Enter" when asked to confirm the file name.

e. We now want to re-start Apache to make sure that the changes take effect. Run:
```
sudo service apache2 restart
```
Which restarts Apache.

f. Finally, we want to check that Apache and PHP are talking nice to each other. To do this, we will create a very basic PHP web page which tells us the configuration. Run:
```
sudo nano /var/www/html/index.php
```

to create a new file in Nano. In the blank file, type (on three separate lines):

```
<?php
phpinfo();
?>
```

Press CTRL+X to exit. When prompted to save, type "y", and then "Enter" when asked to confirm the file name. Now, on that other computer on the same network that you used earlier to test Apache, type the same IP address (or refresh the page). You should now get a different page with the PHP logo at the top-right. If you did not get this, make sure that you made the correct edit in Step 9b and that you actually created the `index.php` file where you intended. Re-do the commands in this step to check.

[STEP 6] INSTALL IMAGE HANDLING & INSTALLATION DEPENDANCIES

We now need to install a bunch of helper software that CollectiveAccess needs to run correctly, primarily when dealing with media files.

a. Git is a platform (like apt, which we have been using) that automatically downloads and installs software and the appropriate dependencies.

b. Various image handling packages. Run (on one line):

```
sudo apt-get install git php-pear liblzma5 liblzma-dev libtiff-dev graphicsmagick libgraphicsmagick1-dev ghostscript dcraw libreoffice poppler-utils mediainfo wkhtmltopdf openctm-tools meshlab meshlab python-pip
```

This installs a whole bunch of packages that allow CollectiveAccess to deal with images, along with some packages that we will need to install other dependencies.

c. Gmagick is a PHP-based extension to GraphicsMagick which provides additional image handling. We install it by running the following (each on a separate line):

i. `pecl channel-update pecl.php.net`
ii. `pecl install gmagick-2.0.4RC1`
iii. `echo -e "; configuration for php gmagick module\n; Added by CollectiveAccess Installer\n; priority=20\nextension=gmagick.so" > /etc/php/7.0/apache2/conf.d/20-gmagick.ini`
`sudo pecl install gmagick-2.0.4RC1`

If you are prompted to enter anything, just press "Enter" to use the default or automatic value.

d. We need to install pdfminer, which is used to get data out of pdfs for indexing purposes:

i. `pip install pdfminer`

e. We need to compile ExifTool, which gets data out of image files like time of creation, camera info, etc. Run each command on a separate line:

i. `mkdir /usr/local/exiftool_sources`
ii. `cd /usr/local/exiftool_sources`
iii. `wget http://www.sno.phy.queensu.ca/~phil/exiftool/Image-ExifTool-10.82.tar.gz`
iv. `gzip -dc Image-ExifTool-10.82.tar.gz | tar -xf -`
v. `cd Image-ExifTool-10.82`
vi. `perl Makefile.PL`
vii. `make test`
viii. `make install`
ix. `cd`

[STEP 7] INSTALL FFMPEG

We now have to install ffmpeg, which handles pretty much everything to do with video and audio

files in CollectiveAccess. Ffmpeg is now ACTUALLY a package again, instead of having to compile it yourself, which was a giant pain.

i. First off, we need to remove an old (broken) version of ffmpeg which may have come pre-installed:
```
sudo apt-get remove ffmpeg
```
ii. Now, we install ffmpeg and all of its dependencies:
```
sudo apt-get install ffmpeg
```

[STEP 8] INSTALL MYSQL

Next, we install MySQL.

a. Run:
```
sudo apt-get install mysql-server
```
Again, confirm changes with "y". This installs MySQL and its dependencies. You will be asked to choose and confirm a root password for MySQL. You wrote this down during Step 0.

b. We now tell MySQL to create its database structure. Run:
```
sudo mysql_install_db
```
You may be prompted for either the user password, the MySQL root password, or both. Make sure that you are entering the correct password (because they wouldn't be the same, would they?).

c. Finally, we tell MySQL to make itself more secure. This runs a script which closes some of the more obvious security holes in MySQL. Run:
```
sudo mysql_secure_installation
```
You will be prompted for the MySQL root password. After that, you will get a series of 5 yes/no questions. Answer "`n`" (no) to the first one (since we already have a root password set, and it is very nice), and then "`y`" (yes) to the other 4.

d. Run:
```
sudo apt-get install php7.0-mysql
```
to link php7.0 to MySQL

[STEP 9] CREATE COLLECTIVEACCESS USER AND DATABASE

We now need to create a user and database for CollectiveAccess to use.

a. Run the following, where [SQLROOTPW] is the SQL root password from step 0, [CAUSER] is the CollectiveAccess MySQL username from Step 0, and [CAPW] is the MySQL non-root password from Step 0 (make sure to remove the square brackets and pay attention to the quotes):

i.
```
mysql -uroot -p[SQLROOTPW] -e "CREATE DATABASE [CAUSER] /*\!40100 DEFAULT CHARACTER SET utf8 */;"
```
ii.
```
mysql -uroot -p[SQLROOTPW] -e "CREATE USER [CAUSER]@'localhost' IDENTIFIED BY '[CAPW]';"
```
iii.
```
mysql -uroot -p[SQLROOTPW] -e "GRANT ALL PRIVILEGES ON [CAUSER].* TO '[CAUSER]'@'localhost';"
```
iv.
```
mysql -uroot -p[SQLROOTPW] -e "FLUSH PRIVILEGES;"
```

[STEP 10] INSTALL COLLECTIVEACCESS PROVIDENCE

We are now finally ready to download CollectiveAccess (Providence, anyway)!

a. We first need to get rid of any default webpages that tell the whole world details about our server.

i. Run:
```
sudo rm /var/www/html/index.php
```
which deletes `index.php`.

ii. While we are at it, we should also delete `index.html` (which is the default Apache page we saw earlier), since it gives out similar information we really don't want to share. Run:

```
sudo rm /var/www/html/index.html
```
Which deletes `index.html`.

b. Run the following command, which downloads the actual CollectiveAccess Providence back-end (all on one line):
```
git clone https://github.com/collectiveaccess/providence.git
/var/www/html/staff
```
This will put Providence in a subdirectory called "staff" in the web root of your server.

c. We now configure Providence to use the database we just set up.
   i. On the server, run (on one line):
   ```
   sudo cp /var/www/html/staff/setup.php-dist
   /var/www/html/staff/setup.php
   ```
   to copy the setup file and then:
   ```
   sudo nano /var/www/html/staff/setup.php
   ```
   to open it for editing in Nano.
   ii. In Nano, we need to change, at a very minimum, four of the first five variables: __CA_DB_USER__, __CA_DB_PASSWORD__, __CA_DB_DATABASE__, and __CA_APP_DISPLAY_NAME__. Change these to reflect the information you worte down in Step 0. (Note that __CA_DB_DATABASE__ is the same as what you entered for __CA_DB_USER__.)
   iii. You also need to change date_default_timezone_set and __CA_DEFAULT_LOCALE__ to reflect the information you wrote down in Step 0. Details of how to do this are in the setup.php file.
   iv. When you are done, press CTRL+X to exit. When prompted to save, type "y", and then "Enter" when asked to confirm the file name.

d. We also have to tell CollectiveAccess where some of the dependencies we installed earlier actually are. Most of the defaults are fine, but we have to fix three of them.
   i. Run:
   ```
   sudo nano
   /var/www/html/staff/app/conf/external_applications.conf
   ```
   ii. Change "/usr/bin/pdf2txt.py" to "/usr/local/bin/pdf2txt.py"
   iii. Change "/usr/bin/exiftool" to "/usr/local/bin/exiftool"
   iv. Change "/usr/local/bin/wkhtmltopdf" to "/usr/bin/wkhtmltopdf"
   v. When you are done, press CTRL+X to exit. When prompted to save, type "y", and then "Enter" when asked to confirm the file name.

[STEP 11] INSTALL COLLECTIVEACCESS PAWTUCKET 2

We now download and configure the CollectiveAccess front end, Pawtucket 2.

a. Run the following commands, which will put Pawtucket directly in the root of the webserver so that any hits on your website go to the public portal instead of the staff portal (each list item on its own line):
   i. `git clone https://github.com/collectiveaccess/pawtucket2.git /var/www/html/p_temp`
   ii. `mv /var/www/html/p_temp/* /var/www/html`

b. We now configure Pawtucket to use the same settings as Providence (yes, this is identical to what we did in Step 10).
   i. On the server, run (on one line):
   ```
   sudo cp /var/www/html/setup.php-dist /var/www/html/ setup.php
   ```
   to copy the setup file and then:
   ```
   sudo nano /var/www/html/setup.php
   ```
   to open it for editing in Nano.

ii. In Nano, we need to change, at a very minimum, four of the first five variables: __CA_DB_USER__, __CA_DB_PASSWORD__, __CA_DB_DATABASE__, and __CA_APP_DISPLAY_NAME__. Change these to reflect the information you worte down in Step 0. (Note that __CA_DB_DATABASE__ is the same as what you entered for __CA_DB_USER__.)

iii. You also need to change date_default_timezone_set and __CA_DEFAULT_LOCALE__ to reflect the information you wrote down in Step 0. Details of how to do this are in the setup.php file.

iv. When you are done, press CTRL+X to exit. When prompted to save, type "y", and then "Enter" when asked to confirm the file name.

c. We also have to tell CollectiveAccess where some of the dependancies we installed earlier actually are. Most of the defaults are fine, but we have to fix three of them (again, yes, this is identical to Step 10).

i. Run:
```
sudo nano /var/www/html/app/conf/external_applications.conf
```

ii. Change "/usr/bin/pdf2txt.py" to "/usr/local/bin/pdf2txt.py"

iii. Change "/usr/bin/exiftool" to "/usr/local/bin/exiftool"

iv. Change "/usr/local/bin/wkhtmltopdf" to "/usr/bin/wkhtmltopdf"

v. When you are done, press CTRL+X to exit. When prompted to save, type "y", and then "Enter" when asked to confirm the file name.

[STEP 12] FIX FILE PERMISSIONS

We now need to fix a whole whack of permissions issues. Run each of the following 12 list items as a separate command:

a. `ln -s /var/www/html/staff/media /var/www/html/media`
b. `mkdir /var/www/html/staff/media`
c. `chmod a+w /var/www/html/staff/app/tmp`
d. `chmod a+w /var/www/html/staff/app/log`
e. `chmod a+w /var/www/html/staff`
f. `chmod a+w /var/www/html/staff/media`
g. `chmod a+w /var/www/html/staff/media/collectiveaccess`
h. `chmod a+w /var/www/html/staff/vendor/ezyang/htmlpurifier/library/HTMLPurifier/DefinitionCache/Serializer`
i. `chmod a+w /var/www/html/app/tmp`
j. `chmod a+w /var/www/html/media`
k. `chmod a+w /var/www/html/media/collectiveaccess`
l. `chmod a+w /var/www/html/vendor/ezyang/htmlpurifier/library/HTMLPurifier/DefinitionCache/Serializer`

[STEP 13] CLEAN UP INSTALLATION FILES

All that remains is to clean up from the install process. Run the following two commands:

a. `sudo apt-get autoremove`
b. `sudo apt-get clean`

[STEP 14] FINALIZE AND TEST INSTALLATION

We need to finalize the installation and make sure everything works.

a. Go to http://[COMPIP]/staff/install (where [COMPIP] is the ip address you wrote down in Step 0) in a web browser (preferably on a different computer) to use the web installer to complete the installation of CollectiveAccess.

i. If you get an error complaining about file permissions, re-do Step 12.
ii. If you get an error complaining about not being able to find the database, make sure you did Step 10c correctly.
iii. Choose a profile to install and install it.
iv. At the end of the installation process, write down the default admin username and password the same somewhere safe you wrote the stuff in Step 0. We'll change the default password in a minute.

b. Go to http://[COMPIP]/staff/ and log in with the administrator username and password you wrote down in Step 14a.iv.
   i. Go to the "Manage" menu at the top-right and click on "Access Control". Click the "paper" icon in the row where the default username is located. Change the "Password" and "Confirm Password" boxes to something else (see passwords, above). Write this password down the same safe place as above. Click the "save: button.
   ii. Go to the "Manage" menu and click on "Administration", then "Configuration Check" in the left-hand pane. REFRESH THE PAGE. Everything under "Status" should say either "OK", "Available", or "Not used" EXCEPT BinaryFile and PhantomJS. If it doesn't, first make sure Steps 10c and 11c are done correctly, and make sure that both files say the same thing. Otherwise, try redoing Step 6 and Step 7.

c. Go to http://[COMPIP]. You should get a screen that is mostly blank except for some example text.
   i. If you get an error complaining about file permissions, re-do Step 12.
   ii. If you get an error complaining about not being able to find the database, make sure you did Step 10c correctly.

YAY!!! YOU ARE DONE!!